INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÂO EM ENGENHARIA ELETRÔNICA

CONRADO BECKER GRESSLER

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA BANCADA DE TESTES DE MÓDULO DE CONTROLE DE CARROCERIA

FLORIANÓPOLIS, 2025.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA - CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO DE GRADUAÇÂO EM ENGENHARIA ELETRÔNICA

CONRADO BECKER GRESSLER

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA BANCADA DE TESTES DE MÓDULO DE CONTROLE DE CARROCERIA

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador:
Prof. Renan Augusto Starke, Dr. Eng.

Coorientador:
Alexandre Marcondes, Msc. Eng.

FLORIANÓPOLIS, 2025.

Ficha de identificação da obra elaborada pelo autor.

Gressler, Conrado
DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA BANCADA DE TESTES DE MÓDULO DE CONTROLE DE CARROCERIA / Conrado Gressler; orientação de Renan Starke; coorientação de Alexandre Marcondes. - Florianópolis, SC, 2025.
45 p.

Trabalho de Conclusão de Curso (TCC) - Instituto Federal de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Eletrônica. Departamento Acadêmico de Eletrônica.
Inclui Referências.

1. Módulo de controle de carroceria. 2. LabVIEW. 3. C++. 4. Simulação. I. Starke, Renan. II. Marcondes, Alexandre. III. Instituto Federal de Santa Catarina. IV. DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA BANCADA DE TESTES DE MÓDULO DE CONTROLE DE CARROCERIA.

DESENVOLVIMENTO DE UM SISTEMA DE CONTROLE PARA BANCADA DE TESTES DE MÓDULO DE CONTROLE DE CARROCERIA

CONRADO BECKER GRESSLER

Este trabalho foi julgado adequado para obtenção do título de Bacharel em Engenharia Eletrônica e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Eletrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 19 de Dezembro, 2025.

Banca Examinadora:

gov.br Documento assinado digitalmente
RENAN AUGUSTO STARKE
Data: 19/02/2026 16:03:44-0300
Verifique em https://validar.iti.gov.br

_____
Prof. Renan Augusto Starke, Dr. Eng.

gov.br Documento assinado digitalmente
ALEXANDRE MARCONDES
Data: 20/02/2026 17:24:50-0300
Verifique em https://validar.iti.gov.br

____ _____

gov.br Documento assinado digitalmente
MATHEUS LEITZKE PINTO
Data: 20/02/2026 14:40:20-0300
Verifique em https://validar.iti.gov.br

_____
Prof. Matheus Leitzke Pinto, Msc. Eng.

gov.br Documento assinado digitalmente
DANIEL LOHMANN
Data: 19/02/2026 20:06:53-0300
Verifique em https://validar.iti.gov.br

_____
Prof. Daniel Lohmann, Msc. Eng.

AGRADECIMENTOS

Gostaria de agradecer aos meus familiares que me ajudaram durante esta jornada; sem o suporte da minha família eu nunca chegaria aonde cheguei. Também gostaria de agradecer aos meus amigos, que ajudaram a deixar meus dias mais alegres. Sou extremamente grato por aqueles que me acompanharam nesta jornada, especialmente pelas amizades que forjei com colegas e espero que estas amizades se extendam para a vida toda.

Gostaria de agradecer a todos os meus colegas de equipe da Fundação CERTI, em especial ao meu coordenador, Alexandre Marcondes, por ter acreditado na minha capacidade e me oferecido uma das melhores oportunidades da minha vida.

Finalmente, gostaria de agradecer a todos os professores do IFSC, que abriram meus olhos para o mundo da engenharia e não apenas me moldaram como um bom profissional, mas também me tornaram uma pessoa melhor. Por fim, agradeço ao meu orientador, um profissional pelo qual tenho grande respeito e que sempre esteve disposto a me auxiliar.

"Ódio é o lugar para onde
vai o homem que não
consegue suportar a tristeza."
- Kentaro Miura

RESUMO

Este trabalho é fruto de uma parceria entre a empresa Magneti Marelli e a Fundação CERTI, buscando auxiliar no desenvolvimento de equipamentos usados no setor automotivo na forma de um protótipo capaz de simular componentes reais de forma programática. Este trabalho tem como objetivo o desenvolvimento de um sistema capaz de emular o comportamento de motores DC e potenciômetros digitais utilizando uma abordagem de *hardware-in-the-loop*, visando emular os componentes presentes em veículos automotivos. O protótipo foi concebido como ferramenta de apoio para o desenvolvimento de *firmware* de um módulo de controle de carroceria automotivo BCM (*Body Control Module*). A BCM gerencia sistemas não críticos do veículo, como acionamento de vidros elétricos, movimentação de espelhos retrovisores e controle do ar-condicionado. Neste trabalho, o foco é um subsistema específico da BCM, denominado DMM (*Door-Mirror Module*), responsável pelo controle dos vidros e espelhos do veículo. No DMM, os motores DC realizam o acionamento dos vidros elétricos, enquanto os potenciômetros digitais são utilizados no controle dos espelhos retrovisores. A simulação dos motores é feita por meio do acionamento de cargas eletrônicas, controladas via conversores analógico-digital. Já os potenciômetros digitais são controlados por meio da interface I2C. O projeto também inclui o desenvolvimento de um software em LabVIEW, capaz de se comunicar com o protótipo. Essa interface permite ao usuário definir as características físicas que serão simuladas. Para validação do sistema, foi utilizada uma placa eletrônica com o mesmo circuito do sistema real, conectada ao protótipo. As medições foram realizadas utilizando o sistema de aquisição NI PXI-5152 em conjunto com o chassi PXIe-1084. Os resultados obtidos ficaram dentro do esperado, embora tenham sido identificadas limitações relacionadas principalmente ao uso da interface I2C. Também foram apontadas oportunidades de melhoria envolvendo componentes, microcontrolador e otimizações no código, visando à implementação de um sistema final com características de *hard real time*. O trabalho demonstrou a viabilidade de desenvolver um sistema em tempo real capaz de simular funções como inclinação dos espelhos (*mirror tilt*), rebatimento dos espelhos (*mirror fold*) e acionamento dos vidros elétricos (*window lift*). Além disso, o projeto estabelece uma base sólida para futuros desenvolvimentos na área.

Palavras-chave: Módulo de controle de carroceria. C++. LabVIEW. Simulação.

ABSTRACT

This work is the result of a partnership between Magneti Marelli and CERTI Foundation, aiming to support the development of equipment used in the automotive sector through a prototype capable of programmatically simulating real components. The objective of this work is the development of a C++ firmware for the Arduino Giga R1 platform. The system is responsible for controlling a prototype designed to simulate DC motors and to drive digital potentiometers in order to replicate components found in automotive vehicles. The prototype was conceived as a support tool for the development of firmware for an automotive Body Control Module (BCM). The BCM manages non-critical vehicle systems, such as power window operation, side mirror adjustment, and air conditioning control. In this work, the focus is on a specific BCM subsystem called the Door-Mirror Module (DMM), which is responsible for controlling the vehicle's windows and mirrors. In the DMM, DC motors are used to operate the power windows, while digital potentiometers are used to control the side mirrors. The motor simulation is carried out by driving electronic loads controlled via analog-to-digital converters. The digital potentiometers, in turn, are controlled through the I2C interface. The project also includes the development of a LabVIEW-based software application capable of communicating with the prototype. This interface allows the user to define the physical characteristics to be simulated. For system validation, an electronic board with the same circuitry as the real system was connected to the prototype. Measurements were performed using the NI PXI-5152 data acquisition system together with the PXIe-1084 chassis. The results obtained were within the expected range, although limitations were identified, mainly related to the use of the I2C interface. Opportunities for improvement were also identified regarding components, microcontroller selection, and code optimization, aiming at the implementation of a final system with hard real-time characteristics. This work demonstrated the feasibility of developing a real-time system capable of simulating functions such as mirror tilt, mirror fold, and power window operation (window lift). Furthermore, the project establishes a solid foundation for future developments in this field.

Keywords: Body Control Module. C++. LabVIEW. Simulation.

LISTA DE FIGURAS

Figura 1 – Gráfico de corrente da carga (modo corrente constante) 16
Figura 2 – Circuito de uma carga eletrônica em modo de corrente constante . . 17
Figura 3 – Gráfico da corrente na carga (Modo de Tensão Constante) 19
Figura 4 – Circuito de uma carga eletrônica em modo de tensão constante . . 21
Figura 5 – Forma de onda da comunicação I2C 22
Figura 6 – Diagrama de blocos do Arduino Giga R1 24
Figura 7 – Diagrama de um sistema com HIL 26
Figura 8 – Exemplo de aplicação usando LabVIEW 28
Figura 9 – Diagrama de bloco do potenciômetro digital MCP45HV51 29
Figura 10 – Protótipo montado e sistema com cargas eletrônicas e potenciômetros digitais 32
Figura 11 – Esquemático do circuito do potenciômetro digital 35
Figura 12 – Forma de onda dos motores DC 36
Figura 13 – Diagrama de blocos do sistema 37
Figura 14 – Circuito das cargas eletrônicas 40
Figura 15 – Circuito de identificação de sentido de movimento 41
Figura 16 – Diagrama de funcionamento da carga eletrônica 42
Figura 17 – Diagrama de classe dos potenciômetros e motores 43
Figura 18 – Front panel do software 45
Figura 19 – Setup de medições 46
Figura 20 – Protótipo e sistema de acesso à sinais 47
Figura 21 – Teste do valor mínimo do potenciômetro digital 48
Figura 22 – Teste do valor máximo do potenciômetro digital 49
Figura 23 – Saída de controle do DAC7678 50
Figura 24 – Comparação das saídas de controle do DAC7678 51
Figura 25 – Sinais de controle 52
Figura 26 – Sinais de controle e tensão no resistor shunt 53

