
**UNIVERSIDADE FEDERAL DO CEARÁ**

**CAMPUS QUIXADÁ**

**TECNÓLOGO EM REDES DE COMPUTADORES**

**CÁSSIO DE OLIVEIRA LEITAO**

# REDSEC – SISTEMA DE MONITORAMENTO E IMOBILIZAÇÃO PARA AUTOMÓVEIS

**QUIXADÁ**

**2022**

CÁSSIO DE OLIVEIRA LEITAO

# REDSEC – SISTEMA DE MONITORAMENTO E IMOBILIZAÇÃO PARA AUTOMÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de tecnólogo em Tecnologia em Redes de Computadores.

Área de concentração: Computação.

Orientador: Prof. Me. Marcos Dantas Ortiz

QUIXADÁ

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

---

L548r Leitão, Cássio de Oliveira.
REDSEC : sistema de monitoramento e imobilização para automóveis / Cássio de Oliveira Leitão. – 2022.
50 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2022.
Orientação: Prof. Me. Marcos Dantas Ortiz.

1. Internet das coisas. 2. Automóveis. 3. Sistemas embarcados (computadores). 4. Monitoramento. I. Título.

CDD 004.6

---

CÁSSIO DE OLIVEIRA LEITAO

# REDSEC – SISTEMA DE MONITORAMENTO E IMOBILIZAÇÃO PARA AUTOMÓVEIS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Tecnologia em Redes de Computadores do Campus Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de tecnólogo em Tecnologia em Redes de Computadores.

Aprovado em: _/_/_

BANCA EXAMINADORA

_____

Prof. Me. Marcos Dantas Ortiz (Orientador)

Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Francisco Helder Candido dos Santos Filho

Universidade Federal do Ceará (UFC)

_____

Prof. Dr. Wagner Guimarães Al-Alam

Universidade Federal do Ceará (UFC)

## RESUMO

Aquisição de automóveis seja novo ou seminovo está cada vez mais comum nas sociedades contemporâneas, contudo como forma de proteger o investimento feito, alguns cuidados são necessários. Um dos riscos que o dono pode sofrer é o furto/roubo do veículo, o risco de ter perda total. O presente trabalho tem como objetivo desenvolver uma solução tecnológica de segurança (um protótipo denominado de REDSEC) para rastreio de automóveis, o qual envia para um servidor *broker* a localização do veículo, sendo capaz, se necessário, também de imobilizar o veículo remotamente. O REDSEC foi desenvolvido como uma solução *IoT (Internet of Things)* que utiliza o protocolo *MQTT* para comunicação com o sistema embarcado.Foram realizados testes onde foi obtido uma boa média de valores de latência e zero perda de pacotes, após os testes feitos, demonstrando que a rede celular 2G atende os requisitos para ser o meio de comunicação para o sistema embarcado. Espera-se que no futuro o dispositivo seja aperfeiçoado, permitindo que além da localização e imobilização remota do veículo, ele seja capaz de monitorar e enviar ao servidor códigos de falhas do veículo para servirem de entrada ao servidor de manutenção preventiva.



**Palavras chaves:** Internet das coisas. Automóveis. Sistemas embarcados (computadores). Monitoramento

## ABSTRACT

Car acquisition, whether new or used, is increasingly common in contemporary societies, however, as a way of protecting the investment made, some care is needed. One of the risks that the owner can suffer is the theft / theft of the vehicle, the risk of having a total loss. The present work aims to develop a technological security solution (a prototype called REDSEC) for tracking cars, which sends the location of the vehicle to a broker server, being able, if necessary, also to immobilize the vehicle remotely. REDSEC was developed as an IoT (Internet of Things) solution that uses the MQTT protocol to communicate with the embedded system. Tests were carried out where a good average of latency values and zero packet loss were obtained, after the tests performed, demonstrating that the 2G cellular network meets the requirements to be the means of communication for the embedded system. It is expected that in the future the device will be improved, allowing that, in addition to the location and remote immobilization of the vehicle, it is able to monitor and send the vehicle fault codes to the server to serve as input to the preventive maintenance server.



**Keywords:** IoT. Automobiles. Embedded systems (computer). Monitoring.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

**SUMÁRIO**

# 1. INTRODUÇÃO

Aquisição de automóveis seja novo ou seminovo está cada vez mais comum nas sociedades contemporâneas, contudo como forma de proteger e evitar prejuízos alguns cuidados são necessários. Um prejuízo que o dono pode sofrer é o furto/roubo do veículo, com possível perda total.

Com base em dados da Gerência de Estatística e Geoprocessamento (Geesp) da Superintendência de Pesquisa e Estratégia de Segurança Pública do Estado do Ceará (Susesp/CE), o número de roubos de veículos no primeiro quadrimestre de 2019 foi de 1.726, um número mais baixo se comparado com o mesmo período em anos anteriores. Estratégias de combate ao crime resultaram nessa redução. Comparado ao ano de 2018, que foram registrados 3.414 roubos, o primeiro quadrimestre de 2019 registrou uma queda de 49%, cerca de 1.274 veículos a menos[1].

Embora tenha tido uma redução desse tipo de crime por conta das estratégias de combate por parte da segurança pública, o número ainda é alto e nem sempre o proprietário consegue evitar a ação ou recuperar o veículo posteriormente.

Este trabalho apresenta o desenvolvimento de uma solução *IoT* de monitoramento, segurança e automação de veículos – protótipo denominado de REDSEC, integrando algumas funcionalidades desenvolvidas em Maicon (2017) como o rastreio geográfico. O objetivo principal deste trabalho é o desenvolvimento de uma solução *IoT* que permite o rastreio e desligamento, caso necessário, do sistema de injeção eletrônica de automóveis, utilizando plataformas eletrônicas como *Arduino, ESP8266 ou ESP32*, desse modo vindo a ajudar proprietários na segurança de seus veículos.

Na seção 1 deste trabalho são abordados o objetivo geral e os objetivos específicos. Na seção 2 são abordados os conceitos chaves que compõem a fundamentação teórica deste trabalho, como por exemplo: *ESP32*, Internet das coisas, Sistemas embarcados, entre outros. Na seção 3, são apresentados e comparados os trabalhos relacionados ao REDSEC. Na seção 4, são apresentados os os procedimentos metodológicos, sendo descritos os passos para desenvolvimento da solução proposta . Na seção 5 – intitulada de análise da perda de dados – são apresentados os resultados dos experimentos realizados. Por fim, são elencados os trabalhos futuros com para melhorias a serem desenvolvidas no REDSEC.

[1] Disponível em: < https://www.ceara.gov.br/2019/05/09/roubos-de-veiculos-no-ceara-registram-menor-numero-em-oito-anos/> Acesso em: 24 set. 2019.

## 1.1. OBJETIVOS

A seguir são apresentados o objetivo geral e objetivos específicos deste trabalho.

### 1.1.1 Objetivo geral

- Desenvolver uma solução *IoT* de segurança para rastreio e imobilização de automóveis.

### 1.1.2 Objetivos específicos

- Desenvolvimento de um dispositivo *IoT* para controle dos veículos
- Monitoramento e visualização remota da localização de veículos;
- Imobilização remota de veículos.

# 2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é a abordagem de alguns conceitos fundamentais e necessários para a concretização do presente trabalho, como Internet das Coisas que trata da conexão dos objetos(automóvel) com as pessoas(proprietário), os protocolos de comunicação, como será feita a troca de mensagens entre o carro e o sistema *web*, automação elétrica que trata do controle dos componentes elétricos do carro, o sistema embarcado que se trata do equipamento instalado no veículo, os *hardwares* que compõem esse sistema embarcado, a injeção eletrônica que terá a bomba de combustível que é um atuador, controlado e o *Node-Red* que é a ferramenta visual e de programação do sistema w*eb*..

## 2.1 Internet das Coisas

Internet das Coisas – *Internet of Things (IoT)*- é um paradigma da computação que surgiu nos anos 90 quando Kevin Ashton apresentou para executivos da *Procter & Gamble*, a ideia de etiquetar-se eletronicamente produtos da empresa com identificadores *RFID*. Ela se caracteriza como uma infraestrutura de comunicação que permite o fornecimento de serviços entre “coisas” físicas e virtuais.

Mota e Batista (2013) acreditam que:

> Em um futuro próximo, qualquer “coisa” (*thing*) poderá ser endereçada na grande rede. A Internet, então, tornarse-á a Internet das coisas (*Internet of things - IoT*). As comunicações serão concebidas não apenas entre humanos, mas também entre humanos e coisas e entre coisas sem a interação com seres humanos ( p. 297).

O futuro previsto por Mota e Batista nos anos 2010 está se concretizando pouco a pouco, à medida que, atualmente, são bilhões de coisas conectadas à internet. Contudo, essa expansão conta com a necessidade ampliação de redes móveis de alta velocidade e mais estáveis, como o caso da 5G , que ainda está muito restrita a determinados espaços.

A conectividade e interação são dois elementos indispensáveis e fundamentais quando se trata de internet das coisas, geralmente a partir da ação humana.

Segundo Valente (2001):

> A Internet das Coisas é um paradigma que tem como objetivo criar uma ponte entre acontecimentos do mundo real e as suas representações no mundo digital. O objetivo é integrar o estado das Coisas que constituem o nosso mundo em aplicações de software, beneficiando do contexto onde estão instaladas (VALENTE, 2011, p. 1).

A Internet das coisas para o autor apresenta-se como um paradigma, tendo como finalidade criar uma ponte entre algo do mundo real e o mundo digital. Esses dois mundos podem integrar-se, resultando em transformações e mudanças. Importante frisar que a conectividade entre os ambientes são atributos de relevância para o autor.

Somado a perspectiva apresentada por Valente (2001) do conceito de IoT, este trabalho traz a citação de MARTINS E ZEM (2015) onde não apenas esclarece como a IoT funciona, mas faz uma comparação essencial ao diferenciá-la do conceito de Comunicações Machine-to-Machine:

> A ideia consiste em poder conectar uma variedade de objetos (sensores, tags RFID, smartphones, computadores, e até objetos de uso mais comum) entre si. Estes objetos geram um fluxo de dados e a conexão permite que eles transmitam estes dados para outros objetos no meio, formando assim uma Internet de coisas. Este termo confunde-se com o termo 'Comunicações Machine-to-Machine‘ (M2M Communications), que trata da comunicação máquina a máquina, ou seja, entre dois ou mais dispositivos, sem um ‘usuário final’ envolvido no processo da troca de informações (p. 65).

Um dos objetivos da existência e uso da internet das coisas é ser benéfica para aqueles que a utilizam. Ressalta-se que a aplicabilidade e benefícios desse tipo de tecnologia apresenta amplitude para diversas áreas. Podendo ser usada tanto para uso privado ou público, no ambiente doméstico ou de trabalho.

Um dos ambientes que se utiliza desse modelo de tecnologia são as indústrias, que inserem máquinas monitoradas para execução de trabalhos, essas máquinas possuem sensores

e inteligência para atender e realizar os comandos. Para muitas empresas do setor industrial, o uso das máquinas gera maior desenvolvimento econômico, pois há produtividade e agilidade.

Muitos setores da sociedade se favorecem com a ideia proposta pela internet das coisas de comunicação entre dispositivos, podemos citar as lâmpadas inteligentes como no caso da *Philips Lighting*, a pulseira inteligente *Nike+ FuelBand SE*, os carros elétricos e autônomos da *Tesla Motors*. Nota-se que as possibilidades de utilização que podemos da *IoT* são inúmeras, seja para gerar conforto, segurança, praticidade ou lucrar. A *IoT* se adequa às diversas finalidades, ambientes, e obtém resultados positivos. Por essa razão, afirma-se que *IoT* tem três Cs: comunicação, controle/automação e custos reduzidos.

Este trabalho se desenvolveu a partir desta perspectiva à medida que construiu um dispositivo que se comunica diretamente com a internet sem a necessidade de um dispositivo intermediário para essa comunicação. Adotando assim o modelo *device-to-cloud* de comunicação da internet das coisas.

Figura 1 – Internet das Coisas

Fonte: Harvard Business Review Brasil[2]

## 2.2 Protocolos de comunicação para *IoT*

Ao nos debruçarmos a respeito da *IoT* é necessário conhecer modelos de protocolos que são adequados e eficientes para cada usuário e serviço, pois como nos esclarece Mazzer e Parreira (2018) não podemos ser taxativos em dizer que um protocolo é melhor que o outro, mas qual deles é mais apropriado para o cenário de aplicação proposto, em alguns casos,

[2]Disponível em: < https://hbrbr.uol.com.br/estrategia-internet-das-coisas/>Acesso em: 19 maio. 2019.

podemos até escolher mais de um. Para conhecer e selecionar eficientemente um protocolo também se faz necessário o conhecimento sobre o conceito e funcionamento de cada um.

Os principais protocolos propostos para a comunicação *M2M* com foco em ambientes limitados (MAZZER E PARREIRA, 2018) são: *MQTT – Message Queue Telemetry, CoAP – Constrained Application Protocol* e o *AMQP – Advanced message Protocol*. Entretanto, neste estudo abordaremos em profundidade o *MQTT* e *CoAp* por serem os mais conhecidos e utilizados.

Vale ressaltar que o protocolo *HTTP - Hypertext Transfer Protocol* por mais que seja vastamente utilizado, sobretudo na *Web* atual, não é viável para os sistemas *IoT*, já que seus pacotes podem ocupar muito espaço e geralmente dispositivos *IoT* possuem pouca memória e processamento.

