



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA MECÂNICA
CURSO DE ENGENHARIA MECÂNICA

Gabriel Campos Bernardo

**IMPLEMENTAÇÃO DE PLATAFORMA DE COLETA,
ARMAZENAMENTO, TRATAMENTO E DISPONIBILIDADE DE
DADOS EM CARRO DE FÓRMULA SAE ELÉTRICO**

**Florianópolis
2025**

Gabriel Campos Bernardo

**IMPLEMENTAÇÃO DE PLATAFORMA DE COLETA,
ARMAZENAMENTO, TRATAMENTO E DISPONIBILIDADE
DE DADOS EM CARRO DE FÓRMULA SAE ELÉTRICO**

Trabalho apresentado ao Curso de Graduação em Engenharia Mecânica da Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do título de Engenheiro Mecânico.

Orientador: Prof. Lauro Cesar Nicolazzi, Dr.Eng.

Coorientador: Luís Felipe Ávila D'aquino Noronha,
Me. Eng.

Florianópolis, 2025

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Campos Bernardo, Gabriel
Implementação de plataforma de coleta, armazenamento,
tratamento e disponibilidade de dados em carro de fórmula
SAE elétrico / Gabriel Campos Bernardo ; orientador, Lauro
Cesar Nicolazzi, coorientador, Luís Felipe Ávila D'aquino
Noronha, 2025.
91 p.

Trabalho de Conclusão de Curso (graduação) -
Universidade Federal de Santa Catarina, Centro Tecnológico,
Graduação em Engenharia Mecânica, Florianópolis, 2025.

Inclui referências.

1. Engenharia Mecânica. 2. Tratamento de Dados. 3.
Automobilismo. 4. Aquisição de Dados. I. Cesar Nicolazzi,
Lauro. II. Ávila D'aquino Noronha, Luís Felipe. III.
Universidade Federal de Santa Catarina. Graduação em
Engenharia Mecânica. IV. Título.

Gabriel Campos Bernardo

IMPLEMENTAÇÃO DE PLATAFORMA DE COLETA, ARMAZENAMENTO, TRATAMENTO E DISPONIBILIDADE DE DADOS EM CARRO DE FÓRMULA SAE ELÉTRICO

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Engenheiro Mecânico e aprovado em sua forma final pelo Curso de Engenharia Mecânica

Florianópolis, 11 de Julho de 2025.



Documento assinado digitalmente
Antonio Carlos Valdiero
Data: 22/07/2025 09:44:10-0300
CPF: ***.227.377-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Antônio Carlos Valdiero, Dr.Eng.
Coordenador do Curso



Documento assinado digitalmente
LAURO CESAR NICOLAZZI
Data: 22/07/2025 09:20:34-0300
CPF: ***.706.519-**
Verifique as assinaturas em <https://v.ufsc.br>

Prof. Lauro Cesar Nicolazzi, Dr.Eng.
Orientador

Prof. Rodrigo de Souza Vieira, Dr.Eng.
Universidade Federal de Santa Catarina

Prof. Jonny Carlos da Silva, Dr.Eng.
Universidade Federal de Santa Catarina

Florianópolis
2025

Dedico este trabalho ao meu avô, Luzenir Teixeira, pelos inúmeros incentivos a buscar racionalmente entender o porquê dos fenômenos do universo.

Agradecimentos

Chegar até onde cheguei nesta graduação só foi possível graças ao apoio de muitas pessoas na minha vida.

O primeiro e mais importante apoio de todos foi o de meus pais, Clarissa Campos da Silva Bernardo e Carlos Roberto Bernardo, que lutaram pela minha educação desde minhas primeiras memórias, sempre apoiando meus estudos e aprendizados, apesar de todas as dificuldades ao longo de nossa história – e foram muitas, eu sei. Destaco, pai, o seu papel no meu amor por carros e automobilismo, mas especialmente por ter me ensinado que trabalho duro leva à satisfação. Destaco, mãe, o seu papel no meu amor pela matemática e por ter apoiado muito no meu ensino básico, onde entendi fundamentos que me apoiaram, e muito, na graduação. Por este apoio incondicional, eu agradeço de coração.

A todo o restante de minha família, meus avós maternos e paternos por ter construído essa história linda, meus tios e tias por serem inspirações, ficam meus agradecimentos. Agradeço em especial meu avô, Luzenir, que não está mais conosco, mas que me ajudou a construir quem eu sou hoje, nos incontáveis momentos de minha infância que ele me ensinou, talvez até inconscientemente, a ser um homem racional, objetivo e sério. Minha irmã, Alice, obrigado por ser tão especial e compartilhar comigo um gosto musical impecável, conte comigo sempre.

Ao professor Lauro Nicolazzi, agradeço muito pela orientação e apoio, que apareceu quando eu mais precisava, após muitas dificuldades para conciliar a vida profissional com a vida acadêmica. Professores como você são o motivo pelo qual este departamento é tão prestigiado e suas aulas (tanto elemaq quanto veículos) foram sempre inspiradoras e estarão para sempre em minha memória.

A minha namorada, Victoria Knih, eu agradeço por compartilhar sua vida comigo diariamente, com muito amor e muita leveza. Seu apoio foi fundamental para conquistarmos tudo que conquistamos nos últimos anos e espero conquistar muito mais ao seu lado. Este trabalho se deve em peso a motivação que você me traz, de me mostrar que sou capaz e me trazer para cima nos momentos de desânimo.

Ao meu amigo Luis Schiavo, você é minha inspiração! Aos meus amigos Matheus Dettmar e Bianca Constante, dos quais eu tenho prazer de ser padrinho de casamento, sou grato pela eterna amizade. Sou muito grato por ter encontrado vocês três no meu caminho nesta graduação, pois não sei o que faria sem os melhores amigos da minha vida.

Ao meu grande amigo Noronha, coorientador deste trabalho, agradeço muito pela amizade ao longo desses anos, por ter compartilhado comigo o amor pelo automobilismo e parte desse vasto conhecimento seu, que está se refletindo em uma carreira brilhante. Voe, amigo!

Ao restante dos meus amigos do *Griffinóia*, vocês fizeram todos esses anos serem mais fáceis, levo comigo tantas memórias e espero levá-los próximo de mim para todo sempre. Contem sempre comigo!

Aos colegas da 14.2, foi um prazer compartilhar esta jornada com vocês. Obrigado por todas as experiências compartilhadas e noites mal dormidas.

*“As coisas são como são no nosso universo
porque, se não fossem, não estaríamos aqui
para perceber.”
(Brian Greene, 1999)*

Resumo

Este trabalho propõe uma plataforma integrada para coleta, armazenamento, organização, tratamento e análise de dados de sensores embarcados em um protótipo de veículo FSAE Elétrico da equipe Ampera Racing da Universidade Federal de Santa Catarina. O objetivo principal é apoiar o processo de validação do projeto dos sistemas e subsistemas do carro, fornecendo informações cruciais para aprimorar o desempenho. A solução se integra ao sistema de aquisição de dados existente no protótipo, que utiliza sensores conectados a um barramento CAN, um computador embarcado Jetson Orin e a plataforma de dados Foxglove. A proposta expande a capacidade de tratamento desses dados, organizando-os em um banco de dados PostgreSQL em formato tabular. Para garantir a recuperação e a análise eficiente, utiliza-se a linguagem Python para extrair dados do Foxglove e gerar relatórios, transformações e cálculos matemáticos. Por fim, a plataforma é aplicada na geração de um relatório com informações essenciais para a análise da prova de aceleração da competição Fórmula SAE, demonstrando seu potencial para otimizar a avaliação de desempenho do veículo.

Palavras-chaves: tratamento de dados, aquisição de dados, análise de dados, automobilismo.

Abstract

This work proposes an integrated platform for the collection, storage, organization, processing, and analysis of data from sensors embedded in an Electric FSAE vehicle prototype from the Ampera Racing team at the Federal University of Santa Catarina. The main objective is to support the design validation process of the car's systems and subsystems, providing crucial information to enhance performance. The solution integrates with the prototype's existing data acquisition system, which uses sensors connected to a CAN bus, a Jetson Orin embedded computer, and the Foxglove data platform. The proposed platform expands the data processing capability, organizing the data in a PostgreSQL database in tabular format. To ensure efficient retrieval and analysis, Python is used to extract data from Foxglove and generate reports, transformations, and mathematical calculations. Finally, the platform is applied to generate a report with essential information for the analysis of the Formula SAE competition's acceleration event, demonstrating its potential to optimize vehicle performance evaluation.

Key-words: data transformation, data acquisition, data analysis, motorsport.

Lista de ilustrações

Figura 1 – Imagem ilustrando componentes de um carro de fórmula 1 que habilitam sensoriamento.	34
Figura 2 – Diagramas comparando processos de ETL e ELT.	36
Figura 3 – Exemplo de uma tabela mostrando dados do odômetro do carro da Ampera Racing repousados em um sistema de banco de dados PostgreSQL. A visualização gráfica obtida nesta imagem é fornecida através da ferramenta DBeaver.	38
Figura 4 – Exemplo dos atributos de uma tabela de dados de odometria do carro da Ampera Racing conforme demonstrado na ferramenta DBeaver.	38
Figura 5 – Exemplo de um sinal ruidoso, separando os componentes de sinal e ruído.	43
Figura 6 – Representação matemática das coordenadas sensibilizadas pelo IMU.	44
Figura 7 – Modelo representando um sensor IMU.	44
Figura 8 – Resposta em frequência de um filtro <i>Butterworth</i> em função da ordem.	46
Figura 9 – Força longitudinal em função da razão de escorregamento	49
Figura 10 – Diagrama de blocos representando a interação entre os componentes de <i>software</i> e os processos executados.	55
Figura 11 – Diagrama de blocos demonstrando a containerização do <i>software</i> em um contêiner Docker, executando dentro de um computador e suas comunicações internas ou externas.	57
Figura 12 – Fluxograma dos componentes de <i>hardware</i>	59
Figura 13 – Fluxograma total mostrando as etapas de Aquisição de Dados, Extração e Carregamento e Transformação	61
Figura 14 – Exemplo de visualização da tela de mensagens do tópico na Foxglove durante prova de <i>Autocross</i> no simulador, mostrando gráficos de velocidade longitudinal do carro e velocidade angular das quatro rodas.	61
Figura 15 – Exemplo de mensagens de IMU e Odometria contidas em uma <i>bag</i> do ROS2 organizada através do formato MCAP.	63
Figura 16 – Exemplo dos dados do IMU já organizados em um DataFrame do Pandas, manipulado através do Python.	63
Figura 17 – Tela da Foxglove mostrando gravações de ROS2 do carro sendo armazenadas na Foxglove.	64
Figura 18 – Fluxograma demonstrando papel do <i>pipeline</i> em coletar os dados brutos e transformá-lo em dados tratados.	67
Figura 19 – Diagrama funcional do <i>pipeline</i> de Extração e Carregamento (EL)	68
Figura 20 – Diagrama automaticamente criado no Mage através da construção do código do <i>pipeline</i> de Extração e Carregamento (EL).	69

Figura 21 – Diagrama funcional do <i>pipeline</i> de Relatório de Aceleração.	70
Figura 22 – Velocidade e distância do carro do simulador em uma prova de aceleração, destacando o percurso de 75 metros da prova, o tempo total da prova e a velocidade final ao atingir a marca de 75 metros.	71
Figura 23 – Aceleração longitudinal ao longo do tempo, destacando marca de 75 metros.	72
Figura 24 – Comportamento da aceleração no espectro de frequência, calculado através da Transformada Discreta de Fourier	73
Figura 25 – Pesquisa de Grade (ou <i>grid search</i> , no inglês) utilizada para auxiliar o método de inspeção visual.	74
Figura 26 – Aceleração original e filtrada em função do tempo. Filtro utilizado: Butterworth. Parâmetros utilizados: frequência de corte = 1 Hz e ordem = 3.	75
Figura 27 – Velocidade e distância percorrida calculados através da aceleração filtrada.	75
Figura 28 – Taxa de deslizamento calculada para as 4 rodas para uma tomada de dados da prova de aceleração via simulador FSDS (<i>Formula SAE Driving Simulator</i>). O raio do pneu utilizado nesta análise foi de 0,2286 metros, conforme configurado no simulador.	78
Figura 29 – Aceleração, velocidade e distância derivadas da velocidade do sensor IMU e filtradas através de um filtro <i>Butterworth</i> com frequência de corte de 2 Hz e ordem 2.	79
Figura 30 – Gráficos de aceleração pelo tempo e aceleração por velocidade conforme apresentada no relatório final do DAS.	80

Lista de tabelas

Tabela 1 – Frequências normalizadas e em Hz para $F_s = 1000$ Hz com $N = 8$	41
Tabela 2 – Interpretação do resultado de λ durante aceleração ou frenagem.	49
Tabela 3 – Regras de classificação do estado do carro com base na distância e velocidade longitudinal.	72
Tabela 4 – Número de amostras, tempo total e frequência de amostragem para um exemplo de prova de aceleração.	76

Lista de abreviaturas e siglas

ABS	<i>Anti-lock Braking System</i>
ADEV	<i>Allan Deviation</i>
API	<i>Application Programming Interface</i>
AWS	<i>Amazon Web Services</i>
CAN	<i>Controller Area Network</i>
CPU	<i>Central Processing Unit</i>
CSV	<i>Comma-Separated Values</i>
DAQ	<i>Data Acquisition</i>
DAS	<i>Data Analytics Software</i>
DOF	<i>Degrees of Freedom</i>
EC2	<i>Elastic Compute Cloud</i>
ELT	<i>Extract, Load and Transform</i>
ETL	<i>Extract, Transform and Load</i>
FSAE	<i>Formula Society of Automotive Engineers</i>
FSDS	<i>Formula SAE Driving Simulator</i>
GNSS	<i>Global Navigation Satellite System</i>
GPU	<i>Graphics Processing Unit</i>
GPS	<i>Global Positioning System</i>
IDE	<i>Integrated Development Environment</i>
IMU	<i>Inertial Measurement Unit</i>
INS	<i>Inertial Navigation System</i>
MCAP	<i>Message Capture</i>
OADEV	<i>Overlapped Allan Deviation</i>

OLAP	<i>Online Analytical Processing</i>
OLTP	<i>Online Transaction Processing</i>
ROS 2	<i>Robot Operating System 2</i>
RTOS	<i>Real Time Operating System</i>
SaaS	<i>Software as a Service</i>
SAE	<i>Society of Automotive Engineers</i>
SQL	<i>Structured Query Language</i>
TCS	<i>Traction Control System</i>
UI	<i>User Interface</i>
UTF-8	<i>8-bit Unicode Transformation Format</i>
YAML	<i>Ain't Markup Language</i>

Lista de símbolos

A	Amplitude do sinal
F	Frequência de sinal
F_s	Frequência de amostragem
f	Frequência normalizada do sinal
k	Índice de frequência discretizada ou componentes de frequência
t	Tempo
T	Período de amostragem
n	Índice de tempo discretizado
N	Número total de amostras
φ	Deslocamento de fase do sinal
SNR	Razão de sinal para ruído
P	Potência de sinal
x	Coordenada espacial em x
y	Coordenada espacial em y
z	Coordenada espacial em z
T^{-1}	Desalinhamentos de calibragem em sensor IMU
K	Fator de escala em sensor IMU
b	Viés, ou <i>bias</i> , do sensor IMU
v_k	Ruídos em um sensor IMU
u_k	Quantidade física medida por um sensor IMU
y_k	Saída de sinal de um sensor IMU
$\vec{F}$	Vetor de força
$\vec{a}$	Vetor de aceleração

m	Massa
λ	Razão de escorregamento
v	Velocidade longitudinal do veículo
v_t	Velocidade tangencial da roda
$\dot{\theta}$	Velocidade angular da roda
r_{ef}	Raio efetivo da roda
a	Aceleração
H	Função de transferência de filtro <i>Butterworth</i>
m	Ordem de um filtro <i>Butterworth</i>
ω_c	Frequência de corte

Sumário

1	INTRODUÇÃO	27
1.1	Objetivos	28
1.1.1	Objetivo Geral	28
1.1.2	Objetivos Específicos	29
1.1.3	Estrutura do Trabalho	29
2	REVISÃO BIBLIOGRÁFICA	31
2.1	Sistemas Embarcados	31
2.1.1	Barramento CAN	32
2.1.2	Sensores	33
2.2	<i>Pipelines de Dados</i>	33
2.3	Banco de Dados	36
2.4	Tratamento de Sinais	38
2.4.1	Processamento de Sinais	38
2.4.2	Ruídos em Sensores	42
2.4.2.1	Modelagem de Sensores IMU	43
2.4.3	Filtro <i>Butterworth</i>	45
2.5	Escorregamento do Pneu	46
2.6	Estado da arte	49
3	MODELAGEM DO PROBLEMA	53
3.1	Análise do Problema	53
3.2	Requisitos Funcionais	54
3.3	Proposta de Solução	55
3.3.1	Arquitetura da Solução	55
3.4	<i>Hardware</i>	58
3.4.1	Aquisição de Dados	58
3.4.1.1	Sensores Utilizados	59
3.4.2	Jetson Orin	59
3.5	<i>Software</i>	60
3.5.1	Armazenamento dos Dados	60
3.5.2	Foxglove	63
3.5.3	Python	65
3.5.4	Mage	65
3.5.5	Visualização e Disponibilização de Dados	66
3.5.6	<i>Pipeline de Dados</i>	67

3.5.6.1	Extração e Carregamento	68
3.5.6.2	Transformação	69
3.6	Métodos	70
3.6.1	Tempo de Prova	70
3.6.2	Tratamento do sinal de aceleração	72
3.6.3	Junção dos dados	76
3.6.4	Cálculo do <i>Slip Ratio</i>	77
3.6.5	Diagrama Aceleração e Velocidade	77
4	RESULTADOS	81
4.1	Desempenho da Plataforma	81
4.2	Capacidade de Análise	82
4.3	Custos	83
4.4	Impacto no Projeto	84
5	CONCLUSÃO	85
5.1	Considerações Finais	85
5.2	Próximos Passos	86
	REFERÊNCIAS	89

1 Introdução

A Fórmula 1, categoria máxima do automobilismo mundial e a competição de carros de corrida mais avançada, é notavelmente conhecida pelo seu longo legado no aproveitamento do estado da arte das tecnologias automotivas. Um carro de Fórmula 1 é sinônimo de performance, refletida na sua capacidade de atingir velocidades acima de 350 km/h, fazer curvas, otimizar a degradação dos pneus, aumentar a eficiência aerodinâmica e muito mais. Todas essas áreas de performance são levadas ao extremo a cada ano que passa e tudo isso é graças à meticulosa engenharia focada em dados e inovação.

Um carro de Fórmula 1 gera 1,5 Terabytes de dados em um final de semana de corrida com informações provenientes de mais de 300 sensores embarcados. Esses dados são utilizados durante a corrida para tomada de decisão, estratégia de corrida, analisar ganhos de performance ou identificar potenciais defeitos. Porém, não são só os dados em tempo real que se mostram valiosos, pois o levantamento histórico desses dados ao longo de uma temporada permite identificar sistemas prioritários para serem melhorados, podem auxiliar no desenvolvimento de novos projetos e permitem analisar corridas passadas para, por exemplo, identificar pontos de feedback para o piloto ([SHAPIRO, 2023](#)).

Desde 1982, a Sociedade dos Engenheiros Automobilísticos (SAE) promove a competição Fórmula SAE, uma plataforma que oferece aos futuros engenheiros a oportunidade de projetar, desenvolver e competir com veículos de alto desempenho em um ambiente de engenharia multidisciplinar. A complexidade desses projetos exige o uso de ferramentas avançadas para projetar, testar, validar e otimizar os diversos sistemas envolvidos.

Nesse contexto estudantil, informações provenientes de sensores embarcados é também uma realidade nas competições da Fórmula FSAE Brasil. Todo ano, desde 2004, dezenas de equipes de universidades por todo o Brasil se comprometem a projetar, construir e testar um carro de corrida do tipo fórmula, em um projeto multidisciplinar que permite aos estudantes de diversas engenharias exercitarem não somente o seu conhecimento teórico aprendido nas disciplinas da graduação, como também aprenderem sobre diferentes áreas da engenharia, colocar em prática habilidades de projeto e construção. Além dos aspectos técnicos, o projeto de um carro de corrida permite aos membros exercitarem o trabalho em equipe, organização e responsabilidade com o projeto e o orçamento, exigindo soluções criativas e engenhosas para os inúmeros desafios ([SAE BRASIL, 2004](#)).

Na competição, os membros da equipe precisam demonstrar detalhadamente o projeto do carro e atender a uma série de regras da competição, exigindo que todos os subsistemas do carro estejam funcionando em conjunto. Conforme mencionado por [Li, Wang e Kang \(2021\)](#) em seu artigo sobre o desenvolvimento de aquisição de dados em um carro de Fórmula SAE,

garantir a integração desses subsistemas e fazê-los se comportarem de acordo com o projeto é uma tarefa árdua e, para isso, a aquisição e análise dos dados, que refletem os fenômenos físicos aos quais o carro está sujeito, permite, assim como na Fórmula 1, ter mais capacidade de validar o projeto, melhorar o desempenho do carro e identificar ajustes que seriam difíceis de observar de outra forma.

O objetivo de uma equipe de competição é desenvolver um carro que não somente seja capaz de desempenhar positivamente na competição, como também um carro que se baseie em um projeto completo e cauteloso de todos os subsistemas que o compõem. Com a finalidade de tirar maior proveito do seu projeto e replicar um carro que reflita as decisões tomadas, é fundamental que o carro seja construído com antecedência suficiente para serem feitos testes, que fornecem dados para as potenciais análises, permitindo melhorias, adaptabilidade e validação das decisões de projeto.

Com o objetivo de desenvolver um carro que desempenhe o máximo positivamente na competição, a equipe Ampera Racing da Universidade Federal de Santa Catarina, que desenvolve seus carros para participar da categoria de veículos elétricos da FSAE, estimula a valorização de dados obtidos de sensores através de uma área especializada dentro da equipe em Análise de Dados. Uma das entregas de valor da área é o fornecimento de tecnologias para coletar, armazenar, organizar e analisar os resultados dos testes para todas as outras áreas do time que irão utilizar esses dados para melhorar seus respectivos subsistemas.

Visando essa entrega de valor, este trabalho consiste em apresentar a implementação do Data Analytics Software (DAS), uma plataforma robusta de dados que se utiliza dos dados coletados por inúmeros sensores para fornecer inteligência e assertividade para os projetos, bem como para potenciais análises de desempenho e confiabilidade. O DAS é uma plataforma de dados projetada e desenvolvida pela e para a Ampera, com a motivação de aumentar a maturidade analítica necessária para construir um carro de sucesso em 2025, mas também permitir a evolução do projeto para os próximos anos. Portanto, neste trabalho é apresentada toda a metodologia por trás da construção do DAS, bem como a aplicação do mesmo no contexto do tratamento e geração de relatório de análise para uma prova de Aceleração, com todas as informações necessárias para melhorar o desempenho do carro nesta prova.

1.1 Objetivos

Nas seções que seguem estão descritos o objetivo geral e os objetivos específicos deste trabalho.

1.1.1 Objetivo Geral

O objetivo deste trabalho consiste em **desenvolver um sistema integrado de coleta, armazenamento, organização, tratamento, disponibilidade e análise de dados para o pro-**

tótipo FSAE Elétrico da equipe Ampera Racing, visando apoiar a validação do projeto, a otimização de desempenho e a tomada de decisões embasadas em dados. O sistema deve ser robusto, escalável e capaz de atender às demandas técnicas da competição, proporcionando uma ferramenta essencial para a melhoria contínua do veículo.

1.1.2 Objetivos Específicos

Com a finalidade de alcançar o objetivo geral, seguem os objetivos específicos que regem este trabalho:

- Desenvolver uma solução de integração da plataforma com os dados coletados armazenados no banco de dados da Foxglove ¹, bem como garantir o fluxo de dados através de *pipelines* de dados que permitam o tratamento, formatação e armazenamento eficiente em um banco de dados Postgres ² de forma estruturada, facilitando o acesso e a consulta para diferentes finalidades.
- Disponibilizar os dados para análise através de uma interface de acesso aos dados por meio da linguagem Python e relatórios em planilhas, permitindo a geração de relatórios automatizados e análises gráficas personalizadas.
- Executar o fluxo de dados de ponta a ponta com a finalidade de gerar um relatório para a prova de aceleração da competição, gerando métricas, análises gráficas e cálculos necessários para compreender o comportamento do carro durante esta prova, utilizando dados de simulação e teste do carro.

