UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

APLICAÇÃO PARA MONITORAMENTO VEICULAR EM TEMPO REAL

MAICON MACHADO GERARDI DA SILVA

BLUMENAU
2017

MAICON MACHADO GERARDI DA SILVA

APLICAÇÃO PARA MONITORAMENTO VEICULAR EM TEMPO REAL

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Miguel Alexandre Wisintainer - Orientador

BLUMENAU
2017

APLICAÇÃO PARA MONITORAMENTO VEICULAR EM TEMPO REAL

Por

MAICON MACHADO GERARDI DA SILVA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof(a). Nome do(a) Professor(a), Titulação – Orientador, FURB
Membro:	Prof(a). Nome do(a) Professor(a), Titulação – FURB
Membro:	Prof(a). Nome do(a) Professor(a), Titulação – FURB

Blumenau, dia de mês de ano [data da apresentação]

Dedico este trabalho à minha família, amigos e colegas que me apoiaram e ajudaram para a realização do mesmo.

AGRADECIMENTOS

Primeiramente agradeço à Deus por prover conhecimento, saúde, paciência e capacidade para a realização deste trabalho.

À minha família, sem eles nada disso seria possível. Em especial minha mãe Maristela Machado Gerardi e meu pai Eri Junior Miranda Lopes por financiar boa parte do meu sonho que é tornar-me um bacharel em ciência da computação. À minha irmã Milena Kohhausch, que ajudou a corrigir e auxiliou na ortografia deste documento e do pré-projeto.

À minha namorada Jaine Marcírio, por me apoiar, estar ao meu lado e ter paciência comigo nessa etapa da minha vida.

Ao professor Miguel Alexandre Wisintainer pela orientação, entusiasmo e perseverança principalmente nos momentos iniciais, onde dedicou tempo para viabilizar este trabalho. Agradeço por acreditar no projeto, na minha capacidade e por me orientar de forma adequada para a realização e sucesso deste projeto.

Ao Charles Trevizan e Claumir dos Santos da empresa Dynamix Software, por flexibilizar o meu horário de trabalho para desenvolvolver este projeto e o apoio e financeiro com a metade da mensalidade da faculdade.

Ao Nykolas Baumgarten por emprestar o seu modem 4G; agradeço também por me inspirar com seu excelente trabalho de conclusão de curso, bem como Ricardo Starosky.

Aos meus colegas de trabalho e amigos Silvio Gonçalves Neto e Alessandro Jefferson Carvalho por me darem ideias de melhoria do projeto, aconselharem com o desenvolvimento e emprestarem seus veículos para testes da aplicação.

Ao meu amigo Renan Ramos dos Santos Vieira por emprestar o carro para testes e também me ajudar com o seu conhecimento sobre infraestrutura de redes.

Ao meu amigo e colega Plamedi L. Lusembo por ajudar no decorrer do trabalho com o chip da operadora TIM, além de apoios com a monografia e o desenvolvimento.

Ao professor Francisco Adell Péricas por me auxiliar com explicações e orientações sobre infraestrutura de redes 3G e servidores virtuais utilizados na comunicação do servidor deste projeto.

À todas as pessoas que contribuíram direta e indiretamente para a realização deste trabalho.

“ Nada se cria, tudo se transforma ”.

Antonie Lavoisier

RESUMO

O resumo é uma apresentação concisa dos pontos relevantes de um texto. Informa suficientemente ao leitor, para que este possa decidir sobre a conveniência da leitura do texto inteiro. Deve conter OBRIGATORIAMENTE o **OBJETIVO**, **METODOLOGIA**, **RESULTADOS** e **CONCLUSÕES**. O resumo deve conter de 150 a 500 palavras e deve ser composto de uma sequência corrente de frases concisas e não de uma enumeração de tópicos. O resumo deve ser escrito em um único texto corrido (sem parágrafos). Deve-se usar a terceira pessoa do singular e verbo na voz ativa (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2003).

Palavras-chave: Monitoramento veicular. IOT. Internet das coisas. EML327. GPS. OnBoard Diagnostics. OBD. OBDII. OBD2. Raspberry Pi. Python OBD. DTC. Diagnostic Trouble Code.

[Palavras-chave são separadas por ponto, com a primeira letra maiúscula. Caso uma palavra-chave seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]

ABSTRACT

Abstract é o resumo traduzido para o inglês. *Abstract* vem em uma nova folha, logo após o resumo. Escrever com letra normal (sem itálico).

Key-words: Computer science. Monograph. Abstract. Format.

[*Key-words* são separadas por ponto, com a primeira letra maiúscula. Caso uma *key-word* seja composta por mais de uma palavra, somente a primeira deve ser escrita com letra maiúscula, sendo que as demais iniciam com letra minúscula, desde que não sejam nomes próprios.]

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

Nenhuma entrada de índice de ilustrações foi encontrada.

LISTA DE ABREVIATURAS E SIGLAS

3G - 3rd Generation

4G - 4rd Generation

CAN - Controller Area Network

CI - Cicuinto Integrado

CSI - Camera Serial Interface

DDNS - Dynamic Domain Name System

DTC - Diagnostic Trouble Code

ECU - Engine Control Unit

GPIO - General Purpose Input/Output

GPS – Global Position System

IOT - Internet of Things

IP - Internet Protocol

ISO - International Starndardization Organization

JSON - JavaScript Object Notation

LED - Light Emitting Diode

MIL - Malfunction Indicator Lamp

OBD - On-Board Diagnostic

OBD2 - On-Board Diagnostic 2

PHP - Personal Home Pages

PID - Códigos de Parâmetros

RF - Requisito Funcional

RNF - Requisito não Funcional

SAE - Society for Automotive Engineers

SO - Sistema operacional

TCP/IP - Transmission Control Protocol/Internet Protocol

UART - Universal Asynchronous Receiver/Transmitter

USB - Universal Serial Bus

VIN - Vehicle Identification Number

SUMÁRIO

1 INTRODUÇÃO

Nos sete primeiros meses de 2016 foram furtados em Santa Catarina 3.164 carros e pick-ups nacionais e importados com seguro, segundo Odega (2016). Conforme os dados da Superintendência de Seguros Privados (SUSEP), só no primeiro semestre de 2016 as seguradoras registraram 118 mil veículos roubados/furtados no Brasil (ODEGA, 2016).

Odega (2016) cita os veículos com mais índices de roubo, são eles:

a) Chevrolet Celta 1.0 com 6.055 ocorrências de um total de 170.524 veículos segurados;
b) VW Volkswagen Gol 1.0 com 4.514 ocorrências de 253.594 veículos com seguro;
c) Fiat Palio 1.0 com 4.127 ocorrências de um total de 219.654 com seguro.

No ano de 2017, o número de roubos a veículos aumentou 20% em Ribeirão Preto no estado de São Paulo. Segundo G1 Ribeirão e Franca (2017) *"[...] 60 roubos de carros e motocicletas ocorreram durante janeiro de 2017. Ao todo, 50 casos ocorreram durante os primeiros trinta dias de 2016 [...]"*.

Além dos furtos, ainda pode-se analisar a quantidade de veículos com falhas nas estradas. Entre janeiro e outubro de 2014 foram registrados pouco mais de meio milhão de veículos que ficaram parados nos mais de 6 mil quilômetros de rodovias do Programa de Concessões Rodoviárias do Estado de São Paulo por apresentarem problemas de manutenção, por exemplo pneu furado e superaquecimento do motor (SOUZA, 2016). Essa estatística equivale à um pouco mais de 83 carros parados por quilômetro nesse período. Uma pesquisa realizada pelo Instituto Scaringella de Trânsito aponta que a falta de manutenção preventiva no automóvel é relacionada com 30% dos acidentes rodoviários e urbanos no Brasil (CZERWONKA, 2016). Ainda segundo o autor, a manutenção preventiva do veículo não só beneficia a segurança no trânsito, bem como ajuda o condutor a economizar. Cuidar do carro antes que alguma peça apresente defeito custa, em média 30% a menos do que fazer somente a checagem de rotina.

Diante desse cenário, Baumgarten (2016) desenvolveu um dispositivo que possibilitasse o rastreamento veicular através de geolocalização e uma imagem capturada através de uma câmera acoplada neste dispositivo. Paralelamente à Baumgarten (2016), Staroski (2016) desenvolveu um protótipo de software embarcado em uma placa Raspberry Pi para capturar dados da porta On-Board Diagnostic (OBD) de um veículos e disponibilizá-los em uma página *web*.

Com base nesses argumentos, este trabalho consiste em integrar as principais funcionalidades desenvolvidas por Baumgarten (2016) e o protótipo de Starosky (2016) em uma única plataforma, isto é realizado através de um software embarcado em uma placa Raspberry Pi Zero W para capturar a geolocalização de um veículo, imagens deste automóvel e os dados de sua porta OBD. Foi desenvolvida uma aplicação *mobile* para disponibilizar os dados do software embarcado.

1.1 OBJETIVOS

O objetivo deste trabalho é a construção de uma aplicação que abrange o desenvolvimento de um software embarcado em uma placa Raspberry Pi Zero W para coletar a geolocalização, imagens de uma câmera e dados da porta OBD de um automóvel, bem como o desenvolvimento de uma aplicação *mobile* para capturar as informações desse software embarcado.

Os objetivos específicos são:

a) integrar a placa Raspberry Pi Zero W com um módulo Global Positioning System (GPS), um módulo Bluetooth OBD e uma câmera;
b) desenvolver um software embarcado onde será possível verificar a localização atual, as últimas localizações do veículo, capturar imagens e disponibilizar informações da porta OBD;
c) desenvolver uma aplicação *mobile* para consultar as informações disponíveis pelo software embarcado;
d) notificar o usuário sobre falhas no motor retornados pela porta OBD.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo abordar os principais assuntos utilizados para a realização deste trabalho. Eles foram subdivididos em cinco partes, onde a seção 2.1 apresenta a Internet das Coisas (Internet of Things - IOT). A seção 2.2 expõe os assuntos relacionados a OBD:

a) seção 2.2.1: OBD versão 1;
b) seção 2.2.2: OBD versão 2;
c) seção 2.2.3: adaptadores ELM327 utilizados na leitura da porta OBD versão 2;
d) seção 2.2.4: protocolos de comunicação;
e) seção 2.2.5: serviços de diagnósticos OBD versão 2;
f) seção 2.2.6: por fim, é explicado sobre códigos de erro (Diagnostic Trouble Code – DTC).

Após os assuntos explorados sobre OBD, é apresentado na seção 2.3 a plataforma Raspberry Pi. Na seção 2.4 são descritos dois trabalhos correlatos e, por fim, a seção 2.5 são apresentadas duas ferramentas atuais.

2.1 INTERNET DAS COISAS

Em meados de 1991 iniciou-se a discussão sobre a conexão de objetos na rede. Nesse período, a conexão Transmission Control Protocol/Internet Protocol (TCP/IP) e a Internet se tornam acessíveis. Com isso, Bill Joy, o cofundador da Sun Microsystems foi o principal pensador da ideia de conectar várias redes de dispositivos. Por volta de 1999, Kevin Aston do Massachusets Institute Technology (MIT) propôs o termo Internet das Coisas ou Internet of Things (IOT) após dez anos de estudos e realização de projetos, ele escreveu um artigo chamado “A Coisa da Internet das Coisas” para o RFID Journal. De acordo com Aston, a falta de tempo é um fator que contribui para a necessidade de conectar os dispositivos à Internet (APLICAÇÕES DE AUTOMAÇÃO EM IOT - INTERNET OF THINGS, 2016, p. 3).

