UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

NATANAEL COSTA MATIAS

DESENVOLVIMENTO DE UM SIMULADOR DE STRIP-SENSOR PARA SISTEMA DE PESAGEM DE VEÍCULOS EM MOVIMENTO

CORNÉLIO PROCÓPIO

2025

NATANAEL COSTA MATIAS

DESENVOLVIMENTO DE UM SIMULADOR DE STRIP-SENSOR PARA SISTEMA DE PESAGEM DE VEÍCULOS EM MOVIMENTO

Development of a strip-sensor simulator for a weigh-in-motion system

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Engenharia Elétrica do Curso de Bacharelado em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. Paulo Junior Silva Costa

CORNÉLIO PROCÓPIO

2025

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
Departamento Acadêmico de Elétrica
Engenharia Elétrica

FOLHA DE APROVAÇÃO

Natanael Costa Matias

DESENVOLVIMENTO DE UM SIMULADOR DE STRIP-SENSOR PARA SISTEMA DE PESAGEM DE VEÍCULOS EM MOVIMENTO

Trabalho de conclusão de curso apresentado às 08:20hs do dia 02/12/2025 como requisito parcial para a obtenção do título de Bacharel em Engenharia Elétrica da Universidade Tecnológica Federal do Paraná. O(A) candidato(a) foi arguido(a) pela Banca Avaliadora composta pelos(as) professores(as) abaixo. Após deliberação, a Banca Avaliadora considerou o trabalho aprovado.

_____
Prof(a). Dr(a). Paulo Junior Silva Costa - Presidente (Orientador(a))

_____
Prof(a). Dr(a). Luis Fernando Caparroz Duarte - (Membro)

_____
André Feracin - (Membro)

A folha de aprovação assinada encontra-se na coordenação do curso.

Dedico este trabalho à minha esposa, Layara Mayany, e a meus filhos, Laura, Catarina e Natanael Filho, pelo amor incondicional.

AGRADECIMENTOS

Agradeço ao meu Senhor e Salvador pela graça, amor e cuidado para com minha vida em todos os momentos. Sou grato à minha esposa, Layara pelo carinho, compreensão e amor nesta trajetória que por muitas vezes exigiu tudo dela. Agradeço às meus filhos, Laura, Catarina e Natanael Filho, por cada momento que me dá propósito e pelo amor incondicional de todos os dias. Sem vocês eu nada seria.

Sou grato a meu orientador, Prof. Dr. Paulo Júnior, pela amizade e pelos momentos fora do ambiente acadêmico. Agradeço também aos professores Alessandro Goedtel, Marcelo Favoretto Castoldi e Wesley Angelino de Souza que de igual modo fizeram parte de minha trajetória.

Também agradeço ao André Feracin pela amizade sem igual que fez de mim uma pessoa melhor. Sou grato a todos os professores e amigos que fiz durante minha trajetória na UTFPR. Sempre lembrarei de todos com muito carinho.

[11]Quando eu era menino, falava como menino, pensava como menino e raciocinava como menino. Quando me tornei homem, deixei para trás as coisas de menino. (1 Coríntios 13)

RESUMO

Um sistema de pesagem de veículos em movimento (WIM - Weigh In Motion) representa uma oportunidade de monitoramento eficiente de rodovias para identificação de veículos com sobrepeso, característica essa que representa uma das principais causas de danos ao pavimento e de acidentes em rodovias envolvendo veículos de grande porte, pois o excesso de carga altera as características dinâmicas dos veículos. Para a utilização direta de um sistema WIM na fiscalização de uma rodovia, a medida de peso bruto do veículo, assim como o peso por eixo, deve ser precisa e confiável, o que representa um desafio no desenvolvimento de tal sistema, principalmente devido às características dinâmicas dos veículos em alta velocidade. Neste trabalho será apresentado o desenvolvimento de uma interface de simulação de veículos em movimento. Será utilizado o conversor digital analógico (DAC) de um microcontrolador para reproduzir o sinal de uma célula de carga sobre a qual passou um veículo de grande porte. Este sinal é tratado por um circuito de condicionamento e injetado na interface da balança, simulando a passagem de um veículo sobre os sensores da mesma, ou seja, o simulador fará o papel da célula de carga. O software da balança fará a detecção dos veículos simulados, possibilitando a validação do funcionamento do dispositivo simulador WIM. Os resultados obtidos apresentaram boa reprodutibilidade dos parâmetros medidos, validando assim o funcionamento do sistema.

Palavras-chave: simulação; pesagem em movimento; conversor digital para analógico; condicionamento de sinais.

ABSTRACT

A Weigh-In-Motion (WIM) system represents an opportunity for efficient highway monitoring aimed at identifying overweight vehicles—one of the main causes of pavement deterioration and roadway accidents involving heavy-duty vehicles, since excessive loading alters vehicle dynamic characteristics. For the direct use of a WIM system in highway enforcement, both the gross vehicle weight and the axle loads must be accurate and reliable. Achieving such performance is challenging, particularly due to the dynamic behavior of vehicles at high speeds. This work presents the development of a simulation interface for vehicles in motion. The digital-to-analog converter (DAC) of a microcontroller is employed to reproduce the signal of a load cell over which a heavy vehicle has passed. This signal is processed by a conditioning circuit and injected into the scale interface, thereby simulating the passage of a vehicle over its sensors; in other words, the simulator emulates the behavior of a load cell. The scale's software performs the detection of the simulated vehicles, enabling validation of the WIM simulator's functionality. The results obtained demonstrate good reproducibility of the measured parameters, thus validating the operation of the proposed system.

Keywords: simulation; weigh in motion; digital-analog convertion.

LISTA DE FIGURAS

Figura 1 – *Strip-Sensor* Intercomp. **13**
Figura 2 – Exemplo de sinal para veículo de 5 eixos. **14**
Figura 3 – Zoom aplicado a um grupo de eixos para o caso do exemplo na Figura 2. . . . . . **14**
Figura 4 – Simbolo de circuito de um DAC. **16**
Figura 5 – Conversor simples single-ended para diferencial AD8476. . . . . . **18**
Figura 6 – Protótipo do simulador montado em placa perfurada. **22**
Figura 7 – Interface wimlogix Intercomp. **23**
Figura 8 – Circuito para simulação do laço indutivo. **23**
Figura 9 – Circuito charge-pump com LTC1983-5. . . . . . **24**
Figura 10 – Circuito para conversão de sinal para diferencial. **24**
Figura 11 – Silhueta das classes de veículos simulados. **26**
Figura 12 – Sequência de simulação. **29**
Figura 13 – Forma de onda para o veículo de 3 eixos classe 3C. **30**
Figura 14 – Forma de onda para o veículo de 5 eixos classe 2S3. . . . . . **30**
Figura 15 – Forma de onda para o veículo de 6 eixos classe 3S3. . . . . . **31**
Figura 16 – Forma de onda para o veículo de 7 eixos classe 3D4. . . . . . **31**
Figura 17 – Print da aba de status dos sensores. **32**
Figura 18 – Print da aba de veículos detectados. **32**

LISTA DE ABREVIATURAS E SIGLAS

Siglas

DAC	Conversor Digital para Analógico
DMA	Acesso Direto à Memória
HAL	*Hardware Abstraction Layer*
HSWIM	*High Speed Weigh-In-Motion*
IDE	Ambiente de Desenvolvimento Integrado
ITM	*Instrumentation Trace Macrocell*
WIM	*Weigh-In-Motion*

SUMÁRIO

1 INTRODUÇÃO 11

1.1 Objetivos 12

1.1.1 Objetivo Geral 12

1.1.2 Objetivos Específicos 12

1.2 Estrutura do Trabalho 12

2 REFERENCIAL TEÓRICO 13

2.1 *Strip Sensor* Intercomp 13

2.2 Laço Indutivo 15

2.3 Conversor Digital-Analógico (DAC) 16

2.4 Acesso Direto à Memória - DMA 17

2.5 Conversão de sinal *single-ended* para diferencial 18

2.6 *Instrumentation Trace Macrocell* (ITM) 19

2.7 *Timer* 20

3 METODOLOGIA 22

3.1 Montagem do Protótipo 22

3.1.1 Circuito para laço indutivo 23

3.1.2 Circuito para condicionamento de sinal 24

3.2 Codificação do Firmware 25

3.3 Procedimento de Testes 28

4 RESULTADOS 30

4.1 Análise dos Dados 33

5 CONCLUSÃO 39

REFERÊNCIAS 40

1 INTRODUÇÃO

As rodovias brasileiras são especialmente importantes para o desenvolvimento do país devido a grande participação no transporte de cargas. O setor de transporte rodoviário é responsável por aproximadamente 65% das cargas escoadas e por quase 95% dos passageiros (CNT BRASíLIA, 2024), portanto é fundamental que as rodovias sejam mantidas em bom estado de conservação, para garantir segurança na via e evitar congestionamentos.

Um dos principais fatores que provocam grandes danos ao pavimento é o tráfego de caminhões carregados acima dos limites permitidos, devido à pressão excessiva na superfície da pista, provocando vários tipos de danos à estrutura da rodovia, em pontes e em bueiros, além disso, esses veículos representam risco à segurança da via por serem mais propensos a acidentes de várias naturezas. (SUJON; DAI, 2021).

Por conta disso é essencial a fiscalização adequada do peso dos veículos de grande porte que trafegam nas estradas, para impor o cumprimento das regulamentações de peso e ajudar a preservar a infraestrutura rodoviária (ROCHETI; BACURAU, 2024). Os modos mais tradicionalmente utilizados na medição de peso desses veículos são por meio de balanças estáticas ou de baixa velocidade, que operam em velocidades abaixo de 5 km/h. Embora sejam métodos bastante precisos, eles apresentam pontos inconvenientes como congestionamentos de trânsito, filas extensas e atrasos para os motoristas, tornando esses métodos financeiramente ineficientes (ROCHETI; BACURAU, 2024).

