UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE BACHARELADO EM ENGENHARIA ELÉTRICA

CÉSAR AUGUSTO FÜHR CARÁ

APLICAÇÃO DE PROTOCOLO IOT PARA MONITORAMENTO VEICULAR

São Leopoldo
2021

César Augusto Führ Cará

APLICAÇÃO DE PROTOCOLO IOT PARA MONITORAMENTO VEICULAR

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Engenharia Elétrica pela Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Me. Armando Leopoldo Keller

São Leopoldo
2021

RESUMO

O objetivo deste trabalho é explorar os conceitos e aplicabilidade da Internet das Coisas (IoT) e seus principais protocolos, no processo de extração e publicação de dados provenientes dos sistemas de diagnóstico embarcados nos automóveis modernos. Dessa forma é apresentado uma revisão bibliográfica dos assuntos, explorando os temas relacionados ao automóvel moderno, motor a combustão e suas características, centrais de controle, comunicação e seus protocolos. De forma complementar os temas relacionados a IoT, sua caracterização e definição, principais protocolos de comunicação e suas aplicações também são abordados. Diante da limitação em relação a localização do usuário para o acesso a informações de diagnóstico veicular, o trabalho se propõe a desenvolver um dispositivo que atue como *gateway* para a conexão de veículos a um sistema centralizador de dados capaz de gerar ao usuário uma experiência de monitoria. O sistema *gateway* é responsável por realizar uma tradução do padrão OBDII (*On-Board Diagnostic II*) para o protocolo MQTT (*Message Queuing Telemetry Transport*) e publicar os dados através da Internet. Testes com veículos reais foram realizados no intuito de determinar a compatibilidade, viabilidade e performance do sistema proposto realizando medidas de tempo de processamento de mensagens. Estes testes, após uma análise dos resultados, confirmam o funcionamento correto do sistema proposto no trabalho, que processou 99,06% das mensagens em até 800 milissegundos.

Palavras-chaves: Internet das Coisas (IoT). MQTT. OBDII. Monitoria. Diagnóstico. Automotivo.

ABSTRACT

The objective of this work is to explore Internet of Things (IoT) concepts, applicability and its main protocols, in the process of extraction and publication of data originated from modern vehicle on-board diagnose systems. A bibliographic review about the subject is presented, exploring themes related to the modern car, combustion engine and its characteristics, control units, communication and protocols. Other themes related to IoT, its characterization and definition, main communication protocols and usage are also explored. There is a limitation in relation to the user location in order to have access to a vehicle diagnostic information. With this problem in mind, the work is proposing a system that acts as a gateway to connect vehicles to a central system capable of giving the user a monitoring experience. The gateway system is responsible for translating the messages in the OBDII (On-Board Diagnostic II) format to a MQTT (Message Queuing Telemetry Transport) protocol format and publish it in the Internet. Tests with real vehicles were made with the intent of ensuring compatibility, viability and performance of the proposed system while collecting measurements of message processing time. This tests, after the data was analysed, confirmed that the proposed system has a correct operation, processing 99.06% of the messages in 800 milliseconds.

Key-words: Internet of Things (IoT). MQTT. OBDII. Monitoring. Diagnosis. Automotive.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABS	Antilock Brake System (Sistema de Freio Anti Travamento)
ACK	Acknowledgement (Tomada de Conhecimento)
CAN	Controller Area Network (Rede do Controlador de Área)
CEP	Complex Event Processing (Processamento de Eventos Complexos)
CARB	California Air Resources Board (Comitê de Recursos Aéreos da California)
CoAP	Constrained Application Protocol (Protocolo de Aplicações Limitadas)
CRC	Cyclic Redundancy Code (Código de Redundância Cíclica)
CRUD	Create Read Update Delete (Criar Ler Atualizar Deletar)
CSV	Comma Separated Values (Valores Separados por Vírgula)
DTC	Diagnostic Trouble Codes (Códigos de Diagnóstico de Falhas)
ECU	Electronic Control Unit (Unidade de Controle Eletrônica)
EPA	Environmental Protection Agency (Agencia de Proteção Ambiental)
FD	Flexible Data-rate (Taxa de Dados Flexível)
GPS	Global Positioning System (Sistema de Posicionamento Global)
GSM	Global System for Mobile Communications (Sistema Global de Comunicações Móveis)
HMI	Human Machine Interface (Interface Homem Máquina)
HTTP	Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto)
IEEE	Institute of Electrical and Electronics Engineers (Instituto de Engenharia Eletrônica e Elétrica)
IERC	Internet of Things Research (Pesquisa da Internet das Coisas)
IoT	Internet of Things (Internet das Coisas)
M2M	Machine to Machine (Máquina para Máquina)
MCU	Microcontroller Unit (Unidade Microcontroladora)

MIL	Malfunction Indicator Lamp (Lampada Indicadora de Mal Funcionamento)
MQTT	Message Queuing Telemetry Transport (Transporte de Telemetria por Enfileiramento de Mensagens)
OBDII	On-Board Diagnostic II (Diagnóstico Embarcado II)
PID	Parameter Identifiers (Identificadores de Parâmetros)
QoS	Quality of Service (Qualidade de Serviço)
REST	Representational State Transfer (Transferência de Estado Representativo)
RPM	Revolutions Per Minute (Rotações por Minuto)
RTR	Remote Transmission Request (Requisição de Transmissão Remota)
SAE	Society of Automotive Engineers (Sociedade de Engenheiros Automotivos)
TCP	Transmission Control Protocol (Protocolo de Transmissão de Controle)
UDP	User Datagram Protocol (Protocolo de Dados de Usuário)
USB	Universal Serial Bus (Barramento Universal Serial)
UTF-8	Unicode Transformation Format – 8-bit (Formato de Transformação Unicode - 8-bit)

SUMÁRIO

1 INTRODUÇÃO

Os dispositivos inteligentes ganham cada dia mais espaço nas nossas vidas e rotinas, onde um dos aspectos mais relevantes é a capacidade de se comunicarem e tornar acessíveis informações antes ignoradas ou de difícil acesso. Assim como as nossas casas estão sendo preenchidas com diversos dispositivos que conectam-se e trazem comodidades para o dia a dia, os automóveis também tiveram grande avanço em relação a comunicação dos componentes que atuam na experiência de condução e manutenção do veículo. Grande parte do impacto dessas inovações se dá através da conectividade de máquinas com outras máquinas, também conhecido como *M2M* (do inglês *machine-to-machine*) que aliada ao aumento da disponibilidade de internet acaba gerando um ramo caracterizado por essa união, chamado Internet das Coisas. Com a abundância de informações acerca da saúde do veículo e a permeabilidade das tecnologias IoT (Internet das Coisas, do inglês *Internet of Things*) abre-se uma gama de oportunidades de avanço para áreas como manejo, manutenção preditiva e monitoramento de frotas de veículos.

Dada a grande quantidade de dispositivos e protocolos de comunicação utilizados nessa expansão da Internet das Coisas, é natural que se busque formas para integrar tão variada gama de equipamentos entre si ou ainda com outros sistemas supervisórios. Sendo um dos protocolos mais utilizados em aplicações desse contexto, o MQTT é um protocolo com baixo custo computacional para dispositivos cliente, onde as mensagens trafegam no modelo *publish/subscribe* com o auxílio de um servidor *broker*. Podendo ser facilmente embarcado em um dispositivo de borda por suas características, adere muito a idéia de ser usado como uma forma de transporte de dados de diversos pontos para um sistema central.

Uma forma de permitir a integração entre dispositivos e supervisórios é a utilização de um elemento chamado *gateway* que faz a conversão entre os protocolos. Assim, dados que inicialmente estão confinados em um contexto limitado como a rede interna do veículo, podem ser propagados a outros contextos como um sistema de gerenciamento de frotas que permita o acompanhamento desses dados em tempo real ou alimente bancos de dados para análises para a previsão de possíveis falhas, identificação de maus hábitos de condução ou planejamento de revisões e manutenções dos veículos.

Apesar de restritas à rede interna do veículo, informações como códigos de falha, velocidade, rotações por minuto do motor (rpm), razão entre ar e combustível na combustão (conhecido como *lambda*) e tantas outras que podem ser acessadas através do procolo OBDII (do inglês *On-Board Diagnostic*), protocolo que foi padronizado pela SAE (do inglês *Society of Automotive Engineers*) e é um item obrigatório em qualquer automóvel na atualidade, são importantes para a gerência de frotas e seriam muito úteis em um sistema com esse propósito. Por isso neste trabalho é proposta uma solução que, através de um dispositivo eletrônico de diagnóstico veicular, o

desenvolvimento de *gateway* que atue entre os protocolos *OBDII* e *MQTT* e uma aplicação que centralize os dados, permita o acompanhamento de informações de diagnóstico de automóveis de forma remota através da Internet, dando ao usuário uma visão da saúde e parâmetros do motor de uma frota de veículos.

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo são apresentados os principais referenciais teóricos para um completo entendimento deste trabalho. Os conceitos acerca de automóveis modernos, seus sistemas de controle, propulsão e diagnóstico, Internet das Coisas (IoT) e protocolos de comunicação utilizados para alcançar o objetivo proposto pelo trabalho são abordados e descritos.

2.1 Automóvel Moderno

A indústria automobilística é tradicionalmente forte no âmbito da engenharia mecânica. Em contrapartida é possível verificar nos últimos anos um aumento significativo na utilização da engenharia elétrica para resolver diversas demandas nesse setor. Esse impulso, gerado pela grande queda nos custos para a produção de dispositivos eletrônicos, equipou o carro atual com mais hardware e capacidade de processamento que a espaçonave Apollo 11, que foi até a lua, possuía. Os veículos da atualidade podem ter cerca de até de oitenta controladores conectados em cinco ou mais redes diferentes, um grande número de sensores e atuadores, interfaces HMI (do inglês *Human Machine Interfaces*), além de conexões com telefones móveis e dispositivos de diagnóstico para auxiliar seus passageiros no seu dia-a-dia. (BROY, 2005).

Segundo Broy (2005) veículos contam com o auxílio da combinação de hardware e software para muitas funções e processos que influenciam tanto na condução como na interação com o veículo como um todo. Controle de estabilidade, de velocidade de cruzeiro e inclusive de trajeto, como as tecnologias de *lane keeping assist* e assistência de estacionamento, dependem dos sensores e atuadores presentes no veículo que se comunicam e são controlados por complexos sistemas de software.

Para Tummala *et al.* (2016) os veículos automotores estão se tornando dispositivos eletrônicos, diferente da visão passada que os tinha como dominantemente mecânicos. Atualmente a eletrônica automotiva é responsável por aproximadamente um terço do valor de produção de um veículo e o número total esperado de carros em 2025 é de cem milhões em todo planeta, gerando um mercado imenso. A empresa Tesla, em 2016, anunciou que seus carros seriam completamente autônomos em três anos e os impactos dos avanços na área de autonomia veicular estão gerando grandes saltos de qualidade em funcionalidades relacionadas a assistência a condução.

Podem ser encontradas ECU's (do inglês *Electronic Control Unit*) controlando sistemas essenciais do veículo, como motores, caixas de câmbio automático, sistemas ABS (do inglês *Antilock Brake System*), controle de estabilidade, controle de tração e assistentes de freio enquanto atuadores mais simples são utilizados para funções menos críticas e de conforto. Sensores com eletrônica embarcada de toda sorte são encontrados no motor realizando leituras de pressão, fluxo de massa de ar, aceleração, movimento rotacional e outras tantas grandezas envolvidas na

propulsão do veículo, enquanto atuadores abrem e fecham válvulas, dosam combustível e geram faíscas para um controle completo da ECU sobre o grupo propulsor. (TILS, 2006).

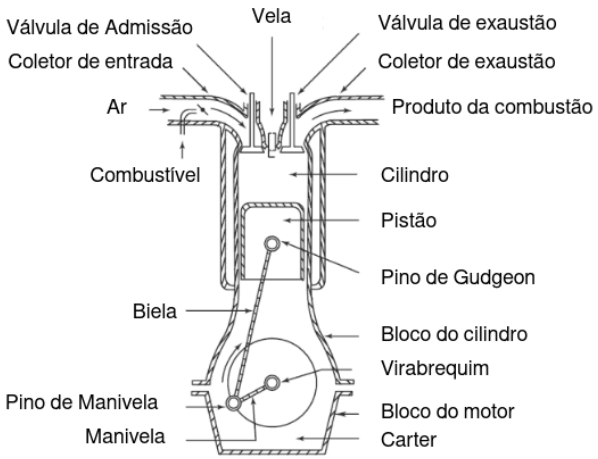
2.1.1 Motor de Combustão Interna

O propósito do motor de combustão interna é a transformação da energia química do combustível em energia mecânica através da oxidação ou queima deste combustível no interior do bloco do motor. É devido sua simplicidade, robustez e alta relação potência/peso que os motores de combustão interna, que realizam a ignição por faiscamento ou compressão, são utilizados predominantemente nos setores de transporte e geração de energia. Dentro deste grupo os motores com ciclo de quatro cursos de pistão são os mais utilizados pela indústria automotiva, um modelo desenvolvido pelo engenheiro alemão Nicolaus August Otto em 1876 que até hoje segue sendo aprimorado. (HEYWOOD, 1988).

Para Ganesan (2012) os motores de combustão interna são máquinas de calor que se dividem em dois grupos principais: os rotacionais e os recíprocos. Dentre estes, os motores recíprocos de quatro cursos de pistão tem a dominância de mercado no espectro automotivo. Estes apresentam vantagens como maior eficiência energética, capacidade de admitir fluidos combustíveis em uma temperatura elevada e simplicidade. As desvantagens mais relevantes do motor recíproco são a vibração gerada pela movimentação recíproca de seus componentes e a pouca variedade de combustíveis utilizáveis, que são relativamente mais caros.

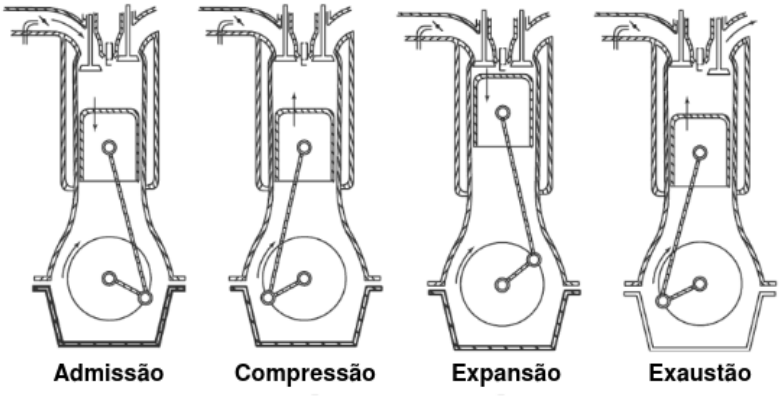
Apesar da simplicidade aparente deste motor, sua constituição é complexa e cada componente precisa executar de forma eficaz sua função para que se possa produzir potência útil. Cada componente e seu nome é demonstrado na Figura 1. Um ciclo completo do cilindro, exemplificado na Figura 2, envolve quatro processos: admissão, compressão, expansão e exaustão, correspondendo a duas revoluções do virabrequim. Dessa sequência de eventos somente a etapa de expansão que gera potência para ser consumida pelo eixo do motor. (HEYWOOD, 1988).

