**FACULDADE DE TECNOLOGIA SENAI “MARIANO FERRAZ”**

**CURSO DE PÓS-GRADUAÇÃO EM INTERNET DAS COISAS**

**RAFAEL GOMES DE PAULA**

**TÍTULO DO TRABALHO: e subtítulo se houver**

**SÃO PAULO**

**2021**

**RAFAEL GOMES DE PAULA**

**TÍTULO DO TRABALHO: e subtítulo se houver**

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia Senai “Mariano Ferraz” como requisito parcial para obtenção do título de Especialista em Internet das Coisas

Orientador: Prof. Esp. CAIO VINICIUS RIBEIRO DA SILVA

**SÃO PAULO**

|  |
| --- |
| ESTA PAGINA DEVERÁ CONTER A FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DA INSTITUIÇÃO.  ESTE ITEM DEVERÁ SER IMPRESSO NO VERSO DA FOLHA DE ROSTO |

**RAFAEL GOMES DE PAULA**

**TÍTULO DO TRABALHO: e subtítulo se houver**

Trabalho de Conclusão de Curso apresentado à FACULDADE DE TECNOLOGIA SENAI “MARIANO FERRAZ” como requisito parcial para obtenção do título de Especialista em Internet das Coisas

Banca examinadora

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Mestre André

Faculdade de Tecnologia SENAI “Mariano Ferraz”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Especialista Alexandro

Faculdade de Tecnologia SENAI “Mariano Ferraz”

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Especialista Caio

Faculdade de Tecnologia SENAI “Mariano Ferraz”

São Paulo, \_\_\_\_\_ de \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20

**DEDICATÓRIA**

Ao meu pai, exemplo que sempre levarei comigo..

**AGRADECIMENTOS**

Aos professores e colegas de curso, que contribuíram para a realização deste trabalho com dedicação e conhecimento.

Agradecimentos especiais à minha esposa e filha, pela paciência e carinho.

A toda equipe da Faculdade de Tecnologia Senai Mariano Ferraz.

*“Pense globalmente, aja localmente”.*

John Lennon

**RESUMO**

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento; deve ser composto de uma sequência de frases concisas e objetivas e não de enumeração de tópicos em um parágrafo único. A primeira frase deve ser significativa, explicando o tema principal do documento, a seguir, deve-se indicar a informação sobre a categoria do trabalho (estudo de caso, análise de situação etc.). Não se deve colocar nos resumos: símbolos, fórmulas, equações, citações de outros autores. Descrever utilizando a terceira pessoa do plural ou singular, em sua extensão o resumo deve ter aproximadamente 150 palavras.

Palavras-chave: as palavras-chave devem ser em número de 3 a 5 palavras, devem ser sugeridas e, em seguida, deve ser verificada, junto à biblioteca, a possibilidade de sua utilização.

**Exemplo:**

Palavras chave: Resumo. Normatização. Pesquisa.

**ABSTRACT**

Tradução do Resumo para o Inglês

Keywords: tradução das palavras-chave para o inglês.

**LISTA DE ILUSTRAÇÕES**

Figura 1 – Raspberry Pi 4 – Model B.................................................................... XX

Figura 2 – LilyGo TTGO T-Call............................................................................. XX

Figura 3 – Módulo SN65HVD230 CAN Bus........................................................... XX

Figura 4 – Módulo OLED SSD1306...................................................................... XX

Figura 5 – Módulo RTC DS3231........................................................................... XX

**LISTA DE TABELAS**

Tabela 1 – Cronograma 22

Tabela 2 – Recursos 22

**LISTA DE ABREVIATURAS E SIGLAS**

|  |  |
| --- | --- |
| IoT | *Internet of Things -* Internet das Coisas |
| IP | *Internet Protocol -* Protocolo de Internet |
| TCP | *Transmission Control Protocol -* Protocolo de Controle de Transmissão |
| TCP/IP | *Transmission Control Protocol/Internet Protocol* - Protocolo de Controle de Transmissão/Protocolo de Internet |
| IPv4 | *Internet Protocol version 4 -* Protocolo de Internet versão 4 |
| IPv6 | *Internet Protocol version 6 -* Protocolo de Internet versão 6 |
| CI | *Circuito Integrado* |
| SO | Sistema Operacional |
| ARM | *Advanced RISC Machine* |
| API | *Application Programming Interface* |
| JSON | *JavaScript Object Notation* |
| MQTT | *Message Queue Telemetry Transport* |
| OASIS | *Organization for the Advancement of Structured Information Standards* |
| CAN BUS | Barramento *Controller Area Network* |