1.1.3 Estrutura do Trabalho

Este trabalho é composto por cinco capítulos. No capítulo 1 é apresentado uma visão geral do contexto em que o trabalho está inserido, suas motivações, bem como os objetivos geral e específicos do mesmo. O capítulo 2 aborda os temas teóricos que embasam este trabalho e são considerados conceitos fundamentais para a compreensão do mesmo. No capítulo 3 são apresentados os métodos e desenvolvimentos necessários para atingir o objetivo e construir a plataforma proposta pelo trabalho. Já os resultados obtidos por este trabalho são apresentados no capítulo 4, enquanto o capítulo 5 é o capítulo final que conclui o proposto neste trabalho, trazendo as considerações finais e visões de futuro.

¹ Foxglove é um *software* de gerenciamento de dados voltada para robótica e veículos autônomos cuja atuação é detalhada na seção 3.5.2.

² Postgres é um banco de dados, cuja teoria está exposta na seção 2.3.

2 Revisão Bibliográfica

Este capítulo apresenta a revisão bibliográfica, que constitui um levantamento crítico das principais produções acadêmicas relacionadas ao tema deste trabalho, com o objetivo de contextualizá-lo, identificar contribuições teóricas relevantes e apontar lacunas no conhecimento existente.

2.1 Sistemas Embarcados

Sistemas embarcados são conjuntos de *hardware* e *software* integrados que desempenham funções específicas dentro de um sistema maior. Em carros de corrida e competição, esses sistemas são essenciais para o controle, monitoramento e otimização do desempenho do veículo em tempo real ou em análises posteriores. A complexidade e a criticidade das funções desempenhadas por sistemas embarcados em veículos de alta performance demandam soluções altamente especializadas e robustas, capazes de operar sob condições extremas de temperatura, vibração e estresse mecânico (WALDO, 2005).

Os sistemas embarcados desempenham um papel fundamental no automobilismo moderno, especialmente na Fórmula 1, onde permitem a aquisição de dados em tempo real, o controle do veículo e a otimização de desempenho. Os carros da Fórmula 1 funcionam como redes de sensores móveis, equipados com centenas de sensores que monitoram parâmetros como temperatura do motor, pressão hidráulica, tensão na suspensão e forças aerodinâmicas. Uma unidade de controle eletrônico (ECU, do inglês *Electronic Control Unit*) a bordo processa esses dados para gerenciar o motor, o câmbio e outros sistemas críticos, enquanto transmite telemetria à equipe no box para ajustes estratégicos durante a corrida, além de salvar o histórico para análises posteriores. Esses sistemas embarcados operam em condições extremas suportando altas temperaturas, vibrações e transmissão rápida de dados sem comprometer a confiabilidade e a segurança. As informações coletadas não apenas auxiliam nas decisões durante a prova, mas também impulsionam a inovação contínua no projeto dos carros, tornando a computação embarcada indispensável para o avanço tecnológico do esporte (WALDO, 2005).

Segundo Lee e Seshia (2017), a principal função de um sistema embarcado é a sua capacidade de realizar controle e monitoramento em tempo real, o que é particularmente relevante em um ambiente de competição onde frações de segundo podem determinar o resultado de uma corrida. Em corridas de alto nível, como a Fórmula 1, sistemas embarcados avançados são empregados para controlar todos os aspectos do carro, desde a injeção de combustível até a gestão da energia dos sistemas híbridos, passando pelo controle de tração e estabilidade.

Uma das tecnologias chave nos sistemas embarcados para carros de competição é o

CAN Bus, um protocolo de comunicação robusto e de alta velocidade que permite a troca de informações entre os diversos módulos do carro. O uso do CAN Bus é amplamente difundido na indústria automobilística devido à sua capacidade de operar em ambientes ruidosos e de alta interferência, garantindo a integridade dos dados transmitidos (BOSCH, 2012). Além disso, a telemetria é outra tecnologia crucial, permitindo que os dados coletados pelos sensores embarcados sejam transmitidos em tempo real para a equipe técnica, que pode monitorar o desempenho do carro e tomar decisões estratégicas rapidamente.

Além do *hardware*, o *software* embarcado desempenha um papel igualmente importante. Sistemas operacionais em tempo real (RTOS) são frequentemente utilizados para garantir que todas as tarefas críticas sejam executadas com a prioridade adequada e dentro dos prazos necessários. O *software* de controle é geralmente implementado em linguagens de baixo nível, como C ou C++, para garantir a máxima eficiência e previsibilidade na execução (LEE; SESHIA, 2017).

2.1.1 Barramento CAN

Uma rede de área de controlador (CAN, do inglês *Controller Area Network*) é um padrão de barramento de veículos projetado para garantir a comunicação entre diferentes unidades de controle eletrônico e seu protocolo é usado em inúmeras aplicações devido à sua alta eficiência de comunicação e redução de complexidade. O protocolo segue o princípio de troca de mensagens, garantindo que cada unidade de controle consiga enviar suas mensagens sem que informações sejam perdidas (BOSCH, 2012).

Um automóvel moderno pode ter dezenas de unidades de controle, cada uma responsável por um subsistema. Essas unidades de controle são responsáveis por controlar inúmeros atuadores com base no *feedback* de sensores, muitas vezes há comunicação entre subsistemas e para organizar essas comunicações o padrão CAN foi criado. Uma das grandes vantagens do padrão CAN é a possibilidade de integrar os sistemas mecânicos e elétricos em um carro com um computador embarcado e controlá-lo através de *software*. No passado, antes do CAN, os sistemas elétricos dos carros se apoiavam apenas na eletricidade automotiva tradicional que era bastante limitada (VOSS, 2005).

O CAN é um barramento do tipo serial, isto é, ele transmite informações bit por bit, mantendo uma sequência no tempo. Geralmente, um barramento CAN consiste em um componente eletrônico que é posicionado embarcado no carro e é conectado aos sensores e unidades de controle, recebendo suas informações e salvando-as em algum ambiente de armazenamento, que pode ser um cartão de memória externo ou algum componente de memória embarcado no próprio produto. Diferente de outros barramentos paralelos e seriais utilizados no automobilismo, o barramento CAN permite o *download* dos dados através de um cabeamento USB, permitindo que vastas quantidades de dados sejam transmitidas para um computador dentro de segundos, tornando mais eficiente o tempo de disponibilidade dos dados. Em sistemas mais

recentes, é possível, inclusive, fazer o *download* destes dados através de *wireless*, permitindo que os dados sejam coletados sem cabeamento (VOSS, 2005).

2.1.2 Sensores

Um sensor é um dispositivo que retorna um sinal elétrico em reação a um fenômeno físico. Para cada medida imaginável, um sensor para essa medida provavelmente existe. Segundo Segers (2008), justificar a escolha e aplicação de um sensor específico passa por entender os requerimentos de cada mensuração, que devem ser previamente determinados através da resposta de o que estará sendo medido, além de compreender qual a expectativa de região das medidas para a aplicação, qual o ambiente ao qual o sensor será exposto, qual a precisão necessária e, por fim, qual o orçamento disponível.

De forma geral, todo sensor no automobilismo se resume a medição de alguma dessas variáveis fundamentais: temperatura, pressão, vazão, posição, velocidade, aceleração e força. Em vários desses tipos de sensores há uma dependência forte com as condições do ambiente em que o sensor está inserido, visto que condições como temperatura, pressão, vibração e até contaminação do ambiente podem influenciar no sinal obtido pelo sensor (SEGERS, 2008).

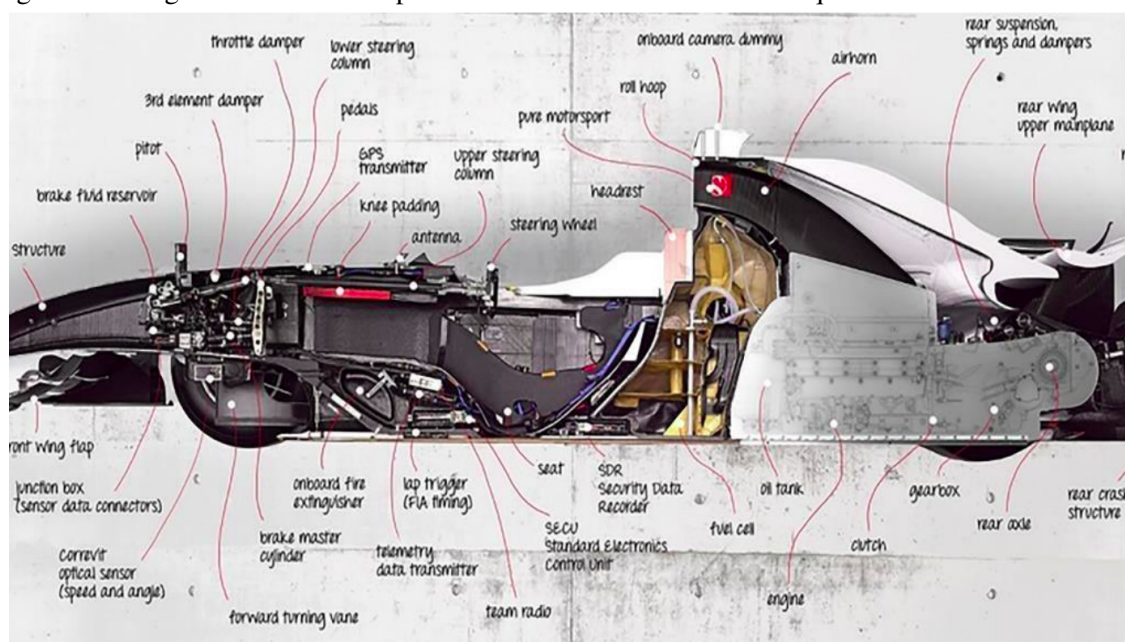
Todo sensor contém uma ficha técnica com as informações necessárias para responder às perguntas apresentadas anteriormente. Nesta ficha, geralmente existe uma curva representando o retorno elétrico versus a entrada física, isto é, como a tensão do sensor se comporta em função da variável física que o sensor está medindo. Com estas informações e os dados coletados através dos sensores, é possível converter os sinais elétricos na informação relevante (SEGERS, 2008).

Carros de Fórmula 1 possuem sensores que fornecem uma representação detalhada do carro e seu desempenho. São sensores de instrumentação (como de temperatura, pressão, vazão de combustível), monitoramento (para identificar a saúde do sistema do carro) e controle (que transformam comandos do piloto em atuações no carro). Os dados gerados por esses sensores são cruciais para a tomada de decisão durante corridas, por exemplo dados da temperatura no pneu monitoram o desempenho do mesmo, permitindo otimizar decisões de quando fazer uma parada no *pit stop*, alterando drasticamente estratégias e resultados das corridas (NIMMALA; NIMMALA, 2024). A Figura 1 lista vários sensores e o local onde são implementados em um carro de Fórmula 1.

2.2 Pipelines de Dados

Pipelines de dados são conjuntos de processos que movem e transformam dados de alguma fonte para algum destino, com o objetivo de permitir que esses dados sejam utilizados para alguma entrega de valor (DENSMORE, 2021). Segundo Pardalis (2021), os *pipelines* de dados são as artérias de qualquer arquitetura moderna de dados e são responsáveis por replicar o estado de um dado de um Sistema A para um Sistema B. No contexto de infraestrutura de

Figura 1 – Imagem ilustrando componentes de um carro de fórmula 1 que habilitam sensoriamento.



Fonte: Nimmala e Nimmala (2024)

dados, usualmente o Sistema A é um sistema que captura ou gera dados (como um aplicativo, uma aplicação em nuvem ou um banco de dados) e o Sistema B é um sistema focado em análise e manipulação de dados (como, por exemplo, um *Data Warehouse* ou *Data Lake*).

A origem dos *pipelines* de dados está relacionada a dificuldade que um banco de dados tem de suportar todas as diferentes cargas de processamento que os usuários, analistas e engenheiros precisam. Inicialmente, os *pipelines* de dados foram criados para integrar sistemas de banco de dados de tecnologia OLTP (Processamento Transacional *Online*, do inglês *Online Transaction Processing*) a sistemas de tecnologia OLAP (Processamento Analítico *Online*, do inglês *Online Analytical Processing*). Enquanto os bancos de dados OLTP focam em registrar transações de alguma aplicação, os bancos OLAP são focados em transformação e análise destes dados transacionais em dados analíticos, como tabelas resumidas, métricas, relatórios, entre outros (PARDALIS, 2021).

Sistemas OLAP, por si só, podem conter inúmeros *pipelines* de dados que, a depender da organização e arquitetura escolhida, são responsáveis por modelar e transformar os dados dentro do próprio banco de dados, levando de *schemas*¹ brutos para *schemas* refinados (CHAUDHURI; DAYAL, 1997). Além da modelagem dentro do próprio sistema OLAP, muitos *pipelines* também são responsáveis por levar de volta para os bancos transacionais e operacionais OLTP os dados que foram tratados no banco OLAP.

A complexidade de um *pipeline* de dados depende de características como tamanho, estado e estrutura dos dados, bem como das diferentes necessidades dos projetos analíticos.

¹ *Schemas* são uma unidade de organização dentro de um banco de dados que é composta por uma ou mais tabelas. Não confundir com os metadados que compõem a tabela, que também é usualmente chamado de *schema* da tabela.

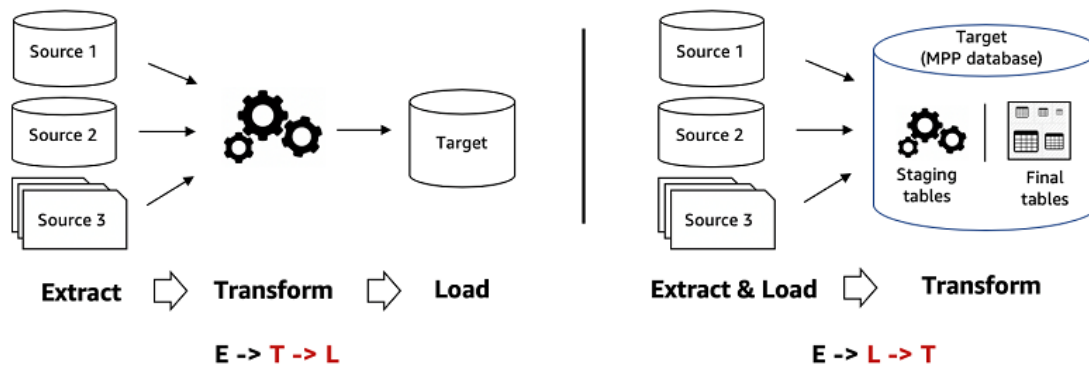
Na sua forma mais simples, *pipelines* consistem em extrair dados de uma fonte e carregar em um destino, porém, tipicamente, os *pipelines* consistem em inúmeras operações como extração, pré-processamento, validação, limpeza e modelagem. Com o advento da computação em nuvem e das plataformas de *Software* como serviço (*Software as a Service*, ou SaaS), somado à crescente demanda por dados para alimentar modelos de inteligência artificial ou aprendizado de máquina, modelagens analíticas e análises, surge o Engenheiro de Dados, um papel dentro de times de dados que é responsável por criar e manter *pipelines* (CHAUDHURI; DAYAL, 1997). Além disso, surge a necessidade de criar padrões de organização, geralmente chamados de arquitetura de *pipelines* (SEGNER, 2023).

Existem diferentes modelos de arquitetura de *pipelines* de dados, sendo os mais conhecidos as arquiteturas ETL e ELT. As letras em ambas as siglas representam os três processos fundamentais de um *pipeline* de dados: extrair (*Extract*), carregar (*Load*) e transformar (*Transform*) os dados (SEGNER, 2023). Arquiteturas do tipo ETL extraem os dados necessários e processam todas as transformações necessárias antes de carregar os dados no destino desejado. Os primeiros *pipelines* ETL surgiram na década de 70 com a criação do *Data Warehouse*, conceito de banco de dados com objetivo de centralizar dados de múltiplas fontes com a finalidade de analisar. *Pipelines* de ETL tiveram sua fase de maior importância no advento das tecnologias de nuvem e das tecnologias de *Big Data* a partir de 2010 (AMAZON WEB SERVICES, 2024c).

De acordo com a AMAZON WEB SERVICES (2024c), empresa pioneira nas tecnologias de nuvem modernas e uma das maiores representações dessa tecnologia atualmente, a evolução das tecnologias de nuvem e a redução nos custos de armazenamento e processamento de dados mudou o paradigma de como os *pipelines* podem ser construídos. O ETL, de forma geral, apresentam fluxos mais lentos, caros e difíceis de sustentar, pois dado a natureza do seu funcionamento, cada novo desenvolvimento significa a criação de novos *pipelines* e isso pode causar problemas de escalabilidade. A partir deste contexto, surge o ELT, no qual os *pipelines* iniciais são apenas responsáveis por carregar os dados e, ao não misturar essa etapa com a transformação como no ETL, há muito mais flexibilidade de transformar da forma que for necessário. Esse carregamento completo dos dados extraídos para que a transformação seja feita apenas depois só foi possível depois de todo o advento das tecnologias de nuvem (CHAUDHURI; DAYAL, 1997). Os diagramas na Figura 2 mostram as diferenças entre ETL e ELT de forma esquemática.

Pipelines do tipo ETL e ELT executam em lote (*batch*), ou seja, todos os dados extraídos em um período de tempo são carregados e transformados periodicamente através de algum agendamento. Em contraste aos *pipelines* de lote, existe o modelo de entrega contínua (*streaming*). Na Fórmula 1, por exemplo, esses tipos de *pipeline* são utilizados para sincronizar cerca de 1,1 milhões de pontos de dados por segundo. Através de tecnologias como o Apache Kafka, os membros da equipe têm acesso a alto volume de dados em tempo real, com escalabilidade e confiabilidade (NIMMALA; NIMMALA, 2024).

Figura 2 – Diagramas comparando processos de ETL e ELT.



Fonte: [AMAZON WEB SERVICES \(2024c\)](#)

Para garantir a confiabilidade dos dados, um *pipeline* robusto deve adotar práticas como validação de entradas, padronização de formatos e tratamento de exceções. Isso inclui a remoção de registros com valores nulos, a correção de unidades ou a aplicação de limites aceitáveis para cada variável. Além disso, a modularidade e o monitoramento contínuo são componentes essenciais de *pipelines* robustos. A modularidade garante que componentes individuais do *pipeline* possam ser mantidos ou atualizados de forma independente, reduzindo o risco de falhas em todo o sistema. O monitoramento contínuo ajuda a identificar gargalos, inconsistências ou erros de maneira antecipada, garantindo que os dados fluam de forma contínua e precisa.

Ao transformar dados brutos e eliminar anomalias, o *pipeline* cria uma base confiável para as análises, evitando erros e garantindo que as conclusões derivadas dos dados sejam precisas e relevantes. Dessa forma, o *pipeline* de dados não apenas aumenta o valor dos dados em si, mas também maximiza a eficiência e a confiança nas decisões baseadas nessas análises.

2.3 Banco de Dados

Um sistema de gestão de dados (DBMS) é o conjunto formado por uma coleção de dados que usualmente se relacionam entre si e uma série de funções e programas que permitem o acesso e interação aos dados. Essa coleção é comumente conhecida como banco de dados e ela contém informações relevantes sobre algum sistema, organização ou aplicação. A função do DBMS, portanto, é fornecer uma maneira de armazenar e consultar informações de maneira conveniente e eficiente. Os sistemas DBMS modernos são projetados para lidar com alto volume de dados e permitem a gestão dos mesmos através de estruturas de armazenamento, mecanismos de manipulação e compartilhamento seguro. ([SILBERSCHATZ; KORTH; SUDARSHAN, 2019](#)).

Os primeiros bancos de dados surgiram nos anos 60 a partir da necessidade de lidar com a gestão de dados comerciais. As primeiras aplicações eram relativamente simples quando comparadas com os sistemas modernos, porém toda aplicação de dados, antiga ou moderna, compartilha um elemento em comum: um sistema não é feito com a finalidade de gerar relatórios

ou fornecer cálculos, mas sim organizar os dados em si. Os dados se tornaram um bem valioso de qualquer organização e mantê-los de forma organizada é o grande objetivo destes sistemas. Uma instituição financeira precisa manter registrado todas as operações bancárias e informações sobre seus clientes, uma universidade precisa garantir integridade das informações sobre disciplinas, estudantes e matrículas, uma indústria precisa manter um inventário atualizado de todos materiais necessários e seu consumo, assim por diante. Perder essas informações é perder parte dos bens de uma organização e, portanto, mantê-las seguras é garantir o sucesso desta organização (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

A forma como as pessoas interagem com os bancos de dados evoluiu ao longo do tempo. Atualmente, existem dois modos principais com os quais as pessoas podem interagir com esses sistemas. O primeiro modo é para garantir o processamento transacional dos dados, no qual o banco de dados é responsável por manter as informações sobre um sistema e os usuários usam este banco para consultar ou atualizar informações. O segundo modo, por sua vez, visa análise dos dados, no qual os usuários consultam e processam as informações com o objetivo de chegar a conclusões analíticas, inferir regras ou tomar decisões, permitindo a evolução do negócio da organização (SILBERSCHATZ; KORTH; SUDARSHAN, 2019). Devido a diferença estrutural que um sistema precisa ter para garantir um modo ou o outro, diferentes tipos de sistemas foram construídos. Os sistemas de Processamento Transacional Online (OLTP) focam no processamento de dados transacionais, como operações bancárias, cadastros de clientes e produtos, informações comerciais como vendas e oportunidades, entre outros. Já os sistemas de Processamento Analítico Online (OLAP) possuem funcionalidades robustas para a análise de dados, focados em gerar relatórios, como por exemplo segmentação de campanhas, resultado de vendas, calcular taxas de conversão, entre outros (CHAUDHURI; DAYAL, 1997).

O principal modelo utilizado em bancos de dados (analíticos ou transacionais) é o modelo relacional. Um banco de dados que segue o modelo relacional é composto por tabelas formadas por colunas (campos ou atributos), cada qual com suas características como tipo de dados e domínio, e linhas, representando os múltiplos registros que uma tabela pode ter. O termo “relação” descreve, no modelo relacional, as tabelas, pois cada nova linha adicionada a uma tabela representa a relação entre múltiplas informações. Uma tabela de vendas por exemplo pode ser composta por linhas que representam uma venda, onde a venda de tal identificação ocorreu em uma data específica, foi vendida para um cliente específico e a venda foi operada por um colaborador específico. Todas essas informações são atributos da tabela e são representados pelas múltiplas colunas (SILBERSCHATZ; KORTH; SUDARSHAN, 2019). A Figura 3 mostra um exemplo de tabela em um banco de dados, neste caso especificamente uma tabela contendo dados de odometria do carro da Ampera Racing. A Figura 4, por sua vez, mostra os atributos desta tabela em um banco de dados PostgreSQL, com a visualização fornecida através da ferramenta DBeaver, que fornece uma interface intuitiva de interação com diferentes bancos de dados.

A linguagem mais comumente utilizada nos bancos de dados para que um usuário

Figura 3 – Exemplo de uma tabela mostrando dados do odômetro do carro da Ampera Racing repousados em um sistema de banco de dados PostgreSQL. A visualização gráfica obtida nesta imagem é fornecida através da ferramenta DBeaver.

	A-Z publish_time	123 position_x	123 position_y	123 position_z
1	2025-02-14T17:35:58.139083157	0,0002294922	0,0013671875	0,2438860983
2	2025-02-14T17:35:58.143329182	0,0002294922	0,0013671875	0,2438860983
3	2025-02-14T17:35:58.147222274	0,0002294922	0,0013671875	0,2438860983
4	2025-02-14T17:35:58.151211003	0,0002294922	0,0013671875	0,2438860983
5	2025-02-14T17:35:58.155972964	0,0002294922	0,0013671875	0,2438860983

Fonte: Autoria própria

Figura 4 – Exemplo dos atributos de uma tabela de dados de odometria do carro da Ampera Racing conforme demonstrado na ferramenta DBeaver.

	Column Name	#	Data type	Identity	Collation	Not Null	Default	Comment
Columns	A-Z publish_time	1	text		default	[]		
Constraints	123 position_x	2	float8			[]		
Foreign Keys	123 position_y	3	float8			[]		
Indexes	123 position_z	4	float8			[]		
Dependencies	123 orientation_x	5	float8			[]		
References	123 orientation_y	6	float8			[]		
Partitions	123 orientation_z	7	float8			[]		
Triggers	123 orientation_w	8	float8			[]		
Rules	123 twist_linear_x	9	float8			[]		
Policies	123 twist_linear_y	10	float8			[]		
Statistics	123 twist_linear_z	11	float8			[]		
Permissions	123 twist angular_x	12	float8			[]		
DDL	123 twist angular_y	13	float8			[]		
Virtual	123 twist angular_z	14	float8			[]		

Fonte: Autoria própria

interaja com os dados é o SQL, sigla para Linguagem Estruturada de Consulta (do inglês, *Structured Query Language*). Apesar do nome descrevê-la como uma linguagem de consulta, ela é capaz de fazer operações no banco como criação de recursos, modificar dados e especificar controles de segurança. Criada pela IBM no início dos anos 70 e originalmente chamada Sequel, a linguagem SQL evoluiu desde então e em 1986 recebeu sua primeira padronização no Instituto de Padrões Americanos (ANSI) e na Organização Internacional de Padronização (ISO), que foi atualizada em sua versão mais recente em 2016 (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

2.4 Tratamento de Sinais

Esta seção explica conceitos por trás de tratamento de sinais, presença de ruídos em sinais e aplicação de filtros.