Figura 1 – Corte transversal de um motor a combustão interna recíproco.



Fonte: Adaptado de Ganesan (2012).

Figura 2 – Etapas do ciclo de um cilindro num motor recíproco.



Fonte: Adaptado de Ganesan (2012).

A admissão é a etapa em que a mistura de ar e combustível é admitida na câmara de combustão. Essa mistura pode ser feita na câmara de admissão ou ainda, no caso de injeção direta, dentro do próprio cilindro. Com a válvula de admissão aberta e a válvula de exaustão fechada, a sucção da mistura é feita pelo movimento do pistão, que aumenta o volume da câmara de combustão, reduzindo a pressão interna e admitindo o fluido existente no coletor de entrada. (HEYWOOD, 1988).

A compressão é uma etapa vital para a potencialização da combustão e coloca o pistão em posição para receber a carga de expansão dos gases. Ela acontece com as duas válvulas (admissão e exaustão) fechadas. Ao fim dessa etapa a vela gera uma faísca e a ignição do combustível é iniciada gerando um rápido aumento de pressão na câmara. (HEYWOOD, 1988). Neste momento a temperatura interna da câmara chega até aproximadamente 2000 graus Celsius. (GANESAN, 2012).

Expansão é a etapa em que os gases gerados pela queima do combustível no interior da câmara se expandem aumentando o volume entre a parte superior da câmara e o pistão, empurrando o pistão. O trabalho realizado pelos gases é aproximadamente cinco vezes maior que o realizado pelo pistão na compressão. Esse trabalho é transmitido pelo pistão para o eixo do motor pela biela. Ao fim dessa etapa a válvula de exaustão é aberta para que na próxima etapa o produto da combustão possa ser coletado. (HEYWOOD, 1988).

Inicialmente a exaustão é feita somente pela alta pressão dos gases no interior da câmara, que é consideravelmente maior que a pressão no coletor de exaustão. Em seguida o pistão reduz o volume da câmara forçando o restante dos gases a sair. Ao fim desta etapa a válvula de admissão volta a abrir preparando o sistema para começar mais um ciclo. (HEYWOOD, 1988).

2.1.2 Unidade de Controle do Motor ECU

A ECU é, segundo Sutar e Shinde (2018) , o dispositivo eletrônico responsável pelo monitoramento e controle dos sistemas do veículo. Estes dispositivos usam sistemas embarcados e microcontroladores para o controle das funcionalidades do automóvel e desde o final dos anos 90 as ECU possuem a funcionalidade de diagnóstico embarcado (OBDII).

A unidade de controle do motor pode ser responsável por um ou mais sistemas ou subsistemas do veículo e através de um algoritmo de controle utiliza os dados dos diversos sensores conectados a ela para comandar dispositivos e atuadores buscando uma performance ótima do veículo. As ECU's podem buscar otimizações de performance utilizando dados previamente coletados e um controle adaptativo. Elas são classificadas de acordo com o sistema do veículo que são aplicadas e podem ter nomes como *Body Control Module*, *Transmission Control Module*, *Engine Management Systems*, etc. (KHARCHE, 2018).

Para Bosch (2020) a *Electronic Engine Control Module* (Módulo Eletrônico de Controle do Motor) é o controlador central do sistema de operação do motor. Ela controla parâmetros chave do funcionamento de um motor a combustão interna como a alimentação de combustível, entrada de ar, injeção de combustível e tempo de ignição através do monitoramento de sensores que indicam, por exemplo, a posição do pedal do acelerador ou a leitura de oxigênio livre pela sonda de exaustão.

2.1.3 Comunicação Interna

O aumento do uso de dispositivos eletrônicos em veículos iniciou com a introdução dos sistemas de injeção de combustível no fim da década de 60 (1960). Por um longo período, mesmo sendo numerosos, esses dispositivos operavam de forma isolada controlando a sua área de interesse. Com a introdução da rede CAN (do inglês *Controller Area Network*) no início dos anos 90 esses sistemas passaram a comunicar-se internamente, trocando informações sobre sua operação. Atualmente todo automóvel novo contém uma rede de ECU's e existe uma forte tendência para que essa característica siga em crescimento. (TILS, 2006).

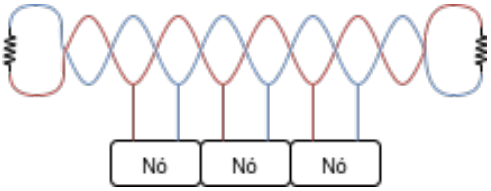
Um veículo automotor moderno pode ter aproximadamente até 70 subsistemas eletrônicos conectados e inicialmente a comunicação interna dos sistemas dos automóveis era um cabeamento ponto a ponto complexo e caro. Gradualmente esse arranjo foi sendo substituído por redes internas mais baratas, menos complexas e menos pesadas. A rede serial CAN (*Controller Area Network*), por ter uma alta integridade, destacou-se tornando a rede padrão utilizada na comunicação interna dos veículos na atualidade. (NAIK *et al.*, 2018).

2.1.3.1 CAN

Segundo Bosch (1991), CAN é um protocolo de comunicação serial que suporta de forma eficiente o controle em tempo real distribuído. Em eletrônica automotiva, unidades de controle

do motor, sensores e outros componentes são conectados usando CAN com velocidade de banda de até 1 M*bit*/s. Esse sistema se torna efetivo em relação a custo devido sua composição simples: somente dois fios são necessários para a conexão de todos componentes do veículo como ilustra a Figura 3. Dessa forma, torna-se vantajoso não só a aplicação desse protocolo ao controle de dispositivos ligados ao motor, mas também a outros periféricos como lâmpadas, janelas e travas elétricas que anteriormente necessitavam de um complexo e volumoso arranjo de cabos.

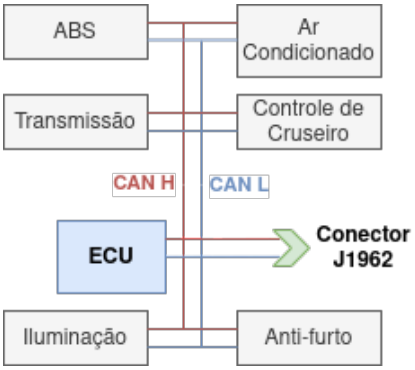
Figura 3 – Representação da rede CAN e seus nós.



Fonte: Adaptado de CAN-Cia (2020).

Diferente de outros barramentos como USB (*Universal Serial Bus*) ou redes como *Ethernet*, o CAN não envia grandes blocos de dados de um ponto a outro sob a supervisão de um mestre central de rede. Em uma rede CAN várias pequenas mensagens são emitidas como um *broadcast*, provendo uma consistência de dados em todos os nós do sistema. Assim informações como temperatura, rpm e tantas outras medições existentes em um veículo automotor são conhecidas por qualquer nó conectado na rede. (CORRIGAN, 2002). A Figura 4 demonstra um exemplo de rede CAN composta por elementos existentes nos veículos modernos.

Figura 4 – Exemplo de rede CAN em um veículo automotor.



Fonte: Elaborado pelo autor.

Inicialmente o protocolo CAN foi desenvolvido para o uso em veículos automotores para transporte de passageiros. Atualmente é utilizado em muitas outras indústrias. Aplicações na área de transportes, controle indústrial de máquinas, automação residencial e predial, equipamentos médicos e automações de laboratórios fazem uso do protocolo. Em 2011 a empresa Robert Bosch GmbH (criadora do CAN) e outros especialistas desenvolveram melhorias no protocolo, o resultado foi nomeado protocolo CAN FD (*Flexible Data-rate*). (CAN-CIA, 2020).

O protocolo CAN pode ser dividido em duas camadas: o *Data Link Layer* e o *Physical Layer*. A camada física (*Physical Layer*) define como os sinais são transmitidos. A camada de *link* de dados se divide em duas outras camadas: transferência (*Transfer Layer*) e objeto (*Object Layer*).

A camada de transferência representa o centro do protocolo CAN, apresenta mensagens recebidas para o *Object Layer* e aceita as mensagens a serem transmitidas. Essa camada é responsável por temporização e sincronização de *bits*, *framing* de mensagens, arbitragem, confirmações de recebimento (*acknowledgement*), detecção e sinalização de erros e confinamento de falhas. Por sua vez a camada de objeto se responsabiliza pela filtragem e tratamento de mensagens e estados. (BOSCH, 1991).

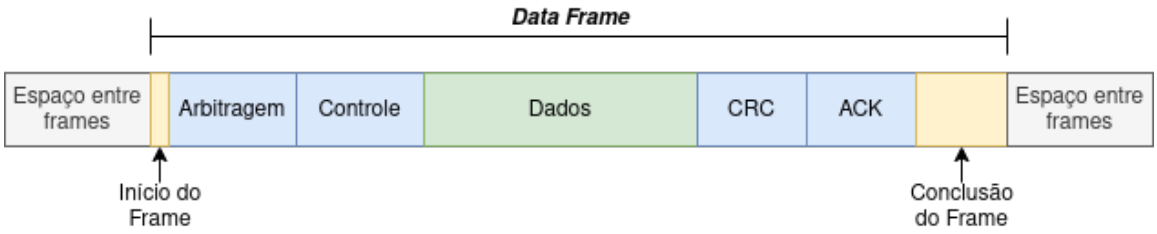
Frames CAN

A transferência de mensagens é controlada e realizada por quatro tipos diferentes de *frames* CAN: *Data Frame*, *Remote Frame*, *Error Frame* e *Overload Frame*. O *Data Frame* carrega dados de um transmissor aos receptores. O *Remote Frame* e o *Error Frame* são transmitidos por uma unidade na rede para solicitar a transmissão de um *Data Frame* com o mesmo identificador e quando é detectado um erro na rede respectivamente. Já os *Overload Frames* são usados para prover um tempo de aguardo extra antes ou após um *Data Frame* ou *Remote Frame*. (BOSCH, 1991).

Os níveis da rede CAN são conhecidos como dominante e recessivo. A especificação do protocolo não prevê como esses níveis são manifestados na aplicação física e utiliza essa denominação justamente para não arbitrar acerca desse detalhe. No caso de uma implementação física *AND* (operação E lógico) da rede, o nível dominante seria representado pelo lógico "0"e o recessivo representado pelo lógico "1".

O *Data Frame* é composto de sete campos, como demonstra a Figura 5. O início do *frame* é composto por um único *bit* dominante, esse campo só pode ser executado por uma estação quando a rede CAN está em repouso (*idle*). Todas as estações conectadas na rede devem sincronizar com este sinal de início de *frame*. Em seguida vem o campo de arbitragem (*Arbitration Field*) que consiste no identificador e no *bit* RTR (*Remote Transmission Request*). O identificador é um campo que identifica a mensagem e é utilizado pela camada de objeto na filtragem e o RTR em um *Data Frame* deve ser dominante.

Figura 5 – Frame de dados CAN (*Data Frame*).

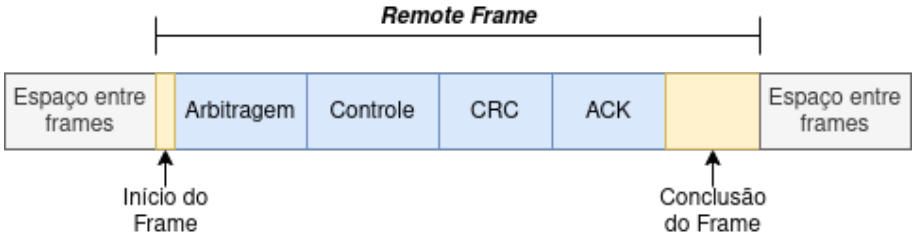


Fonte: Adaptado de Bosch (1991).

Após a arbitragem segue o controle que consiste em seis *bits* contendo a quantidade de *bytes* que o campo de dados terá. O campo de dados é o seguinte e é a informação que o *Data Frame* está transportando. Ele pode ter um tamanho zero e por isso é possível que não faça parte da mensagem. De qualquer forma depois do campo de dados ou de controle, segue um campo CRC (do inglês *cyclic redundancy code*) para conferência de conteúdo da mensagem. Após o CRC, vem o campo chamado *ACK Field*, ou *Acknowledgement*, tem um tamanho de dois *bits* sendo o primeiro *bit* chamado *Ack Slot*, que deve ser um *bit* recessivo, e o segundo *Ack Delimiter* que deve ser dominante. Caso um recebedor tenha sucesso em ler a mensagem, deve sobrescrever o *Ack Slot* com um bit dominante. Encerrando o *Data Frame* a conclusão do frame é composta por sete *bits* recessivos.

O *frame* de requisição de dados (*Remote Frame*) é composto por seis diferentes campos, como mostra a Figura 6. Semelhante ao *Data Frame* os primeiros três campos são o início de *frame*, arbitragem e controle. Ao contrário do *frame* de dados, no campo de controle do *Remote Frame* o parâmetro RTR deve ser recessivo e o tamanho dos dados expresso nesse campo deve ser o dos dados desejados. Como este é um *frame* de requisição de dados o *Data Field* não está presente e após o campo de controle seguem o CRC, *Ack* e a conclusão do *frame*.

Figura 6 – Frame de requisição CAN (*Remote Frame*).

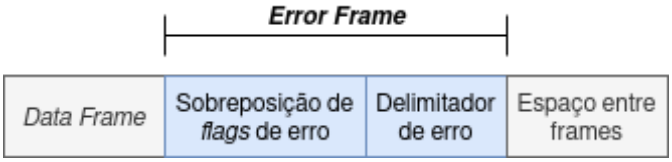


Fonte: Adaptado de Bosch (1991).

O *frame* de erro, é composto por dois campos somente. O primeiro é uma superposição de marcadores (*flags*) de erro provenientes de unidades diferentes da rede. O segundo é um delimitador de erro. Existem dois tipos de *flags* de erro a ativa e a passiva. A ativa consiste em seis *bits* dominantes consecutivos. A passiva consiste em seis *bits* recessivos consecutivos, a menos que alguma outra unidade na rede sobreponha com *bits* dominantes. A Figura 7 demonstra

a estrutura de um *frame* de erro.

Figura 7 – Frame de erro CAN (*Error Frame*).



Fonte: Adaptado de Bosch (1991).