LISTA DE TABELAS

Tabela 1 – Modos de operação I2C 22
Tabela 2 – Estrutura das mensagens 44

LISTA DE ABREVIATURAS E SIGLAS

BCM	*Body Control Module*
CMOS	*Complementary Metal-Oxide-Semiconductor*
CRC	*Cyclic Redundancy Check*
DAC	*Digital-to-Analog Converter* (Conversor Digital-Analógico)
DC	*Direct Current* (Corrente contínua)
DMM	*Door-Mirror Module* (Módulo Porta-Espelho)
EoT	*End of Travel*
FET	*Field-Effect Transistor*
HIL	*Hardware in the loop*
I2C	*Inter-Intergrated Circuit*
IDE	*Integrated Development Environment*
IFSC	Instituto Federal de Santa Catarina
IGBT	*Insulated-Gate Bipolar Transistor*
JSON	*JavaScript Object Notation*
MOSFET	*Metal-Oxide Semiconductor Field Effect Transistor*
QMH	*Queued Message Handler*
RPC	*Remote Procedure Call*
SRAM	*Static Random Access Memory*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 Justificativa 13
1.2 Definição do Problema 14
1.3 Objetivo Geral 14
1.4 Objetivos Específicos 14
1.5 Estrutura do Trabalho 15
2 FUNDAMENTAÇÃO TEÓRICA 16
2.1 Cargas Eletrônicas 16
2.1.1 Modo de Corrente Constante 16
2.1.2 Modo de Tensão Constante 19
2.2 Interface I2C 21
2.3 Placa de desenvolvimento Arduino Giga R1 23
2.4 Hardware-in-the-loop 25
2.5 LabVIEW 27
2.6 Potenciômetros digitais 28
2.7 NI PXIe-8135 e NI PXI-5152 30
2.8 Resumo 31
3 METODOLOGIA 32
3.1 Métodos Aplicados 32
3.1.1 Teste dos potenciômetros digitais 33
3.1.2 Análise da forma de onda dos motores DC 35
3.1.3 Arquitetura protótipo 36
3.1.4 Funcionamento das cargas eletrônicas 38
3.1.5 Utilização do Arduino Mega 2560 41
3.1.6 Arquitetura de software e firmware 42
4 APRESENTAÇÃO DOS RESULTADOS 46
4.1 Análise e discussão dos resultados 46
4.1.1 Validação dos potenciômetros digitais 47
4.1.2 Validação da forma de onda gerada 49
5 CONSIDERAÇÕES FINAIS 54
5.1 Sugestões para trabalhos futuros 54
REFERÊNCIAS 55

1 INTRODUÇÃO

Nos veículos modernos, diversos sistemas eletrônicos trabalham de forma integrada para garantir segurança e funcionalidade aos ocupantes. Esses sistemas são organizados em módulos eletrônicos, cada um responsável por um conjunto específico de funções. Entre eles, destacam-se os módulos destinados ao controle de componentes não essenciais ao funcionamento e controle do motor do veículo, mas controlam portas, janelas, travamento das portas, iluminação interna e espelhos retrovisores. É dentro desse contexto que se insere o *Body Control Module* (BCM) e seus submódulos especializados, como o *Door-Mirror Module* (DMM), responsáveis pelo gerenciamento de alguns destes periféricos.

O DMM (*Door-Mirror Module*) é responsável pelo controle das janelas e dos espelhos do carro. O DMM realiza o controle de subida e descida dos vidros das portas e a movimentação nos eixos dos retrovisores, bem como o acionamento da retração dos espelhos.

Este tema surgiu com um projeto realizado pela Fundação CERTI para a empresa Magneti Marelli visando implementar um sistema que auxilie as equipes de desenvolvimento dos DMMs da Magneti Marelli com um sistema capaz de emular o comportamento dos componentes controlados pelo DMM. Para isso, foi realizada uma análise do funcionamento dos DMMs, para que fosse possível entender o componentes do DMM que seriam emulados e como o comportamento deles poderia ser emulado de forma programática, por meio de um *software*.

A forma como é realizado o controle dos periféricos controlados pelo DMM varia de acordo com o fabricante do equipamento. Dessa forma, cada empresa desenvolve seu próprio sistema de controle específico para cada modelo de veículo devido à variação do *hardware* entre diferentes modelos de veículo. No entanto, algumas características são padronizadas entre DMMs, com o acionamento das janelas do carro realizado por motores DC, o *mirror-fold* também é feito usando motores DC e a verificação da posição dos espelhos é realizada por meio da leitura da tensão em potenciômetros, cuja variação da resistência depende da inclinação dos espelhos. Cada espelho possui dois potenciômetros, um para o eixo X e outro para o eixo Y, totalizando quatro potenciômetros por veículo.

Dessa forma, para realizar testes e validações do DMM, é necessária a utilização dos mesmos motores e dos mesmos potenciômetros que serão utilizados no veículo para o qual o DMM está sendo desenvolvido. O fato de existir a necessidade da presença do *hardware* para o desenvolvimento do *firmware* do DMM é algo que pode vir a atrasar o desenvolvimento do produto final. Uma forma de mitigar por completo a necessidade dos motores e potenciômetros é realizar a simulação do acionamento

das janelas por meio de cargas eletrônicas para gerar a forma de onda de corrente dos motores DC. Enquanto a variação da inclinação dos espelhos, que utilizam potenciômetro seria simulada com o uso de potenciômetros digitais.

Portanto, buscou-se utilizar uma abordagem de *Hardware-in-the-Loop* (HIL) para a simulação destes componentes que possuem características físicas variáveis. Segundo MATHWORKS (2025, tradução nossa), "É uma técnica para desenvolvimento de sistemas embarcados. Envolve conectar as entradas e saídas reais da interface do hardware controlador em um ambiente virtual que simula o sistema físico.".

Para que fosse possível emular diversos parâmetros diferentes, foi desenvolvido um *software* utilizando a linguagem de programação LabVIEW onde o usuário consegue configurar cada uma das características dos motores das janelas, cada um dos potenciômetros dos espelhos, os valores de corrente dos potenciômetros e a corrente do motor que realiza o rebatimento dos espelhos. Além disso, o *software* permite importar e exportar configurações, fazendo com que com apenas a troca do arquivo de configuração seja possível trocar todas as características que serão simuladas pelo sistema, essencialmente trocando o carro para o qual o DMM está sendo programado. O *software* foi desenvolvido para computadores que utilizem sistema operacional Windows (versão 10 ou superior) e permite a configuração do protótipo por meio de conexão USB.

Este trabalho adotou uma abordagem experimental e quilitativa, visando desenvolver um protótipo que auxilie no desenvolvimento de BCMs com foco na viabilidade de utilizar um sistema microcontrolado para simulação de motores DC.

1.1 Justificativa

O desenvolvimento do *firmware* responsável pelo controle do DMM está diretamente relacionado aos parâmetros físicos do sistema a ser controlado. Assim, a disponibilidade dos motores responsáveis pelo acionamento das janelas e dos potenciômetros empregados nos espelhos retrovisores torna-se fundamental para o desenvolvimento e validação do sistema de controle.

Entretanto, nem sempre tais componentes encontram-se disponíveis durante as etapas de projeto e teste. Ainda assim, suas características elétricas e mecânicas são, em geral, descritas em folhas de dados (*datasheets*), as quais se encontram amplamente acessíveis.

Diante da dependência dos componentes físicos para o desenvolvimento do DMM, este trabalho propõe a implementação de uma solução capaz de emular o comportamento dos dispositivos originalmente controlados pelo módulo, possibilitando a continuidade das atividades de desenvolvimento e validação do firmware independente da presença do *hardware*. Com a definição dos parâmetros a serem emulados

configurados pelo usuário usando um *software* capaz de salvar e importar configurações, facilita-se a realização de testes para múltiplos veículos diferentes.

O desenvolvimento deste sistema se justifica no auxílio ao desenvolvimento de DMMs e disponibiliza uma base teórica para que seja possível expandir a utilização deste sistema para áreas externas à área automotiva. Este trabalho também documenta o estudo e o uso de cargas eletrônicas para geração de formas de ondas.

1.2 Definição do Problema

O desenvolvimento e a validação de firmware para oDMM dependem diretamente da interação com componentes físicos reais, tais como motores elétricos e potenciômetros. A ausência desses componentes durante as etapas de projeto e teste limita a capacidade de verificação funcional do sistema, dificultando a reprodução de diferentes condições operacionais e comprometendo a eficiência do processo de desenvolvimento. Nesse contexto, o problema principal deste trabalho consiste em: Como desenvolver um sistema capaz de emular todos os elementos físicos que são controlados pelo DMM com os parâmetros dos mesmos configurados por *software*?

1.3 Objetivo Geral

Desenvolver um sistema capaz de emular o comportamento dos componentes presentes no DMM, parametrizados por um usuário via um *software* LabVIEW realizando o controle por meio de um Arduino Giga R1.

1.4 Objetivos Específicos

Para obter um protótipo funcional e dentro das especificações acordadas com a Magneti Marelli, foram definidos os seguintes objetivos específicos:

a) Definir um circuito de controle que permita realizar o controle das cargas eletrônicas e o monitoramento da fonte de entrada.

b) Desenvolver uma maneira de gerar uma onda quadrada de, no mínimo, 500 Hz e até 1500 Hz em quatro cargas eletrônicas simultâneamente para emular o comportamento dos motores DC;

c) Realizar a variação de todos os potenciômetros de forma determinista, respeitando o tempo de variação dos mesmos.

d) Implementar funcionalidades para simulações de travamentos do motor (*pinch*) e contagem de pulsos.

e) Definir uma forma de salvar e interpretar as configurações criadas pelo usuário;

f) Implementar um método de comunicação entre o *software* LabVIEW e a placa de desenvolvimento;

g) Permitir a criação de arquivos de configuração, bem como a capacidade de importar os mesmos para a placa de desenvolvimento.