2.4.1 Processamento de Sinais

Sinais são definidos como uma quantidade numérica que representa um fenômeno físico que varia com alguma variável independente, como tempo, distância, entre outros. Sinais são

gerados por algum sistema reagindo a um estímulo. Por exemplo, a voz humana é um sinal emitido pelas cordas vocais com a passagem de ar pelo trato vocal. Neste caso, o sistema é composto pelas cordas vocais e o estímulo é o ar que um ser humano precisa forçar através das cordas vocais para gerar voz (PROAKIS; MANOLAKIS, 1996).

Existem sistemas que recebem sinais como estímulo e executam alguma operação neste sinal, entregando outro sinal, por exemplo quando filtros são utilizados para reduzir ruídos. Quando um sinal é tratado através desses sistemas, essas operações são chamadas de processamento de sinais. Essa definição foi criada antes da era digital, voltado para sinais analógicos e sistemas físicos, porém o mesmo conceito pode ser replicado para o mundo digital, portanto todo sinal que é tratado, seja via *hardware* ou via *software*, é considerado um sinal processado (PROAKIS; MANOLAKIS, 1996).

No campo do processamento digital de sinais, os sinais são amplamente classificados em duas categorias fundamentais: sinais contínuos no tempo e sinais discretos no tempo. A distinção entre esses dois tipos é crítica, pois influencia as metodologias empregadas na aquisição, processamento e análise de sinais. Um sinal contínuo no tempo é definido para todos os valores reais da variável independente, tipicamente o tempo t (PROAKIS; MANOLAKIS, 1996). Matematicamente, tal sinal é expresso pela Equação:

$$x(t), t \in \mathbb{R} \quad (2.1)$$

Sendo que $x(t)$ pode assumir qualquer amplitude dentro de um intervalo contínuo e $\mathbb{R}$ é o conjunto de números reais. Um exemplo clássico de sinal contínuo é o sinal senoidal, que pode ser definido pela Equação:

$$x(t) = A \sin(2\pi Ft + \varphi) \quad (2.2)$$

Sendo:

A - amplitude do sinal

F - frequência em Hertz (Hz)

φ - deslocamento de fase

t - tempo

Este sinal é infinitamente diferenciável e existe para todo t , tornando-o um bloco fundamental na análise de sinais analógicos. Tanto o tempo quanto a amplitude são contínuos, permitindo uma representação exata de fenômenos físicos (e.g., ondas sonoras, variações de tensão). O processamento analógico requer sistemas contínuos, como filtros analógicos e integradores (PROAKIS; MANOLAKIS, 1996).

Em contraste aos sistemas contínuos, um sinal discreto no tempo é definido apenas em instantes discretos, geralmente uniformemente espaçados. Ele é representado pela Equação:

$$x(n), n \in \mathbb{Z} \quad (2.3)$$

Uma versão discreta da senoide é obtida por amostragem e demonstrado na Equação:

$$x(n) = A \sin(2\pi f n + \theta) \quad (2.4)$$

Sendo:

$\mathbb{Z}$ - conjunto dos números inteiros positivos

f - frequência adimensional normalizada $f = F/F_s$

F_s - frequência de amostragem dos dados (em amostras/segundo)

Diferentemente da versão contínua, $x(n)$ é indefinido entre os instantes de amostragem, pois a equação é definida apenas em múltiplos inteiros do período de amostragem T , que pode ser calculado como o inverso da frequência de amostragem, através da Equação:

$$T = \frac{1}{F_s} \quad (2.5)$$

Em implementações digitais, as amplitudes são frequentemente quantizadas para uma precisão finita. O processamento de sinais discretos é fundamental em processamentos digitais, visto que é passível de manipulação algorítmica em sistemas digitais.

Tanto em sinais discretos quanto em sinais contínuos, é fundamental compreender como os sinais se comportam ao longo das diferentes frequências que o compõem. Para este fim, utiliza-se o espectro de frequência como representação da distribuição da energia (potência) do sinal ao longo das frequências. Matematicamente, ele é obtido através de transformadas que convertem o sinal do domínio do tempo para o domínio da frequência, revelando características não evidentes na representação temporal.

Para um sinal contínuo no tempo $x(t)$, o espectro é dado pela Transformada de Fourier (TF), demonstrado pela Equação:

$$x(\omega) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi F t} dt \quad (2.6)$$

Para sinais discretos $x(n)$, utiliza-se a Transformada Discreta de Fourier (DFT), demonstrado pela Equação:

$$x(k) = \sum_{n=0}^{N-1} x(n) e^{-i2\pi \frac{k}{N} n}, \quad k = 0, 1, 2, \dots, N-1 \quad (2.7)$$

Sendo:

n - índice de tempo discreto

k - índice de frequência discreta, i.e os componentes de frequência

N - número total de amostras

Cada valor de k representa uma frequência discreta F em Hz, calculada pela Equação:

$$F = \frac{k}{N} F_s \quad (2.8)$$

Na Tabela 1 é mostrado um exemplo de como as frequências se relacionam umas com as outras com a variação de k .

Tabela 1 – Frequências normalizadas e em Hz para $F_s = 1000$ Hz com $N = 8$

k	f	$\frac{k}{N} F_s$ (Hz)
0	0	0
1	1/8	125
2	2/8	250
3	3/8	375
4	4/8	500
5	5/8	625
6	6/8	750
7	7/8	875

O espectro de frequência revela a composição harmônica e a distribuição de energia de um sinal ao longo das diferentes frequências que o compõem. Através da Transformada de Fourier, é possível identificar componentes periódicas (como senoides puras), representadas por picos estreitos em frequências específicas, e componentes não periódicas (como ruído), que aparecem como um espectro contínuo e espalhado. Além disso, o espectro permite determinar a largura de banda do sinal, mostrando em quais faixas de frequência a energia está concentrada (útil para aplicações como filtragem e compressão). Por exemplo, em um sinal de áudio, tons musicais aparecem como picos distintos, enquanto interferências ou distorções se manifestam como harmônicas adicionais ou energia fora da banda esperada.

O espectro também expõe artefatos de processamento e características físicas do sistema que gerou o sinal. Distorções não lineares (como as causadas por amplificadores) introduzem harmônicas em múltiplos inteiros da frequência fundamental, enquanto fenômenos como subamostragem (em inglês, *aliasing*) ou vazamento espectral (devido a janelamento inadequado) podem ser detectados como anomalias no domínio da frequência. Em aplicações práticas — como diagnóstico médico (EEG/ECG), telecomunicações ou análise de vibrações mecânicas —, a interpretação do espectro é essencial para isolar falhas, otimizar sistemas ou extrair informa-

ções relevantes, transformando dados brutos em *insights* acionáveis (PROAKIS; MANOLAKIS, 1996).

2.4.2 Ruídos em Sensores

Na grande maioria das aplicações práticas de obtenção de sinais através de sensores, há sinais que não podem ser descritos de forma acurada através de fórmulas matemáticas explícitas ou são complicados demais para um uso prático. Neste caso, esta falta de relacionamento implica em sinais que se comportam ao longo do tempo em uma maneira imprevisível. Esses sinais são considerados aleatórios. Muitas vezes, sinais não aleatórios que são obtidos através de sensores, que deveriam representar um fenômeno físico, se misturam com sinais aleatórios provenientes de outros fenômenos, como condições ambientais, vibrações, corrosões químicas e até mesmo mal funcionamento ou calibragem de um projeto de sensor. Estes fenômenos podem causar tais sinais aleatórios, afetando a acurácia da variável inferida (PROAKIS; MANOLAKIS, 1996).

O sinal discreto obtido por um sensor que apresenta a presença de um sinal ruidoso aditivo pode ser modelado através da Equação:

$$y(n) = x(n) + w(n) \quad (2.9)$$

Sendo:

$x(n)$ - sinal sem ruídos na sua forma discreta

n - índice da amostra

$w(n)$ - componente de ruído do sinal na sua forma discreta

Para avaliar a representação de um ruído no sinal e o quanto ele afeta a qualidade do sinal obtido, é comum calcular a razão de sinal para ruído (SNR , do inglês *Signal-to-Noise Ratio*) através da Equação:

$$SNR \text{ (dB)} = 10 \log_{10} \left(\frac{P_{\text{signal}}}{P_{\text{ruído}}} \right) \quad (2.10)$$

Sendo:

SNR - razão de sinal para ruído em decibéis

P_{signal} - potência do sinal real $x(n)$

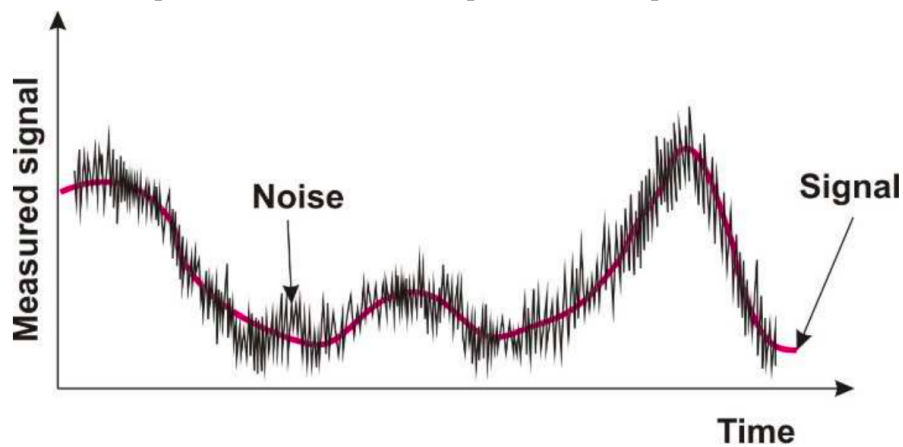
$P_{\text{ruído}}$ - potência do ruído $w(n)$

Quanto maior o valor de SNR , menor a influência do ruído no sinal original. Por outro lado, valores próximos de zero representam a presença considerável de ruído. A potência de um sinal ou ruído pode ser calculada através da Equação:

$$P_x = \frac{1}{N} \sum_{n=0}^{N-1} |x(n)|^2 \quad (2.11)$$

Uma parte importante do processamento de sinais é a separação dos sinais ruidosos dos sinais reais, pois qualquer aplicação deste sinal deve utilizar a representação real do fenômeno físico para garantir o resultado esperado. A Figura 5 mostra como um sinal pode ser afetado por ruído ao longo do tempo e os valores medidos diferem consideravelmente do valor real. Existem múltiplos tipos de ruído como por exemplo ruídos de impulso, harmônicos de alta ou baixa frequência e ruído branco (KELEMENOVA; BENEDIK; KOLARIKOVA, 2020).

Figura 5 – Exemplo de um sinal ruidoso, separando os componentes de sinal e ruído.



Fonte: Kelemenova, Benedik e Kolarikova (2020)

Para reduzir a presença de ruído em sinais, existem diferentes métodos aplicados. Um método muito utilizado no tratamento de sinais analógicos ou em sistemas elétricos é o método da Gaiola de Faraday, que consiste em enclausurar o sensor e o fenômeno físico em um volume fechado cuja superfície é feita de algum material condutor de eletricidade. O interior deste volume estará protegido contra efeitos externos de origem elétrica, eletrostática e eletromagnética. Outro método utilizado é através de blindagem, onde materiais isolantes são utilizados para proteger o sistema de efeitos externos, agindo como uma Gaiola de Faraday. Porém, o método mais utilizado de tratamento de ruído é através da filtragem, composta por circuitos eletrônicos (ou funções digitais, para o caso de sinais discretos) que habilitam a passagem de algumas frequências de sinais e bloqueiam outras frequências indesejadas (KELEMENOVA; BENEDIK; KOLARIKOVA, 2020).

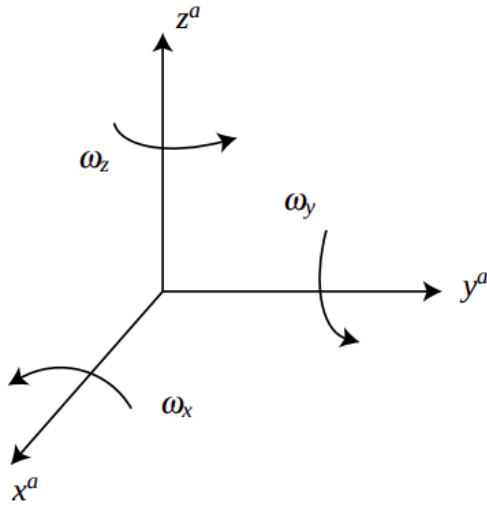
2.4.2.1 Modelagem de Sensores IMU

O IMU é um termo que denota uma Unidade de Medição Inercial (do inglês, *Inertial Measurement Unit*) que é um sensor capaz de inferir aceleração linear ou angular de um objeto. O desenvolvimento de sistemas microeletromecânicos permitiu a construção de acelerômetros e giroscópios menos custosos, tornando-os altamente adotados em muitas aplicações de sensoria-mento inercial. Sistemas como o Sistema Global de Navegação por Satélites (GNSS, do inglês

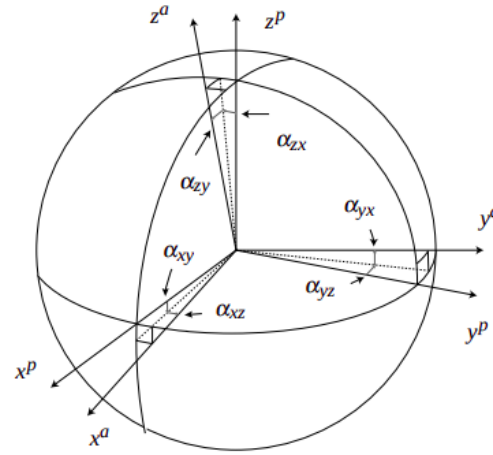
Global Navigation Satellite Systems) e o Sistema de Navegação Inercial (INS, do inglês *Inertial Navigation Systems*) foram possíveis através das evoluções tecnológicas dos sensores inerciais como o IMU (SKOG; HÄNDEL, 2017).

Conforme a modelagem de Skog e Händel (2017) de um sensor IMU, a representação geométrica do sensor é demonstrada através da Figura 6. Os eixos de sensibilidade do acelerômetro $\{x^a, y^a, z^a\}$ são montados para abranger um espaço tridimensional, e os giroscópios medem as velocidades angulares $\{\omega_x, \omega_y, \omega_z\}$ em torno desses eixos.

Figura 6 – Representação matemática das coordenadas sensibilizadas pelo IMU.



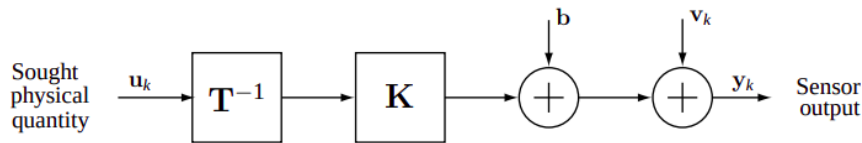
(a) Eixos de medição de um acelerômetro e um giroscópio.



(b) Plataforma de coordenadas de um acelerômetro.

Segundo a modelagem de Skog e Händel (2017), um sensor IMU pode ser descrito através do diagrama de blocos da Figura 7. A quantidade física medida é influenciada por desalinhamentos na calibração, representados por T^{-1} , pelo fator de escala K , o *bias* b e potenciais ruídos v_k .

Figura 7 – Modelo representando um sensor IMU.



Fonte: Skog e Händel (2017)

Yang, Bajenov e Shen (2017), por sua vez, recomenda uma metodologia para estudar as características dos sensores IMU através do Método de Desvio de Allan (ADEV, do inglês *Allan Deviation*), especificamente uma evolução do método chamado Desvio Sobreposto de Allan (OADEV, do inglês *Overlapped Allan Deviation*), que permitiu uma análise refinada para entender o comportamento do ruído e característica de estabilidade do sensor IMU. A metodologia proposta neste artigo oferece uma abordagem prática para estimar parâmetros de ruído em

sensores IMU, mesmo quando as especificações do fabricante não estão disponíveis. Ao coletar dados estáticos do sensor e analisar seus componentes de ruído, esse método gera características de ruído confiáveis que podem ser aplicadas a tarefas de navegação combinada GNSS/IMU. A abordagem é especialmente útil para sensores de baixo custo ou mal documentados, pois dispensa a necessidade de equipamentos laboratoriais complexos. Os valores de ruído obtidos, embora nem sempre ideais, servem como base para a integração dinâmica GNSS/IMU e podem ser ajustados para condições específicas, como vibrações externas ou frequências operacionais variáveis.

2.4.3 Filtro *Butterworth*

O filtro *Butterworth* é amplamente utilizado em aplicações de processamento de sinais devido à sua característica de resposta em frequência suave e monotonicamente decrescente, tanto na banda passante quanto na atenuação. Diferente de outros filtros, como o filtro de Chebyshev ou o filtro elíptico, o *Butterworth* busca maximizar a planicidade na banda passante, o que significa que ele não apresenta ondulações na resposta de ganho, proporcionando uma transição mais natural entre as frequências que devem ser preservadas e aquelas que devem ser atenuadas (OPPENHEIM; SCHAFER, 2010). Esse comportamento o torna particularmente adequado para aplicações em que a preservação da forma do sinal é crucial, como na filtragem de dados fisiológicos, sinais de áudio ou telemetria embarcada em veículos, onde distorções podem comprometer a análise.

Matematicamente, a função de transferência de um filtro *Butterworth* de ordem m é projetada para apresentar uma magnitude que decresce suavemente em função da frequência, sem oscilações, sendo definida pela Equação:

$$|H(j\omega)| = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2m}}} \quad (2.12)$$

Sendo:

ω - frequência angular do sinal de entrada

ω_c - frequência angular de corte

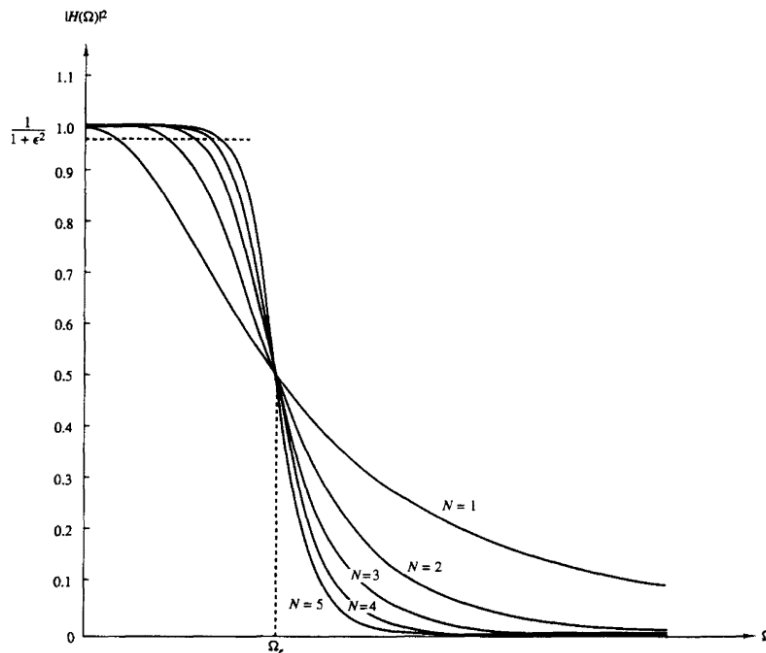
$|H(j\omega)|$ - módulo da função de transferência do filtro em frequência angular

m - ordem do filtro

A resposta em frequência de um filtro *Butterworth* pode ser visualizado na Figura 8, para diferentes valores de ordem. A suavidade da curva de resposta facilita a interpretação de fenômenos físicos representados no domínio da frequência, sendo uma escolha comum em projetos que priorizam estabilidade e previsibilidade no comportamento do sistema filtrado (PROAKIS; MANOLAKIS, 1996). Além disso, o filtro pode ser implementado tanto em sua

forma analógica quanto digital, com métodos como transformação bilinear e a aproximação de *Butterworth* digital permitindo sua utilização em sistemas embarcados e aplicações de engenharia moderna (SMITH, 1997).

Figura 8 – Resposta em frequência de um filtro *Butterworth* em função da ordem.



Fonte: Proakis e Manolakis (1996)

2.5 Escorregamento do Pneu

Esta seção definirá o material teórico utilizado para criar o relatório de aceleração do carro em relação a análise das forças longitudinais e do escorregamento dos pneus, definindo o equacionamento e conceitos levados em consideração.

Para que qualquer objeto possua aceleração ou desaceleração, a Segunda Lei de Newton demonstra que uma força na mesma direção do movimento deve ser implementada. Esta lei pode ser descrita pela Equação:

$$\vec{F} = m\vec{a} \quad (2.13)$$

No caso de um veículo, essa força é criada através da mecânica da interface entre o pneu e o chão. Essa força surge a partir de dois mecanismos complementares. O primeiro está relacionado à deformação do pneu na superfície de contato com o solo quando a roda está sob algum torque. A frente da superfície de contato se comprime, causando um encurtamento do perímetro do pneu. A parte comprimida gera atrito com o solo, criando a força que impulsiona o veículo. Na parte traseira da área de contato, o pneu volta ao seu formato original, mas como houve a deformação, ocorre um pequeno escorregamento entre o pneu e o solo (SEGERS, 2008).

O segundo mecanismo diz respeito à adesão intermolecular entre o pneu e o chão, que também colabora para que o carro se movimente (MILLIKEN; MILLIKEN, 1995).

Esse efeito de escorregamento implica que, em aceleração, a roda gire mais rápido do que giraria se estivesse rolando livremente. Já em desaceleração, a roda trava parcialmente, ou seja, gira menos rápido do que giraria se estivesse rolando livremente. O deslizamento pode ser medido como uma razão entre o quanto a roda está girando efetivamente através de sua velocidade tangencial, comparado com a velocidade com a qual o carro se movimenta. Essa taxa de escorregamento, popularmente conhecida como *Slip Ratio*, pode ser calculada para aceleração pela Equação:

$$\lambda = \left(\frac{v_t - v}{v} \right) \cdot 100\% \quad (2.14)$$

Sendo:

λ - taxa de escorregamento (*Slip Ratio*), em formato percentual

v - velocidade longitudinal do veículo

v_t - velocidade tangencial da roda

A velocidade tangencial da roda pode ser definida como:

$$v_t = \dot{\theta} \cdot r_{ef} \quad (2.15)$$

Sendo:

$\dot{\theta}$ - velocidade angular da roda

r_{ef} - raio efetivo, ou seja, a distância do centro da roda até o ponto de contato do pneu com o chão

Para fins práticos, é comum aproximar o raio efetivo r_{ef} como sendo o raio do pneu quando o carro está parado, influenciado apenas pelo peso do veículo. Apesar de gerar um erro nos cálculos, na maioria dos casos esse erro não é considerável (SEGERS, 2008).

Quando $\lambda = 0\%$, significa que o pneu está rodando livremente sem que haja transferência de esforços (caso ideal, por exemplo, para os pneus livres que não tracionam), já quando $\lambda > 100\%$ o pneu está "patinando" (por exemplo, os pneus traseiros de um carro fazendo *drift*). Para casos de travamento de rodas em frenagem, a velocidade tangencial da roda $v_t = 0$ e $\lambda = -100\%$.