2.1.3.2 OBDII

O sistema de diagnóstico embarcado OBDII (do inglês *On-Board Diagnostic System*) é um padrão desenvolvido para permitir a regulamentação de emissão de gases poluentes veiculares definida pelo EPA (*Environmental Protection Agency*), criado nos Estados Unidos da América em 1996 pelo SAE. (MALEKIAN *et al.*, 2017). Com o advento da especificação OBDII todos os veículos vendidos a partir de 1996 são obrigados a estarem em conformidade com o padrão tanto no software quanto no hardware. (SAWANT, 2018).

A primeira versão deste sistema originou-se em 1988, onde a CARB (do inglês *California Air Resources Board*) introduziu o OBD para controle da poluição do ar gerada pelo tráfego. Além do controle de emissões foi introduzida a indicação de problemas de funcionamento através da MIL (do inglês *Malfunction Indicator Lamp*) presente no painel dos veículos da época. A primeira especificação, desenvolvida pela SAE, do OBD foi nomeada OBDI e tinha o propósito de fomentar a fabricação de veículos mais confiáveis e mais favoráveis ao meio ambiente. Com a evolução da tecnologia embarcada nos veículos, uma nova versão da especificação foi desenvolvida, cobrindo as principais limitações da primeira e permitindo acesso a leituras de diagnósticos e sensores disponíveis nas novas ECU's, ela foi chamada de OBDII. (SAWANT, 2018).

Três coisas são definidas pelo padrão OBDII: os códigos de falha ou DTC's (do inglês *Diagnostic Trouble Codes* ou códigos de falha para diagnóstico) padrões, os comandos de consulta a sensores e estados do veículo ou PID's (do inglês *Parameter Identifiers* ou identificadores de parâmetros) padrões e o padrão de comunicação com a ECU. (SIM; SITOHANG, 2014).

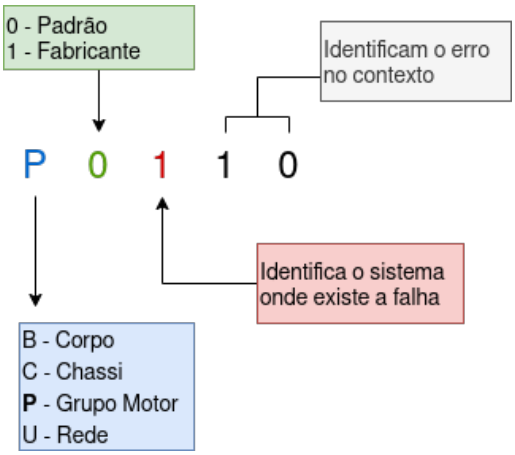
Códigos de Falha para Diagnóstico

Sendo um dos sistemas computacionais embarcados nos veículos modernos, o OBDII permite ao usuário acesso a DTC's registrados internamente no veículo devido a ocorrência de possíveis falhas no grupo motor ou sistemas como frenagem, catalização de gases resultantes da combustão ou até mesmo de câmbio no caso de veículos com automatização de troca de marchas. Isso torna a identificação e solução de problemas no veículo, que anteriormente exigiriam grande

esforço para localização e diagnóstico, mais fácil e simplifica esse processo dada a padronização proposta pela especificação. (SAWANT, 2018).

Códigos DTC são formatados com o intuito de permitir a identificação da falha somente pelo código de erro. O primeiro caractere representa o sistema do veículo que gerou a falha, o segundo determina se o código faz parte do padrão ou é específico do fabricante, o terceiro qual o subsistema registrou o problema e por fim os últimos dois caracteres identificam o erro dentro do contexto colocado pelos caracteres anteriores. Na Figura 8 demonstra-se a composição desses códigos de falha. (KULKARNI; RAJANI; VARMA, 2017).

Figura 8 – Diagrama de composição de códigos DTC



Fonte: Adaptado de Kulkarni, Rajani e Varma (2017).

Identificadores de Parâmetros

Segundo Malekian *et al.* (2017) o sistema OBDII tem também a capacidade de acessar informações em tempo real de sistemas e subsistemas do veículo monitorados por sensores, esses parâmetros são identificados por seus PID's. Com esses dados pode-se auxiliar na redução da emissão de gases poluentes monitorando constantemente os processos de combustão do veículo, identificar problemas de forma preventiva e acompanhar o funcionamento do veículo e seus padrões de trabalho.

Quando um dispositivo deseja ler uma informação do veículo ele envia comando OBDII PID para a ECU, que por sua vez, caso suporte o comando requisitado, responde com os dados solicitados. Os PID's são expressos em códigos hexadecimais de dois caracteres iniciando em "00". Existem nove modos de operação para a comandos PID como demonstra a Quadro 1. Como a especificação não determina uma cobertura mínima do padrão, é possível que fabricantes não implementem todos os comandos previstos pelo OBDII. Em contrapartida, o fabricante pode definir seus próprios PID's tornando o sistema mais sofisticado. (SIM; SITOHANG, 2014). Alguns dos principais PID's e seus significados são listados no Quadro 2.

Quadro 1 – Modos de comando dos OBDII PID's.

Modo	**Informações Resultantes**
Mode 1	Retorna leituras em tempo real do motor
Mode 2	Retorna leituras do momento do último DTC registrado
Mode 3	Retorna os DTC's registrados na ECU no momento do comando
Mode 4	Apaga todos os DTC's registrados no ECU no momento do comando
Mode 5	Retorna resultado de testes do sensor de oxigênio
Mode 6	Retorna resultados de testes de outros sensores
Mode 7	Retorna os DTC's pendentes
Mode 8	Modo de controle do sistema embarcado
Mode 9	Retorna o VIN (*Vehicle Identification Number*)

Fonte: Adaptado de Sim e Sitohang (2014).

Quadro 2 – Principais PID's do *Mode 1*

PID (Mode 1)	**Descrição**	**Unidade**
00	PID's suportados [1 - 20]	Arranjo de *bits*
04	Carga do motor	Percentual
05	Temperatura o líquido de arrefecimento	Graus Celsius
0C	Revoluções por minuto	rpm
0D	Velocidade	Quilômetros por hora
10	Fluxo de ar de entrada	Gramas por segundo
11	Posição da válvula de admissão	Percentual
14	Sensor de Oxigênio 1 (Banco 1)	Volt
15	Sensor de Oxigênio 2 (Banco 1)	Volt
16	Sensor de Oxigênio 3 (Banco 1)	Volt
17	Sensor de Oxigênio 4 (Banco 1)	Volt
1F	Tempo de funcionamento do Motor	Segundos
2F	Nível de combustível	Percentual

Fonte: Adaptado de OBD (2020a).

Comunicação com a ECU

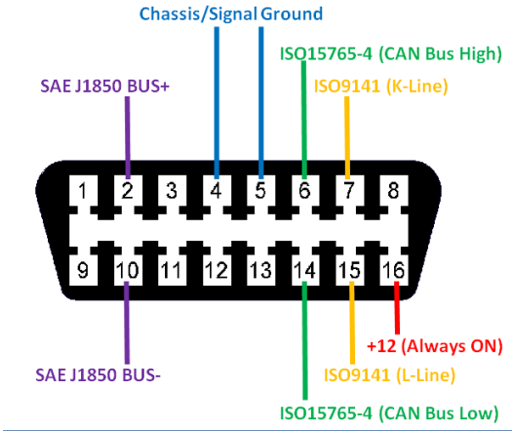
A interface OBDII contempla cinco protocolos de comunicação com a ECU, apresentados no Quadro 3, apesar disso a maior parte dos veículos suportam somente um destes, sendo o protocolo CAN o mais comumente adotado. Além da interface de comunicação na camada de dados, também é exigido um conector J1962 em veículos fabricados para estarem conforme a especificação, esse conector é demonstrado na Figura 9. Muitas vezes é possível determinar qual ou quais protocolos são suportados no veículo pelos pinos disponíveis para conexão no *plug* OBDII. (SAWANT, 2018).

Quadro 3 – Protocolos previstos pela interface OBDII

Especificação	**Protocolo**
SAE J1850 PWM	*Class B Data Communication Network Interface with Pulse Width Modulation*
SAE J1850 VPW	*Class B Data Communication Network Interface with Variable Pulse Width*
ISO 9141-2	K-Line ou L-Line
ISO 14320 KWP2000	*Keyword Protocol 2000*
ISO 15765 CAN	*Controller Area Network* (CAN *bus*)

Fonte: Adaptado de Sim e Sitohang (2014).

Figura 9 – Diagrama de pinos do conector OBDII (J1962)



Fonte: Extraído de Components101 (2020).

Segundo Sim e Sitohang (2014) apesar de existirem cinco protocolos definidos na especificação, de um ponto de vista elétrico, os protocolos ISO 9141-2 e ISO 14230 são iguais, apenas diferentes devido a uma mudança na ECU dos veículos que os utilizam. É possível realizar integrações com estes cinco protocolos tanto via hardware como via software. Uma das implementações mais populares são os *chips* da família ELM327 que traduzem a comunicação com a ECU em texto puro dentro de um envelope serial RS232.

2.2 Internet das Coisas

Nessa seção são abordados alguns temas e definições acerca da Internet das Coisas, exemplos de aplicações da tecnologia e alguns dos principais protocolos utilizados.

2.2.1 Definição

O termo Internet das Coisas ou na língua inglesa *Internet of Things* (IoT) não possui uma definição objetiva e delimitada. Existem diversas definições que procuram determinar quais as

características dessa tecnologia e quais os pontos que delimitam seu escopo. (MINN; ZENG; BHARGAVA, 2015).

Para Chaudhary *et al.* (2019) a idéia principal da Internet das Coisas é a interconexão de dispositivos eletrônicos, sendo estes analógicos ou digitais, homogêneos ou heterogêneos, mas com uma transmissão sobreposta entre eles, para que consigam se comunicar de forma eficiente.

De forma análoga Rouse, Gillis e Rosencrance (2020) definem como um sistema de dispositivos, máquinas mecânicas e digitais, objetos, animais ou pessoas inter-relacionados e capazes de realizar computação, identificados de forma única e com a habilidade de transferir dados na rede sem a necessidade de iteração humana. Sendo a "coisa" da Internet das Coisas uma pessoa com um implante cardíaco para monitoramento, um animal com um *biochip*, um automóvel com sensores ou ainda quaisquer dispositivos capazes de receber um endereço de IP (*Internet Protocol*).

Segundo Dorsemaine *et al.* (2016) para se definir bem o que é IoT é necessário analisar a arquitetura, sendo que sempre se associa com as seguintes etapas: dados que precisam ser transportados do seu ambiente local, armazenados, processados e tornados disponíveis. Onde somente o ambiente local é de fato específico para o IoT e os outros três são comuns em outras tecnologias onde grandes quantidades de dados são tratados. Após considerar esses fatos, pode-se definir IoT como o grupo de infraestruturas interconectando os objetos conectados e permitindo o manejo desses, aquisição e acesso aos dados por eles gerados.

Já o IERC (*Internet of Things Research*) define que IoT é uma infraestrutura dinâmica de rede global com capacidades de se auto configurar baseada em padrões e protocolos de comunicações interoperáveis onde "coisas" físicas e virtuais tem identidades, atributos físicos e personalidades virtuais e usam interfaces inteligentes além de se integrar de forma transparente a rede de informação. (MINN; ZENG; BHARGAVA, 2015).

Diante de tantas definições é possível perceber alguns pontos convergentes como a existência de dispositivos interconectados, capazes de operar sem intervenção humana, comunicando-se através de protocolos e que tem uma identificação única dentro da rede que pertencem.

2.2.2 Principais Protocolos

A iteração entre dispositivos IoT pode seguir o conceito de M2M (*Machine to Machine*). Aplicações que seguem essa visão podem ser usadas para entregar serviços em lugares remotos sem a necessidade de intervenção humana. Esses dispositivos muitas vezes são limitados em recursos devido ao seu tamanho físico e frequente baixa conectividade. Dessa forma protocolos de comunicação leves e que permitam a execução das duas categorias de comunicação comuns no meio IoT, telemetria e telecomando, são os ideais para aplicações desse meio. (MINN; ZENG; BHARGAVA, 2015).

Dois dos principais protocolos utilizados no cotexto IoT são o MQTT (*MQ Telemetry*

Transport) e o CoAP (*Constrained Application Protocol*) segundo Minn, Zeng e Bhargava (2015). Estes protocolos tem as características exigidas pelo conceito de M2M, segundo suas documentações e especificações, sendo leves e simples de se utilizar. (OASIS Open, 2015)(SHELBY; HARTKE; BORMANN, 2014).

2.2.2.1 CoAP

CoAP (do inglês *Constrained Application Protocol*) é um protocolo de transferência na web especializado para uso em redes de baixa confiabilidade por dispositivos com recursos escassos. Desenvolvido para uso em aplicações M2M como automações prediais ou de redes de energia inteligentes. (SHELBY; HARTKE; BORMANN, 2014). Segundo Rayes e Salam (2019) CoAP é uma alternativa leve ao protocolo HTTP (*Hypertext Transfer Protocol*), enquanto um request HTTP precisa de sete mensagens TCP (*Transmission Control Protocol*) para realizar uma simples operação *Get*, o CoAP reduz essa quantidade de mensagens utilizando UDP (*User Datagram Protocol*) podendo chegar até duas mensagens para a mesma operação. Mesmo com estas diferenças, o CoAP foi projetado para ter uma interface simples com HTTP para integração com a internet. (SHELBY; HARTKE; BORMANN, 2014).

CoAP é um protocolo RESTful (*Representational State Transfer*). Suporta os verbos CRUD (*Create Read Update Delete*) e ainda o paradigma *publish*/*subscribe* com o verbo *observe*. (RAYES; SALAM, 2019). O modelo de iteração é semelhante ao HTTP, onde estão presentes um cliente e um servidor, todavia em um contexto M2M é comum um dispositivo atuar nos dois papéis. (SHELBY; HARTKE; BORMANN, 2014).

2.2.2.2 MQTT

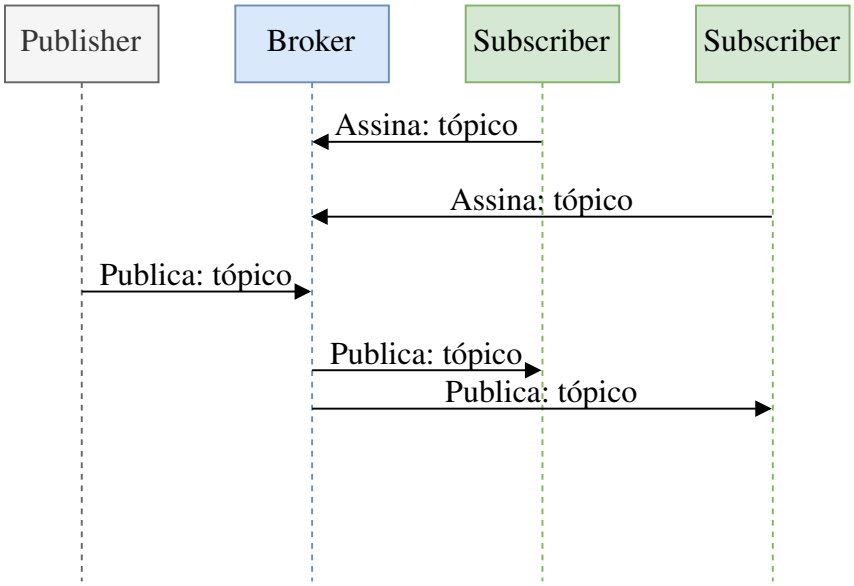
O MQTT é um protocolo que utiliza a arquitetura cliente e servidor e tem suas mensagens transportadas no modelo publicador/assinante, muito conhecido no inglês como *publish*/*subscribe*. Foi projetado para ser leve, simples e fácil de implementar. (OASIS Open, 2015).

