CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA

FATEC SANTO ANDRÉ

Tecnologia em Eletrônica Automotiva

ANDRÉ INFANTE QUINTAS

GUSTAVO MOTA SILVA

VITOR WIETKY GARCIA

MONITORAMENTO DE CONDUÇÃO

Santo André – São Paulo

2018

ANDRÉ INFANTE QUINTAS

GUSTAVO MOTA SILVA

VITOR WIETKY GARCIA

MONITORAMENTO DE CONDUÇÃO

Trabalho de Conclusão de Curso entregue à Fatec Santo André como requisito parcial para obtenção do título de Tecnólogo em Eletrônica Automotiva.

Orientador: Prof.

Weslley Medeiros Torres

Santo André – São Paulo

2018

FICHA CATALOGRÁFICA

Q78m
Quintas, André Infante
Monitoramento de condição / André Infante Quintas, Gustavo Mota Silva, Vitor Wietky Garcia. - Santo André, 2018. 55f: il.

Trabalho de Conclusão de Curso – FATEC Santo André.
Curso de Tecnologia em Eletrônica Automotiva, 2018.

Orientador: Prof. Wesley Medeiros Torres

1. Eletrônica. 2. Veículos. 3. Sistemas eletrônicos. 4. Desenvolvimento. 5. Sensores. 6. Atuadores. 7. Rede CAN. 8. Consumo. 9. Combustível. I. Silva, Gustavo Mota. II. Garcia, Vitor Wietky. III. Monitoramento de condição.

612.389

Fatec CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Santo André

LISTA DE PRESENÇA

SANTO ANDRÉ, 20 DE DEZEMBRO DE 2018.

LISTA DE PRESENÇA REFERENTE À APRESENTAÇÃO DO TRABALHO DE CONCLUSÃO DE CURSO COM O TEMA **"MONITORAMENTO DE CONDUÇÃO"** DOS ALUNOS DO 6º SEMESTRE DESTA U.E.

BANCA

PRESIDENTE:
PROF. WESLLEY MEDEIROS TORRES _____

MEMBROS:
PROF. CARLOS ALBERTO MORIOKA _____

PROF. FERNANDO GARUP DALBO _____

ALUNOS:

ANDRÉ INFANTE QUINTAS _____

GUSTAVO MOTA SILVA _____

VITOR WIETKY GARCIA _____

AGRADECIMENTOS

Agradecemos a nossa família por nos apoiarem em todos os momentos do curso. E agradecemos aos professores da Faculdade de Tecnologia (FATEC). Citaremos os nomes de todos os professores porque acreditamos que cada professor foi fundamental para a aprendizagem e conhecimento adquirido no curso.

Agradecimento aos Prof. Ademauro Volponi, Prof. Adriano Ribolla, Prof. Armando Laganá, Prof. Carlos Alberto Morioka, Prof.Celso Aparecido João, Prof. Celso Tabajara Teixeira, Prof. Cleber Willian Gomes, Prof. Dirceu Lavoisier Graci Fernandes, Prof. Edson Caoru Kitani, Prof. Fábio Delatore, Prof. Flávio Barrela, Prof. Jhonny Frank Sousa Joca, Prof. Kleber Nogueira Hodel, Prof. Luciano Breve Abrahão, Prof. Luiz Roberto Kanashiro, Prof. Luiz Vasco Puglia, Prof. Manoel Francisco Guaranha, Prof. Marco Aurélio Fróes, Prof. Moacyr da Silva Caminada, Prof. Orlando de Salvo Junior, Prof. Paulo Tetsuo Hoashi, Prof.ª Priscilla Lastremski, Prof. Roberto Nicolosi, Prof. Rogério Rodrigues Lima Cisi, Prof.ª Suely Midori Aoki, Prof. Wagner Massarope.

Agradecemos também aos profissionais que trabalham na secretaria, profissionais de limpeza, profissionais de manutenção, e todos os colaboradores que ajudam de alguma forma para o funcionamento da instituição.

E um agradecimento especial ao Prof. Fernando Garup Dalbo por todo apoio e ajuda no desenvolvimento da nossa monografia, e ao nosso orientador o Prof. Weslley Medeiros Torres, por toda a ajuda e atenção durante todo o processo do desenvolvimento e testes do nosso projeto de graduação. Além disso, agradecemos ao aluno Rafael Oviedo que nos ajudou emprestando um equipamento de diagnóstico para desenvolvimento do projeto e compartilhando sobre a sua experiência em um sistema de rastreamento. E ao grupo composto pelos alunos Carlos Henrique, Gustavo Ortega, Vitor Benicio, que tinham um projeto semelhante, e na fase inicial foi possível trabalharmos juntos e assim conseguirmos alguns progressos.

"Estamos em risco de nos destruir por conta de nossa cobiça e estupidez. Não podemos permanecer olhando para dentro de nós em um planeta pequeno e crescentemente poluído e superpovoado."

Stephen Hawking

RESUMO

Nesse trabalho foi desenvolvido um sistema que busca encontrar a forma ideal de condução do veículo, utilizando os dados de sensores/ atuadores existentes e obtendo essas informações através da rede de comunicação CAN (*Controller Area Network)* do veículo, é possível coletar os dados desses sensores.

Os dados obtidos são processados por uma placa embarcada, cuja função é realizar a comparação com as informações presentes no manual do veículo, a partir dessas informações, o sistema indica qual o modo atual de condução do veículo. O principal objetivo desse sistema é a redução de consumo de combustível e o impacto no meio ambiente.

Palavras chaves: CAN, condução, consumo de combustível, meio ambiente, sensores.

ABSTRACT

Searching for the ideal driving form, were used some sensors that are already present in the vehicle. Through the Controller Area Network (CAN) of the vehicle, is possible to collect the data of those sensors because is through this line of communication that various informations about the engine and other electronic modules of the vehicle communicate.

The data obtained are processed by an embedded board, whose function is to perform the comparison with the information present in the vehicle manual, from this information, the system indicates the current mode of driving of the vehicle. The main objective of this system is the reduction of fuel consumption and the impact on the environment.

Keywords: CAN, driving, fuel consumption, environment, sensors.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE SIGLAS E ABREVIATURAS

ABS	*Anti-lock Braking System*
ACK	*Acknowledgement*
ARM	*Acorn RISC Machine*
CAN	*Controller Area Network*
CRC	*Cyclic Redundancy Check*
CV	Cavalo-vapor
DTC	*Diagnostic Trouble Codes*
GB	*GigaByte*
GHz	*Gigahertz*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
HDMI	*High-Definition Multimedia Interface*
INMETRO	Instituto Nacional de Metrologia, Qualidade e Tecnologia
IOT	Internet das Coisas
Kg	Quilograma
Kgfm	Quilograma força x metro
Km/h	Quilômetros por Hora
OBD	*On-Board Diagnostic*
OICA	Organização Internacional de Construtores Automotivos
PC	*Personal Computer*
RAM	*Random Access Memory*
RPM	Rotações por Minuto
TPS	*Throttle Body Position*
UART	*Universal Asynchronous Receiver-Transmitter*
USB	Universal Serial Bus
Wi-Fi	*Wireless Fidelity*

SUMÁRIO

1. INTRODUÇÃO

Segundo a Organização Internacional de Construtores Automotivos (OICA), em 2017 o Brasil, ficou na nona posição dos maiores produtores de veículos do mundo. Isso mostra o quanto nosso país é importante dentro no cenário mundial principalmente se levarmos em consideração que entre os dez maiores produtores de veículos, somente a Índia e o Brasil não são considerados países desenvolvidos. (OICA, 2017)

Os veículos deixaram de ser considerados um bem supérfluo e passaram a se tornar um bem essencial, principalmente em grandes centros urbanos como a cidade de São Paulo. Pessoas estão sempre precisando se locomover, seja para trabalhar ou para lazer. Em uma grande metrópole existem diversos meios de transportes, como trens, ônibus e carros. No caso de veículos a combustão sempre temos a necessidade de ter um motorista, seja ele dono do próprio veículo, motorista de ônibus, taxista, motorista de aplicativo, veículo alugado, entre outros.

Em um veículo com motor à combustão sempre temos um regime de rotação para extrair a máxima eficiência do motor com o mínimo de consumo de combustível. Para uma condução ideal visando consumo de combustível é muito importante definir a marcha a ser utilizada em relação à velocidade, para manter o motor o mais próximo possível desse regime de rotação. A falta de conhecimento ou até mesmo a preguiça de realizar reduções ou passar a marcha são fatores que acabam levando o motorista a não utilizar esse regime de rotação, e isso impacta diretamente no aumento de consumo de combustível. (BROMBACHER,2017)

Um outro fator que pode fazer com que o regime ideal de rotação não seja utilizado é a distração do motorista. Seja por distrair de fatores externos ao veículo ou até mesmo interno. Como por exemplo quando ocorre algum tipo de acidente na via e o motorista por curiosidade acaba tentando identificar o que ocorreu ou quando temos crianças dentro do veículo, toda a atenção é dada à elas quando comparado a condução do veículo. Além disso, no curso para obtenção da carteira nacional de habilitação, o futuro motorista emprega um tempo muito pequeno para aprendizagem, dessa forma fica impossível lhe apresentar ou vivenciar todas as situações de

condução do veículo que ocorrerão no dia-a-dia. E esse é um dos motivos de o motorista criar manias que podem influenciar na forma de condução.

Com a evolução dos sistemas embarcados nos veículos, a quantidade de sensores nos veículos vem aumentando com o passar dos anos. Dificilmente encontramos veículos que possuem apenas um único módulo de gerenciamento. Geralmente temos diversos módulos onde cada um fica responsável por uma função, entre elas injeção eletrônica, sistema de frenagem antitravamento *(*ABS), sistema de conforto, além de entretenimento.