Para carros de tração dianteira ou traseira, um par de pneus será de rolamento livre, isto é, eles não aplicam tração no chão e, por consequência, é possível considerar que a velocidade tangencial dele é igual a velocidade longitudinal do carro. Essa consideração, porém, requer como premissa que o carro esteja se movimentando em linha reta e sem apresentar forças laterais

(como força do vento). Neste caso, é possível calcular o deslizamento comparando a velocidade tangencial do pneu tracionado e a velocidade tangencial do pneu livre. A Equação a seguir adapta o cálculo de λ para esta condição para um carro de tração traseira:

$$\lambda_{rl} = \left(\frac{v_{rl} - v_{fl}}{v_{fl}} \right) \cdot 100\% \quad (2.16)$$

Neste caso, v_{rl} é a velocidade tangencial da roda traseira esquerda (a roda que traciona) e v_{fl} é a velocidade tangencial da roda dianteira esquerda (a roda que gira livremente). Neste caso, é possível calcular separadamente a razão de escorregamento para as rodas esquerda e direita. Essa separação pode ser útil para avaliar, por exemplo, o comportamento do diferencial (SEGERS, 2008). A Equação a seguir demonstra o cálculo do λ para a roda traseira direita.

$$\lambda_{rr} = \left(\frac{v_{rr} - v_{fl}}{v_{fl}} \right) \cdot 100\% \quad (2.17)$$

A razão de escorregamento longitudinal global pode ser calculada usando a média de ambas rodas, conforme demonstrado na Equação:

$$\lambda_{rr} = \left(\frac{(v_{rr} + v_{rl}) \frac{1}{2} - v_{fl}}{v_{fl}} \right) \cdot 100\% \quad (2.18)$$

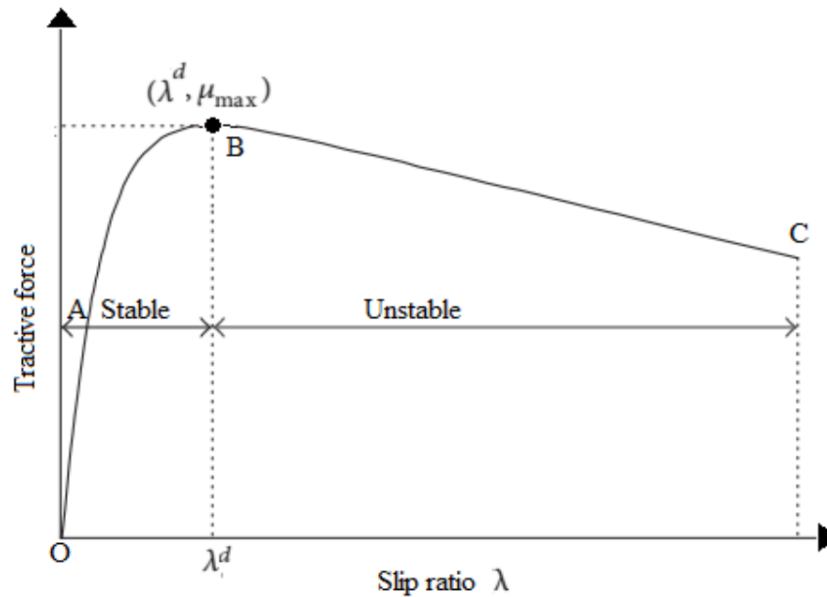
As equações apresentadas por Segers (2008) e demonstradas até o momento nesta seção produzem resultados ideais para a análise de carros em movimento retilíneo, por exemplo no caso de aceleração. Contudo, quando a velocidade do carro está próxima de zero (no início de acelerações, por exemplo), o denominador tende a zero e o valor da taxa de deslizamento tende a infinito. Para evitar essa singularidade e facilitar a análise dos resultados em casos de aceleração, é comum utilizar uma versão normalizada da Equação, conforme apresentado por Rajamani (2011):

$$\lambda_{norm} = \left(\frac{v_t - v}{v_t} \right) \cdot 100\% \quad (2.19)$$

Nesta versão, cujo denominador é alterado para que seja utilizada a velocidade tangencial do pneu, em contraste a velocidade longitudinal do carro, os valores de λ_{norm} possuem um domínio de $[0, 100\%]$ em casos de aceleração e tornam a análise facilitada.

Analisar a razão de escorregamento é crucial em carros de corrida porque ela determina a eficiência da transferência de potência entre os pneus e o asfalto, afetando diretamente a aceleração, a frenagem e a estabilidade nas curvas. Por exemplo, em uma largada, uma razão muito baixa pode limitar a tração, reduzindo a aceleração, enquanto um valor excessivo faz com que os pneus patinem, perdendo tempo e desgastando prematuramente os compostos. Este comportamento pode ser observado na Figura 9, visto que o ponto ótimo da curva, representando o ponto de máxima força longitudinal, se dá em um valor de razão de escorregamento maior que

Figura 9 – Força longitudinal em função da razão de escorregamento



Fonte: Rajendran et al. (2018)

zero. Sistemas como o Sistema de Controle de Tração (TCS, do inglês *Traction Control Systems*) e o Sistema Anti-travamento de Rodas (ABS, do inglês *Anti-lock Braking System*) utilizam-se do conceito de escorregamento para otimizar seus sistemas, permitindo acelerações e frenagens mais eficientes, que consomem menos pneu e permitem acelerar ou frear mais rapidamente. Portanto, otimizar esse parâmetro é essencial para melhorar o controle do carro em diferentes condições de pista (RAJAMANI, 2011).

A Tabela 2 mostra uma forma geral de interpretar o resultado da razão de escorregamento quando calculada pela Equação 2.18.

Tabela 2 – Interpretação do resultado de λ durante aceleração ou frenagem.

Sinal do λ	Fase	Significado
$\lambda > 0$	Durante aceleração	Pneu traseiro próximos de patinar
$\lambda > 0$	Durante frenagem	Pneu dianteiro próximo de travar
$\lambda < 0$	Durante aceleração	Pneu dianteiro próximo de travar
$\lambda < 0$	Durante frenagem	Pneu traseiro próximo de patinar

Fonte: Segers (2008)

2.6 Estado da arte

Com a finalidade de entender o cenário atual de aquisição e tratamento de dados nas competições de Fórmula SAE, esta seção sintetiza trabalhos acadêmicos, demonstrando o estado da arte no tema e fornecendo um comparativo importante com o que foi construído neste trabalho.

Ayyad e Fathizadeh (2017) descrevem a implementação de um sistema de telemetria e aquisição de dados para um veículo Fórmula SAE, utilizando um microcontrolador Arduino Mega e módulos de expansão compatíveis. O sistema coleta dados de diversos sensores, como deslocamento da suspensão, posição do acelerador e freio, ângulo de direção e parâmetros do motor, transmitidos por meio dos protocolos CAN BUS, I2C e SPI. A equipe priorizou custo acessível e confiabilidade, empregando componentes como um conversor *buck* para estabilidade de tensão, CAN BUS para comunicação com a unidade de controle eletrônico (ECU) e módulos XBee-Pro para transmissão sem fio dos dados a uma interface em LabVIEW. Por meio de calibração e processamento de sinais, alcançaram alta precisão, com erros tão baixos quanto 0,04% no deslocamento da suspensão. O sucesso do sistema demonstrou a importância de dados em tempo real para otimizar o desempenho do veículo, diagnosticar falhas e aprimorar o *feedback* ao piloto, mantendo-se dentro de restrições orçamentárias. O trabalho evidencia a viabilidade de desenvolver um sistema de telemetria eficiente em ambientes acadêmicos de competição automotiva.

Holland (2002), por sua vez, descreve o desenvolvimento de um sistema de aquisição de dados sem fio para um carro de Fórmula SAE, com o objetivo de facilitar e acelerar a análise de desempenho do motor. O projeto consistiu na construção de dois dispositivos: um *Data Logger* instalado no carro e uma ferramenta de monitoramento portátil, ambos baseados na placa CME11E9-EVBU com microprocessador 68HC11E9. O sistema foi projetado para captar os dados do controlador de injeção eletrônica e transmiti-los via protocolo RF para a ferramenta de monitoramento, que permite armazenar dados e interação com os dados. A escolha por componentes prontos, como a placa EVBU, visou reduzir a complexidade e focar no desenvolvimento de *software*, enquanto a decisão de usar a linguagem Assembly, e não C, buscou otimizar o uso de memória limitada. Um ponto de destaque do trabalho foi a preocupação em permitir o salvamento dos dados para posterior análise, o que, além de aumentar a eficiência do ajuste do motor em testes, evitava a necessidade de interromper a condução para baixar dados. O projeto evidenciou a importância de armazenar e recuperar dados históricos, já que as atualizações nas tabelas de injeção do motor poderiam ser realizadas de forma mais ágil e precisa.

Giraud (2019) descreve um processo abrangente de aquisição, tratamento e análise de dados de um carro de Fórmula SAE Elétrico da equipe *Squadra Corse do Politecnico di Torino*. O fluxo de dados inicia-se com a aquisição por uma rede CAN integrada a sensores como IMU, sensores de pedal, direção e motores, e armazenamento em um *Data Logger* Vector GL, complementado por um sistema de telemetria bidirecional para monitoramento e ajustes em tempo real. A partir dos dados brutos coletados como pressões de freio, ângulo de direção, velocidades de roda e acelerações, foi realizada uma limpeza e pós-processamento para separar fases de aceleração, frenagem e curvas, além de extrair variáveis relevantes como velocidades médias e desvios de trajetória. O tratamento e a análise dos dados foram conduzidos utilizando a ferramenta dSpace Control Desk para acompanhamento em tempo real e ferramentas como

Matlab e o próprio *software* do *data logger* para análises mais profundas dos dados armazenados. As escolhas metodológicas — como a implementação de um sistema de telemetria bidirecional e a centralização dos dados via CAN — foram motivadas pela necessidade de controle em tempo real e pela importância de registrar dados históricos para análises detalhadas pós-teste, visando otimizar o desempenho e assegurar a confiabilidade do veículo. Com estas definições, o time conseguiu identificar comportamentos dinâmicos críticos, otimizar estratégias de corrida e ajustar parâmetros eletrônicos e mecânicos com base em evidências objetivas extraídas dos dados.

De forma geral, percebe-se uma tendência em trabalhos acadêmicos de Fórmula SAE para desenvolvimentos próprios em contraste à contratação de soluções comerciais. Essa tendência se apresenta pelo fato de que equipes de Fórmula SAE possuem recursos limitados e construir essas soluções internamente auxilia o desenvolvimento dos membros da equipe.

3 Modelagem do Problema

3.1 Análise do Problema

Em 2023, a Ampera Racing competiu na Competição FSAE com o protótipo AMP-223 e um dos focos principais da equipe para este projeto foi a evolução de toda a aquisição de dados embarcada. Embora tal aquisição já tivesse sido utilizada em protótipos anteriores, ela frequentemente era negligenciada à medida que a competição se aproximava, uma vez que outros sistemas mais críticos precisavam de atenção. Em 2023, com um projeto de telemetria mais robusto, a equipe ainda estava limitada à utilização dos dados em tempo real ou à exportação para planilhas, tornando a análise dos dados uma tarefa árdua. Assim, ao fim da temporada de 2023, visando o desenvolvimento para 2024 e os anos seguintes, surgiu a ideia do DAS (*Data Analytics Software*). Esta plataforma foi concebida com o objetivo de permitir a análise detalhada dos dados coletados durante as corridas e testes, oferecendo *insights* que pudessem auxiliar na evolução do protótipo para a próxima temporada.

O principal desafio, então, consiste em garantir que o processamento dos dados ocorra de forma coesa com as definições de coleta de dados. Isso envolve organizar os dados de forma estruturada e cruzá-los com outras fontes de informação, como planilhas de *setup* do carro, características da pista e condições ambientais. A implementação do DAS visa criar uma plataforma robusta de coleta, armazenamento, organização e pós-processamento de dados, permitindo a criação de análises mais profundas sobre o comportamento do veículo e do desempenho em pista.

O desenvolvimento do DAS enfrenta desafios como a necessidade de criar uma plataforma sustentável a longo prazo, capaz de atender futuras competições, equipes e carros, dada a alta rotatividade de membros. Para isso, é essencial adotar princípios como generalização e documentação, garantindo que a plataforma seja flexível e bem orientada. Além disso, é crucial assegurar a qualidade e o processamento eficiente dos dados, desde a coleta até a disponibilização, evitando ruídos ou perdas, e permitindo que diferentes membros da equipe acessem e utilizem os dados de forma colaborativa, preferencialmente em ambientes na nuvem com versionamento de código.

A plataforma também deve ser escalável e ter alto desempenho para lidar com grandes volumes de dados, especialmente durante testes intensivos, mantendo a usabilidade e precisão. A flexibilidade na visualização é outro ponto crítico, permitindo gráficos e filtros personalizados para atender às diversas necessidades de análise, como monitoramento de velocidade e acelerações do carro, e movimento da suspensão. A usabilidade é igualmente importante, exigindo uma interface intuitiva e acessível mesmo para usuários sem expertise técnica, facilitando a

interpretação rápida de dados em situações de alta pressão, como competições.

Por fim, o orçamento limitado, comum em equipes universitárias que dependem de patrocínios e apoio de laboratórios, impõe restrições às ferramentas e equipamentos disponíveis. O grande desafio do trabalho é, portanto, desenvolver uma plataforma que atenda a todos esses requisitos — sustentabilidade, processamento eficiente, acessibilidade, escalabilidade, flexibilidade e usabilidade — dentro das limitações financeiras.

3.2 Requisitos Funcionais

O DAS surge no contexto da Ampera Racing com dois principais objetivos para a equipe. O primeiro é a capacidade de analisar dados do carro de forma eficiente após os testes em simulador ou testes reais. O segundo objetivo é garantir o amadurecimento da Área de Dados da equipe, permitindo evolução da mesma nos anos seguintes.

Para sincronizar o objetivo deste trabalho de conclusão de curso com os objetivos da Área de Dados da Ampera Racing, uma série de requisitos funcionais foram coletados com os membros da equipe. Esses requisitos funcionais baseiam o desenvolvimento da plataforma e delimitam o escopo de implementação. Os requisitos funcionais foram organizados na lista abaixo.

- Toda solução técnica desenvolvida para o DAS deve ser coerente com o orçamento delimitado para o desenvolvimento de dados na Ampera Racing.
- O desenvolvimento deve ser feito em Python, linguagem comumente conhecida por membros da equipe, permitindo a capacitação técnica da equipe no futuro.
- Existem membros da equipe que não são familiarizados com programação, logo o DAS deve criar e disponibilizar relatórios facilmente, permitindo a democratização dos dados na equipe.
- Os relatórios não precisam ser gerados em tempo real, porém em tempo suficiente para que sejam analisados pós testes e as ações obtidas pela análise sejam implementadas em tempo suficiente dentro do período de testes.
- Todos os componentes, processos e ferramentas utilizadas no processo do DAS devem ser devidamente documentadas, garantindo a sustentação da plataforma pela própria equipe no futuro.
- As visualizações presentes nos relatórios devem ser validadas com as áreas técnicas relacionadas à análise, garantindo a confiabilidade dos dados.

3.3 Proposta de Solução

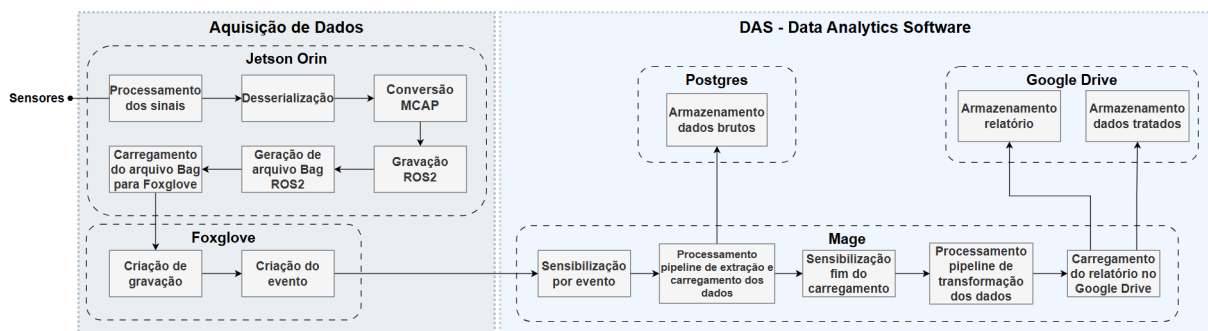
Com base na análise do problema e os requisitos funcionais levantados, esta seção visa apresentar a solução proposta que atende este cenário e realiza o objetivo principal e objetivos específicos da proposta deste trabalho. A solução proposta é apresentada nesta seção de forma abrangente, apresentando uma visão global de como os componentes e ferramentas se correlacionam e as decisões tomadas que levaram a esta solução. O detalhamento da arquitetura e da modelagem de cada componente é mostrado a partir da seção 3.4.

Para descrever a solução proposta, optou-se por demonstrar a arquitetura através de diagramas de blocos representando os componentes eletrônicos de *hardware*, componentes de *software* e, por fim, o fluxo de dados e os componentes envolvidos.

3.3.1 Arquitetura da Solução

O diagrama de blocos da Figura 10 demonstra uma representação gráfica de como os processos de *software* ocorrem desde a aquisição de dados até o tratamento. Esta representação gráfica é comumente chamada de Arquitetura de Solução no desenvolvimento de *softwares*. Os blocos menores de linha contínua são os processos, enquanto os blocos tracejados são os componentes (equipamento ou *software*) e os blocos coloridos de linha pontilhada são os grandes agrupamentos funcionais de tratamento de dados.

Figura 10 – Diagrama de blocos representando a interação entre os componentes de *software* e os processos executados.



Fonte: Autoria própria.

O tratamento inicial acontece no sistema de Aquisição de Dados que, por sua vez, ocorre fundamentalmente no computador embarcado, representado pela Jetson Orin, e o banco de dados de eventos robóticos utilizado pela Ampera, a Foxglove. Estes componentes não fazem parte exclusivamente do DAS, visto que são utilizados também para outros sistemas de tratamento de dados, voltados para o caráter autônomo do carro, utilizados nas operações com o carro sem piloto, isto é, no modo *driverless*. Controlar o carro remotamente é parte da competição de FSAE elétrico e existem operações de dados necessárias para este controle, como a utilização de câmeras para identificação de obstáculos na pista através de visão computacional, inferir posição e velocidade do carro para controle do motor elétrico, entre outras operações.

O bloco pontilhado em azul é, portanto, exclusivo ao DAS e todos os componentes que estão representados nele são, por consequência, o DAS. O primeiro componente da solução é o Mage, um orquestrador de processos disponibilizado através da linguagem Python que nesta solução possui o papel de organizar as etapas de tratamento de dados necessárias para cumprir os objetivos da plataforma. A escolha do Mage como orquestrador dos processos atende aos requisitos funcionais por se tratar de uma ferramenta Python, bem como seu caráter de *software* de código aberto que não possui custos de utilização, tornando seu uso compatível com a realidade de orçamento da equipe. Dentro do Mage, os processos executados são todos processados através da linguagem Python, o que permite o uso de qualquer biblioteca do ecossistema do Python nas operações.

Os processos dentro do Mage são organizados através de *pipelines*. O Mage permite a criação de sensores digitais que executam os *pipelines* com base em algum evento. O evento escolhido para esta solução é, coincidentemente, os eventos criados na plataforma da Foxglove. Logo, todo evento novo criado na plataforma da Foxglove aciona o *pipeline* de extração e carregamento dos dados, que tem o papel de coletar os dados da Foxglove e armazená-los em formato tabular na camada de armazenamento de dados brutos, o banco de dados PostgreSQL. Por exemplo, quando efetuado um teste de aceleração com o carro, os dados são adquiridos e salvados na Foxglove e um novo evento é criado identificando o tipo de teste e as datas e horas de início e fim do evento. O Mage coleta estes dados do teste e gera o relatório referente à prova de aceleração, com os dados, análises e visualizações necessárias para esta prova.

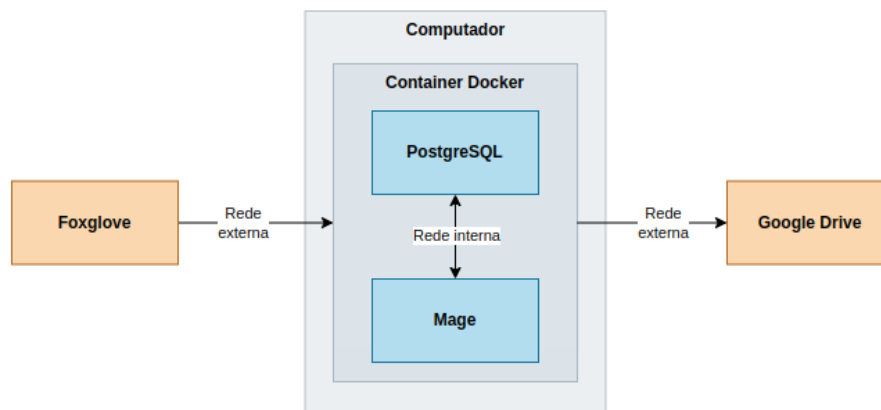
O PostgreSQL é o segundo componente da solução do DAS. O PostgreSQL é um banco de dados de código aberto relacional e orientado a objetos (ORDMBS) ([AMAZON WEB SERVICES, 2024b](#)). Assim como o Mage, atende aos requisitos funcionais pelo seu extenso ecossistema, por ser de código aberto (por consequência, gratuito) e por integrar facilmente com o Python. Além dessas vantagens, o PostgreSQL é uma solução de mercado muito utilizada pela sua flexibilidade. Apesar de ser um banco transacional (OLTP) por definição, ele possui funcionalidades analíticas que permitem usá-lo como um banco analítico (OLAP) também, auxiliando no desafio de escalabilidade da plataforma, pois ele é capaz de não só armazenar um alto volume de dados com baixa necessidade de memória, mas também é capaz de efetuar operações analíticas com baixo impacto em processamento.

Apesar de que a utilização do PostgreSQL e do Mage são gratuitas devido ao seu caráter de serem de código aberto, ambos *softwares* devem ser provisionados em um computador. As tarefas que ocorrem em ambos *softwares* utilizam processamento e armazenamento deste computador. Com a finalidade de utilizar os recursos disponíveis pela Ampera e atender ao requisito de efetuar o DAS sem onerar o orçamento de 2025, optou-se por utilizar um computador dedicado na estrutura da Fundação CERTI, uma fundação da Universidade Federal de Santa Catarina que apoia o desenvolvimento da tecnologia e apoia a Ampera Racing em suas competições. A outra alternativa seria utilizar computação em nuvem através de uma plataforma como a Amazon Web

Services (AWS) ou a Google Cloud Platform (GCP), porém o alto custo para provisionamento de máquinas virtuais destas plataformas levaram a decisão pelo uso do computador da CERTI.

A Figura 11 demonstra a containerização dos *softwares* Mage e PostgreSQL provisionadas em um computador. Containerização é um processo tradicional na implementação de *softwares* que agrupa todo código necessário para ser executado e as dependências necessárias (bibliotecas, compiladores, etc) nas versões corretas, para evitar problemas de incompatibilidade do *software* quando executado em diferentes computadores, permitindo portabilidade do *software*. A ferramenta utilizada para a containerização do código do DAS é o Docker, ferramenta vastamente utilizada em computação para efetuar a containerização. O Docker é um ambiente de execução de contêineres de código aberto que permite aos desenvolvedores de *software* construir, implantar e testar aplicações em contêiner (AMAZON WEB SERVICES, 2024a).

Figura 11 – Diagrama de blocos demonstrando a containerização do *software* em um contêiner Docker, executando dentro de um computador e suas comunicações internas ou externas.



Fonte: Autoria própria.

Segundo RADIANT LOGIC (2025), as configurações mínimas de um computador necessárias para provisionar um banco de dados PostgreSQL são de 16 GB à 32 GB de memória, armazenamento conforme necessário, porém preferencialmente do tipo disco de estado sólido (SSD), e pelo menos 14 núcleos de processamento na CPU. O computador utilizado possui 16 GB de memória e 8 núcleos de processamento na CPU e um disco 512 GBs de memória, que está um pouco abaixo das recomendações, podendo gerar problemas de desempenho.

Por fim, o terceiro e último componente do DAS é o Google Drive, uma ferramenta do Google para armazenamento de arquivos variados. Foram avaliadas múltiplas ferramentas e plataformas para fazer parte da camada de disponibilização do DAS como Streamlit, Power BI, Google Data Studio, entre outros. A escolha do Google Drive como camada de disponibilização dos dados foi escolhida pela facilidade de acesso do time aos relatórios gerados. O Google Drive já é amplamente utilizado na Ampera e todos possuem acesso à plataforma, permitindo que qualquer membro da equipe possa fazer uso dos resultados disponibilizados.

Além do aspecto de facilidade do uso, o Google Drive é flexível em relação ao tipo de

arquivo carregado a sua plataforma, permitindo gerar relatórios em formato de valores separados por vírgula (CSV), formato este que pode ser facilmente integrado na plataforma de escolha do analista para visualizar e analisar os dados. Além do CSV, o Google Drive permite o carregamento de imagens em formato JPEG, formato este utilizado pelo pacote Matplotlib no salvamento de gráficos, utilizado no processamento e geração de relatórios do DAS. Logo, a escolha do Google Drive atende ao requisito funcional de democratização dos dados.

3.4 Hardware

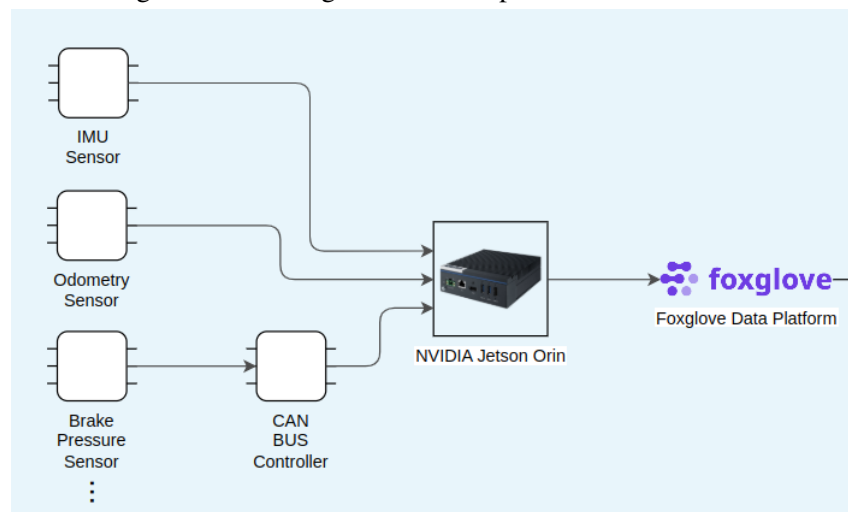
Nesta seção, são demonstrados todos os componentes de *hardware* utilizados no projeto do DAS, desde a aquisição dos dados até o processamento destes através da transformação e geração do relatório.