Originalmente projetado pela IBM para telemetria corporativa, os criadores Andy Standford-Clark e Arlen Nipper precisavam de um protocolo capaz de operar com um consumo de bateria mínimo e uma largura de banda mínima para conectar oleodutos via satélite. O acrônimo MQTT tem nas suas primeiras duas letras uma referência ao produto da IBM MQ *Series* ®e as duas últimas correspondem a *Telemetry Transport*. (HIVEMQ, 2020). Em 2010 a IBM publicou o protocolo MQTT 3.1 livre de *royalties* e em outubro de 2014 a OASIS (*Organization for the Advancement of Structured Information Standards*) publicou uma especificação do protocolo de forma aberta. (RAYES; SALAM, 2019).

Este protocolo, que trabalha no paradigma *publish*/*subscribe* (*pub*/*sub*), organiza seu processo de transferência de dados através de tópicos. Clientes podem atuar tanto como publicadores quanto como assinantes, enquanto o *broker* realiza o direcionamento das mensagens

(como demonstra a Figura 10). Os assinantes e publicadores não tem contato direto um com o outro, sendo sempre intermediados pelo *broker*, isso desacopla os emissores da informação dos receptores. O trabalho dos *brokers* é filtrar todas as mensagens recebidas e distribuí-las corretamente aos assinantes.

Figura 10 – Fluxo de mensagens MQTT.



Fonte: Elaborado pelo autor.

Além de desacoplar a origem das mensagens do destino, o protocolo MQTT trata todas as mensagens de forma assíncrona, ou seja, não bloqueia o fluxo de mensagens enquanto aguarda uma resposta ou nova mensagem e não é fiel ao instante em que a mensagem foi enviada desacoplando o tempo de publicação do tempo de recebimento. O *broker* MQTT também deve ser capaz de armazenar mensagens de clientes que não estejam *online*. Essas características demonstram o quanto a lógica de processo do protocolo é concentrada no lado do *broker* e faz com que o cliente seja um *pub*/*sub* em sua essência, tornando o MQTT um protocolo leve e favorecendo os dispositivos pequenos e limitados que costumam operar nas bordas. (HIVEMQ, 2020).

Tópicos

Segundo HIVEMQ (2020) tópicos, dentro do contexto do protocolo MQTT, são *strings*, codificadas em UTF-8, que o *broker* utiliza para filtrar e enviar cópias de mensagens para cada cliente que tem uma assinatura que corresponda. Os tópicos podem ser delimitados em níveis, sendo o separador de níveis o caractere "/", utilizados para a organização em estrutura de árvore e criação de uma hierarquia para os nomes de tópicos (*Topic Names*). (OASIS Open, 2015). Na Figura 11 são apresentados exemplos de tópicos com separação hierárquica.

Figura 11 – Tópicos com separação hierárquica.

Fonte: Elaborado pelo autor.

Os nomes de tópicos são obrigatórios em qualquer pacote de publicação (*PUBLISH*). Devem ser o primeiro campo no cabeçalho e não podem conter caracteres curingas (*wildcards*). Apesar disso, como é permitido ao *broker* sobrescrever o nome do tópico de uma mensagem, o assinante pode receber um tópico diferente do originalmente utilizado pelo publicador. (OASIS Open, 2015).

Já os filtros de tópicos (*Topic Filters*), utilizados pelos assinantes para criar uma assinatura junto ao *broker*, podem conter caracteres curinga afim de demonstrar interesse em um grupo de nomes de tópicos. Uma lista desses filtros deve estar presente nos pacotes de assinatura (*SUBSCRIBE*) e remoção de assinatura (*UNSUSBSCRIBE*) enviado pelo cliente para o *broker*. Um *broker* deve suportar o uso de curingas no filtro de tópicos, caso não suporte ele deve rejeitar quaisquer pedidos de assinatura que contenham tais caracteres.

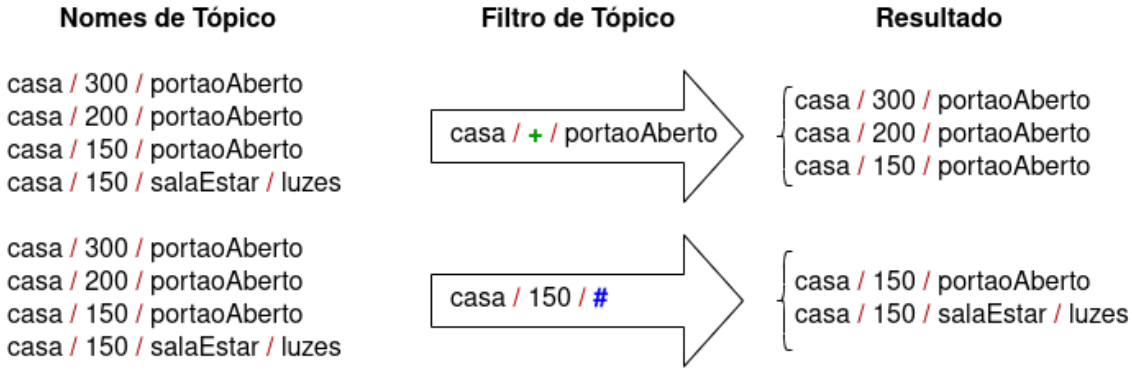
Os caracteres especiais utilizados para seleção de mensagens baseadas em nome de tópicos são "+" e "#". Onde o caractere "+" é um curinga para um nível (e somente um) de tópico e o caractere "#" para um nível ou mais de acordo com a hierarquia do tópico. Além disso caracteres de múltiplos níveis ("#") só podem ser utilizados na última posição de um filtro e devem ser precedidos de um separador ou ser o único caractere de um filtro. (OASIS Open, 2015). Um exemplo de aplicação destes caracteres pode ser visto na Figura 12.

Um terceiro caractere especial ("$") se refere a funções internas do *broker* e pode ser usado como primeiro caractere de um nome de tópico. O *broker* de mensagens deve prevenir o uso desse caractere em nomes de tópicos de clientes e não deve entregar mensagens com esse caractere inicial a assinaturas que comecem com os curingas "#" e "+".

QoS

QoS ou *Quality of Service level* (Nível de Qualidade de Serviço) é um acordo existente entre a origem e o destino de uma mensagem que define qual a garantia de envio de uma mensagem especifica. (HIVEMQ, 2020). Segundo OASIS Open (2015) quando o *broker* está entregando mensagens de aplicação de mais de um cliente, cada cliente é tratado de forma independente. O protocolo MQTT define o uso de três níveis de qualidade de serviço, sendo eles numerais de zero a dois.

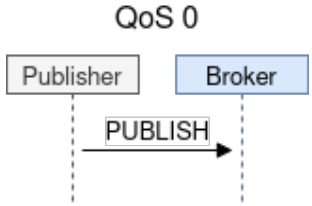
Figura 12 – Exemplo de seleção com caracteres curinga em filtros de tópicos.



Fonte: Elaborado pelo autor.

No primeiro nível de serviço (nível zero) a entrega da mensagem acontece de acordo com as capacidades da rede pela qual ela trafega. Nenhuma resposta é enviada pelo recebedor e não é previsto nenhum mecanismo de tentativas em caso de falha, demonstrado pela Figura 13. A mensagem é entregue uma vez ou nenhuma. (OASIS Open, 2015). É o nível de serviço mais baixo e deve ser usado quando a perda de mensagens é aceitável e não há necessidade de enfileiramento de mensagens (persistência).

Figura 13 – Pacotes para uma publicação com QoS 0.

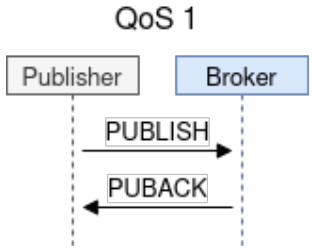


Fonte: Elaborado pelo autor.

Já no segundo nível de serviço (nível um) o protocolo garante a entrega da mensagem pelo menos uma vez ao destino. A mensagem de publicação (*PUBLISH*) tem um identificador de pacote e a origem é notificada do recebimento através de um pacote *PUBACK* (retorno de publicação), ilustrado pela Figura 14. Deve ser usado quando não é aceitável a perda de mensagens, mas a duplicação não é um problema. (OASIS Open, 2015). Segundo HIVEMQ (2020) é o nível de serviço mais comum.

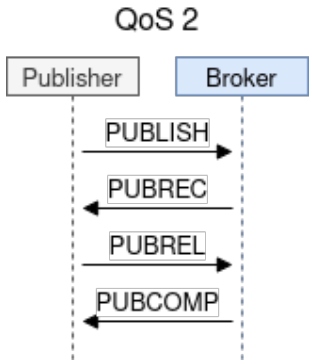
No mais alto e terceiro nível de serviço (nível dois) existe, além da certeza de entrega, a garantia de que ela foi entregue uma única vez. Com o uso desse nível de serviço ocorre um acréscimo considerável de esforço e tempo para a entrega das mensagens. Para alcançar esses pré-requisitos uma única mensagem necessita de quatro pacotes *PUBLISH*, *PUBREC*, *PUBREL* e *PUCOMP* como demonstrado da Figura 15. (OASIS Open, 2015). Para HIVEMQ (2020) QoS 2 deve ser usado quando é crítico que a aplicação destino receba todas as mensagens e as receba uma única vez. Um pequeno resumo dos níveis de serviço é demonstrado na Quadro 4.

Figura 14 – Pacotes para uma publicação com QoS 1.



Fonte: Elaborado pelo autor.

Figura 15 – Pacotes para uma publicação com QoS 2.



Fonte: Elaborado pelo autor.

Quadro 4 – *Quality of Service* (Qualidade de Serviço).

QoS	**Comportamento resumido**
0	A mensagem é entregue uma vez ou nenhuma
1	A mensagem é entregue pelo menos uma vez
2	A mensagem é entregue exatamente uma única vez

Fonte: Elaborado pelo autor.

O nível de serviço pode sofrer reduções entre o cliente que envia a mensagem (publicador) e o cliente que recebe a mensagem (assinante). Quem define o nível de serviço de uma mensagem inicialmente é o publicador, mas o *broker*, ao entregar uma mensagem para os assinantes, deve utilizar o QoS definido pelo assinante no momento da assinatura. (HIVEMQ, 2020).

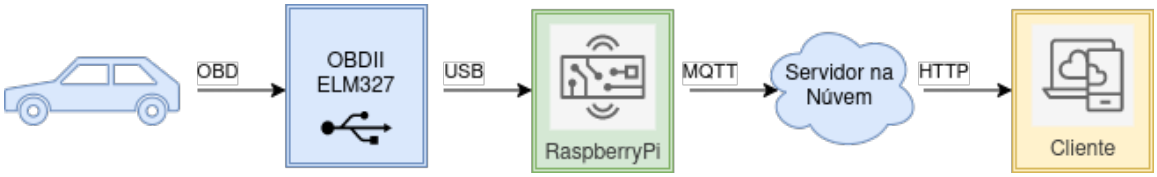
Essa mesma propriedade (QoS) é utilizada para o *broker* determinar quando guardar as mensagens sem confirmação de recebimento do assinante no caso de seções persistentes. Somente com os níveis um ou dois a persistência é realizada e as mensagens são mantidas até uma nova conexão do cliente que criou a seção persistente com o *broker*.

2.3 Trabalhos relacionados

Neste capítulo são apresentados trabalhos que trazem propostas próximas ao sistema proposto neste trabalho. Os trabalhos abordados nesse capítulo estão organizado em ordem de semelhança de forma descendente.

Em Binmasoud e Cheng (2019), um sistema de monitoramento veicular é projetado e implementado utilizando os protocolos de comunicação OBDII e MQTT. Neste trabalho os componentes arquiteturais são uma ferramenta de leitura OBDII, um Raspberry Pi e um servidor *web* (como demonstra a Figura 16), sendo o Raspberry Pi o microcomputador responsável pela interpretação dos dados coletados no veículo e propagação destes ao servidor de aplicação. A comunicação entre a ferramenta OBDII (baseada no *chip* ELM327) e o Raspberry Pi se dá pela porta USB através de uma aplicação escrita em *Python*, já o envio das informações para o servidor é realizada pela internet (assumindo a disponibilidade de uma rede móvel) utilizando o protocolo MQTT.

Figura 16 – Arquitetura da solução proposta por Binmasoud e Cheng (2019).

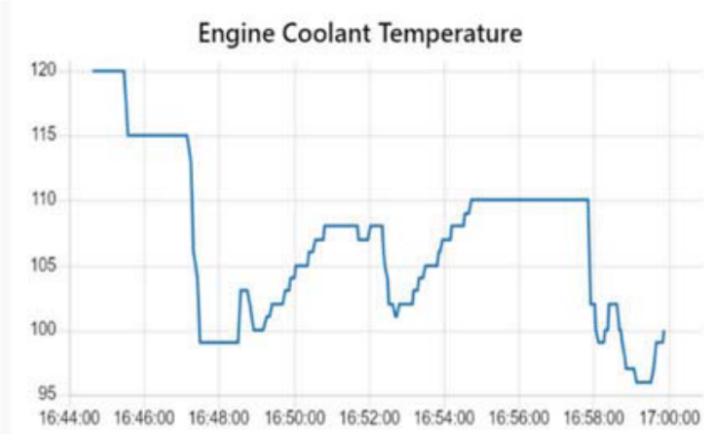


Fonte: Adaptado de Binmasoud e Cheng (2019).

O foco de Binmasoud e Cheng (2019) é o monitoramento dos parâmetros de um veículo, habilitando um possível mecânico ou outro usuário a consultar à distância os dados coletados pelos sensores do automóvel. O trabalho resulta no registro de leituras, com o veículo em movimento e parado, de diversos sensores de um automóvel, dando acesso ao usuário via uma interface *web* como exemplifica a Figura 17. Como proposta de continuidade do trabalho os autores sugerem a detecção de falhas no motor em tempo real.