1.5 Estrutura do Trabalho

Este trabalho é estruturado em cinco capítulos, sendo o primeiro destes uma breve introdução ao tema.

O segundo Capítulo se refere à fundamentação teórica utilizada que viabilizou este trabalho, mostrando quais tecnologias já são utilizadas na área e também quais se adequam mais ao projeto.

No terceiro Capítulo é explicada a metodologia aplicada neste trabalho. Demonstrando como o projeto foi realizado de forma modular, validando cada uma das funcionalidades de forma isolada e depois implementando as mesmas e testando no sistema completo.

O quarto Capítulo apresenta os resultados obtidos, quais possíveis pontos de melhoria, problemas encontrados, análises da geração de forma de onda do protótipo final e verificando que o sistema mostra uma ferramenta de auxílio ao desenvolvimento de BCMs.

No quinto Capítulo são apresentadas as conclusões, ideias de novos trabalhos a partir dos resultados e análises realizadas neste projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta as tecnologias utilizadas para o desenvolvimento do protótipo responsável por emular os componenetes controlados pelo DMM, o funcionamento de cargas eletrônicas e como estas podem ser controladas para gerar níveis de tensão e corrente variáveis, as configurações do Arduino Giga R1 e a utilização do protocolo I2C utilizado na comunicação entre a placa de desenvolvimento e os DACs e os potenciômetros digitais.

2.1 Cargas Eletrônicas

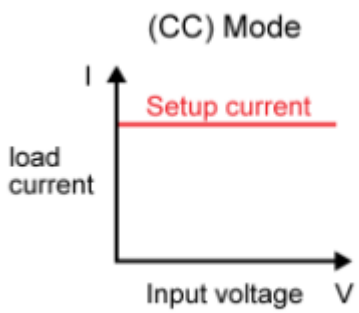
Cargas eletrônicas são circuitos normalmente utilizados para testes de fontes, possuindo diversos modos de operação. Os principais modos de operação são de corrente constante, tensão constante e potência constante. Segundo Keysight (2025), estes modos possibilitam a realização de testes de estresse de fontes de diversas maneiras. Em adição aos modos de operação, cargas eletrônicas também contam com proteções de sobrecorrente e sobretensão, permitindo testar as capacidades do dispositivo sem que a carga seja danificada. Cargas eletrônicas são boas formas de testar baterias, fontes, painéis solares ou qualquer tipo de equipamento que seja alimentado com tensão DC.

Uma carga eletrônica utiliza um MOSFET operando na região linear para fazer o ajuste do valor de corrente com base na tensão entre *gate* e *source* do MOSFET, segundo Matsusada (2025).

2.1.1 Modo de Corrente Constante

Quando em modo de corrente constante, a impedância da carga altera-se dinamicamente de forma a não variar o valor da corrente, independentemente da tensão de entrada que seja aplicada, conforme ilustrado na Figura 1. A configuração de corrente constante foi o modo utilizado neste trabalho

Figura 1 – Gráfico de corrente da carga (modo corrente constante)

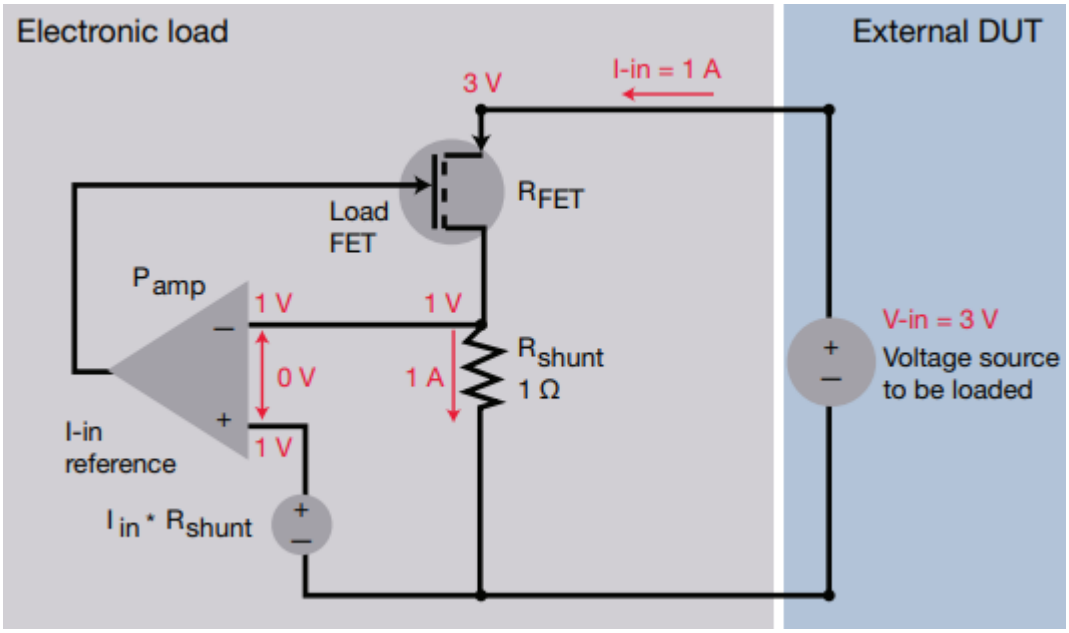


Fonte: Matsusada (2025).

Cargas eletrônicas são comumente utilizadas em modo de corrente constante, para testes de baterias, fontes de alimentação DC, ou em cenários onde se deseja verificar a resposta do equipamento a ser testado quando a tensão varia, mas a corrente permanece a mesma.

> A maioria das cargas eletrônicas utiliza transistores de potência, FETs ou IGBTs que atuam como um resistor variável para regular a corrente que passa pela carga. Os transistores são tipicamente arranjados em uma configuração em paralelo para lidar com potências maiores. (KEYSIGHT,2025, tradução nossa).

Figura 2 – Circuito de uma carga eletrônica em modo de corrente constante



Fonte: Keysight (2022, p. 5).

A Figura 2 ilustra uma carga eletrônica operando em modo de corrente constante, evidenciando seus principais blocos funcionais: a fonte sob teste (*External DUT*), o MOSFET de carga (*Load FET*, representado por R_{FET}), o resistor de medição R_{shunt}, o amplificador operacional (Pamp) e a fonte de referência de corrente.

O terminal positivo da fonte sob teste (V_{in}) é conectado ao dreno do MOSFET de potência. A corrente de entrada (I_{in}) fornecida pela fonte percorre inicialmente o MOSFET, em seguida atravessa o resistor R_{shunt} (1 Ω) e retorna ao terminal negativo da fonte, fechando o circuito.

O resistor R_{shunt} é o elemento responsável pela medição da corrente. Conforme indicado na figura, quando a corrente é de 1 A e o resistor possui valor de 1 Ω, estabelece-se uma queda de tensão de 1 V sobre ele. Essa relação é descrita pela Lei de Ohm:

$$V_{shunt} = I_{in} \cdot R_{shunt} \tag{1}$$

Dessa forma, a corrente é convertida em uma grandeza de tensão proporcional, permitindo sua comparação com uma referência.

A tensão medida no nó entre o MOSFET e o resistor R_{shunt} é aplicada à entrada inversora $(-)$ do amplificador operacional (Pamp). Na entrada não inversora (+) é aplicada uma tensão de referência identificada na figura como V_{ref}, definida por:

$$V_{ref} = I_{desejada} \cdot R_{shunt} \tag{2}$$

O amplificador operacional realiza a comparação contínua entre a tensão de referência e a tensão medida no resistor shunt. Esse mecanismo caracteriza uma malha de realimentação negativa, na qual o sistema se estabiliza quando:

$$V_{shunt} = V_{ref} \tag{3}$$

Substituindo a Equação (1) na condição de equilíbrio dada pela Equação (3), obtém-se a corrente de operação:

$$I_{in} = \frac{V_{ref}}{R_{shunt}} \tag{4}$$

Observa-se, portanto, que a corrente drenada da fonte depende exclusivamente da tensão de referência e do valor do resistor shunt.

Na condição apresentada na figura, com V_{in} = 3 V e V_{shunt} = 1 V, a tensão restante aparece sobre o MOSFET. A potência dissipada no dispositivo é dada por:

$$P = V_{DS} \cdot I_{in} \tag{5}$$

Onde V_{DS} é a tensão entre dreno e fonte do MOSFET.

Dessa forma, com base nos componentes mostrados na Figura 2, o circuito constitui um sistema de controle analógico em malha fechada, no qual o resistor R_{shunt} realiza a medição da corrente, o amplificador operacional (Pamp) executa a comparação com a referência, o MOSFET (R_{FET}) atua como elemento de controle e a fonte externa (DUT) fornece a energia que será dissipada pela carga eletrônica.

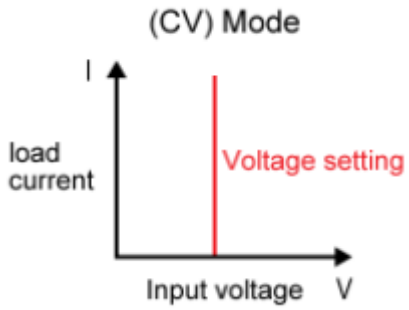
Neste trabalho foi utilizada a carga eletrônica no modo de corrente contínua, com o valor de tensão que controla a corrente da carga como sendo a saída do microcontrolador, desta forma foi possível gerar uma forma de onda variável usando a placa de desenvolvimento.