Neste cenário, os sistemas de pesagem em movimento, *Weigh-In-Motion* (WIM), surgem como uma alternativa eficiente para superar os desafios mencionados anteriormente. A tecnologia WIM permite o monitoramento do peso dos veículos enquanto estão em movimento na estrada, eliminando a necessidade de paradas desnecessárias em postos de fiscalização, fornecendo dados em tempo real, sem interromper o fluxo normal da via (SUJON; DAI, 2021).

No entanto, os sistemas de pesagem em alta velocidade (velocidade máxima da pista) conhecidos como *High Speed Weigh-In-Motion* (HSWIM) possuem importantes limitações devido à interação dinâmica entre a estrada e os caminhões. Além disso, para o desenvolvimento adequado de *hardware* e *software* do sistema, os testes em campo apresentam limitações significativas, como interdição de pista, disponibilidade de diferentes classes de veículos e complexidade do sistema para aquisição de dados (JACOB; BEAUMELLE, 2010).

Deste modo, este trabalho apresenta o desenvolvimento de um hardware para simulação de até 4 pares de sensores utilizados em sistemas HSWIM. Serão utilizados conversores digital-analógico para reprodução do sinal dos sensores, e um circuito de condicionamento de sinal será utilizado para ajuste de amplitude e *offset* dos sinais dos sensores, assim como a converção dos sinais *single-ended* para diferencial. Também será implementado a simulação de presença de veículo em laço indutivo, também chamado de *loop*, para indicação de início e fim do veículo. Ao todo serão simulados 4 classes diferentes de veículos, classe 3C, 2S3, 3S3 e 3D4, com mesma velocidade e peso em cada eixo aproximadamente iguais.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um simulador de célula de carga (*strip sensor*) funcional, para gerar os sinais referentes aos sensores em suas respectivas posições no pavimento, assim como os detectores de presença indutivos, para simulação de diferentes classes de veículos a serem pesados pelo sistema HSWIM.

1.1.2 Objetivos Específicos

- Estudar as propriedades dos sensores reais que serão utilizados.
- Gerar *arrays* com valores digitais, com resolução de 12 bits, dos sinais de cada sensor.
- Realizar a conversão de digital para analógico dos sinais.
- Desenvolver *firmware* para STM32 para utilização dos periféricos necessários.
- Desenvolver *hardware* embarcado para a aplicação.
- Validar o funcionamento do sistema utilizando uma interface para leitura dos sinais e processamento dos veículos simulados.

1.2 Estrutura do Trabalho

Este trabalho está estruturado de modo que no Capítulo 2 é apresentado o referencial teórico pertinente ao objetivo do trabalho, abordando os aspectos físicos do sensor utilizado como referência, os periféricos utilizados do microcontrolador e os circuitos para o desenvolvimento do *layout* da placa. No Capítulo 3 é apresentada a metodologia aplicada ao trabalho, descrevendo detalhes do código fonte do *firmware* para o microcontrolador, a forma de desenvolvimento do circuito e aquisição e tratamento dos dados coletados. No Capítulo 4 são apresentados os resultados das análises descritas no capítulo anterior, e no Capítulo 5 são apresentadas as conclusões do trabalho.

2 REFERENCIAL TEÓRICO

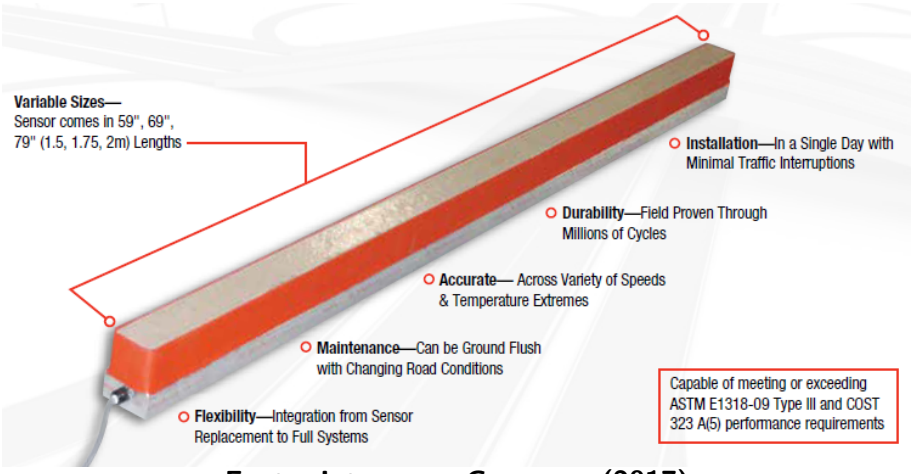
O processo de simulação de um sistema físico é pertinente ao desenvolvimento de equipamentos de alto desempenho, pois permite que sejam feitos testes, melhorias e correções no sistema sem que seja necessário o *setup* completo, que muitas vezes envolve uma grande quantidade de equipamentos em um espaço impraticável no ambiente de laboratório. O sistema, cuja simulação é objetivo deste trabalho, envolve a instalação dos sensores em pavimento adequado, utilização de caminhões com peso conhecido e o procedimento em si de medida de peso dos veículos, tarefa que não pode ser feita de forma simples e que envolve alto custo financeiro.

Neste capítulo será feita uma breve discussão a respeito dos sensores que serão simulados, e dos periféricos utilizados para se obter a geração do sinal equivalente a um veículo passando sobre os sensores.

2.1 *Strip Sensor* Intercomp

O sensor *Strip Sensor* da Intercomp é um transdutor baseado em tecnologia de extensometria (*strain gauge*), desenvolvido para aplicações de pesagem em movimento (*Weigh-In-Motion* – WIM). Seu princípio de funcionamento baseia-se na variação de resistência elétrica de *strain gauges* dispostos em uma ponte de Wheatstone, a qual gera um sinal analógico do tipo mV/V proporcional à deformação causada pela carga aplicada sobre o sensor. Esse sinal, por sua vez, é ratiométrico em relação à tensão de excitação e requer condicionamento, amplificação diferencial e conversão A/D para posterior processamento (Intercomp Company, 2016). A Figura 1 apresenta o sensor da marca Intercomp, o qual será o modelo de referência para geração dos sinais simulados neste trabalho.

Figura 1 – *Strip-Sensor* Intercomp.



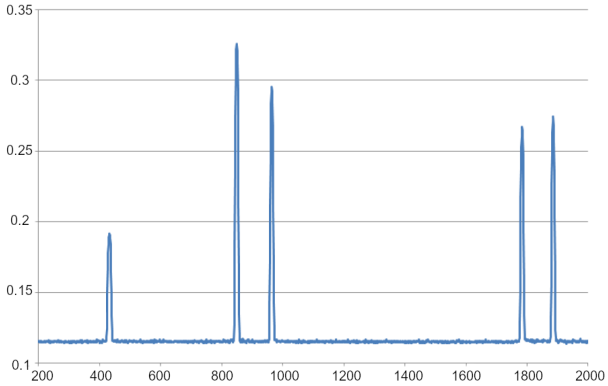
Fonte: Intercomp Company (2017).

O sensor mede exclusivamente a componente vertical da força exercida pelos eixos dos

veículos, sendo imune a carregamentos laterais e variações de posicionamento do pneu sobre sua extensão. Ele apresenta linearidade e histerese inferiores a 0,2 % do fundo de escala, com saída nominal típica de 1 mV/V para uma carga de 8000 lb e resistência de ponte entre 150 Ω e 205 Ω. Além disso, o sistema possui compensação interna de temperatura, garantindo estabilidade e reduzindo a necessidade de recalibração quando comparado a sensores piezoelétricos (Intercomp Company, 2017).

Um exemplo de sinal de saída proporcional ao peso sobre o sensor é apresentado na Figura 2, onde pode ser observado a relação de sinal mV/V no eixo vertical e quantidade de amostras no eixo horizontal. Na Figura 3 foi aplicado zoom sobre um grupo de eixos para melhor visualizar a forma de onda produzida por um eixo do veículo sobre o sensor.

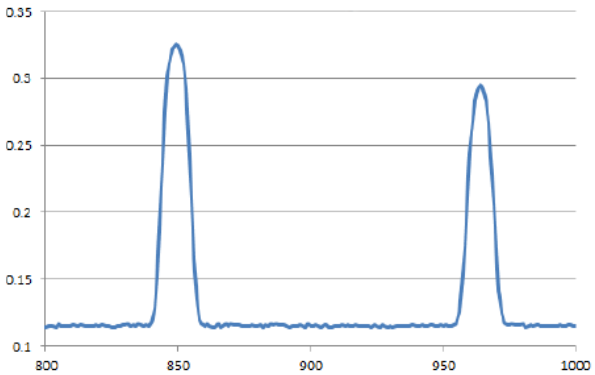
Figura 2 – Exemplo de sinal para veículo de 5 eixos.



Fonte: Intercomp Company (2016).

É possível observar na Figura 3 que a forma de onda para cada eixo do veículo pode ser modelada por uma função gaussiana, onde a amplitude deve obedecer a relação mV/V para o fundo de escala, e a largura, definida pelo desvio padrão da gaussiana, possui relação com a velocidade e o peso do veículo. Esta modelagem foi utilizada para gerar os vetores contendo os dados que seriam captados por cada par de sensores no pavimento, para serem salvos na memória do microcontrolador, e por fim, serem transmitidos para a CPU por meio dos canais de Conversor Digital para Analógico (DAC).

Figura 3 – Zoom aplicado a um grupo de eixos para o caso do exemplo na Figura 2.



Fonte: Intercomp Company (2016).

Os pulsos foram gerados utilizando o método `signal.gausspulse` da biblioteca `scipy`, da linguagem Python, a Listagem 1 apresenta o código Python para gerar o pulso na forma de onda gaussiana. A saída do método é normalizada de 0 a 1, então foi posteriormente tratada para valores de 12 bits, que é a máxima resolução dos canais DAC do STM32G474.