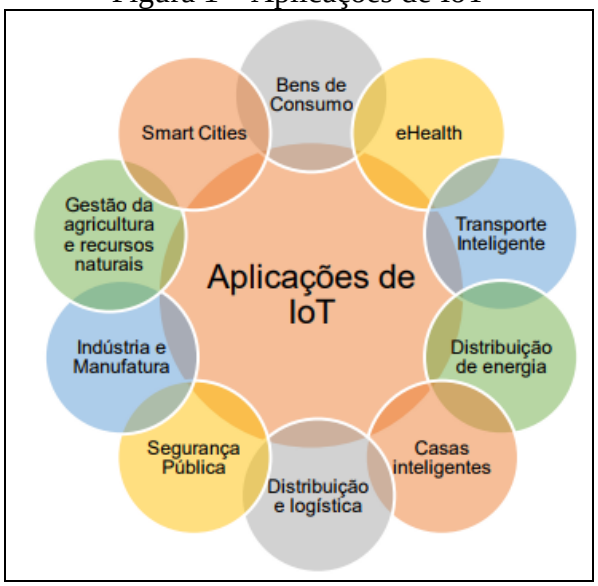
Santos et al. (2016) afirma que a IOT é uma extensão da Internet atual que proporciona para quaisquer tarefas do dia-a-dia, uma maior capacidade computacional e de comunicação por meio da Internet. A conexão com a rede mundial de computadores viabiliza controlar os objetos remotamente e com isso, gerar oportunidades no âmbito acadêmico e industrial. Para AUTOMAÇÃO EM IOT - INTERNET OF THINGS (2016), IOT é um conceito que surgiu com a convergência de tecnologias que envolvem comunicação sem fio, sistemas embarcados e eletromecânicos. Com isso, os principais componentes da rede IOT são:

a) as próprias coisas, por exemplo: aparelhos eletrônicos, sensores, atuadores, computadores e celulares;
b) redes de comunicação.

As aplicações de IOT são diversas, incluem desde tarefas diárias até a sociedade como um todo (MANCINI, 2017). Dias (2016) afirma que o conceito de IOT transforma o mundo em um *smart world*. Na Figura 1 são ilustradas as diversas aplicações da Internet das Coisas e dentro deste cenário, é explicado abaixo cada conceito:

a) bens de consumo: adquiridos pelos consumidores, como *smart* TV e smartphone;
b) eHealth: boa forma, bioeletrônica e cuidados com a saúde;
c) transporte inteligente: notificação das condições de tráfego, controle inteligente de rotas e monitoramento remoto do veículo;
d) distribuição de energia: acompanhamento de instalações elétricas e subestações inteligentes;
e) casas inteligentes: medições remotas de consumo, economia de energia e controle de equipamentos remotamente;
f) distribuição e logística: *smart e-commerce*, rastreabilidade, gerenciamento na distribuição e inventário;
g) segurança pública: monitoramento no transporte de cargas perigosas, monitoramento de construções e utilidades públicas;
h) indústria e manufatura: economia de energia, controle de poluição, segurança na manufatura, ciclo de vida dos produtos, rastreamento e cadeia de abastecimento;
i) gestão da agricultura e dos recursos naturais: segurança e rastreio de produtos agrícolas, monitoramento ambiental para produção, cultivo e gerenciamento no processo de produção;
j) *smart cities*: monitoramento estrutural, como por exemplo vibrações e condições dos materiais em edifícios, pontes e monumentos históricos, iluminação inteligente, monitoramento para prevenção de incêndios, estradas com avisos de desvio conforme condições climáticas, monitoramento em tempo real de espaços de estacionamento e auxílio na coleta de lixo.

Figura 1 – Aplicações de IoT



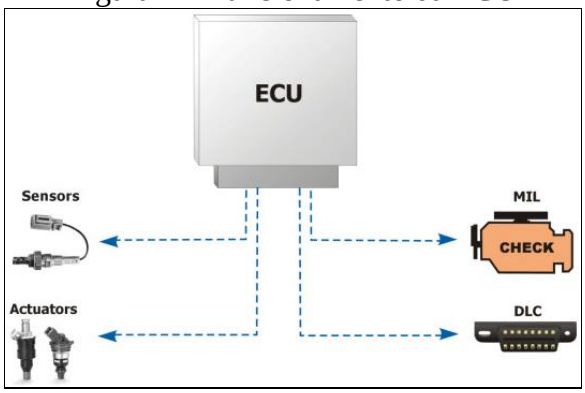
Fonte: Mancini (2017).

A IOT é um grande desafio em seu nível conceitual e tecnológico, pois são várias tecnologias embutidas num único sistema, além de ter muitos segmentos de mercado com aplicação de internet das coisas e cada vez surgem mais aplicações (DIAS, 2016). Segundo Mancini (2016), a Internet das Coisas fará parte da estratégia e gestão, diante disso, serão necessários novos modelos empresarias. Ainda segundo a autora, abrem-se assim novas oportunidades para a atuação de gerentes de projetos para implementar essas soluções e lidar com a transformação organizacional, tecnológica e sócio cultural.

2.2 OBD

Segundo Santos (2016), On-Board Diagnostic (OBD) é um sistema de autodiagnóstico disponível na maioria dos veículos. A ligação ao sistema ocorre por meio de um conector padronizado, que foi sancionado como obrigatório na Europa e nos Estados Unidos a partir de 1996. Um sistema OBD básico consiste em uma Unidade de Controle Eletrônico (Engine Control Unit - ECU) que utiliza a entrada de vários sensores para manipular atuadores, como por exemplo sensores de oxigênio controlando injetores de combustível, esse fim é utilizado para obter um desempenho desejado (OBD SOLUTIONS, 2017). Ainda segundo o autor, a luz conhecida como Malfunction Indicator Lamp (Luz de Mal Funcionamento - MIL) fornece aviso prévio de avaria no veículo ao proprietário. Um veículo moderno suporta centenas de parâmetros que podem ser acessados através do Diagnostic Link Connector (DLC) usando um dispositivo chamado ferramenta de verificação. Na Figura 2 podemos visualizar um diagrama do funcionamento de uma ECU.

Figura 2 – Funcionamento da ECU



Fonte: OBD SOLUTIONS, 2017

Segundo Santos (2016), no Brasil foi sancionado como obrigatório somente a partir de 2010, com o padrão da segunda geração OBD. O autor cita que *"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."*.

2.2.1 OBD1

O sistema OBD1 teve pouco êxito por causa da falta de padronização entre os fabricantes de veículos (como pode ser observado na Figura 3) e pela falta de informações específicas em cada sistema. As dificuldades técnicas de se obter as informações corretas de todos os tipos de veículos inviabilizaram o plano de inspeções veiculares (MCCORD, 2011, p.1).

Segundo Machado e Oliveira (2007), apesar desses fatores de falta de padronização, os sistemas apresentavam os seguintes itens:

a) sensor de oxigênio;
b) sistema de EGR;
c) sistema de combustível;
d) componentes eletrônicos;
e) sistemas eletrônicos;
f) informação de diagnóstico;
g) códigos de erros.

Figura 3 - Exemplo de falta de padronização de conectores OBD-I

Fonte: MCCORD (2011)

2.2.2 OBD2

No início dos anos 90, a Society of Automotive Engineers (SAE) e a International Standardization Organization (ISO) emitiram um conjunto de normas que descrevem o intercâmbio de informações entre ECUs e uma ferramenta de diagnóstico (OBD SOLUTIONS, 2017). Ainda segundo o autor, todos os veículos compatíveis com On-Board Diagnostic 2 (OBD2) foram obrigados a utilizar um conector de diagnóstico padrão (SAE J1962) e comunicar através de um protocolo padrão.

A diferença da atualização para o OBD2 foi a eliminação do grande defeito do OBD1 que consiste na falta de coerência entre os vários sistemas existentes (MACHADO; OLIVEIRA, 2007). Segundo Machado e Oliveira (2007), ouve uma normalização de procedimentos, ou seja, uma estandardização no que diz respeito a métodos de conexão e acima de tudo a nível de protocolos. A lista de itens disponíveis para acesso e controle também foi ampliada:

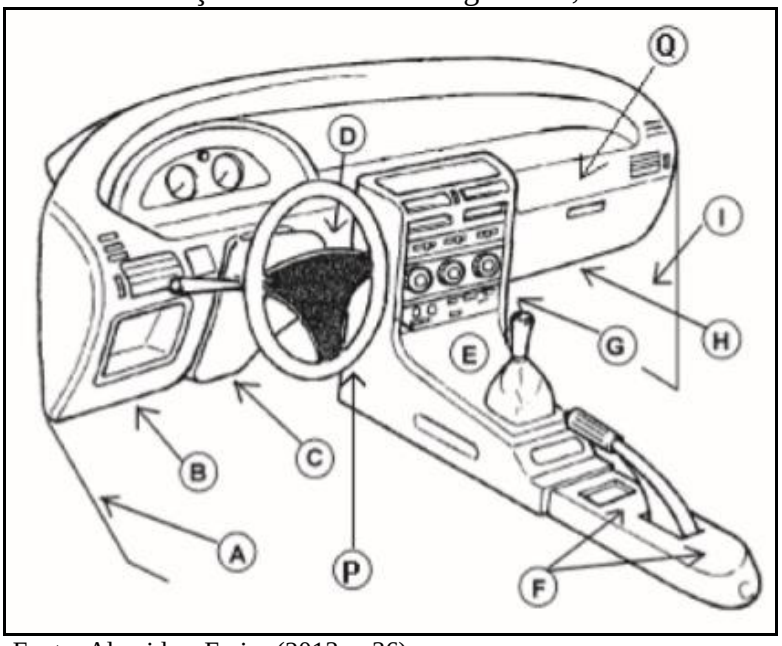
a) sensor de oxigênio;
b) sistema de EGR;
c) sistema de combustível;
d) componentes eletrônicos;
e) sistemas eletrônicos;
f) eficiência de catalisador;

g) aquecimento de catalisador;
h) combustão espontânea;
i) sistema de evaporação;
j) sistema de ar secundário;
k) informações de diagnóstico;
l) códigos de falha;
m) parâmetros do motor;
n) memorização de avarias;
o) estandardização de ligações.

Machado e Oliveira (2007) afirmam que com esses itens a ser constantemente analisados, conseguiu-se cada vez mais diminuir a emissão de gases poluentes. Além disso, o conector OBD2 (ilustrado na Figura 5) está localizado perto do console central do carro na maioria dos casos. Na Figura 4, são apresentadas as possíveis localizações da tomada de diagnóstico. Ela deve atender as seguintes especificações:

a) próximo ao assento do passageiro ou motorista;
b) próximo ao painel de instrumentos;
c) distância de 300mm além da ECU;
d) fácil acesso ao assento do motorista;
e) entre a coluna de direção e a ECU.

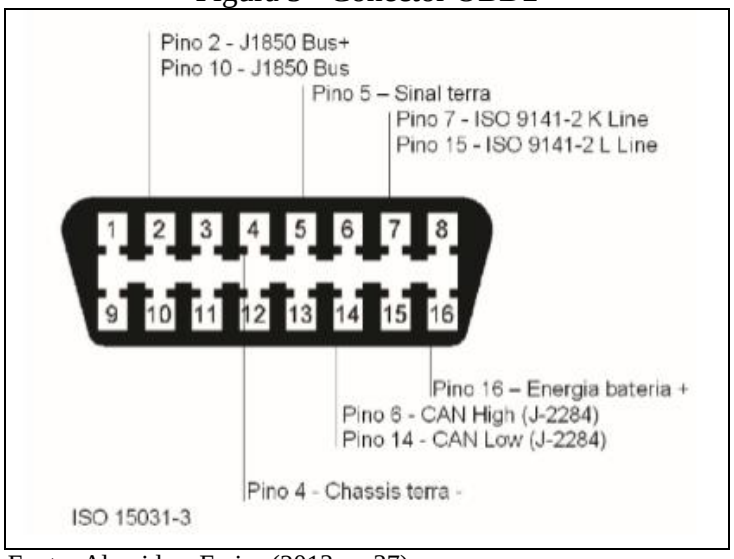
Figura 4 - Localização da tomada de diagnóstico, interior do veículo.



Fonte: Almeida e Farias (2013 p. 36)

Segundo Almeida e Farias (2013), o conector OBD2 possui 16 pinos. Esses, juntamente com seus protocolos de comunicação são ilustrados na Figura 5. Pacheco (2016) afirma que este conector precisa estar facilmente acessível à partir do banco do motorista, caso haja alguma tampa ou conector, o mesmo deve ser de fácil remoção sem o uso de ferramentas.