O *Coap* desenvolvido pelo *Constrained RESTful Environments – CoRE* do *Internet Engineering Task Force - IETF* foi desenvolvido baseado no *HTTP*, porém não como um substituto dele, mas como uma forma de otimizar os protocolos em "dispositivos com potência e capacidade de processamento limitados, geralmente aplicado a objetos inteligentes em ambientes *IoT*" (ibid, p.3).

*MQTT* traduzindo "ao pé da letra", significa: transporte de telemetria da fila de mensagens, isto é, um protocolo de troca de mensagens. Foi desenvolvido nos anos 99 por *Andy Stanforf-Clark* do *IBM* e *Arlen Nipper* da *Eutoech* para conectar oleodutos por conexão via satélite nas plataformas de petróleo (AQUINO, 2018). Inicialmente o foco era a supervisão e aquisição de dados na área de sistemas, com o decorrer do tempo e aprimoramento tecnológico, esse modelo evoluiu e vem sendo aplicado em vários cenários.

Assim como o protocolo *HTTP*, o *MQTT* também se encontra na camada de aplicação do modelo *OSI*, a maior diferença entre eles dá-se pelo tamanho do *payload* , pois o *HTTP* é maior, o que torna inviável o uso em conexões de baixa qualidade e em dispositivos menores. Outras vantagens do *MQTT* comparando-se ao *HTTP* são: permite uma conexão de 1 para N, possibilitando um servidor atender milhares de clientes; cabeçalho menor; maior número de níveis de *QoS*.

Reforça-se que os benefícios de utilizar o protocolo *MQTT* são segurança, qualidade de serviço, facilidade de implementação, flexibilidade e simplicidade. Da mesma maneira, o

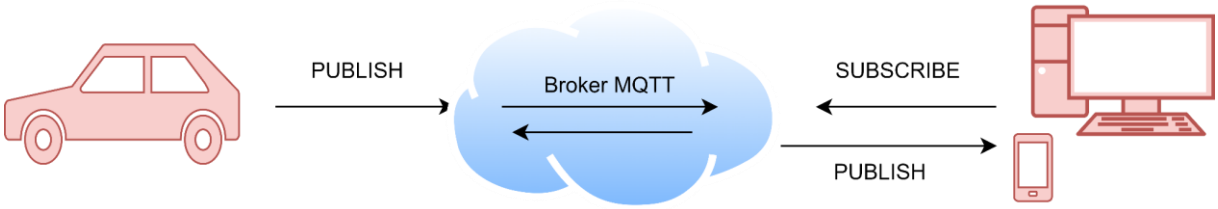
*CoAP* atua, pois utiliza menos volume de tráfego e baixa sobrecarga de cabeçalho (AQUINO, 2018; MAZZER E PARREIRA, 2018).

Ao contrário do *Coap* que usa *Request-Reponse*, em que um cliente solicita informações a um servidor e este responde a requisição do cliente - semelhante ao HTTP, porém de forma assíncrona- o *MQTT* utiliza o método *Publish-Subscribe*:

> Quando o cliente ou elemento da rede conhecido também como subscriber deseja receber uma determinada informação, ele dá um subscribe no tópico de interesse através de uma requisição para um outro elemento da rede que é conhecido como Broker que atua como um intermediário no processo de comunicação. Subscribers que desejam publicar algum tipo de informação o fazem através do Broker também, enviando-lhe as informações que possui e este fica responsável por rotear a mensagem até o destinatário. É importante frisar que geralmente o fluxo de comunicação se dá majoritariamente de quem publica informações como sensores por exemplo, para quem assina e deseja receber aquela informação que são os mais variados clientes como por exemplo aplicações rodando na nuvem ou um simples Raspberry (SILVA; CARVALHO; NAZARIO, 2018, p.1).

Na figura 2, temos um exemplo de *Publish-Subscribe*, onde o veículo faz um *publish* para o *broker* que encaminha os dados ao cliente .

Figura 2 – Exemplo funcionamento Protocolo *MQTT*




Fonte: Autor

Resumidamente, compreendemos que é um protocolo que se conecta a um *broker*, responsável por receber mensagens, filtrar e direcionar aos clientes interessados, ou seja, aqueles que estão escritos nos tópicos. Importante dizer que o *broker* deve ser escalável, integrável, monitorável e tolerante a falhas (MARQUES E KNIESS, 2013).

Por mais que *Coap* e *MQTT* sejam parecidos e viáveis para *IoT*, o presente estudo optou por utilizar o *MQTT* por implementar *QoS*, garantindo a retransmissão de pacotes, e também por ser mais utilizado de forma mais abrangente e possui inúmeros estudos ligados ao seu uso em rastreadores de veículos.

A seguir apresentamos uma breve tabela comparativa entre esses dois protocolos e as vantagens da nossa escolha pelo *MQTT* para esta pesquisa:

Tabela 01 – Comparativo entre os protocolos

| MQTT | COAP |
|---|---|
| Utiliza baixo volume de tráfego, baixa sobrecarga de cabeçalho e interessante para dispositivos com recursos limitados. | Utiliza baixo volume de tráfego, baixa sobrecarga de cabeçalho e interessante para dispositivos com recursos limitados. |
| Funciona bem com conexões via satélites; | Não foi pensado inicialmente para conexões via satélites; |
| Aplicável em dispositivos que pedem uma NAT (Network Address Translation), como é o caso do desenvolvido neste trabalho. | Apresenta problemas na comunicação com dispositivos que estão atrás de uma NAT (Network Address Translation). |
| Comunicação de um para muitos, ou seja, o dispositivo pode conversar com mais de um cliente (MARQUES E KNIESS, 2013). | Comunicação de um para um, conversa apenas com um cliente. |
| Mensagem pode ser pensada em quatro níveis de serviço: QoS 0 (at most once)- não se tem confirmação de entrega;<br>QoS 1 (at least once) – existe confirmação de entrega;<br>QoS 2 (exactly once) – Existe confirmação de recebimento e confirmação de recebimento de confirmação. (BARROS, 2015). | Mensagem não confiável (message unreliability): UDP não garante que datagramas foram entregues. Embora empregue o método CON-ACK para suas mensagens, isso não verifica se foi recebido em sua totalidade e decodificado corretamente (SILVA; CARVALHO; NAZARIO, 2019). |

Fonte: Autor

Desta forma, compreendemos que os dois protocolos têm suas particularidades e são possíveis de serem utilizados em dispositivos menores na *IoT*, contudo pelos motivos enumerados na tabela anterior optamos aqui por utilizar o *MQTT* ao *Coap*.

### 2.3 Automação elétrica

Segundo Jardini (1997), sistemas digitais são utilizados para a automação da geração, transmissão e distribuição de energia elétrica. As automações elétricas tiveram início com o objetivo de reduzir a mão de obra humana nos processos industriais. O termo automação elétrica é usado para designar estes sistemas, que são utilizados para monitoramento, comando, controle e proteção dos vários componentes do sistema elétrico. Este trabalho trata-

se da automação para controle de componentes elétricos de um automóvel, caracterizando-se assim como uma automação elétrica.

Neste trabalho, o atuador vai controlar a bomba de combustível do veículo, responsável por levar o combustível do tanque do veículo para os bicos injetores do motor do automóvel.

## 2.4 Sistemas embarcados

Sistemas embarcados são computadores: possuem memória, processador, interfaces de entrada e saída, mas ao contrário dos computadores convencionais, desempenham uma tarefa específica. Esse dispositivo possui sistemas que contém todas as funcionalidades a serem executadas pelo seu processador (microcontrolador).

De acordo com Oliveira e Andrade (2010) sistemas embarcados podem ser definidos como:

> Sistemas que possuem uma capacidade de processamento de informações vinda de um *software* que está sendo processado internamente nessa unidade. Ou seja, o software está embarcado na unidade de processamento (denominado de *firmware...)*. ( p.25)

Observa-se nas palavras dos autores uma definição concisa e precisa sobre os sistemas embarcados que permite compreender melhor as escolhas teóricas e práticas deste trabalho.

Logicamente, por se tratar de um sistema que faz parte da *IoT*- Internet das coisas - esse sistema interage com o ambiente e com o usuário que o manuseia, ou seja, há uma comunicação direta e uma conexão entre o objeto, meio e indivíduo. Essa é uma das razões para que esse sistema seja de fácil acesso, aplicabilidade, além de ter funções específicas.

Uma máquina importante e que está diariamente no nosso cotidiano e utiliza sistemas embarcados é o automóvel. Os carros usam dispositivos que possuem este tipo sistema, isso demonstra como essa tecnologia contribui para o contexto social e econômico dos sujeitos.

Assim, neste trabalho é proposto um sistema embarcado para prover o rastreamento e em caso de furto ou roubo oferecer a possibilidade de imobilizar o veículo.

## 2.5 *Hardwares* utilizados

A seguir apresentamos alguns dos componentes utilizados na construção do protótipo do presente trabalho.

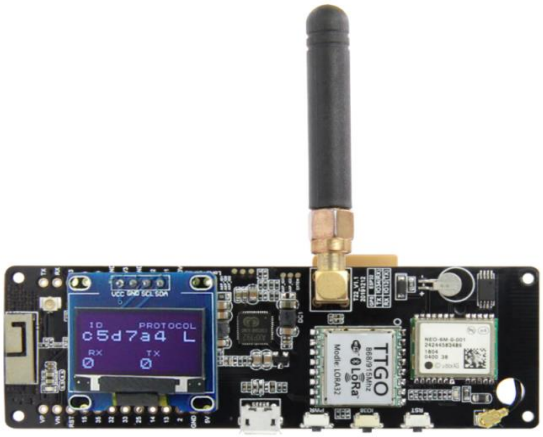
### 2.5.1 *ESP32 TTGO T-BEAM V 1.1 LORA 915MHZ WIFI BLUETOOTH GPS NEO-6M*

O *ESP32 TTGO T-BEAM V 1.1* é uma placa de desenvolvimento com um microcontrolador de baixo custo e alta performance que possui as seguintes especificações técnicas:

- *WI-FI*
- *BLUETOOTH*
- *8 MB PSRAM*
- *4MB FLASH*
- 3D ANTENA
- *LORA*
- MÓDULO *GPS NEO-6M*

Na figura 03, podemos visualizar o *ESP32 TTGO T-BEAM V 1.1*, utilizado neste trabalho para fazer a comunicação do sistema web com o veículo.

Figura 3 – *ESP32 TTGO T-BEAM V 1.1*




Fonte: Lilygo[3]

### 2.5.2 Placas *Shields*

As *Shields* são placas de circuito impresso que podem ser conectadas na parte superior do *Arduino* ou *ESP32*, estendendo seus recursos. As diferentes *Shields* seguem a mesma filosofia do kit de ferramentas original: são fáceis de montar e baratas de produzir.[4]

[3] Disponível em: < http://www.lilygo.cn/claprod_view.aspx?TypeId=62&Id=1281&FId=t28:62:28/> Acesso em: 01 abril. 2021.
[4] Disponível em: <https://www.arduino.cc/en/Main/arduinoShields/> Acesso em: 03 dez. 2018.

Através dessa conexão entre o *Arduino* ou *ESP32* e as *Shields* é possibilitado a execução de atividades específicas como abrir/fechar circuitos, conectar-se à internet, obter informações de sensores entre outros, isso se deve os circuitos contidos nas *Shields* possuírem alguns componentes com funcionalidades que o *Arduino ou ESP32* não possuem. Neste trabalho, as *Shields* são usadas para conectar o carro à Internet, abrir/fechar circuitos de atuadores elétricos do automóvel como a bomba de gasolina.

### 2.5.3 Módulo *SIM800l*

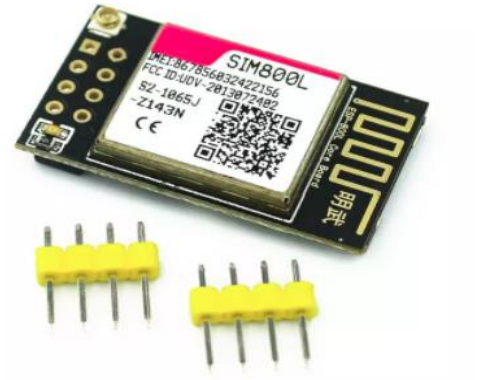
O módulo *SIM800L* é utilizado para comunicação via dados *GSM/ GPRS*, necessita de um chip de operadora de telefonia móvel para comunicação. O módulo pode ter suas ações controladas por diversos tipos de microcontroladores, como o *Arduino, ESP32*, por exemplo.

**Características do SIM800L:**

- *Quad-band* 850/900/1800/1900 MHz
- Conecta a qualquer rede *GSM* 2G
- Capaz de fazer e receber ligações
- Envio e recebimento de sms
- Envio e recebimento de dados *GPRS* (TCP / IP, http, etc.).
- Tensão de Operação: 3.7 > V < 4.2
- Consumo até 2 amperes

Na figura 4, encontra-se uma *Shield SIM800L*

Figura 4 – SIM800L




Fonte: Aliexpress[5]

### 2.5.4 Módulo Relé *Shield*

[5] Disponível em: < https://pt.aliexpress.com/item/32924278790.html/> Acesso em: 01 abril. 2021.