Listagem 1 – Geração do pulso gaussiano usado na simulação.

```
1     def generate_gaussian_pulse(numberOfSamples:int) -> np.array:
2       """
3       Generates a Gaussian pulse signal array ranging from 0.0 to 1.0
4       :param int numberOfSamples: Number of samples available for the pulse.
5       Will be the size of the array
6       :return: array with pulse signal
7       :rtype: np.array
8       """
9       assert numberOfSamples > 0, "Incorrect numberOfSamples"
10      t = np.linspace(-1, 1, numberOfSamples)
11      sig = signal.gausspulse(t, fc=3, retquad=True, retenv=True)[2]
12      sig[np.argmax(sig)] = 1.0 # Forcing the highest pulse value to be 1
13      return sig
14
15
```

Fonte: Autoria própria (2025).

2.2 Laço Indutivo

Os laços indutivos são amplamente utilizados em sistemas de detecção veicular, principalmente em aplicações de controle de tráfego, como semáforos inteligentes, portões automáticos e sistemas de pedágio eletrônico. Esses sensores funcionam com base na variação das propriedades eletromagnéticas de uma bobina instalada sob o pavimento, sendo uma tecnologia madura, confiável e de custo relativamente baixo para detecção de presença e passagem de veículos (COIFMAN; KIM, 2006).

Essencialmente, em uma espira ou conjunto de espiras de fio condutor enterrado em formato retangular ou quadrado sob o pavimento, conectado a um circuito oscilador eletrônico formam o detector de veículos. Esse circuito forma um oscilador LC, no qual a indutância do laço e a capacitância associada determinam a frequência de oscilação. Quando um veículo contendo material metálico (principalmente ferroso) passa sobre o laço, ocorre uma variação na indutância devido à interação do campo magnético da bobina com as massas metálicas do veículo. Essa alteração de indutância causa uma mudança mensurável na frequência do oscilador, permitindo que o sistema identifique a presença ou passagem de um veículo (KLEIN; MILLS; GIBSON, 2006).

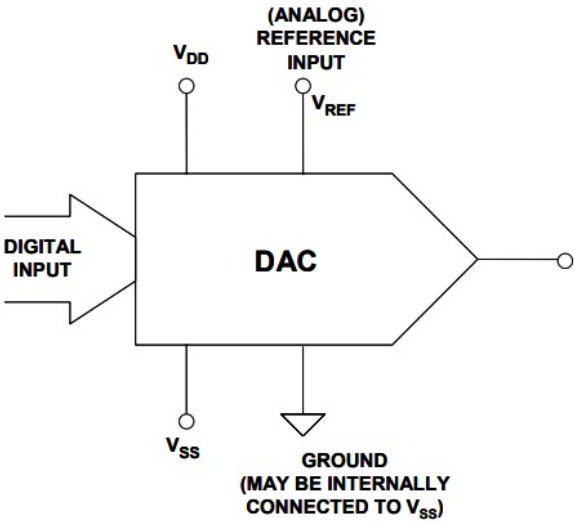
Para o propósito deste trabalho é necessário apenas que o sinal de presença do veículo simulado seja aplicado ao circuito oscilador da interface de processamento HSWIM. Os padrões

NEMA para detectores de circuitos de laço indutivo estabelecem que a unidade eletrônica contendo o oscilador para o laço deve operar de forma satisfatória na faixa de valores de indutância entre 50 e 700 µH (KLEIN; MILLS; GIBSON, 2006), sendo escolhido o valor de 100 µH para acionar o detector de presença de veículos, que será descrito em mais detalhes posteriormente.

2.3 Conversor Digital-Analógico (DAC)

Os conversores digital-analógicos (DAC) são utilizados na transformação de dados transmitidos ou armazenados, ou os resultados de processamento digital, para variáveis de controle de tensão ou corrente analógicos, exibição de informações ou para processamento analógico posterior, proporcionais a uma determinada entrada digital. O valor de saída de um DAC é determinado de acordo com sua resolução e o valor máximo na saída, também chamado de saída de fundo de escala, é dado por um valor de referência que pode ser interno ou externo ao DAC. A Figura 4 abaixo apresenta um símbolo genérico para um DAC (TOCCI; WIDMER; MOSS, 2011).

Figura 4 – Simbolo de circuito de um DAC.



Fonte: Tocci, Widmer e Moss (2011).

A resolução de um DAC é definida como sendo o menor valor possível que pode ser obtido em sua saída, como resultado de uma entrada digital. O valor na saída analógica é diretamente proporcional à resolução do DAC, e pode ser expresso pela equação 1

$$\text{anolog output} = K \times \text{digital input}, \tag{1}$$

em que K representa a resolução do DAC, e é definida como:

$$K = \frac{A_{fs}}{2^n - 1} \qquad [\text{V}],$$

em que A_{fs} é saída de fundo de escala e n é o numero de bits na entrada digital. Deste modo, quanto maior o número de bits, mais refinada será a resolução, fazendo com que haja uma faixa mais ampla de valores possíveis na saída (TOCCI; WIDMER; MOSS, 2011).

Dentre as possíveis aplicações dos conversores D/A, a que tem especial relevância para este trabalho é a possibilidade de reconstrução de sinais previamente digitalizados, ou seja, pontos sucessivos de um determinado sinal são convertidos para seu equivalente digital e armazenados em memória. Cada valor digital alimenta a entrada do DAC, que então converte o dado digital para seu correspondente analógico, um ponto por vez, produzindo desta forma um sinal muito próximo ao sinal original. É importante considerar a taxa na qual o sinal original foi digitalizado, pois a taxa de conversão do DAC deve ser igual a esta, caso contrário, o sinal na saída do DAC será diferente do original em relação ao parâmetro de tempo, ou seja, o sinal será "comprimido" ou "esticado" no eixo horizontal (TOCCI; WIDMER; MOSS, 2011).

2.4 Acesso Direto à Memória - DMA

A entrada/saída (E/S) de dados orientada por *polling* ou interrupção concentra a transferência dos dados entre o processador e os dispositivos de E/S. Isso acrescenta um *overhead* significativo à execução do programa, afetando fortemente o desempenho e a eficiência do mesmo, porque muitas instruções devem ser executadas para cada palavra de dado transferida. Para transferir grandes blocos de dados em alta velocidade pode ser utilizada uma abordagem alternativa, utilizando o periférico de acesso direto à memória (*direct memory access* - DMA) (STMICROELECTRONICS, 2025).

O periférico de DMA é um módulo de barramento avançado que permite a transferência de dados em segundo plano, sem a intervenção do processador, garantindo que a transmissão de dados ocorra sem bloquear a execução do programa principal. Essa transferência de dados pode ocorrer de um periférico para a memória, da memória para um periférico, de memória para memória, ou de periférico para periférico, a depender de cada aplicação. A utilização de DMA melhora o fluxo de transferência de dados dos periféricos, reduzindo o consumo de energia pelo sistema, pois permite que a CPU opere em modo de baixo consumo (STMICROELECTRONICS, 2025).

O STM32G474 possui 2 dispositivos DMA, contendo 16 canais ao todo, cada um dedicado a gerenciar as requisições de um ou mais periféricos, de forma independente. A utilização deste periférico será fundamental para o correto funcionamento do dispositivo pois os sinais que correspondem a cada sensor devem ser gerados independentes da execução do programa, ou seja, de forma não bloqueante (STMICROELECTRONICS, 2025), pois a

simulação precisará replicar o caso em que mais de um sensor é acionado simultaneamente, por isso será utilizado um canal de DAC para simular cada sensor separadamente.

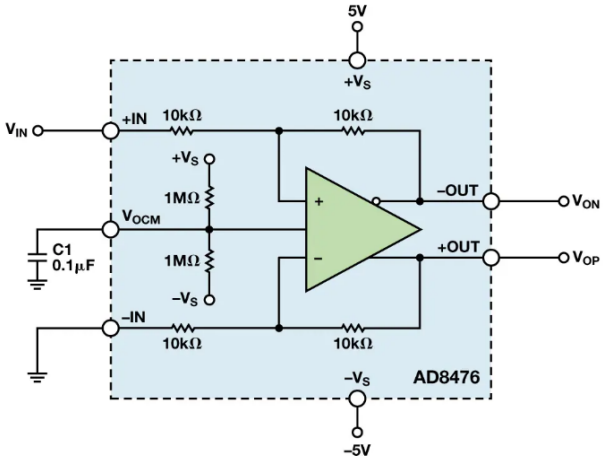
2.5 Conversão de sinal *single-ended* para diferencial

Um conversor *single-ended* para diferencial tipicamente recebe um único sinal de entrada (com referência a 0 V ou a algum nível comum) e gera dois sinais de saída: V_{OP} e V_{ON}, que variam em oposição de fase (idealmente 180° defasados) e representam o mesmo conteúdo de informação (ex: $V_{OP} - V_{ON} \propto V_{\text{in}}$). Além disso, pode haver ajuste do nível de modo comum de saída (*output common-mode*, V_{OCM}) para que o par diferencial fique centrado em um valor compatível com a entrada do ADC ou estágio seguinte(TOLENTINO; HERRERA, 2015).

Sinais diferenciais encontram aplicações úteis em circuitos onde se deseja alta relação sinal-ruído, alta imunidade a ruído de modo comum e menor distorção de segunda harmônica (TOLENTINO; HERRERA, 2015). Algumas das principais aplicações deste tipo de sinal são em circuitos de condicionamento de sinais de áudio de alta fidelidade, acionamento de conversores analógico-digitais, e transmissão de sinais por cabos de par trançado, utilizados no padrão RS-485, por exemplo (HERRERA; GERSTENHABER, 2012).

Neste trabalho foi utilizado o CI AD8476, um amplificador diferencial de precisão, com resistores de ganho internos *laser-trimmed* para ganho unitário, projetado para conversão *single-ended-to-differential* ou *differential-to-differential*, com baixo consumo de corrente e compatível com ADCs de alta resolução.

Figura 5 – Conversor simples single-ended para diferencial AD8476.



Fonte: Herrera e Gerstenhaber (2012).