Dentro do veículo os módulos precisam compartilhar certas informações e isso ocorre através do canal de comunicação chamada *Controller Area Network* (CAN). Esses dados podem ser utilizados para auxiliar o motorista a tomar decisões, visando um modo de condução que promova um uso mais eficiente do veículo e traz benefícios como diminuir o consumo de combustível. Porém informações como rotação do motor, velocidade do veículo e temperatura do líquido de arrefecimento, acabam sendo difíceis para um motorista leigo fazer uma relação entre elas.

1.1. Objetivo

Realizar a coleta de dados através da rede CAN do veículo, utilizando uma interface de comunicação via conector *On-Board Diagnostic* (OBD) II e enviar essas informações utilizando conexão Bluetooth para uma unidade de processamento. Todos esses dados serão processados, analisados e ficaram registrados na unidade de processamento.

Essas informações serão relacionadas com o modo de condução do motorista e será sugerido uma condução mais econômica de acordo com o recomendado pelo fabricante do veículo para assim diminuir o consumo de combustível.

Outro fator importante que poderemos atingir, é em relação ao meio ambiente. Diminuindo o consumo de combustível, iremos indiretamente melhorar o impacto que o veículo causa ao meio ambiente através da diminuição de emissões de gases poluentes.

1.2. Motivação

No setor automotivo temos alguns exemplos de tendência visando economia de combustível e uma condução mais segura, como por exemplo a Porto Seguro que oferece desconto no seguro, dependendo da forma de condução. Caso o motorista não receba nenhuma multa no período de um ano, o motorista irá receber desconto na renovação do seguro. Uma outra situação, é a possibilidade de contratação de um plano diferenciado para pessoas de faixa etária entre 18 e 24 anos, chamado Auto Jovem, nesse plano é instalado um rastreador no veículo, onde dois parâmetros são analisados: tempo de veículo rodando no período de madrugada e tempo com velocidade acima de 70 km/h durante um ano, esses fatores são determinantes para a renovação do seguro veicular. (PORTO SEGURO, 2018)

Recentemente, a Porto Seguro lançou um aplicativo para celular que realiza o monitoramento de alguns parâmetros e atribui nota para o motorista. São cinco categorias, aceleração, frenagem, curva, velocidade, uso do celular. O aplicativo usa o acelerômetro do próprio celular do motorista para definir uma pontuação ao motorista e no final os melhores colocados são premiados. (TRÂNSITO MAIS GENTIL, 2018)

Utilizando essa metodologia o nosso projeto realizará a sugestão de condução ao motorista visando economia de combustível.

2. FUNDAMENTAÇÃO

O sistema de monitoramento de condução irá analisar o modo de condução do motorista, coletando dados e enviando para processamento, com isso iremos poder promover medidas para o condutor obter uma maior economia de combustível.

Para realizar toda a análise que estamos propondo, buscar parâmetros que indicam um melhor modo de condução, as informações mais superficiais estão no próprio manual do veículo, onde podemos encontrar sugestões da velocidade ideal para troca de marcha do veículo. Com base nessas sugestões atreladas aos dados obtidos no veículo poderemos começar nossas análises.

Além disso, serão aplicadas as principais estratégias de funcionamento do motor que serão cruciais para realizar as análises.

2.1. Estado da arte

Como principais inspirações para o desenvolvimento deste projeto, foram utilizados dois artigos com projetos desenvolvidos que visam economizar combustível com base no modo de condução do motorista.

O artigo "*Artemisa: Using an Android device as an Eco-Driving assistant"* demonstra um sistema de direção ecológica, chamado Artemisa, como principal objetivo alcançar um valor acessível para os seus clientes devido ao preço alto dos demais sistemas já desenvolvidos. E como recurso de redução de custo é utilizado o sistema Android, que é uma plataforma utilizada por grande parte da população, em forma de aplicativo. O Artemisa monitora a forma como o motorista está dirigindo, principalmente em situações que envolvem um número elevado de trocas de marchas, baseado em dados obtidos pelo veículo comparando com o consumo de combustível. Assim é possível alertar o motorista de pontos críticos dando-lhe dicas de como reduzir o consumo. (Magaña e Organero, 2011)

O artigo "*Intelligent eco-driving suggestion system based on vehicle loading model"* demonstra um sistema de condução inteligente baseado na carga do veículo,

onde a economia de combustível gerada é calculada por meio de dados do sistema de diagnóstico de bordo. O algoritmo desenvolvido sugere para o motorista qual forma ele deve conduzir o veículo para chegar no valor mais próximo de consumo ideal em cada situação de carga do veículo. De acordo com testes realizados, este sistema implementado consegue reduzir o consumo em 7% do que era utilizado. Além de ser desenvolvido na plataforma Android que torna o custo do projeto menor. (Choue e col. 2013)

Com base nesses artigos, o projeto foi desenvolvido pensando na forma de condução do veículo, capturando dados que serão transformados em informações e então gerar uma análise sobre o consumo de combustível do veículo e informar o motorista como ele está dirigindo. Para isso foi empregado um computador embarcado *Raspberry* e utilizado a linguagem Python para o desenvolvimento do software.

2.2. Arquitetura eletrônica

Com o passar dos anos e o avanço da tecnologia, os veículos foram ficando mais modernos, novos motores com mais sensores, mais entretenimento para os ocupantes, mais recursos de conforto e performance. Com o aumento da quantidade de componentes eletrônicos dentro dos veículos, o poder de processamento das CPUs necessárias para realizar a tarefa de interpretar toda essa quantidade de informação, também cresceu. Muitos sistemas dos veículos passaram a possuir uma central eletrônica exclusiva para determinada tarefa. E por causa dessa grande quantidade de informações, e de sensores e atuadores começaram os problemas com uma grande quantidade de cabos necessários para transmitir todas essas informações.

Surgiu a partir desta necessidade a implementação de um método para diminuir essa quantidade de cabos. A empresa alemã BOSCH desenvolveu a tecnologia de comunicação de rede CAN na década de 80, e através dessa tecnologia foi possível diminuir consideravelmente a quantidade de cabos dentro dos veículos. (J. Massamy Taniguchi, 2017)

2.2.1. Rede CAN

A rede CAN é um canal de comunicação entre os diversos módulos localizados dentro do veículos, sensores e atuadores, por meio de barramento onde são transmitidas as mensagens codificadas.

Os dados são enviados em forma de *frame*, que é formado por um conjunto de bytes, onde cada byte exerce uma função no *frame*.

Os *frames* podem ser divididos em 4 tipos (dados, remoto, erro e *overload*):

- *Frame* de dados: é o *frame* mais usado, onde transmite uma informação para o funcionamento do sistema;
- *Frame* remoto: quando um módulo ou atuador necessita de uma informação que não foi enviada para o sistema, é enviado um *frame* requisitando esta informação. Mas não é utilizado na indústria automotiva;
- *Frame* de erro: é utilizado quando existe algum erro no barramento;
- *Frame* de *overload*: quando existe uma sobrecarga de mensagem no sistema, este *frame* é enviado para informar o tempo necessário para ler os *frames*. (Kleber, 2018)

O *frame* é estruturado a partir de vários blocos, ou seja, é um conjunto de bytes onde eles são divididos indicando cada parte do *frame*, como mostrado na figura 1. O quadro 1 possui uma breve descrição do que representa cada parte do *frame*.

Figura 1 - Blocos do *frame*.

Fonte Imagem: Professor Kleber – Material de Aula da Fatec Santo André

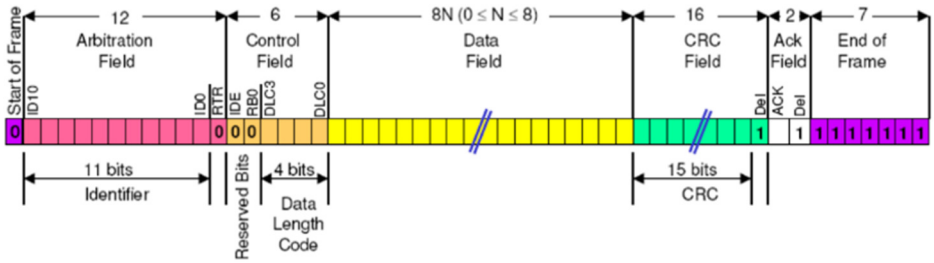
Quadro 1 - Blocos do *frame* da Rede CAN.

BLOCO	FUNÇÃO
Start of Frame	Sincroniza todos os membros da rede indicando que é o bit de início do *frame*.
Identifier	Identifica o para onde o dado deve ser enviado e o seu nível de prioridade em relação aos demais *frames*. Dependendo do tipo de CAN que o veículo possui, o "*identifier*" pode ter dois tamanhos, no CAN 2.0A são de 11 bits e no CAN 2.0B são de 29 bits.
Control	Controla qual é o tipo de mensagem enviado e o tamanho da mensagem.
DATA	Conteúdo do *frame*, ou seja, os dados a serem enviados. Pode ter o tamanho de até 8 byte controlado pelo "*Control*".
CRC	É um conjunto de byte para gerar segurança para a mensagem, onde o CRC da mensagem é comparado com o CRC do receptor e assim liberando ou não a função ACK.
ACK	É o bit que indica se a mensagem está correta ou errada. Caso a comparação do CRC estiver correta é enviado valor bit dominante e caso contrário é enviado bit recessivo (gerando um *frame* de erro).
End of Frame	Indica o final do *frame*, gerando tempo para enviar o frame de erro caso for necessário. É formado por 7 bits recessivos.
Interframe Space	É o espaço necessário que existe entre dois *frames*, formado por 3 bits recessivos.