O módulo relé *Shield* é uma placa de interface de relé, que pode ser controlada diretamente por uma ampla gama de microcontroladores como *Arduino*, *ESP32*, *PIC*, *ARM*, *PLC*, etc. É capaz de controlar vários aparelhos e equipamentos, amplamente utilizado para todo tipo de controle, por exemplo, no setor industrial, controle *PLC* ou controle de casa inteligente.[6] O módulo é utilizado neste projeto para ligar/desligar a bomba de combustível do veículo.

Na figura 5, temos um módulo Relé *Shield* responsável por ativar e desativar a bomba de combustível do veículo.

Figura 5 – Módulo Relé *Shield*

Fonte: masterwalkershop [7]

## 2.6 Injeção Eletrônica

A Injeção Eletrônica utilizada em automóveis substituiu os antigos carburadores e foi desenvolvida com o intuito de ter uma melhor queima de combustível, obtendo economia, assim como também uma redução dos gases tóxicos resultado desta queima, os quais são lançados no ambiente. A injeção eletrônica é responsável por controlar e monitorar várias atividades que o motor realiza, como a injeção de combustível e admissão de ar.

O módulo de injeção eletrônica é um sistema embarcado que monitora o sistema através de sensores como: sensor de rotação, sensor de posição da borboleta, sensor de temperatura do ar, entre outros.

---

[6] Disponível em: < https://www.diymore.cc/products/5v-8-channel-relay-module-with-optocoupler-for-arduino/> Acesso em: 03 dez. 2018.

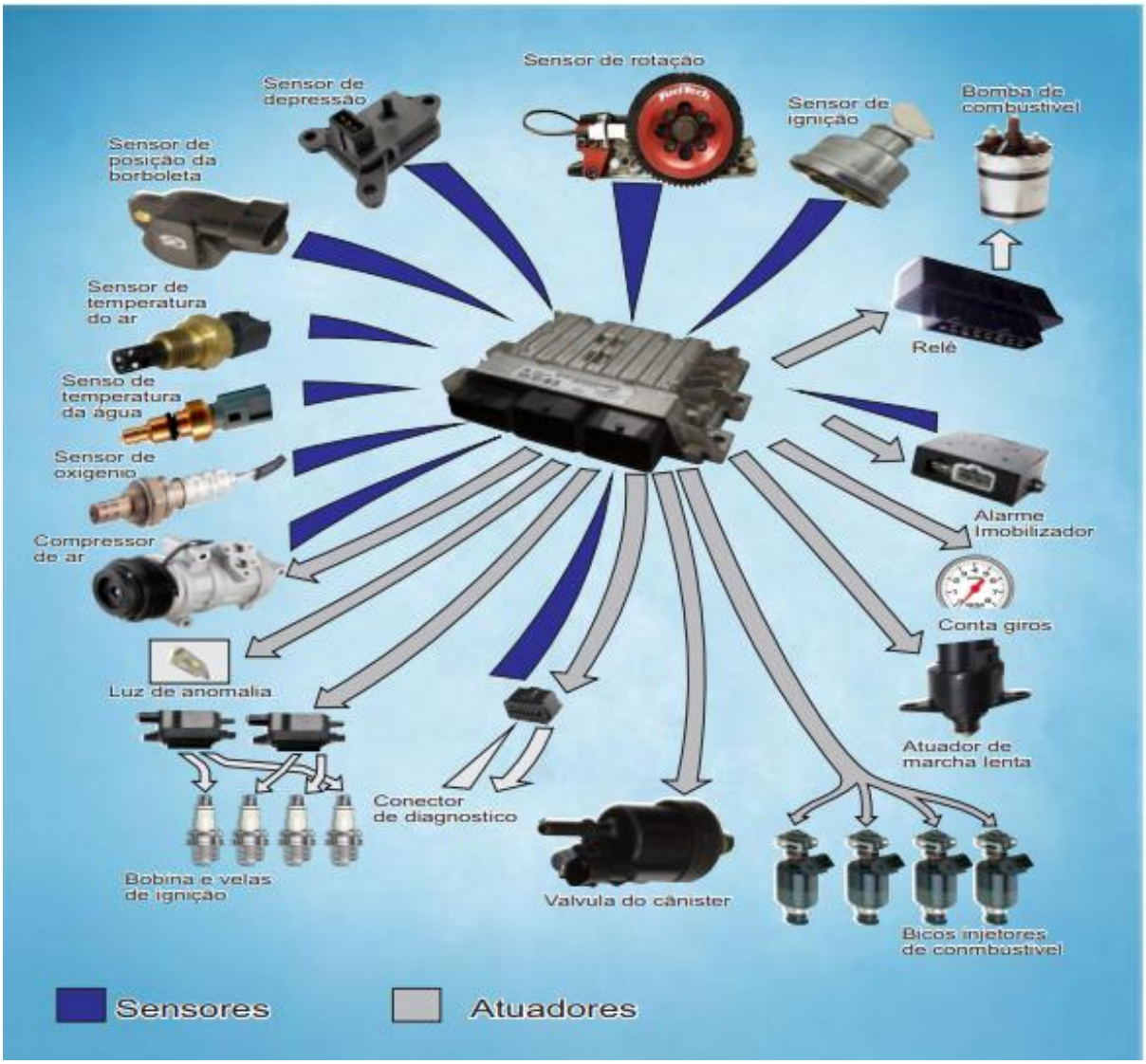
[7] Disponível em: < https://www.masterwalkershop.com.br/modulo-rele-3v-1-canal/> Acesso em: 01 abril. 2021.

Através das informações captadas pelos sensores, o módulo de injeção realiza os cálculos necessários e envia os comandos para os atuadores alimentarem e realizarem a queima do combustível. São exemplos de atuadores: bobinas de ignição, bomba de combustível e bicos injetores.

O sistema proposto neste trabalho visa controlar atuadores como bomba de combustível para evitar a locomoção do veículo em caso de um possível furto ou roubo, podendo ser feito remotamente.

Na figura 6, temos o módulo de injeção eletrônica, sensores e atuadores do sistema de injeção eletrônica. No centro da figura 6 temos o módulo de injeção eletrônica, os elementos ligados na cor azul a ele são os sensores responsáveis pela coleta de informações de funcionamento do motor; os elementos ligados na cor cinza são os atuadores responsáveis pelo controle do motor recebendo do módulo de injeção os comandos elétricos necessários para desempenho da sua função.

Figura 6 – Sistema de injeção eletrônica




Fonte: Adaptado de Dispemec Autopeças[8]

## 2.7 *Node-Red*

Tendo início em 2013 pelo Grupo de Serviços de Tecnologia Emergentes da *IBM*, começou como uma prova de conceito para visualizar e manipular mapeamentos entre tópicos *MQTT*, rapidamente tornou-se uma ferramenta escalável. O *Node-Red* é uma ferramenta visual de programação construída no *Node.js*, de forma aberta projetada para conectar dispositivos de *hardware*, *APIs* e serviços online, como parte da *IoT*.

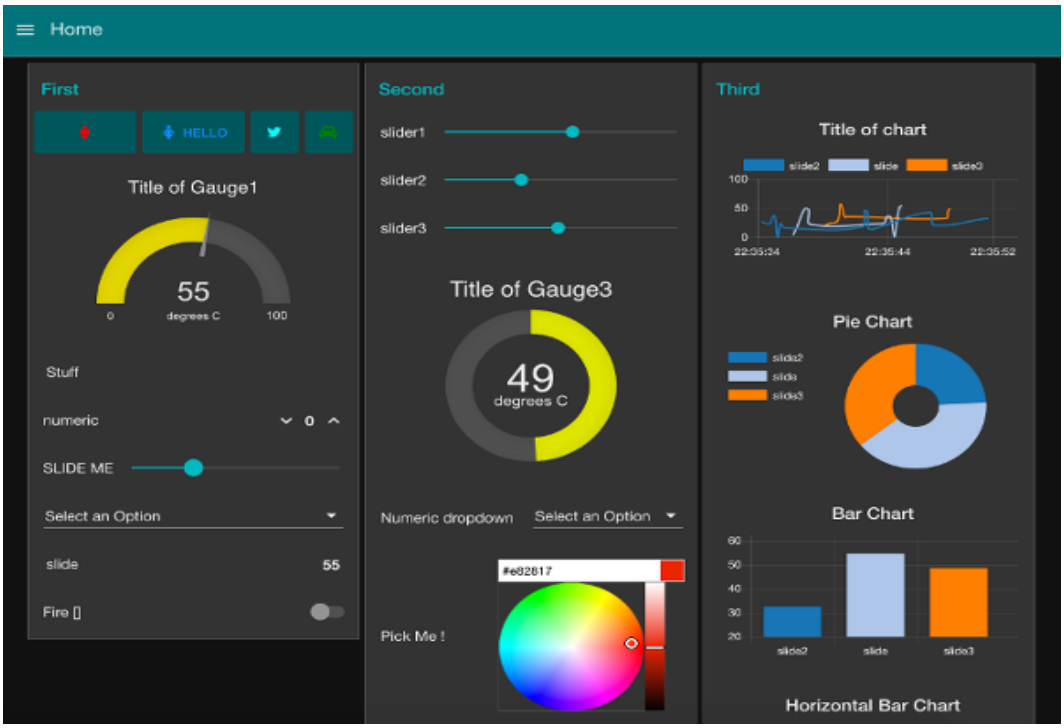
A ferramenta possui um editor baseado em navegador, que facilita a conexão de fluxos usando uma paleta de funcionalidades, contabilizando cerca de 40 chamadas de *nodes*, que podem ser adicionadas através da *Node-Red Library* em tempo de execução, como por

[8] Disponível em: < http://blog.dispemec.com/como-funciona-a-injecao-eletronica/>Acesso em: 19 maio. 2019.

exemplo: ler arquivos CSV, escutar eventos *http*, *tcp*, *websockets*, mídias sociais, *dashboards*, banco de dados. Outra vantagem é a possibilidade de ser implantado em sistemas operacionais *Windows* ou *Linux*.

Na figura 7, um exemplo de *dashboard Node-Red* que possui os controles das operações e monitor do status dos sensores.

Figura 7 – Exemplo de *Dashboard Node-Red*




Fonte: Node-RED[9]

## 3. TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados três trabalhos relacionados, que possuem funcionalidades semelhantes ao projeto proposto. Na Seção 3.1, é apresentada uma solução denominada Monitoramento e Gestão de uma Frota de Veículos, utilizando sistemas embarcados, desenvolvido por Pacheco (2016). Na Seção 3.2, é apresentada a solução Aplicação para Monitoramento Veicular em Tempo Real desenvolvida por Maicon (2017); e por último *Design and Development of Flexible On-Board Diagnostics and Mobile Communicatication for Internet of Vehicles* produzido por V.Kirthika, A.K.Veeraraghavan (2018).

[9] Disponível em: < https://flows.nodered.org/node/node-red-dashboard />Acesso em: 21 maio. 2019.

### 3.1 Monitoramento e gestão de uma frota de veículos

No trabalho de Pacheco (2016), foi desenvolvido um sistema para gerenciamento de frotas de veículos. A solução foi dividida em duas partes: o sistema embarcado e a plataforma *web* para visualização de dados.

O sistema embarcado é comandado por uma placa *Raspberry Pi 3*[10] que obtém a geolocalização através de uma *Shield GPS*[11] e as informações disponibilizadas pelo computador de bordo do veículo como velocidade e distância percorrida são obtidas pela porta *ODB* conectada a um adaptador *ELM327* no veículo. O *Raspberry* troca dados com o *ELM327* através de *bluetooth*. Todas essas informações são enviadas para um servidor web via *wi-fi*.

Seu trabalho coleta os dados do veículo, armazenando em um banco *SQLITE* e através da solução *web* visualiza informações sobre toda a frota, como foto, modelo, placa, cor e leitura dos sensores.

Entre outras características, este trabalho diferencia-se do proposto em Monitoramento e gestão de uma frota de veículos por propor uma solução que imobiliza o veículo quando solicitado pelo usuário da plataforma *web*.

### 3.2 Aplicação para Monitoramento Veicular em Tempo Real

Em Maicon (2017), a solução tem como proposta desenvolver uma aplicação que informe possíveis falhas mecânicas ou furtos de veículos. Como funcionalidades principais têm: visualização da localização do veículo, obtenção de imagens do interior do veículo e dados de sua porta *ODB*.