Algumas das principais características do AD8476 são a corrente de alimentação de cerca de 330 µA, a faixa de tensão de alimentação única de 3 V a 18 V (ou alimentação dupla de $\pm$1.5 V a $\pm$9 V), tensão de *offset* de saída típica de 200 µV máxima, nível de ruído 39 nV/$\sqrt{\text{Hz}}$, distorções harmônicas muito baixas (HD2 = –126 dB a 10 kHz, HD3 = –128 dB a 10 kHz). Sua arquitetura inclui malha de realimentação diferencial e malha de modo

comum, permitindo que o pino V_{OCM} defina o nível de modo comum de saída ou, se deixado sem conexão, o nível se auto-regule para metade da tensão de alimentação interna.

Como a tensão de referência para os sinais gerados pelos canais de DAC do STM32G474 será de 3 V, e a amplitude dos sinais será da ordem de mV, optou-se em utilizar um valor de 1,5 V no pino V_{OCM} para evitar a utilização de tensões negativas na alimentação do AD8476, o que poderia elevar o nível de ruído na saída do CI, devido a conversores chaveados ou *charge-pump*, além de evitar o uso de alimentação simétrica no circuito.

2.6 *Instrumentation Trace Macrocell* (ITM)

O periférico *Instrumentation Trace Macrocell* (ITM) presente nos processadores ARM Cortex-M permitem o rastreamento por *software* para depuração e disgnóstico, gerando pacotes de rastreamento de eventos do sistema operacional e da aplicação. O ITM possibilita a escrita por *software* diretamente nos registradores para a emissão de pacotes, possibilitando realizar a depuração no estilo printf. Os pacotes emitidos pelo ITM são enviados para o periférico TPIU (Unidade de Interface de Porta de Rastreamento). O formatador do TPIU adiciona alguns pacotes extras e, em seguida, envia a sequência completa de pacotes para o host depurador (STMICROELECTRONICS, 2025).

Um dos principais usos do ITM é suportar a saída de mensagens de depuração utilizando a função printf(). Para isso, o ITM contém 32 portas de estímulo, permitindo que diferentes processos de software enviem mensagens para diferentes portas (YIU, 2014). As mensagens podem ser separadas posteriormente no *host* de depuração. Cada porta pode ser habilitada ou desabilitada pelo registrador *Trace Enable* e pode ser programada (em grupos de oito portas) para permitir ou proibir que processos de usuário (sem privilégios) gravem nela. Ao contrário da saída de texto baseada em UART, o uso do ITM para a saída não causa muito atraso para a aplicação. Um *buffer* FIFO é usado dentro do ITM, permitindo que as mensagens de saída de gravação sejam armazenadas em *buffer*. No entanto, ainda é necessário verificar se o FIFO está cheio antes de gravar nele (YIU, 2014).

As mensagens de saída podem ser coletadas na interface da porta de rastreamento ou na interface *Serial Wire Viewer* (SWV) na TPIU. Não há necessidade de remover o código que gera as mensagens de depuração do código final, pois se o bit de controle TRCENA estiver baixo, o ITM ficará inativo e as mensagens de depuração não serão geradas. Você também pode ativar a mensagem de saída em um sistema "ativo" e usar o registro *Trace Enable* no ITM para limitar quais portas são ativadas para que apenas algumas das mensagens possam ser emitidas (YIU, 2014).

A Listagem 2 apresenta a implementação da configuração do periférico ITM para o processador ARM Cortex-M3 ou M4. É possível observar no código que não há configuração direta do pino que contém a função SWO. Isso se dá pelo fato de o pino (PB3 no caso STM32G474) já estar configurado para este periférico em seu estado de *reset*.

Listagem 2 – Configuração do ITM para mensagens de debug.

```
1  // Implementação do periférico ITM
2
3  // Debug Exception and Monitor Control Register
4  #define DEMCR      *((volatile uint32_t *) 0xE000EDFCUL)
5
6  // Enable ITM features for each Stimulus Port
7  #define ITM_TRACE_EN    *((volatile uint32_t *) 0xE0000E00)
8
9  // Select Stimulus Port
10 #define ITM_STI_PORT0   *((volatile uint32_t *) 0xE0000000)
11
12 void SendChar(char *ch) {
13   DEMCR |= (1 << 24);      // TRCENA_bit: habilita o periférico ITM
14   ITM_TRACE_EN |= (1 << 0);    // Habilita o registrador ITM_STIM 0
15
16   // FIFOREADY_bit: aguarda indicação de que o buffer pode aceitar dados.
17   while(!(ITM_STI_PORT0 & 1));
18
19   ITM_STI_PORT0 = (uint32_t) *ch; // Escreve um byte no buffer
20 }
21
```

Fonte: Autoria própria (2025).

2.7 *Timer*

Os *Timers* (temporizadores) são blocos fundamentais em microcontroladores, responsáveis por realizar medições de tempo, geração de sinais periódicos, contagem de eventos externos e controle de periféricos como PWM, captura de entrada e geração de interrupções temporizadas sem a intervenção direta da CPU (STMICROELECTRONICS, 2021).

Essencialmente, o *timer* é um contador que incrementa (ou decrementa) seu valor em função de um sinal de *clock*. O valor atual do contador é armazenado em um registrador denominado *Counter Register*. O incremento do contador ocorre a cada ciclo de *clock* dividido por um *prescaler* configurável, o que permite ajustar a frequência efetiva de contagem. A taxa de incremento do contador é dada por:

$$f_{\text{timer}} = \frac{f_{\text{clock}}}{PSC + 1} \qquad [\text{Hz}]$$

onde f_{clock} é a frequência do *clock* do barramento (por exemplo, APB1 ou APB2), e PSC é o valor do *prescaler* configurado no registrador `TIMx_PSC` (STMICROELECTRONICS, 2021).

Quando o valor do contador atinge o valor definido no registrador de *auto-reload* (`TIMx_ARR`), ocorre um estouro (*overflow*) e o contador é reiniciado a partir de zero (ou do valor inicial configurado). Este evento pode ser utilizado para gerar uma interrupção periódica,

permitindo que o processador execute uma rotina de serviço (ISR) em intervalos regulares de tempo (STMICROELECTRONICS, 2021). Esse mecanismo é essencial para aplicações que necessitam de tarefas periódicas, como controle em malha fechada, aquisição de dados em intervalos regulares, ou geração de sinais de referência temporais (YIU, 2014). O intervalo de tempo no qual ocorrerá o *overflow* do *timer* pode ser determinado por:

$$t_{\text{overflow}} = (ARR + 1) \times \frac{1}{f_{\text{timer}}} \qquad \text{s}$$

Neste trabalho são utilizados dois *timers* para fins distintos. Um *timer*, TIM3 é utilizado para gerar uma interrupção a cada 6 segundos, e na ISR é realizado a simulação de um dos 4 tipos de veículos determinados. O outro *timer*, TIM2 é configurado de modo a servir como gatilho para a conversão D/A, realizando uma conversão em intervalos de $\approx 208\,\mu\text{s}$, correspondendo a uma frequência de $\approx 4800\,\text{Hz}$.

3 METODOLOGIA

Para o desenvolvimento do simulador de células de carga, utilizados em sistemas de pesagem de veículos em alta velocidade (HSWIM), foi montado um protótipo com os circuitos (descritos posteriormente) em uma placa perfurada, em que foi posicionado duas barras de pinos para utilizar um kit de desenvolvimento NUCLEO STM32G474, que contém o microcontrolador utilizado no projeto. Foram utilizados somente os pinos referentes aos conversores digital-analógico do microcontrolador STM32G474ZIT6.

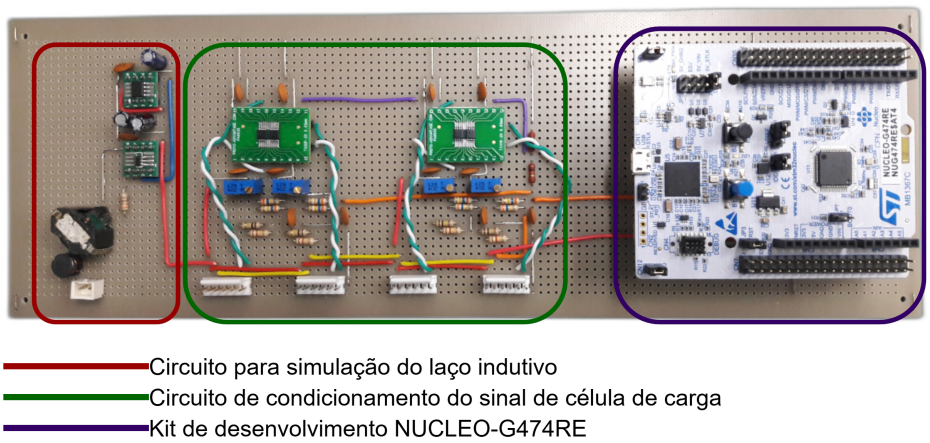
Para evitar complexidades na elaboração do código para o simulador, escolheu-se utilizar a biblioteca *Hardware Abstraction Layer* (HAL), onde é possível configurar os periféricos do microcontrolador por meio da interface gráfica CubeMX integrada ao Ambiente de Desenvolvimento Integrado (IDE) STM32CubeIDE, direcionando o foco do desenvolvimento para a implementação propriamente dita. A codificação da configuração dos periféricos do microcontrolador foi feita de forma automática pelo IDE.

Após a finalização da montagem do protótipo, o mesmo foi conectado a uma interface WIMLOGIX da marca Intercomp, fornecida pela empresa Fiscaltech, para realizar a validação do sistema de simulação dos sensores. Utilizando o software Wim_Calibration da Intercomp, foi possível visualizar a detecção dos veículos pela interface, assim como obter os dados da simulação para verificar o grau de repetibilidade das características de cada veículo simulado. Estes dados foram analisados utilizando um *script* escrito em Python para tratar os dados e plotar a distribuição de velocidade e peso medido em cada ciclo de simulação.

3.1 Montagem do Protótipo

A Figura 6 apresenta o estado final da montagem da placa, onde cada bloco de circuito está destacado por um retângulo de cor diferente.