Fonte Quadro: Professor Kleber Nogueira Hodel – Material de Aula da Fatec Santo André

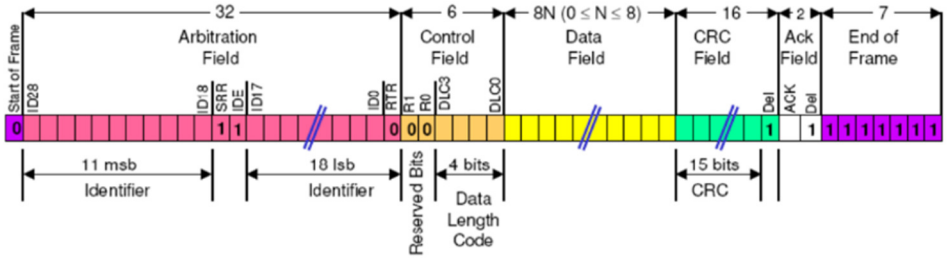
A figura 2 e a figura 3, monstra a divisão e o tamanho de cada parte do *frame,* com a diferença que na primeira possui o campo de identificação de 11 Bits e na segunda figura o campo de identificação é de 29 Bits

Figura 2 - *Frame* rede CAN 11 Bits *Identifier.*



Fonte Imagem: Professor Kleber Nogueira Hodel – Material de Aula da Fatec Santo André

Figura 3 - *Frame* Rede CAN 29 Bits *Identifier*.



Fonte Imagem: Professor Kleber Nogueira Hodel – Material de Aula da Fatec Santo André

2.2.2. Conector de diagnóstico

O conector para *On-Board Diagnostic* (OBD) II é utilizado para realizar a comunicação entre os dois hardwares: o módulo de diagnose e a rede CAN. Assim podendo receber dados necessários para executar a análise do que é solicitado pelo usuário. Na figura 4 temos a imagem do conector.

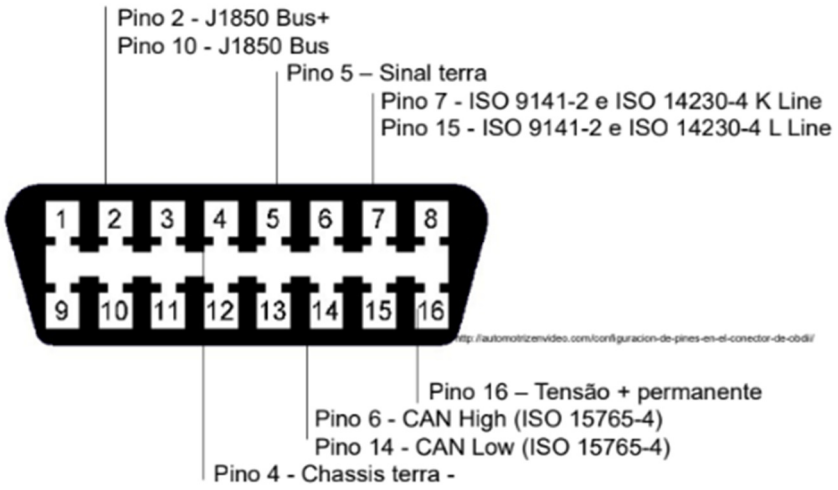
Figura 4 - Conector OBD II.

Fonte Imagem: http://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos%202007-08/Trabalhos/SIAUT_OBD.pdf

O OBD II surgiu da necessidade de melhorar as emissões de gases, com esse sistema era possível realizar um controle eletrônico do motor do veículo. Esse conector de segunda geração definiu um padrão onde todas as fabricantes de veículos na época teriam que seguir. Além de possuir um padrão de conector único. (Machado e Oliveira, 2007)

Na figura 5, mostra como é distribuída o posicionamento dos protocolos em relação aos pinos do conector.

Figura 5 - Pinagem do Conector OBD II.



Fonte Imagem: Professor Orlando de Salvo Junior – Material de Aula da Fatec Santo André.

Os fabricantes de veículos devem seguir as regras de pinagem de acordo com o protocolo de comunicação que irão utilizar. Além disso existe uma padronização em relação aos códigos de falhas chamados *Diagnostic Trouble Codes* (DTC). O quadro 2 mostra como são divididas as categorias desses códigos.

Quadro 2 - DTC's.

Grupo	Intervalo de Códigos
Carroceria	B0xxx - B3xxx
Chassi	C0xxx - C3xxx
Trem de força	P0xxx - P3xxx
Rede	U0xxx - U3xxx

Fonte Quadro: Professor Orlando de Salvo Junior – Material de Aula da Fatec Santo André.

2.2.3. Acesso ao barramento de dados do veículo

O ELM327 é um tipo de scanner automotiva, que utiliza o padrão de conector OBD II para conseguir coletar as informações do veículo através da rede CAN, como por exemplo, rotação do motor, temperatura do líquido de arrefecimento, temperatura

do óleo, avanço de ignição, temperatura do ar admitido, velocidade do veículo, pressão no coletor, entre diversas outras opções. Tudo vai depender também dos sensores que o veículo já possui. A figura 6 mostra o conector que será utilizado no projeto.

Figura 6 - Conector ELM327.

Fonte da Imagem: Próprio Autor.

Internamente o ELM327 trabalha usando um microcontrolador modelo PIC 18F2480 e utiliza uma interface do tipo *Universal Asynchronous Receiver-Transmitter* (UART) para enviar os dados, que podem ser enviados por *Universal Serial Bus* (USB) ou por *Bluetooth* dependendo do modelo do ELM327.

Segundo Elm Electronics Inc (2018) fabricante do scanner, os protocolos compatíveis são:

- SAE J1850-PWM;
- SAE J8150-VPW;
- ISO 9141-2;
- ISO 14230-4;
- ISO 15765-4 (CAN);
- SAE J2411 (SWCAN);
- KW1281 (SAE J2818);

- SAE J1939;
- SAE J1708 (J1587);
- SAE J1708 (J1922).

2.2.4. Unidade de processamento de dados

Nosso projeto utiliza uma unidade de processamento dedicada para realizar a função de análise dos dados. Utilizamos uma *Raspberry*, que possui um alto poder de processamento fundamental para realizar o processamento de uma grande quantidade de dados. Além disso o *Raspberry* é bastante utilizado no conceito de internet das coisas (IOT), e possui Linux como seu sistema operacional.

O *Raspberry* modelo PI 3 B é um tipo de "micro *Personal Computer* (PC)", possui um processador *quad-core Broadcom BCM2837* de 64 bits com *clock* 1.2 *Ghz*, além de ter 1 GB de memória *Random Access Memory* (RAM) e possuir *Wireless Fidelity* (WIFI) e *Bluetooth* integrado. Tem 4 portas USB para conexão de algum periférico externo e ainda temos uma conexão *High-Definition Multimedia Interface* (HDMI) para ligação em monitores ou Tvs. A figura 7 mostra como é o layout do *Raspberry* que será utilizado no projeto. (Eduardo Gonçalves, 2018)

Figura 7 - *Raspberry* PI 3 B.

Fonte Imagem: Próprio Autor

O *Raspberry* utiliza a arquitetura *Advanced RISC Machine* (ARM) no processador em vez do modelo de x86 que é utilizado no PC. A principal diferença entre eles é que o modelo x86 possui mais recursos integrados no próprio processador, que permitem executar instruções mais complexas. Isso torna os processadores do tipo x86 com um poder de processamento maior. Porém na arquitetura ARM temos um consumo de energia menor, o que dá a preferência para esse tipo de processador ser utilizado em dispositivos portáteis, como smartphones e aparelhos *global positioning system (*GPS).

Assim como o PC, o *Raspberry* precisa de um sistema operacional para ser utilizado. Para isso utilizaremos uma distribuição *Linux*. Temos ainda os *General Purpose Input Output* (GPIO) que funcionam como portas de entrada ou saída de determinado sinal.

Esse dispositivo será utilizado para realizar toda a comunicação com nosso ELM327 e realizar o processamento das informações coletadas do veículo.

2.3. Características do Veículo de Testes

Utilizaremos para realizar o teste um veículo da marca Nissan, modelo March, como o mostrado na Figura 8.

Figura 8 - Veículo de Teste.

Fonte Imagem: Próprio Autor

Algumas características importantes do veículo, de acordo com o site “Carros Na Web”:

- Ano: 2013/2014;
- Peso: 964 kg;
- Potência: 111 cv a 5600 Rotações por minuto (RPM);
- Torque: 15,1 kgfm a 4000 RPM;
- Flex.

2.4. Consumo de combustível

O Instituto Nacional de Metrologia, Normalização e Qualidade Industrial (INMETRO) divulga todos os anos o consumo dos veículos fabricados no Brasil. O consumo de combustível é usado para medir a eficiência energética dos veículos,

gerando uma classificação de veículos mais eficientes. O teste é realizado em laboratório com temperatura e umidade do ar controlada para estabelecer um padrão em comum com todos os testes. O teste é realizado para simular situações de rodagem na cidade e na estrada para determinar o consumo do veículo. No Quadro 3, temos os valores que o nosso veículo de teste obteve.

Quadro 3 - Consumo de Combustível do Veículo de Teste.

Etanol		**Gasolina**	
Cidade	**Estrada**	**Cidade**	**Estrada**
8,1 Km/l	9,3 Km/l	11,6 Km/l	13,2 Km/l

Fonte Quadro: Adaptada do INMETRO

Nosso veículo recebeu nota B na categoria, as notas vão de “A” até “E”, sendo nota “A” como melhor consumo. Isso significa que nosso veículo possui um bom consumo, porém não é o melhor do grupo. (INMETRO, 2018)

Na classificação geral recebeu nota “A”, isso significa que se levarmos em consideração todos os veículos da frota nacional ele possui um ótimo consumo.