No trabalho proposto por Vijaya Shetty *et al.* (2017) é apresentado um cenário semelhante, utilizando a capacidade de coletar dados de sensor do veículo via OBDII, a leitura de um módulo GPS (*Global Positioning System*) e a instalação de uma câmera de monitoramento, os autores propõem o desenvolvimento de um sistema de informação e entretenimento conectado ao automóvel. Com grande foco no aplicativo que interage com o usuário final, o sistema seria aplicável a situações como consulta do proprietário (a fim de que conheça mais o seu automóvel), controle e acompanhamento a distância do uso do veículo por um responsável e controle de velocidade. A comunicação entre os componentes não é um objeto de estudo direto, sendo explorados somente cenários de falha nos componentes e suas implicações na aplicação desenvolvida.

Figura 17 – Exemplo de resultado da apresentação dos dados de monitoramento por Binmasoud e Cheng (2019).



Fonte: Extraído de Binmasoud e Cheng (2019).

Como conclusão Vijaya Shetty *et al.* (2017) citam principalmente a possibilidade de manutenção preventiva (devido a maior acessibilidade do estado do veículo) como benefício ao usuário em conhecer seu automóvel e a capacidade de acompanhamento e persistência das viagens devido a instalação da câmera e GPS. Os autores ressaltam a grande aplicabilidade dessas capacidades em casos como o de seguradoras que desejam ter informações sobre as condições em que um veículo se envolveu em um acidente ou ainda monitoramento de hábitos de direção.

Amarasinghe *et al.* (2016) abordam o mesmo problema utilizando CEP (*Complex Event Processing*) para avaliar a importância dos dados coletados no veículo via OBDII. Diferente da proposta de Binmasoud e Cheng (2019), onde o microcomputador tem somente o papel de *gateway*, uma aplicação *Android*, além de retransmitir ao servidor os dados sensoriais do automóvel, realiza um processamento sob os eventos coletados e apresenta ao usuário na forma de *dashboards* (painéis de visualização de medidas) de forma local. Além disso o trabalho aborda o tema de consumo de banda de rede móvel do *smartphone* do usuário com a técnica de CEP criando um filtro de mensagens e inibindo a propagação nos casos em que a informação é percebida como inútil ou de pouca significância. O papel do servidor *backend* neste trabalho consiste em realizar análises mais complexas e que custam mais processamento, como reconhecer padrões de direção irresponsável, e manter um banco de dados histórico.

A principal desvantagem da solução proposta por Amarasinghe *et al.* (2016) é a dependência completa do sistema na comunicação de dados realizada pelo *smartphone*, caso o usuário não permita a transmissão de dados no *background* o sistema perde a sua utilidade, visto a necessidade de comunicação com o servidor e a natureza multifuncional do dispositivo *Android*. Em uma iniciativa para redução desse risco os autores propõem o desenvolvimento de um sistema *blackbox* dispensando o uso do *smartphone* do usuário.

O tema de gerenciamento de frotas de veículos é abordado por Malekian *et al.* (2017).

Com o objetivo de medir alguns parâmetros como velocidade e consumo de combustível para acompanhamento e análise, os autores utilizam um leitor de códigos OBDII. O meio de transporte desses dados, diferentemente dos trabalhos anteriores, é *WiFi* diretamente e o servidor remoto fica responsável pelo armazenamento e apresentação dos dados no formato gráfico. Um módulo de conectividade *WiFi* chamado *Carambola2* é utilizado em união a MCU (*Microcontroller Unit*), que se comunica com a ferramenta de leitura OBDII via RS232, e permite a transmissão das leituras sensoriais do automóvel para o servidor remoto. Com essa configuração foi possível uma comunicação satisfatória com a distância total de 900 metros entre os dois módulos *WiFi* (servidor e leitor OBDII).

Apesar do sucesso nos experimentos, alguns pontos são apontados por Malekian *et al.* (2017) como possíveis melhorias no sistema como um todo: o desligamento do equipamento embarcado no veículo quando o mesmo é desligado, o alcance dos módulos de comunicação *Wifi* e o tempo de inicialização do sistema. Possíveis soluções seriam a adição de uma bateria para alimentação do leitor de OBDII para solucionar a questão de desligamento e tempo de inicialização e a utilização de rede GSM (*Global System for Mobile Communications*) para a limitação de distância na comunicação.

Em Kowalik (2018), com uma configuração semelhante aos trabalhos anteriores, o foco é a utilização de dados coletados no veículo para diagnóstico de falhas nos sensores ou componentes do motor. Persistindo os dados coletados em arquivos CSV (do inglês *Comma Separated Values*) e posteriormente realizando a importação em um banco de dados, no caso *InfluxDB*, com auxílio de uma aplicação escrita na linguagem de programação Go, o trabalho faz uma análise das medidas coletadas focando nas leituras do sensor de temperatura de líquido de arrefecimento e carga do motor.

Durante a análise Kowalik (2018) consegue demonstrar a diferença nas leituras do termostato com a presença de anomalia no veículo e sem. Além disso consegue de forma experimental relacionar grandezas envolvidas na condução do veículo, como velocidade e carga do motor, com a operação do sistema de arrefecimento do automóvel. O autor conclui então que a falha existente no veículo não poderia ter sido evitada, mas que os dados comportamentais dos componentes observados serão utilizados para detecção de futuras falhas. Sugere ainda que aliando o sistema atual com modelos de *machine learning* podem facilitar esse diagnóstico.

Quadro 5 – Trabalhos relacionados.

Autores	**Trabalho realizado**	**Relação com o trabalho**
Binmasoud e Cheng (2019)	Monitoramento veicular utilizando OBDII e MQTT	Inspiração de arquitetura da solução
Vijaya Shetty *et al.* (2017)	Monitoramento veicular com adição de GPS e Filmagem	Inspiração para a proposta de acompanhamento a distância dos dados
Amarasinghe *et al.* (2016)	Diagnóstico veícular e monitoramento do motorista utilizando HTTP	Inspiração de utilização da rede móvel para aumentar o alcance dos dados
Malekian *et al.* (2017)	OBDII sem fio para gestão de frotas	Não utilizar *WiFi* diretamente, causa uma limitação no alcance dos dados
Kowalik (2018)	Monitoramento veicular utilizando OBDII para diagnóstico remoto	Não utilizar persistência local dos dados e extração posterior devido a incapacidade de acompanhamento remoto em tempo real

Fonte: Elaborado pelo autor.

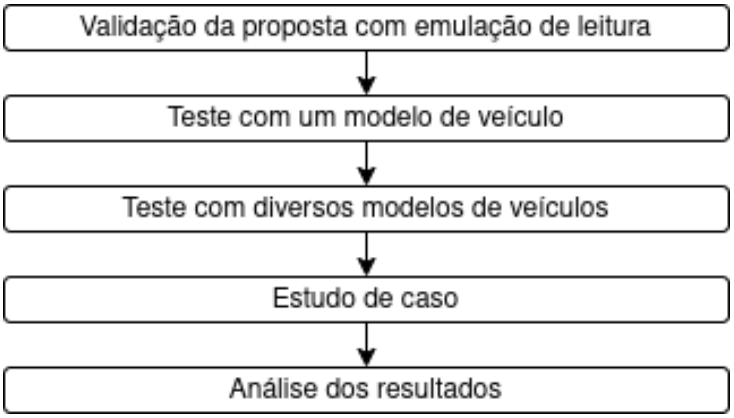
O Quadro 5 demonstra o relacionamento dos trabalhos abordados nessa seção com o trabalho desenvolvido neste documento. O próximo capítulo descreve a metodologia utilizada para atingir o objetivo proposto.

3 METODOLOGIA

Nesse capítulo são apresentadas as etapas e ferramentas utilizadas para o desenvolvimento deste trabalho, além de detalhes de como ele foi executado.

Este trabalho foi desenvolvido em cinco etapas demonstradas pela Figura 18. Primeiramente, após o desenvolvimento dos componentes de comunicação MQTT e armazenamento de dados, uma etapa de validação do modelo de transmissão de dados foi realizada emulando leituras no veículo, em seguida testes com um modelo de veículo e o conjunto completo de equipamentos necessários. Com a conclusão dessas etapas o sistema foi testado em mais veículos de modelos e fabricantes variados. O estudo de caso proposto foi a utilização do sistema em uma viagem e serve para a avaliação da viabilidade do sistema em casos reais de uso.

Figura 18 – Etapas do trabalho.

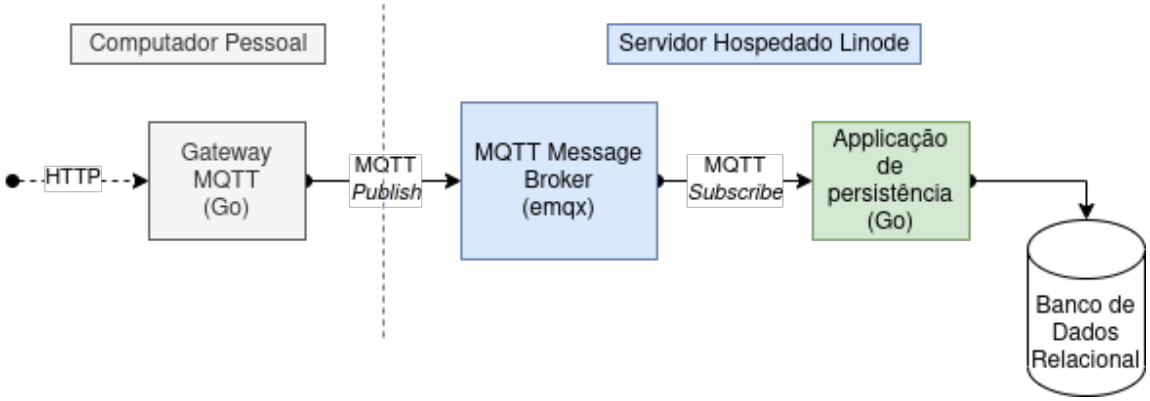


Fonte: Elaborado pelo autor.

A validação da proposta foi realizada em um computador pessoal inicialmente com comunicações locais e em seguida com um servidor localizado na cidade de Dallas, Estados Unidos da América (EUA), hospedado pela empresa Linode, todos com o sistema operacional GNU/Linux. Sendo o estação de trabalho local conectada a internet pela rede sem fio IEEE 802.11.

Nesta primeira validação o sistema de comunicação MQTT foi alimentado com leituras emuladas não estando conectado a um veículo diretamente. Isso permitiu a avaliação da modelagem de tópicos e assinaturas do protocolo *publish/subscribe*. Os componentes dessa validação estão demonstrados na Figura 19.

Figura 19 – Componentes do teste de validação da proposta.



Fonte: Elaborado pelo autor.

A implementação do sistema é uma combinação de tecnologias, as linguagens de programação Go e Python foram utilizadas para as camadas de transmissão via MQTT e leitura dos dados veículares via OBDII respectivamente, um banco de dados relacional para a persistência dos dados coletados e como ferramenta de visualização e acompanhamento o Grafana. Todos componentes são suportados em plataformas Unix, sendo ainda as duas linguagem de programação multi-plataforma.

A linguagem de programação Go foi desenvolvida para a utilização em processos concorrentes e com o intuito de gerar aplicações leves, vantajosa para sistemas com restrições de recursos e para cenários onde a comunicação é um ponto central da funcionalidade. Python, por sua vez, foi utilizado pela maturidade da biblioteca *python-OBD* (OBD, 2020b), atualmente a iniciativa de código aberto, capaz de ser executada pelo Raspberry Pi, com uma rica documentação e grande popularidade na proposta de comunicação pelo protocolo OBDII segundo as estatísticas do site www.github.com, apresentando 635 *Stars* (medida de popularidade da plataforma Github) no momento que este trabalho foi desenvolvido.

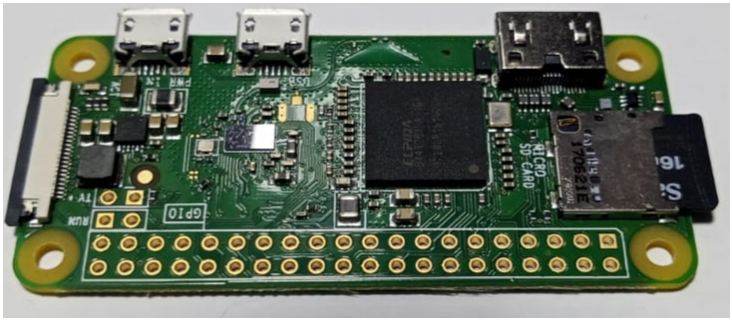
As validações foram realizadas com nível 1 de QoS, no protocolo MQTT, visando uma comunicação com baixa latência mas com garantia de entrega, visto que se trata de uma aplicação de monitoramento e que para evitar problemas de duplicidade de leituras cada mensagem foi marcada com um identificador sendo implementado um conceito de idempotência, onde em caso de repetição a mensagem é ignorada. Os dois componentes de extração e transformação de dados, o adaptador MQTT e o leitor OBDII, se comunicam dentro do mesmo *host* pelo protocolo HTTP.

Com a validação da proposta, foram realizados testes em um automóvel que suporta o protocolo OBDII com o conversor USB Serial OBDII, como o modelo Corsa *Hatch* Maxx com ano de fabricação 2011 da Chevrolet. A área de cobertura da comunicação via protocolo OBDII é determinada pelo fabricante, no caso do modelo de veículo supracitado 21 do total de 94 PID's são suportados pela biblioteca utilizada.

Com os testes de um veículo concluídos e o sistema validado, foram realizados novos testes em mais veículos para verificar a compatibilidade com modelos variados. A verificação em múltiplos veículos é necessária em vista da realidade plural da proposta do trabalho, dado que não se pode garantir que frotas veículares tenham consistência no modelo ou ano dos automóveis.

Para o estudo de caso, uma aplicação real do sistema desenvolvido no monitoramento de um veículo em uma viagem, foi embarcado o gateway OBDII para MQTT em um Raspberry Pi Zero W (Figura 20) e conectado o dispositivo na rede móvel através de um ponto de acesso Wifi gerado por um *smartphone*. Durante o trajeto o sistema manteve um constante processo de leitura de uma seleção dinâmica de parâmetros expostos pelo protocolo OBDII. Este estudo foi conduzido para que fosse possível coletar informações acerca do funcionamento e viabilidade do sistema.

Figura 20 – Raspberry Pi Zero W.



Fonte: Elaborado pelo autor.

Do estudo de caso foi feito uma análise quantitativa e de viabilidade do tráfego de dados e visibilidade na ferramenta de monitoria, no intuito de avaliar a performance e capacidade do sistema de solucionar o problema proposto no trabalho. Para estas análises dois indicadores são utilizados para determinar a viabilidade do sistema: a frequência de leitura dos parâmetros e o tempo entre a leitura e a persistência dos dados no servidor remoto. Para alcançar esse objetivo, toda operação de aquisição de valores sensoriais do veículo tem sua data e hora (com resolução de milissegundos) registrada nas mensagens enviadas para o servidor remoto, que ao persistir os dados também insere a data e hora de gravação.