Figura 6 – Protótipo do simulador montado em placa perfurada.



Fonte: Autoria própria (2025).

A tensão de alimentação do circuito é fornecida pela interface wimlogix (apresentada na Figura 7) por meio do cabo de conexão com o sensor. Originalmente essa tensão é utilizada para alimentar as células de carga.

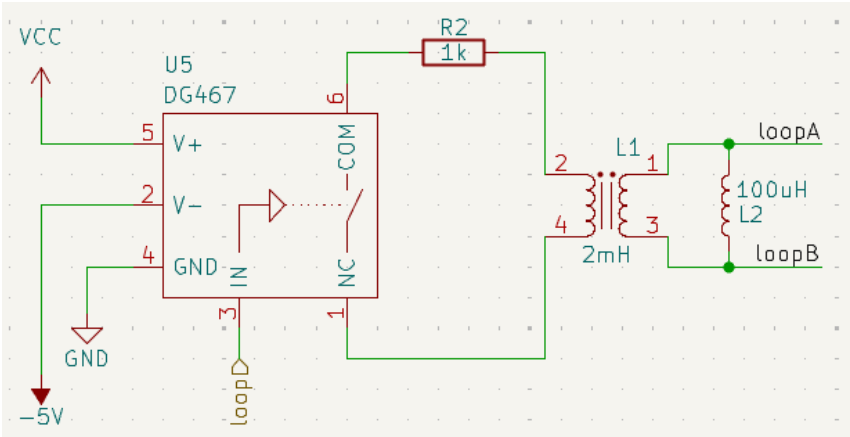
Figura 7 – Interface wimlogix Intercomp.

Fonte: Autoria própria (2025).

3.1.1 Circuito para laço indutivo

Iniciando pelo circuito de simulação de laço indutivo, a Figura 8 apresenta o circuito que foi utilizado para gerar a variação de frequência do oscilador da CPU, indicando a presença de veículo.

Figura 8 – Circuito para simulação do laço indutivo.

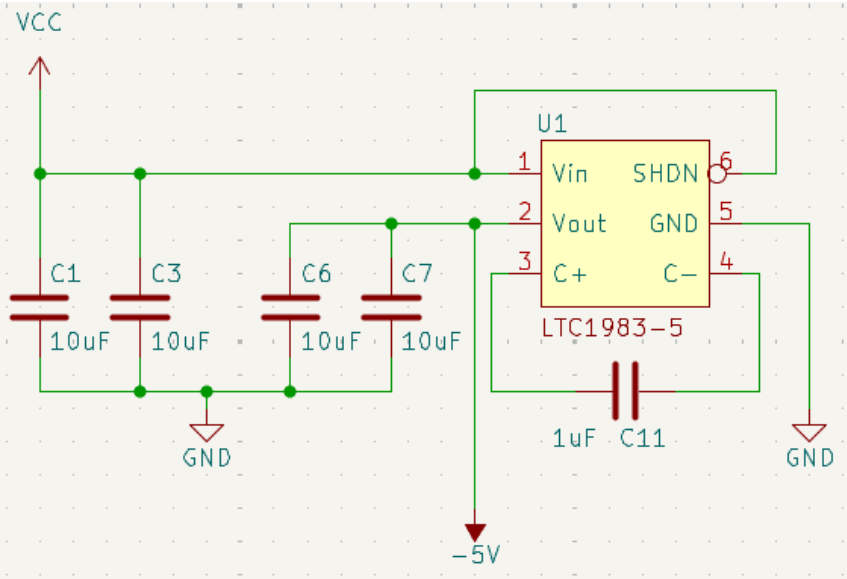


Fonte: Autoria própria (2025).

O CI DG467 é uma chave analógica normalmente aberta, e é utilizada para que o secundário do transformador $L1$ esteja em circuito aberto. O primário, conectado em paralelo ao indutor $L2$, gera uma indutância equivalente de $\approx 100\,\mu H$. Os pontos `loopA` e `loopB` são conectados ao oscilador da interface CPU da balança (conector verde da figura 7), fazendo com que o detector de presença da CPU identifique o laço indutivo, semelhantemente ao qual é utilizado no pavimento. Ao ser enviado um sinal lógico para o pino 3 da chave analógica, o circuito do secundário do transformador é fechado fazendo com que a indutância equivalente do circuito seja alterada, modificando a frequência do oscilador, pois o mesmo é um circuito oscilador LC, o que gera a detecção de um veículo sobre o laço.

A tensão de 5 V negativa é gerada pelo CI *charge pump* LTC1983-5, usado especificamente para o circuito de simulação do laço. A Figura 9 apresenta o circuito para gerar a tensão negativa utilizada no CI de DG467.

Figura 9 – Circuito charge-pump com LTC1983-5.

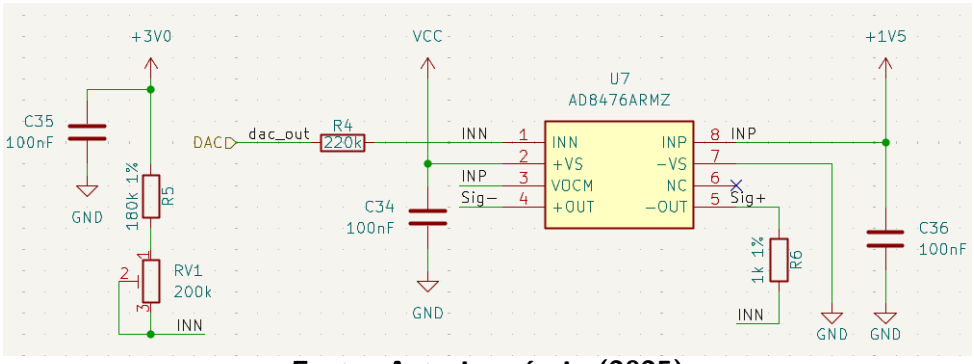


Fonte: Autoria própria (2025).

3.1.2 Circuito para condicionamento de sinal

O circuito para condicionamento do sinal das células de cargas, gerado pelo DAC do STM32, é apresentado na Figura 10. O CI AD8476 recebe um sinal *single-ended* na entrada INN (com INP normalmente ligada a um nível de referência, nesse caso a 1,5 V), e gera nas saídas +OUT e -OUT um sinal diferencial com modo comum definido por V_{OCM} (também conectado a referência de 1,5 V). Isso permite alimentar o ADC diferencial logo em seguida. A tensão de referência de 1,5 V é gerada por uma saída analógica do MCU devidamente configurada para gerar este valor também utilizando o DAC.

Figura 10 – Circuito para conversão de sinal para diferencial.



Fonte: Autoria própria (2025).

É feita uma realimentação negativa com um resistor de precisão de 1 kΩ, fazendo com que o sinal, que no pino do microcontrolador está com fundo de escala em 3 V, passe a ter o fundo de escala na ordem de mV. Mais precisamente, considerando a realimentação interna do AD8476 para ganho unitário e os resistores R_4 e R_6 na Figura 10, o ganho nessa etapa do

circuito passa a ser dado por:

$$\frac{\texttt{Sig+}}{\texttt{DAC}} = \frac{1}{1 + \dfrac{R_4}{R_A}}, \tag{2}$$

com

$$R_A = \frac{R_i R_6}{2R_i + R_6},$$

onde R_i é o resistor de realimentação interno do AD8476 que pode ser visto na Figura 5. Substituindo os valores dos resistores do circuito e do CI, o ganho desta malha do circuito passa a ser de $\approx 2{,}16\,\text{mV/V}$, tornando o sinal proporcional ao sinal da célula de carga real.

O potenciômetro RV1 é utilizado para fazer ajustes de offset no sinal do DAC, onde a tensão de 3 V é gerada pelo kit de desenvolvimento. Finalmente, a tensão de 1,5 V foi obtida por um divisor resistivo com dois resistores de mesmo valor, cuja tensão utilizada foi também os 3 V fornecidos pela placa NUCLEO.

3.2 Codificação do Firmware

Os periféricos do STM32 que foram utilizados para o desenvolvimento deste trabalho foram os canais de DAC e os canais correspondentes para Acesso Direto à Memória (DMA). Os sinais digitais referentes à cada célula de carga foram armazenados em vetores de elementos inteiros e sem sinal, de 16 bits. Para evitar problemas com relação ao armazenamentos dos dados nos vetores, os mesmos foram declarados no programa de modo que na compilação os dados fossem definidos como constantes, para que assim, fossem salvos na memória flash e não na memória RAM, como indicado na Listagem 3.

Listagem 3 – Declaração dos vetores.

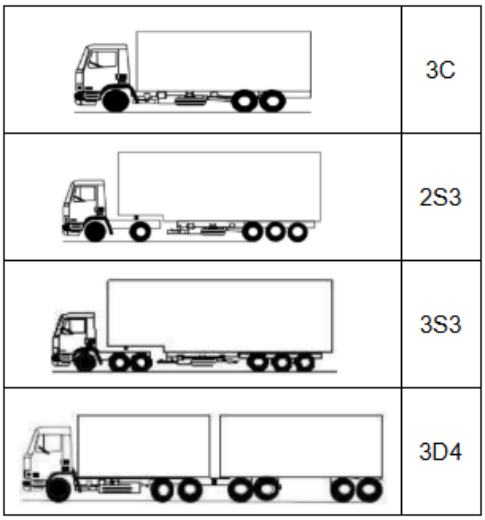
```
1   #include <stdint.h>
2
3   const uint16_t _stripSensor1_3C[TAMANHO_ARRAY_3C];
4   ...
5
6   const uint16_t _stripSensor1_2S3[TAMANHO_ARRAY_2S3];
7   ...
8
9   const uint16_t _stripSensor1_3S3[TAMANHO_ARRAY_3S3];
10  ...
11
12  const uint16_t _stripSensor1_3D4[TAMANHO_ARRAY_3D4];
13  ...
14  const uint16_t _loop_3D4[TAMANHO_ARRAY_3D4];
15
```

Fonte: Autoria própria (2025).