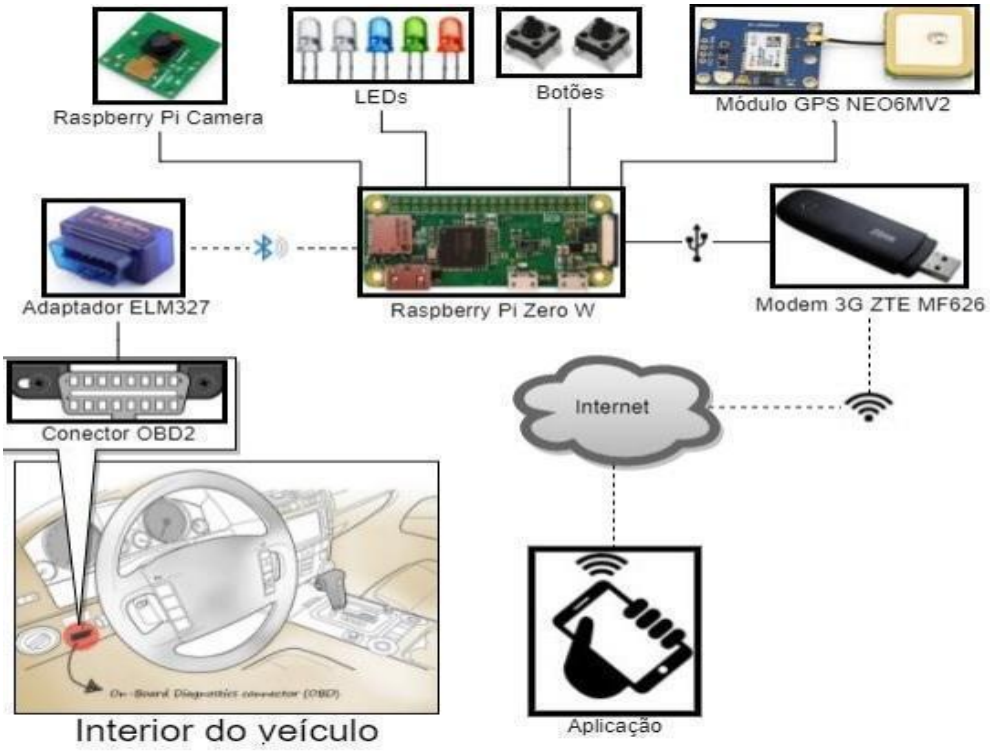
Nesse trabalho, uma *Shield* GPS e uma *Raspberry Pi Câmera* estão conectados ao *Raspberry*, em possíveis casos de furto, é possível obter a localização além de fotos do interior do veículo. Todo esse conjunto de dados obtidos é transmitido através de um *modem* 3G conectado ao *Raspberry*, em que é possível visualizar esses dados através de um aplicativo para dispositivos móveis. Como diferencial, nesta pesquisa dispõe-se a oferecer envio de

[10] https://www.raspberrypi.com/products/raspberry-pi-3-model-b/

[11] https://www.google.com/url?q=https://www.adafruit.com/product/2324&sa=D&source=docs&ust=1642031557834688&usg=AOvVaw0uT-PWUUdjA4dn3LD5c_gF

dados através de uma *Shield GSM/GPRS* ao invés de um modem 3G que aumenta custos do projeto.

Figura 08 – Diagrama esquemático de aplicação para monitoramento veicular em tempo real.




Fonte: Maicon (2017)

### 3.3 *Design and Development of Flexible On-Board Diagnostics and Mobile Communicatication for Internet of Vehicles*

V. Kirthika e A.K.Veeraraghavan (2018) propõem um sistema que é adaptável para ser conectado com à porta *ODB* de qualquer veículo. Nesse modelo, um *Arduino Mega ADK* conecta-se diretamente à porta *ODB* através de uma *Shield Can Bus*. O *Arduino* está ligado à *Raspberry Pi*, que processa os dados coletados pelo *arduino*, como dados de sensores. Uma *Shield wi-fi* está ligada ao *arduino*, que envia os dados para o celular do motorista, que pode visualizar as informações através de aplicativo instalado no celular. Outras funcionalidades que podem ser citadas nesse sistema através do uso do *Raspberry Pi*, são:

- Visualização da traseira do veículo através de uma câmera de ré utilizando uma tela *lcd*.
- Exibição da navegação na tela *lcd* através da *Shield* GPS .

### 3.4 Comparação dos trabalhos

Em relação a leitura de sensores, todos os trabalhos relacionados oferecem esse recurso, menos REDSED. O uso de *Shield* GPS para aferimento de localização geográfica é utilizada nos trabalhos relacionados, mas neste é utilizado o módulo *gps* nativo da placa de desenvolvimento.

O modo de comunicação com a Internet em REDSEC se dá através da rede celular, semelhante ao trabalho de Maicon (2017), diferenciando-se de Pacheco (2016) que utilizava *wi-fi* e de V. Kirthika e A.K.Veeraraghavan (2018) que não se comunicam com a Internet .

Durante o desenvolvimento de REDSEC tentou-se implementar a funcionalidade de ler valores dos sensores do veículo e código de falhas a fim de disponibilizar para o proprietário essas informações para fins de manutenção preventiva, mas não obteve-se êxito.

Apenas em nosso trabalho a funcionalidade de desligar o automóvel remotamente é possível.

Na tabela 02, temos um resumo comparativo entre as principais características dos trabalhos relacionados e este trabalho:

**Tabela 02** – Comparativo entre os trabalhos relacionados

| **Trabalhos/Características** | Monitoramento e gestão de uma frota de veículos. | Aplicação para Monitoramento Veicular em Tempo Real | *Design and Development of Flexible On-Board Diagnostics and Mobile Communicatication for Internet of Vehicles* | REDSEC |
|---|---|---|---|---|
| Hardware | Raspberry Pi 3 | Raspberry Pi Zero W | Raspberry Pi 3/Arduino Mega ADK | ESP32 TTGO TBEAM |
| Leitura de sensores/Leitura de | Sim/Não | Sim/Sim | Sim/Não | Não/Não |

| | | | | |
|---|---|---|---|---|
| erros da injeção | | | | |
| Aferimento de localização geográfica | Shield GPS | Shield GPS | Shield GPS | GPS nativo da ESP32 TTGO |
| Aplicativo móvel | Não possui | Possui | Possui | Não possui |
| Sistema web | Sim | Não | Não | Sim |
| Desligamento remoto do veículo | Não | Não | Não | Sim |
| Meio de comunicação com a internet | Wi-fi | Rede móvel de dados(WCDMA) | Não se comunica | Rede móvel de dados (GPRS) |
| Meio de comunicação com a porta ODB | ELM327 | ELM327 | Shield Can Bus | Não possui |

Fonte: Autor

## 4. Desenvolvimento

Na seção 4.1 é apresentado como foi desenvolvido o protótipo do sistema embarcado.

### **4.1** Montagem do sistema embarcado

#### **4.1.1** Escolha do *Hardware*

Na escolha do *hardware*, foi priorizado a seleção de um *hardware* livre, ou seja, que é possível encontrar disponível o esquema elétrico e drivers para comunicação, que pudesse atender o alto processamento de informações. Foi comparado então os *hardwares* A*rduino Uno, Arduino Mega 2560, ESP8266* e o *ESP32 TTGO TBEAM* os quais tínhamos disponíveis para desenvolvimento.

Na tabela 03, é apresentado um comparativo entre as principais placas analisadas para o desenvolvimento deste trabalho.

Tabela 03 – Comparativo entre placas de desenvolvimento[12]

| **Descrição** | **Arduino Uno** | **Arduino Mega 2560** | **ESP32 TTGO TBEAM** | **ESP8266** |
|---|---|---|---|---|
| Alimentação | 5V | 5V | 2,2V ~ 3,3V DC | 2,2V ~ 3,3V DC |
| Frequência de Operação | 0 ~ 16 MHz | 0 ~ 16 MHz | 80MHz ~ 240MHz | 80MHz ~ 160MHz |
| Memoria Flash | 32KB | 256 KB | 4MB | 4MB |
| Memoria Ram/Sram | 2KB | 8 KB | 520KB | 36kB |
| Memoria Rom/EEPROM | 4KB | 4KB | 448KB | 64KB |
| Processador | AVR® 8-bit RISC | ATmega2560 RISC com até 16 MIPS | Xtensa® Dual-Core 32-bit LX6 | Tensilica® L106 ultra-low power 32-bit |
| Pinos de I/O | 23 pinos com 6 PWM | 54 pinos com 14 PWM | 34 pinos com 16 PWM | 13 pinos com 9 PWM |
| Bluetooth | Somente com Shield | Somente com Shield | Incluso | Somente com Shield |
| GPS | Somente com Shield | Somente com Shield | Incluso | Somente com Shield |

Fonte: Autor

Devido ter melhor processamento e memória foi escolhido o *ESP32 TTGO TBEAM*, ressalta-se ainda que possui *GPS* incluso, eliminando a necessidade de se adquirir *Shield* com essa funcionalidade.

Para comunicação com a internet escolheu-se usar a tecnologia de internet móvel de celular por ter maior cobertura de sinal, a fim de garantir que o sistema embarcado localizado no veículo sempre envie as informações ao servidor. Na escolha da *Shield* de comunicação com a rede móvel celular foi comparado então os hardwares *SIM800L, SIM5320E e SIM7600CE*..

Na tabela 04, é apresentado um comparativo entre as placas de comunicação rede móvel de celular. As informações da tabela foram retiradas do site *aliexpress*[13].

[12] Adaptado de: < https://xprojetos.net/arduino-esp32-e-esp8266-comparacao/>Acesso em: 21 maio. 2019.

Tabela 04 – Comparativo entre placas de comunicação a rede móvel.

| Modelo | Rede | Taxa de download/ upload | Valor | Consumo |
|---|---|---|---|---|
| SIM800L | 2G | 85.6 kbps /85.6 kbps | R$15,17 | Até 2 Amperes |
| SIM5320E | 3G | Até 3.6 Megabit/s /Até 2 Megabit/s | R$202,82 | A partir de 1 Ampere |
| SIM7100CE | 4G | Até 100 Megabit/s /Até 50 Megabit/s | R$405,54 | Até 2 Amperes |

Fonte: Aliexpress

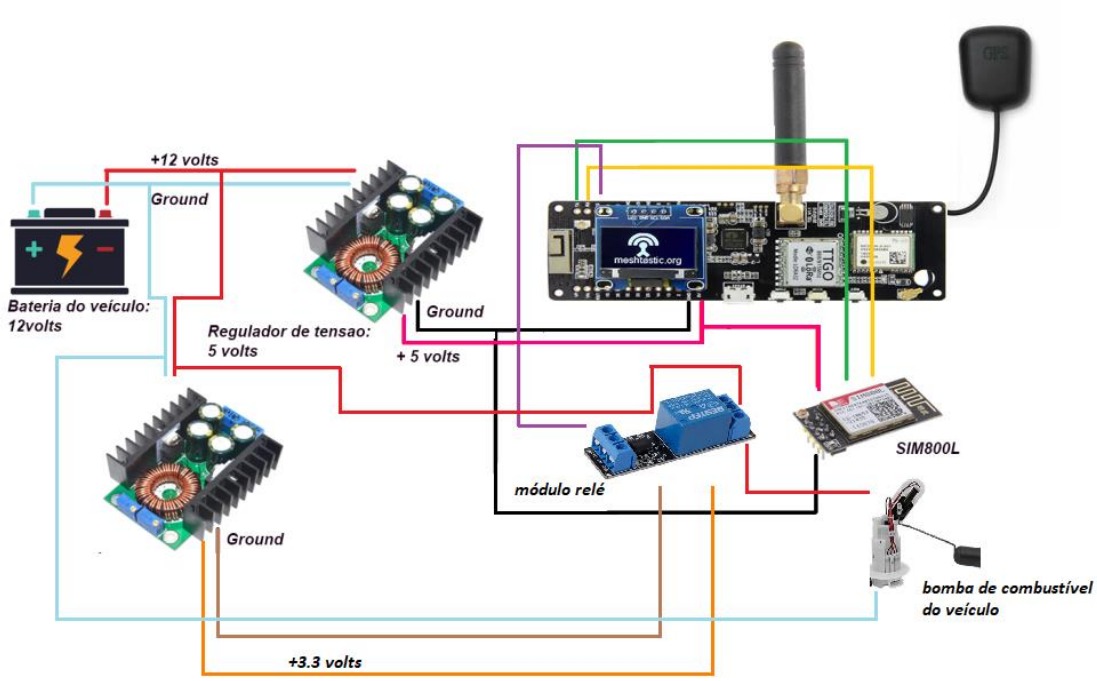
Devido ter menor custo e atender o requisito do *MQTT* que necessita de pouca banda, o módulo *SIM800L* foi escolhido para o protótipo.

### 4.1.2 Montagem do *Hardware*

Na figura 09, está ilustrada a ligação em que a bateria do veículo alimentará os reguladores de tensão com 12 *volts* na entrada. Por sua vez, o primeiro regulador de tensão oferece em sua saída uma tensão de 5 *volts* que alimenta o *ESP32* e o módulo *SIM800L*. O segundo regulador de tensão alimenta o módulo relé com *3 volts*. Os pinos de *RX* e *TX* do *SIM800L* estão conectados aos *TX* e *RX* do *ESP32* respectivamente. O pino de ativação do módulo relé deve ser conectado ao pino 4 do *ESP32*. O módulo relé, por sua vez, fica conectado no chicote da bomba de combustível. Nas figuras 10 e 11 temos como ficou montado o *hardware* que ficará no veículo, onde temos o *ESP32*, módulo relé, o *SIM800L*, os reguladores de tensão e as antenas de comunicação.