A utilização da carga eletrônica em modo de corrente contínua também permite que seja possível obter o valor de corrente desejável independente da tensão de entrada, uma vez que uma bateria de carro que é responsável pelo acionamento dos motores pode operar com uma tensão variável entre 11 V e 15 V.

2.1.2 Modo de Tensão Constante

Uma carga eletrônica de modo de tensão constante não varia a tensão da sua entrada, a carga ajusta a sua tensão com base na corrente de entrada, como ilustrado na Figura 3. Menos usual que o modo de corrente constante, podem ser utilizadas em testes de carregadores de baterias, segundo KEYSIGHT (2025, tradução nossa), "Testar um carregador de bateria operando em modo de corrente constante é um exemplo de como você usaria uma carga eletrônica operando em modo de tensão constante".

Figura 3 – Gráfico da corrente na carga (Modo de Tensão Constante)



Fonte: Matsuda (2025).

Uma carga eletrônica operando em modo de tensão constante (CV – *Constant Voltage*) tem como objetivo manter a tensão aplicada aos terminais do dispositivo sob teste (DUT) em um valor previamente definido, independentemente da variação da corrente fornecida por esse dispositivo. Para isso, o circuito utiliza a região linear de operação do MOSFET, explorando sua resistência equivalente variável como elemento de controle.

O circuito representado na Figura 4 é composto por um amplificador operacional, um transistor MOSFET de potência (Load FET), um divisor resistivo formado por R_1 e R_2, e uma fonte de tensão de referência. O princípio de funcionamento baseia-se em uma malha de realimentação negativa, responsável por ajustar dinamicamente a condução do MOSFET de forma a estabilizar a tensão nos terminais de entrada da carga.

A tensão aplicada pelo DUT aos terminais In+ e In– é amostrada pelo divisor resistivo composto pelos resistores R_1 e R_2. A tensão no ponto intermediário do divisor, denotada por V_m, é dada por:

$$V_m = V_{in} \cdot \frac{R_1}{R_1 + R_2} \tag{6}$$

Essa tensão V_m é aplicada à entrada inversora do amplificador operacional. A entrada não inversora recebe uma tensão de referência V_{ref}, definida externamente e correspondente ao valor de tensão desejado na entrada da carga. O amplificador operacional compara continuamente V_m com V_{ref} e ajusta sua saída de forma a minimizar o erro entre essas duas grandezas.

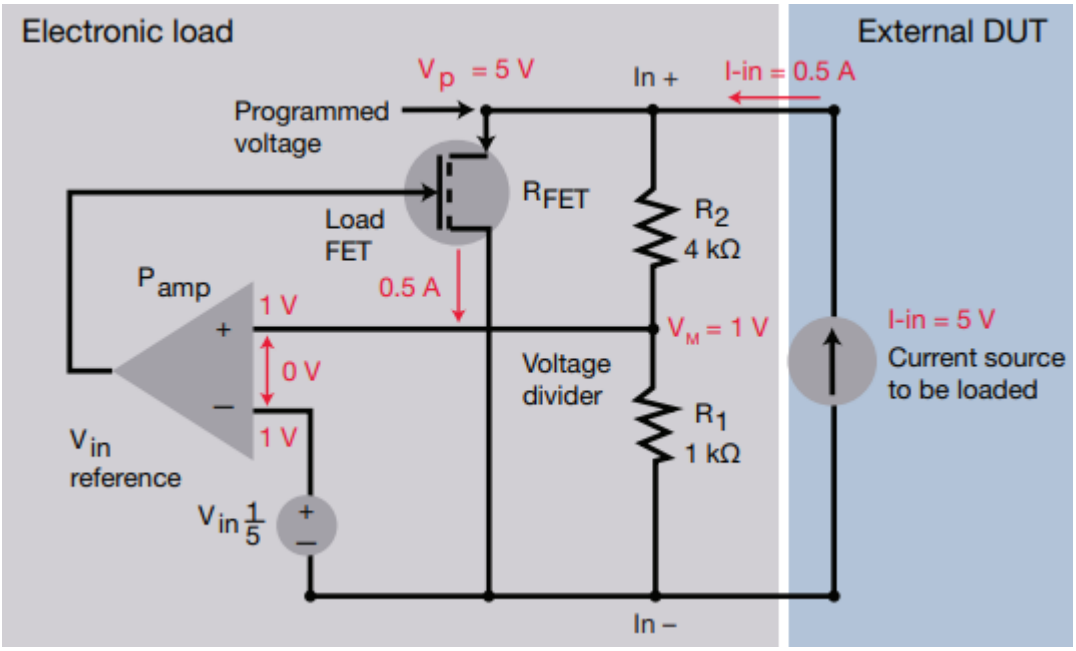
A saída do amplificador controla a tensão de gate do MOSFET. Caso a tensão de entrada V_{in} aumente acima do valor programado, a tensão V_m também aumenta, tornando-se maior que V_{ref}. Como consequência, o amplificador reduz a tensão aplicada ao gate do MOSFET, diminuindo sua condução. Esse aumento da resistência equivalente do MOSFET provoca uma redução da corrente drenada, resultando na queda da tensão até o valor desejado.

De forma análoga, se a tensão V_{in} diminuir, o valor de V_m torna-se inferior a V_{ref}. O amplificador então aumenta a tensão no gate do MOSFET, elevando sua condução e reduzindo sua resistência equivalente, permitindo que uma maior corrente seja drenada e restaurando a tensão de entrada ao valor programado.

Assim, o MOSFET atua como um elemento de dissipação controlada, ajustando continuamente sua resistência dinâmica para compensar variações na corrente fornecida pelo DUT. O equilíbrio do sistema é atingido quando a tensão no ponto de medição V_m se iguala à tensão de referência V_{ref}, garantindo que a tensão nos terminais da carga permaneça constante.

Esse método de controle permite que a carga eletrônica opere de forma estável no modo de tensão constante, sendo amplamente utilizado em testes de fontes de alimentação, baterias e conversores de potência, onde é necessário avaliar o comportamento do sistema sob condições de tensão fixa e corrente variável.

Figura 4 – Circuito de uma carga eletrônica em modo de tensão constante



Fonte: Keysight (2022, p. 5).

2.2 Interface I2C

Segundo WU (2022, tradução noss), "I2C é um protocolo de comunicação serial de dois fios que utiliza uma linha de dados serial (*Serial Data Line* – SDA) e uma linha de clock serial (*Serial Clock Line* – SCL). O protocolo suporta múltiplos dispositivos de destino em um barramento de comunicação e também pode suportar múltiplos controladores que enviam e recebem comandos e dados. A comunicação é enviada em pacotes de *bytes* com um endereço único para cada dispositivo de destino.".

Esta interface permite o endereçamento individual de dispositivos conectados ao mesmo barramento, possibilitando a identificação e o controle de múltiplos periféricos em uma única rede de comunicação. Dessa forma, tornou-se viável o controle de múltiplas cargas eletrônicas utilizando apenas um único barramento I2C, reduzindo a complexidade do sistema e a quantidade de interconexões necessárias.

De acordo com WU (2022, tradução nossa), "o dispositivo controlador pode se comunicar com qualquer dispositivo-alvo através de um único endereço I2C enviado através de um único barramento serial. I2C é simples e econômico para fabricantes de dispotivos implementarem.". Desta forma, a utilização de I2C se mostra uma boa alternativa para realizar o controle das cargas eletrônicas.

A interface I2C não foi inicialmente uma interface de alta taxa de transmissão de dados, sendo apenas 100 kbps. Contudo, com o desenvolvimento de novas tecnologias, a interface pode contar com 5 modos de operação, elevando a veloci-

dade de transmissão para até 5 Mbps, conforme mostra a Tabela 1. Neste trabalho foi utilizado o *Fast Mode Plus* para realizar a comunicação entre as cargas eletrônicas, potenciômetros digitais e o protótipo, este modo foi escolhido por ser o modo de maior frequência suportado pela placa de desenvolvimento, pelas cargas eletrônicas e pelos potenciômetros digitais.

Tabela 1 – Modos de operação I2C

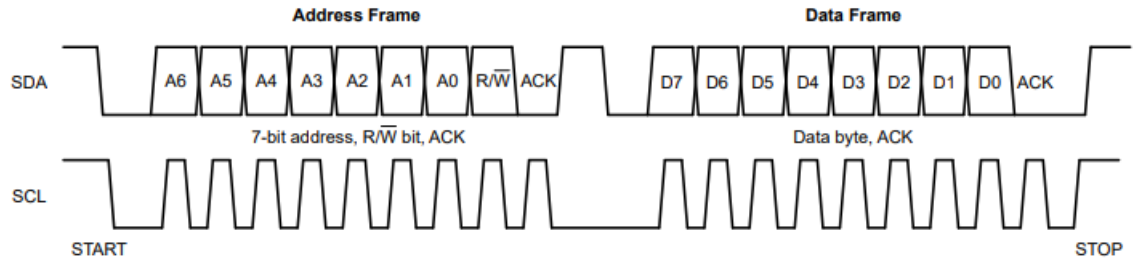
Modo	Velocidade de transmissão máxima
Padrão	100 kbps
Fast	400 kbps
Fast Mode Plus	1 Mbps
High Speed	3.4 Mbps
Ultra-fast	5 Mbps

A interface I2C organiza a comunicação em quadros (*frames*), os quais estruturam a troca de dados entre os dispositivos conectados ao barramento. Toda transmissão é iniciada pelo dispositivo mestre, que gera uma condição de início (*START*), seguida pelo envio de um quadro de endereço. Esse endereço é responsável por identificar o dispositivo escravo (*slave*) que deverá responder à comunicação no barramento.