Os três pontos indicam a declaração dos outros pares de sensores para cada tipo de

veículo e `_loop...` representa o sinal do laço indutivo para cada tipo de veículo. Os tipos de veículos utilizados como referência para a simulação seguem a classificação descrita pelo Departamento Nacional de Infraestrutura e Transporte (DNIT) segundo a distribuição de eixos. A Figura 11 apresenta a silhueta de cada veículo usado como referência para a simulação.

Figura 11 – Silhueta das classes de veículos simulados.



Fonte: Autoria própria (2025).

Para se criar um veículo na simulação, foi definido uma estrutura com os campos necessários para utilização do DAC, e na declaração de cada veículo foi montado um vetor de tipo da estrutura contendo a inicialização de cada campo correspondentes ao veículo construído. A Listagem 4 apresenta a forma de declaração de um veículo.

Listagem 4 – Estrutura para construção de um veículo.

```
1   #define NUM_SENSOR      5
2   #define NUM_VEHICLES    4
3
4   typedef struct{
5     DAC_HandleTypeDef     *pDAC;
6     uint32_t               channel;
7     const uint16_t        *pDATA;
8     uint32_t               arrayLength;
9   }Sensor_TypeDef;
10
11  Sensor_TypeDef Vehicle_3C[NUM_SENSOR] = {
12    {&hdac1, CHANNEL_1, _stripSensor1_3C, TAMANHO_ARRAY_3C},
13    {&hdac1, CHANNEL_2, _stripSensor2_3C, TAMANHO_ARRAY_3C},
14    {&hdac2, CHANNEL_3, _stripSensor3_3C, TAMANHO_ARRAY_3C},
15    {&hdac3, CHANNEL_4, _stripSensor4_3C, TAMANHO_ARRAY_3C},
16    {&hdac4, CHANNEL_5, _loop_3C,         TAMANHO_ARRAY_3C}
17  };
18  ...
19
```

Fonte: Autoria própria (2025).

Os três pontos indicam a declaração dos demais veículos. E para rodar a simulação de cada veículo, foi utilizado a interrupção por estouro do timer, do periférico TIM3. Para simplificar a simulação dos veículos de forma circular, foi declarado um vetor de ponteiros para o tipo definido `Sensor_TypeDef`. A Listagem 5 apresenta a declaração, assim como a utilização do vetor na função de DAC por DMA da biblioteca HAL.

Listagem 5 – Vetor de veículos.

```
1     Sensor_TypeDef* Vehicles_HSWIM[NUM_VEHICLES] = {Vehicle_3C, Vehicle_2S3,
2       Vehicle_3S3, Vehicle_3D4};
3
4     void HSWIM_Start(const Sensor_TypeDef* pD){
5       for (int i = 0; i < NUM_SENSOR; ++i)
6         HAL_DAC_Start_DMA(pD[i].pDAC, pD[i].channel, pD[i].pDATA,
7           pD[i].arrayLength, ALIGN12R);
8     }
9
```

Fonte: Autoria própria (2025).

Deste modo, ao acontecer o estouro do timer 3, configurado para 6 segundos, a rotina de interrupção será executada chamando a função que aciona o DAC enviando o veículo correspondente, imprime uma mensagem para debug e incrementa o índice do vetor de veículos, como mostrado na Listagem 6.

Listagem 6 – Rotina de interrupção.

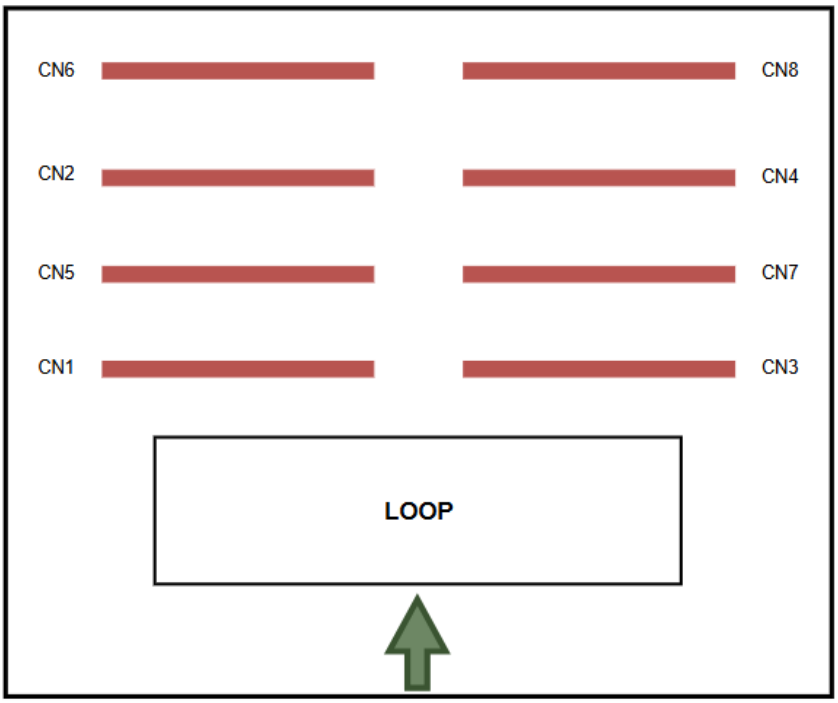
```
1     void TIM3_Callback(void){
2       HSWIM_Start(DACs_HSWIM[id]);
3       PrintStatus();
4       id = (id + 1)%NUM_VEHICLES;
5     }
6
7     void PrintStatus(void) {
8       switch(id){
9         case 0:
10        printf("*****************************\r\n");
11        printf("1' Vehicle: 3C\r\n");
12        break;
13        case 1:
14        printf("*****************************\r\n");
15        printf("2' Vehicle: 2S3\r\n");
16        break;
17        case 2:
18        printf("*****************************\r\n");
19        printf("3' Vehicle: 3S3\r\n");
20        break;
21        case 3:
22        printf("*****************************\r\n");
23        printf("4' Vehicle: 3D4\r\n");
24        break;
25      }
26      printf("Speed: 90 km/h\r\n");
27    }
28
```

Fonte: Autoria própria (2025).

3.3 Procedimento de Testes

Imaginando-se a situação idealizada na qual cada eixo do veículo possuirá o mesmo peso em cada roda, o simulador foi desenvolvido com cada canal de DAC representando um par de sensores posicionados no pavimento, nos quais os rodados dos lados esquerdo e direito estarão simultaneamente sobre os sensores. A Figura 12 apresenta um esboço de como deve ser realizada a sequência de sinais para que a simulação seja processada corretamente pela interface de processamento, onde a seta em verde indica o sentido que o veículo passaria sobre os sensores.

Figura 12 – Sequência de simulação.



Fonte: Autoria própria (2025).

Como cada rodado de um determinado eixo do veículo possuirá o mesmo peso, o sinal proveniente de um DAC pode ser replicado e aplicado a dois sensores na mesma linha, por exemplo, CN1 e CN3 da Figura 12.

4 RESULTADOS

As figuras 13 a 16 apresentam as formas de onda nas saídas dos canais DAC do microcontrolador. As escalas do osciloscópio foram ajustadas para garantir o correto posicionamento dos sinais na tela, a posição vertical de cada canal do osciloscópio foi ajustada para não ocorrer sobreposição de sinais.

Figura 13 – Forma de onda para o veículo de 3 eixos classe 3C.

Fonte: Autoria própria (2025).

Figura 14 – Forma de onda para o veículo de 5 eixos classe 2S3.

Fonte: Autoria própria (2025).

O primeiro pulso (mais a esquerda nas figuras) representa o eixo dianteiro dos veículos que é mais leve que os demais quando sob carga. Outro detalhe importante pode ser notado no deslocamento temporal entre os sinais em cada canal. Como poder ser observado na Figura 12 existe um distanciamento físico entre os pares de sensores ao longo do pavimento, essa distância foi levada em consideração para que cada linha de sensores seja simulada adequadamente. A defasagem entre os sinais foi feita de modo a sequência de detecção de eventos de pesagem feita pela interface de processamento identificasse que o veículo estava a $\approx 90\,\mathrm{km/h}$.

Figura 15 – Forma de onda para o veículo de 6 eixos classe 3S3.

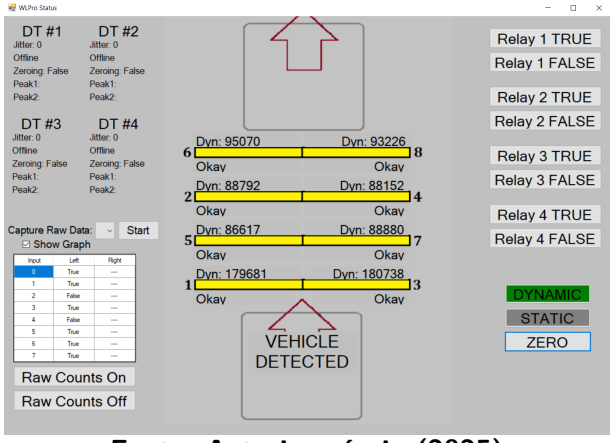
Fonte: Autoria própria (2025).

Figura 16 – Forma de onda para o veículo de 7 eixos classe 3D4.

Fonte: Autoria própria (2025).