As medições do banco de dados relacional foram exportadas para um arquivo de texto ordenado e importadas no software Google Sheets®, onde é possível extrair a distribuição dos tempos e frequências e seus pontos mínimos, máximos e médios a fim de determinar os limites dessas grandezas para utilização válida na monitoria da saúde e de parâmetros de automóveis, para verificar a sua viabilidade em situações reais.

3.1 Conversor USB Serial OBDII

Para a conversão dos protocolos de mercado suportados pelos veículos utilizados nos testes para a comunicação serial via USB (RS232) foi escolhido o circuito integrado ELM327. A escolha se deve a alta especialização do componente na interpretação desses protocolos. (ELM, 2021), ao seu preço, acessibilidade no mercado e as diversas bibliotecas disponíveis em código aberto capazes de abstrair a interface com o dispositivo.

Figura 21 – Conversor USB Serial OBDII.

Fonte: Elaborado pelo autor.

Com a metodologia definida foi possível iniciar o desenvolvimento do sistema proposto. Uma explicação dos componentes, objetivos e detalhes de implementação é apresentada no capítulo 4.

4 SISTEMA PROPOSTO

Neste trabalho é proposto um sistema capaz de realizar leituras em um veículo que suporte o padrão OBDII, realizar o papel de *gateway* publicando as leituras através do protocolo MQTT, coleta dos dados via um aplicativo *subscriber* e apresentar os dados lidos em uma interface disponível via *web* de forma que se possa acompanhar os dados lidos no veículo a distância, de forma *online* ou historicamente.

Para a validação da proposta, primeiramente foi implementado o componente *gateway*. Para um desacoplamento da funcionalidade de transmissão de dados via MQTT e leitura dos parâmetros via OBDII, estas foram divididas em duas aplicações que se comunicam dentro do mesmo *host* via HTTP. Apesar disso é a atuação conjunta das duas que configuram a capacidade de *gateway* entre os dois protocolos supracitados como demonstra a Figura 22.

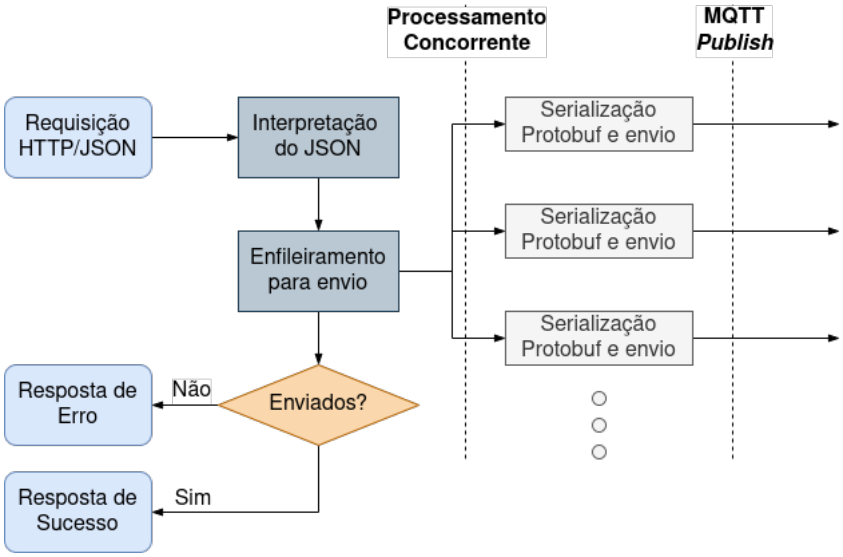
Figura 22 – Conceito de gateway OBDII MQTT.

Fonte: Elaborado pelo autor.

Inicialmente a aplicação de tradução das mensagens para MQTT foi implementada e testada com um cliente HTTP realizando a emulação das leituras de parâmetros (PID) ou códigos de falha (DTC). O fluxo de processamento das mensagens e tradução para MQTT pode ser visto na Figura 23. Para a interface de entrada da aplicação foi escolhida a serialização dos dados em JSON. (FORCE, 2021) por ser de rápido processamento e muito difundida tanto na linguagem Go quanto na linguagem Python, ambas utilizadas no trabalho. Já para a serialização dos dados que são enviados via MQTT para o servidor remoto, por percorrerem uma grande distância e serem transmitidos por rede móvel, foi selecionado o *Protocol Buffers*, que tem como característica sua rapidez de processamento e mensagens com tamanho reduzido. (GOOGLE, 2021b).

Para o envio das mensagens MQTT uma abordagem de eventos foi utilizada, considerando cada leitura um evento e por consequência uma mensagem MQTT em um tópico específico, como expresso pela Figura 24. A biblioteca de código aberto *paho.mqtt.golang* da organização Eclipse foi selecionada, devido a sua maturidade e por ser suportada pelo projeto *paho* da mesma organização.

Figura 23 – Fluxograma do processamento de mensagens do *gateway*.



Fonte: Elaborado pelo autor.

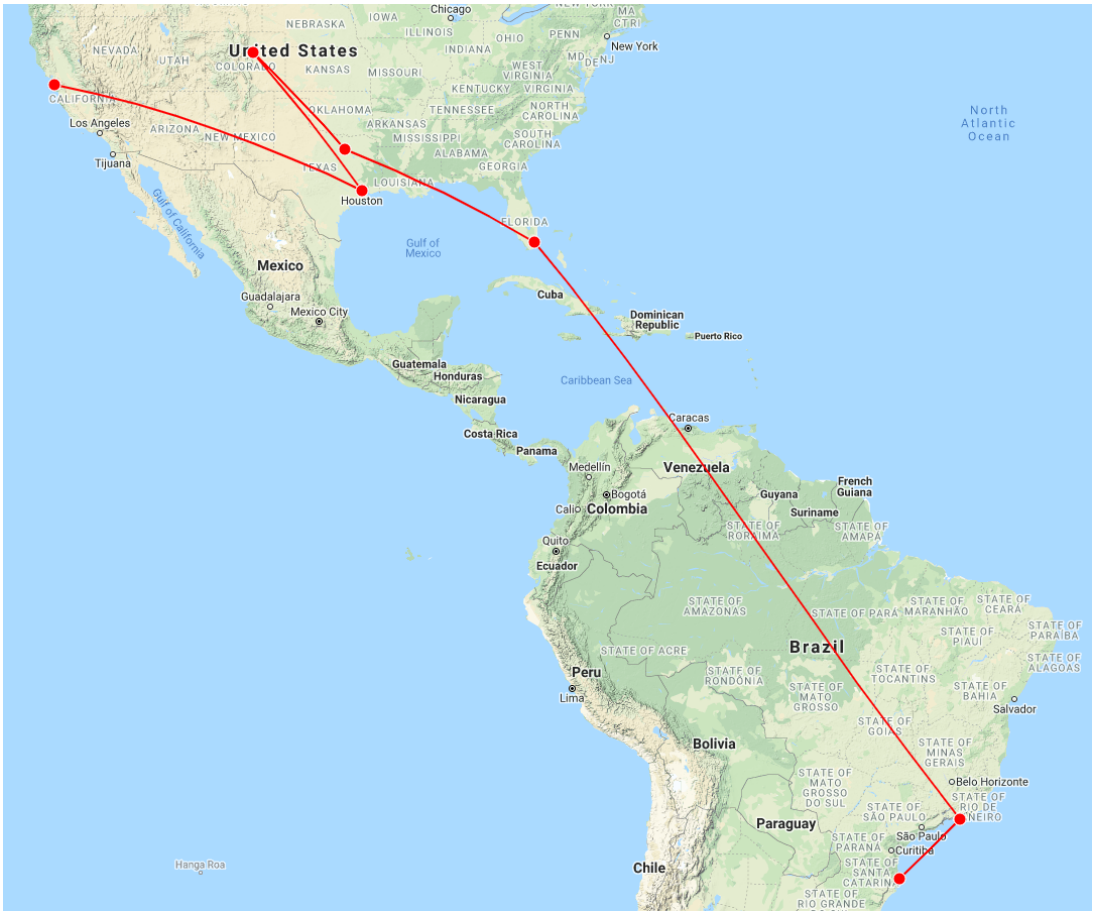
Figura 24 – Tópicos de publicação do gateway MQTT.

	Estrutura	Exemplo Publicação	Exemplo de Assinatura por Veículo
PID	carMon/{placa}/param/{pid}	carMon/ISS-0988/param/10	carMon/ISS-0988/param/+
DTC	carMon/{placa}/dtc/{dtc}	carMon/ISS-0988/dtc/P0104	carMon/ISS-0988/dtc/+

Fonte: Elaborado pelo autor.

No servidor remoto, localizado na região de Dallas no estado do Texas dos EUA e hospedado pela empresa Linode, o serviço EMQ X Broker ®, um *broker* MQTT, aguardava as mensagens da aplicação. Este servidor foi escolhido pelo baixo custo e pela distância da região onde os testes com veículos ocorreriam. Sendo o padrão OBDII e os procolos suportados pelos módulos de comunicação dos automóveis disponíveis somente a equipamentos conectados ao veículo, as distâncias de aproximadamente 11.000 km (Figura 25) que as mensagens do sistema desenvolvido percorreram seriam inviáveis para um equipamento tradicional sem um uso de um *gateway* semelhante ao proposto.

Figura 25 – Caminho aproximado dos pacotes.



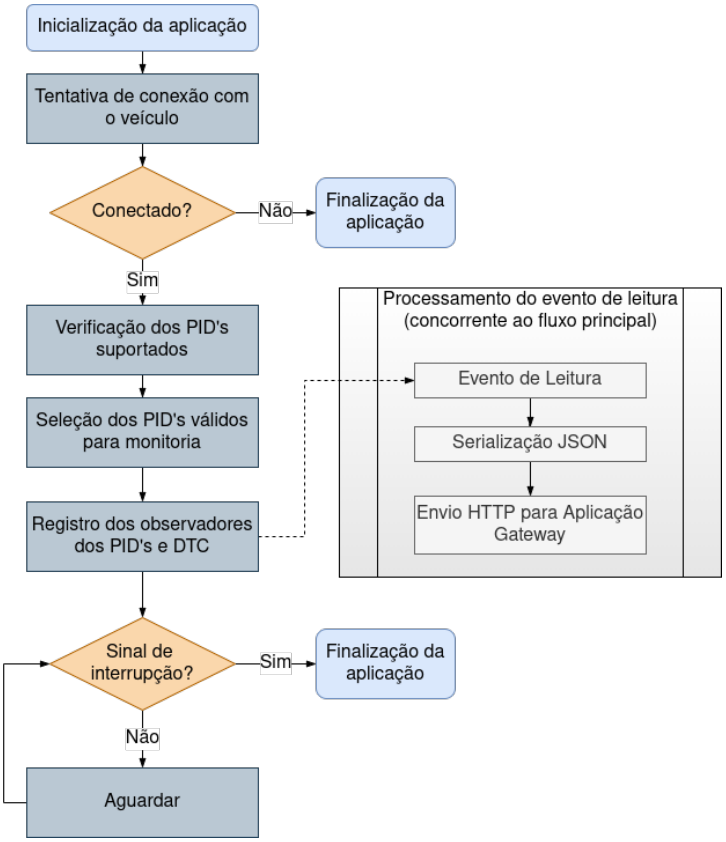
Fonte: Elaborado pelo autor.

Neste mesmo servidor onde o *broker* estava hospedado, a aplicação *subscriber* (implementada na linguagem de programação Go utilizando a mesma biblioteca que o *publisher*) responsável pela persistência dos dados publicados pelo componente de leitura foi hospedada. Esta, em conjunto com o banco de dados relacional, garantem a disponibilidade dos dados para a aplicação de consulta, que no trabalho é uma instância da ferramenta de código aberto Grafana. A implementação foi realizada de forma que cada mensagem MQTT recebida gere um registro em banco de dados, vinculando a placa do veículo e as leituras, tornando possível uma consulta direcionada por veículo. Além disso toda mensagem contém a data e hora em que foi realizada a leitura e ao persistir o dado a aplicação grava a data e hora de registro, permitindo assim realizar uma medida de tempo entre a aquisição da informação no veículo e a persistência no servidor remoto.

Assim que a proposta foi validada a aplicação de leitura dos dados do veículo via OBDII foi implementada, removendo o cliente HTTP utilizado para os testes. O fluxo de controle do componente de leitura dos dados é expresso pela Figura 26. Este foi desenvolvido utilizando a linguagem de programação Python com a biblioteca *python-OBD*. Para os fins de monitoria dos parâmetros de operação do veículo (leitura de sensores e códigos de falha) um algoritmo de seleção dinâmica dos mesmos foi implementado, de acordo com o fluxograma expresso na

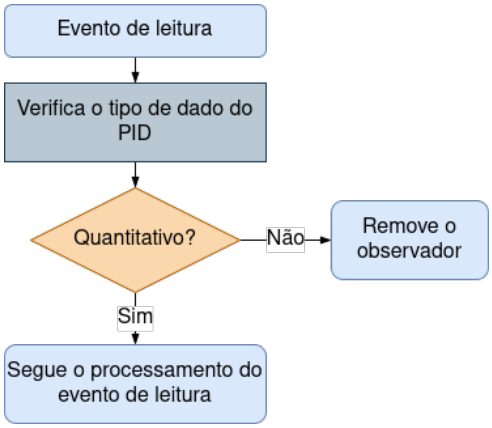
Figura 27. Esta seleção de parâmetros se faz necessária também devido a limitação de velocidade de leituras do circuito integrado ELM327, causando uma frequência maior de leitura dos PID's selecionados ao ignorar PID's que não são interessantes para o caso de acompanhamento das grandezas que envolvem o funcionamento habitual do automóvel.

Figura 26 – Fluxograma do processamento de leituras OBDII.



Fonte: Elaborado pelo autor.

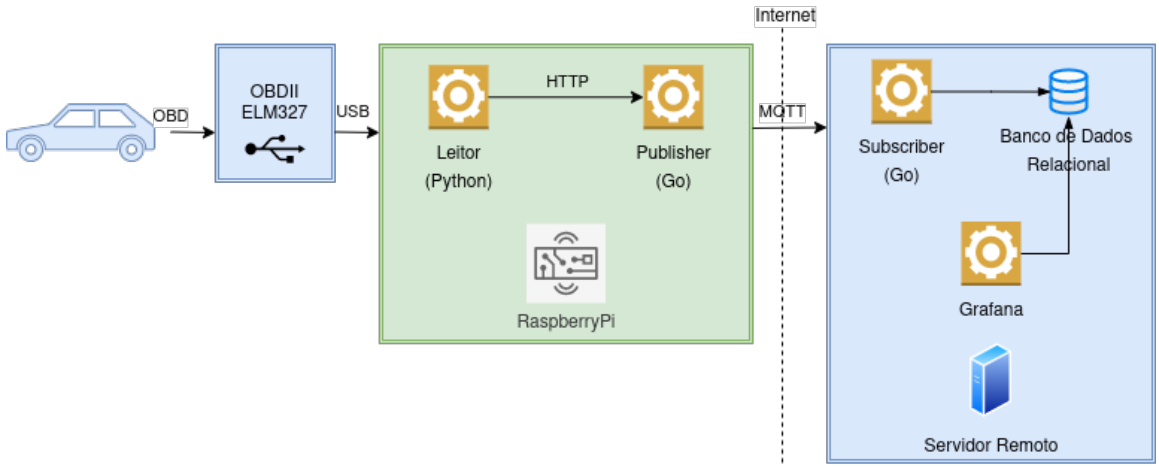
Figura 27 – Fluxograma do processo de seleção de PID's para envio.