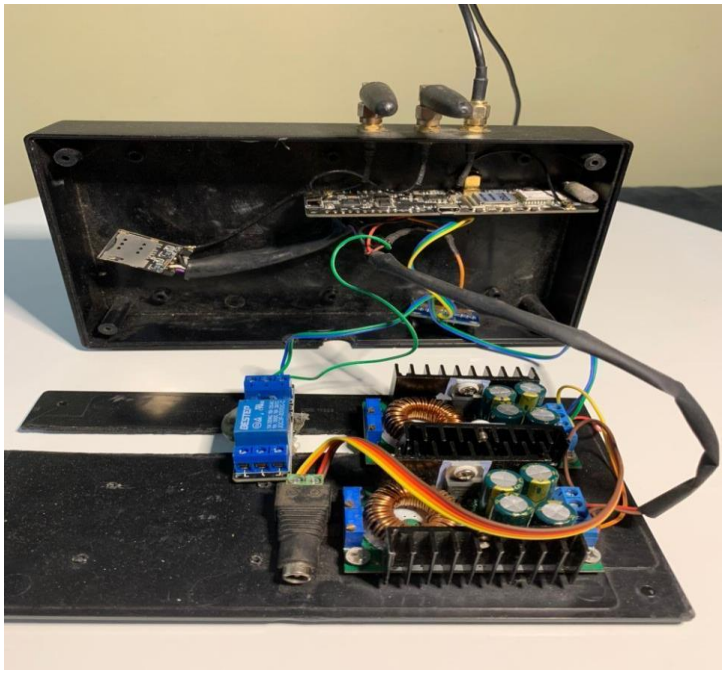
Figura 09– Montagem do *Hardware*

[13] Disponivel em <http://www.aliexpress.com>



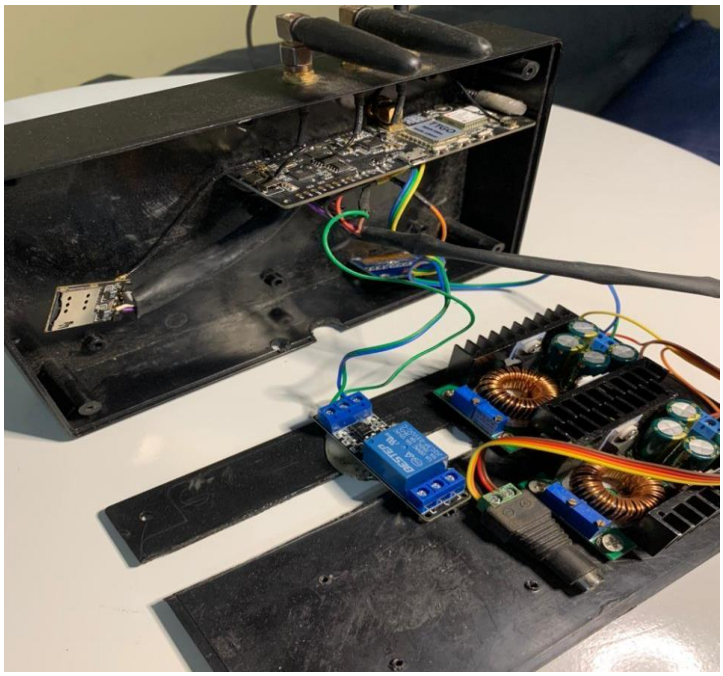

Fonte: Autor

Figura 10– Montagem do *Hardware*



Fonte: Autor

Figura 11– Montagem do *Hardware*



Fonte: Autor

### 4.1.3 Escolha do protocolo de comunicação

Como dito anteriormente na seção Protocolos de comunicação para *IoT*, o protocolo *MQTT* foi escolhido por todas as vantagens apresentadas na seção 2.2.

### 4.1.4 Código Fonte

O código fonte do sistema embarcado que foi desenvolvido em linguagem "c" e dos fluxos do *Node-Red* em "*node.js*" pode ser obtido em github[14]

## 4.2 Criação Máquina Virtual

A fim de reduzir custos com virtualização na nuvem, para configuração do servidor que trata das informações recebidas pelo *hardware*, foi definida a utilização do *software virtual box* para criação de uma máquina virtual com as seguintes configurações:

- Disco rígido de 63.61 GB
- Memória RAM de 2 GB

[14] Disponível: https://github.com/kassiooliver/cassioTCCUFC/

- Memória de vídeo de 256 Megas
- Um adaptador de rede configurado como *nat*
- Sistema Operacional *Windows* 7 64 *bits*

O sistema *Windows* foi escolhido devido a configuração, *backup* e restauração das configurações do *Node-Red* serem simples de serem realizadas, enquanto no *linux* as mesmas facilidades não foram verificadas.

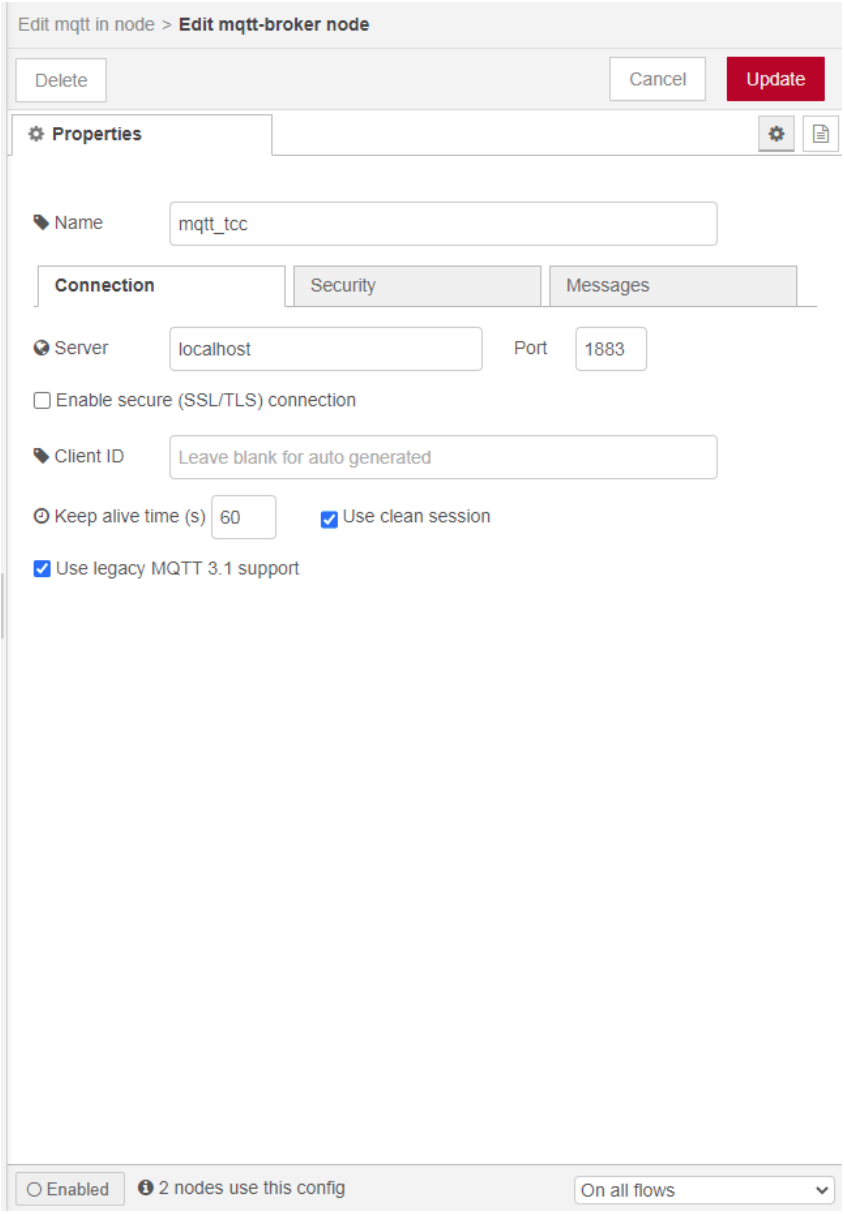
### 4.3 Instalação do *Node-red* e criação de fluxos

O passo a passo de instalação e criação de fluxos utilizando *Node-Red* pode ser obtido em Filipeflop (2019)[15]. O passo a passo de instalação do *MQTT* no *Windows* pode ser obtido em *Eclipse knowledgebase*[16].

Na figura 12, apresentamos a criação do *Broker MQTT* no *Node-Red*, onde configuramos os parâmetros do *broker*, como nome, *ip* e porta do servidor, segurança da conexão, tempo de vida dos pacotes, limpeza da conexão e outras configurações.

[15] Disponível em: < https://www.filipeflop.com/blog/primeiros-passos-node-red-arduino-uno//>Acesso em: 21 maio. 2019.
[16]Disponível em: < https://kb.elipse.com.br/aplicacao-exemplo-driver-mqtt-em-comunicacao-com-broker-mosquitto-mqtt/>

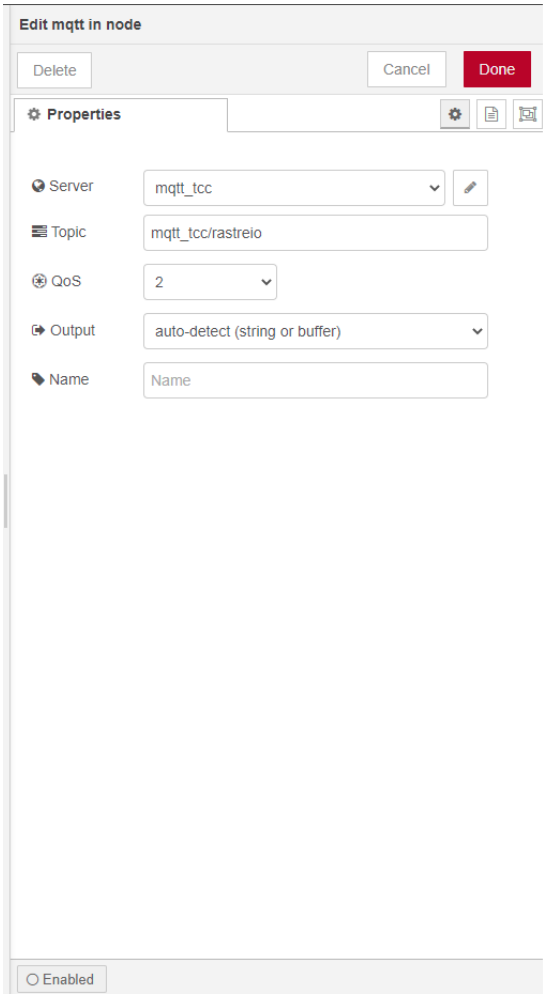
Figura 12– *Broker MQTT* no *Node-Red*




Fonte: Autor

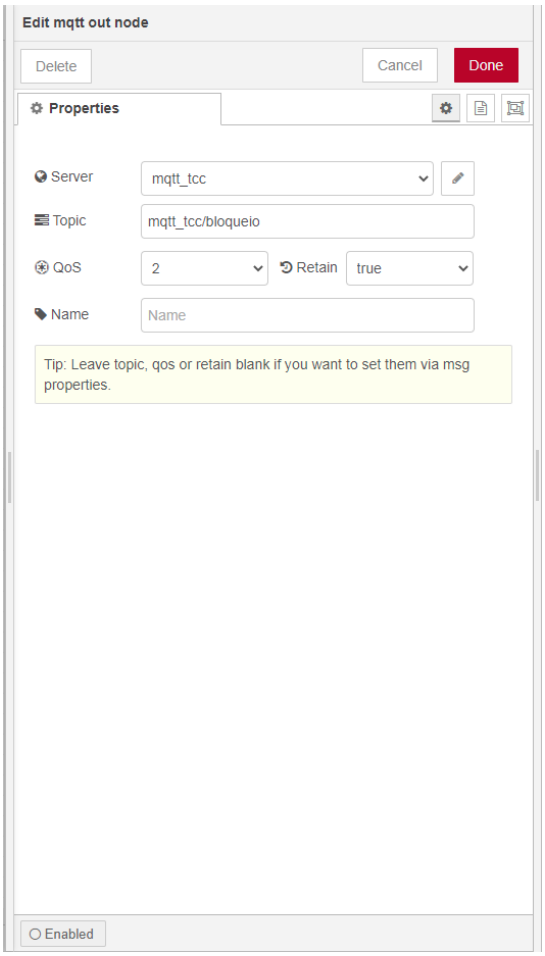
Na figura 13, é exibido a criação do tópico rastreio responsável por receber as coordenadas geográficas no *Broker MQTT*. Na figura 14, é ilustrada a criação do tópico bloqueio, responsável por enviar e receber comandos e informações do bloqueio do veículo. Na figura 15, é apresentado como ficou o fluxo responsável por tratar a localização do veículo. Na figura 16, pode-se visualizar como ficou o fluxo responsável por tratar o bloqueio do veículo.