**LISTA DE SÍMBOLOS**

**1 INTRODUÇÃO**

Desde 1886, quando a primeira patente de veículo automotor foi registrada, a indústria automobilística vem revolucionando a forma como a sociedade se locomove e executam as tarefas diárias, porém, por muito tempo a implementação de tecnologia embarcada nos carros não foi uma prioridade, pois os esforços eram direcionados principalmente aos motores e ao design. Em 1996, 100 anos após a primeira patente, a indústria automobilística deu o primeiro passo para a padronização da tecnologia embarcada nos veículos, a implementação do OBD - On-Board Diagnostic, sistema de autodiagnóstico do veículo. No Brasil, a implementação do OBD, se tornou obrigatório apenas em 2010. Desde então, diversas fabricantes estão implementando novas tecnologias em seus veículos que variam de sensores até sistemas inteligentes de condução semiautônoma do veículo. Apesar do desenvolvimento tecnológico na indústria automobilística ter avançado bastante desde 1886, a tecnologia utilizada nos veículos, ainda é bastante restrita ao fabricante e pouco divulgada no contexto acadêmico.

Se por um lado, a tecnologia embarcada demorou para ser prioridade na indústria automobilística, na indústria eletrônica, os microcontroladores tiveram um desenvolvimento muito rápido. O primeiro microcontrolador foi desenvolvido pela Intel em 1971 e tinha apenas 4 bits para processamento. Após 45 anos do primeiro microcontrolador, em 2016, já era possível encontrar microcontroladores com 32 bits de processamento, ou seja, 8 vezes a capacidade de processamento do primeiro microcontrolador.

Com a evolução dos microcontroladores e da conectividade dos dispositivos, surgiu uma nova categoria de dispositivos inteligentes, a Internet das Coisas (*Internet of Things* - IoT), baseada em dispositivos pequenos, conectados e eficientes.

O paradigma de comunicação Máquina a Máquina (*Machine to Machine* – M2M), é um conceito simples e versátil que sugere integração entre dispositivos eletrônicos, como o próprio nome sugere. A base do conceito é a comunicação entre máquinas através de um meio de comunicação.

A junção do IoT com o M2M, gera diversas possibilidades de aplicações que podem ser criadas. A Internet das Coisas se define com dispositivos e sistemas embarcados que possuem tecnologia de conectividade através da Internet e são interconectados para a execução de determinada tarefa com seu potencial máximo, criando espaço para a inovação de serviços e automações.

Dispositivos IoT já são populares na sociedade, são facilmente encontrados em forma de assistentes virtuais, lâmpadas, sensores de clima, vestíveis, entre outros dispositivos que fazem parte do dia a dia da sociedade moderna. A diversidade de dispositivos IoT associado à internet, gera uma vasta gama de possibilidades de automações, dando oportunidade para integrações entre dispositivos que antes disso, era pouco viável, como é o caso das automações nos veículos automotores. Com a chegada da IoT, se torna viável a integração entre as tecnologias do carro com um dispositivo inteligente, que a partir desta automação, é possível realizar a leitura dos sensores, se conectando a rede CAN Bus. Nesse contexto, é possível criar análises dos dados para a obtenção de aspectos que variam da condução feita por uma motorista em uma viagem, até a customização dos módulos mecânicos do veículo.

Dado o contexto, a pesquisa realizada e descrita neste artigo, propor realizar a análise da condução veicular, que será baseada em dados coletados do veículo durante viagem , usando como parâmetro, os dados estabelecidos no manual do carro para uma boa condução, que visa estabelecer um perfil comportamental para o condutor do veículo.

* 1. **Problematização e hipóteses**

Em 2010, as fabricantes atuantes no Brasil, adotaram a padronização OBD II nos seus veículos. O OBD II possui diversos protocolos de comunicação que dão acesso as informações do carro em tempo real. Apesar de essa tecnologia estar estabelecida há mais de 10 anos no nosso país, o acesso a esta tecnologia e o conhecimento sobre a como extrair as informações ainda é muito escasso. A problematização deste trabalho é explorar as possibilidades de comunicação entre dispositivos IoT e os veículos que possuem essa tecnologia embarcada.