Fonte: Elaborado pelo autor.

Para cada leitura de PID efetuada são enviados o identificador, a descrição, a magnitude, a unidade e a data e hora (com resolução de milissegundos) para a aplicação *gateway* via requisição HTTP, que ao receber os dados publica estes pelo protocolo MQTT para o *broker*. A leitura de códigos de falha (DTC) é realizada no inicio da rotina, sendo executado sempre ao ligar o sistema, e de forma semelhante a leitura de parâmetros envia a descrição (se disponível), o código da falha e a data e hora para o *gateway*. Os componentes do processo e seus relacionamentos são demonstrados na Figura 28.

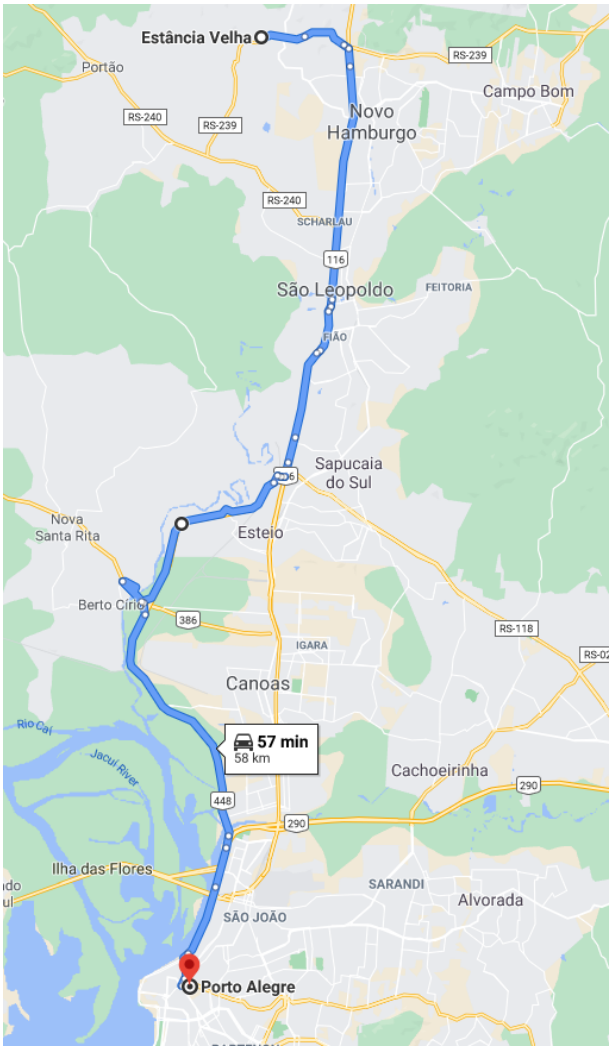
Figura 28 – Componentes do sistema proposto.



Fonte: Elaborado pelo autor.

Para o estudo de caso, foi selecionada uma viagem do município de Estância Velha ao município de Porto Alegre através da Rodovia BR-448 (Figura 29), ambos no estado do Rio Grande do Sul. Sendo o percurso de aproximadamente 58 km de distância percorrida e 57 minutos de duração a amostragem de dados será suficiente para análise quantitativa e de viabilidade do sistema desenvolvido.

Figura 29 – Trajeto selecionado para o estudo de caso.



Fonte: Retirado de Google (2021a).

Como o trajeto é composto de trechos urbanos e estradas, será possível a observação de variadas situações de trânsito, trechos com velocidades na faixa de 0 km/h a 100 km/h. Esse trajeto ainda compõe regiões com uma boa cobertura da rede móvel e regiões com baixa cobertura da rede móvel que influenciam a latência e performance dos componentes fornecendo dados para análise.

5 ANÁLISE DE RESULTADOS

Este capítulo apresenta os resultados alcançados nos testes de compatibilidade e no estudo de caso, assim como a análise destes dados, considerações acerca do comportamento geral do sistema e da ferramenta de monitoria dos dados em tempo real ou histórico.

5.1 Testes de Compatibilidade com Diversos Veículos

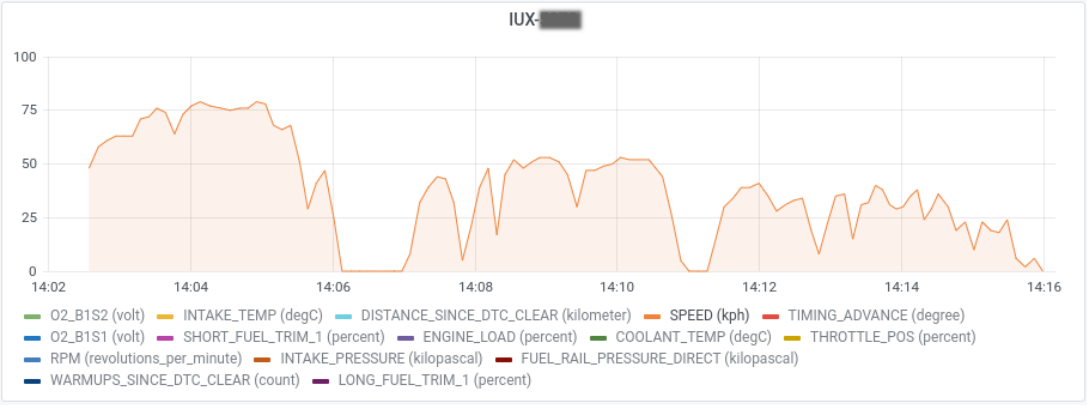
Ao concluir a implementação do processo de leitura dos parâmetros e códigos de falha, foram realizados os testes de compatibilidade com diversos veículos. Estes testes tem um grande valor para a proposta do trabalho, visto a imprevisibilidade da composição de modelos e montadoras em uma frota veícular. Os resultados destes testes são expressos no Quadro 6. A compatibilidade do sistema com o modelo foi determinada pela capacidade do dispositivo de comunicar com sucesso e realizar leituras de parâmetros de operação do motor, estas leituras são demonstradas na Figura 30, onde se apresenta em foco os valores do parâmetro de velocidade, mas estão listados diversos outros suportados pelo veículo testado. As razões para a reprovação na compatibilidade dos automóveis é a inexistência do suporte, por parte do dispositivo ELM327, ao protocolo disponibilizado pela ECU para comunicação, mas nenhum dos veículos testados apresentou problemas no processo do teste de compatibilidade.

Quadro 6 – Compatibilidade do sistema com os veículos testados.

Montadora	**Modelo**	**Motorização**	**Ano**	**Resultado**
Chevrolet	CORSA HATCH MAXX	1.4 L Flex	2011	Aprovado
Renault	DUSTER 16 D 4X2	1.6 L Flex	2013	Aprovado
Volkswagen	NOVO VOYAGE CL MBV	1.4 L Flex	2017	Aprovado
Volkswagen	TIGUAN ALLSPACE CL	1.4 L Turbo FLEX	2018	Aprovado
Volkswagen	FOX 1.0 GII	1.0 L Flex	2011	Aprovado
Volkswagen	FOX 1.0 GII	1.0 L Flex	2013	Aprovado
Ford	KA SE 1.0 HA	1.0 L Flex	2016	Aprovado
Chevrolet	ONIX 1.4MT LTZ	1.4 L Flex	2015	Aprovado
Renault	SANDERO AUT1016V	1.0 L Flex	2013	Aprovado
Chevrolet	SPIN 1.8L AT LTZ	1.8 L Flex	2016	Aprovado
Peugeot	206 SOLEIL	1.6 L	2000	Aprovado
Fiat	IDEA ESSENCE 1.6	1.6 L Flex	2014	Aprovado

Fonte: Elaborado pelo autor.

Figura 30 – *Dashboard* do Grafana de monitoria do veículo durante o teste de compatibilidade (focando no parâmetro de velocidade).



Fonte: Elaborado pelo autor.

5.2 Análise dos Resultados do Estudo de Caso

Tendo implementado todos os componentes da estrutura proposta para o estudo de caso (Figura 28) foi realizado o teste no trajeto previsto efetuando as leituras e as medições dos tempos de processamento das mensagens. Deste teste, foram conduzidas analises para determinar o tempo mínimo, máximo e médio entre as leituras e persistências dos dados, que são expostos na Tabela 1 acompanhados da quantidade total de mensagens.

Observa-se que os tempos de processamento tem uma variação entre 82,60 ms e 4.471 ms. Apesar desta grande variação, sendo o máximo aproximadamente 54 vezes maior que o mínimo, a média se manteve relativamente próxima ao mínimo registrando 209,02 ms. Essa característica é reforçada pela distribuição dos tempos dentro dessa faixa.

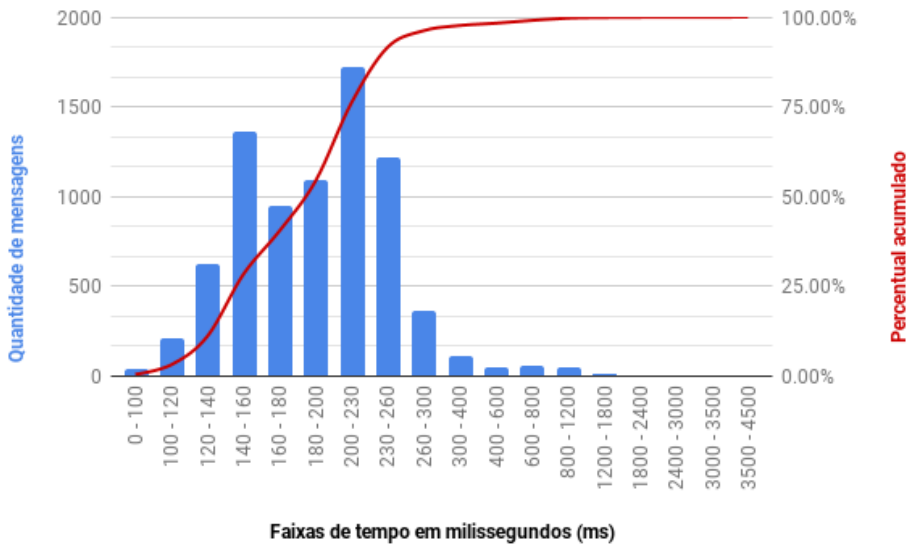
Tabela 1 – Resumo dos tempos de processamento de mensagens em geral durante o estudo de caso.

Mensagens	**Tempo Mínimo [ms]**	**Tempo Máximo [ms]**	**Tempo Médio [ms]**	**Desvio Padrão [ms]**
7888	82,6	4471	209,02	140,53

Fonte: Elaborado pelo autor.

A distribuição do tempo de processamento das mensagens do teste executado como estudo de caso é apresentada na Figura 31. Observando a Tabela 2, que trás dados mais detalhados em relação a distribuição, nota-se a concentração dos tempos ao entorno dos 200 ms, sendo a faixa mais significativa de 140 a 260 ms. Segundo estes mesmos dados detalhados, é possível afirmar que 96,20% das mensagens foram processadas antes do marco de 300 ms e 99,06 % das mensagens antes de 800 ms, mas somente um pequeno grupo de 40 mensagens teve seu tempo entre leitura e persistência menor que 100 ms.

Figura 31 – Histograma do tempo de processamento em geral durante o estudo de caso.



Fonte: Elaborado pelo autor.

Na Tabela 3 são demonstrados o mínimo, máximo e média do intervalo de tempo entre medidas de três parâmetros dos 13 lidos ciclicamente durante o trajeto do estudo de caso. O tempo máximo de intervalo entre leituras foi de 12 segundos, o mínimo de 6 segundos e a média dos intervalos se manteve em 6,62 segundos. A convergência dos valores médios de intervalo de todos os PID's se dá devido a natureza do processo periódico do sistema proposto, que realiza as leituras individualmente através de uma interface de comunicação serial (RS232) que não suporta concorrência.

Esta relação entre a quantidade de parâmetros observados e a frequência de leitura dos mesmos é uma das limitações do sistema proposto, uma vez que existe uma relação inversamente proporcional entre as duas grandezas. A implementação do sistema considerou um processo de seleção dinâmica dos PID's que, entre os parâmetros suportados, escolhe segundo a capacidade do parâmetro de ser expresso quantitativamente.

Devido a característica assíncrona da publicação de mensagens via protocolo MQTT a latência da transmissão dos dados via Internet, excluindo a situação de indisponibilidade de conectividade ou esgotamento dos recursos computacionais do componente embarcado no veículo, não influencia na frequência de leitura dos PID's. Este processo está diretamente ligado com a velocidade de leitura do ELM327 e em última instância da ECU do veículo, visto que estes tem uma característica síncrona em sua comunicação.

Tabela 2 – Distribuição dos tempos de processamento de mensagens em geral durante o estudo de caso.

Faixa de tempo [ms]	**Mensagens**	**Percentual**	**Mensagens (acumulado)**	**Percentual (acumulado)**
0 - 100	40	0,51%	40	0,51%
100 - 120	210	2,66%	250	3,17%
120 - 140	625	7,92%	875	11,09%
140 - 160	1364	17,29%	2239	28,38%
160 - 180	950	12,04%	3189	40,43%
180 - 200	1096	13,89%	4285	54,32%
200 - 230	1724	21,86%	6009	76,18%
230 - 260	1215	15,40%	7224	91,58%
260 - 300	364	4,61%	7588	96,20%
300 - 400	114	1,45%	7702	97,64%
400 - 600	50	0,63%	7752	98,28%
600 - 800	62	0,79%	7814	99,06%
800 - 1200	50	0,63%	7864	99,70%
1200 - 1800	14	0,18%	7878	99,87%
1800 - 2400	5	0,06%	7883	99,94%
2400 - 3000	3	0,04%	7886	99,97%
3000 - 3500	1	0,01%	7887	99,99%
3500 - 4500	1	0,01%	7888	100,00%

Fonte: Elaborado pelo autor.

Tabela 3 – Resumo dos dados de intervalo de leitura de parâmetros.

Parâmetro	**Mensagens**	**Intervalo Mínimo [s]**	**Intervalo Máximo [s]**	**Intervalo Médio [s]**
Velocidade	606	6,00	11,00	6,62
rpm	606	6,00	12,00	6,62
Carga do Motor	606	6,00	12,00	6,62

Fonte: Elaborado pelo autor.