Figura 13 – Tópico Rastreio



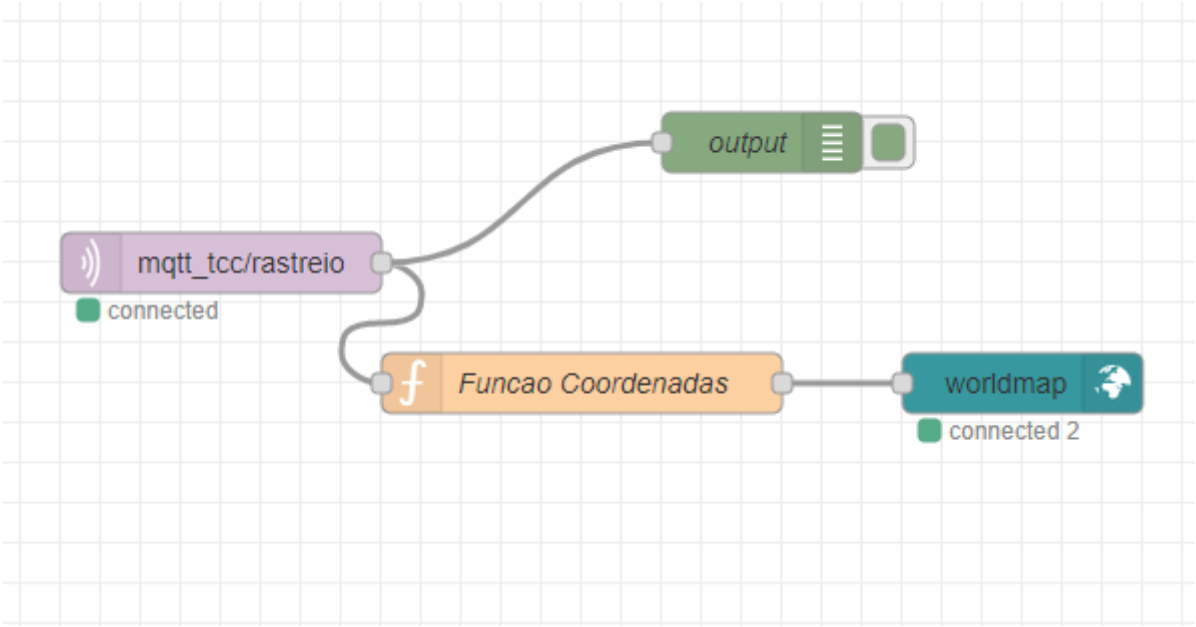

Fonte: Captura de tela feita pelo autor

Figura 14– Tópico Bloqueio




Fonte: Captura de tela feita pelo autor

Figura 15– Fluxo Rastreio

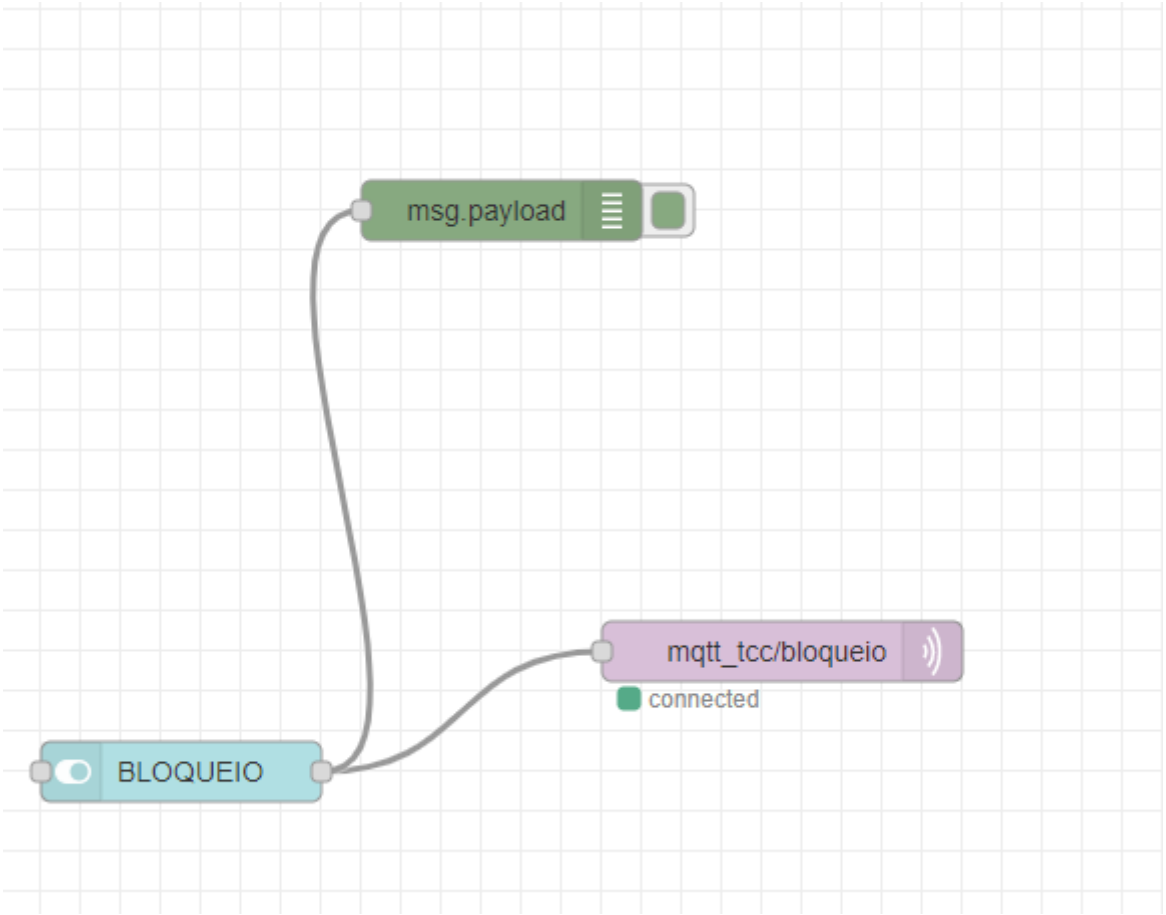



Fonte: Imagem capturada pelo autor

Na figura 15, pode-se ver o primeiro fluxo criado onde foram utilizados os nós ***mqtt in, function, ui_worldmap, debug:***

- ***Mqtt in(mqtt_tcc/rastreio)*** permite a entrada de dados ao broker Mqtt.
- ***Function(Função Coordenadas)*** permite que o código *JavaScript* seja executado nas mensagens que entraram a partir do nó ***Mqtt in.***
- ***Ui_worldmap(worldmap)*** permite fornecer uma pagina web do mapa mundial com a localização recebida pelo nó ***Mqtt in*** e tratada pelo nó ***Function.***
- ***Debug(output)*** permite visualizar as mensagens passadas pelo nó ***Mqtt in*** dentro do editor de fluxos..

Figura 16- Fluxo bloqueio




Fonte: Imagem capturada pelo autor

Na figura 16, vemos o primeiro fluxo criado onde foram utilizados os nós ***ui_switch, mqtt out, debug:***

- ***Ui_switch(BLOQUEIO)*** permite mudar o estado no painel *Node-Red*, estando ativado ele transmite a informação "1" para o dispositivo no carro para ativar o bloqueio, ao ser desativado envia a informação "0" para desativar o bloqueio;
- ***Mqtt out(mqtt_tcc/bloqueio)*** permite a saída de dados ao broker Mqtt;
- ***Debug(msg.payload)*** permite visualizar as mensagens passadas pelo nó ***Mqtt out*** dentro do editor de fluxos.

## 4.4 Configuração *ngrok*

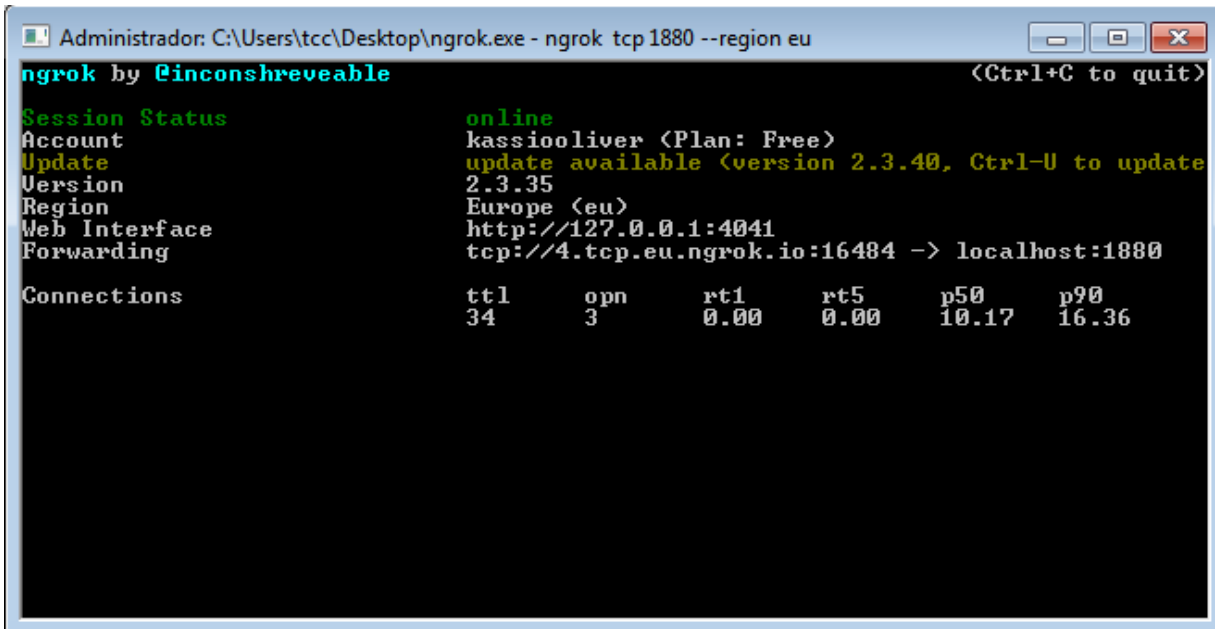
A aplicação *web* precisa de um endereço ip real para ser acessada externamente, mediante isso e custos com ip reais serem elevados, foi utilizada a ferramenta *ngrok* .

O *ngrok* é uma poderosa ferramenta que cria túneis entre as portas da sua máquina local e os servidores dele, o que possibilita outras pessoas acessarem o que quer que esteja naquela porta disponibilizada. O passo a passo de instalação pode ser encontrado em CONECTAAI.[17]

Na figura 17, é ilustrado o tunelamento iniciado, onde é feito o direcionamento da porta 1880 do *Node-Red* do servidor de aplicação para o endereço 4.tcp.eu.ngrok.io:16844 através do comando: "*ngrok tcp 1880 –region eu*".

Na figura 18 visualiza-se o tunelamento iniciado, no qual é feito o direcionamento da porta 1883(Porta *MQTT*) do nosso servidor para o endereço 2.tcp.eu.ngrok.io:16559 através do comando: "ngrok tcp 1883". Ressalta-se que o endereço de redirecionamento do *ngrok* é feito aleatoriamente.

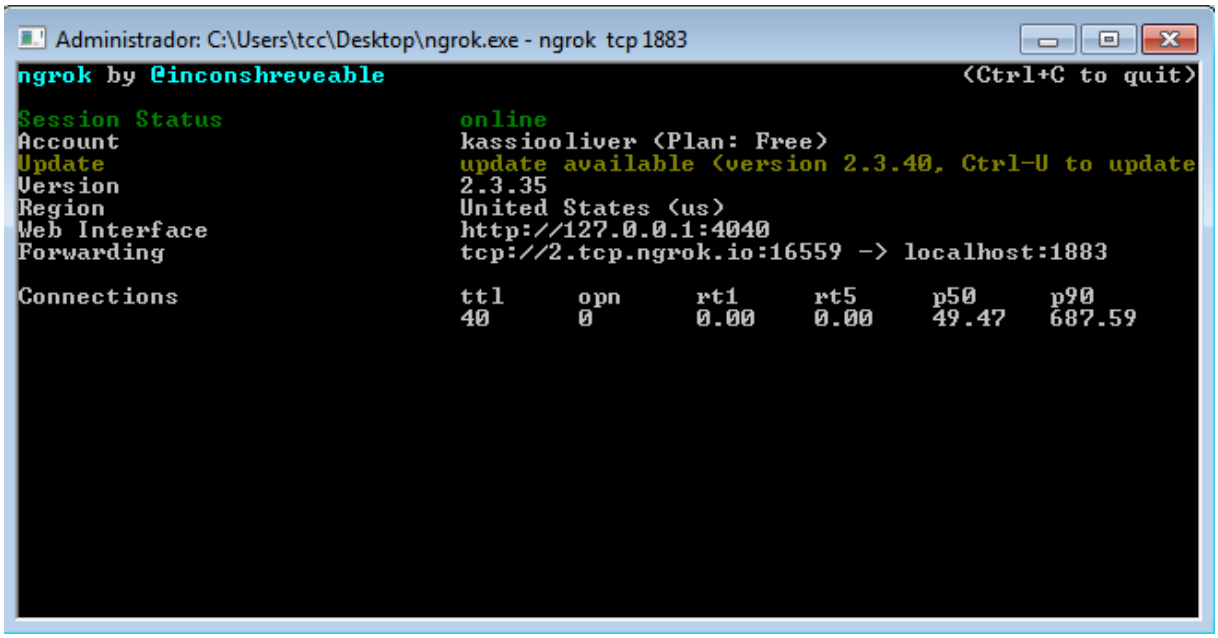
Figura 17– Túnel para porta 1880




Fonte: Captura de tela feita pelo autor

[17] Disponível em: < https://www.conectaai.com/conhecendo-o-ngrok/Acesso em: 25 jun. 2021.