As hipóteses sobre essa comunicação seguem abaixo:

* A leitura dos dados do veículo é possível?
* A leitura dos dados do veículo não será possível?
* Os dados do veículo estarão criptografados pela montadora?
* O dispositivo poderá interagir com o sistema do veículo, alterando dados?

**1.3 Objetivos da pesquisa**

1.3.1 Objetivo geral

Realizar um estudo sobre a capacidade de determinar a condução veicular através de um dispositivo IoT, usando algoritmos que faça predições sobre a conduta do motorista baseada nos dados coletados do veículo.

1.3.2 Objetivos específicos

* Construir um dispositivo protótipo IoT, usando tecnologias e arquiteturas de redes em sistemas que estão presentes no contexto de Internet das Coisas;
* Coletar os dados de sensores do veículo, processar e enviar para análise de comportamento;
* Demostrar os benefícios de forma clara na utilização de Internet das Coisas;
  1. **Justificativa e delimitação da pesquisa**

1.4.1 Justificativa

A tecnologia está constantemente evoluindo para dispositivos menores, mais rápidos e cada vez mais inteligentes. Através dessa evolução, surgiu a Internet das Coisas, como uma nova habilidade da internet em se conectar e coletar dados de dispositivos de onde não eram possíveis as comunicações com a rede de computadores.

Com dispositivos pequenos, com alta capacidade de processamento, baixo consumo de energia e conexão sem fio, a Internet das Coisas abriu uma porta para diversas inovações tecnológicas.

Cada vez mais a sociedade tem suas rotinas automatizada por dispositivos inteligentes como os celulares, wearables e até cafeteiras inteligentes, porém, veículos automotivos, apesar de possuírem tecnologia embarcada, ainda não estão inseridos no contexto da Internet das Coisas, isso foi a principal causa que motivou esse estudo a explorar novas possibilidades de automação e análises feitas a partir da coleta de dados do automóvel.

1.4.2 Delimitação e Universo da Pesquisa

O estudo tem seus limites definidos dentro de cada contexto, como é mostrado a seguir:

* Dispositivo IoT, se limitando a coleta de velocidade atual e rotação do motor, com a finalidade de enviar essas informações somente durante o percurso em que o carro está em movimento.
* Nuvem privada, se limitando a comunicação com o dispositivo IoT, com foco no recebimento e processamento dos dados coletados para análise de condução do veículo.
* Pesquisa e Estudo, se limitando a um único veículo específico utilizado.
* Comunicação IoT, se limitando a rede GSM para troca de informações
  1. **Metodologia**
* Análises bibliográficas
* Compreensão sobre o tema
* Protótipo de dispositivo e rede IoT
  1. **Viabilidade da pesquisa**
     1. Cronograma

|  |
| --- |
| Tabela 1 – Cronograma de Atividades |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **#** | **Atividades** | **jan/21** | | **fev/21** | | **mar/21** | | **abr/21** | | **mai/21** | | **jun/21** | | | 1. | Desenvolvimento Monografia |  |  |  |  |  |  |  |  |  |  |  |  | | 2. | Desenvolvimento Hardware |  |  |  |  |  |  |  |  |  |  |  |  | | 3. | Desenvolvimento Software |  |  |  |  |  |  |  |  |  |  |  |  | | 4. | Desenvolvimento Firmware |  |  |  |  |  |  |  |  |  |  |  |  | | 5. | Testes |  |  |  |  |  |  |  |  |  |  |  |  | |
| Fonte: Autoria Própria |

* + 1. Recursos

Para o desenvolvimento da pesquisa e prototipação do dispositivo, foram necessários os itens apresentados na tabela a seguir.

|  |
| --- |
| Tabela 2 – Recursos |
| |  |  |  |  | | --- | --- | --- | --- | | **Recurso** | **Quantidade** | **Custo** | **Já Possuía?** | | Protoboard | 2 | R$ 20,00 | Sim | | Jumps | 50 | R$ 9,90 | Não | | LilyGo T-Call (ESP32) | 1 | R$ 150,00 | Não | | Raspberry Pi 4 | 1 | R$ 450,00 | Não | | SDCard 64GB | 1 | R$ 70,00 | Não | | Módulo CAN BUS SN65HVD230 | 1 | R$ 25,90 | Não | | OLED SSD1306 | 1 | R$ 35,00 | Não | | RTC 3231 | 1 | R$ 25,00 | Não | | **Total** | **58** | **R$ 785,80** |  | |
| Fonte: Autoria Própria |

O presente estudo será dividido em ??? capítulos sendo estes:

* 1 – Introdução incluindo apresentação dos objetivos, metodologia e delimitação do problema;
* 2 – Referencial teórico sobre definição, arquitetura e tecnologias;

1. **Revisão Bibliográfica**

Neste capítulo são apresentados os principais conceitos teóricos necessários para a compreensão deste trabalho além de contextualizar a relevância da pesquisa para o meio acadêmico.