2.4.1. Estimativa de impacto

O Quadro 4 mostra a quantidade de veículos comercializados no ano de 2017, considerando todas as categorias. Isso demonstra a grande quantidade de veículos que podem se tornar candidatos para utilização do nosso sistema.

Quadro 4 - Vendas de Veículos no Ano 2017.

Categoria	**Qtd. de Vendas**
Autos	1.855.874
Com. Leves	316.361
Caminhões	52.069
Ônibus	15.099
Total	2.239.403

Fonte Quadro: Adaptado da Fenabrave

Tomando como base uma média de dez mil quilômetros rodados por ano com o veículo, o quadro abaixo demonstra uma estimativa de gastos com gasolina, levando

em consideração a gasolina a um preço fixo e o consumo do nosso veículo de teste. Além disso também estimamos a economia gerada levando em consideração a meta de 1% de economia, como mostrado no Quadro 5.

Quadro 5 – Valores para Proprietários Comuns.

Média de Km por ano	**Gasto em 1 ano**	**Gastos com Gasolina a R$ 4,00* em 1 ano**	**Economia de 1% em 1 ano**
10000 Km*	862 Litros de Gasolina*	R$ 3448,00*	**R$ 34,00***

Fonte Quadro: Próprio Autor

O veículo não é utilizado somente para lazer ou para se locomover até o trabalho, temos também a opção do veículo se tornar uma fonte de renda, como por exemplos motoristas de táxis e motoristas de aplicativos. Esse tipo de profissional utiliza mais intensamente o veículo ao decorrer do mês e por isso acaba tendo um gasto maior com combustível. No Quadro 6, temos uma estimativa de gastos em um ano levando em consideração a média de mil quilômetros rodados por semana e o consumo do nosso veículo de teste e a estimativa de 1% de economia.

Quadro 6 - Valores para Táxi e Motorista de Aplicativo.

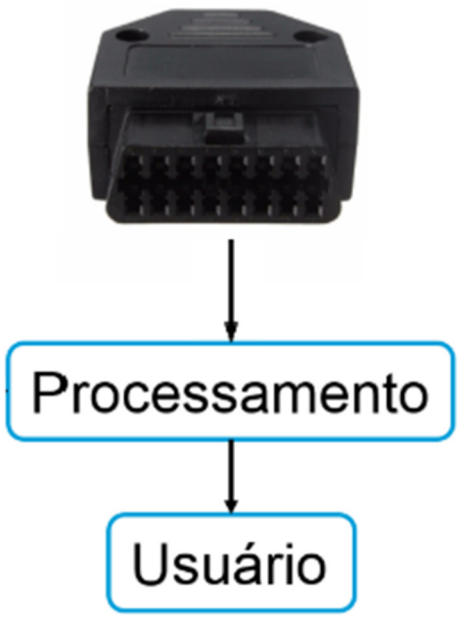
Média de Km por ano	**Gasto em 1 ano**	**Gastos com Gasolina a R$ 4,00* em 1 ano**	**Economia de 1% em 1 ano**
52.000 Km*	4.482 Litros de Gasolina*	R$ 17.931,00*	**R$ 179,31***

Fonte Quadro: Próprio Autor

3. METODOLOGIA

O sistema desenvolvido realiza a leitura dos dados do veículo através da interface de diagnóstico do veículo (OBD II), onde estarão disponibilizadas através da rede CAN do veículo. Posteriormente, as informações serão processadas para avaliar os dados obtidos. A Figura 9, exemplifica esse processo.

Figura 9 - Diagrama de Funcionamento.



Fonte Imagem: Próprio Autor.

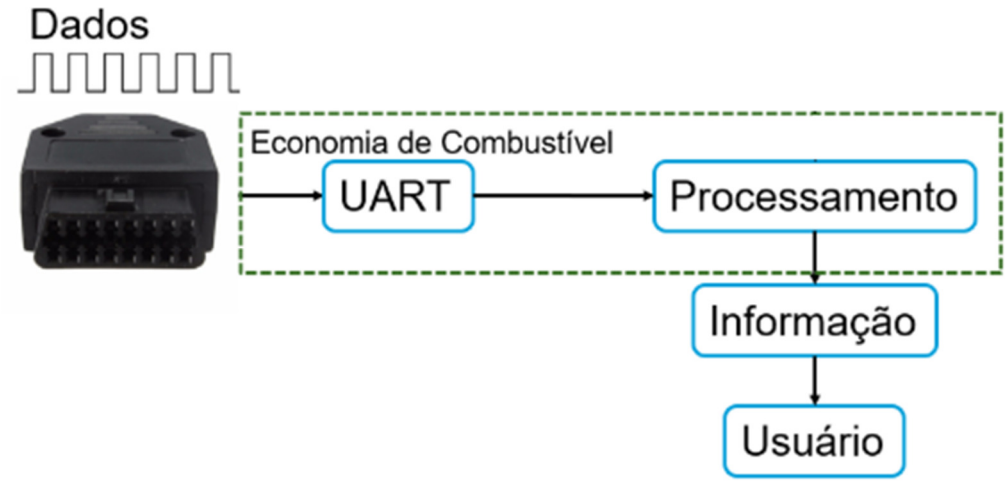
3.1. Impacto no consumo de combustível

Através dos dados que serão coletados na rede CAN do veículo via conector OBD II será possível realizar uma análise e sugerir algumas ações ao motorista. Essas ações podem impactar diretamente no consumo de combustível do veículo e em determinadas situações ter diminuição significativa no consumo de combustível.

Por menor que seja a economia gerada já é de grande benefício, ainda mais se levarmos em conta empresas que possuam grandes frotas de veículos, a soma de pequenos resultados pode gerar grandes valores para toda a frota.

Através da redução do consumo, por consequência também iremos diminuir a emissão de gases poluentes através do sistema de exaustão do veículo. Isso impacta diretamente no meio ambiente melhorando a qualidade do ar. A figura 10 exemplifica o impacto na economia de combustível.

Figura 10 - Impacto Consumo Combustível.



Fonte Imagem: Próprio Autor.

3.2. Como será analisado

No manual do veículo, a fabricante disponibiliza recomendações que podem proporcionar ao motorista uma maior economia de combustível. A figura 11 mostra uma página do manual do veículo.

Figura 11 - Velocidades sugeridas para economia de combustível.

VELOCIDADES SUGERIDAS PARA ECONOMIA DE COMBUSTÍVEL

As seguintes velocidades para a troca de marcha são recomendadas pela Nissan para aumentar a economia de combustível.

	Troca de marcha	Fase fria (Até 8 min) (km/h)	Fase quente (Após 8 min) (km/h)
Motor 1.6 l	1ª para 2ª	15	15
	2ª para 3ª	32	28
	3ª para 4ª	45	42
	4ª para 5ª	55	50

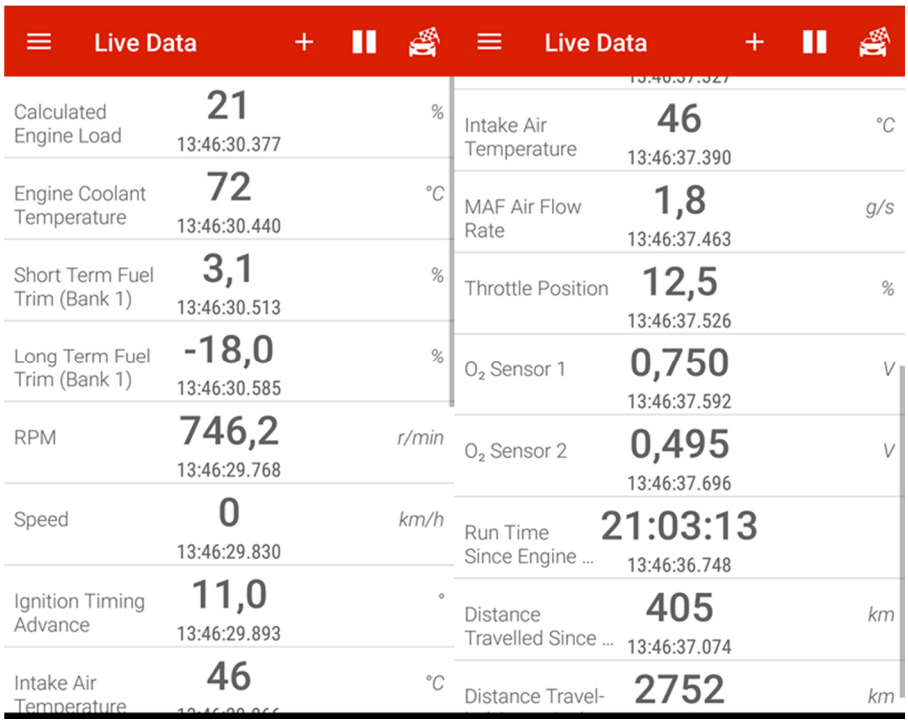
Fonte Imagem: Manual do Veículo March da Nissan.

Serão coletados dados do veículo, enviados para unidade de processamento e baseado nas recomendações do fabricante, iremos realizar sugestões no modo de condução para proporcionar uma economia de combustível.