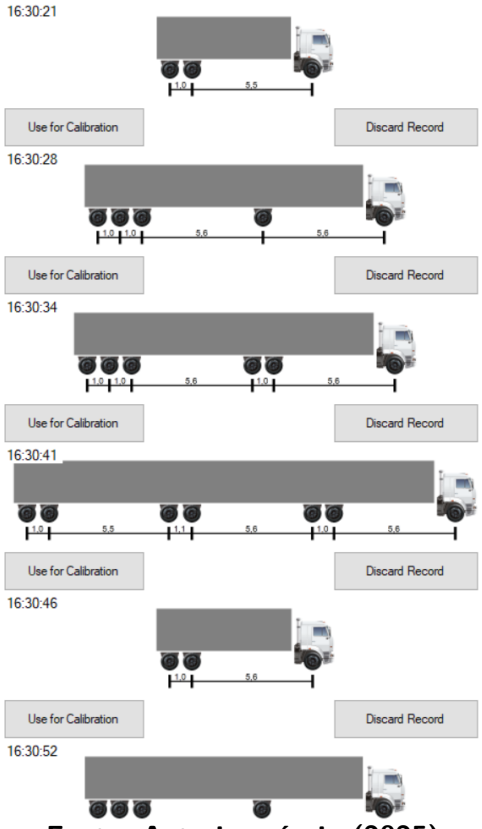
Quando o simulador foi devidamente conectado aos canais da interface de processamento, foi possível observar por meio do software da própria Intercomp a detecção de conexão dos sensores. A Figura 17 apresenta um print da tela do software na aba de status de conexão dos sensores, onde é possível constatar o status `Okay` para os 8 sensores. O print da tela foi feito no momento de detecção de sinal nos sensores, sendo possível observar que o laço indutivo foi devidamente acionado com `VEHICLE DETECTED` sendo exibido na parte inferior da imagem. Na Figura 18 é apresentado o print da tela de veículos detectados pela interface, onde é possível observar que a sequência de detecção está de acordo com o planejado, tanto a ordem de simulação quanto a questão da repetição cíclica dos veículos.

Figura 17 – Print da aba de status dos sensores.



Fonte: Autoria própria (2025).

Figura 18 – Print da aba de veículos detectados.



Fonte: Autoria própria (2025).

O sistema foi deixado ligado por pouco mais de 4,5 h coletando informações da medida de peso de cada eixo, em cada sensor. Posteriormente os dados foram tratados para processamento estatístico com o objetivo de verificar o grau de repetibilidade das medidas em cada sensor. Esta verificação permite avaliar se a etapa de condicionamento do sinal foi devidamente realizada pelo circuito, sem injetar distorções ou ruídos demais ao sinal, principalmente ao se levar em consideração tratar-se de um sinal de baixa frequência e baixa amplitude (na casa de mV).

Foram utilizadas duas abordagens distintas, porém complementares para se verificar o grau de repetibilidade nas leituras feitas pela interface. Uma abordagem foi a análise de variância (ANOVA) para testar a hipótese nula de que não há diferença estatisticamente significativa nas médias dos grupos de eixos, sendo a hipótese alternativa de que pelo menos um grupo possui média diferente. Ao se aplicar o teste ANOVA, avalia-se uma quantidade chamada de `p-valor`, o qual é comparado com um limiar chamado "nível de significância", que neste caso, é definido como 0,05. Se o `p-valor` for maior que este nível de significância não se pode rejeitar a hipótese nula. Isso significa que não há evidências suficientes para afirmar que as médias dos grupos são diferentes.

A outra abordagem utilizada foi o coeficiente de variação (CV), este parâmetro é uma medida estatística que expressa o grau de dispersão relativa dos dados em relação à média. Ele é utilizado para comparar a variabilidade entre diferentes conjuntos de dados, especialmente quando as médias desses conjuntos são distintas ou quando as unidades de medida não são as mesmas. O CV é definido como a razão entre o desvio padrão e a média aritmética, sendo geralmente expresso em porcentagem. Um CV baixo indica que os valores estão próximos da média, sugerindo alta repetibilidade ou precisão nas medições. Por outro lado, um CV elevado aponta para maior variabilidade e, portanto, menor consistência.

4.1 Análise dos Dados

A análise foi feita utilizando a linguagem python, pois apresenta bibliotecas que facilitam o tratamento e a visualização dos resultados. Foi utilizado a ferramenta `Jupyter notebook` para melhor organização dos passos adotados durante a análise. Inicialmente, as bibliotecas necessárias são carregadas, e é feita a leitura do dataset.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import f_oneway
```

[2]:
```
df = pd.read_csv("VehLog.csv")
```

Agrupamento e limpeza do dataset para melhor organização e retirada dos NaN, que podem atrapalhar os cálculos.

[3]:
```
# Identifica as classes de veículos
classes = df['Class'].unique()
```

```
# Cria um dicionário com os dados filtrados por classe
grupos_classes = {c: df[df['Class'] == c] for c in classes}

groups = {}
for sensor in [c for c in df.columns if "Scale" in c]:
  sensor_value = {}
  for c in classes:
    values = grupos_classes[c][sensor].dropna()
    if len(values) > 0:
      sensor_value[c] = values
  groups[sensor] = sensor_value
```

São aplicados dois métodos para verificar a repetibilidade dos dados. O Coeficiente de variação (CV) para indicar de forma percentual o quão espalhados os dados estão em torno da média, e análise de variância (ANOVA) para avaliar diferenças estatisticamente relevantes nas médias entre diferentes grupos, onde cada grupo é representado pela classe de veículo correspondente, relativo ao número de eixos.

[4]:
```
resultados = {}
anova = {}
for sensor_axle, v in groups.items():
  sensor_resultados = {}
  estat_values = {}
  arg = []
  # Coeficiente de variação - CV
  for classe, values in v.items():
    media = np.mean(values)
    desvio = np.std(values)
    cv = desvio / media * 100
    sensor_resultados[classe] = {'Média': media, 'Desvio':
       desvio, 'CV (%)': cv}

    # ANOVA
```

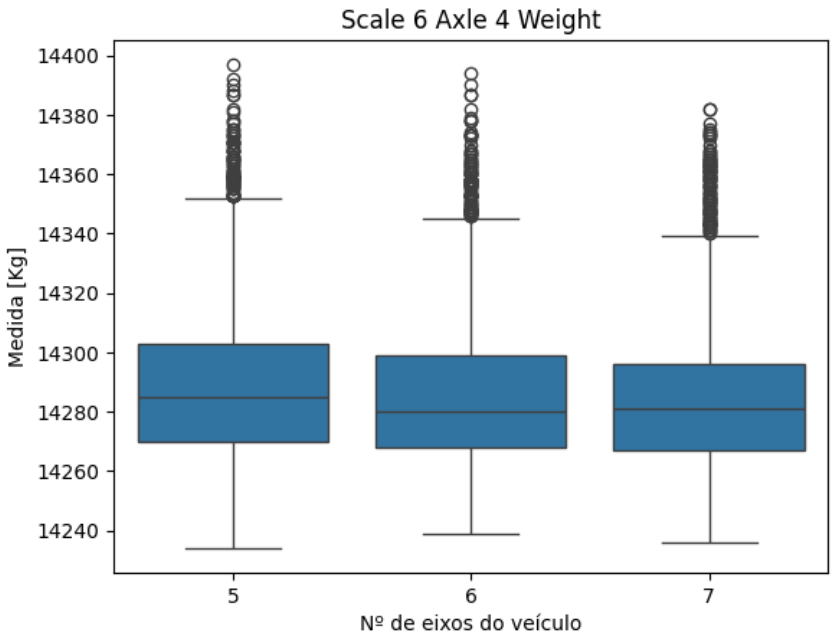
```
    arg.append(values)
  if len(arg) > 1:
    f_stat, p_valor = f_oneway(*arg)
    estat_values['f_stat'] = f_stat
    estat_values['p_valor'] = p_valor
    resultados[sensor_axle] = {'Estat. por classe':
       sensor_resultados, 'anova': estat_values}
  else:
    resultados[sensor_axle] = {'Estat. por classe':
       sensor_resultados}
```

Como o sinal para cada classe de veículo simulado é igual, diferindo apenas na quantidade de eixos, verifica-se quais grupos de sensores não passaram no teste ANOVA, ou seja, os casos em o p-valor ficou abaixo do nível de significância de 0,05. Para os casos em que a hipótese nula é rejeitada (p-valor $<$ 0,05) é plotado um boxplot para visualização da distribuição e é apresentado o CV (%) para completar a avaliação. Cada boxplot é apresentado com o valor da medida de peso no eixo vertical, e no eixo horizontal são separados os grupos de acordo com o número de eixo, deste modo, é possível comparar as medidas de peso de um determinado eixo, em um determinado sensor, para cada classe de veículo. Por exemplo, o número 5 no eixo horizontal representa o grupo de veículos que possuem 5 eixos, no caso, a classe 2S3.

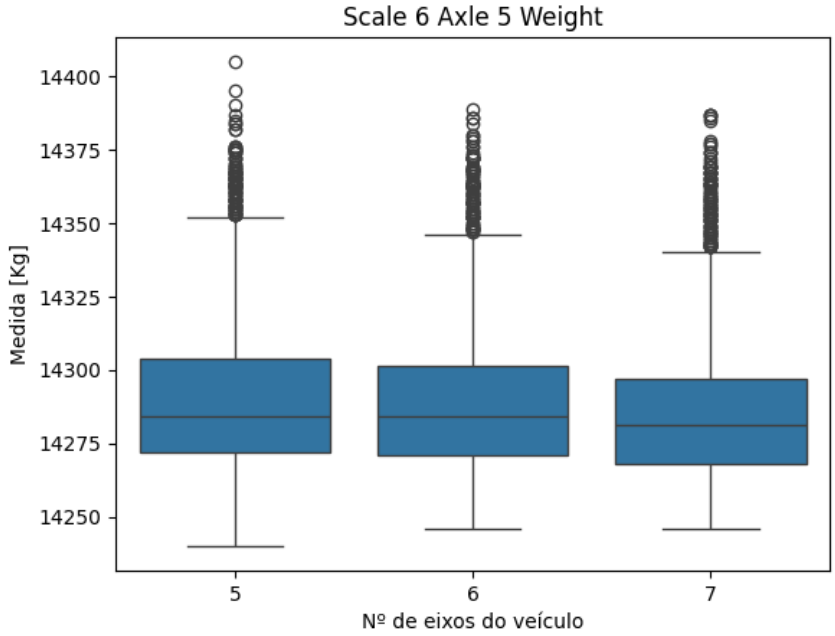
[5]:
```
sigf_level = 0.05
for k in groups.keys():
  if len(resultados[k]) > 1:
    if resultados[k]['anova']['p_valor'] < sigf_level:
      for classe, valores in resultados[k]['Estat. por
                classe'].items():
        print(f"Classe: {classe} -> CV (%): {valores['CV
                 (%)']:.4f}")