Figura 5 - Conector OBD2



Fonte: Almeida e Farias (2013, p. 37)

2.2.3 ADAPTADORES ELM237

Os adaptadores ELM327 são dispositivos utilizados para acessar os dados da porta OBD2 de um veículo (OUTILS OBD FACILE, 2017, p. 1). Eles possuem um circuito integrado (CI) chamado ELM327 que são fabricados pela empresa Elm Electronics (ELM ELECTRONICS, 2017, p. 1). Segundo Almeida e Farias (2013), este CI é baseado em outros, como o ELM320, ELM322 e ELM323. Conforme Figura 6, existem diferentes adaptadores, são eles:

a) RS232 (RS ou Série): é utilizado para comunicação serial entre computador e interface OBD2 do veículo. Segundo Outils Obd Facile (2017), este tipo de saída está depreciando-se de computadores modernos;
b) Universal Serial Bus (USB);
c) Bluetooth;
d) Wireless.

Figura 6 - Diferentes adaptadores ELM 327

Fonte: Santos (2016, p. 26).

Segundo Elm Electronics (2017), na maioria dos veículos são utilizados os protocolos Controller Area Network (CAN) (ISO 15765-4), porém, o ELM327 foi projetado para suportar todos os protocolos OBD2 padrão. Por ser multiprotocolo, o autor afirma que é o CI ELM327 é atualizado desde 2005 e possui quatro versões, que estão ilustradas no Quadro 1.

Quadro 1 - Diferentes versões ELM327

	ELM327 v1.3a	ELM327 v1.4b	ELM327 v2.2	ELM327L v2.2
Tensão operacional	4.5V a 5.5V	4.5V a 5.5V	4.2V a 5.5V	2,0 V a 5,5 V
Modo Low Power (Sleep)	Não	sim	sim	sim
Configurações Retained on Wake	-	Não	sim	sim
RS232 Transmit Buffer Bytes	256	256	512	2048
Comandos AT	93	115	128	128
Verificação de frequência da CAN	Não	Não	sim	sim
Suporte de resposta pendente (7F)	Não	Não	sim	sim

Fonte: Elm Electronics (2017, p. 1).

Segundo Almeida e Farias (2013), para tornar o circuito integrado funcional, é necessária uma configuração prévia dependendo do dispositivo utilizado, por exemplo, se utilizar o adaptador Bluetooth, requer pareá-lo com o sistema operacional (SO). Após a

configuração, é possível utilizar comunicação serial através de um computador ou smartphone, no caso do sistema operacional Windows, podem-se utilizar softwares como exemplo: Putty, ZTerm e Tera Term. No sistema operacional Android podem-se utilizar aplicativos, como por exemplo: Serial USB Terminal e UsbTerminal.

Almeida e Farias (2013) afirmam que após estabelecer uma conexão serial com o adaptador, o ELM327 envia uma mensagem informando a versão do circuito integrado. Além disso, ele envia um caractere “>” indicando que está pronto para receber um comando. Ainda, segundo os autores, os comandos podem ser de dois tipos:

a) comandos internos: são destinados para utilização interna do circuito ELM327 e começam com os caracteres “AT”;
b) comandos para barramento: utilizados no barramento OBD2 e contém apenas códigos ASCII para dígitos hexadecimais (de 0 a F).

As mensagem que não são compreendidas pelo ELM327 respondem com um ponto de interrogação “?”. Porém, segundo Almeida e Farias (2013), isso não significa que a mensagem foi desentendida, pode ser que a mensagem não é suportada pelo circuito. Além disso, caracteres maiúsculos e minúsculos não são distinguidos, bem como são ignorados caracteres de espaços e todos os caracteres de controle (tab, enter, etc).

2.2.4 PROTOCOLOS DE COMUNICAÇÃO

Cada protocolo de comunicação possui características específicas (tempo de resposta, banda, redundância, detecção de erros, arquitetura de redes e software de programação), sendo normal encontrar mais de um barramento implantado em um veículo (ALMEIDA; FARIAS, 2013, p. 51). Com isso, os autores afirmam que, em 1994 a Sociedade dos Engenheiros Automotivos dos Estados Unidos (Society for Automotive Engineers - SAE) definiu uma classificação para os protocolos de comunicação automotiva. Esta classificação é baseada na velocidade de transmissão de dados e a função que são distribuídas pela rede.

Segundo Bastos (2012), o conector OBD2 disponibiliza cinco protocolos automotivos, são eles:

a) SAE J1850 PWM: possui a taxa de transferência de 41.6 Kbps, utiliza 2 pinos do conector OBD2 e o tamanho da sua mensagem é de 12 bytes (padrão utilizado pela Ford Motors);
b) SAE J1850 VPW: possui um único fio de conexão, por isso é considerado de baixo custo. Taxa de transferência de 10.4 Kbps e tamanho para mensagens de 12 bytes (utilizado pela General Motors (GM) com o nome de GM Class 2);

c) ISO 1941-2: a comunicação é assíncrona Universal Asynchronous Receiver/Transmitter (UART), taxa de transmissão 10.4 Kbps e mensagem de 5 a 11 bytes (protocolo utilizado pela Crysler, fabricantes europeus e asiáticos);
d) ISO 14230: popularmente chamado de Keyword 2000 (KW2000), é apenas um link de diagnóstico e não pode ser utilizado para transmitir mensagens. Foi bastante utilizado antes mesmo da OBD2 por fabricantes como Bosh, Opel e outros fabricantes europeus, que pressionavam órgãos regulamentadores para viabilizar a utilização deste link. Sua liberação ocorreu na década de 90, e só foi aprovada por ter o protocolo e os requerimentos de *hardware* quase idênticos ao ISO 9141-2. Este protocolo possui a velocidade entre 1.2 e 10.4 Kbps e suas mensagens podem conter até 255 bytes;
e) ISO 15765: mais conhecido como Controller Area Network (CAN), foi desenvolvido pela Bosh, sua comunicação é serial e sua velocidade é de 500 Kbps. O padrão ISO 15765-4 determina os requisitos para a aplicação OBD.

2.2.5 SERVIÇOS DE DIAGNÓSTICO

Os serviços de diagnóstico de um sistema OBD2 são organizados por modos de operação e códigos de parâmetros (PIDs) (SANTOS, 2016, p.19). O padrão OBD2 regulamenta um conjunto de PIDs e modos de operação, sendo que alguns deles são de implementação obrigatória pelo fabricante, especialmente os que são relacionados à emissão de gases poluentes. Alguns PIDs possuem implementação opcional que o fabricante opta ou não por disponibilizar de acordo com a sua necessidade.

Segundo Almeida e Farias (2013), existem dez serviços disponíveis, de modo que cada veículo ou ECU deverá implementar seus serviços de acordo com a sua legislação. Santos (2016) afirma que cada serviço é um valor hexadecimal de dois dígitos, que deve estar entre 0x01 e 0x0A. Cada operação pode ter até 256 PIDs, que também são representados por um número hexadecimal de dois dígitos.

Existe também o serviço 0x00, que é reservado à aderência do veículo ao padrão OBD, ou seja, a resposta retornada por ele é um vetor de bits em que cada valor binário indica se o PID correspondente é suportado ou não pela ECU (SANTOS, 2016, p. 19). Os serviços 0x01 à 0x0A são descritos abaixo:

a) serviço 0x01: Bastos (2012) relata que este serviço permite o acesso às informações de dados relacionados ao *powertrain*, dentre eles sinais de entrada/saída analógicas e digitais, além de informações do sistema. Alguns

exemplos de PIDs suportados por este serviço são:

- PID 05: temperatura do fluido de arrefecimento (°C);
- PID 00: velocidade do veículo (km/h);
- PID 11: posição da borboleta;

b) serviço 0x02: exibe dados do *Freeze Frame*, ou seja, no momento que aconteceu uma falha estes dados são "congelados" e armazenados neste serviço. O PID 02 deste serviço indica o código de erro (Diagnostic Trouble Code - DTC) que causou o *Freeze Frame*;

c) serviço 0x03: lista os DTCs confirmados que impactam emissões, este serviço permite à ferramenta de *scanner* listar todos os DTCs de cinco dígitos que estão presentes no momento ou que já iluminaram a lâmpada de mal funcionamento (MIL) recentemente;

d) serviço 0x04: segundo Bastos (2012), este serviço permite que a ferramenta de *scanner* possa comandar a ECU para limpar as informações de diagnóstico, incluindo os DTCs armazenados, dados de *Freeze Frame* e distância que ocorreu a falha. Este serviço também reinicia todos os contadores de diagnóstico e retira as ações de degradação do sistema. Ele só deverá funcionar com a ignição ligada e motor desligado, sendo aplicado a todos os controladores ECUs;

e) serviço 0x05: requisita resultados de teste de sensor de oxigênio, este serviço não é suportado pela CAN e sua funcionalidade está implementada no serviço 0x06;

f) serviço 0x06: Bastos (2012) afirma que este serviço informa testes de monitoramento para componentes e sistemas específicos que são constantemente monitorados (ex: *misfire* ou falha de combustão). O resultado é exibir o último valor vigente do teste e também os limites máximo e mínimo;

g) serviço 0x07: corresponde aos códigos DTCs pendentes de acender a lâmpada de mal funcionamento MIL no ciclo de condução vigente e anterior. Segundo Bastos (2012), este serviço é independente do serviço 0x03, porém, com o mesmo formato. O seu principal objetivo é auxiliar o técnico a verificar problemas em componentes durante o reparo dos mesmos após suas substituições, bem como na limpeza das informações de diagnóstico com o serviço 0x04.

h) serviço 0x08: Segundo Santos (2016), este serviço é bidirecional, ou seja, com ele é possível ler, alterar parâmetros de operação e também é possível testar diferentes hipóteses para identificar a causa do problema. Este serviço deve ser utilizado com cuidado, pois pode causar avarias ao veículo. Santos (2016) também afirma que,

através desse módulo, é possível alterar a potência do motor em alguns cavalos de força apenas por meio de reprogramação da ECU;

i) serviço 0x09: permite obter informações sobre o software instalado na ECU. É mais utilizado para consultar o Vehicle Identification Number (VIN), que é o código identificador único para carros e caminhões. Também serve para identificar a versão do software que está instalado e este serviço permite obter relatórios sobre sistema catalizador, sensor de oxigênio, vazamento no sistema de evaporação, controle de pressão, contadores de eventos como o número de ignição e testes de autodiagnóstico (SANTOS, 2016, p. 21);

j) serviço 0x0A: por fim, este módulo ou serviço refere-se à lista de todos os erros gerados cada vez que a luz MIL foi acesa ou uma indicação de falha foi ativada. Diferente do serviço 0x03 e 0x07, ele não pode ser apagado nem mesmo com a desconexão da alimentação da ECU. Isto se dá pelo fato de manter um histórico de anomalias do veículo e evita que o condutor ou técnico apague estes erros.