A seção 2.1 descreve todos os hardwares que foram utilizados no projeto.

A seção 2.2 descreve todos os softwares e protocolos que foram utilizados no projeto

* 1. **Hardware**

Para a construção do protótipo deste projeto, foi necessário realizar a separação da arquitetura em duas partes, onde a primeira parte é responsável pela coleta de dados do veículo e a segunda é responsável pelo processamento das informações coletadas.

A coleta de informações é feita através da interface OBDII do carro pelo transceptor SN65HVD230 utilizando a rede CanBus para comunicação. Os dados recebidos pelo transceptor são enviados ao LilyGo, que utiliza ESP32 como MCU. No processamento dos dados, o MCU coleta as informações de data e hora, através do RTC DS3231, exibe os dados no OLED SSD1306, e envia um JSON com as informações através do módulo GSM SIM800L integrado a placa LilyGo.

Todo o processamento dos dados e exposição da API para consulta dos dados é feito no Raspberry PI 4.

* + 1. **Raspberry PI 4**

O Raspberry PI é um pequeno computador de baixo custo, criado pela Federação Raspberry PI Em 2006 no Reino Unido. Sua primeira versão disponível para venda, teve o preço de $35 dólares, muito abaixo dos valores de computadores pessoais encontrados nas lojas. A ideia dos criadores era desenvolver um produto com um preço acessível, que fosse pequeno e funcional, capaz de integrar facilmente o desenvolvimento de projetos eletrônicos com software. Atualmente esta placa contém 4 grandes versões, cada versão lançada se refere a atualização do modelo anterior. A última versão disponível do Raspberry PI é a 4, ela possui apenas o modelo B, que é equipado com o processador da Broadcom, o BCM2711, Quad core Cortex-A72 (ARM v8), processamento de 64-bit SoC e frequência de 1.5GHz, três opções de escolha para a memória RAM, podendo ser de 2GB , 4GB ou 8GB LPDDR4-3200 SDRAM, Conectividade WiFi 2.4 GHz e 5.0 GHz IEEE 802.11ac, Bluetooth 5.0, BLE, Gigabit Ethernet, 2 portas USB 3.0 e 2 portas USB 2.0. O Raspberry também possui 2 saídas de vídeo micro-HDMI, com capacidade de transmissão 4K em cada monitor. O dispositivo não contém nenhum equipamento para o armazenamento de dados integrado a placa, porém, é compatível com cartões micro SD, sendo necessário que o usuário insira um cartão com o sistema operacional para que ele funcione. Na figura 1, é possível ver um exemplar do Raspberry 4 model B.

|  |
| --- |
| Figura 1 – Raspberry Pi 4 – Model B |
|  |
| Fonte: Retirado de [1] |

* + 1. **LilyGo TTGO T-Call**

A LilyGo TTGO T-Call é uma placa de desenvolvimento de hardware baseado no MCU ESP32, integrado a uma placa GSM SIM800L. O ESP32 é fabricado pela Espressif e foi criado em 2016 com o intuito de facilitar e baratear o desenvolvimento de dispositivos inteligentes, além de melhorar seu antecessor, o ESP8266. Para isso o ESP32 possui as seguintes configurações: conectividade WIFI, Bluetooth e BLE, têm processamento ajustável, podendo variar de 80 MHz a 240MHz, baixo consumo de energia e memória flash suficiente para grandes aplicações. A versão utilizada do ESP32 é a WROVER-B, com 4MB de Flash SPI e 8MB de PSRAM. O SIM800L, que também integra o LilyGo, é um módulo para comunicação GSM que necessita de um chip para seu funcionamento e possui interfaces de entrada e saída de áudio que estão integradas nos pinos do LilyGo. A placa de desenvolvimento é alimentada por uma entrada USB-C e têm tensão de funcionamento entre 2.7v e 3.6v. Na figura 2 é possível ver um exemplar da placa.