Figura 18- Túnel para porta 1883*(MQTT)*

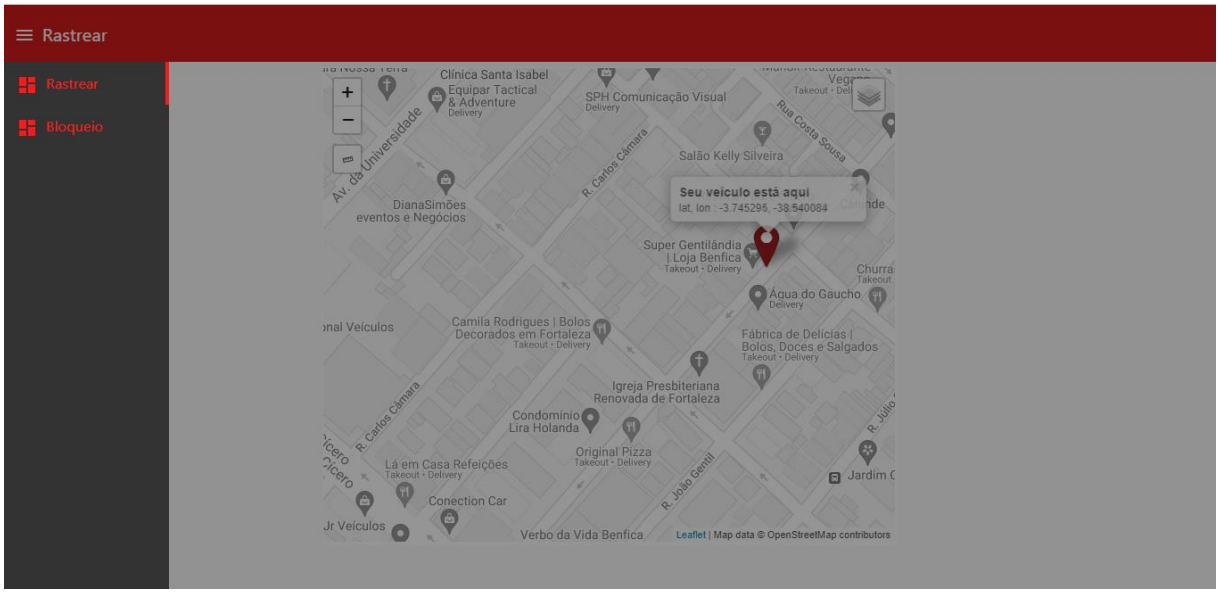



Fonte: Captura de tela feita pelo autor

### 4.5 Execução

Na figura 19, é exibida  a página de rastreamento onde se visualiza a localização do veículo. Na figura 20, a página de bloqueio do veículo, onde é possível ativar o bloqueio e verificar o status do bloqueio do veículo.

Figura 19- Página de rastreamento




Fonte: Captura de tela feita pelo autor

Figura 20- Página de bloqueio


Fonte: Captura de tela feita pelo autor

- O vídeo de execução de bloqueio do veículo pode ser acessado neste link[18]

[18]Disponível em: < https://github.com/kassiooliver/cassioTCCUFC/blob/main/exemplo%20bloqueio%20remoto.mp4/>Acesso em: 22 ago. 2021.

- O vídeo de rastreio do veículo pode ser acessado neste link[19]
- O código fonte do sistema embarcado e o arquivo de backup com as configurações do *Node-Red* pode ser obtido no *GitHub*.[20]

## 5 Avaliação e Resultados

Como os objetivos do trabalho são monitoramento em tempo real e imobilização remota imediata caso o veículo seja roubado, é interessante que as informações cheguem ao servidor o mais rápido possível, assim obtendo localizações exatas e imobilizações quase instantâneas e precisas assim que necessárias para evitar que o veículo não se distancie muito em caso de furto/roubo. Sendo assim métricas como latência(tempo que um pacote chega da origem ao destino) e perda de pacotes(quando um ou mais pacotes não chegam ao destino) sejam analisadas no trabalho a fim de verificar se são satisfatórias.

Os procedimentos para a criação do ambiente e testes foram:

1. Montagem do *hardware*.
2. *Upload* do *firmware* para o sistema embarcado.
3. Instalação do sistema embarcado no veículo.
4. Criação da máquina virtual com sistema operacional Windows 7.
5. Instalação e configuração do *Node-red*, *ngrok*, *wireshark* na máquina virtual.
6. Testes utilizando as operadoras Oi e Claro na cidade de Fortaleza-Ce.

No estudo, foi analisada a latência e descarte de pacotes da conexão, o cenário do experimento pode ser encontrado na figura 21. O sistema proposto foi desenvolvido conforme mostrado na seção 4, o sistema embarcado foi instalado em um veículo *Volkswagen* Gol, e o servidor *broker* instalado em uma máquina virtual *Windows* em um computador da marca *acer* com processador i3 e 8 gigas de memória ram. Foram realizadas cinco rodadas dos experimentos com o carro parado e em movimento utilizando a operadora Claro e cinco utilizando a operadora Oi, as quais estavam disponíveis para testes que foram realizados na cidade de Fortaleza. O *Keep Alive* foi configurado no servidor *MQTT* para um segundo, ou seja, sendo um segundo como intervalo máximo entre o envio de um pacote e o próximo, portanto caso o envio de um pacote passe de um segundo, ele é descartado e o próximo

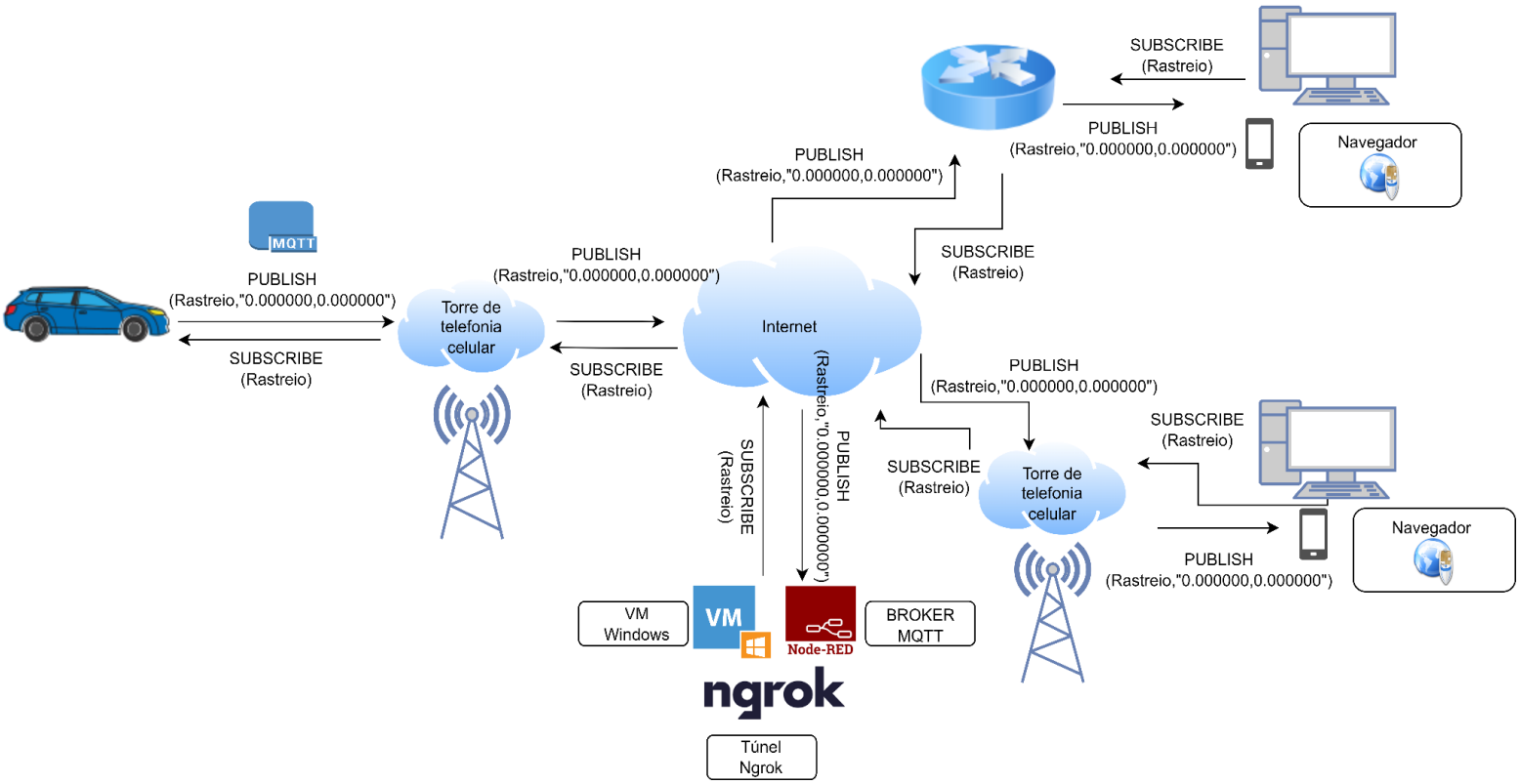
[19]Disponível em: < https://github.com/kassiooliver/cassioTCCUFC/blob/main/exemplo%20rastreamento.mp4/>Acesso em: 22 ago. 2021.
[20] Disponível em: <https://github.com/kassiooliver/cassioTCCUFC/>Acesso em: 22 ago. 2021.

enviado. Para a análise dos dados, foi utilizado o software wireshark que é um analisador de pacotes amplamente utilizado para diagnósticos e análise de desempenho de redes de computadores. No experimento realizado, foram observadas as métricas de latência e perda de pacotes. Dessa forma, foram definidos filtros no wireshark para análise das mensagens MQTT. O filtro “mqtt” em conjunto com o filtro “frame.time_delta > 0.999” foram definidos para agrupar os pacotes com a informação “Ping-Request” com um segundo ou mais caracterizando, dessa forma, a perda de pacotes.

A análise de latência e perda de pacotes se deu filtrando pacotes *MQTT* com mensagens do tipo *PUBLISH* contendo informações dos tópicos rastreio e bloqueio. Rastreio envia informações da localização do veículo e bloqueio atualiza o estado do bloqueio da bomba de combustível.

Na tabela 05 temos os resultados obtidos nos experimentos. A latência foi calculada tirando uma média do tempo de todos pacotes *MQTT* capturados. Durante os testes a conexão não teve perda de pacotes e foi obtido bons valores de latência, uma média de 81,6 ms para a operadora Oi e 76,2 ms para a operadora Claro. A fim de se identificar no tráfego a perda de pacotes foi retirada a antena do módulo *SIM800L* para ele ficar sem sinal e se desconectar da Internet. Foi observado que sempre que um *PING REQUEST* atrasa um segundo ou mais é porque houve perda de pacotes. A análise pode ser observada na figura 22.

Figura 21 – Cenário de testes




Fonte: Autor

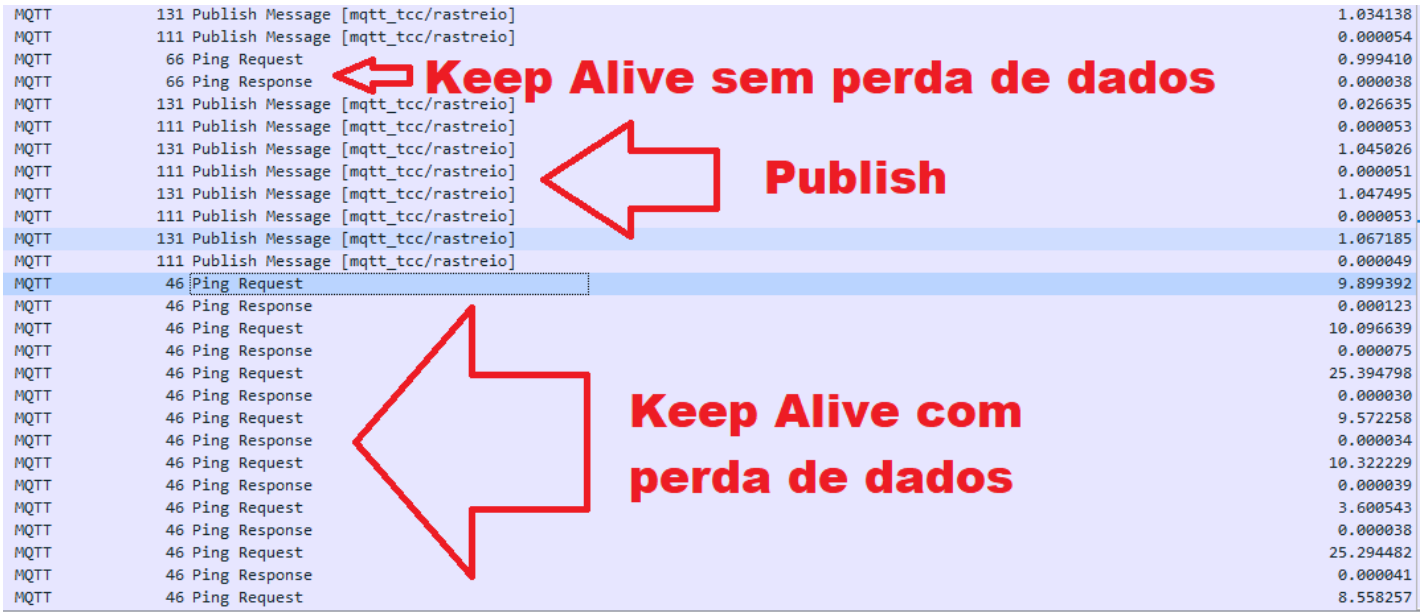
Tabela 05 – Análise Latência e Perda de pacotes

| Teste | Operadora | Quantidade de Pacotes Analisados | Pacotes MQTT | Tempo de Análise | Latência | Perda de pacotes |
|---|---|---|---|---|---|---|
| | | | | | | |
| 1 | Oi | 68738 | 2532 | 20 min | 91ms | 0 |
| 2 | Oi | 56829 | 2490 | 20 min | 85ms | 0 |
| 3 | Oi | 56609 | 2411 | 20 min | 82ms | 0 |
| 4 | Oi | 68230 | 2446 | 20 min | 80ms | 0 |
| 5 | Oi | 59647 | 2475 | 20 min | 76ms | 0 |
| Média de atrasos | | | | | 81,6ms | |

| Teste | Operadora | Quantidade de Pacotes Analisados | Pacotes MQTT | Tempo de Análise | Latência | Perda de pacotes |
|---|---|---|---|---|---|---|
| | | | | | | |
| 6 | Claro | 62590 | 2538 | 20 min | 89ms | 0 |
| 7 | Claro | 65796 | 2565 | 20 min | 87ms | 0 |
| 8 | Claro | 91988 | 2516 | 20 min | 67ms | 0 |
| 9 | Claro | 71850 | 2416 | 20 min | 70ms | 0 |
| 10 | Claro | 78186 | 2540 | 20 min | 68ms | 0 |
| Média de atrasos | | | | | 76,2ms | |

Fonte: Autor

Figura 22 – Observação da Perda de Pacotes




Fonte: Autor

## 6. CONCLUSÃO

Foi comprovado que utilizando a rede de operadora celular 2G(que utiliza baixa largura de banda) e o protocolo *MQTT* é possível desenvolver soluções tanto para rastreamento de veículos como para criação de novas soluções remotas. Podemos concluir que o trabalho proposto realizou seus objetivos de rastrear o veículo com precisões exatas e desligá-lo remotamente em tempo mínimo(3 segundos), tanto *software* como *hardware* atingiram seus objetivos.