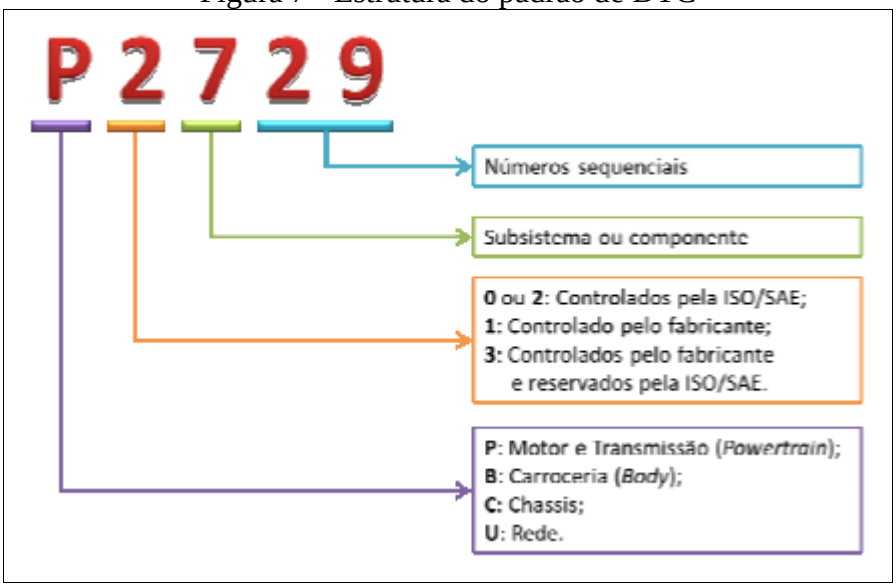
2.2.6 DIAGNOSTIC TROUBLE CODE (DTC)

Diagnostic Trouble Code (DTC) são os códigos de erros retornados e armazenados pelo sistema OBD2 quando ocorre algum problema no ECU. Segundo Almeida e Farias (2013), para identificar falhas no sistema OBD2, são executados testes no circuito de todos os sensores e monitores procurando por possíveis curtos circuitos, circuitos abertos, plausibilidade do sinal e também é processada a ECU (codificação, memória e etc). Se uma falha for detectada por algum dos testes, o código de defeito (DTC) é armazenado e caso essa for relacionada com emissão de poluentes, a luz MIL será acionada. Almeida e Farias (2013) relatam que o DTC não é necessariamente a causa do defeito e sim, isolar a falha a uma área funcional específica do veículo.

Segundo Almeida e Farias (2013), para que o sistema de diagnose detecte a necessidade de gerar um código de falha, a informação deve ser pertinente por até 10 ciclos de motor. A fim de interpretar que esta falha foi sanada, é necessário que este código não esteja presente por no mínimo 40 ciclos de motor. Com intuito de apagar a MIL são necessários apenas 3 ciclos. Cada ciclo de motor é equivalente a ligá-lo, aquecê-lo, percorrer uma distância pré-determinada e verificar todos os sensores.

Os códigos de falhas DTC são formados por 2 bytes ou 16 bits e seguem um padrão explicados na Figura 7. Este padrão segue uma nomenclatura conforme norma SAE J2012 (BASTOS, 2012, p. 30).

Figura 7 - Estrutura do padrão de DTC



Fonte: Bastos (2012, p. 30)

Segundo Almeida e Farias (2013), o primeiro caractere (uma letra) refere-se ao sistema que a falha pertence (conforme Quadro 2).

Quadro 2 - Primeiro caractere do código de falha.

Letra	Valor binário	Significado
P	00	Motor *(Powertrain)*
C	01	Chassi *(Chassis)*
B	10	Corpo *(Body)*
U	11	Rede *(Network)*

Fonte: Almeida e Farias (2013, p. 38).

O primeiro dígito após o caractere (número de 0 a 3) é a entidade responsável pela criação do código. Através dele, é possível verificar se pertence à algum fabricante específico ou comum para todos os fabricantes (ALMEIDA; FARIAS, 2013). Os possíveis valores estão ilustrados no Quadro 3.

Quadro 3 - Primeiro valor numérico do código de falhas (segundo caractere)

VALOR	Entidade Responsável
0	ISO/SAE
1	Fabricante.
2	ISO/SAE
3	ISO/SAE, reservado e Fabricante

Fonte: Almeida e Farias (2013, p. 37).

O terceiro caractere refere-se a um subgrupo de funções do veículo. No Quadro 4, é apresentado os valores do terceiro dígito. O quarto e quinto dígito são falhas específicas do subgrupo identificado.

Quadro 4 - Terceiro dígito do código de falhas

Valor	Descrição
0	Sistema eletrônico completo.
1	Controle Ar / Combustível.
2	Controle Ar / Combustível; Circuito de injeção.
3	Sistema de ignição.
4	Controle Auxiliar de Emissões
5	Controle de velocidade do veículo e de rotação em marcha lenta.
6	Circuitos de entrada e saída da central eletrônica.
7	Transmissão.
8	Transmissão.

Fonte: Almeida e Farias (2013, p. 37).

Se a comunicação for feita através de protocolo serial, o valor do DTC é apresentado em hexadecimal e precisa ser decodificado para o código correspondente. O Quadro 5 ilustra um exemplo para a decodificação dó código C0123 para hexadecimal e binário.

Quadro 5 - Exemplo de decodificação DTC

Unidade	Valores															
Código DTC	C		0		1				2				3			
Binário	0	1	0	0	0	0	0	1	0	0	1	0	0	0	1	1
Hexadecimal	4				1				2				3			

Fonte: elaborado pelo autor.

2.3 RASPBERRY PI

Raspberry Pi é um computador do tamanho de um cartão de crédito que se conecta a um monitor ou uma TV, usa um teclado e mouse padrão e foi desenvolvido no Reino Unido pela Fundação Raspberry Pi. Todo o hardware é integrado à uma única placa, a Figura 8 demonstra a comparação de tamanho entre dois modelos de placas Raspberry Pi. O principal objetivo é promover o ensino da ciência da computação básica em escolas (RASPBERRY PI FOUNDATION, 2017, p. 1). Ainda segundo o autor, existem vários modelos disponíveis com as mais diversas configurações, o Quadro 6 cita suas principais diferenças e especificações entre elas.

Quadro 6 - Comparativo entre os modelos Raspberry Pi

Características / Modelos	Velocidade	RAM	Portas USB	Ethernet	Wireless e Bluetooth	Preço
Raspberry Pi Model A+	700Mhz	512MB	1	Não	Não	$20
Raspberry Pi Model B+	700Mhz	512MB	4	Sim	Não	$25
Raspberry Pi 2 Model B	900Mhz	1GB	4	Sim	Não	$35
Raspberry Pi 3 Model B	1200Mhz	1GB	4	Sim	Sim	$35
Raspberry Pi Zero	1000Mhz	512MB	1	Não	Não	$5
Raspberry Pi Zero W	1000Mhz	512MB	1	Não	Sim	$10

Fonte: adaptado de RASPBERRY PI FONDATION (2017, p. 1).

O modelo A+ é a variante de baixo custo, possuindo 512MB de RAM e 700Mhz de processamento sem nenhuma conexão *wireless*, Bluetooth e Ethernet. Já a placa modelo B+ é a adaptação dela e por um preço razoável já vem com porta Ethernet (RASPBERRY PI FONDATION, 2017, p. 1).

Segundo Raspberry Pi Foundation (2017), em fevereiro de 2015 o modelo B+ foi substituído pelo Pi 2 Model B, que contém um *quad-core* ARM Cortex-A7 CPU de 900Mhz de processamento e 1GB de memória RAM. E o modelo Pi 3 Model B foi lançado em fevereiro de 2016, utilizando um processador ARM Cortex-A53 de 1.2GHz, integrando *wireless*, Ethernet e Bluetooth 4.1. Na Figura 8, esta placa é a maior tamanho, este modelo é o mais utilizado e recomendado para uso em escolas devido à sua flexibilidade para o aluno (RASPBERRY PI FONDATION, 2017, p. 1).

Figura 8 - Comparação entre Pi 3 e Zero W

Fonte: Null Byte (2017, p. 1)

Conforme a Figura 8, o Raspberry Pi Zero e Zero W são a metade do tamanho de um modelo Pi 3, com um único núcleo de 1GHz, 512MB de RAM, porta mini-HDMI e USB. Além disso ele integrou *wireless* 802.11n LAN e Bluetooth na placa com a versão Pi Zero W (RASPBERRY PI FONDATION, 2017, p. 1).

2.4 TRABALHOS CORRELATOS

São apresentados dois trabalhos correlatos que possuem características semelhantes à este desenvolvido. Primeiramente, seção 2.4.1 apresenta uma solução para gestão de frotas denominado Gestão de uma Frota de Veículos Utilizando Sistemas Embarcados desenvolvido por Pacheco (2016). Por fim, a seção 2.4.2 trata de uma aplicação chamada Sistema de Localização de Veículos Para Smartphone Android feita por Pina (2015).

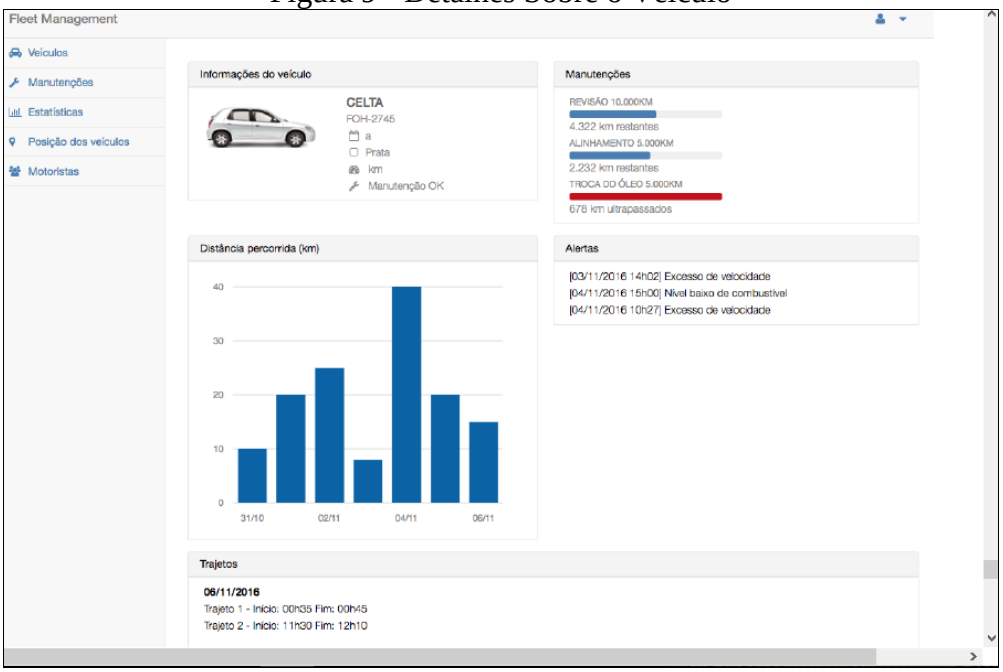
2.4.1 GESTÃO DE FROTA DE VEÍCULOS

Segundo Pacheco (2016), o objetivo do trabalho é o desenvolvimento de um sistema embarcado para o monitoramento de veículos integrado a uma plataforma *web* para o gerenciamento de frota de veículos. Foi desenvolvido um sistema embarcado construído para uma placa Raspberry Pi, um adaptador OBD e um módulo GPS. A placa Raspberry Pi é um computador que utiliza o sistema operacional Linux que ocupa o papel central do sistema. Através dela é possível coletar os dados do adaptador OBD e do módulo GPS. Além disso, ela transmite os dados ao servidor que hospeda a uma plataforma *web*.

Pacheco (2016) desenvolveu um sistema embarcado tal que, dentro do contexto de gestão de frotas, pudesse coletar informações através da porta OBD, como por exemplo:

velocidade do veículo, rotação do motor, carga do motor, distância percorrida e geolocalização. Essas informações foram coletadas para analisar como o veículo é conduzido (conforme Figura 9). O monitoramento pode indicar se os limites de velocidade são respeitados e se há acelerações e frenagens bruscas. Valores de rotação muito elevados também seriam detectados. A carga do motor está associada ao consumo de combustível. A distância percorrida para o agendamento sem a necessidade de consultar o odômetro, a Figura 9 mostra na parte de manutenções essas informações prévias de quilometragem para manutenção, troca de óleo e alinhamento do veículo. Por fim, utiliza a geolocalização para registrar os trajetos realizados.

Figura 9 - Detalhes Sobre o Veículo



Fonte: Pacheco (2016, p. 70)

Estes dados são coletados e armazenados no cartão de memória da placa Raspberry Pi. Para isso, foi considerado um intervalo de amostragem desta coleta. Após a coleta dos dados, eles são enviados à um servidor que hospeda a plataforma de gestão de frotas. O envio de dados é feito através de uma conexão Wi-Fi no momento que o veículo retorna à garagem.