3.3. Etapas

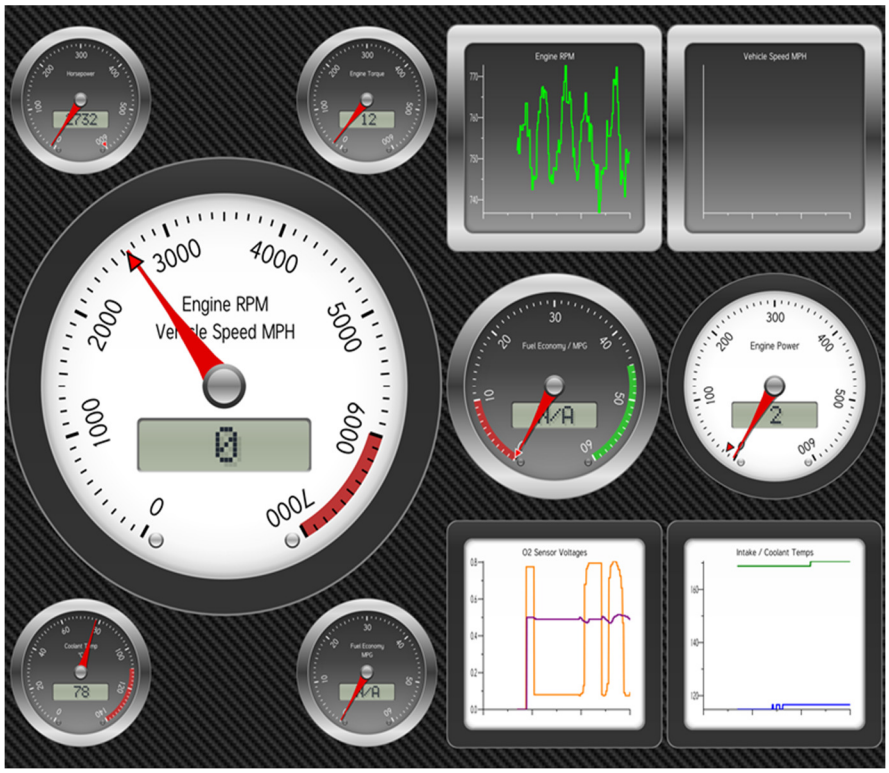
Como primeiro ponto de partida utilizou-se um dispositivo ELM327 com conexão *Bluetooth*, para testar a comunicação do veículo e o funcionamento do conector. A partir de um smartphone, instalamos dois aplicativos de OBD II, *Piston* e *DashCommand,* realizou-se comunicação com veículo, e a verificação de alguns dados que são possíveis de se obter. Ambos os aplicativos estão disponibilizados na loja de aplicativos oficial do *Android*, a *Google Play*. A figura 12 mostra o aplicativo *Piston* em execução e a figura 13 mostra o aplicativo *DashCommand* em execução.

Figura 12 - Aplicativo *Piston* em execução.



Fonte: Próprio Autor

Figura 13 - Aplicativo *DashCommand* em execução.



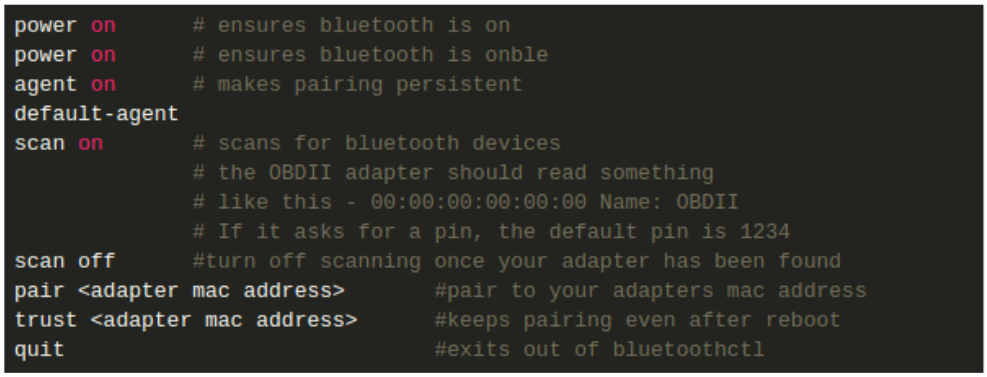
Fonte da Imagem: Próprio Autor

Com a confirmação do funcionamento do conector e da comunicação do veículo, iniciou-se a próxima etapa dos testes com o hardware, *Raspberry.*

3.3.1. Configurando conexão

Nesse momento realizou-se a conexão com o conector OBD II, com base no site "*hackster.io",* o autor publicou um projeto utilizando uma *Raspberry* e um conector OBD II, segundo a matéria, primeiramente era necessário executar alguns comandos de configuração do *Bluetooth*, demonstrado na figura 14.

Figura 14 - Comandos Bluetooth.



Fonte da Imagem: https://www.hackster.io/tinkernut/raspberry-pi-smart-car-8641ca

Esses comandos são responsáveis por realizar a conexão com o conector OBD II, parear e marcar como sendo um dispositivo confiável. Isso é necessário para garantir que o conector OBD II possa conectar automaticamente na próxima vez que o sistema da *Raspberry* reiniciar.

Após esse processo ainda foi preciso realizar uma configuração fundamental para a comunicação, que consiste em modificar a conexão *Bluetooth* para funcionar como sendo uma porta serial, como na figura 15.

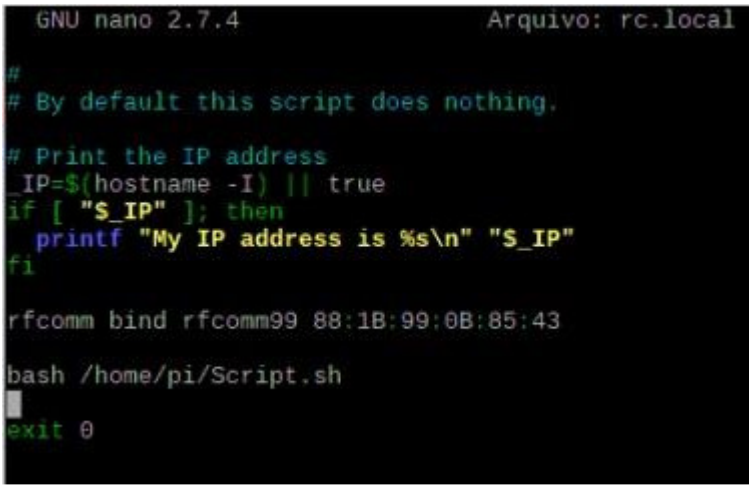
Figura 15 - Comando de configuração serial.

Fonte da Imagem: https://www.hackster.io/tinkernut/raspberry-pi-smart-car-8641ca

Nessa etapa foi preciso que a configuração da porta serial ocorre automaticamente sempre que o sistema inicializar. Na documentação do *Raspberry* é possível encontrar uma recomendação para definir a melhor maneira para programas ou comandos serem executados após a inicialização.

A recomendação cita o arquivo “rc.local”, localizado no diretório “/etc” do sistema operacional, como mostrado na figura 16. Dentro desse arquivo foi inserido o comando acima, além da chamada de script, onde através dele é realizado a execução do programa. Na figura 17 mostra o script com a chamada para executar o programa.

Figura 16 - Arquivo rc.local requisitando script.



Fonte da Imagem: Próprio autor

Figura 17 - Script com a chamada do software.

Fonte da Imagem: Próprio autor

3.3.2. Integração com o software

Foi utilizado uma biblioteca implementada em Python chamada “*python-OBD*” para o desenvolvimento do software. A biblioteca tem a função de realizar a leitura de dados em tempo real de sensores, atuadores, etc. A figura 18 mostra um exemplo de uso da biblioteca.

Figura 18 - Exemplo de código da biblioteca da linguagem *Python*.

```
import obd

connection = obd.OBD() # auto-connects to USB or RF port

cmd = obd.commands.SPEED # select an OBD command (sensor)

response = connection.query(cmd) # send the command, and parse the response

print(response.value) # returns unit-bearing values thanks to Pint
print(response.value.to("mph")) # user-friendly unit conversions
```

Fonte da Imagem: https://python-obd.readthedocs.io/en/latest/

Utilizando essa biblioteca, foi possível verificar quais dados são possíveis de coletar, a partir disso foi definido quais serão os parâmetros a serem utilizados.

- RPM
- Velocidade
- Temperatura do líquido de arrefecimento

A biblioteca possibilita a aquisição de vários outros dados. Como por exemplo *Throttle Body Position* (TPS), tempo de motor ligado, temperatura ambiente, entre outros. Porém não foi utilizado dentro do escopo deste projeto. A figura 19 mostra os valores obtidos.

Figura 19 - Exemplo de biblioteca da linguagem *Python*.

```
RPM
0.0 revolutions_per_minute
velocidade
0 kph
temperatura
50 degC
tps
15.2941176471 percent
engine
0.0 percent
intake
39 degC
runtime
49 second
relative
4.70588235294 percent
amb_temp
20 degC
atuador
0.392156862745 percent
```

Fonte da Imagem: Próprio Autor

Após a requisição desses dados, foi preciso processá-los e gravá-los na unidade de processamento. Será anexado a este documento o código desenvolvido durante o projeto.

3.3.3. Comunicação com o motorista

Para informar sobre a situação real do modo de condução, foi utilizado um sistema de 3 LEDs, com cores amarelo, vermelho e verde, como mostrado na figura 20. Essas cores são de fácil diferenciação para qualquer pessoa e de fácil interpretação para o motorista, uma vez que são as mesmas cores utilizados em semáforos.