3.4.1 Aquisição de Dados

O sistema de coleta de dados embarcado no protótipo FSAE Elétrico da Ampera Racing é composto por um conjunto de componentes de *hardware* selecionados para atender às demandas técnicas da competição. Os componentes eletrônicos que fazem parte da arquitetura do DAS são os sensores, um barramento CAN (*Controller Area Network*) para comunicação de dados e um computador embarcado NVIDIA Jetson Orin, que atua como o núcleo de processamento do sistema, recebendo os dados dos sensores através de sua conexão com o CAN. Como muitas outras escolhas em uma equipe de competição, a seleção desses componentes é uma combinação de oportunidades que as parcerias proporcionam, com a finalidade de reduzir o custo do projeto como um todo, além, é claro, de se basear na garantia do atendimento aos requisitos técnicos mínimos para fazer o processo funcionar de ponta a ponta.

O fluxograma da Figura 12 demonstra os componentes de *hardware* que compõem a arquitetura do DAS e mostra como o fluxo de dados ocorre desde sua tomada nos sensores até sua publicação na plataforma da FoxGlove. Os sensores são responsáveis por capturar sinais através de um comportamento físico do veículo, como os sensores IMU (Unidade de Medida Inercial) que permitem a coleta de dados de orientação e rotação do carro.

Existem múltiplos sensores instalados no carro e eles são detalhados na seção 3.4.1.1. O barramento CAN (Rede de Área de Controlador), por sua vez, permite a comunicação eficiente e confiável entre estes sensores e o computador embarcado, garantindo que os dados sejam transmitidos com baixa latência e alta integridade. Porém, não são todos os sensores que precisam ser integrados via CAN, pois alguns sensores podem ser integrados diretamente no computador embarcado, a NVIDIA Jetson Orin. Por fim, este componente possui um poder de processamento avançado e capacidade de operar em ambientes de alta demanda, servindo como a unidade central de aquisição e pré-processamento dos dados, preparando-os para armazenamento. Na

Figura 12 – Fluxograma dos componentes de *hardware*

Fonte: Autoria própria

Jetson Orin, os dados são desserializados, isto é, convertidos do seu formato de bits para unidades compreensíveis como aceleração, velocidade, etc.

Essa combinação de *hardware* não apenas atende às necessidades atuais da equipe, mas também oferece escalabilidade para futuras expansões, como a inclusão de novos sensores integrados ou não à CAN. A seguir, é detalhado cada um desses componentes, explicando sua função, características técnicas e integração no sistema.

3.4.1.1 Sensores Utilizados

As informações utilizadas no relatório de aceleração apresentado neste trabalho são provenientes de três principais sensores: um Odômetro, um sensor de Unidade de Medida Inercial (IMU, do inglês *Inertial Measurement Unit*) e quatro Sensores de Rotação. O Odômetro é integrado à câmera posicionada acima do banco do motorista, utilizada pela área de *Driverless* para as finalidades de dirigibilidade autônoma. O IMU é um equipamento separado e instalado no chassi do carro, posicionado o mais próximo do centro de gravidade possível. Já os Sensores de Rotação são instalados nas quatro rodas do carro. Tanto o veículo do simulador quanto o veículo real possuem essas três fontes de informações, provendo os dados de forma semelhante.

3.4.2 Jetson Orin

O carro da Fórmula SAE da Ampera Racing apresenta inúmeros componentes eletrônicos que conversam entre si, sejam eles sensores, atuadores, o próprio motor elétrico, painéis de informação, entre outros. Para efetuar essa comunicação entre esses sistemas, existe a necessidade de um microprocessador ou computador que orquestre essas operações com eficiência. Para o ano de 2025, a Ampera terá embarcada em seu carro uma NVIDIA Jetson Orin, um potente computador embarcado muito utilizado em aplicações de IA generativa, visão computacional e

robótica avançada.

A Ampera Racing possui um braço da equipe responsável pelo *Driverless*, onde modelos de visão computacional são desenvolvidos para permitir que o carro seja guiado de forma autônoma, isto é, sem piloto. Tal tecnologia requer alto poder de processamento e a Jetson Orin é produzida pela NVIDIA com este objetivo. Este equipamento é feito para ser embarcado em algum sistema e possui todos os componentes que se espera em um computador de alto desempenho: CPU, GPU, memória, armazenamento e até conexão com a *internet*. Sua capacidade de operação pode chegar a até 275 trilhões de operações por segundo (KARUMBUNATHAN, 2022).

É evidente que para a finalidade deste trabalho e para as operações contidas na arquitetura proposta, não haveria necessidade de um equipamento com tamanho desempenho, porém dado a sua necessidade em *Driverless* e o fato de que esse equipamento já está disponibilizado para usar no carro, este trabalho aproveita a presença deste equipamento como parte do *hardware* de aquisição de dados. Os sensores, sejam conectados ao CAN ou não, são integrados à Jetson Orin, que, por sua vez, foi programada para executar os *scripts* necessários de desserialização dos dados, captura dos dados em formato ROSBAG2, empacotamento no formato de mensagens do MCAP e, por fim, carregamento dos dados na plataforma da FoxGlove através da *internet*.

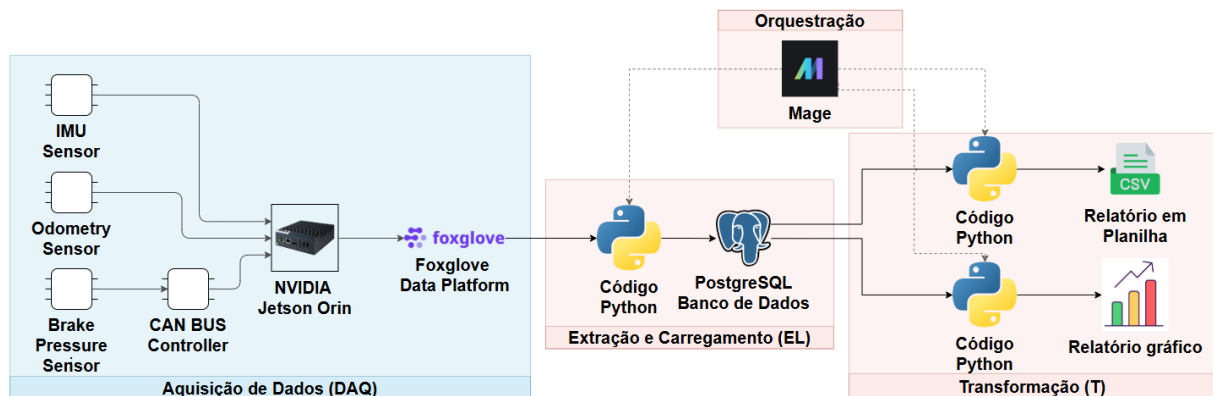
3.5 Software

Na seção 3.4, todos os componentes de *hardware* foram detalhados, fornecendo uma visão de como os dados são coletados pelos sensores e qual a função de cada componente na trajetória do dado desde sua aquisição até a geração dos relatórios. Nesta seção, portanto, são explicitados todos os *softwares* utilizados no caminho dos dados desde seu armazenamento em estado bruto na plataforma da Foxglove, os tratamentos iniciais em Python e o protocolo de comunicação que é utilizado nas etapas de *software*, representado pelo formato MCAP e transmissão de dados. Os *softwares* utilizados pelo sistema de aquisição de dados não são cobertos por este trabalho, visto que o objetivo do mesmo é demonstrar uma plataforma que extraia os dados brutos já coletados. A Figura 13 mostra o fluxograma de ponta a ponta dos processos de *software* construídos neste trabalho e que serão detalhados nesta seção.

3.5.1 Armazenamento dos Dados

Conforme explicado na seção 3.4.1, os dados coletados pelo barramento CAN são desserializados, isto é, convertidos de bits para as grandezas físicas através de uma série de códigos C. Estes dados são gravados pelo sistema ROS 2 e salvos no formato de tópicos MCAP, posteriormente carregados na plataforma da Foxglove em nuvem através de sua Interface de Programação de Aplicativos (API, do inglês *Application Programming Interface*) da plataforma da Foxglove para fazer carregamento de arquivos. Desta forma, toda vez que o carro é ligado, os

Figura 13 – Fluxograma total mostrando as etapas de Aquisição de Dados, Extração e Carregamento e Transformação



Fonte: Autoria própria

sensores começam a adquirir os sinais e são armazenados temporariamente na Jetson Orin. Ao fim da sessão, estes dados são automaticamente guardados na plataforma em nuvem, de forma que outros sistemas na *internet* consigam acessar esses dados. A Figura 14 mostra como estes dados ficam armazenados em formato de tópicos na plataforma da Foxglove e mostra também alguns gráficos simples, possíveis de serem feitos diretamente na plataforma.

Figura 14 – Exemplo de visualização da tela de mensagens do tópico na Foxglove durante prova de *Autocross* no simulador, mostrando gráficos de velocidade longitudinal do carro e velocidade angular das quatro rodas.



Fonte: Autoria própria

Os dados são capturados através de um sistema de captura ROS 2 (a segunda versão do Sistema de Operações Robóticas, ou *Robot Operating Systems*), um sistema responsável por organizar, convencionar e operar mensagens de um sistema complexo de robótica. Os dados do ROS2 são armazenados utilizando o formato MCAP, que é um formato de código aberto para

salvar dados de *log* multimodais provenientes de diferentes fontes, como por exemplo ROS 2. O MCAP permite salvar múltiplos canais de dados pré-serializados ou serializados em função de um ponteiro de tempo (*timestamp*) identificando quando cada dado foi coletado corretamente. Além disso, o MCAP é ideal para integrar em sistemas de mensageria de publicação e subscrição¹ e é vastamente utilizado em aplicações de robótica. Um componente muito importante da competição FSAE é a parte de *Driverless*. Dado esse contexto, existem muitas aplicações dentro da equipe que não necessariamente se relacionam com o DAS mas que se beneficiam também da presença destes dados em formato de mensagens multimodal, logo o MCAP é fundamental para garantir o funcionamento dos algoritmos do *Driverless*.

Para o DAS, o maior benefício do uso do MCAP é a capacidade de organização dos dados. Através do MCAP, é garantido que os dados sempre chegam com o mesmo formato e com os mesmos nomes e referências. Portanto, esta capacidade do MCAP facilita a gestão e organização dos códigos de tratamento de dados que ocorrem no DAS, auxiliando o atendimento de um dos componentes do objetivo deste trabalho, que é a capacidade de organizar os dados.

Porém, apesar do MCAP ser uma ótima ferramenta para organizar a entrada de dados na plataforma, como é possível perceber pela Figura 15, este formato não é muito flexível na capacidade de análise de dados, pois operações analíticas se tornam difíceis dado ao seu caráter pouco humanizado de visualização dos dados, visto que o MCAP é feito para integração de sistemas. Logo, fazer operações matemáticas, lógicas e algoritmos utilizando puramente o MCAP se torna inviável. Para deixar os dados em um formato mais fácil de manipular, os dados em MCAP são convertidos para um Pandas DataFrame por meio da linguagem de programação Python, conforme a Figura 16. Os dados são coletados da Foxglove através de sua API, permitindo que as mensagens sejam coletadas e convertidas para o formato Pandas. Além de converter os dados em Pandas, esses dados também são carregados para um banco de dados transacional (Postgres), que permite o armazenamento de múltiplas tabelas de forma organizada, com restrições de tipos de dados, chaves únicas e capacidade de utilizar o SQL para fazer análises rápidas dos dados.

Para armazenar os dados em Postgres é utilizado a API de comunicação com o banco de dados da SQL Alchemy, que é um pacote em Python que permite conexões com múltiplos bancos de dados. Esta API é facilmente utilizável através do conector nativo do Mage. Através dessa comunicação, o Mage utiliza um motor de conversão que transfere os dados do Pandas para o banco de dados. Todas essas transferências são feitas em formato incremental, isto é, os dados históricos são mantidos no banco de dados e toda nova inclusão de dados é incrementada nas tabelas pré-definidas no banco.

Para que o banco de dados seja acessado por diferentes pessoas, é necessário carregá-lo

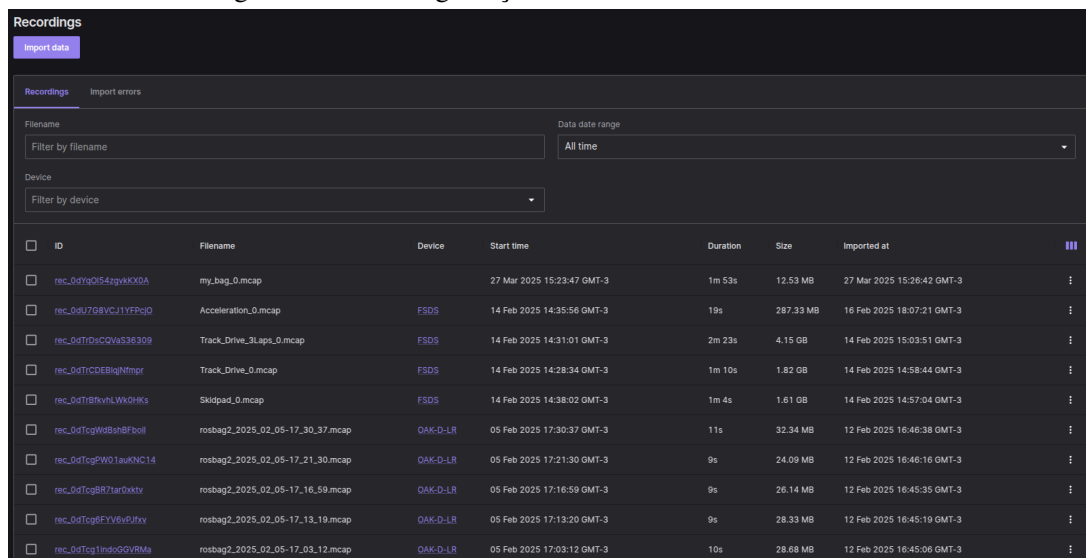
¹ Sistemas de mensageria como o Apache Kafka e Google Pub/Sub conectam aplicações, permitindo que troquem dados de forma assíncrona, desacoplando-os para maior flexibilidade e escala. Em termos simples, eles agem como intermediários que permitem que uma aplicação envie mensagens (ou eventos) para outra sem precisar saber diretamente quem vai receber ou como.

Além da interface gráfica, que permite interação manual, a Foxglove também conta com uma API que permite a interação automatizada com o *software*, de tal modo que pode-se carregar os dados de forma massiva para o DAS ou coletar dados de múltiplas sessões de uma só vez dentro do *pipeline* de dados.

A plataforma da Foxglove possui alguns conceitos importantes que foram adaptados para o contexto da Ampera Racing. O primeiro conceito é o de dispositivo, que podem ser dispositivos físicos ou lógicos que geram dados, como sensores, controladores ou sistemas de comunicação. Cada dispositivo é identificado de forma única no sistema, permitindo a organização e a filtragem dos dados com base em sua origem. Os dispositivos registram gravações que são os arquivos que armazenam dados coletados durante um período específico, como testes em pista ou simulações. Esses arquivos contêm informações provenientes do dispositivo em questão, organizados em um formato estruturado (como MCAP) para facilitar a análise posterior.

Por fim, as gravações são organizadas em eventos, que representam ocorrências específicas durante a operação do veículo, como falhas detectadas, mudanças de estado ou ações do piloto. Os eventos são marcados com uma data e hora de início e fim e podem ser usados para identificar padrões ou problemas no desempenho do veículo, mas também para identificar aplicações específicas e tipos de carga que os dispositivos estão registrando. Na figura 17, é possível identificar diferentes gravações enviadas à Foxglove e visualizadas diretamente na interface gráfica da plataforma.

Figura 17 – Tela da Foxglove mostrando gravações de ROS2 do carro sendo armazenadas na Foxglove.



The screenshot shows the 'Recordings' tab in the Foxglove application. It features a search bar for 'Filter by filename' and a dropdown for 'Data date range' set to 'All time'. Below these are filters for 'Device' and 'Filter by device'. The main area displays a table of recordings with columns for ID, Filename, Device, Start time, Duration, Size, and Imported at. Each row has a checkbox on the left and a menu icon on the right.

ID	Filename	Device	Start time	Duration	Size	Imported at
☐ rec_0d7yQ54zgvkX0A	my_bag_0.mcap		27 Mar 2025 15:23:47 GMT-3	1m 53s	12.53 MB	27 Mar 2025 15:26:42 GMT-3
☐ rec_0dU7GBVCJ1YFqQ	Acceleration_0.mcap	FSDS	14 Feb 2025 14:35:56 GMT-3	19s	287.33 MB	16 Feb 2025 18:07:21 GMT-3
☐ rec_0dTQUCQVas3E309	Track_Drive_3Laps_0.mcap	FSDS	14 Feb 2025 14:31:01 GMT-3	2m 23s	4.15 GB	14 Feb 2025 15:03:51 GMT-3
☐ rec_0dTtCDEBiqNmr	Track_Drive_0.mcap	FSDS	14 Feb 2025 14:28:34 GMT-3	1m 10s	1.82 GB	14 Feb 2025 14:58:44 GMT-3
☐ rec_0dT8RvLWxQHKs	Skidpad_0.mcap	FSDS	14 Feb 2025 14:38:02 GMT-3	1m 4s	1.61 GB	14 Feb 2025 14:57:04 GMT-3
☐ rec_0dTcgWd8h8Fbol	rosbag2_2025_02_05-17_30_37.mcap	OAK-D-LR	05 Feb 2025 17:30:37 GMT-3	11s	32.34 MB	12 Feb 2025 16:46:38 GMT-3
☐ rec_0dTcgW01auKNC14	rosbag2_2025_02_05-17_21_30.mcap	OAK-D-LR	05 Feb 2025 17:21:30 GMT-3	9s	24.09 MB	12 Feb 2025 16:46:16 GMT-3
☐ rec_0dTcgR7tarDxtv	rosbag2_2025_02_05-17_16_59.mcap	OAK-D-LR	05 Feb 2025 17:16:59 GMT-3	9s	26.14 MB	12 Feb 2025 16:45:35 GMT-3
☐ rec_0dTcgFYXv8Jfxy	rosbag2_2025_02_05-17_13_19.mcap	OAK-D-LR	05 Feb 2025 17:13:20 GMT-3	9s	28.33 MB	12 Feb 2025 16:45:19 GMT-3
☐ rec_0dTcg1hdoSGVRMa	rosbag2_2025_02_05-17_03_12.mcap	OAK-D-LR	05 Feb 2025 17:03:12 GMT-3	10s	28.68 MB	12 Feb 2025 16:45:06 GMT-3

Cada dispositivo gera uma série de gravações que são organizadas em eventos. Os eventos são organizados para representar cada uma das provas tradicionais da competição FSAE. São elas as provas de aceleração, *autocross*, *endurance* e *skidpad*. Além das provas, os testes livres são a quinta categoria que descreve qualquer trajeto que o carro percorreu sem necessariamente ser parte de uma prova.

3.5.3 Python

A linguagem de programação escolhida para ser a principal linguagem para construção do DAS é o Python. Esta escolha foi baseada em uma série de fatores que destacam sua versatilidade, eficiência e ampla adoção na comunidade científica e de engenharia. O principal motivo para esta escolha é o robusto ecossistema de bibliotecas e pacotes disponíveis para o Python referente a processamento, análise e visualização de dados. Bibliotecas como NumPy, Pandas, Matplotlib e SciPy oferecem ferramentas que facilitam a manipulação dos dados, aplicação de algoritmos e a geração de gráficos e relatórios, aspectos relevantes para a construção de uma plataforma de armazenamento e tratamento de dados, essencial para o escopo do trabalho.

Outro aspecto relevante é a integração do Python com ferramentas modernas de *pipeline* de dados como o Mage. Com esta integração, é possível criar fluxos de dados automatizados e escaláveis, desde a extração dos dados até o processo de visualização e geração de relatórios. Além da integração com ferramentas modernas, o Python também possui suporte nativo para comunicação com APIs, fundamental para conseguir extrair os dados da FoxGlove e gerar integração entre diferentes componentes do DAS.

Por fim, a popularidade e a facilidade de aprendizado do Python foram fatores decisivos. A linguagem é amplamente utilizada em projetos acadêmicos e industriais, o que garante uma vasta documentação, tutoriais e suporte da comunidade. Além disso, sua sintaxe simples e intuitiva permite um desenvolvimento ágil, reduzindo o tempo necessário para implementar e testar novas funcionalidades. Essas características, combinadas com a portabilidade do Python (que pode ser executado em diferentes sistemas operacionais), tornaram-no a escolha ideal para atender às demandas técnicas e operacionais deste projeto.

O DAS utiliza a versão 3.10 do Python e as seguintes bibliotecas: pandas (1.5.3), mcap-protobuf-support (0.0.10), mcap-ros2-support (0.1.0), mcap-ros1-support (0.4.0), foxglove-data-platform (0.2.3), scipy (1.15.2), matplotlib (3.10.1), google-api-python-client (2.70.0) e google-auth (2.38.0).

3.5.4 Mage

Utilizar o Mage como orquestrador de dados foi uma escolha motivada pela capacidade da ferramenta de oferecer organização, escalabilidade e flexibilidade na criação e gestão dos *pipelines* de dados. O Mage é uma ferramenta de código aberto projetada para simplificar a criação, execução e monitoramento de fluxos de trabalho de dados, sendo especialmente útil em projetos que demandam integração com múltiplas fontes de dados e ferramentas de análise, pois conta com uma série de conectores nativos que permitem mobilizar os dados de um lugar para o outro. Sua arquitetura modular permite a construção de *pipelines* robustos e escaláveis, que podem ser facilmente adaptados às necessidades do projeto, desde a extração de dados até a geração de relatórios automatizados.

O principal diferencial do Mage é o fato de ele ser distribuído como uma biblioteca do Python, isto é, a ferramenta foi criada de ponta a ponta utilizando esta linguagem amplamente adotada na comunidade para processamento e análise de dados. Essa integração permite fazer uso do Mage em conjunto com todo o ecossistema do Python diretamente nos *pipelines*, facilitando a implementação de transformações complexas e análises personalizadas. Além disso, o Mage oferece suporte a diversas ferramentas de banco de dados, como PostgreSQL, o que é essencial para garantir a interoperabilidade com o sistema de armazenamento de dados utilizado no projeto.

Outro aspecto relevante é o fato de o Mage ser uma solução de código aberto, o que proporciona maior transparência, customização e redução de custos. Essa característica é particularmente importante para a equipe Ampera Racing, que opera com recursos limitados e necessita de uma ferramenta que possa ser adaptada às suas necessidades específicas sem custos adicionais. A combinação desses fatores — organização, escalabilidade, integração com Python e bancos de dados, e sua natureza código aberto — tornou o Mage a escolha ideal para orquestrar os *pipelines* de dados do projeto, garantindo eficiência e confiabilidade no processamento e na análise dos dados coletados.

3.5.5 Visualização e Disponibilização de Dados

Para a visualização dos dados coletados e processados e a geração de gráficos, foram utilizadas as bibliotecas Matplotlib e Seaborn, ambas amplamente reconhecidas no ecossistema Python por sua praticidade, flexibilidade e capacidade de gerar gráficos de alta qualidade. A escolha dessas ferramentas foi motivada pela necessidade de criar visualizações claras e informativas que pudessem ser geradas diretamente a partir do código, em tempo de execução, sem a necessidade de ferramentas externas ou interfaces adicionais. Essa praticidade permitiu uma integração perfeita com o *pipeline* de dados, facilitando a geração de relatórios e a análise iterativa dos resultados.

Além da praticidade, a customização oferecida por essas bibliotecas é um fator decisivo. Durante o projeto, foi estabelecido um padrão visual para todos os gráficos, garantindo consistência e clareza nas apresentações. O Matplotlib, com sua ampla gama de opções de personalização, permite ajustar detalhes como cores, fontes, legendas e escalas, enquanto o Seaborn, construído sobre o Matplotlib, trouxe funcionalidades adicionais para a criação de gráficos estatísticos mais complexos, como *heatmaps* e gráficos de distribuição. Essa combinação de ferramentas possibilitou a criação de visualizações que não apenas atendiam às necessidades técnicas do projeto, mas também facilitam a interpretação dos dados por parte de toda a equipe.