Pacheco (2016) utilizou a linguagem de programação Python orientado à objetos. Ele cita que Python é uma linguagem de programação que tem uma quantidade considerável de bibliotecas disponíveis e podem ser desenvolvidas aplicações, tais como: aplicações *web*, cálculo científico e numérico, gráficos, *Machine Learning*, *Data Science*, visualização de dados, interfaces gráficas, entre outras. Foram utilizadas neste trabalho bibliotecas que permitissem a criação de *threads*, o acesso ao banco de dados SQLite, o uso de expressões

regulares e o *back-end* de um servidor. Para o banco de dados, foi utilizado o SQLite por ser possível executar comandos SQL, salvar e fazer alterações em registros.

Para comandar o sistema embarcado, foi escolhida a placa Raspberry Pi 3 Model B, a qual funciona com o sistema operacional Linux. Possui Bluetooth e uma porta serial, podendo assim, comunicar-se com o adaptador OBD e com o módulo GPS.

O adaptador OBD é o dispositivo utilizado para a comunicação com o computador de bordo do carro e é instalado diretamente na porta OBD do veículo. A troca de dados entre a Raspberry Pi é realizada através de Bluetooth. O adaptador utilizado é baseado no circuito integrado ELM327. O ELM327 é um interpretador multiprotocolo projetado para funcionar com todos os protocolos automotivos de veículos previstos na especificação OBD2. Este dispositivo custa aproximadamente trinta reais. Foi utilizado também o módulo GPS GY-NEO6VM2 para determinar a posição dos veículos utilizando a porta serial para fazer a comunicação dele com a placa Raspberry Pi.

Pacheco (2016) cita que a solução mostrou-se eficiente na coleta de dados, pois irá trabalhar com informações atualizadas lidas direto dos sensores do veículo, bem como elimina a necessidade da leitura constante do odômetro do veículo. O esforço das empresas que possuem um número elevado de veículos, diminui pelo fato do sistema executar a leitura dos sensores e também melhora a capacidade de detecção para manutenção da frota (PACHECO, 2016).

Pacheco (2016) sugere que sejam implementados soluções para veículos pesados, pois a solução desenvolvida por ele abrange somente veículos de passeio, em razão de que a interface do sistema OBD é diferente da implementada por ele. O autor destaca a necessidade da portabilidade da plataforma *web* para um sistema *mobile*. Sugere-se também, uma redução de custo da placa, utilizando ao invés da placa Raspberry Pi 3 (que custa 35 dólares aproximadamente), uma placa Raspberry Pi Zero (que custa 5 dólares) que atende a especificação e ainda consome menos energia.

2.4.2 LOCALIZAÇÃO DE VEÍCULOS PARA ANDROID

Pina (2015) cita que o trabalho consiste no desenvolvimento de um sistema de localização de veículos para *smartphone* Android. Para isso, foram desenvolvidas duas aplicações: uma aplicação de localização Android e uma aplicação *web* para monitoramento.

A aplicação de localização permite capturar dados de localização de GPS e estabelecer uma rede *piconet* Bluetooth, admitindo assim uma comunicação com uma unidade de controle do carro através de um adaptador OBD2 e com até sete sensores/dispositivos Bluetooth que

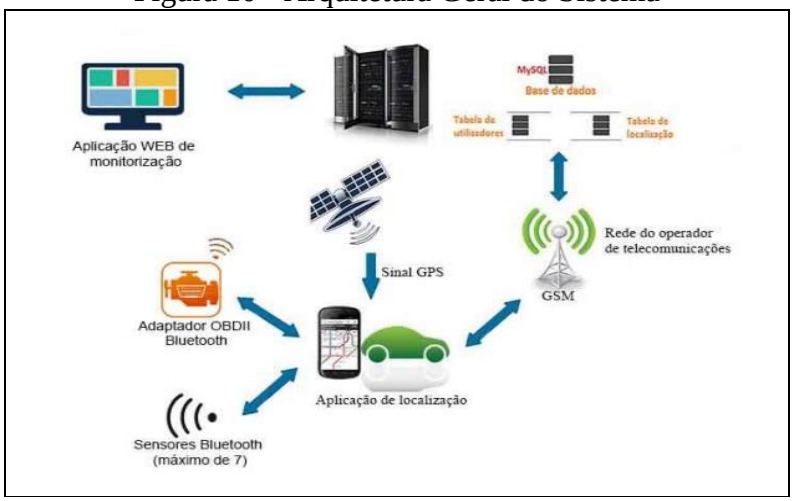
podem ser instalados no veículo. Os dados adquiridos pela aplicação Android são enviados periodicamente para um servidor *web*.

A aplicação *web* desenvolvida por Pina (2015), permite ao gestor da frota, efetuar o monitoramento dos veículos em circulação registrados no sistema. É possível visualizar a posição geográfica dos veículos em um mapa, bem como, os dados do mesmo e sensores/dispositivos Bluetooth para cada localização enviada pela aplicação Android.

A Figura 10 exemplifica a arquitetura geral do sistema desenvolvido por Pina (2015), ele idealiza o sistema em quatro principais componentes:

a) aplicação Android de localização;
b) aplicação *web* de monitoramento;
c) adaptadores OBD2/Bluetooth;
d) dispositivos/sensores Bluetooth.

Figura 10 - Arquitetura Geral do Sistema

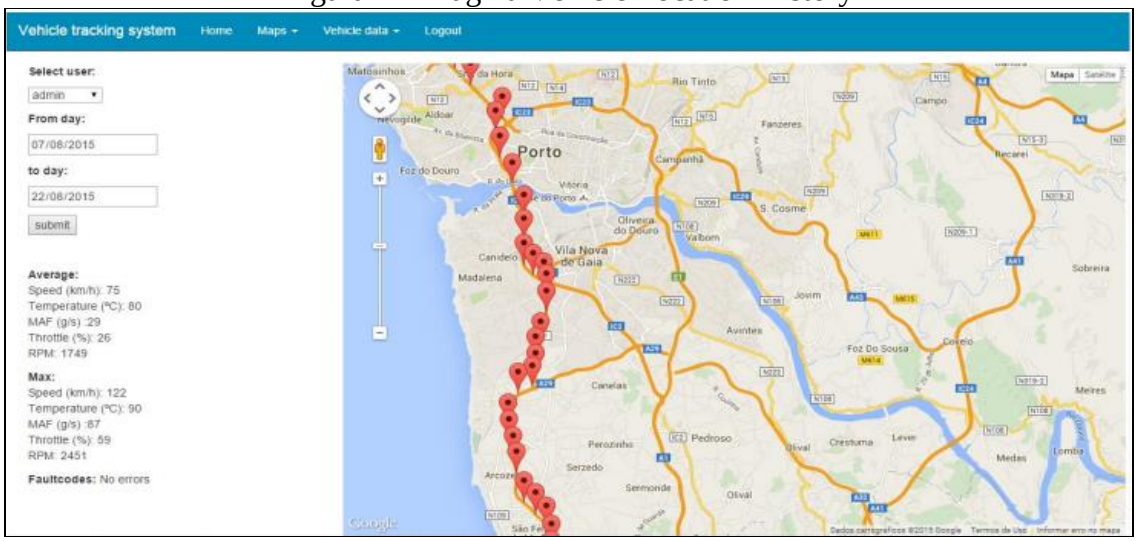


Fonte: Pina (2015, p. 79)

A aplicação Android desenvolvida é responsável por gerir a comunicação com o adaptador OBD2/Bluetooth, possíveis sensores Bluetooth adicionais, posição geográfica e transmitir esses dados para o servidor *web*. Para essa aplicação, utilizou-se JAVA com Android e para a persistência de dados no aparelho, utilizou-se o banco de dados SQLite.

Já a aplicação *web* foi desenvolvida para verificar onde está cada veículo geograficamente e as informações capturadas das portas OBD2 e sensores Bluetooth. Na Figura 11, pode-se observar a funcionalidade de visualizar as informações geográficas em um mapa. Essa aplicação *web* foi desenvolvida utilizando a tecnologia Personal Home Pages (PHP) juntamente com o banco de dados MySQL para persistência.

Figura 11 - Página Vehicle Location History



Fonte: Pina (2015, p. 115)

Pina (2015) conclui que o sistema desenvolvido é completamente funcional e teve alguma complexidade com o desenvolvimento na plataforma Android, mas que possui uma documentação completa. O autor cita também, que teve facilidade em integrar os dispositivos devido à simplicidade de instalação e portabilidade. A possibilidade de interação com o adaptador OBD2 Bluetooth e com até sete dispositivos simultaneamente torna o sistema extremamente versátil, que pode ser utilizada à inúmeras aplicações, até mesmo fora do contexto de veículos como foi apresentado.

Pina (2015) relata que uma das desvantagens foi a fragilidade do adaptador OBD2/Bluetooth, sugere então, que seja utilizado outro com maior qualidade. O sistema pode aumentar a quantidade de funcionalidades disponíveis, tais como: a leitura de mais parâmetros da porta OBD2, melhoras no layout da aplicação e também um sistema de mensagens entre gestor e utilizador da aplicação.

2.5 FERRAMENTAS ATUAIS

Nesta seção serão apresentados dois trabalhos, ambos desenvolvidos pelo curso de Ciência da Computação na Universidade Regional de Blumenau. A seção 2.5.1 trata de um trabalho denominado Findcar desenvolvido por Baumgarten (2016). Por fim, a seção 2.5.2 apresenta o trabalho OBD-JRP, desenvolvido por Starosky (2016).

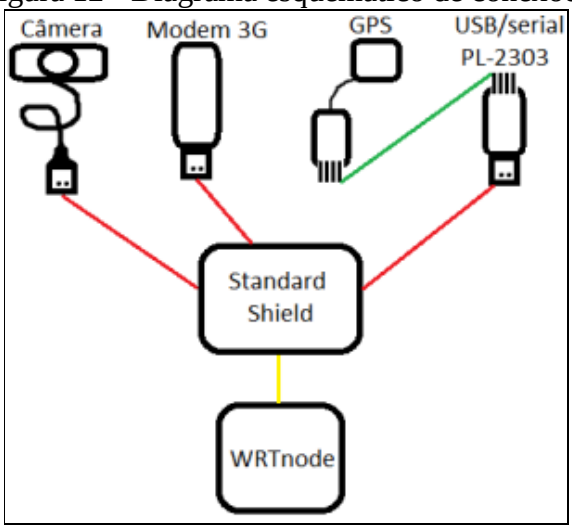
2.5.1 FINDCAR

Baumgarten (2016) desenvolveu um dispositivo que possibilitasse o rastreamento veicular através de geolocalização e uma imagem capturada dor meio de uma câmera acoplada neste dispositivo. Os objetivos cumpridos no trabalho foram:

a) realizou-se a integração do OpenWRT com: um modem 3rd Generation (3G), um módulo Global Positioning System (GPS) e uma câmera;
b) desenvolveu uma plataforma *web* para verificar a localização atual, ultimas localizações do veículo, capturar imagens e configurar o envio de notificações por e-mail;
c) tornou o rastreador o próprio servidor onde a aplicação é executada.

Para este rastreador veicular, Baumgarten (2016) utilizou uma placa WRTnode de modelo MT7620 com OpenWRT, que é uma distribuição customizável do Linux para sistemas embarcados. Utilizou também, um módulo de GPS Ublox GY-NEO6MV2 para capturar a geolocalização e uma webcam Logitech C270 para obter as imagens do veículo. Todas as informações são disponibilizadas através de um modem 3G/4G Huawei E3272. O esquema de conexões pode ser observado na Figura 12.