Após o envio do endereço, o dispositivo mestre transmite um ou mais quadros de dados, sendo cada quadro composto por 1 byte. Ao término de cada quadro, é enviado um bit de reconhecimento, denominado *ACK* (*Acknowledge*) ou *NACK* (*Not Acknowledge*), utilizado para indicar se o dado foi corretamente recebido pelo dispositivo destinatário.

Por fim, a interface I2C define uma condição de parada (*STOP*), que sinaliza o encerramento da comunicação em andamento e libera o barramento para novas transmissões, conforme ilustrado na Figura 5.

Figura 5 – Forma de onda da comunicação I2C



Fonte: Texas Intruments (2022, p. 8).

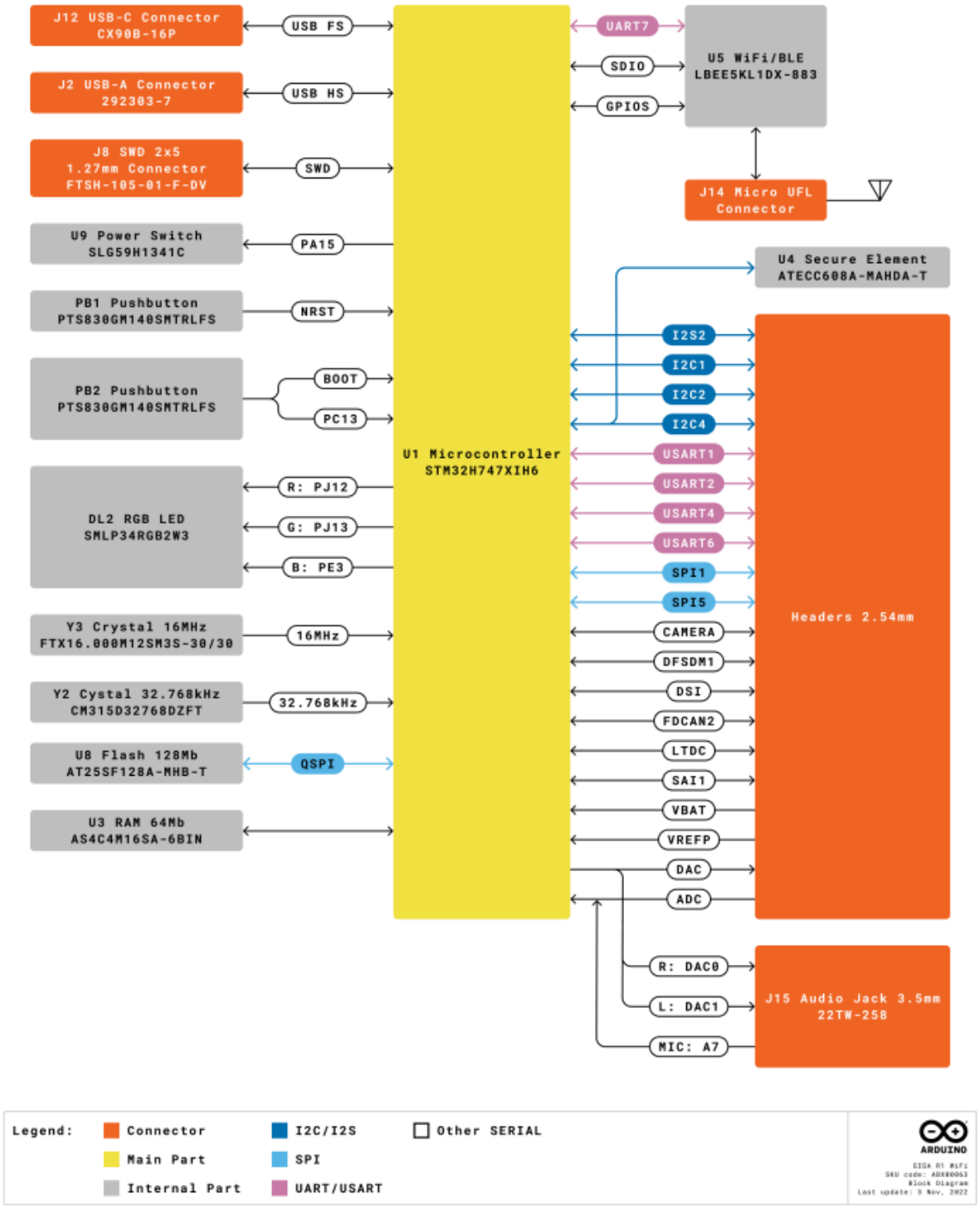
2.3 Placa de desenvolvimento Arduino Giga R1

Uma das premissas iniciais deste projeto foi a redução do custo total do sistema. Dessa forma, buscou-se inicialmente uma solução comercial pronta, de baixo custo e de fácil aquisição no mercado. Nesse contexto, a primeira alternativa considerada foi o Arduino Mega 2560 e o *hardware* do protótipo foi desenvolvido considerando o Arduino Mega 2560. Entretanto, em razão de limitações técnicas identificadas durante o desenvolvimento, as quais serão discutidas adiante, optou-se pela substituição da plataforma pelo Arduino Giga R1. Esse modelo mantém compatibilidade de *pinout* com o Arduino Mega 2560, porém apresenta melhorias significativas em termos de desempenho e recursos disponíveis.

O Arduino Giga R1 é baseado em uma arquitetura de duplo núcleo, composta por um processador principal ARM Cortex-M7 e um processador secundário ARM Cortex-M4, ambos com arquitetura de 32 bits. A placa dispõe de 1 MB de memória SRAM (*Static Random Access Memory*), 2 MB de memória Flash interna e 16 MB de memória Flash externa. Além disso, oferece 76 pinos de entrada e saída digital, 14 entradas analógicas e 2 saídas analógicas, ampliando consideravelmente as possibilidades de interfaceamento com dispositivos externos.

A Figura 6 apresenta o diagrama de blocos do Arduino Giga R1, destacando o microcontrolador STM32H747XI e suas conexões com os periféricos embarcados na placa. À esquerda do diagrama são mostradas as interfaces físicas e circuitos auxiliares, como portas USB, interface SWD, botões, LEDs, osciladores e memórias externas. Na região central encontram-se os principais barramentos e periféricos internos do microcontrolador, incluindo I2C, SPI, UART, ADC, DAC, LTDC, DSI e interfaces para câmera. À direita estão representados os *headers* com espaçamento de 2,54 mm, conexões de áudio e o módulo de comunicação sem fio Wi-Fi/BLE.

Figura 6 – Diagrama de blocos do Arduino Giga R1



Fonte: Arduino (2025, p. 8).

A placa de desenvolvimento é capaz de operar no modo *Fast Mode Plus* com I2C, com isso, foi possível gerar as quatro formas de onda nas frequências de 500 Hz até 1500 Hz. Além da velocidade de transmissão de dados maior, outras vantagens surgiram com a troca para este dispositivo, uma delas sendo a possibilidade de utilizar

o microprocessador secundário para controlar um *display* que indica o status da carga.

A comunicação entre núcleos é realizada através de RPCs (*Remote Procedure Calls*), uma função que permite a troca de dados entre núcleos. Segundo a ARDUINO (2025, tradução nossa), "Estes dois núcleos podem executar aplicações em paralelo, por exemplo, controlar um servo motor em um núcleo e um display no outro, sem que ocorra nenhum tipo de bloqueio".

O sistema também conta com memória não-volátil, suportando os sistema de arquivos LittleFS e FATFS, no escopo deste trabalho foi utilizado o LittleFS. A placa de desenvolvimento também conta com módulo de rádio Murata LBEE5KL1DX-883, que não foi utilizados neste projeto.

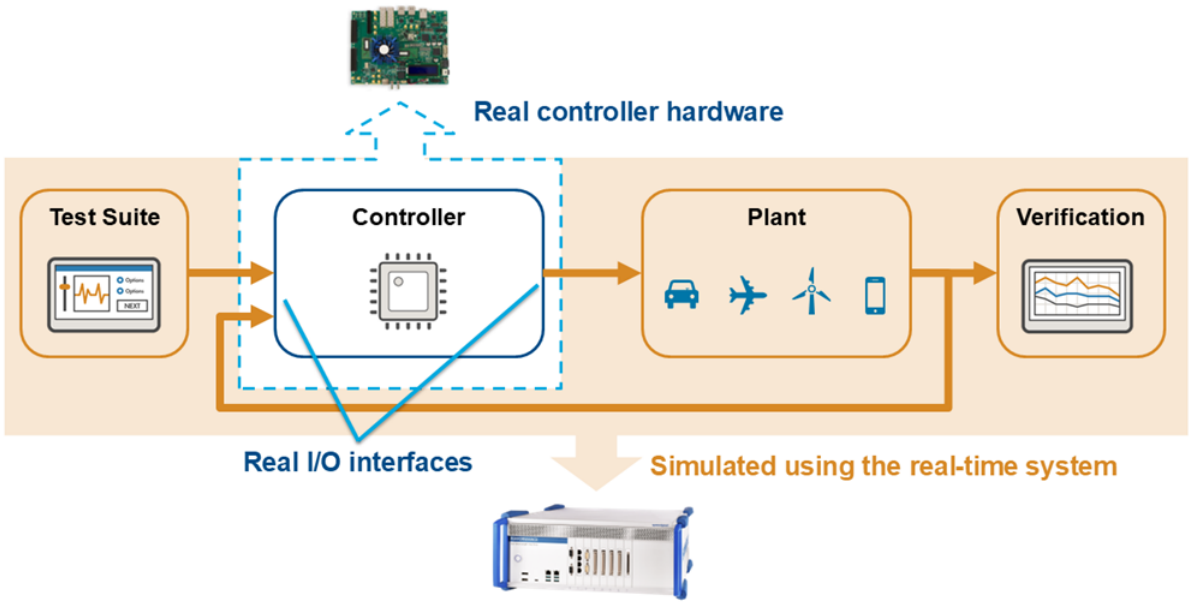
Estas características consolidam a escolha desta placa de desenvolvimento como uma alternativa viável para execução deste projeto, apesar de conter múltiplas outras características não listadas neste trabalho e de não terem sido utilizadas todas, o Arduino Giga R1 WiFi se mostra uma placa de desenvolvimento de bom desempenho e com múltiplos campos de aplicação.