Outro aspecto relevante é a integração dessas bibliotecas com o ecossistema Python, o que permitiu a utilização de dados diretamente dos DataFrames do Pandas, sem a necessidade de conversões ou etapas intermediárias. Essa integração, aliada à vasta documentação e à comunidade ativa em torno dessas ferramentas, garantiu um desenvolvimento ágil e eficiente, reduzindo o tempo necessário para implementar e ajustar as visualizações. Dessa forma, a escolha

do Matplotlib e do Seaborn como ferramentas de visualização de dados mostrou-se alinhada com os objetivos do projeto, proporcionando uma solução robusta e adaptável para a análise e a comunicação dos resultados.

Os dados tratados são salvos em tabelas através do formato CSV. Este formato é amplamente utilizado como armazenamento de dados de forma prática, permitindo a integração com inúmeras ferramentas de análise e visualização de dados. Com os dados em CSV, os analistas podem gerar gráficos e cálculos matemáticos através do Excel e carregar ao Python ou qualquer linguagem de programação compatível com a leitura deste formato para fazer análises avançadas em cima dos dados tratados. Estas formas de análise são compatíveis com as capacidades da equipe de análise de dados, que já vem utilizando essas ferramentas ao longo dos últimos anos.

3.5.6 Pipeline de Dados

O *pipeline* de dados é a camada responsável pela interação com os dados brutos da FoxGlove e a sua devida transformação em informações de alto valor para análise através de processos de limpeza e tratamento de dados. Conforme observado na Figura 18, o *pipeline* é responsável por coletar os dados brutos e transformá-los em dados tratados que, por sua vez, são utilizados para análise. A construção de *pipelines* robustos é fundamental, pois garante que os dados passem por uma série de etapas automatizadas que aumentam sua qualidade, consistência e integridade. Um *pipeline* bem estruturado atua como um filtro, eliminando ou corrigindo fatores que podem comprometer a análise, como dados ausentes, incorretos, duplicados ou fora do padrão.

Figura 18 – Fluxograma demonstrando papel do *pipeline* em coletar os dados brutos e transformá-lo em dados tratados.



Fonte: Autoria própria

O *pipeline* de dados do DAS foi cuidadosamente projetado para processar os dados dos sensores de maneira eficiente e robusta, garantindo que os resultados sejam precisos, acessíveis e organizados. Cada atividade executada no *pipeline* desempenha um papel crucial no fluxo, resolvendo desafios específicos e assegurando que o sistema entregue informações de alta qualidade para a equipe de corrida.

O *pipeline* do DAS foi construído seguindo o princípio de ELT, isto é, os dados brutos são coletados dos seus devidos sistemas e carregados em um banco de dados ainda no formato bruto, sem que haja nenhuma transformação. Este processo é a parte de extração e carregamento (EL). Após os dados brutos estarem todos no mesmo local e de forma organizada, a etapa de

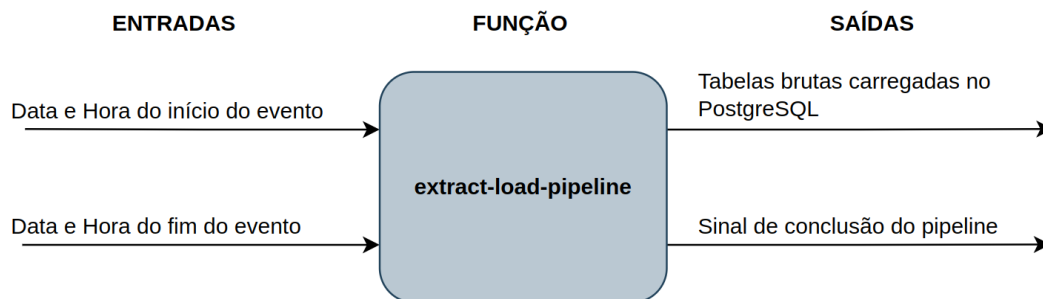
transformação (T) é responsável por dar os devidos tratamentos e cruzar os dados conforme necessário para chegar em uma modelagem final que atende aos requisitos de análise. Para o caso dos *pipelines* do DAS, os dados primeiro são coletados na fonte e inseridos na Foxglove (esta etapa não faz parte do ELT, mas sim da fonte de dados extraída), depois extraídos da Foxglove e carregados ao Postgres (que caracteriza a etapa de EL), por fim transformados via Python (que caracteriza a etapa de T). Todas essas etapas, com exceção da Aquisição de Dados, são orquestradas pelo Mage.

3.5.6.1 Extração e Carregamento

A etapa de extração e carregamento (EL) se inicia com a extração dos dados da plataforma da FoxGlove através da API da mesma. Um código Python é responsável por se conectar com a API e retornar todas as mensagens do arquivo MCAP salvo na FoxGlove para um determinado dispositivo. É necessário fornecer data e hora de início e data e hora de fim para que sejam coletados os dados corretos. Essa informação de entrada pode ser organizada diretamente na FoxGlove através de Eventos e desta forma a API retornará todos os dados existentes no evento. Porém, na ausência de um evento, também é possível extrair os dados diretamente das gravações passando início e fim da gravação.

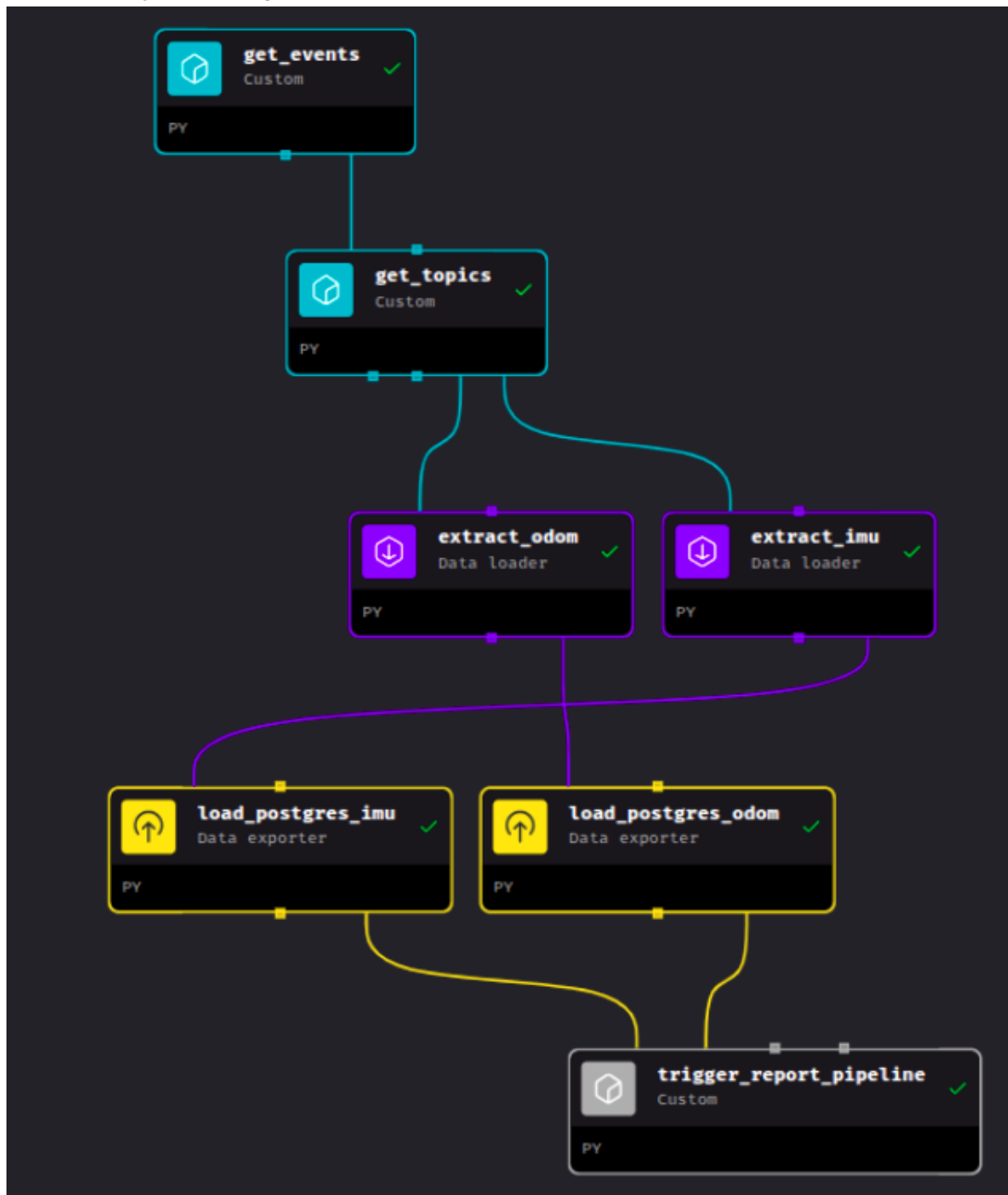
No Mage, o código Python do EL foi organizado em um *pipeline* chamado *extract-load-pipeline*. O papel do Mage aqui é organizar os múltiplos passos e garantir o sequenciamento correto das atividades. No caso do *pipeline extract-load-pipeline*, a primeira atividade é a extração dos eventos via API, seguido pelas atividades de extração das mensagens via API, organização das mensagens em um DataFrame do Pandas, inserção dos dados do DataFrame na respectiva tabela do SQL para cada sensor e, por fim, o envio do sinal para a API interna do Mage de que o *pipeline* finalizou. A Figura 19 representa o diagrama funcional deste *pipeline* representando as entradas e saídas do mesmo. O Mage fornece em sua interface visual uma ferramenta para visualizar o diagrama das operações sendo executadas. A Figura 20 contém este diagrama.

Figura 19 – Diagrama funcional do *pipeline* de Extração e Carregamento (EL)



Fonte: Autoria própria

Figura 20 – Diagrama automaticamente criado no Mage através da construção do código do *pipeline* de Extração e Carregamento (EL).



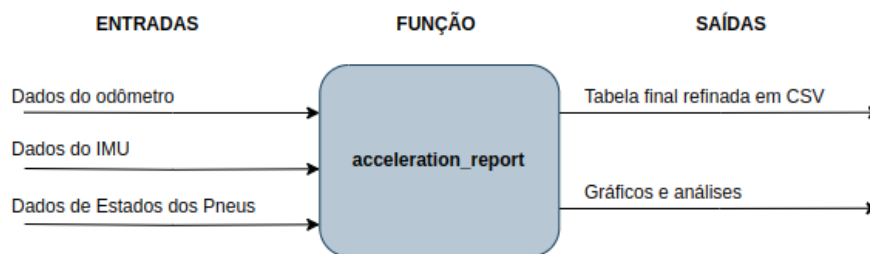
Fonte: Autoria própria

3.5.6.2 Transformação

O *pipeline extract-load-pipeline* finaliza com o carregamento de forma organizada dos dados no banco de dados PostgreSQL ainda em forma bruta. Portanto, a partir deste momento, qualquer manipulação destes dados possui o objetivo de deixá-los estruturados para análise, isto é, prontos para serem usados para criar gráficos, relatórios e calcular métricas. Portanto, em contraste ao *pipeline* de extração e carregamento que possui entradas e saídas semelhantes independente da finalidade de análise, a camada de transformação não é tão genérica e, por este motivo, múltiplos *pipelines* de transformação podem ser construídos com finalidades diferentes.

Apesar da plataforma ter sido pensada para gerar múltiplas transformações, este trabalho foca no desenvolvimento de um *pipeline* de transformação que deve gerar gráficos e análises necessários como relatório de uma prova de aceleração com o carro da Ampera Racing. Em uma prova de aceleração, o objetivo é que o carro seja capaz de acelerar do ponto inicial até o ponto final, totalizando 75 metros de distância, no menor tempo possível. Logo, a motivação das análises por trás desta prova estão ligadas ao interesse do time em reduzir o tempo de execução. A figura 21 mostra o diagrama funcional do *pipeline* de transformação com foco na prova de aceleração. A seção 3.6 a seguir detalha os métodos, compostos por transformações de dados e cálculos matemáticos realizados para chegar no resultado final esperado, representados pelas saídas do diagrama funcional.

Figura 21 – Diagrama funcional do *pipeline* de Relatório de Aceleração.



Fonte: Autoria própria

3.6 Métodos

Nesta seção, são explicados os métodos utilizados ao longo do *pipeline* de transformação para tratar os dados e chegar nos resultados finais que permitirão a geração do relatório de aceleração.

3.6.1 Tempo de Prova

A principal variável que o relatório deve apresentar é o tempo de execução da prova, que é a variável foco da otimização pela equipe. Para calcular o tempo total da prova, cruzou-se os dados de posição relativa do carro, provenientes do sinal *twist linear x* do sensor de odometria, com os dados de ponteiro, isto é, data e hora de cada aquisição, representados pelo campo *publish time*, proveniente do *clock*, ou relógio, da aquisição via ROS 2.

Para identificar o início da prova de aceleração, dois métodos foram utilizados. O primeiro método consiste em fornecer a data e hora de início da prova conforme cronometrado por algum membro da equipe. Esse método, apesar de depender da precisão do observador para registrar a data e hora correta, garante preciosismo na hora de calcular o tempo de prova, pois vai considerar o tempo de reação do piloto. O segundo método, que foi utilizado no *pipeline* de aceleração para todos os resultados de simulação, consiste em calcular o início da prova como

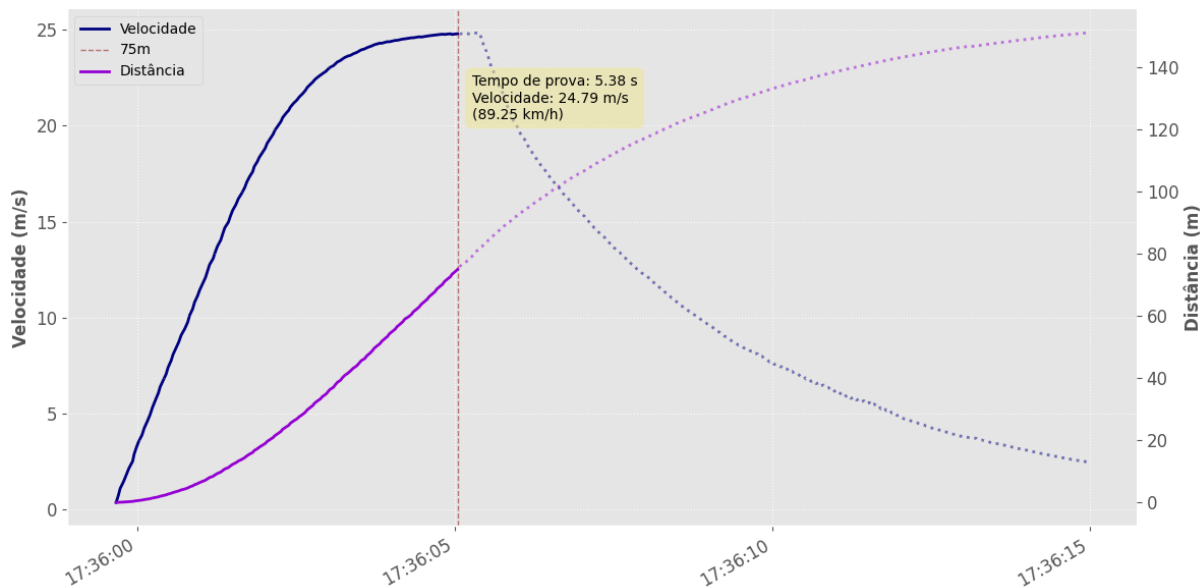
sendo o momento que o carro começa a se movimentar. No início da gravação do ROS 2, o carro está na posição 0, a partir do momento que o carro começa a acelerar, o valor da posição começa a subir. Para levar em consideração as imperfeições do odômetro, utilizou-se um limite de 0,1 metros como sendo o momento que o algoritmo identifica o carro como em movimento, isto é, o momento que o carro deixou de estar na posição 0 metros e atingiu a primeira posição maior que 0,1 metros.

Para identificar o fim de prova, o algoritmo utiliza o momento em que o carro passa pela marca de 75 metros. Essa marca é utilizada na competição FSAE como a distância oficial da prova de aceleração e é a marca de chegada. Logo, quando os dados mostram que o carro atravessou a marca de 75 metros, utiliza-se esse momento como fim de prova. O tempo total de prova é, por consequência, o tempo total t_{prova} que o carro levou para atingir os 75 metros, conforme a equação:

$$t_{prova} = t_{x=75} - t_{x=0} \quad (3.1)$$

A Figura 22 demonstra um dos gráficos que fazem parte do relatório final de aceleração. Este gráfico mostra velocidade e distância longitudinal do carro ao longo do tempo, destacando o período da prova e trazendo a informação do tempo de prova e velocidade atingida na marca de 75 metros. A velocidade neste instante é obtida da mesma maneira que o tempo final de prova, sendo a velocidade que o carro possui no primeiro instante após passar a marca de 75 metros.

Figura 22 – Velocidade e distância do carro do simulador em uma prova de aceleração, destacando o percurso de 75 metros da prova, o tempo total da prova e a velocidade final ao atingir a marca de 75 metros.



Fonte: Autoria própria

Com o tempo de prova, o código Python atualiza a tabela de odometria com o estado do carro. Na Tabela 3, é mostrado um resumo das regras utilizadas no algoritmo para classificar

o estado do carro com base na distância e velocidade. O estado é utilizado em regras posteriores para separar os períodos da prova e identificar o carro em momento de aceleração. Os estados possíveis considerados foram o de carro parado, carro em aceleração e carro em frenagem.

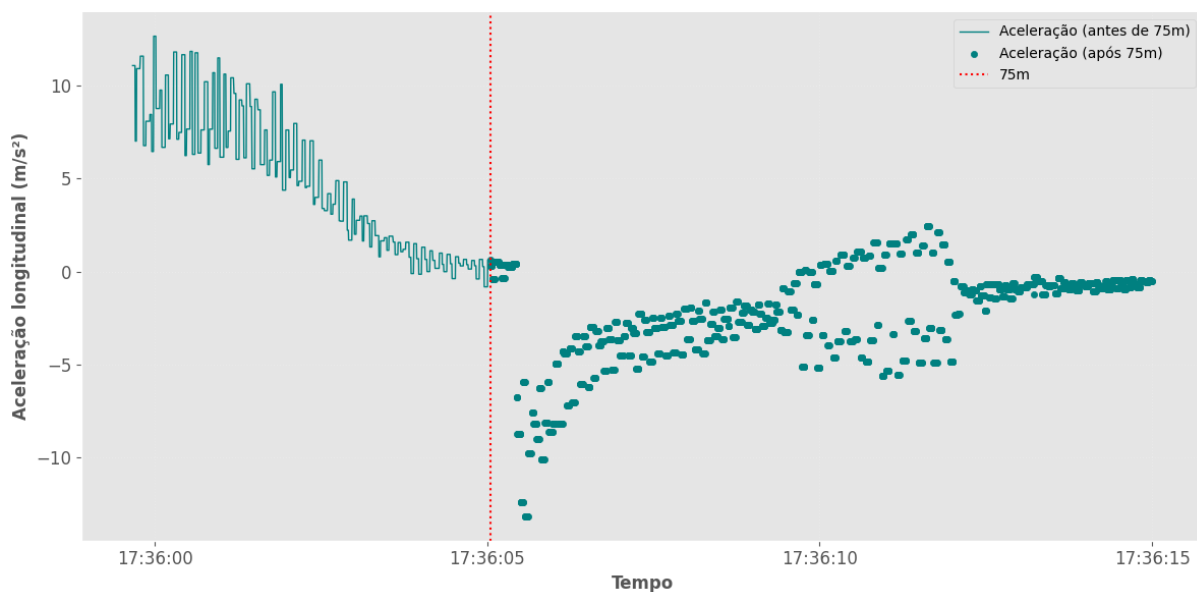
Tabela 3 – Regras de classificação do estado do carro com base na distância e velocidade longitudinal.

Distância (m)	Velocidade (m/s)	Estado do carro
$0 \pm 0,1$	$0 \pm 0,1$	Parado
Entre 0,1 e 75	Maior que 0	Aceleração
Maior que 75	Maior que 0	Frenagem
Maior que 75	$\pm 0,1$	Parado

3.6.2 Tratamento do sinal de aceleração

Ao analisar os dados de aceleração provenientes do IMU, tanto para os gerados pelo simulador quanto os gerados pelo carro real, notou-se a presença de ruído no sinal, gerando uma variância irregular em cima do comportamento real do carro. A Figura 23 mostra a aceleração obtida pelo IMU durante uma das simulações para a mesma prova de aceleração analisada na seção anterior. Nesta visualização, é possível perceber a variância do sinal.

Figura 23 – Aceleração longitudinal ao longo do tempo, destacando marca de 75 metros.

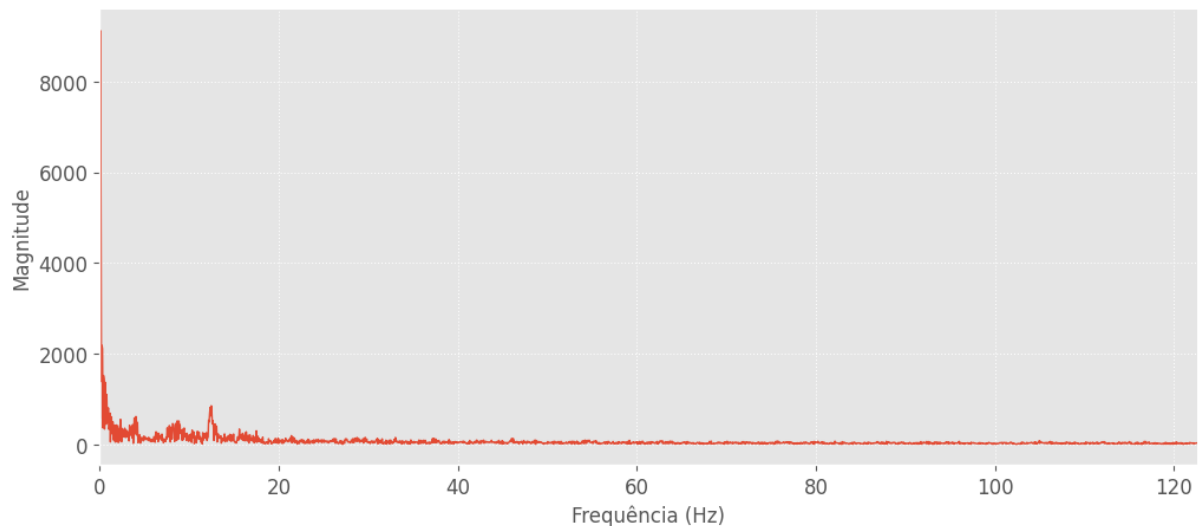


Fonte: Autoria própria

Para entender o comportamento deste sinal, calcula-se a Transformada Discreta de Fourier nos dados, convertendo o sinal para o domínio da frequência, permitindo identificar a natureza do ruído apresentado. Utilizando a biblioteca Scipy do Python, aplicou-se o módulo de Transformada Rápida de Fourier nos dados de aceleração do IMU, removendo os estados de carro parado e carro em frenagem. A Figura 24 mostra o resultado da Transformada Rápida de Fourier. Percebe-se a presença de uma magnitude maior próximo de 0 Hz, o que faz sentido com o componente real da aceleração que é um evento de baixa frequência. Percebe-se também a

presença de picos de magnitude em variadas frequências ao longo do espectro, identificando potencial ruído proveniente da vibração do motor e rotação dos pneus. Essa energia do sinal de alta frequência está apenas escondendo a real aceleração do carro e devem ser filtrados do sistema.