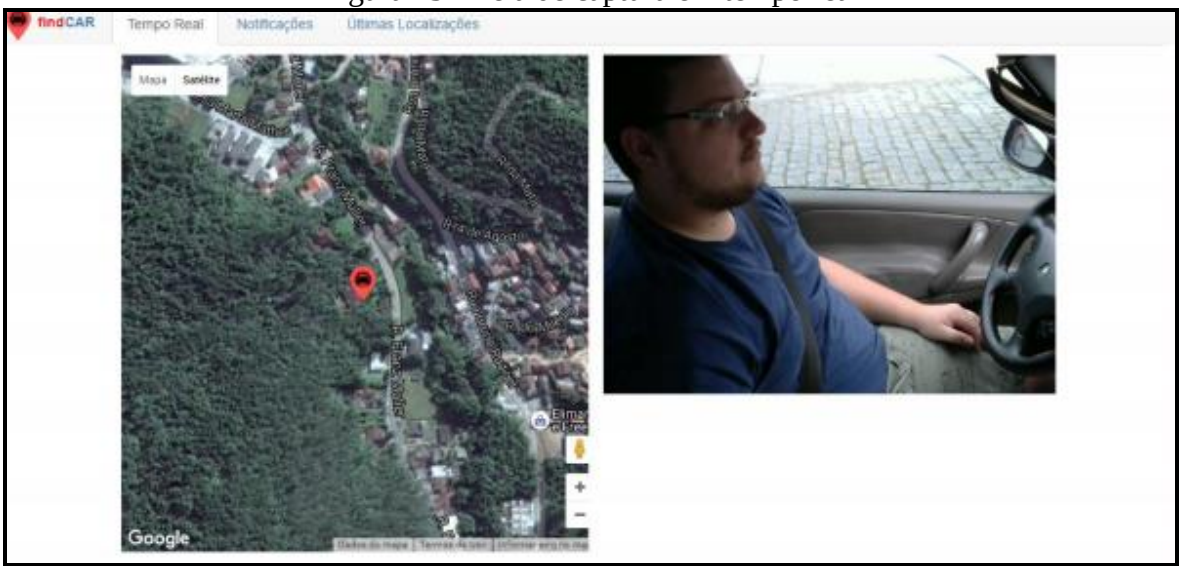
Figura 12 - Diagrama esquemático de conexões



Fonte: Baumgarten (2016, p. 26)

Para desenvolver a aplicação *web*, Baumgarten (2016) utilizou para interface gráfica HTML5, CSS3 e Bootstrap3. Com relação à comunicação com o servidor, foi utilizado PHP e a persistência de dados foi feita através do banco de dados MySQL. Utilizou também Google Maps Javascript Api para mostrar as coordenadas capturadas pelo GPS no mapa, conforme Figura 13.

Figura 13 - Tela de captura em tempo real



Fonte: Baumgarten (2016, p. 39)

Baumgarten (2016) ressalta que o objetivo do trabalho foi atingido adequadamente. O uso da linguagem PHP supriu as necessidades do sistema. Ele enfatiza o uso do banco de dados MySQL que foi facilmente integrado e manipulado com o OpenWRT.

2.5.2 OBD-JRP

Staroski (2016) desenvolveu um protótipo de software embarcado em uma placa Raspberry Pi. Esta placa foi utilizada para capturar dados da porta OBD de um veículo e disponibilizá-los em uma página *web*. Ele enumerou e concluiu alguns objetivos específicos que foram atendidos, são eles:

a) desenvolver o firmware, que irá monitorar a porta OBD2 do carro, coletar dados e os enviar para um servidor;
b) desenvolver o software servidor, que irá receber os dados coletados pelo firmware e armazenar os mesmos;
c) desenvolver uma página *web* para consultar o histórico dos dados.

Para a elaboração do trabalho, Starosky (2016) utilizou o ambiente de desenvolvimento Java com a biblioteca BlueCove para realizar a comunicação com a interface ELM327 Bluetooth. No desenvolvimento do servidor, foi utilizado a biblioteca Google Charts para criar gráficos com a linguagem Javascript. Foi utilizado a placa Raspberry Pi 3 Model B com o sistema operacional Raspian GNU/Linux 8 que é disponibilizada com a versão 1.8 do Java. Os dispositivos citados e utilizados podem ser visualizados na Figura 14. Além desses dispositivos, para concluir a comunicação com o servidor, foi utilizado um modem 3G/4G da marca Huawei.

Figura 14 - Instalação no Volkswagen SpaceFox 2009

Fonte: Starosky (2016, p. 73)

Segundo o autor, a aplicação foi testada em três veículos, sendo eles: GM Corsa Sedan 2003, Volkswagen Gol 2010 e um Volkswagen SpaceFox 2009. Todos possuíam o conector OBD2, entretanto o Corsa Sedan 2003 não implementava nenhum protocolo OBD2 apesar de possuir a porta. O protótipo atendeu os objetivos propostos e o Raspberry Pi atendeu as exigências computacionais desenvolvidas.

3 DESENVOLVIMENTO DA APLICAÇÃO

--MAKE

3.1 ESPECIFICAÇÃO

Nesta seção é exposta a especificação da aplicação através de requisitos, diagramas e fluxogramas. Inicialmente são apresentados os Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF) e sua rastreabilidade com cada caso de uso. Em seguida é exposto o diagrama de casos de uso, também é apresentado diagramas sobre a arquitetura de hardware, bem como o fluxograma de inicialização do sistema embarcado. Todos os diagramas Unified Modeling Language (UML) foram construídos utilizando a ferramenta online Draw.io. O diagrama para esquema de componentes eletrônicos utilizou a ferramenta online denominada Scheme-It para o seu desenvolvimento.

3.1.1 REQUISITOS

O Quadro 7 apresenta os principais requisitos funcionais (RF) da aplicação e sua respectiva rastreabilidade com cada caso de uso correspondente.

Quadro 7 - Requisitos funcionais da aplicação e rastreabilidade

Requisitos funcionais	Caso de uso
RF01: permitir que o servidor embarcado seja iniciado automaticamente ao ligar a placa Raspberry Pi W Zero	UC01
RF02: permitir ao usuário fazer a configuração de e-mail, telefone e notificações	UC02
RF03: permitir que o usuário capture imagens em tempo real do interior do veículo	UC03
RF04: disponibilizar informações da geolocalização do veículo	UC04
RF05: disponibilizar dados de sensores do automóvel através da porta OBD2	UC05
RF06: disponibilizar os códigos de erro (DTCs)	UC06
RF07: permitir limpar os códigos de erro (DTCs)	UC06
RF08: permitir consultar o status da luz de mal funcionamento (MIL)	UC06
RF09: permitir receber notificação via e-mail e SMS caso ocorram falhas (DTCs) no veículo	UC08
RF10: permitir reiniciar o servidor embarcado através de um botão acoplado a placa Raspberry PI Zero W	UC07
RF11: permitir desligar e ligar o sistema operacional embarcado através de um botão acoplado a placa Raspberry PI Zero W	UC01
RF12: permitir visualizar estágios de execução do servidor embarcado através de Diodos Emissores de Luz (Light Emitting Diode – LED)	UC09

Fonte: elaborado pelo autor.

Após a exposição dos requisitos funcionais, são apresentados abaixo os principais requisitos não funcionais (RFN) previstos para a aplicação desenvolvida. A aplicação deve:

a) utilizar a placa Raspberry PI Zero W;

b) permitir integrar a placa Raspberry PI Zero W com o conector padrão SAE J1962

do veículo via Bluetooth através de um adaptador ELM327;

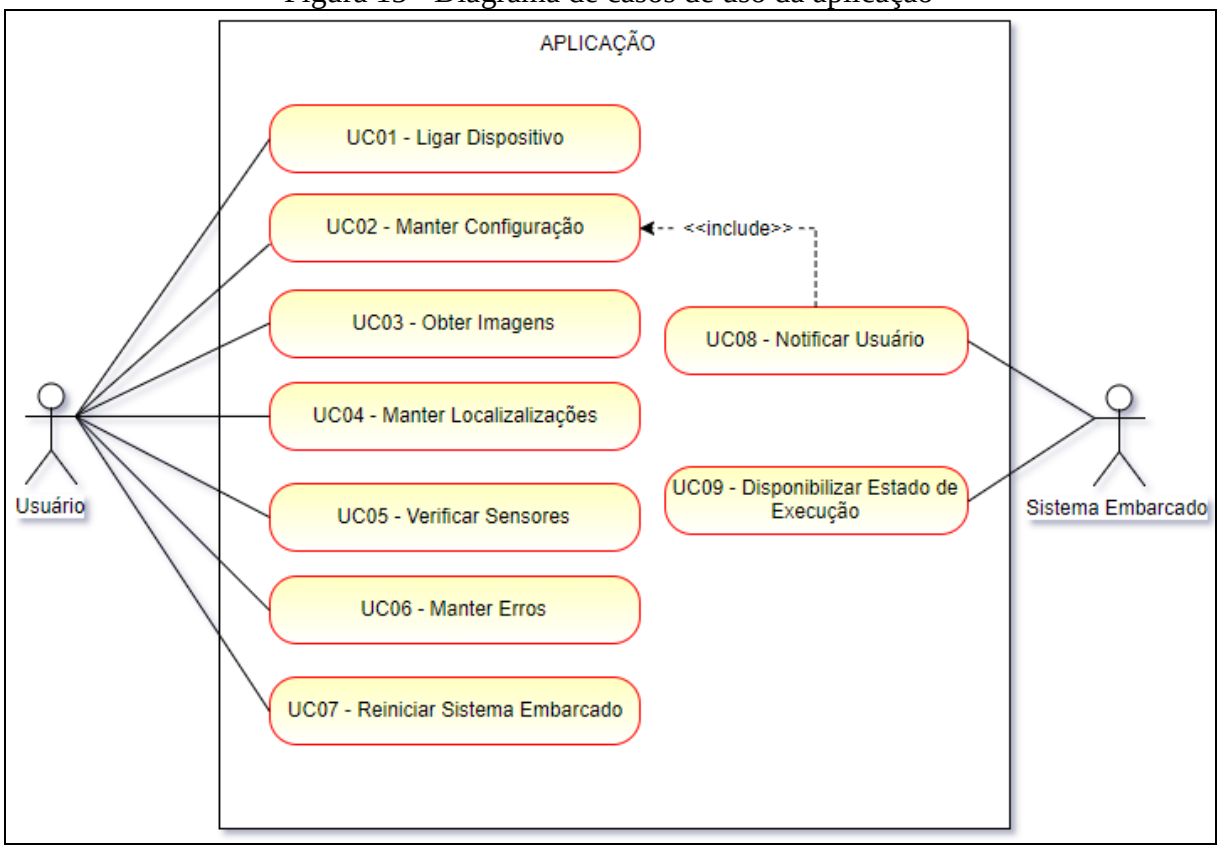
c) permitir integrar a placa Raspberry PI Zero W com o módulo GPS Ubox GY-GPS6MV2;
d) permitir integrar a placa Raspberry PI Zero W com o módulo WAVGAT Raspberry Pi Camera Revision 1.3;
e) utilizar a operadora TIM com o modem USB 3G da marca ZTE MF626;
f) utilizar o sistema operacional Raspian instalado na placa Raspberry Pi Zero W;
g) permitir que o sistema embarcado persista dados em formato JavaScript Object Notation (JSON), sem a necessidade de banco de dados;
h) permitir que a aplicação persista dados em formato JSON na memória do aplicativo *mobile*.
i) utilizar a linguagem de programação Python;
j) utilizar a biblioteca Python OBD;
k) utilizar a biblioteca Flask como servidor de aplicações *web*;
l) utilizar a biblioteca Ionic Framework para desenvolver a aplicação *mobile*;
m) utilizar a ferramenta Draw.io para a modelagem de diagramas UML;
n) utilizar a ferramenta Scheme-it para desenvolver o diagrama de esquema eletrônico.

3.1.2 DIAGRAMA DE CASOS DE USO

Nesta sessão é apresentado o diagrama de casos de uso (exibido na Figura 15). No total, foram necessários nove casos de uso que representam as principais funcionalidades da aplicação desenvolvida. Conforme pode-se observar, foram identificados dois atores, são eles:

a) `Usuário`: ator responsável por comandar a aplicação;
b) `Sistema Embarcado`: este ator é designado à dar *feedback* do dispositivo e notificar alguma falha (caso existir) no OBD2.

Figura 15 - Diagrama de casos de uso da aplicação



Fonte: elaborado pelo autor.