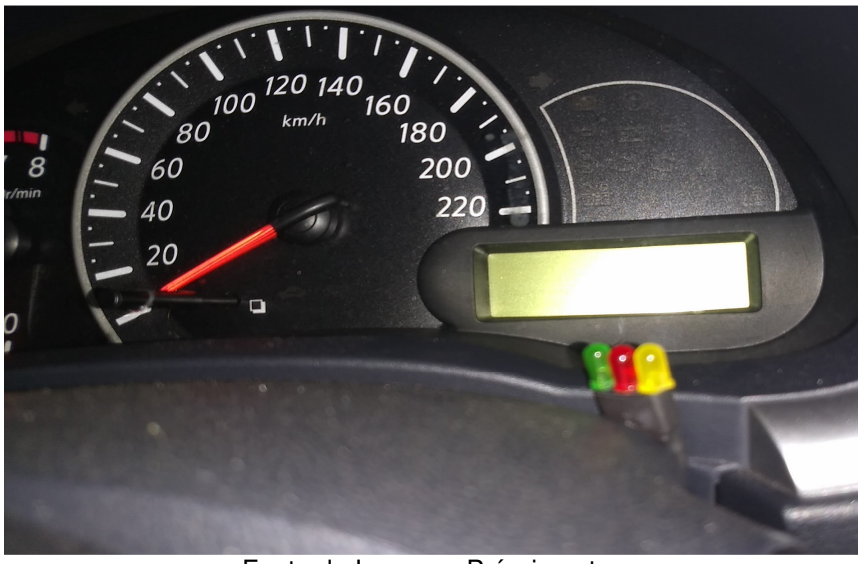
Figura 20 - LEDs de sinalização.

Fonte da Imagem: Próprio autor

Ficou definido que o LED Verde representa a faixa de rotação ideal para determinada velocidade, ou seja, está economizando combustível. O LED Amarelo, funciona como um alerta, que indica que está numa região intermediária, onde não está com consumo excessivo, porém não está na faixa de rotação de baixo consumo. O LED Vermelho representa o consumo excessivo e desnecessário de combustível, uma vez que não está sendo respeitado a recomendação do fabricante do veículo.

Para evitar de ocorrer qualquer tipo de distração os LEDs foram instalados próximo ao campo de visão do motorista. A figura 21 mostra o posicionamento dos LEDs em relação ao painel do veículo.

Figura 21 - Posicionamento dos LEDs.



Fonte da Imagem: Próprio autor

Na figura 22 é demonstrado a *Raspberry* na parte da frente do console, perto da uma fonte de energia para poder ligá-la utilizando um adaptador com saída USB.

Figura 22 - Posicionamento *Raspberry*.

Fonte da Imagem: Próprio autor

Na porta de comunicação OBD II do veículo é conectado o dispositivo ELM327, responsável por coletar os dados através da rede CAN. Como mostrado na figura 23.

Figura 23 - Conector OBD II

Fonte da Imagem: Próprio autor

3.3.4. Estratégia de recomendação

Com base nas recomendações que a fabricante do veículo disponibiliza, registramos a rotação do motor na velocidade definida em cada marcha. O projeto toma como referência principal, a rotação do motor e a velocidade. Como mostrado no quadro 8.

Quadro 7 – Velocidade e RPM para realizar a troca de marcha.

	Fase Fria		Fase Quente	
Troca de Marcha	Km/h	RPM	Km/h	RPM
1° para 2°	15	2500	15	2500
2° para 3°	32	2500	28	2200
3° para 4°	45	2400	42	2300
4° para 5°	55	2200	50	2000

Fonte Quadro: Próprio Autor

Com isso foi criado a estratégia de recomendação para o motorista utilizando os LEDs como sinalização. Como mostrado no quadro 9 e quadro 10.

Quadro 8 – Estratégia de LEDs para recomendar a troca de marcha na fase fria.

Fase Fria	LED Verde	LED Amarelo	LED Vermelho
Km/h	Faixa de RPM	Faixa de RPM	Faixa de RPM
15	Marcha lenta até 2500	2500 a 3500	Acima de 3500
32	Marcha lenta até 2500	2500 a 3500	Acima de 3500
45	Marcha lenta até 2400	2400 a 3400	Acima de 3400
55	Marcha lenta até 2200	2200 a 3200	Acima de 3200

Fonte Quadro: Próprio Autor

Quadro 9 – Estratégia de LEDs para recomendar a troca de marcha na fase quente.

Fase Quente	LED Verde	LED Amarelo	LED Vermelho
Km/h	Faixa de RPM	Faixa de RPM	Faixa de RPM
15	Marcha lenta até 2500	2500 a 3500	Acima de 3500

32	Marcha lenta até 2200	2200 a 3200	Acima de 3200
45	Marcha lenta até 2300	2300 a 3300	Acima de 3300
55	Marcha lenta até 2000	2000 a 3000	Acima de 3000

Fonte Quadro: Próprio Autor

O LED Verde permanece acesa na faixa de rotação recomendada pelo fabricante de acordo com a velocidade. Foi definido um intervalo acima desse valor de 1000 RPM como sendo a faixa de LED Amarelo. E acima desse intervalo permanece acesso o LED Vermelho.

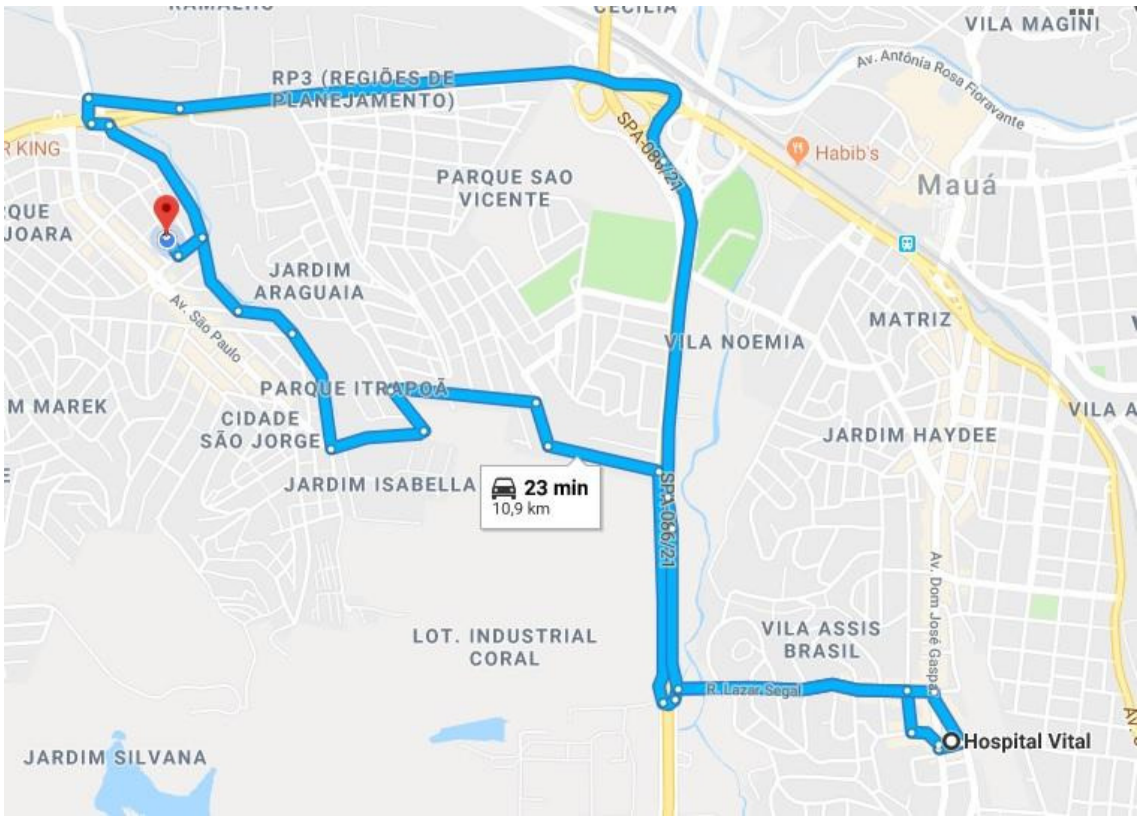
Além disso foi acrescentado uma condição onde caso a velocidade seja igual a zero, ou seja, o veículo esteja parado, o LED Amarelo permanece acesa. Caso o motorista aumente a rotação sem haver aumento de velocidade o LED Vermelho se acende.

Em uma situação ideal, para máxima economia de combustível, o correto seria o motor se desligar quando estivesse parado. Esta é a função do sistema conhecido como Start-Stop. Por isso foi classificado como LED Amarelo, a situação em que temos rotação de marcha-lenta e velocidade zero. E no outro caso em que ocorre o aumento da rotação sem velocidade, existe gasto de combustível desnecessário, por isso é classificado como LED Vermelho.

3.3.5. Trajeto definido para testes

Após a montagem do sistema, foram realizados testes de campo empregando a solução. E para isso foram coletados valores em um trajeto pré-definido, que ficou determinado como o trajeto padrão para os testes, como mostrado na figura 24. O trajeto possui 10,9 Km de percurso, e foi realizado durante um horário em que o trânsito não interferisse nas medições.

Figura 24 - Trajeto definido.



Fonte da Imagem: Próprio autor

Neste trajeto existe diversas situações que exigem o motorista a trocar de marcha para manter a rotação adequada de acordo com o sistema, como um aclive com o porcentual alto no trecho mostrado na figura 25.

Figura 25 – Aclive.

Fonte da Imagem: Próprio autor

Na figura 26 mostra um trecho que também exige muita troca de marcha devido a quantidade de lombadas.

Figura 26 – Lombadas.

Fonte da Imagem: Próprio autor

4. RESULTADOS OBTIDOS

Antes de iniciar os testes foi escolhido o tipo de combustível que seria utilizado no veículo. Sendo assim foi mantido o mesmo que estava sendo usado no veículo, que é a gasolina comum. Após a escolha do combustível, foi possível começar os testes de rodagem.

Com o veículo parado foi resetada todas as informações do computador de bordo do veículo antes do início do trajeto para assegurar precisão nos resultados obtidos. E ao final do trajeto foi confirmado a distância definida para o teste, como mostrado na figura 27.

Figura 27 - Distância percorrida.

Fonte da Imagem: Próprio Autor

O computador de bordo tem a capacidade de mostrar o consumo médio do veículo, onde foi possível realizar o mesmo trajeto duas vezes dirigindo de formas diferentes obtendo o valor de consumo.