Apesar da natureza não determinística da Internet, que atua na metodologia *best effort*, o sistema proposto, dado a utilização do QoS 1 na publicação das mensagens MQTT no estudo de caso, não deveria sofrer perdas de leituras. Isso se deve a funcionalidade de qualidade de serviço do protocolo, que em seu nível 1 garante pelo menos uma vez a entrega da mensagem. De qualquer forma, por ser uma aplicação voltada a monitoria e acompanhamento da operação de veículos no dia a dia, caso houvesse uma perda de leitura não seria um problema crítico nem adicionaria riscos ao condutor do veículo.

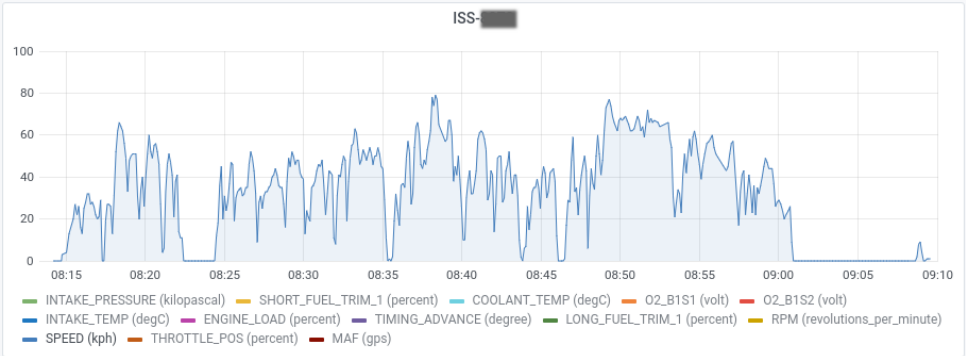
5.3 Monitoria em Tempo Real e Histórico

Uma das propostas de utilização do sistema é a monitoria dos veículos com o dispositivo proposto embarcado. Para este objetivo foi utilizado uma ferramenta chamada Grafana. Esta ferramenta é um software especializado em extrair e apresentar dados persistidos em bancos de dados vinculando as visões a intervalos de tempo como demonstra a Figura 32. O acompanhamento dos PID's durante o trajeto proposto no estudo de caso é apresentado na Figura 33.

Figura 32 – *Dashboard* do Grafana de monitoria demonstrando os últimos 30 dias de dois veículos.

Fonte: Elaborado pelo autor.

Figura 33 – *Dashboard* do Grafana de monitoria do veículo durante o estudo de caso (focando no parâmetro de velocidade).



Fonte: Elaborado pelo autor.

Correlacionando os temas de tempo de processamento de mensagens e monitoria em tempo real, ao se considerar a probabilidade de 99,06 % (Tabela 2) de ter acesso aos dados de uma leitura em até 800 milissegundos, percebe-se que seria possível o acompanhamento dos veículos em seu dia a dia através do sistema proposto. A ferramenta Grafana prevê uma funcionalidade de atualização dos dados apresentados com uma frequência configurada pelo usuário, visto a baixa chance (0,30 %) de ocorrer um tempo maior que 1,2 segundos da leitura a persistência do dado, pode-se afirmar que seria seguro a configuração da ferramenta para 1 segundo entre atualizações, dando ao usuário uma experiência de estar acompanhando em tempo real a operação dos veículos.

6 CONCLUSÃO

O padrão OBDII é um item obrigatório em todos os veículos produzidos nos dias de hoje, mas o acesso as informações disponibilizadas pelo padrão são limitadas a conexões locais com a ECU. Em contrapartida, a monitoria remota de veículos, utilizando as funcionalidades OBDII, tem sido explorada em diversas pesquisas da atualidade, buscando criar uma nova dimensão de conectividade nos automóveis atuais. Para alcançar tal conectividade, dispositivos IoT tem se mostrado de grande valia, dando capacidade computacional e acesso a Internet a uma vasta gama de objetos anteriormente incapazes de comunicar-se.

Este trabalho, que se propôs a desenvolver um sistema que conecte o automóvel a um servidor de monitoramento, demonstrou que através das metodologias, fundamentos e técnicas pesquisadas é possível a proposição e desenvolvimento deste sistema. Com a integração e implementação dos componentes propostos para a solução, foi provada a factibilidade deste. Com os testes emulados e reais pode-se observar que satisfez com sucesso os seus objetivos.

Com o estudo de caso, a aplicabilidade do sistema em uma situação real foi atestada. Neste estudo o dispositivo de leitura e o servidor remoto foram utilizados para o monitoramento em tempo real de um veículo em um trajeto de 58 quilômetros. A aplicação do conceito de *gateway* foi fundamental para a transparência da comunicação entre o dispositivo leitor e o servidor remoto centralizador dos dados, independente da distância existente entre eles. Para conduzir uma avaliação da atuação do sistema, todas as mensagens trocadas tiveram seu tempo de processamento mensurados, assim como o período entre leituras de um mesmo parâmetro.

Através dos dados e medições extraídos do estudo de caso, foi possível determinar que a experiência do usuário do sistema de monitoria seria semelhante ao acompanhamento em tempo real, visto que, mesmo estando a uma distância maior que 11.000 quilômetros, os tempos de processamento das mensagens foram abaixo de 800 milissegundos em 99,06 % dos casos. Além disso a capacidade de consulta de histórico do sistema dá ao usuário a possibilidade de comparação dos parâmetros de funcionamento do motor em diferentes situações e com diferentes condutores.

O sistema tem uma aplicabilidade vasta na monitoria de frotas veiculares, sendo uma fonte rica de informações em relação a saúde e operação dos automóveis da mesma. Outra forma de usufruir das capacidades do sistema é o diagnóstico remoto de veículos, onde o técnico pode realizar a instalação do equipamento e acompanhar à distância o dia a dia do automóvel com seu condutor original.

Como continuidade deste trabalho, em futuros trabalhos, pode-se buscar a implementação de um fluxo de controle das leituras de parâmetros com estímulo partindo do servidor remoto, dando ao sistema de monitoria um papel mais ativo. Uma outra forma de evolução seria a

implementação do equipamento responsável pelas leituras e do *gateway* em um único sistema embarcado especializado em forma de produto, tornando-o mais atrativo comercialmente no tema de custo. Aliando a disponibilidade das informações e sistemas de inteligência artificial, pode-se dar seguimento ao trabalho propondo um processo de predição de falhas veiculares ou de reconhecimento de padrão de direção, sendo o último aplicável tanto para identificação do condutor como para percepção de condução agressiva.

REFERÊNCIAS

AMARASINGHE, M. *et al.* Cloud-based driver monitoring and vehicle diagnostic with OBD2 telematics. *15th International Conference on Advances in ICT for Emerging Regions, ICTer 2015 - Conference Proceedings*, IEEE, p. 243–249, 2016. Citado 2 vezes nas páginas 31 e 33.

BINMASOUD, A.; CHENG, Q. Design of an IoT-based Vehicle State Monitoring System Using Raspberry Pi. *2019 International Conference on Electrical Engineering Research and Practice, iCEERP 2019*, IEEE, n. June, 2019. Citado 4 vezes nas páginas 4, 30, 31 e 33.

BOSCH, L. R. *Eletronic Engine Control Unit*. 2020. Disponível em: <https://www.bosch-mobility-solutions.com/en/products-and-services/passenger-cars-and-light-commercial-vehicles/powertrain-systems/gasoline-direct-injection/electronic-engine-control-unit/>. Citado na página 16.

BOSCH, R. CAN Specification Version 2.0. *Rober Bousch GmbH, Postfach*, v. 300240, p. 72, 1991. Disponível em: <http://esd.cs.ucr.edu/webres/can20.pdf>. Citado 4 vezes nas páginas 16, 18, 19 e 20.

BROY, M. Automotive software and systems engineering. *Proceedings - Third ACM and IEEE International Conference on Formal Methods and Models for Co-Design, MEMOCODE'05*, IEEE, v. 2005, p. 143–149, 2005. Citado na página 13.

CAN-CIA. *CAN Knowledge*. 2020. Disponível em: <https://www.can-cia.org/can-knowledge>. Citado na página 17.

CHAUDHARY, S. *et al.* CRAIoT: Concept, Review and Application(s) of IoT. *Proceedings - 2019 4th International Conference on Internet of Things: Smart Innovation and Usages, IoT-SIU 2019*, IEEE, p. 29–32, 2019. Citado na página 24.

COMPONENTS101. *OBD-II Connector*. 2020. Disponível em: <https://components101.com/connectors/obd2>. Citado na página 23.

CORRIGAN, S. Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN). n. May, p. 1–17, 2002. Disponível em: <www.ti.com>. Citado na página 17.

DORSEMAINE, B. *et al.* Internet of Things: A Definition and Taxonomy. *Proceedings - NGMAST 2015: The 9th International Conference on Next Generation Mobile Applications, Services and Technologies*, IEEE, p. 72–77, 2016. Citado na página 24.

ELM. *OBD*. 2021. Disponível em: <https://www.elmelectronics.com/products/ics/obd/>. Citado na página 37.

FORCE, I. E. T. *The JavaScript Object Notation (JSON) Data Interchange Format*. [S.l.], 2021. Disponível em: <https://datatracker.ietf.org/doc/html/rfc7159>. Citado na página 38.

GANESAN, V. *Internal Combustion Engines*. [S.l.: s.n.], 2012. ISBN 9781259006197. Citado 2 vezes nas páginas 14 e 15.

GOOGLE. *Google Maps*. [S.l.], 2021. Disponível em: <https://www.google.com/maps/dir/Est%C3%A2ncia+Velha,+RS,+93600-000/Rodovia+Do+Parque+(BR+448)+-+Rodovia+do+Parque,+Canoas+-+RS/Porto+Alegre,+RS/@-29.8546594,-51.3264675,11.32z/data=!4m20!4m19!1m5!1m1!1s0x95195b3508e0aceb:0xdbfa23ee28e8d333!2m2!1d-51.1848488!2d-29.6540396!1m5!1m1!1s0x9519642ccd1a0143:0xf38efd0202046c23!2m2!1d-51.2214076!2d-29.8493685!1m5!1m1!1s0x95199cd2566acb1d:0x603111a89f87e91f!2m2!1d-51.2176584!2d-30.0346471!3e0s>. Citado na página 43.

GOOGLE. *Protocol Buffers*. [S.l.], 2021. Disponível em: <https://developers.google.com/protocol-buffers>. Citado na página 38.

HEYWOOD, J. *Internal Combustion Engines Fundamentals*. [S.l.: s.n.], 1988. ISBN 007028637X. Citado 2 vezes nas páginas 14 e 15.

HIVEMQ. *MQTT Essentials: Part 1*. 2020. Disponível em: <https://www.hivemq.com/blog/mqtt-essentials-part-1-introducing-mqtt>. Citado 5 vezes nas páginas 25, 26, 27, 28 e 29.

KHARCHE, P. UDS Implemetation for ECU I / O Testing. p. 137–140, 2018. Citado na página 16.

KOWALIK, B. Introduction to car failure detection system based on diagnostic interface. *2018 International Interdisciplinary PhD Workshop, IIPhDW 2018*, IEEE, p. 4–7, 2018. Citado 2 vezes nas páginas 32 e 33.

KULKARNI, P.; RAJANI, P. K.; VARMA, K. Development of On board diagnostics (OBD) testing tool to scan emission control system. *Proceedings - 2nd International Conference on Computing, Communication, Control and Automation, ICCUBEA 2016*, IEEE, p. 1–4, 2017. Citado na página 21.

MALEKIAN, R. *et al.* Design and Implementation of a Wireless OBD II Fleet Management System. *IEEE Sensors Journal*, IEEE, v. 17, n. 4, p. 1154–1164, 2017. ISSN 1530437X. Citado 5 vezes nas páginas 20, 21, 31, 32 e 33.

MINN, H.; ZENG, M.; BHARGAVA, V. Towards a definition of the Internet of Things (IoT). *IEEE Internet Initiative*, p. 1–86, 2015. Citado 2 vezes nas páginas 24 e 25.

NAIK, P. *et al.* An automotive diagnostics, fuel efficiency and emission monitoring system using CAN. *2017 International Conference on Big Data, IoT and Data Science, BID 2017*, v. 2018-Janua, p. 14–17, 2018. Citado na página 16.

OASIS Open. MQTT Version 3.1.1. *OASIS standard*, n. December, p. 1–81, 2015. Disponível em: <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html>. Citado 4 vezes nas páginas 25, 26, 27 e 28.

OBD python. *Command Tables*. [S.l.], 2020. Disponível em: <https://python-obd.readthedocs.io/en/latest/Command%20Tables/>. Citado na página 22.

OBD python. *Getting Started*. [S.l.], 2020. Disponível em: <https://python-obd.readthedocs.io/en/latest/>. Citado na página 35.

RAYES, A.; SALAM, S. *Internet of Things From Hype to Reality*. [S.l.: s.n.], 2019. ISBN 9783319995151. Citado na página 25.

ROUSE, M.; GILLIS, A.; ROSENCRANCE, L. *internet of things (IoT)*. 2020. Disponível em: <https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT>. Citado na página 24.

SAWANT, R. (OBD) Device for Cars. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, IEEE, p. 1–4, 2018. Citado 3 vezes nas páginas 20, 21 e 22.

SHELBY, Z.; HARTKE, K.; BORMANN, C. *The Constrained Application Protocol (CoAP)*. [S.l.], 2014. Disponível em: <https://tools.ietf.org/rfc/rfc7252>. Citado na página 25.

SIM, A. X. A.; SITOHANG, B. OBD-II standard car engine diagnostic software development. *Proceedings of 2014 International Conference on Data and Software Engineering, ICODSE 2014*, IEEE, 2014. Citado 4 vezes nas páginas 20, 21, 22 e 23.

SUTAR, A. D.; SHINDE, S. B. ECU diagnostics validator using CANUSB. *Proceedings of the International Conference on Inventive Computing and Informatics, ICICI 2017*, n. Icici, p. 856–860, 2018. Citado na página 16.

TILS, V. V. Trends and challenges in automotive electronics. *Proceedings of the International Symposium on Power Semiconductor Devices and ICs*, v. 2006, p. 18–20, 2006. ISSN 10636854. Citado 2 vezes nas páginas 14 e 16.

TUMMALA, R. *et al.* Industry consortium for new era of automotive electronics with entire system-on-package vision at Georgia Tech. *20th European Microelectronics and Packaging Conference and Exhibition: Enabling Technologies for a Better Life and Future, EMPC 2015*, IMAPS Europe, n. September, p. 1–5, 2016. Citado na página 13.

Vijaya Shetty, S. *et al.* Iot based automated car maintenance assist. *2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017*, v. 2017-Janua, p. 501–508, 2017. Citado 3 vezes nas páginas 30, 31 e 33.