No caso de uso `UC01 - Ligar Dispositivo`, é possível que o usuário ligue o dispositivo e desligue-o através de um botão. O caso de uso `UC02 - Manter Configuração` destina-se para que o usuário possa fazer a configuração de e-mail, telefone e se deseja receber as notificações emitidas caso houverem falhas DTC. Já o `UC03 - Obter imagens` propõe-se a capturar imagens do dispositivo, elas podem ser salvas na aplicação *mobile* ou visualizadas através de *stream* em tempo real. No caso de uso `UC04 - Manter Localizações`, o usuário poderá capturar a geolocalização do veículo e também é permitida a visualização das últimas localizações solicitadas. No caso de uso `UC05 - Verificar Sensores` o usuário pode visualizar dados dos sensores do automóvel em tempo real. O caso de uso `UC06 - Manter Erros` possibilita que o usuário visualize dados de códigos de erro (DTC) e estado da lâmpada MIL, além disso, este caso de uso permite a limpeza dos DTCs. Por fim, no `UC07 - Reiniciar Sistema Embarcado` o usuário pode fazer a reinicialização do sistema embarcado no dispositivo.

Já o `Sistema Embarcado` tem acesso à dois casos de uso, são eles:

a) `UC08 - Notificar Usuário`: permite que o sistema embarcado notifique o usuário se ocorrer algum erro DTC, essa notificação é configurada previamente pelo `Usuário` no `UC02`;

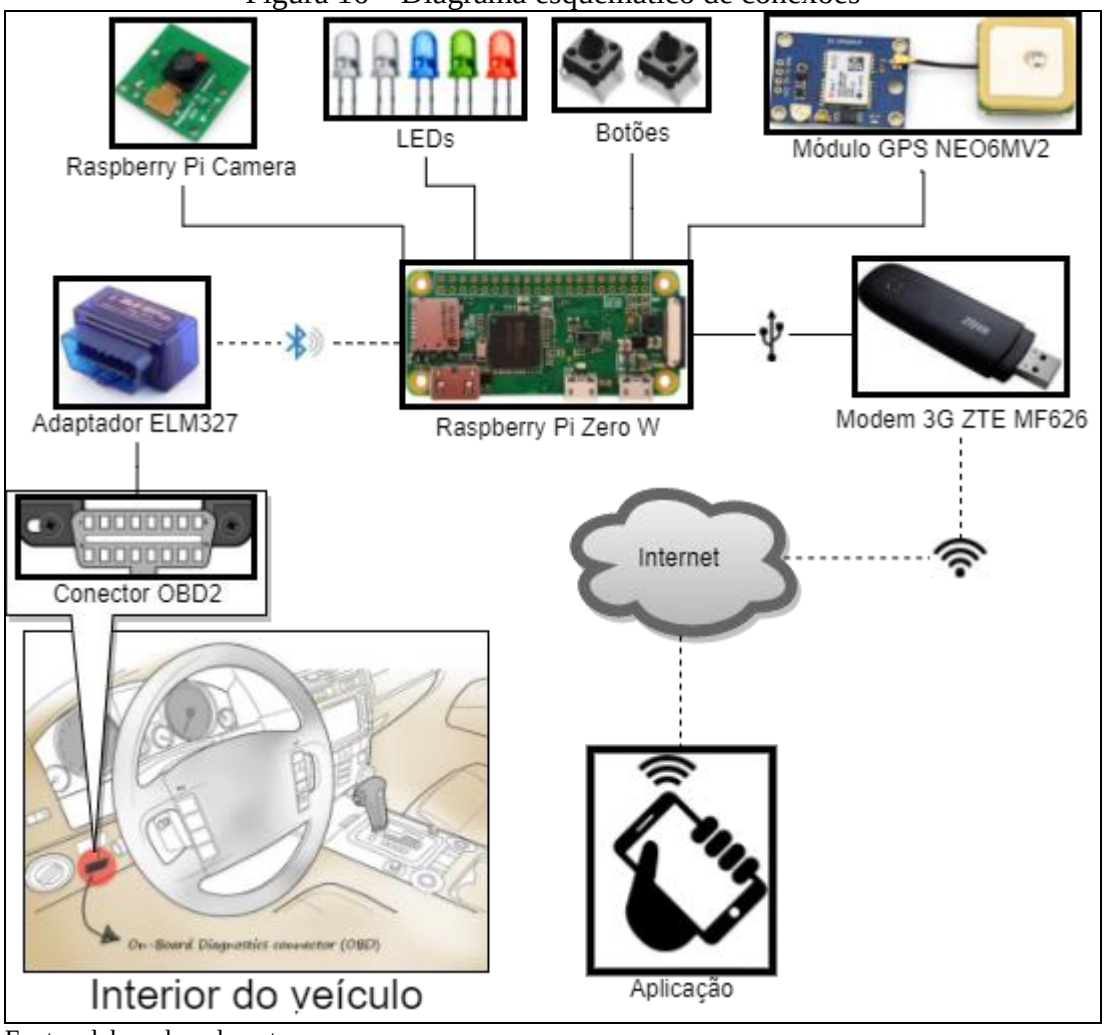
b) `UC09 – Disponibilizar Estado de Execução`: permite que o sistema embarcado possa fornecer o estado de execução atual para o `Usuário` através de LEDs.

3.1.3 ARQUITETURA DE HARDWARE

Nesta seção, é exposta a arquitetura de hardware aplicada no desenvolvimento deste trabalho. Foram elaborados diagramas para apresentar as ligações de componentes eletrônicos utilizados, bem como do esquema elétrico dos elementos acoplados na placa Raspberry Pi Zero W através de General Purpose Input/Output (GPIO).

A Figura 16 ilustra o diagrama esquemático de conexões realizadas e seus respectivos meios de comunicação. Pode-se observar que a placa `Raspberry Pi Zero W` é o *core* do projeto, sendo que todas as principais conexões passam por ela até a disponibilização das informações na aplicação *mobile*.

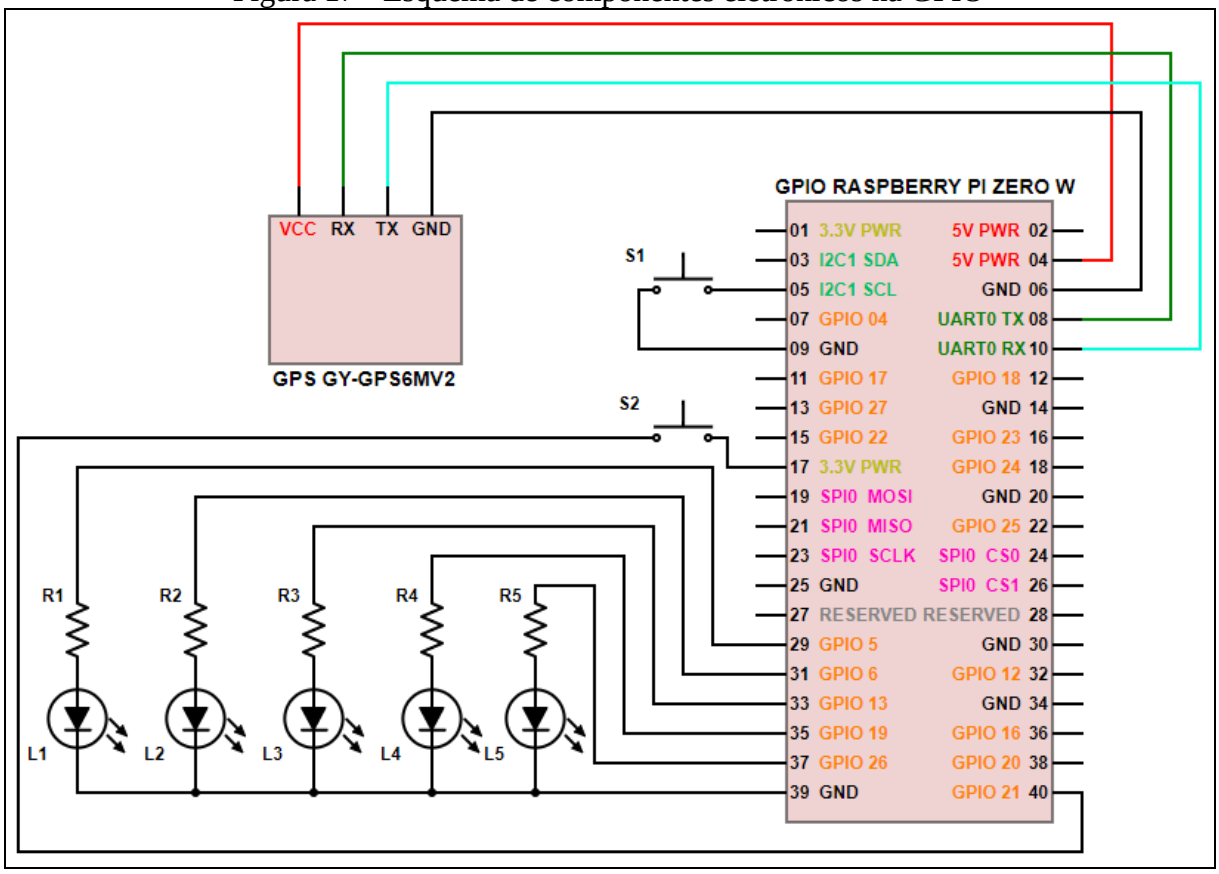
Figura 16 – Diagrama esquemático de conexões



Fonte: elaborado pelo autor.

Na Figura 17, trata-se da apresentação do esquema de componentes eletrônicos conectados através de GPIO na placa `Raspberry Pi Zero W`. Também são detalhadas as conexões e a relação do diagrama esquemático (Figura 16) com o esquema de componentes (Figura 17).

Figura 17 - Esquema de componentes eletrônicos na GPIO



Fonte: elaborado pelo autor

A placa `Raspberry Pi Zero W` conecta-se com:

a) `adaptador ELM327`: este é ligado fisicamente ao `Conector OBD2` do veículo. A conexão entre o ELM372 e a placa `Raspberry Pi Zero W` é feita através do meio de comunicação sem fio Bluetooth. É necessário pareamento prévio entre eles antes de iniciar o servidor embarcado;

b) `Raspberry Pi Camera`: esta é feita através da Camera Serial Interface (CSI) disponível nativamente na placa `Raspberry Pi Zero W`, este módulo é utilizado para capturar as imagens do interior do veículo;

c) `LEDs`: são necessários para que o usuário visualize o estado do sistema embarcado e também, alertar sobre possíveis falhas durante a inicialização. Os `LEDs` são conectados à placa `Raspberry Pi Zero W` através de GPIO. Como pode-se

observar no esquema da Figura 17 a placa contém os seguintes LEDs:

- L1: cor branca, este indica que o software embarcado está obtendo endereço de Internet Protocol (IP) público. Se a luz estiver acesa, informa que o IP foi obtido com sucesso;
- L2: cor vermelha, a qual é acesa quando ocorre alguma falha no software embarcado como por exemplo dispositivo ELM327 Bluetooth não encontrado;
- L3: cor verde, que indica a configuração da comunicação OBD2;
- L4: cor azul, que informa a configuração da comunicação entre a placa Raspberry Pi Zero W e o adaptador ELM327;
- L5: cor branca, que indica o início da execução do servidor embarcado.

Todos os LEDs estão ligados à resistores de 220 ohms, GPIO distintas e também ao mesmo terra (GND) no pino 39;

d) Botões: da mesma forma que os LEDs, dois botões são conectados fisicamente à Raspberry Pi Zero W através de GPIO (ilustrado na Figura 17). Os botões são:
   - S1: utilizado para ligar e desligar o SO do Raspberry Pi Zero W. Este botão é ligado ao terra, pois, quando o pino 05 é aterrado e o SO estiver desligado, o sistema operacional é inicializado automaticamente sem o uso de nenhuma programação. Já para desliga-lo, é criado um processo para verificar se o botão é pressionado.
   - S2: a utilidade deste botão é a reinicialização da aplicação embarcada quando pressionado por três segundos, ele é conectado ao pino 40 do barramento e ao 3.3V para que, quando pressionado, o sistema embarcado identifique que a GPIO 21 foi ativada;

e) modem 3G ZTE MF626: dispositivo conectado via USB utilizando um adaptador conversor USB, pois a placa Raspberry Pi Zero W disponibiliza somente uma porta micro USB. Este modem é utilizado para conectar a Raspberry Pi Zero W à Internet através da rede 3G. Isto é realizado através da operadora de telefonia móvel TIM;

f) módulo GPS NEO6MV2: ele estabelece uma conexão serial com a placa Raspberry Pi Zero W e é conectado através de GPIO. Sua utilidade é a captura da geolocalização do veículo. Na Figura 17, o módulo é representado pelo nome GPS GY-GPS6MV2. A atenção que deve-se ter é com relação às ligações, o RX do módulo é ligado ao pino UART0 TX (pino 08) da placa e o TX do módulo GPS é ligado ao

`UART0 RX` (pino `10`) da placa, evitando desse modo, conexões errôneas.