Na primeira realização do trajeto, foi decidido dirigir usando as recomendações do quadro mostrado anteriormente para assegurar o melhor consumo possível pelo veículo. Depois de percorrer todo o trajeto, foi observado no computador de bordo que o consumo médio foi de 12,8 Km/h, como mostrado na figura 28. Sendo o percurso executado em 25 minutos e 3 segundos.

Figura 28 – Consumo da primeira viagem.

Fonte da Imagem: Próprio autor

Na segunda realização do trajeto, com um novo reset do computador de bordo, foi decidido dirigir de modo a fazer o percurso em menos tempo, e para conseguir esse ganho em tempo foi necessário dirigir de forma agressiva. Depois de percorrer todo o trajeto, foi observado no computador de bordo que o consumo médio foi de 7,3 Km/h como mostrado na figura 29. Sendo o percurso executado em 20 minutos e 49 segundos.

Figura 29 – Consumo da segunda viagem.

Fonte da Imagem: Próprio autor

Com estas três informações coletadas, é possível realizar o cálculo pela diferença de consumo descobrindo o quanto o motorista consegue economizar ou gastar um pouco mais para ganhar tempo. E com uma estimativa média do preço do combustível a R$ 4 reais o litro consultado em 10 de dezembro de 2018, que foi utilizado como base para elaboração do quadro 11. É visível o quanto pode ser economizado em um trajeto.

Quadro 10 – Análise de dados.

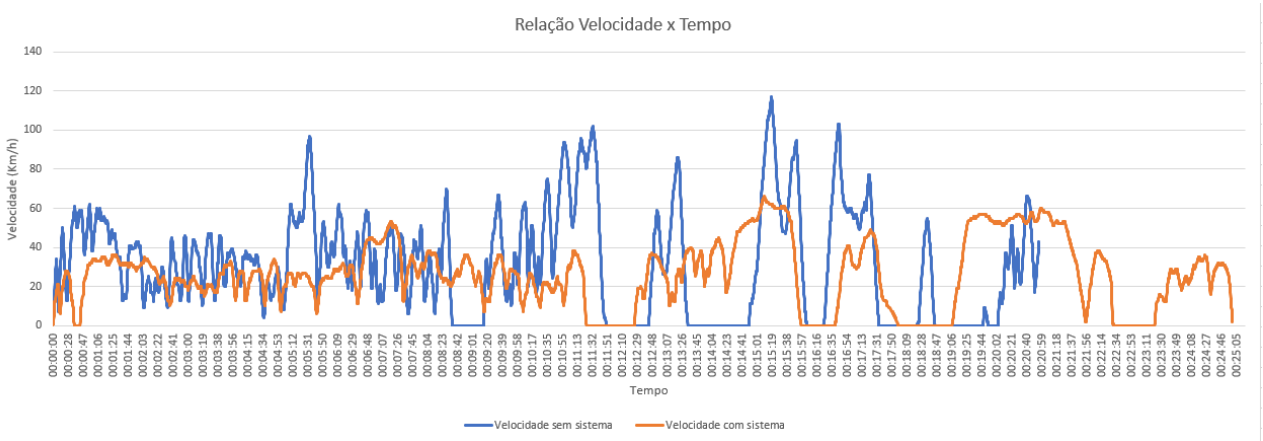
-	Km rodados	Km/l	Litros total	Preço R$ (Reais)	Tempo de trajeto (min:seg)
1º Viagem	11	12,8	0,85	3,4	20:49
2º Viagem	11	7,3	1,5	6	25:09

Fonte Quadro: Próprio Autor

Além dos LEDs indicando para o motorista o modo de condução ideal, o sistema registra os valores de rotação do motor, velocidade e temperatura do motor. A partir dos dados coletados pelo projeto foi possível realizar análises via gráficos para poder comparar as informações das duas viagens no mesmo trajeto.

Na comparação da velocidade pode se observar que na viagem sem a sinalização dos LEDs, houve uma grande oscilação com diversos picos excessivos de velocidade. Chegando a ter como velocidade máxima de 117 Km/h, sendo que logo em seguida temos uma queda. Em um trajeto urbano dificilmente atingimos uma velocidade tão significativa, porém fizemos nosso percurso na intenção de baixar o tempo, e buscar o extremo, por isso temos

Figura 30 - Gráfico de velocidade.

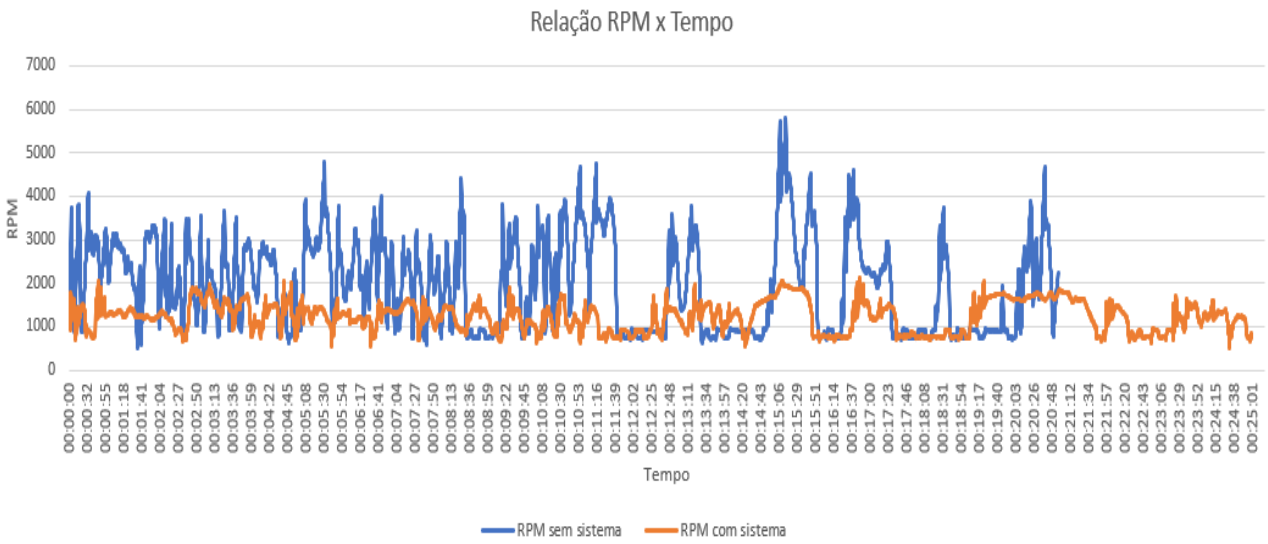


Fonte da Imagem: Próprio autor

Na comparação levando em consideração a rotação, é possível observar que na viagem onde não utilizamos a indicação ao motorista, temos altas rotações em

determinados instantes. Já na viagem utilizando o sistema, em nenhum momento ultrapassa 2500 RPM.

Figura 31 - Gráfico de RPM.



Fonte da Imagem: Próprio autor

Na tabela a seguir mostra alguns dados complementares, como velocidade máxima, velocidade média, rotação máxima e rotação média. Uma grande surpresa foi em relação a velocidade média que não houve uma grande diferença nas duas viagens realizadas. Isso prova que houve acelerações e frenagens desnecessárias na viagem onde não foi utilizado o sistema, sendo que a velocidade máxima do trajeto com sistema e do trajeto sem o sistema obteve uma grande diferença.

Em relação a rotação ocorreu algo bem semelhante, a rotação média não obteve tanta diferença, em compensação a rotação máxima obteve grande diferença.

Quadro 11 – Velocidade máxima e média das viagens.

Velocidade Máxima (Km/h)		Velocidade Média (Km/h)	
Com sistema	Sem sistema	Com sistema	Sem sistema
66	117	26	31

Fonte Quadro: Próprio Autor

Quadro 12 – RPM máxima e média das viagens.

Rotação Máxima (RPM)		Rotação Média (RPM)	
Com sistema	Sem sistema	Com sistema	Sem sistema
2137	5815	1232	1973

Fonte Quadro: Próprio Autor

5. CONCLUSÃO

Nesse projeto foi apresentado um sistema de modo de condução visando economia de combustível, usando uma unidade de processamento para coletar os dados através da rede de comunicação do veículo e indicação para o motorista utilizando LEDs. Além de deixar registrado informações como RPM, Velocidade e Temperatura do Motor.

A principal contribuição desse projeto foi a implementação de uma unidade de processamento com possibilidade de comunicar-se com a rede de comunicação do veículo. Isso serve como base para diversos outros projetos e ideias que possam surgir no futuro, e que de alguma forma precisam utilizar dados que estão disponíveis através da rede de comunicação e precisam ser processados para atingir um determinado objetivo.

Além disso através dos testes realizados, foi observado o grande impacto no consumo de combustível do veículo. Seguindo dicas do fabricante o consumo passou de 7,3 km/l para 12,8 Km/l, o que é um valor considerável.

Muitas vezes os motoristas por diversos motivos não seguem as recomendações dos fabricantes para obter um melhor consumo de combustível. E através do projeto, utilizando as indicações do modo de condução é possível ensinar ao motorista a obter um melhor consumo.

Consequentemente o impacto influencia diretamente no valor gasto com combustíveis no decorrer do uso. Além disso o impacto de gases nocivos ao meio ambiente também diminui com um menor consumo de combustível.

Foi possível concluir que a longo prazo, a economia gerada pelo processo de economia de combustível é um valor considerável, onde em pouco tempo cobre o valor da implementação do projeto, além de trazer benefícios para o meio ambiente. E em muitos casos gera educação ao motorista a ter um melhor modo de condução, podendo influenciar em um trânsito mais gentil.