2.4 Hardware-in-the-loop

Hardware-in-the-loop (HIL) é uma técnica utilizada para validação de sistemas embarcados que necessitam de algum tipo de sistema físico para realizar o controle para validar o seu funcionamento. De acordo com MATHWORKS(2025, tradução nossa), "Testes usando hardware-in-the-loop são usados para validar integrações *hardware-software* e é uma parte do processo de certificação nas indústrias aeroespaciais, automotivas e outras.".

Para implementar uma abordagem usando *hardware-in-the-loop*, primeiro é implementado o algoritmo de controle no *hardware* real e em sequência são conectadas as entradas e saídas ao simulador em tempo real, geralmente o simulador se trata de um sistema embarcado capaz de gerar sinais físicos semelhantes ao do sistema real, que representa o sistema físico, como demonstrado na Figura 7. Em seguida, configura-se o modelo matemático do sistema controlado, define-se os cenários de teste (como falhas, variações de carga e condições extremas) e ajustam-se as interfaces de comunicação do simulador, garantindo que sinais analógicos, digitais ou protocolos como CAN, LIN, TCP/UDP ou I2C sejam transmitidos com os mesmos tempos e características presentes no mundo real.

Figura 7 – Diagrama de um sistema com HIL



Fonte: MathWorks (2025).

A Figura 7 ilustra a arquitetura típica de um sistema *Hardware-in-the-Loop* (HIL), evidenciando a separação entre o controlador real e a planta simulada em tempo real. No lado esquerdo encontra-se o *Test Suite*, responsável por definir e executar os cenários de ensaio. Esse módulo pode incluir scripts automatizados, sequências de teste e ferramentas de análise, permitindo a aplicação sistemática de estímulos ao sistema sob avaliação.

No centro da figura está o *Controller*, que representa o *hardware* real do controlador embarcado, por exemplo, uma ECU automotiva ou microcontrolador programado com o algoritmo de controle desenvolvido. Esse controlador é fisicamente idêntico ao que será utilizado na aplicação final, sendo testado em condições próximas às reais. As conexões entre o controlador e o simulador ocorrem por meio de interfaces de entrada e saída reais (*Real I/O interfaces*), destacadas na parte inferior da figura. Essas interfaces podem envolver sinais analógicos, digitais ou protocolos de comunicação, garantindo fidelidade elétrica e temporal.

À direita do controlador encontra-se a *Plant*, que representa o sistema físico a ser controlado (como um veículo, aeronave, turbina eólica ou dispositivo eletrônico). No contexto HIL, essa planta não existe fisicamente durante o ensaio; em vez disso, seu comportamento é reproduzido matematicamente e executado em um sistema de processamento em tempo real, indicado na parte inferior da figura como o equipamento responsável pela simulação. Esse simulador calcula, a cada passo de tempo, as respostas do modelo matemático e gera sinais elétricos correspondentes às saídas

do sistema físico real.

Por fim, o bloco de *Verification* representa o processo de monitoramento e análise dos resultados obtidos. Nessa etapa, são avaliadas variáveis internas, sinais de controle e respostas do sistema, permitindo verificar o desempenho do algoritmo implementado no hardware real diante dos diferentes cenários impostos pelo *Test Suite*.

Observa-se que a principal característica do HIL, evidenciada na figura, é a substituição da planta física por um modelo computacional executado em tempo real, mantendo o controlador como hardware real. Essa configuração possibilita a validação segura, repetível e controlada do sistema de controle antes da integração com o sistema físico definitivo, reduzindo riscos, custos e tempo de desenvolvimento.

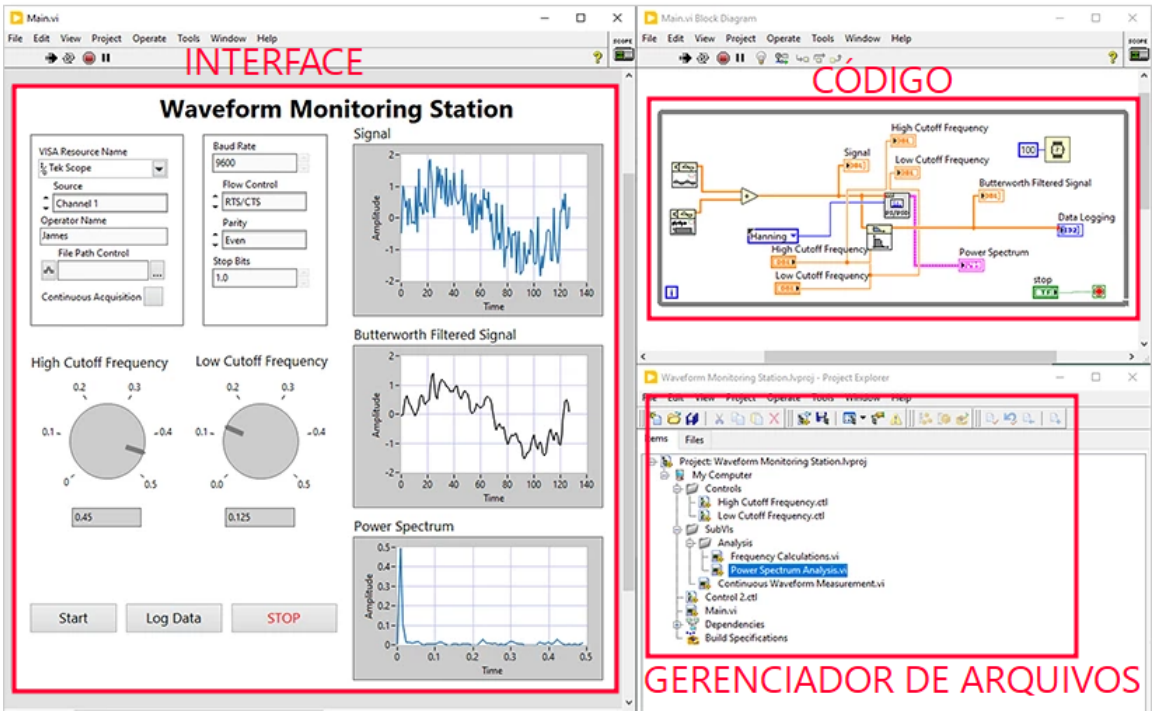
Esta abordagem foi implementada como representado na Figura 7, com uma interface física para conexão com o sistema de controle e uma interface para acesso aos sinais.

2.5 LabVIEW

O LabVIEW é um ambiente de programação gráfica desenvolvido pela National Instruments, amplamente utilizado para aquisição de dados, controle e automação de sistemas. A plataforma apresenta elevada compatibilidade com os equipamentos da própria fabricante, permitindo a integração direta com módulos de instrumentação e sistemas de aquisição. Por meio de sua abordagem baseada em fluxo de dados, o LabVIEW possibilita o controle, a configuração e o monitoramento programático de múltiplos dispositivos de forma estruturada e eficiente. LabVIEW é uma solução amplamente utilizada em indústrias aeroespaciais, automobilísticas, defesa, entre outros. De acordo com a National Instruments (2025, tradução nossa), "LabVIEW é um ambiente de programação gráfica que fornece aceleradores de produtividade únicos para o desenvolvimento de sistemas de testes.".

Assim como ilustrado na Figura 8, é possível realizar a programação e tratar de elementos de interface de usuário simultâneamente. LabVIEW também conta com um sistema para gerenciamento de arquivos que são utilizados no projeto.

Figura 8 – Exemplo de aplicação usando LabVIEW



Fonte: National Instruments (2025).

LabVIEW apresenta uma rápida velocidade de desenvolvimento e possibilidade de desenvolver a inteface gráfica da aplicação em conjunto com o *backend* em um único local. A linguagem de programação utilizada é G, uma linguagem gráfica de alto nível, National Instruments (2025).

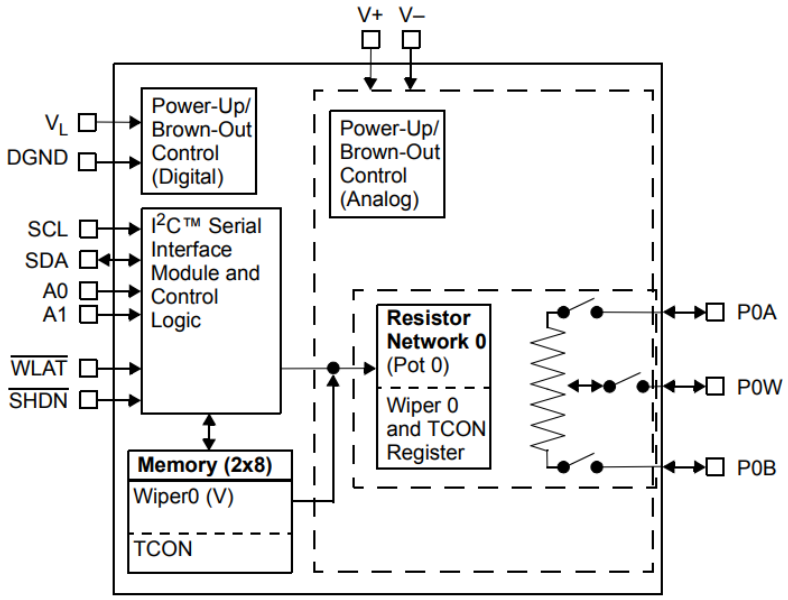
Outras caracteríscas importantes do LabVIEW são o paralelismo intrínseco e o fato do código ser compilado a cada alteração que o usuário realiza, permitindo que o usuário visualize um erro de programação no mesmo instante que ele acontece.