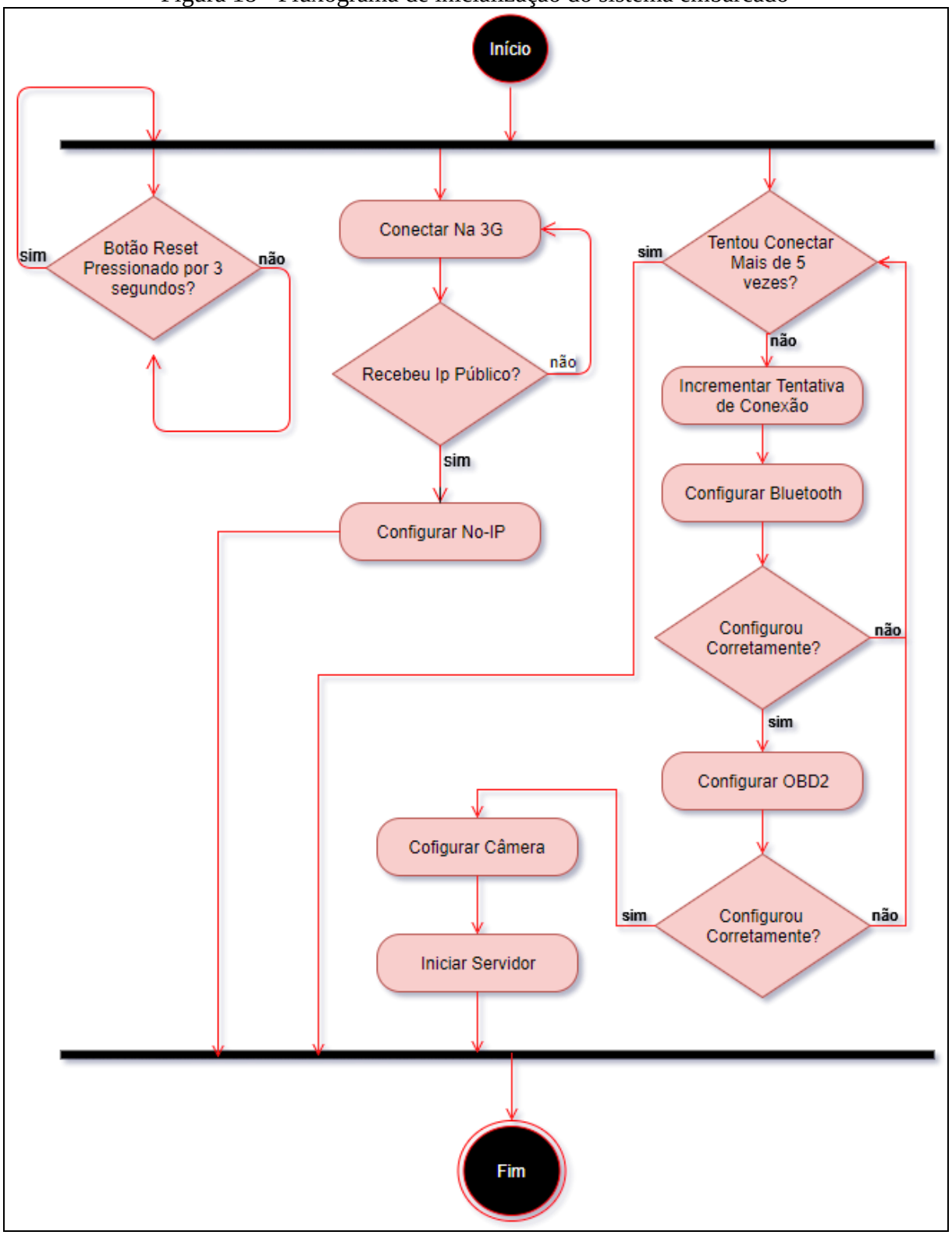
3.1.4 FLUXOGRAMA DO SISTEMA EMBARCADO

Esta sessão aborda o fluxograma ilustrado na Figura 18. Ele representa o processo de inicialização do sistema embarcado na placa Raspberry Pi Zero W. Este engloba o fluxo que é executado após a ligação da placa na energia, inicialização ou reinicialização do sistema operacional.

Na ativação do sistema operacional, executa-se um processo que inicializa o sistema embarcado, este processo obedece os seguintes passos paralelamente:

a) espera que o usuário pressione o botão de reinicialização durante 3 segundos, caso isso aconteça, o processo é reiniciado;
b) executa a discagem para conectar-se com a rede 3G até que receba um IP público, se obtiver sucesso, configura-se o serviço de Dynamic Domain Name System (DDNS) No-Ip com o novo endereço público;
c) tenta durante cinco vezes configurar as conexões Bluetooth e OBD2, caso consiga configurar o Bluetooth, o próximo passo a ser executado é a configuração OBD2. Se a conexão Bluetooth e OBD2 forem configuradas com sucesso, inicializa-se a preparação do módulo da câmera. Por fim, o servidor embarcado é ativado.

Figura 18 - Fluxograma de inicialização do sistema embarcado



Fonte: elaborado pelo autor.

REFERÊNCIAS

ALMEIDA, Eduardo Luciano de; FARIA, Felipe Freitas de. **Scanner OBD-II Em Plataforma LabView**. 2013. 139 f. TCC (Graduação) - Curso de Tecnologia em Eletrônica Automotiva, Faculdade de Tecnologia Fatec Santo André, São Paulo, 2013.

APLICAÇÕES DE AUTOMAÇÃO EM IOT - INTERNET OF THINGS. Extrema - Minas Gerais: Editorial E-locução, v. 10, 2016. Anual. Disponível em: <http://periodicos.faex.edu.br/index.php/e-locucao/article/view/104>. Acesso em: 27 out. 2017.

BARROS, Eduardo. **ESTUDO DAS DIFERENÇAS DOS REQUERIMENTOS DAS PRINCIPAIS LEGISLAÇÕES ON BOARD DIAGNOSTICS PARA PADRONIZAÇÃO DE TESTES DE DESENVOLVIMENTO E VALIDAÇÃO DE TRANSMISSÃO AUTOMÁTICA DE AUTOMÓVEIS**. 2012. 59 f. Monografia (Especialização) - Curso de Engenharia Automotiva, Centro Universitário, Centro Universitário do Instituto Mauá de Tecnologia, São Caetano, 2012.

BAUMGARTEN, Nykolas E. A., **FINDCAR**: RASTREADOR VEICULAR UTILIZANDO OPENWRT. 2016, 56 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CZERWONKA, Mariana, **Falta de manutenção triplica risco de acidentes**. [Goiás?], 2016. Disponível em <http://portaldotransito.com.br/noticias/falta-de-manutencao-triplica-risco-de-acidentes>. Acesso em: 20 mar 2017.

DIAS, Renata. R. F. **Internet das coisas sem mistérios**: uma nova inteligência para os negócios. São Paulo; Netpress Books, 2016

ELM ELECTRONICS. **OBD**. 2017. Disponível em: <https://www.elmelectronics.com/products/ics/obd/?v=19d3326f3137>. Acesso em: 31 out. 2017.

G1 Ribeirão e Franca, **Estatística divulgada pela SSP mostra aumento da violência em Ribeirão**. [São Paulo], 2017. Disponível em: <http://g1.globo.com/sp/ribeirao-preto-franca/noticia/2017/02/estatistica-divulgada-pela-ssp-mostra-aumento-da-violencia-em-ribeirao.html>. Acesso em: 20 mar 2017.

GUIMARÃES, Alexandre de Almeida. **Eletrônica Embarcada Automotiva**. São Paulo: Erica, 2007.

MACHADO, António S. L.; OLIVEIRA, Bruno R. R., **O Sistema OBD (On-Board Diagnosis)**. 2007, 8p, Mestrado em Automação e Sistemas, Instituto Superior de Engenharia do Porto, Porto - Portugal.

MANCINI, Mônica. **Internet das Coisas**: História, Conceitos, Aplicações e Desafios, 2017, 9 f. Artigo Científico. PMI Capítulo São Paulo, Brasil

MCCORD, Kleint. **Automotive Diagnostic System**: Understanding OBD-I & OBD-II. North Branch: CarTech, 2011.

NULL BYTE. **Set Up Kali Linux on the New $10 Raspberry Pi Zero W**. 2017. Disponível em: <https://null-byte.wonderhowto.com/how-to/set-up-kali-linux-new-10-raspberry-pi-zero-w-0176819/>. Acesso em: 25 maio 2017.

OBD SOLUTIONS. **WHAT IS OBD?**. [2017?]. Disponível em: <http://www.obdsol.com/knowledgebase/on-board-diagnostics/what-is-obd/>. Acesso em: 24 maio 2017.

ODEGA, Alessandra, **Confira os modelos de veículos mais roubados e quanto custa o seguro de cada um deles**. Santa Catarina, 2016. Disponível em: <http://ndonline.com.br/florianopolis/coluna/alessandra-ogeda/confira-os-modelos-de-veiculos-mais-roubados-e-quanto-custa-o-seguro-de-cada-um-deles>. Acesso em: 20 mar 2017.

OUTILS OBD FACILE. **ELM327 CAR DIAGNOSTICS INTERFACES**. 2017. Disponível em: <https://www.outilsobdfacile.com/diagnostic-interface-elm-327.php>. Acesso em: 12 nov. 2017.

PACHECO, Lucas V., **Monitoramento e gestão de uma frota de veículos utilizando sistemas embarcados**. 2016, 76 f. Trabalho de Conclusão de Curso (Curso de Engenharia Elétrica) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo.

PINA, Afonso L. P., **SISTEMA DE LOCALIZAÇÃO DE VEÍCULOS PARA SMARTPHONE ANDROID**. 2015, 136p, Tese/Dissertação de Mestrado (Engenharia Eletrotécnica e de Computadores) – Departamento de Engenharia Eletrotécnica, Instituto Superior de Engenharia do Porto, Porto - Portugal.

RASPBERRY PI FOUNDATION, **FAQS**. [2017?]. Disponível em: <https://www.raspberrypi.org/help/faqs/>. Acesso em: 01 abr. 2017.

SANTOS, Arthur Luis V., **Economia de combustível com o uso de telemetria para veículos de passeio**. 2016, 82 f. Trabalho de Conclusão de Curso (Engenharia de Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul.

SANTOS, Bruno P. et al. **Internet das Coisas**: da Teoria à Prática, [2016?], 50 f. Artigo científico. Departamento de Ciência da computação, Universidade Federal de Minas Gerais

SOUZA, Beatriz, **Quatro em cada dez veículos de carga apresentam falha mecânica**. [Santa Catarina?], 2016. Disponível em: <http://www.perkons.com.br/noticia/1694/quatro-em-cada-dez-veiculos-de-carga-apresentam-falha-mecanica>. Acesso em: 20 mar 2017.

STAROSKY, Ricardo A., **OBD-JRP**: monitoramento veicular com java e raspberry pi. 2016, 87 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TOP CAR DIAGNOSTICS SUPPORT. **Description of OBD Communication Standards / Protocols**. 2014. Disponível em: <http://www.totalcardiagnostics.com/support/Knowledgebase/Article/View/74/0/description-of-obd-communication-standards--protocols>. Acesso em: 31 out. 2017.