|  |
| --- |
| Figura 2 - LilyGo TTGO T-Call |
|  |
| Fonte: Retirado de [2] |

* + 1. **SN65HVD230**

O SN65HVD230 é um transceptor para comunicação com a rede CAN (*Controller Area Network*). Esse dispositivo foi projetado para trafegar dados a taxas de até 1 megabit por segundo (Mbps), e incluem muitos recursos de proteção que fornecem aos dispositivos e a rede CAN mais robustez. Os dispositivos destinam-se ao uso em aplicações que empregam a camada física de comunicação serial CAN de acordo com o padrão ISO 11898. Na figura 3 é possível ver um exemplar do módulo.

|  |
| --- |
| Figura 3 – Módulo SN65HVD230 CAN Bus |
|  |
| Fonte: Autoria Própria |

* + 1. **SSD1306**

SSD1306 é um driver CMOS OLED / PLED de chip único com controlador para emissão de luz (orgânica / polímero) em sistema de exibição gráfica por matriz de pontos de diodo. Consiste em 128 segmentos e 64 comuns. Este IC é projetado para painel OLED tipo cátodo comum. O SSD1306 incorpora controle de contraste, RAM de exibição e oscilador, o que reduz o número de componentes externos e consumo de energia. Possui controle de brilho de 256 etapas. Dados / comandos são enviado do MCU geral através da interface paralela compatível com a série 6800/8000 selecionável por hardware, I2C ou Serial. É adequado para muitas aplicações portáteis compactas, como sub-display de telefone celular, MP3 player, calculadora etc. Na figura 4 é possível ver um exemplar do visor.

|  |
| --- |
| Figura 4 – Módulo OLED SSD1306 |
|  |
| Fonte: Autoria Própria |

* + 1. **RTC DS3231**

O DS3231 é um relógio de tempo real I2C (RTC – *Real Time Clock*) extremamente preciso e de baixo custo, possui um oscilador de cristal com compensação de temperatura (TCXO) e cristal integrado. O dispositivo incorpora uma entrada de bateria e mantém uma cronometragem precisa quando a alimentação principal do dispositivo é interrompida. A integração do ressonador de cristal aumenta a precisão de longo prazo do dispositivo e reduz a contagem de peças em uma linha de fabricação. Na figura 5 é possível ver um exemplar do módulo.

|  |
| --- |
| Figura 5 – Módulo RTC DS3231 |
|  |
| Fonte: Autoria Própria |

* 1. **Software**

Esta seção aborda assuntos referentes à camada de software que orquestra a comunicação e o processamento de dados. Os protocolos de comunicação entre os dispositivos, o IDE utilizado na construção do firmware e o sistema operacional, são o tema principal desta seção. Na contextualização destes softwares fica claro a preferência por ferramentas open-source, que geralmente possuem uma grande comunidade de desenvolvedores envolvidas.

* + 1. **Linux**

O Linux é um sistema operacional de código aberto desenvolvido por Linus Torvalds e lançado sua primeira versão em 1992. O desenvolvimento do sistema operacional ainda é liderado pelo seu criador, porém, conta com a ajuda de diversos programadores ao redor do mundo, uma vez que está disponível na internet para que qualquer pessoa possa fazer modificações. O objetivo do sistema Linux era inicialmente replicar e superar um sistema operacional já existente na época, o Minix, porém, com a política de código aberto, o sistema se tornou muito popular e hoje em dia é possível encontrar diversas versões deste SO rodando em computadores pessoais e servidores. O Kernel do Linux recebe diversas atualizações da comunidade de código aberto e em 2002 eles lançaram uma atualização para o Arch Linux, que tornou ele compatível com os processadores de arquitetura ARM, o que permitiu que vários dispositivos que utilizam processadores baseados nessa arquitetura pudessem agora rodar um sistema operacional Linux.

A Raspberry PI Foundation desenvolveu o sistema operacional Raspberry PI OS para seus dispositivos a partir do código fonte do Debian, uma distribuição do Linux.

* + 1. **GIT**

O GIT é um sistema de controle de versão de arquivos. Através deles é possível desenvolver projetos na qual diversas pessoas podem contribuir simultaneamente, editando e criando arquivos, permitindo que eles possam existir sem o risco de suas alterações serem sobrescritas. Desenvolvido em 2005 por Linus Torvalds precisamente para a criação do Kernel do Linux, atualmente ele é amplamente utilizado para colaboração de times de desenvolvimento nas empresas e em comunidades de software.