Figura 24 – Comportamento da aceleração no espectro de frequência, calculado através da Transformada Discreta de Fourier

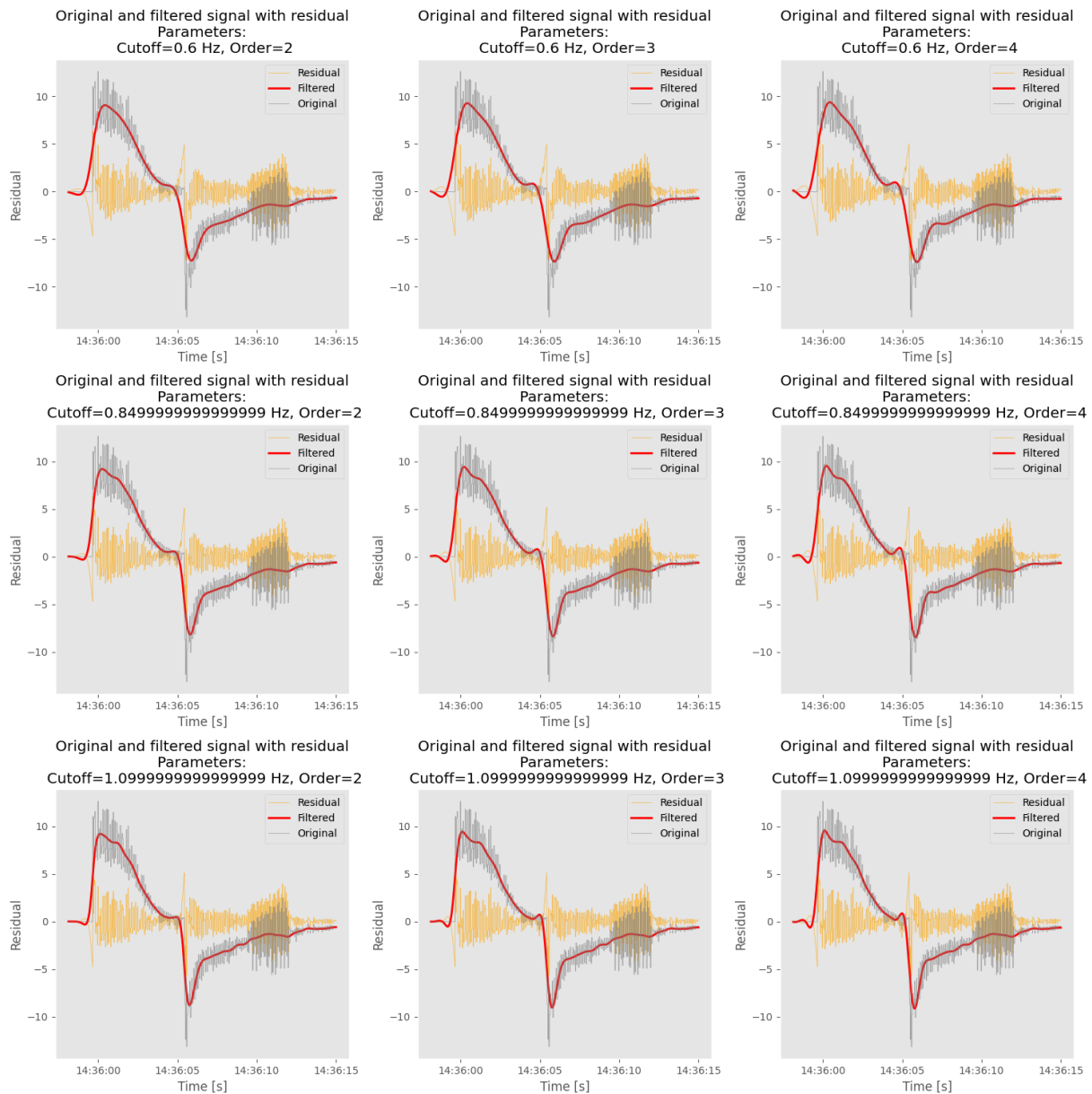


Fonte: Autoria própria

Portanto, como as frequências desejadas são frequências baixas, um filtro passa-baixa foi utilizado. O filtro *Butterworth* foi escolhido para este caso por ser um filtro tradicionalmente utilizado para filtrar sinais de aceleração. O desempenho foi avaliado através de dois métodos: inspeção visual e cálculo do tempo de prova. Os parâmetros do filtro de frequência de corte e ordem foram variados em uma pesquisa de grade para identificar a melhor desempenho do filtro, conforme demonstrado na Figura 25. A Figura 26, por sua vez, mostra ambas as curvas de aceleração original e filtrada utilizando parâmetros específicos, que foram atualizados de maneira abrangente para efetuar este modelo de otimização.

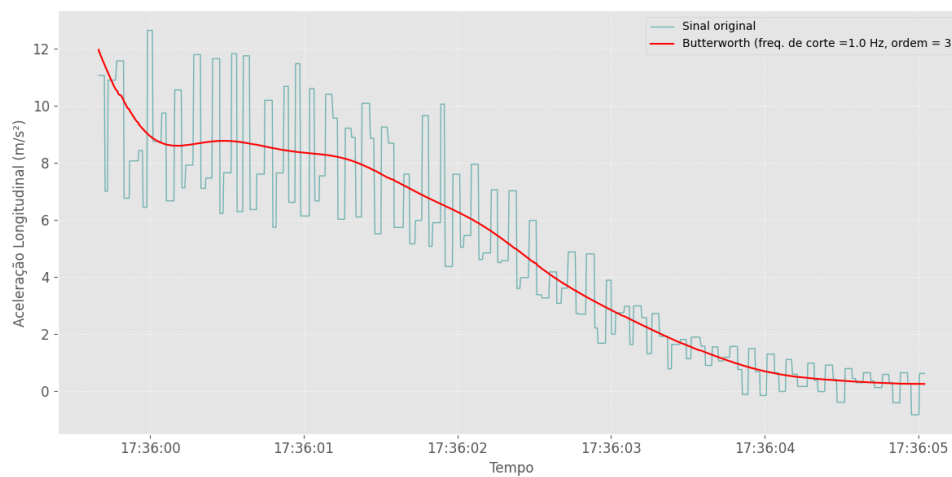
A inspeção visual consistiu em observar o comportamento do gráfico e identificar uma suavidade da curva. Porém, apenas a inspeção visual não é suficiente para julgar o filtro como ótimo. Portanto, a partir da segunda derivada da aceleração, calculou-se o tempo no qual o carro atravessaria a marca dos 75 metros caso ele estivesse operando com as acelerações previstas pós filtragem. Este método de cálculo do tempo de prova permitiu avaliar qualitativamente o desempenho do filtro. A Figura 27 mostra a visualização utilizada para identificar o tempo de prova calculado e como ele se compara com o tempo real. Para o exemplo em questão, o tempo de prova calculado foi de 4,49 segundos. O tempo real de prova foi de 5,38 segundos, logo o filtro apresentou um erro absoluto de 0,89 segundos em relação ao tempo real. Os parâmetros foram otimizados e este foi o menor erro obtido utilizando este método para esta coleta de dados.

Figura 25 – Pesquisa de Grade (ou *grid search*, no inglês) utilizada para auxiliar o método de inspeção visual.



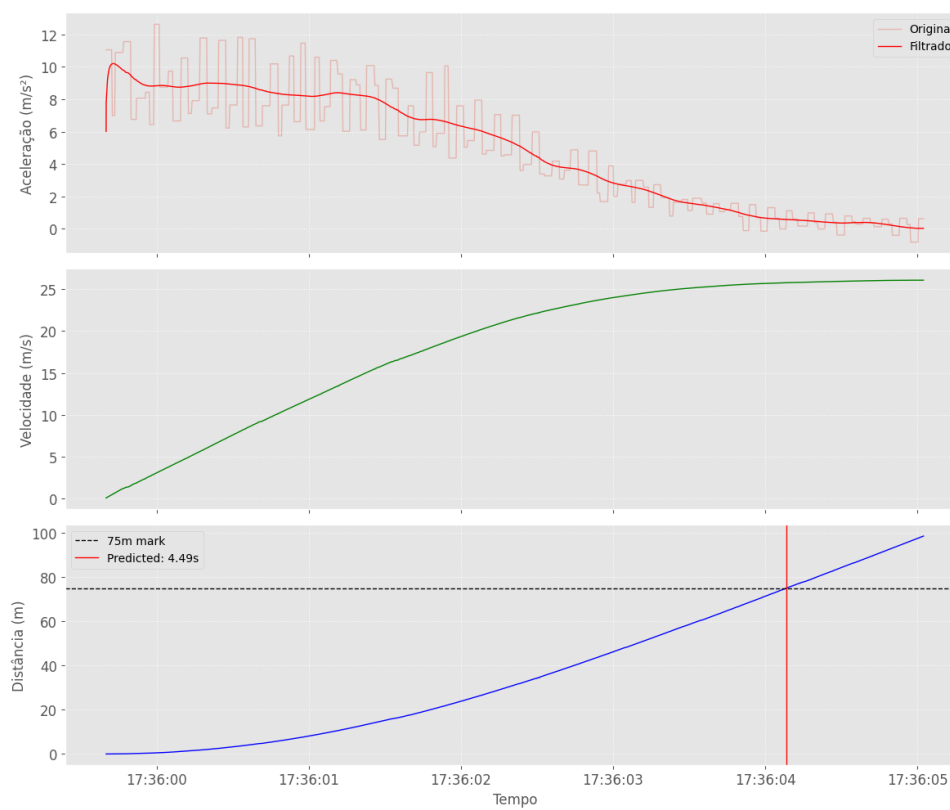
Fonte: Autoria própria

Figura 26 – Aceleração original e filtrada em função do tempo. Filtro utilizado: Butterworth. Parâmetros utilizados: frequência de corte = 1 Hz e ordem = 3.



Fonte: Autoria própria

Figura 27 – Velocidade e distância percorrida calculados através da aceleração filtrada.



3.6.3 Junção dos dados

Para gerar o relatório final, é fundamental cruzar dados de IMU, odômetro e variáveis de estado dos pneus como velocidade angular. Cada um desses sensores possui uma taxa de aquisição diferente, o que implica na necessidade de definir uma estratégia de junção desses dados. Todas as tabelas desses dados possuem um campo de ponteiro proveniente do relógio do ROS 2, porém não há sincronia entre essas aquisições.

Primeiro, calculou-se a frequência de aquisição F_s dos três sensores, através da equação:

$$F_s = \frac{1}{T_s} \quad (3.2)$$

Sendo:

T_s - passo de amostragem, isto é, o tempo entre cada amostra e

F_s - frequência amostral.

O tempo de amostra foi calculado através do número total de amostras n e o tempo total de aquisição t_{total} . Essa operação é representado pela equação:

$$T_s = \frac{t_{total}}{n} \quad (3.3)$$

Logo, em resumo, a frequência de aquisição F_s depende apenas do número total de amostras e o tempo total decorrido.

$$F_s = \frac{n}{t_{total}} \quad (3.4)$$

Na Tabela 4 é mostrado o tempo de aquisição dos três sensores utilizados no relatório para uma prova de aceleração específica. Percebe-se a falta de sincronia de aquisição entre eles.

Tabela 4 – Número de amostras, tempo total e frequência de amostragem para um exemplo de prova de aceleração.

Sensor	n	t_{total} (s)	$F_s = n/t_{total}$ (Hz)
IMU	4137	16,83	245,31
Odômetro	729	16,83	43,31
Velocidade angular dos pneus	1683	16,83	99,82

Dois métodos de junção das tabelas foram utilizados neste trabalho, optando por um ou outro dependendo da situação. O primeiro método foi o de junção pelo mais próximo e a tabela resultante herdou a menor frequência das três. Este método consiste em juntar os dados através dos dados do odômetro, que possui menor frequência, unindo os dados das outras tabelas quando o tempo do ponteiro é o mais próximo possível para cada linha através do método de junção à esquerda (*left join*). Dessa maneira, valores de IMU e do estado dos pneus foram perdidos mas

o volume de dados resultante é menor e permite análises mais rápidas e menos ruidosas. Esse método de junção foi utilizado em cálculos onde a precisão não é importante, como por exemplo médias e cálculos entre fim e começo da aquisição de dados.

O segundo método utilizado foi o método da interpolação e este método resultou em uma tabela com frequência igual a maior frequência dos três. Através dos dados do IMU, os dados foram cruzados utilizando também uma estratégia de junção à esquerda, porém preenchendo as lacunas para os dados de odômetro e estado dos pneus através da média entre dois pontos adjacentes, interpolando as informações. Por exemplo, se a velocidade é 10 m/s no instante t_0 e 12 m/s no instante t_1 , em um instante interpolado $t_{1/2}$ é de 11 m/s. Essa operação foi calculada dinamicamente para todas as linhas da tabela. Esse método é mais utilizado no relatório para casos que necessitam analisar o comportamento do traçado de começo a fim, sem que dados sejam perdidos.

3.6.4 Cálculo do *Slip Ratio*

Durante a prova de aceleração, é fundamental entender o quanto o pneu do carro é capaz de transferir cargas para o chão e, conseqüentemente, gerar mais torque. Portanto, o conceito de taxa de deslizamento é fundamental para identificar o quanto o pneu está deslizando em relação ao chão no período de aceleração. Para efetuar essa análise, é necessário utilizar os dados dos sensores de Estado da Roda (*wheel states*), que contemplam as velocidades angulares de cada uma das quatro rodas. Além disso, a velocidade longitudinal do carro é necessária, proveniente do odômetro.

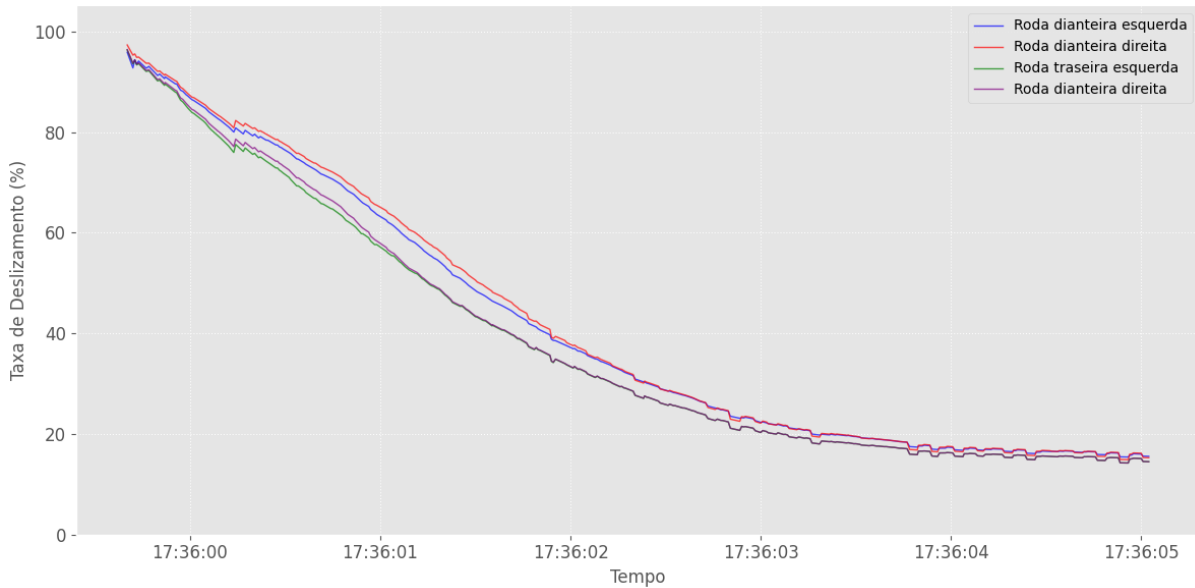
Utilizando a Equação 2.19, que calcula a versão normalizada do escorregamento, foram calculados para cada momento no tempo a taxa de escorregamento para cada uma das quatro rodas e os resultados obtidos são demonstrados na Figura 28. Essa análise gráfica faz parte do relatório final de aceleração do DAS.

Com esta visualização e outros parâmetros calculados como a taxa de deslizamento máxima e o tempo para que a taxa de deslizamento atinja um valor ideal, os membros da equipe e o piloto do carro possuem informação para otimizar a aceleração do carro, bem como essa informação pode ser utilizada pelo *Driverless* com o objetivo de otimizar a aceleração via controle de tração.

3.6.5 Diagrama Aceleração e Velocidade

Durante a prova de aceleração, é importante entender se a potência providenciada pelo motor elétrico está sendo convertida em torque e, conseqüentemente, em aceleração de forma otimizada. Para descrever o comportamento dinâmico do carro e mostrar aos engenheiros um senso de transferência de energia em aceleração, com a finalidade de comparar as diferenças entre iterações nos testes da prova de aceleração, o relatório do DAS fornece um gráfico comparando

Figura 28 – Taxa de deslizamento calculada para as 4 rodas para uma tomada de dados da prova de aceleração via simulador FSDS (*Formula SAE Driving Simulator*). O raio do pneu utilizado nesta análise foi de 0,2286 metros, conforme configurado no simulador.



Fonte: Autoria própria

aceleração e velocidade do carro ao longo do percurso de 75 metros.

Para construir esse gráfico, o algoritmo utiliza da velocidade proveniente do odômetro presente nos dados como *twist linear x* e da aceleração proveniente do IMU apresentada nos dados como *linear acceleration x*. Porém, conforme tratado na seção 3.6.2, o algoritmo utiliza-se dos dados já tratados com a devida remoção de ruídos. São utilizados duas acelerações para gerar os gráficos, sendo uma delas a aceleração filtrada do IMU e a aceleração calculada através da derivada discreta da velocidade do odômetro. O motivo pelo qual essa comparação é fornecida está relacionado aos problemas de desempenho do filtro Butterworth discutido na seção 3.6.2. Em algumas iterações dos testes, o desempenho, calculado pelo tempo total de prova, reduz a confiança nos dados do IMU.

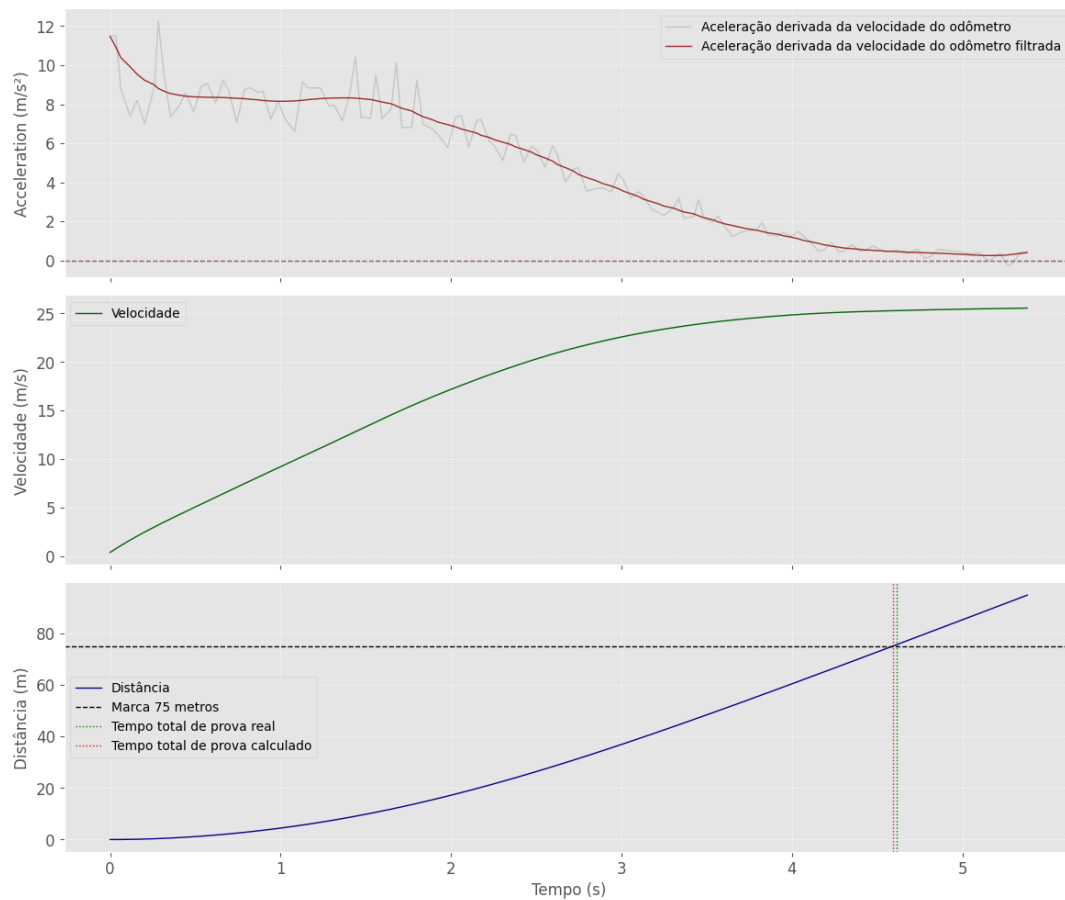
Para determinar a aceleração através da velocidade do odômetro, o algoritmo calcula a derivada discreta da velocidade, utilizando-se da diferença entre as velocidades e tempo entre pontos adjacentes. Esse método de diferenciação é chamado de método de diferenciação central e é escolhido, neste caso, para garantir a precisão no cálculo das acelerações, em detrimento da perda de referência no tempo. Essa perda de referência absoluta do tempo se dá pois os novos pontos de dados são calculados como sendo a metade entre dois pontos previamente estabelecidos, mudando as referências de relógio. O cálculo da aceleração através deste método pode ser descrito pela equação:

$$a_{i-\frac{1}{2}} = \frac{\Delta v}{\Delta t} = \frac{v_i - v_{i-1}}{t_i - t_{i-1}} \quad (3.5)$$

Esse método de diferenciação, apesar de fornecer acurácia na resposta, lidar bem com

ruídos de alta frequência e não induzir mudanças de fase nos dados, ele possui a consequência de ampliar ruídos da velocidade, especialmente os de baixa e média frequência. Por isso, os dados do IMU, mesmo apresentando baixo ruído, foram suavizados utilizando o filtro *Butterworth*, da mesma forma que foi utilizado nos dados do IMU. A Figura 29 mostra os resultados obtidos para o mesmo teste de aceleração avaliado até o momento. Para este caso, o método de diferenciação calculou um tempo total de prova de 4,59 segundos, muito próximo dos 4,61 segundos reais apresentados pelo relógio.

Figura 29 – Aceleração, velocidade e distância derivadas da velocidade do sensor IMU e filtradas através de um filtro *Butterworth* com frequência de corte de 2 Hz e ordem 2.

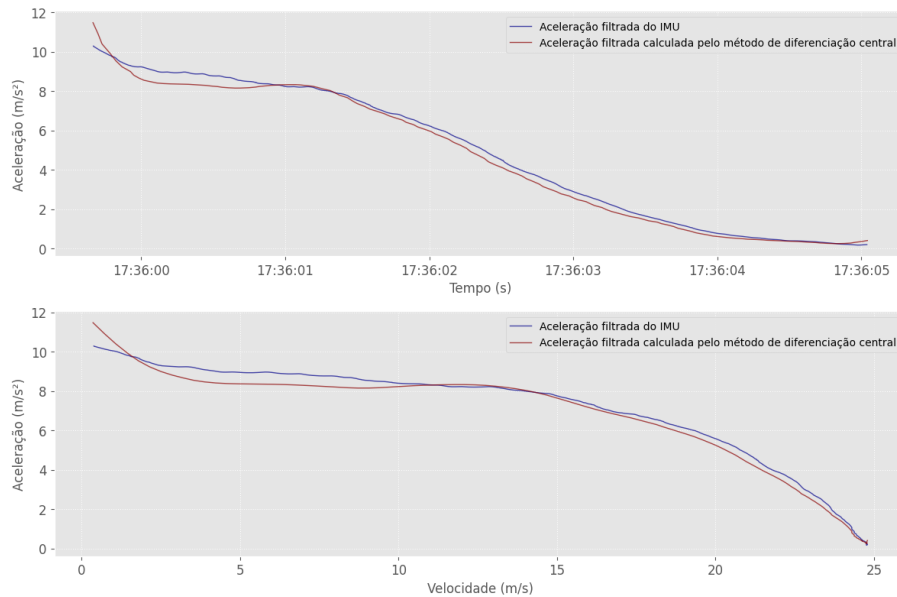


Fonte: Autoria própria

Por fim, cruzando as informações de aceleração (obtidas pelo IMU e odômetro) são adicionadas no gráfico da Figura 30 em um diagrama de aceleração por velocidade. Percebe-se a similaridade no resultado entre os resultados obtidos, com nítida distinção em alguns pontos do gráfico. Com essa visualização, somado às visualizações anteriormente mostradas, os engenheiros podem decidir em qual dado confiar mais e ter uma interpretação gráfica de como o carro está respondendo à aceleração.

Os resultados obtidos relacionando aceleração e velocidade, para ambos os métodos de obtenção, evidenciaram o comportamento esperado do carro. Em baixa velocidade, a aceleração é limitada pela aderência. Este limite fica evidente no gráfico até uma velocidade de

Figura 30 – Gráficos de aceleração pelo tempo e aceleração por velocidade conforme apresentada no relatório final do DAS.



aproximadamente 14 m/s. Em alta velocidade, a aceleração é limitada pela potência do motor e o aumento da força de resistência aerodinâmica. Este comportamento, por sua vez, pode ser identificado no gráfico a partir da velocidade de 14 m/s pelo decaimento da aceleração quanto maior é a velocidade.

O uso de ambos os métodos de obtenção da aceleração, portanto, é justificado pela característica da obtenção dos dados. Caso haja problemas de calibração no IMU gerando ruídos, a aceleração calculada pela derivada da velocidade do odômetro pode ter um desempenho melhor. Porém, se houver o contrário e os dados do odômetro estiverem apresentando ruído, os dados do IMU podem ser mais confiáveis. Por isso, ambos os métodos foram utilizados e os resultados expostos no relatório final do DAS.

4 Resultados

Tendo em vista os métodos utilizados até o momento para a construção do DAS, é possível avaliar a capacidade desta plataforma em atingir os objetivos propostos e apresentar os resultados obtidos.