A utilização do LabVIEW permitiu desenvolver uma interface para o usuário para definir os parâmetros que serão aplicados nas cargas eletrônicas e potenciômetros digitais. Além disso, todas as funcionalidades do *software* também foram implementadas usando LabVIEW, ou seja, a comunicação serial, a geração do arquivo de configuração, importar e exportar configurações foram implementadas utilizando apenas LabVIEW.

2.6 Potenciômetros digitais

Os potenciômetros digitais são dispositivos eletrônicos desenvolvidos para substituir os potenciômetros mecânicos tradicionais por versões totalmente controladas por sistemas digitais. O potenciômetro digital opera por meio de uma rede interna de resistores fixos, associados a chaves eletrônicas que selecionam pontos específicos dessa rede.

Figura 9 – Diagrama de bloco do potenciômetro digital MCP45HV51



Fonte: Microchip (2025).

A Figura 9 apresenta o diagrama funcional de um potenciômetro digital com interface de comunicação I2C. O dispositivo é composto por três blocos principais: lógica de controle digital, memória interna e rede resistiva programável.

No lado esquerdo, observa-se o bloco de alimentação e controle digital. Os pinos V_L e *DGND* alimentam a lógica interna. O módulo “Power-Up/Brown-Out Control (Digital)” é responsável por garantir a inicialização correta do sistema e proteger contra quedas de tensão (*brown-out*).

A comunicação ocorre por meio da interface I2C, utilizando as linhas *SCL* (*Serial Clock*) e *SDA* (*Serial Data*). Os pinos *A*0 e *A*1 permitem a configuração do endereço do dispositivo no barramento. O bloco “I2C Serial Interface Module and Control Logic” decodifica os comandos recebidos e controla os registradores internos.

Os pinos *WLAT* e *SHDN* correspondem, respectivamente, à função de “*write latch*” (armazenamento temporário do valor escrito) e “shutdown” (modo de desligamento).

A memória interna “Memory (2x8)” armazena dois registradores principais: o registrador do cursor (“Wiper0”) e o registrador de controle do terminal (“TCON”). O registrador Wiper0 define a posição do cursor virtual do potenciômetro, enquanto o TCON controla a conexão dos terminais da rede resistiva.

No lado direito do diagrama encontra-se a rede resistiva interna (“Resistor Network 0”). Essa rede é composta por uma cadeia de resistores discretos com múltiplos pontos de derivação (*taps*). A posição do cursor é determinada pelo valor armazenado no registrador Wiper0, ajustando a resistência equivalente entre os ter-

minais.

Os terminais externos são:

- $P0A$ — Terminal superior da rede resistiva;
- $P0B$ — Terminal inferior da rede resistiva;
- $P0W$ — Terminal do cursor (wiper).

A alimentação analógica (V^+ e V^-) está associada ao bloco “Power-Up/Brown-Out Control (Analog)”, que supervisiona a parte analógica do dispositivo.

Em síntese, o dispositivo funciona como um potenciômetro convencional, porém com ajuste digital da posição do cursor por meio da interface I2C, permitindo controle programático da resistência equivalente entre seus terminais.

Potenciômetros mecânicos também são propensos a sofrerem por condições como temperaturas extremas, umidade, entre outros. Segundo a Microchip Technology Inc. (2000, tradução nossa), "como potenciômetros digitais são confeccionados usando um processo CMOS (*Complementary Metal-Oxide-Semiconductor*) sem partes móveis, os efeitos destas mudanças são reduzidas significativamente".

Neste trabalho foi utilizado o potenciômetro digital MCP45HV51, que é controlado usando a interface I2C e possui uma resistência máxima de até 5 kohm.

2.7 NI PXIe-8135 e NI PXI-5152

Neste trabalho foram empregados dois equipamentos principais para a realização das medições dos sinais gerados: um chassi no padrão PXIe (*PCI eXtensions for Instrumentation*) e um módulo digitalizador acoplado a esse chassi.

O padrão PXIe é uma plataforma modular amplamente utilizada em sistemas de teste, medição e automação industrial. Ele combina a arquitetura de comunicação *PCI Express* com recursos específicos para instrumentação, tais como sincronização de alta precisão, linhas dedicadas de *trigger* e comunicação determinística entre módulos. Essas características tornam o PXIe especialmente adequado para aplicações que exigem alto desempenho e temporização rigorosa.

Os sistemas PXIe são compostos por um chassi com múltiplos *slots*, interligados por um *backplane* capaz de distribuir sinais de clock e disparo entre os módulos instalados. Além disso, o sistema geralmente incorpora um controlador embarcado responsável pelo processamento e gerenciamento das tarefas de aquisição e análise de dados. Neste contexto, foi utilizado o NI PXIe-8135 como controlador do sistema. Desenvolvido pela National Instruments, esse módulo é projetado para aplicações que demandam elevado desempenho computacional, baixa latência e robustez em ambientes industriais.

Para a aquisição dos sinais, foi empregado o NI PXI-5152, um digitalizador de alta velocidade com dois canais independentes. Esse equipamento oferece taxa de amostragem de até 2 GS/s e resolução vertical de 8 bits, possibilitando a captura de sinais rápidos com boa fidelidade temporal. Tais características o tornam adequado para medições de radiofrequência moderada, análise de sinais digitais de alta velocidade, observação de fenômenos transitórios e realização de testes elétricos no setor automotivo.

Dessa forma, a utilização do NI PXIe-8135 como plataforma de controle e do NI PXI-5152 como módulo de aquisição permitiu a implementação de um *setup* experimental robusto, adequado à depuração e à análise detalhada dos sinais de controle gerados.

2.8 Resumo

Resumindo todas as tecnologias e técnicas aplicadas neste trabalho.

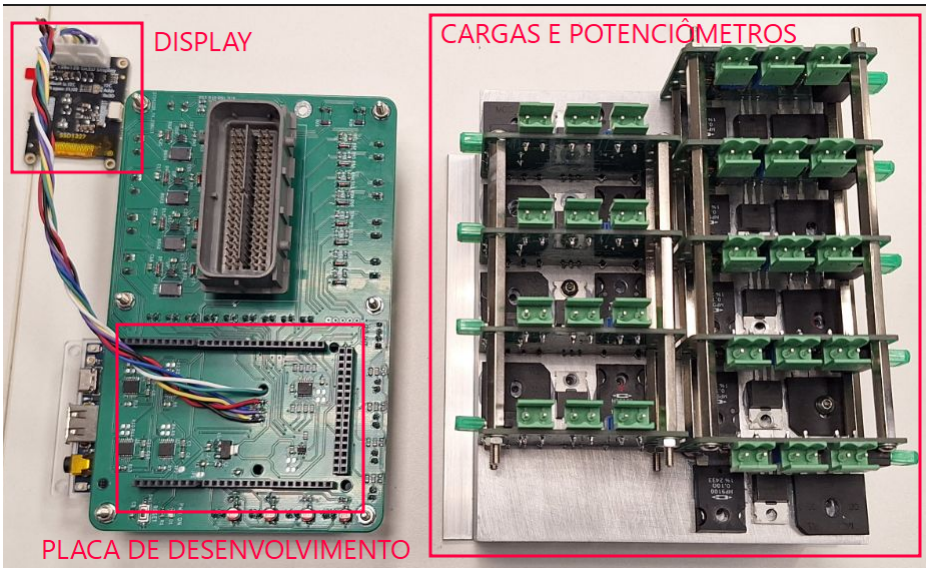
- O tipo de carga eletrônica utilizada foi a de modo de corrente constante para que fosse possível gerar uma forma de onda de corrente.
- O potenciômetro digital utilizado foi o MCP45HV51.
- A comunicação entre a placa de desenvolvimento e os potenciômetros digitais e cargas eletrônicas é realizada por meio da interface I2C.
- O Arduino Giga R1 WiFi foi selecionado devido à sua capacidade de trabalhar com velocidades maiores de I2C e por possuir o mesmo *pinout* do Arduino Mega 2560.
- O *software* LabVIEW permitiu desenvolver as funcionalidades e a interface do usuário simultâneamente.
- HIL foi a abordagem definida para testar o sistema de controle DMM.

3 METODOLOGIA

Este trabalho tem como objetivo desenvolver um protótipo que utiliza uma arquitetura HIL para testes de DMM, simulando formas de onda de corrente de motores DC e variação de potenciômetros. Diante disso o protótipo foi desenvolvido utilizando uma placa de desenvolvimento Arduino Giga R1 WiFi, quatro cargas eletrônica desenvolvidas pela equipe responsável pelo *hardware* da Fundação CERTI, quatro potenciômetros digitais, um display e um *software* LabVIEW.

Foi optado por utilizar uma placa por carga eletrônica e uma placa por circuito do potenciômetro para que fosse possível obter um arranjo vertical dos componentes e possível utilizar apenas um dissipador para todas as placas, como ilustra a Figura 10. Também foi implementada uma placa para emulação do motor responsável pelo rebatimento dos espelhos, este motor, devido às especificações do cliente, pode ser abstraído em apenas uma carga resistiva.