* + 1. **GitHub**

O GitHub é uma plataforma web que oferece hospedagem de código, além da implementação gratuita dos mecanismos do GIT. Ele oferecer um espaço limitado, porém gratuito, para que qualquer usuário possa armazenar seus códigos-fonte de forma segura na nuvem. O GitHub é uma das ferramentas mais conhecidas na comunidade da internet pela quantidade de softwares open-source que eles hospedam. O GitHub foi desenvolvido por Chris Wanstrath, J. Hyett, Tom Preston-Werner e Scott Chacon, e foi lançado em fevereiro de 2008. A empresa, GitHub, Inc., existe desde 2007 e está sediada em São Francisco.

* + 1. **VS Code**

O Visual Studio Code é um editor de código-fonte (texto) multiplataforma disponibilizado pela Microsoft para o desenvolvimento de aplicações com suporte a diversas linguagens. foi anunciado, com uma versão prévia lançada, em 29 de abril de 2015 pela Microsoft na conferência Build de 2015.

* + 1. **PlatformIO**

PlatformIO é uma plataforma para desenvolvimento de firmwares em sistemas embarcados. A plataforma é open-source e possui seu código aberto para que outros desenvolvedores possam contribuir para seu desenvolvimento. A ferramenta tem o principal objetivo de transformar a experiencia do desenvolvimento de firmware em algo simples. Teve sua primeira versão lançada em 2014 no GitHub, até o momento, já foram lançadas mais de 107 versões do software. A plataforma é compatível com as mais modernas placas de desenvolvimento de hardware como o Arduino e o ESP32. Além de promover um ambiente amigável ao desenvolvedor, a ferramenta conta com alguns recursos de auto completar, análises estáticas de código e sugestões de melhores práticas durante o desenvolvimento.

* + 1. **Node-Red**

Node-RED é uma ferramenta de programação open-source que tem como objetivo conectar dispositivos de hardware, API (*Application Programming Interface*) e serviços online de maneira inovadora e simples. Ele fornece um editor web, acessado pelo navegador, que torna fácil a experiencia do usuário. A ferramenta possui diversos módulos pré-configurados que são capazes de executar funções que muitas vezes são demoradas em sua implementação, como conectores de banco de dados e exposição de end-points de API. As funções desenvolvidas na ferramenta podem ser facilmente exportadas para JSON (JavaScript Object Notation) e importadas novamente na ferramenta.

* + 1. **MongoDB**

MongoDB é um banco de dados não relacional open-source, que armazena seus dados no formato de documentos, ele foi projetado para facilitar o desenvolvimento e o dimensionamento de aplicações que precisam de alta velocidade de resposta. Os dados armazenados no MongoDB estão no formato JSON, que permite criar dados com alta complexidade de informações e estrutura de dados.

* + 1. **MQTT (Protocolo)**

O MQTT (*Message Queue Telemetry Transport*) foi inventado e desenvolvido pela IBM no final dos anos 90. Sua aplicação original era vincular sensores em pipelines de petróleo a satélites. Como seu nome sugere, ele é um protocolo de mensagem com suporte para a comunicação assíncrona entre as partes. Um protocolo de sistema de mensagens assíncrono desacopla o emissor e o receptor da mensagem tanto no espaço quanto no tempo e, portanto, é escalável em ambientes de rede que não são confiáveis. Apesar de seu nome, ele não tem nada a ver com filas de mensagens, na verdade, ele usa um modelo de publicação e assinatura. No final de 2014, ele se tornou oficialmente um padrão aberto OASIS (*Organization for the Advancement of Structured Information Standards*), com suporte nas linguagens de programação populares, usando diversas implementações de software livre. Com isso, muitos softwares foram lançados implementando essa tecnologia que se popularizou entre os dispositivos de IoT.

* + 1. **MQTT Mosquitto**

MQTT Mosquitto é um software open-source de mensageria que implementa o protocolo MQTT versões 5.0, 3.1.1 e 3.1. O Mosquitto foi projeto para ser leve e adequado para uso em todos os dispositivos, desde microcontroladores, com o ESP32 até Servidores.

* + 1. **Protocolo CAN**

Muito utilizado na indústria automobilística, o protocolo CAN Bus (Barramento Controller Area Network) foi desenvolvido pela empresa alemã Robert BOSCH e disponibilizado nos anos 80. Sua aplicação inicial foi em ônibus e caminhões. Atualmente, é utilizado em veículos automotivos, navios e até tratores.