4.1 Desempenho da Plataforma

Em termos de desempenho global da plataforma, o fluxo de dados construído tornou possível a análise de dados de forma confiável e escalável. Os dados obtidos de forma padronizada através do MCAP e a construção de um *pipeline* responsável apenas pela extração e carregamento dos dados de forma organizada reduziu os erros de carregamento de dados drasticamente. Não é possível apresentar uma métrica quantitativa de redução de erros, visto que não há dados comparativos. Contudo, no passado, toda alteração no código de aquisição de dados geravam retrabalho no *pipeline* de dados posterior devido a ausência de padronização na troca de mensagens entre os sistemas. Com esse novo protocolo através do MCAP, alterações como adicionar ou remover sensores, alterar a desserialização e até mesmo a refatoração dos cálculos matemáticos - procedimentos comuns de acontecerem durante uma temporada - tiveram impacto consideravelmente reduzido na manutenção da análise de dados.

A presença do banco de dados PostgreSQL tornou mais simples a consulta e análise dos dados, pois não há necessidade do engenheiro desenvolver seu próprio mecanismo de coleta e disponibilização de dados. A interação direto com o banco de dados acontece de forma simplificada.

O tempo médio de execução do *pipeline* de extração e carregamento foi avaliado ao longo de 31 execuções para dados dos mais variados tamanhos. Para a aceleração o volume de dados por iteração pode chegar a 7000 aquisições para todos os sensores utilizados. O tempo médio de extração e carregamento destes dados no PostgreSQL foi de 49 segundos. Já o *pipeline* de tratamento de dados e geração do relatório teve tempo médio de 59 segundos, tempo superior devido à maior complexidade das operações realizadas. Esse resultado significa que, em média, os engenheiros da equipe têm acesso às informações desejadas em até 2 minutos após a geração do evento na Foxglove.

A dependência da geração manual de eventos na Foxglove é uma limitação identificada na plataforma, pois da forma como a plataforma foi desenvolvida no momento, o engenheiro precisa carregar o arquivo *bag* do ROS 2 em MCAP na Foxglove manualmente. Com a evolução do sistema de aquisição, uma geração automática de eventos na Foxglove pode reduzir ainda mais o tempo desde que o carro parou até o momento que os dados são analisados.

Para provas mais longas, como *endurance*, o volume de dados aumenta consideravelmente. A plataforma, disponibilizada através do computador da Certi, apresentou mais lentidão para carregar as informações, indicando necessidade de otimizações no código ou melhorias na máquina o qual a plataforma está provisionada.

A disponibilização de dados através do Google Drive se mostrou suficiente para análises refinadas e geração de gráficos, porém sua interface não se mostrou agradável do ponto de vista de experiência de usuário quando comparado a ferramentas exclusivas para análise de dados como Power BI, Grafana ou Streamlit. Porém, as informações estarem armazenadas no Postgres tornam a conexão com esses dados facilitadas, permitindo evoluções futuras em formas de disponibilização dos resultados.

A arquitetura modular do DAS permitiu que a equipe criasse um repositório de dados históricos, essencial para análises comparativas em futuras competições. Além disso, a documentação detalhada dos *pipelines* garantiu a sustentabilidade do projeto, com novos membros conseguindo operar a plataforma com 8 horas de treinamento em média.

4.2 Capacidade de Análise

Em termos de potencial de análise, o uso do Python como linguagem de transformação permite uma gama de análises dado o seu ecossistema extensivo. O uso de pacotes como SciPy permite a aplicação de algoritmos avançados para lidar com filtragem de sinais e até preparar os dados para análises avançadas. O pacote Pandas atendeu todas as necessidades de engenharia de dados para o caso da prova de aceleração, porém não é a aplicação otimizada para lidar com alto volume de informações. Apesar do bom desempenho em provas curtas como a de aceleração, testes com o carro em simulações mais longas (e.g., *autocross*) revelaram desafios de escalabilidade. O volume de dados gerado em provas longas (até 10 vezes maior que em aceleração) exigiu otimizações adicionais no PostgreSQL, como a indexação de campos-chave (e.g., *publish time*) e particionamento de tabelas. Essas adaptações reduziram o tempo de consulta em 40%, mas evidenciaram a necessidade de migrar para soluções de *big data* (e.g., Apache Cassandra, PySpark, Polars) em futuras iterações.

As análises fornecidas no relatório de aceleração mostraram resultado positivo quando analisadas com o propósito de entender o comportamento do carro. A análise da razão de escorregamento ao longo do tempo de prova e distância percorrida permitiu identificar melhorias no *setup* do carro para gerar menos escorregamento e, conseqüentemente, mais torque, bem como entender os limites de aceleração, fornecendo *feedback* ao piloto de como otimizar a aplicação de força no pedal de aceleração. Os resultados mostraram baixa aderência no pneu, com alta presença de escorregamento, significando que o carro patinou demais nas primeiras iterações dos testes. Apesar de resultados positivos com a análise da razão de escorregamento, a falta de uma modelagem dinâmica do veículo não permitiu correlacionar a taxa de escorregamento com

as forças presentes no carro e dificultou o processo de entender a transmissão de potência do motor ao pneu.

As análises que relacionam aceleração e velocidade do carro permitiram, de forma simples, entender a capacidade do veículo em traduzir potência em aceleração, contudo, sem a modelagem dinâmica do veículo, não foi possível otimizar a aceleração por meio desta informação, pois idealmente um gráfico que correlaciona torque com a velocidade seria mais eficaz nesta análise.

O uso da aceleração do IMU nas análises, apesar de prejudicado pelo ruído, foi de suma importância para melhorar a calibração dos sensores e garantir a medição precisa da aceleração. O uso do método de diferenciação central apresenta erros que são intrínsecos do algoritmo e não podem ser corrigidos por completo, apenas mitigados. Logo, uma medição precisa da aceleração garante a certeza de estar observando os dados corretamente. O uso do filtro *Butterworth* para a suavização das curvas de aceleração e remoção dos ruídos ajudou na análise de tendência da aceleração, permitindo a compreensão de como o carro se comporta sob aceleração. A análise da Transformada Discreta de Fourier e o método de pesquisa em grade dos melhores parâmetros de filtragem foram cruciais para entender a natureza do ruído, permitindo aos engenheiros uma melhor instalação dos sensores e calibragem.

O relatório de tempo de prova e velocidade final obtida foi essencial para entender como o carro se comportaria em uma prova real da competição, permitindo a comparação com outros carros de outras equipes nas edições passadas da competição FSAE, dando visibilidade ao time quanto a posição que a equipe Ampera Racing obteria na competição. Dessa forma, os engenheiros conseguem otimizar as configurações do carro para obter melhores resultados e, consequentemente, mais pontos na competição.

Os dados utilizados por este trabalho são todos provenientes de simulação, através da plataforma FSDS (do inglês, *Formula Sae Driving Simulator*), que é específico para uso em competições FSAE. Devido ao tempo de construção e entrega deste trabalho e o calendário da equipe Ampera Racing, não foi possível testar e validar a plataforma com os dados reais.

4.3 Custos

A plataforma foi construída usando ferramentas de código aberto como o Mage e o Python, permitindo que as análises sejam obtidas com alta precisão e baixo custo. Quando executado em computadores pessoais, as análises são essencialmente gratuitas, o único custo é o custo de depreciação do computador e eletricidade utilizada pelo mesmo. A adoção destas ferramentas de código aberto representou uma economia estimada de R\$ 15 mil em licenças anuais comparado a soluções comerciais (e.g., MATLAB Data Acquisition Toolbox, National Instruments LabVIEW, MoTeC M1 Series). Essa economia permitiu realocar recursos para a aquisição de sensores adicionais, ampliando as capacidades de análise.

A plataforma da Foxglove, onde os dados brutos dos sensores são armazenados inicialmente, foi concedida como licença educacional para a Ampera Racing durante a temporada de 2025, logo ela não inferiu em custos para a equipe. Essa concessão de licenças ajudou consideravelmente na capacidade da plataforma em gerar resultados com baixo custo, pois esta licença possui um valor alto quando comparado aos padrões de orçamento da equipe.

O uso do computador da CERTI como servidor para o sistema de forma *online* trouxe mais um aspecto de redução de custos para o projeto. Apesar dos problemas de desempenho para provas mais extensas, o computador permitiu a análise compartilhada entre a equipe, evitando uma realidade onde cada engenheiro trabalharia em seu computador pessoal e dificultaria o processo de compartilhamento de informação, bem como a padronização das análises.

4.4 Impacto no Projeto

Este trabalho está sendo finalizado ainda em etapas de testes para a competição de 2025, logo não foi possível avaliar o impacto do DAS na capacidade da equipe na competição. Contudo, a presença do DAS para fornecer *insights* valiosos sobre o carro trouxe para a equipe a possibilidade de revisar o projeto do carro com informações relevantes, permitindo otimizações no projeto ainda em tempo de testes que podem refletir em resultados positivos na competição, seja nas provas práticas quanto nas provas de apresentação do projeto.

Além do aspecto de análise, a plataforma DAS forneceu meios para que a equipe consiga escalar as análises, dando autonomia para os engenheiros adicionarem mais relatórios de forma horizontal, sem que uma análise afete a outra. Essa capacidade de escalabilidade, requisito fundamental desta plataforma, mostrou resultados positivos durante as fases de teste do carro onde a equipe trouxe *feedbacks* para os relatórios apresentados e novas ideias para implementação de novos relatórios.

5 Conclusão

Com a apresentação dos resultados obtidos pela plataforma DAS durante a temporada de 2025 da equipe Ampera Racing e os detalhamento de todos os processos construídos ao longo da criação do DAS, esta seção visa consolidar as considerações finais quanto ao trabalho apresentado.

5.1 Considerações Finais

O DAS demonstrou que soluções não convencionais no âmbito da FSAE podem gerar resultados competitivos. O uso da Foxglove, originalmente voltada para robótica, provou ser um diferencial na padronização de dados, garantindo integridade e reprodutibilidade nas análises – um avanço significativo em relação aos métodos fragmentados de anos anteriores. Da mesma forma, a adoção do PostgreSQL como camada de armazenamento não apenas reduziu custos em comparação com soluções comerciais, mas também permitiu consultas complexas que antes exigiam ferramentas especializadas. Este trabalho mostrou que essa capacidade e funcionalidade de armazenamento foi possível de ser construída com uso baixo de recursos. Essa funcionalidade, apesar de comum em categorias de alto desempenho como a Fórmula 1, com orçamento muito maior quando comparado as equipes de competição universitárias, não é comum neste cenário da FSAE.

O uso do Mage para organizar os *pipelines* de dados foi crucial para organizar os relatórios e análises. Esta evolução, quando comparado aos anos anteriores da Ampera Racing, foi notada, pois permitiu que a equipe evolua o tratamento de dados sem perder a qualidade e sem retrabalho. Nos anos futuros, a plataforma DAS pode continuar sendo utilizada em sua totalidade, ajustando apenas os parâmetros para a nova competição. O uso do Mage também fortifica a característica de escalabilidade e modularidade da plataforma, permitindo adicionar mais *pipelines* paralelamente com novos objetivos de análise.

Apesar de atender a disponibilidade simples dos dados, o uso do Google Drive se mostrou limitado em termos de customização para o usuário do DAS. Para efetuar análises mais avançadas e personalizadas com os dados finais, os usuários precisam conhecer a fundo a linguagem Python para conseguir usufruir do DAS com essa finalidade. Idealmente, usuários menos técnicos e sem conhecimento do DAS deveriam ser capazes de usar os dados finais do relatório para explorar e encontrar mais *insights* através de métodos mais simples.

O DAS não apenas consolidou a cultura voltada para dados (do inglês, *data driven*) na Ampera Racing, mas também estabeleceu um marco para o desenvolvimento de futuros protótipos, provando que soluções de baixo custo e código aberto podem ser tão eficazes quanto

ferramentas comerciais em ambientes de alto desempenho.

A solução proposta se mostrou capaz de atender o objetivo deste trabalho, apresentando uma plataforma e metodologia que permite o manuseio dos dados de forma eficiente. Os dados são coletados e armazenados de forma organizada tanto na FoxGlove quanto no Postgres, facilitando o tratamento e padronização dos processos seguintes de processamento e análise. O tratamento das informações através do Python e orquestrados pelo Mage trouxe organização para a geração dos relatórios, que, por sua vez, permitiram análises que refletem o comportamento do carro e auxiliam na validação e otimização do desempenho do mesmo. A plataforma permite a expansão do tratamento e análise para múltiplas provas e o ecossistema do Python garante o atendimento à todas as necessidades de análise que a equipe venha a ter.

Validou-se a capacidade da plataforma em atender a necessidade de análise de uma prova de aceleração, provendo informações relacionadas à dinâmica veicular do carro como a razão de escorregamento, que são utilizadas para otimizar a aceleração e obter uma melhor colocação na prova da competição. Evidentemente, há espaço para evoluções nas análises mecânicas apresentadas e, dado a capacidade da plataforma de ser atualizada com facilidade e escalabilidade, essas evoluções podem ser facilmente atingidas através de desenvolvimentos complementares e modulares utilizando a plataforma e a metodologia apresentada.

Contudo, pela impossibilidade de efetuar os testes com o carro real em movimento em data prévia à entrega deste trabalho, não foi possível validar sua capacidade com dados reais. Porém, como os dados simulados forma confeccionados de forma a gerar os mesmos dados que o carro real, espera-se similaridade na usabilidade da plataforma e capacidade de análise.

5.2 Próximos Passos

Alguns aspectos limitantes do DAS foram abordados ao longo do trabalho e são consolidados nesta seção. Além disto, dado que o DAS foi avaliado neste trabalho apenas em fase de testes na temporada de 2025, é fundamental que o DAS seja utilizado extensivamente na prática durante a competição de 2025. Com isso, a equipe deve aproveitar a utilização para coletar impressões dos membros da equipe quanto a aplicabilidade do DAS e, futuramente, evoluir o DAS com a análise de capacidades da plataforma.

O uso do computador da CERTI apresenta problemas de desempenho. Idealmente, o DAS deve ser provisionado através de um serviço de *cloud* como a Amazon Web Services (AWS), que possui recursos em nuvem recomendados para manter a qualidade da plataforma. O provisionamento do DAS pode ser disponibilizado através do serviço EC2, que permite a criação de máquinas virtuais. A segurança e controle de acesso da plataforma pode ser feito através da funcionalidade de *Security Group* e serviços como AWS Incognito e AWS Load Balancer. A disponibilização do Mage através de um domínio público ou privado (DNS, do inglês *Domain Name System*) facilita o acesso compartilhado.

A camada de disponibilização se beneficiaria de uma evolução em termos de experiência de usuário. Como um complemento aos dados e gráficos que contemplam o relatório do DAS no seu estado atual, recomenda-se o uso da ferramenta Streamlit para apresentar painéis dinâmicos em formato de aplicativo, permitindo interação do usuário com os dados e visualizações. Essa capacidade traria evoluções significativas para a dificuldade apresentada pelos usuários em termos de personalização dos dados.

Para evoluir ainda mais a resposta dos relatórios aos testes do carro, é fundamental automatizar o carregamento dos dados gravados de ROS 2 e a criação de eventos na plataforma Foxglove. Como o DAS está sensibilizado para executar os *pipelines* automaticamente, se os dados surgissem também de forma automática, toda a sincronização do sistema se completa e a usabilidade dessa plataforma se equiparia aos produtos de *data logger* fornecidos no mercado.

Em termos de análises construídas, o relatório de aceleração deve ser evoluído através do desenvolvimento de uma modelagem mais avançada do sistema, levando em consideração mais aspectos de dinâmica veicular. Conforme apresentado por [Segers \(2008\)](#) e [Rajamani \(2011\)](#), uma modelagem de 14 graus de liberdade (14-DOF) traria informações cruciais sobre rotação das rodas, movimento vertical, *yaw*, *roll*, *pitch*, suspensão independente, movimentação de chassis e até aerodinâmica. Além disso, a interação do pneu com o solo pode ser evoluída através das fórmulas mágicas de Pacejka que relacionam a razão e ângulo de escorregamento com cargas verticais e forças laterais e longitudinais, permitindo uma análise mais avançada do comportamento do carro em curvas.

Este trabalho apresenta apenas o relatório de aceleração, logo uma evolução necessária para a completude do DAS em relação à competição FSAE é a geração dos relatórios para as provas de *endurance*, *skidpad* e *autocross*. Além disso, recomenda-se a criação de relatórios para testes livres e comparação entre simulação e carro real.

Evidentemente, a partir do processo de *shakedown* do veículo, isto é, o processo no qual o carro é pela primeira vez montado e colocado para movimentar-se, espera-se que os sensores sejam instalados corretamente e os dados sejam providenciados via Foxglove para dar continuidade ao trabalho proposto, validando as análises com as informações reais. Neste trabalho, apenas os dados de simulação foram utilizados.

Referências

- AMAZON WEB SERVICES. *O que é containerização?* AWS, 2024. Disponível em: <https://aws.amazon.com/what-is/containerization/>. Acesso em: 04 abr. 2025. 57
- AMAZON WEB SERVICES. *O que é o PostgreSQL?* AWS, 2024. Disponível em: <https://aws.amazon.com/pt/rds/postgresql/what-is-postgresql/>. Acesso em: 04 abr. 2025. 56
- AMAZON WEB SERVICES. *Qual a diferença entre ETL e ELT?* AWS, 2024. Disponível em: <https://aws.amazon.com/pt/compare/the-difference-between-etl-and-elt/>. Acesso em: 04 abr. 2025. 35, 36
- AYYAD, A.; FATHIZADEH, M. Telemetry and data collection to improve formula SAE car. In: *Proceedings of the World Congress on Engineering and Computer Science*. San Francisco: WCECS, 2017. p. 317–321. Disponível em: https://www.iaeng.org/publication/WCECS2017/WCECS2017_pp317-321.pdf. 50
- BOSCH, R. *CAN Specification Version 2.0*. Stuttgart, 2012. Document version 2.0. Disponível em: <http://esd.cs.ucr.edu/webres/can20.pdf>. Acesso em: 20 abr. 2025. 32
- CHAUDHURI, S.; DAYAL, U. An overview of data warehousing and olap technology. *SIGMOD Rec.*, Association for Computing Machinery, New York, v. 26, n. 1, p. 65–74, mar. 1997. Disponível em: <https://doi.org/10.1145/248603.248616>. Acesso em: 02 abr. 2025. 34, 35, 37
- DENSMORE, J. *Data Pipelines Pocket Reference*. Sebastopol: O'Reilly Media, 2021. 33
- GIRAUDO, A. *Performance Optimization, Data Analysis and Simulation of a Formula SAE Electric Racing Car*. Dissertação (Tese de Mestrado) — Politecnico di Torino, Torino, 2019. Relator: Prof. Andrea Tonoli. Disponível em: <https://webthesis.biblio.polito.it/12761/1/tesi.pdf>. 50
- HOLLAND, E. Formula-sae wireless data logger. *Journal of Undergraduate Research at Minnesota State University*, Minnesota, v. 2, p. 1, 2002. Disponível em: <https://cornerstone.lib.mnsu.edu/jur/vol2/iss1/1>. 50
- KARUMBUNATHAN, L. S. *NVIDIA Jetson AGX Orin Series Technical Brief*. [S.l.], 2022. Disponível em: <https://www.nvidia.com/content/dam/en-zz/Solutions/gtc/tcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief.pdf>. Acesso em: 07 jul. 2025. 60
- KELEMENOVA, T.; BENEDIK, O.; KOLARIKOVA, I. Signal noise reduction and filtering. *Acta Mechatronica*, Kosice, v. 5, p. 29–34, 06 2020. 43
- LEE, E. A.; SESHIA, S. A. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. Second. Massachusetts: MIT Press, 2017. ISBN-13: 978-0-262-53381-2. Disponível em: <https://leeseshia.org/>. Acesso em: 15 abr. 2025. 31, 32
- LI, Z.; WANG, D.; KANG, Q. The development of data acquisition system of formula sae race car based on can bus communication interface and closed-loop design of racing car. *Hindawi*, Changchun, ago. 2021. Disponível em: <https://scispace.com/pdf/the-development-of-data-acquisition-system-of-formula-sae-3y9qdjkswc.pdf>. Acesso em: 15 abr. 2025. 27

MILLIKEN, W.; MILLIKEN, D. *Race Car Vehicle Dynamics*. SAE International, 1995. (Premiere Series). ISBN 9781560915263. Disponível em: <<https://books.google.com.br/books?id=opgHfQzlnLEC>>. 47

NIMMALA, R.; NIMMALA, J. Racing into the data age: sensor intelligence, advanced analytics and kafka in formula 1 race car. *International Journal of Artificial Intelligence & Machine Learning (IJAIML)*, Palavakkam, v. 3, Mar. 2024. Disponível em: <https://lib-index.com/index.php/IJAIML/article/view/IJAIML_03_01_006>. Acesso em: 09 abr. 2025. 33, 34, 35

OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-Time Signal Processing*. 3. ed. Upper Saddle River: Prentice Hall, 2010. 45

PARDALIS, K. Part 1: The evolution of data pipeline architecture. *RudderStack*, Mar. 2021. Disponível em: <<https://www.rudderstack.com/blog/part-1-the-evolution-of-data-pipeline-architecture/>>. Acesso em: 27 mar. 2025. 33, 34

PROAKIS, J. G.; MANOLAKIS, D. G. *Digital signal processing: principles, algorithms, and applications*. 3. ed. Upper Saddle River: Prentice Hall, 1996. ISBN 0133737624. 39, 42, 45, 46

RADIANT LOGIC. *PostgreSQL Recommendations*. [S.l.]: Radiant Logic, 2025. Disponível em: <<https://developer.radiantlogic.com/ia/descartes/best-practice/02-databases/01-postgres-recommendations/>>. Acesso em: 13 abr. 2025. 57

RAJAMANI, R. *Vehicle Dynamics and Control*. 2nd. ed. Minneapolis: Springer, 2011. 48, 49, 87

RAJENDRAN, S. et al. Estimation of road frictional force and wheel slip for effective anti-lock braking system (abs) control. *University College London*, London, 2018. Disponível em: <https://discovery.ucl.ac.uk/id/eprint/10059709/1/RNCnew_sks_3_9_2018.pdf>. Acesso em: 15 jun. 2025. 49

SAE BRASIL. *Formula SAE Brasil*. 2004. Programa estudantil da SAE Brasil. Acesso em: 15 jun. 2025. Disponível em: <<https://saebrasil.org.br/programas-estudantis/formula-sae-brasil/>>. 27

SEGBERS, J. *Analysis Techniques for Racecar Data Acquisition*. Warrendale: SAE International, 2008. 33, 46, 47, 48, 49, 87

SEGNER, M. *Data Pipeline Architecture Explained*. 2023. Disponível em: <<https://www.montecarlodata.com/blog-data-pipeline-architecture-explained/>>. Acesso em: 09 abr. 2025. 35

SHAPIRO, J. Data-driven at 200 mph: How analytics transforms formula one racing. *Forbes*, jan. 2023. Disponível em: <<https://www.forbes.com/sites/joelshapiro/2023/01/26/data-driven-at-200-mph-how-analytics-transforms-formula-one-racing/>>. Acesso em: 24 mar. 2025. 27

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Database System Concepts*. 7. ed. New York: McGraw-Hill Education, 2019. ISBN 978-0078022159. 36, 37, 38

SKOG, I.; HÄNDEL, P. Calibration of a mems inertial measurement unit. In: *XVII IMEKO World Congress*. Rio de Janeiro: IMEKO, 2017. p. 1–6. Disponível em: <https://www.imeko.org/publications/wc-2006/PWC-2006-TC3-017u.pdf>. 44

SMITH, S. W. *The Scientist and Engineer's Guide to Digital Signal Processing*. San Diego: California Technical Publishing, 1997. Disponível em: <https://img.anfulai.cn/bbs/97312/The%20Scientist%20and%20Engineer%20Guide%20to%20DSP.pdf>. Acesso em: 01 mai. 2025. 46

VOSS, W. *A Comprehensible Guide to Controller Area Network*. 1st. ed. Copperhill Technologies Corporation, 2005. ISBN 0-9765116-0-6. Disponível em: https://books.google.com.br/books?id=PU6ppO3XbUwC&printsec=frontcover&redir_esc=y#v=onepage&q&f=false. Acesso em: 15 jun. 2025. 32, 33

WALDO, J. Embedded computing and formula one racing. *IEEE Pervasive Computing*, Los Alamitos, v. 4, n. 3, p. 18–21, 2005. Disponível em: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1495385>. Acesso em: 01 mai. 2025. 31

YANG, W.; BAJENOV, A.; SHEN, Y. Improving low-cost inertial-measurement-unit imu based motion tracking accuracy for a biomorphic hyper-redundant snake robot. *Robotics and Biomimetics*, v. 4, n. 1, p. 16, nov 2017. ISSN 2197-3768. Disponível em: <https://doi.org/10.1186/s40638-017-0069-z>. 44