6. PROPOSTAS FUTURAS

- Implementação de um hardware próprio;
- Criação de uma biblioteca Python para utilização futuras;
- Processamento das informações aplicando aprendizado de máquina (Inteligência Artificial);
- Envio dos dados para a nuvem (AWS);
- Analisar emissões com a utilização do sistema e sem a utilização do sistema;
- Integração com sistema GPS.

7. REFERÊNCIAS BIBLIOGRÁFICAS

António Sérgio Leite Machado e Bruno Rafael Resende Oliveira. O sistema OBD. Disponível em: <http://ave.dee.isep.ipp.pt/~mjf/act_lect/SIAUT/Trabalhos%202007-08/Trabalhos/SIAUT_OBD.pdf> Acesso em: 2 nov. 2018.

Bluetooth connection. Disponível em: <https://www.raspberrypi.org/forums/viewtopic.php?t=148454> Acesso em: 2 nov. 2018.

BROMBACHER, Patrick. **"Driving Event Detection and Driving Style Classification using Artificial Neural Networks"**. 2017, Tese de mestrado no Instituto de Tecnologia de Sistemas de Veículos, Instituto de Tecnologia de Karlsruhe, Alemanha.

ECONOMATICA. 2018. Disponível em: <https://financeone.com.br/marcas-mais-valiosas-do-mundo-2018> Acesso em: 2 nov. 2018.

Eduardo Gonçalves. O que é Raspberry Pi 3. Disponível em: <https://www.mundotibrasil.com.br/o-que-e-o-raspberry-pi-3-rpi3/> Acesso em: 2 nov. 2018.

Elm Electronics Inc. OBD. Disponível em: <https://www.elmelectronics.com/products/ics/obd/>. Acesso em: 2 nov. 2018.

Ficha Técnica Nissan March. Disponível em: <https://www.carrosnaweb.com.br/fichadetalhe.asp?codigo=4775> Acesso em 2 nov. 2018.

Grupo JSL. Disponível em: <https://pt.wikipedia.org/wiki/Grupo_JSL > Acesso em: 2 nov. 2018.

INMETRO. Quadros PBE Veicular – INMETRO. Disponível em: <http://www.inmetro.gov.br/consumidor/tabelas_pbe_veicular.asp> Acesso em: 2 nov. 2018.

J. Massamy Taniguchi. REDE CAN o que é e como funciona. Disponível em: <http://dicas.webautomotivo.com.br/rede-can-o-que-e-e-como-funciona/> Acesso em: 2 nov. 2018.

Kleber Nogueira Hodel. Material Fatec Santo André - Aula de Rede de Comunicação, Professor Kleber.

OBD. Disponível em: <https://en.wikipedia.org/wiki/On-board_diagnostics>. Acesso em: 2 nov. 2018.

OICA. Organização Internacional de Fabricantes de Veículos Automotores. Disponível em: <http://www.oica.net/category/production-statistics/2017-statistics> Acesso em: 2 nov. 2018.

Orlando de Salvo Junior. Material Fatec Santo André – Aula de Diagnose, Professor Orlando.

PORTO SEGURO. 2018. Disponível em: <https://www.portoseguro.com.br/seguro-auto/seguro-auto-jovem> Acesso em: 2 nov. 2018.

PORTO SEGURO. 2018. Disponível em: <https://www.portoseguro.com.br/beneficios/7-de-desconto-para-motorista-sem-pontos-na-cnh> Acesso em 2 nov. 2018.

Python-OBD. Disponível em: <https://python-obd.readthedocs.io/en/latest/> Acesso em: 2 nov. 2018

Raspberry Pi Smart Car. Disponível em: <https://www.hackster.io/tinkernut/raspberry-pi-smart-car-8641ca> Acesso em: 2 nov. 2018.

TRÂNSITO MAIS GENTIL. 2018. Disponível em: <https://www.transitomaisgentil.com.br/o-app/> Acesso em 2 nov. 2018.

V. Corcoba Magaña ; M. Muñoz-Organero. **“Artemisa: An eco-driving assistant for Android Os”**. 2011, Berlin, Alemanha.

Wei-Yao Chou; Yi-Chun Lin; Yu-Hui Lin; Syuan-Yi Chen. **“Intelligent eco-driving suggestion system based on vehicle loading model”**. 2012, Taipei, Taiwan.

APÊNDICE A: Programação

```
# -*- coding: utf-8 -*-

import datetime
import time
import obd
import RPi.GPIO as GPIO

Verde = 16
Vermelho = 20
Amarelo = 21

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(Amarelo,GPIO.OUT)
GPIO.setup(Verde,GPIO.OUT)
GPIO.setup(Vermelho,GPIO.OUT)


GPIO.output(Amarelo,GPIO.LOW) #Desligado
GPIO.output(Verde,GPIO.LOW) #Desligado
GPIO.output(Vermelho,GPIO.LOW) #Desligado




numconn = 0
while (numconn < 10):
    try:
        connection = obd.OBD("/dev/rfcomm0")

        GPIO.output(Amarelo,GPIO.HIGH)
        GPIO.output(Verde,GPIO.HIGH)
        GPIO.output(Vermelho,GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.LOW)

        conn = True
        break
    except:
        numconn = numconn + 1
        conn = False
        GPIO.output(Vermelho,GPIO.HIGH)
        time.sleep(0.3)
        GPIO.output(Vermelho,GPIO.LOW)
```

```
local = "/home/pi/Desktop/Projeto/dados/Indice.txt"
with open(local,'r') as arquivo:
    indicearquivo = int(arquivo.readline())

    indicearquivo += 1
with open(local,'w') as arquivo:
    arquivo.write(str(indicearquivo))


while(conn):
    rpm = round(float(str(connection.query(obd.commands.RPM)).split(" ")[0]))
    speed = round(float(str(connection.query(obd.commands.SPEED)).split(" ")[0]))
    temp = str(connection.query(obd.commands.COOLANT_TEMP)).split(" ")[0]
    tps =
str(round(float(str(connection.query(obd.commands.THROTTLE_POS)).split(" ")[0])))
    intake_temp = str(connection.query(obd.commands.INTAKE_TEMP)).split(" ")[0]
    run_time = str(connection.query(obd.commands.RUN_TIME)).split(" ")[0]
    amb_temp = str(connection.query(obd.commands.AMBIANT_AIR_TEMP)).split("
")[0]

    if(rpm > 0):

        '''if(temp > 80):
            TempMotor = "Quente"
        else:
            TempMotor = "Frio"""

        if (speed < 3):
            if(rpm < 1500):
                GPIO.output(Amarelo,GPIO.HIGH)
                GPIO.output(Verde,GPIO.LOW)
                GPIO.output(Vermelho,GPIO.LOW)
            else:
                GPIO.output(Amarelo,GPIO.LOW)
                GPIO.output(Verde,GPIO.LOW)
                GPIO.output(Vermelho,GPIO.HIGH)

        elif (speed <= 15):
            if(rpm < 2500):   #rotação de economia de combustivél
                GPIO.output(Amarelo,GPIO.LOW)
                GPIO.output(Verde,GPIO.HIGH)
                GPIO.output(Vermelho,GPIO.LOW)
            elif(rpm < 3500):   #rotação de torque maximo
                GPIO.output(Amarelo,GPIO.HIGH)
                GPIO.output(Verde,GPIO.LOW)
                GPIO.output(Vermelho,GPIO.LOW)
            else:
                GPIO.output(Amarelo,GPIO.LOW)
                GPIO.output(Verde,GPIO.LOW)
```

```
        GPIO.output(Vermelho,GPIO.HIGH)

elif (speed <= 28):
    if(rpm < 2200):   #rotação de economia de combustivél
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.HIGH)
        GPIO.output(Vermelho,GPIO.LOW)
    elif(rpm < 3200):   #rotação de torque maximo
        GPIO.output(Amarelo,GPIO.HIGH)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.LOW)
    else:
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.HIGH)

elif (speed <= 42):
    if(rpm < 2300):   #rotação de economia de combustivél
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.HIGH)
        GPIO.output(Vermelho,GPIO.LOW)
    elif(rpm < 3300):   #rotação de torque maximo
        GPIO.output(Amarelo,GPIO.HIGH)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.LOW)
    else:
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.HIGH)

elif (speed <= 50):
    if(rpm < 2000):   #rotação de economia de combustivél
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.HIGH)
        GPIO.output(Vermelho,GPIO.LOW)
    elif(rpm < 3000):   #rotação de torque maximo
        GPIO.output(Amarelo,GPIO.HIGH)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.LOW)
    else:
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.HIGH)
elif(speed > 50):
    if(rpm < 2000):   #rotação de economia de combustivél
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.HIGH)
        GPIO.output(Vermelho,GPIO.LOW)
    elif(rpm < 4000):   #rotação de torque maximo
        GPIO.output(Amarelo,GPIO.HIGH)
```

```
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.LOW)
    else:
        GPIO.output(Amarelo,GPIO.LOW)
        GPIO.output(Verde,GPIO.LOW)
        GPIO.output(Vermelho,GPIO.HIGH)

timenow = datetime.datetime.now()
local = "/home/pi/Desktop/Projeto/dados/Dados-" + str(indicearquivo) + ".txt"
arquivo = open(local,'a')

arquivo.write(timenow.strftime("%Y-%m-%d %H:%M:%S:%f")[:-3])
arquivo.write(",")
arquivo.write(str(rpm))
arquivo.write(",")
arquivo.write(str(speed))
arquivo.write(",")
arquivo.write(temp)
arquivo.write(",")
arquivo.write(tps)
arquivo.write(",")
arquivo.write(intake_temp)
arquivo.write(",")
arquivo.write(run_time)
arquivo.write(",")
arquivo.write(amb_temp)
arquivo.write("\n")
arquivo.close
```