O CAN é um protocolo de comunicação serial síncrono. O sincronismo entre os módulos conectados à rede é feito em relação ao início de cada mensagem lançada ao barramento, evento que ocorre em intervalos de tempo regulares. Trabalha baseado no conceito multi-mestre, onde todos os módulos podem se tornar mestres em determinado momento e escravos em outro, além de suas mensagens serem enviadas em regime multicast, caracterizado pelo envio de toda e qualquer mensagem para todos os módulos existentes na rede. Outro ponto forte deste protocolo é o fato de ser fundamentado no conceito CSMA/CD with NDA (Carrier Sense Multiple Access / Collision Detection with Non-Destructive Arbitration). Isto signiﬁca que todos os módulos veriﬁcam o estado do barramento, analisando se outro módulo está ou não enviando mensagens com maior prioridade. Caso isto seja percebido, o módulo cuja mensagem tiver menor prioridade cessará sua transmissão e o de maior prioridade continuará enviando sua mensagem deste ponto, sem ter que reiniciá-la. As mensagens que trafegam nesta rede possuem quatro tipos de pacotes de informação, são eles:

* Quadro de dados (Data Frame)
* Quadro Remoto (Remote Frame)
* Quadro de Erro (Error Frame)
* Quadro de sobrecarga (Overload Frame)

Neste artigo, foi explorado apenas o quadro de dados, uma vez que ele foi suficiente para o desenvolvimento da pesquisa. Ele é composto por sete grupos de bytes que formam um pacote, são eles:

* *Start of Frame (SoF)*: Início do quadro, com apenas um bit que denota o início da transmissão do frame de dados.
* *Arbitration Field*: Composto por 12 bits, sendo 11 bits reservados para a identificação única do pacote, e um bit para o RTR - *Remote transmission request*, que determina se o Quadro de Dados é dominante, quando 0, ou recessivo, quando 1.
* *Control*: Composto por 6 bits, sendo 1 bit reservado para identificação do formato do campo anterior, 0 quando o identificador contiver 11 dígitos e 1 para identificador estendido, em seguida há um bit reservado com valor 0 e por último há mais 4 bits que determinam o tamanho do grupo a seguir, que varia de 0 a 64 bits.
* *Data*: Composto por 64 bits, é o dado que será transmitido pelo frame.
* *Cyclic redundancy check (CRC)*: Composto por 16 bits, sendo 15 deles utilizados para validação de erros de transmissão na rede CAN e 1 bit como delimitador.
* *Acknowledge (ACK)*: Composto por 2 bits, sendo 1 bit para o ACK Slot e o outro bit para ACK Delimiter. É usado pela rede pra que o destinatário possa indicar o correto recebimento do pacote.
* *End of Frame (EoF)*: Composto por 7 bits de valor 1 (Recessivos) que delimitam o fim do quadro de dados.
  + 1. **Fritzing**

O Fritzing é um software de código aberto, para desenvolvimento de protótipos de circuitos eletrônicos. Ele foi desenvolvido na *Fachhochschule Potsdam* (*University of Applied Sciences Potsdam*), na Alemanha, e lançado em sua primeira versão 0.1b em novembro de 2008. Ele permite que sejam criados layouts de circuitos eletrônicos de maneira simples e rápida além de permitir ao usuário realizar a documentação dos esquemáticos e seus componentes.

1. **Desenvolvimento**

Falar sobre a utilização do carro específico.

* 1. **Compreensão da rede CAN no contexto automotivo**

Amplamente implementada pela indústria automotiva, a rede CAN se encontra na maioria dos carros que hoje circulam nas ruas. O acesso a rede se da através da interface OBDII, que se tornou obrigatória desde 2010 no Brasil. A interface, muitas vezes escondida e de difícil acesso dentro dos carros, tem a forma de um trapézio com 16 entradas para conexões que dão suporte a implementação de protocolos na rede de comunicação do veículo, dentre eles, o protocolo CAN, conforme exibido na figura 6.

|  |
| --- |
| Figura 6 – Interface OBD II |
|  |
| Fonte: Retirado de [5] |

Com a ISO 15765-4, determinou-se os requerimentos para a aplicação da rede CAN em OBD. Sua velocidade pode chegar até 500 kbps, onde são utilizados os pinos 6 (CAN High) e 14 (CAN Low) do conector SAE J2480, conforme figura 6.

A rede CAN automotiva possui 10 serviços previstos para implementação, porém nem todos os sistemas suportam todos os serviços e cada veículo ou módulo de motor/transmissão automática pode ser configurado pelo fabricante para atender aos serviços suportados, alinhado com o requerimento de sua legislação.