      # Converte o dicionário para formato "longo"
      df_groups = pd.DataFrame.from_dict(groups[k], orient =
         'index').transpose().melt(
```

```
    var_name='Nº de eixos do veículo', value_name='Medida
                [Kg]' )
    # Faz o boxplot
    sns.boxplot(data=df_groups, x='Nº de eixos do veículo',
         y='Medida [Kg]')
    plt.title(k)
    plt.show()
```

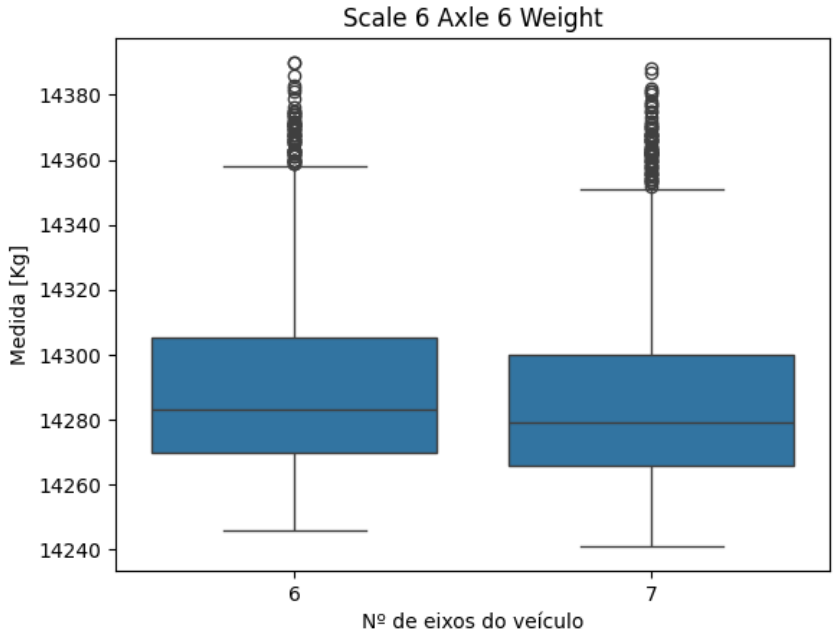
```
Classe: 5 -> CV (%): 0.2379
Classe: 6 -> CV (%): 0.2377
Classe: 7 -> CV (%): 0.2326
```



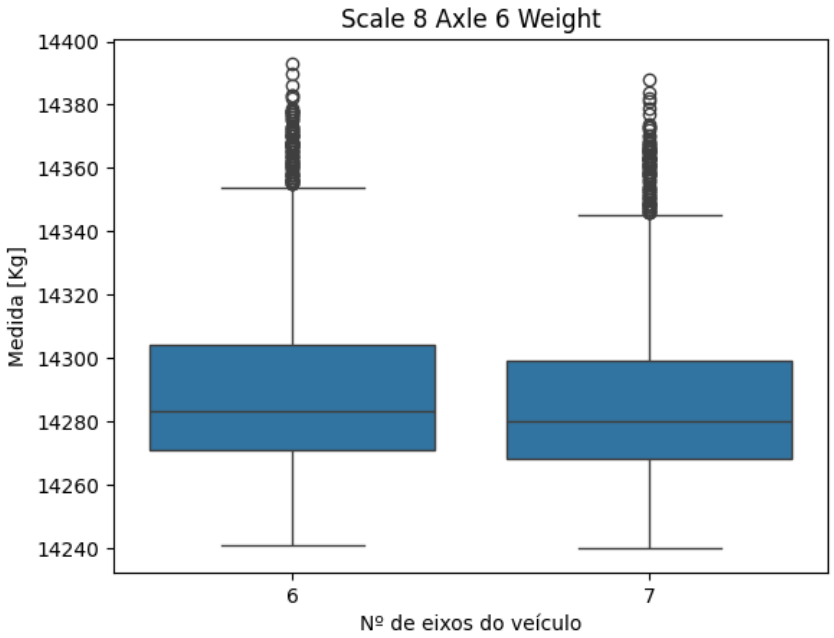

```
Classe: 5 -> CV (%): 0.2462
Classe: 6 -> CV (%): 0.2338
Classe: 7 -> CV (%): 0.2309
```




```
Classe: 6 -> CV (%): 0.2469
Classe: 7 -> CV (%): 0.2431
```




```
Classe: 6 -> CV (%): 0.2511
Classe: 7 -> CV (%): 0.2361
```



É possível observar em todos os grupos um valor para CV abaixo de $1\,\%$, o que significa variabilidade interna muito pequena, ou seja, dentro de cada grupo os valores estão muito próximos da média. Isso indica que a variabilidade entre as médias de grupo é maior do que a variabilidade dentro dos grupos. Deste modo, pode-se afirmar que houve alguma influência do circuito de condicionamento em apenas um DAC, sendo o canal de DAC responsável pela simulação dos sensores CN6 e CN8, no entanto, os canais apresentaram boa repetibilidade individualmente, permitindo afirmar que o simulador funcionou de forma adequada, atendendo ao objetivo estabelecido.

5 CONCLUSÃO

Desenvolver métodos e equipamentos para simulação de sistemas complexos ou de grande porte é de fundamental importância para o desenvolvimento de novas soluções que envolvem tal sistema, assim como executar os mais diversos testes e validações do mesmo, principalmente para implementação de correções e melhorias. Este trabalho teve por objetivo o desenvolvimento de um simulador eletrônico para células de carga strip-sensor, fabricados pela empresa Intercomp. Estes sensores são amplamente utilizados em sistemas de pesagem de veículos de grande porte em rodovias, pois possibilitam que a pesagem seja realizada com o veículo em alta velocidade, sendo este sistema conhecido como HSWIM.

Para o desenvolvimento do simulador, foram utilizados conversores digital-analógico (DAC) presentes em um microcontrolador para gerar os sinais representativos das células de carga. Cada canal de DAC foi utilizado para simular um par de sensores, que seriam responsáveis pela pesagem de cada rodado do veículo. Deste modo, foram simulados 4 pares de sensores seguindo uma determinada configuração de posicionamento dos mesmos no pavimento. Foi implementado também um simulador de laço indutivo para servir como gatilho de presença de veículo para a interface de processamento.

Os resultados obtidos durante o intervalo de algumas horas de simulação foram tratados e analisados por dois métodos para verificação de repetibilidade das medidas, ou seja, para verificar se o sinal gerado pela eletrônica do simulador, e injetado na CPU mantinha algum nível de estabilidade.

Os resultados obtidos mostraram que funcionamento desejado para o simulador foi bem sucedido, sendo reconhecido corretamente pela interface como se fosse um veículo real passando sobre os sensores, apresentando baixo ruido, e boa consistência nos sinais gerados. Deste modo, o simulador poderia ser utilizado para testes, correções e implementações de funcionalidades em sistemas de pesagem em alta velocidade (HSWIM).

REFERÊNCIAS

CNT BRASíLIA. **Pesquisa CNT de Rodovias 2024**. 2024. Disponível em: https://www.cnt.org.br/pesquisas.

COIFMAN, B.; KIM, S. Vehicle reidentification and travel time measurement in real-time on freeways using the existing loop detector infrastructure. **Transportation Research Part C: Emerging Technologies**, Elsevier, v. 14, n. 3, p. 149–169, 2006.

HERRERA, S.; GERSTENHABER, M. **Versatile, Low Power, Precision Single-Ended-to-Differential Converter**. 2012. Mouser Electronics Application Note. Disponível em: https://br.mouser.com/applications/low-power-differential-converter/. Acesso em: 21-09-2025.

Intercomp Company. **Analog Integration – Strip Sensor Data Sheet – SR**. Medina, MN, USA, 2016. Data sheet for HS-WIM Strip Sensors. Disponível em: https://www.intercompcompany.com.

Intercomp Company. **Weigh-In-Motion Strip Sensor Application Guide**. Medina, MN, USA, 2017. Document ID: 700733_Strip-Sensor-APP-GUIDE_SRweb. Disponível em: https://www.intercompcompany.com.

JACOB, B.; BEAUMELLE, V. Feypell-de L. Improving truck safety: Potential of weigh-in-motion technology. **IATSS research**, Elsevier, v. 34, n. 1, p. 9–15, 2010.

KLEIN, L. A.; MILLS, M. K.; GIBSON, D. R. **Traffic Detector Handbook: Third Edition—Volume I**. McLean, VA, 2006. Published under sponsorship of the U.S. Department of Transportation. Disponível em: https://www.fhwa.dot.gov/publications/research/operations/its/06108/06108.pdf.

ROCHETI, E. O.; BACURAU, R. M. Weigh-in-motion systems review: Methods for axle and gross vehicle weight estimation. **IEEE Access**, IEEE, 2024.

STMICROELECTRONICS. **AN4013: STM32 Timers – Overview and Examples**. [*S.l.*], 2021. Disponível em: https://www.st.com/resource/en/application_note/an4013-general-purpose-timers-stmicroelectronics.pdf.

STMICROELECTRONICS. **Reference manual STM32G4 Series - advanced Arm-based 32-bit MCUs**. 2025.

SUJON, M.; DAI, F. Application of weigh-in-motion technologies for pavement and bridge response monitoring: State-of-the-art review. **Automation in Construction**, Elsevier, v. 130, p. 103844, 2021.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas digitais: Princípios e Aplicações**. 10. ed. São Paulo: Pearson Prentice Hall, 2011.

TOLENTINO, D.; HERRERA, S. Versatile, precision single-ended-to-differential signal conversion circuit with adjustable output common mode. **Analog Dialogue**, v. 49, n. 11, 2015. https://www.analog.com/en/resources/analog-dialogue/articles/single-ended-to-differential-signal-conversion.html.

YIU, J. **The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors**. 3rd. ed. [*S.l.*]: Elsevier, 2014. ISBN 978-0124080829.