Mediante análise dos dados capturados pelo *wireshark*, que atendeu sua necessidade, tivemos um ótimo desempenho de latência(máxima de 91ms) durante os testes considerando que o tempo de latência em redes 2G é alto, na ordem de 10 segundos[21], e perda de pacotes, demonstrando que a rede celular 2G possui baixa latência e perda de pacotes necessários para ser o meio de comunicação para o sistema embarcado e a *web*, dentre as duas operadoras analisadas, a claro se saiu melhor pois obteve menor latência..

Durante o desenvolvimento tentamos implementar a funcionalidade de ler falhas do sistema de injeção através do *bluetooth* do *ESP32* obtendo conflito com as outras portas seriais do *ESP* descartando assim a implementação dessa funcionalidade que não era o objetivo principal do trabalho. O proprietário do veículo tem a localização do veículo a possibilidade de imobilizá-lo com o sistema proposto neste trabalho, assim tendo uma maior proteção contra furtos e roubos.

## 7. TRABALHOS FUTUROS

Para trabalhos futuros pretendemos integrar outras funcionalidades, como ler as informações de falha do veículo através do sistema embarcado e a porta *ODBII* do veículo através de *bluetooth*, que não utiliza cabos, que informará ao condutor possíveis problemas a serem corrigidos ou necessidade de revisão preventiva. Um *dashboard* de informações do veículo apresentaria informações como" temperatura", "rotações do motor", "pressão dos pneus" que em discordância com os valores estabelecidos pelo fabricante para funcionamento normal represente alguma anomalia..

---

[21] Disponível em: <https://www.phoenixcontact.com/assets/downloads_ed/local_br/web_dwl_software/br_faq_GPRS_QuadriBand.pdf>/Acesso em: 14 fev. 2022.

Outras funcionalidades a serem adicionadas é a integração do banco de dados para obter um histórico do trajeto do veículo e hospedar o *Node-Red* na nuvem para assim ter um ip fixo. Outra possibilidade seria de testar uma frota de veículos, cada um com um rastreador REDSEC para mesmo usuario(os) do sistema *web*; implementação de uma camada de segurança no tráfego dos dados; criação de uma aplicativo móvel para o usuário do sistema; utilização de uma única fonte de alimentação para o hardware que fica no veículo que no momento é utilizada duas, uma para o *ESP* e outra para o módulo relé.

# REFERÊNCIAS

AQUINO, Gabriel Calda de. **MQTT**. Youtube, 2018. Disponível em: https://www.youtube.com/watch?v=YZW5wNXDs2c&t=357s. Acesso em: 01 de jun. 2021.

ARDUINO Help Question. **Arduino Help Center,** 2018. Disponível em: https://www.arduino.cc/en/Main/FAQ#toc2. Acesso em: 29 de out. 2018.

ASHTON, Kevin. **Finep**. 2018. Disponível em: http://finep.gov.br/noticias/todas-noticias/4446-kevin-ashton-entrevista-exclusiva-com-o-criador-do-termo-internet-das-coisas. Acesso em: 29 de out. 2018.

BARROS, Marcelo. MQTT: Protocolos para IoT. **Embarcados,** 2015. Disponível em: https://www.embarcados.com.br/mqtt-protocolos-para-iot. Acesso em 30 de out. 2018.

BRITO, S. H. B. **A Internet das Coisas** (IoT). Lab/Cisco, 2014. Disponível em: http://labcisco.blogspot.com.br/2014/07/a-internet-das-coisas.html. Acesso em 30 de out. 2018.

CHAPPELL, Laura . **Detect TCP Delays with Wireshark**. YouTube, 2017. Disponível em: https://youtu.be/e_iS4DfLCEM. Acesso em 22 de ago. 2021

CHIPTRONIC. Modelos de comunicação para iot. **Embarcados**, 2016**.** Disponível em: https://www.embarcados.com.br/modelos-de-comunicacao-para-iot/. Acesso em 02 de ago. 2019.

CHIPTRONIC. Entenda como funciona a injeção eletrônica dos carros. **Embarcados**, 2016. Disponível em: https://chiptronic.com.br/blog/entenda-como-funciona-injecao-eletronica-dos-carros-2. Acesso em 02 de ago. 2019.

FALTA de manutenção em veículos triplica risco de acidentes de trânsito. **Revista Amais, 2018.** Disponível em: http://revistaamais.com.br/falta-manutencao-veiculos-triplica-risco-acidentes-transito. Acesso em 07 de ago. 2019.

FIRM The Technology: Packet Loss vs Latency. **Youtube**, 2018. Disponível em: https://youtu.be/X8oZqRHuyPA/. Acesso em 22 de ago. 2021.

INJEÇÃO Eletrônica. **Winkipedia**, 2018. Disponível em: https://pt.wikipedia.org/wiki/Inje%C3%A7%C3%A3o_eletr%C3%B3nica. Acesso em: 29 de out. 2018.

LAJOTO, Mariana. Jornal Trânsito no Brasil mata 47 mil por ano e deixa 400 mil com alguma sequela. **Folha de S. Paulo**, São Paulo, 2018. Disponível em: https://www1.folha.uol.com.br/seminariosfolha/2017/05/1888812-transito-no-brasil-mata-47-mil-por-ano-e-deixa-400-mil-com-alguma-sequela.shtml. Acesso em 04 de ago. 2019.

LIMA, Davi Magalhães. **Um sistema de automação residencial modular sob Internet das Coisas e Computação Ubíqua.** 2018. 67 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Software)- Universidade Federal do Ceará, Campus de Quixadá, Quixadá, 2018. Disponível em: http://www.repositorio.ufc.br/bitstream/riufc/34506/1/2018_tcc_dmlima.pdf. Acesso em: 7 de ago. 2018.

MANUTENÇÃO preventiva gera economia e diminui acidentes. **Sindivapa,** 2019. Disponível em: http://www.sindivapa.com.br/seguranca-no-transito/manutencao-preventiva-gera-economia-e-diminui-acidentes/1092.html. Acesso em 03 de ago. 2019

MARQUES, V. Figueiredo; KNIESS, J. Uma Avaliação de Desempenho de Protocolos de Camada de Aplicação para Internet das Coisas. In: **Anais do Computer on the Beach**. [S.l: s.n], p.120-129, 2018. DOI: https://doi.org/10.14210/cotb.v0n0.p120-129. Acesso em: 08 de maio de 2021.

MARTINS, Ismael Rodrigues; ZEM, José Luís. Estudo dos Protocolos de Comunicação MQTT e Coap para Aplicações Machine-To-Machine e Internet das Coisas. **R.Tec.Fatec** – Americana, [Sl] v.3, n.1 p. 64-87, mar./set. 2015. Disponível: https://fatecbr.websiteseguro.com/revista/index.php/RTecFatecAM/article/view/41/50. Acesso em: 03 de jun. 2019

MAZZER, D.; FRIGIERI, E. P.; PARREIRA, L. F. C. G. **Protocolos m2m para ambientes limitados no contexto do iot:** Uma comparação de abordagens. [S.l: s.n], 2018.

MOTA, R. P. B; BATISTA, D. Um mecanismo para garantia de qos na "internet das coisas" com rfid. In: **Proceedings of the SBRC 2013 (Brazilian Symposium on Computer Networks and Distributed Systems)**. Brasília, p. 297–310, 2013.

MOTA, Levi da Costa. **Uma análise comparativa dos protocolos SNMP, Zabbix e MQTT, no contexto de aplicações de internet das coisas.** 2017, 77f. Dissertação de Mestrado da Pós-graduação em Ciências da Computação- Universidade Federal de Sergipe, Centro de Ciências Exatas e Tecnológicas, Sergipe. 2017. Disponível em:

https://repositorio.ifs.edu.br/biblioteca/bitstream/123456789/467/1/Disserta%c3%a7%c3%a3o%20Levi%20da%20Costa%20Mota.pdf. Acesso em 30 dez. 2018.

OLIVEIRA, André Schneider de; Andrade, Fernando Souza de. **Sistemas Embarcados Hardware e Firmware na Prática.** São Paulo, 2010.

O QUE SÃO SISTEMAS embarcados. **Oficina da Net**, 2020. Disponível em: https://www.oficinadanet.com.br/post/13538-o-que-sao-sistemas-embarcados. Acesso em 22 de ago. 2021.

PACHECO, Lucas V. **Monitoramento e gestão de uma frota de veículos utilizando sistemas embarcados.** 2013, 76f. Trabalho de conclusão de curso de Engenharia-Universidade de São Paulo, São Carlos, 2013. Disponível em: http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-13012017-154535/?&lang=br. Acesso em 30 de dez. de 2018.

PANZOLINI, Breno. Medindo latência. **Tabless**, 2019. Disponível em: https://tableless.com.br/medindo-latencia/. Acesso em 26 de jan. 2022

SANTAELLA, Lucia; GALA, Adelino; POLICARPO, Clayton; GAZONI, Ricardo. Desvelando a Internet das Coisas. **Revista GEMInIS**. Universidade de São Carlos, v. 1, Fascículo dois, p. 19-32. Jul/Dez, 2013.

SIGNIFICADO de Internet das Coisas. **Significados**, 2018. Disponível em: https://www.significados.com.br/internet-das-coisas. Acesso em: 29 de out. 2018.

SILVA, Maicon M. G. da. **Aplicação para monitoramento veicular em tempo real**. 2017. 109 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Regional de Blumenau. Campus de Blumenau, Blumenau, 2017. Disponível em: http://dsc.inf.furb.br/arquivos/tccs/monografias/2017_2_maicon-machado-gerardi-da-silva_monografia.pdf. Acesso em: 15 de fev. 2021.

SILVA, A. B. M. d.; CARVALHO, F. Fadul. d.; NAZARIO, M. D.; **Constrained Application Protocol (Coap).** Rio de Janeiro, 2019. Disponível: https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/coap/conclusao.html# . Acesso em: 29 de out. de 2018.

SILVA, A. B. M. d.; CARVALHO, F. Fadul. d.; NAZARIO, M. D.; **O protocolo MQTT. Leve e simples:** perfeito para o Iot e sistemas embarcados. Rio de janeiro, 2018. Disponível

em: https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/mqtt/# Acesso em: 29 de out.2018.

STAROSKI, Ricardo A. **ODB-JRP**: Monitoramento veicular com java e raspberry pi M**.** 2016, 87 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Regional de Blumenau. Campus de Blumenau, Blumenau, 2016. Disponível em: http://dsc.inf.furb.br/arquivos/tccs/monografias/2016_2_ricardo-artur-staroski_monografia.pdf.

TABELA de códigos odb2. **APTTA Brasil,** 2018. Disponível em: https://www.apttabrasil.com/wp-content/themes/aptta/pdf/tabela-de-codigos-OBD2.pdf. Acesso em 20 de setembro de 2019.

VALENTE, B. A. L. **Um middleware para a Internet das coisas**. Dissertação de Mestrado da Faculdade de Ciências– Universidade de Lisboa, 2011, 100 f. Disponível em: https://repositorio.ul.pt/handle/10451/921.

V. KIRTHIKA ; A.K. Vecraraghavatr . **Design and Development of Flexible On-Board Diagnostics and Mobile Communication for Internet of Vehicles**. [S.l: s.n], 2018. Disponível em: https://ieeexplore-ieee-org.ez11.periodicos.capes.gov.br/document/8452826. Acesso em: 01 de dez 2018.

YUAN, Michael. Conhecendo o MQTT: Por que o MQTT é um dos melhores protocolos de rede para a Internet das Coisas? **IBM Developer,** 2017. Disponível em: https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html. Acesso em 30 de out. 2018.