Neste artigo, será abordado apenas o serviço 1, pois para o desenvolvimento da pesquisa não foi necessário a exploração dos demais serviços, uma vez que este módulo supre as necessidades da pesquisa.

O serviço 1 oferece os dados de informações correntes, relacionadas ao *powertrain*[[1]](#footnote-1), onde é possível solicitar as informações através de um código de identificação, chamado *PID* *– Parameter Identification*. Como a implementação de serviços e informações podem ser customizadas dependendo do veículo, é possível descobrir quais informações estarão disponíveis através de uma requisição feita com o PID 00 para o MCU do veículo. A lista de informações disponíveis para consulta, pode ser encontrada no livro “*THE CAR HACKER’S HANDBOOK*” [6]. Seguindo a proposta da pesquisa, os dados necessários para o desenvolvimento são: VIN Code *(Veiche Identification Number)*, a rotação do motor em RPM e a velocidade do veículo em Km/h. Todas informações necessárias podem ser acessadas por qualquer dispositivo que se comunique nos padrões estabelecidos pelo CAN, basta conhecer o PID correspondente a informação que deseja consultar.

* 1. **Montagem Inicial do Hardware – IoT**

A montagem do hardware foi iniciada com foco na principal funcionalidade em que ele foi planejado, coletar informações do veículo. As seções a seguir descrevem bem o processo de montagem do que será o primeiro protótipo.

* + 1. **Prototipação**

Na prototipação, a primeira etapa foi de separação de todos os hardwares necessários montagem do dispositivo IoT. Foram escolhidas as seguintes peças:

* 1x Mini Protoboard de 170 pontos
* 1x Cabo USB-C
* 1x LilyGo TTGO T-Call
* 1x SN65HVD230
* 8x Jumpers Macho – Fêmea

Com as peças em mãos, a etapa seguinte foi a criação do protótipo na ferramenta Fritzing. O protótipo final, que pode ser visto na Figura 7, foi utilizado para a montagem correta dos componentes na protoboard.

|  |
| --- |
| Figura 7 – Protótipo |
|  |
| Fonte: Autoria Própria |

* + 1. **Setup IDE**

Após a construção do protótipo, a próxima etapa foi o desenvolvimento do firmware, que para esta pesquisa se deu através do IDE PlatformIO.

* 1. **Desenvolvimento Firmware**
     1. **Interação do dispositivo IoT com a rede CAN**
     2. **Conexão IoT e interface OBD II**
     3. **Leitura de dados do veículo**
  2. **Montagem do Hardware – Raspberry**
     1. **Instalação do Linux**
     2. **Setup Node-Red**
     3. **Setup MQTT Mosquitto Broker**
     4. **Setup MongoDB**
  3. **Desenho da Arquitetura do projeto**
  4. **Desenvolvimento do Software/Firmware**
  5. **Interface com o usuário**
  6. **Testes e Resultado**

4. Conclusão

**4.1 Sugestões de melhoria e trabalhos futuros**

**REFERÊNCIAS**

[1] Raspberry PI Foundation. In: Product Brief. **Raspberry Pi 4 Model B.** 2021. Disponível em: <https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf>. Acesso em 15 mai. 2021.

[2] LilyGo. LILYGO® TTGO T-Call V1.4. Disponível em: <http://www.lilygo.cn/prod_view.aspx?TypeId=50044&Id=1127&FId=t3:50044:3>. Acesso em 15 mai. 2021.

[3] SN65HDV2X. SN65HDV23. Disponível em: <https://www.ti.com/lit/ds/symlink/sn65hvd23.pdf>. Acesso em 15 mai. 2021.

[4] *OBD2 Explained - A Simple Intro (2021)*. Disponível em: <https://www.csselectronics.com/screen/page/simple-intro-obd2-explained/language/en>. Acesso em 24 mai. 21

[5] *Which OBD2 Protocol Is Supported By My Vehicle?* Disponível em: <https://obdstation.com/obd2-protocols/>. Acessado em 24 mai. 21

[6] *THE CAR HACKER’S HANDBOOK*. Disponível em: <http://opengarages.org/handbook/ebook/>. Acesso em 24 mai. 21

1. Powertrain: Trem de força: São os componentes responsáveis por gerar potência para entregar às rodas, sendo composto pelo motor, transmissão, eixo cardã e diferencial. [↑](#footnote-ref-1)