Figura 10 – Protótipo montado e sistema com cargas eletrônicas e potenciômetros digitais



Fonte: Autoria própria (2025).

Neste capítulo serão abordados os métodos utilizados para o desenvolvimento deste protótipo, quais os conhecimentos necessários e a evolução do protótipo, variações no *hardware* e variações de *software*.

3.1 Métodos Aplicados

Inicialmente, foram realizadas provas de conceito para validação das capacidades dos componentes selecionados. Também foi realizada a validação de algumas funcionalidades da placa de desenvolvimento e foram feitos diagramas fun cionais do *firmware* e *software*.

Todo o desenvolvimento do *firmware* deste trabalho foi realizado utilizando a IDE VSCode com a extensão Platformio, disponível na mesma, realizando a gravação do programa via comunicação serial. Para desenvolvimento do *software* foi utilizado o LabVIEW versão 2019. Ambos os ambientes em um computador usando um sistema operacional Windows 10 ou superior para que qualquer desenvolver com acesso ao *software* consiga realizar a configuração da carga.

3.1.1 Teste dos potenciômetros digitais

Uma das primeiras provas de conceito realizadas foi o controle dos potenciômetros digitais via I2C, utilizando o CI MCP45HV51. Foram feitos testes para validação da capacidade de controle dos potenciômetros, bem como uma primeira versão de *software* que permitisse definir seus parâmetros de variação.

Alguns testes foram realizados em ordem de validar o tempo de resposta do potenciômetro, o primeiro teste foi referente ao tempo necessário para atualização dos quatro potenciômetros do valor inicial até o valor final de forma incremental. O seguinte código foi utilizado para realizar o teste.

```
#include <Arduino.h>
#include <Wire.h>
#include <CRC32.h>
// I2C addresses (0x7C >> 1 = 0x3E etc.)
const byte potAddresses[4] = { 0x3C, 0x3D, 0x3E, 0x3F };
const byte cmdWriteWiper = 0x00; // Write to wiper 0
int wiperPos = 0;
unsigned long lastMicros = 0;
const unsigned long updateMicrosInterval = 100; // Time between updates
bool sweepDone = false;
unsigned long startTime = 0;
unsigned long endTime = 0;
void setup() {
  Wire.begin();
  Wire.setClock(400000); // Set I2C speed
  Serial.begin(115200);
  while (!Serial);
  // Initialize all potentiometers to 0
  for (int i = 0; i < 4; i++) {
    writeWiper(potAddresses[i], 0);
  }
  startTime = micros();
  Serial.print("First Update - Time (us): ");
  Serial.print(startTime);
  Serial.print(" | Wiper Value: ");
  Serial.println(wiperPos);
}
```

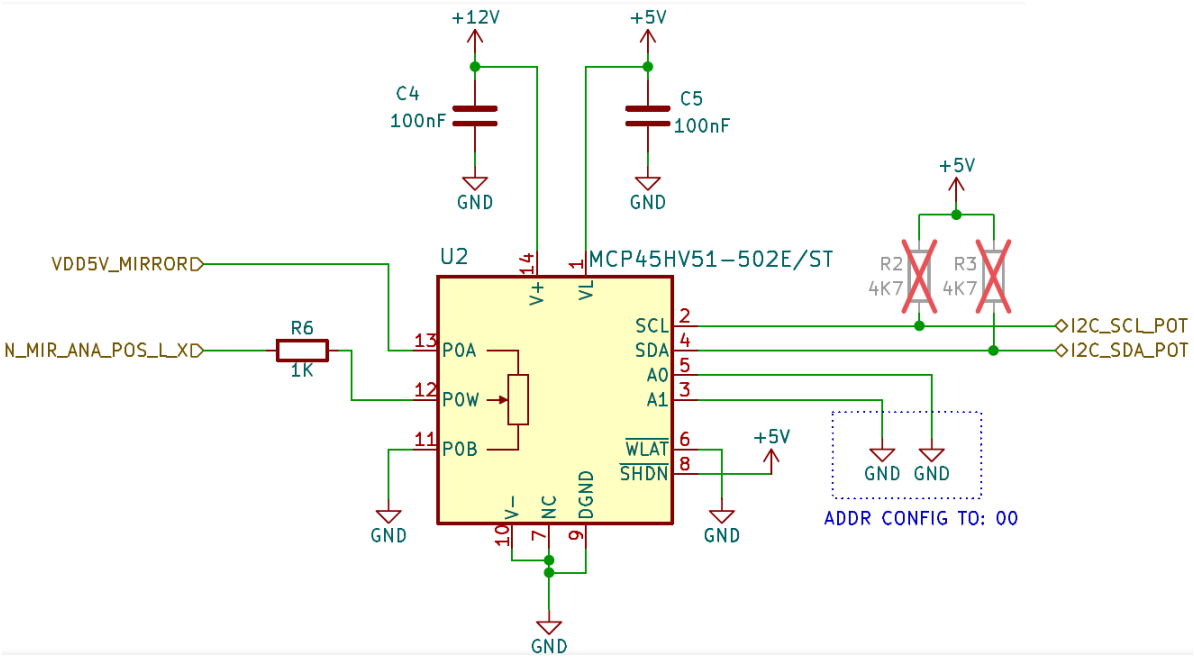
```
void loop() {
  if (!sweepDone) {
    unsigned long now = micros();
    if (now - lastMicros >= updateMicrosInterval) {
      lastMicros = now;
      // Update all 4 potentiometers to the current wiper position
      for (int i = 0; i < 4; i++) {
        writeWiper(potAddresses[i], wiperPos);
      }
      wiperPos++;
      if (wiperPos >= 256) {
        sweepDone = true;
        endTime = micros();
        Serial.print("Last Update  - Time (us): ");
        Serial.print(endTime);
        Serial.print(" | Wiper Value: ");
        Serial.println(wiperPos - 1);
        Serial.print("Total Sweep Duration (us): ");
        Serial.println(endTime - startTime);
        Serial.println("Sweep completed.");
      }
    }
  }
}
void writeWiper(byte address, byte value) {
  Wire.beginTransmission(address);
  Wire.write(cmdWriteWiper);
  Wire.write(value);
  Wire.endTransmission(false); // Repeated start for speed
}
```

As condições de teste utilizadas foram um clock i2c de 400 kHz e um baud rate de 500 kHz para a comunicação serial com o computador. A partir disso foram obtidos os seguintes resultados:

```
First Update - Time (us): 460 | Wiper Value: 0
Last Update  - Time (us): 115252 | Wiper Value: 255
Total Sweep Duration (us): 114792
Sweep completed.
```

Com base nos resultados obtidos, são necessários, em média, 30ms para incrementar um potenciômetro do valor mínimo até o seu valor máximo. Desta forma, pôde-se validar a capacidade do sistema de emular o circuito físico que já era utilizado com precisão, usando o circuito ilustrado pela Figura 11.

Figura 11 – Esquemático do circuito do potenciômetro digital



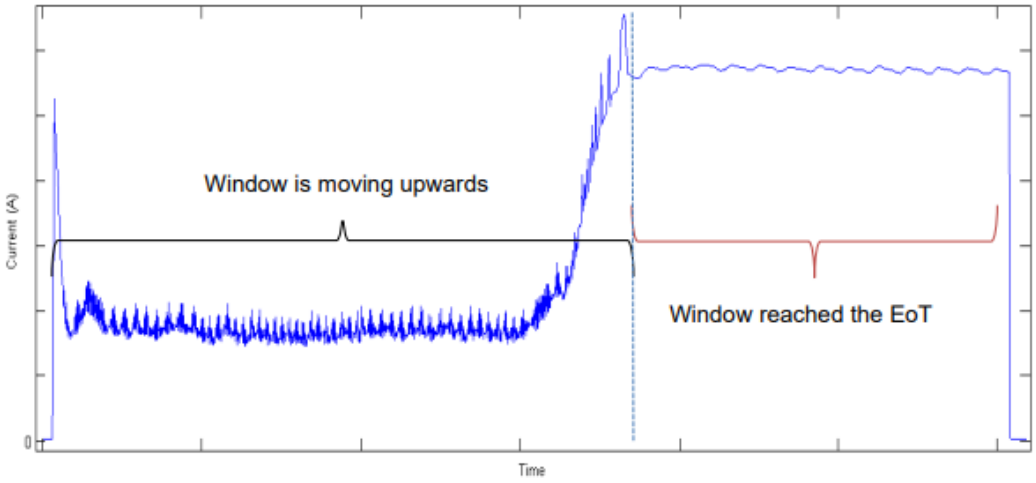
Fonte: Autoria própria (2025).

Do ponto de vista de *firmware*, os potenciômetros foram abstraídos em uma única classe que realiza seu controle individual de cada potenciômetro, dessa forma, com uma classe e 4 objetos, foi possível simular o acionamento de todos os espelhos do veículo.

3.1.2 Análise da forma de onda dos motores DC

Para definir como seria o sistema de geração de forma de onda, foi realizado uma análise das formas de onda de corrente dos motores utilizados nas janelas como ilustrado na Figura 12, esta figura é referente aos ensaios dos motores DC realizados pela equipe da Magneti Marelli.

Figura 12 – Forma de onda dos motores DC



Fonte: Magneti Marelli (2025).

Considerando a forma de onda dos motores utilizados ilustrados pela Figura 12, é possível observar algumas das características da mesma. Observa-se o pico referente à quebra da inércia do motor, os picos que caracterizam o movimento, um valor de regime quando em movimento e um valor de pico que sinaliza o final do movimento do motor.

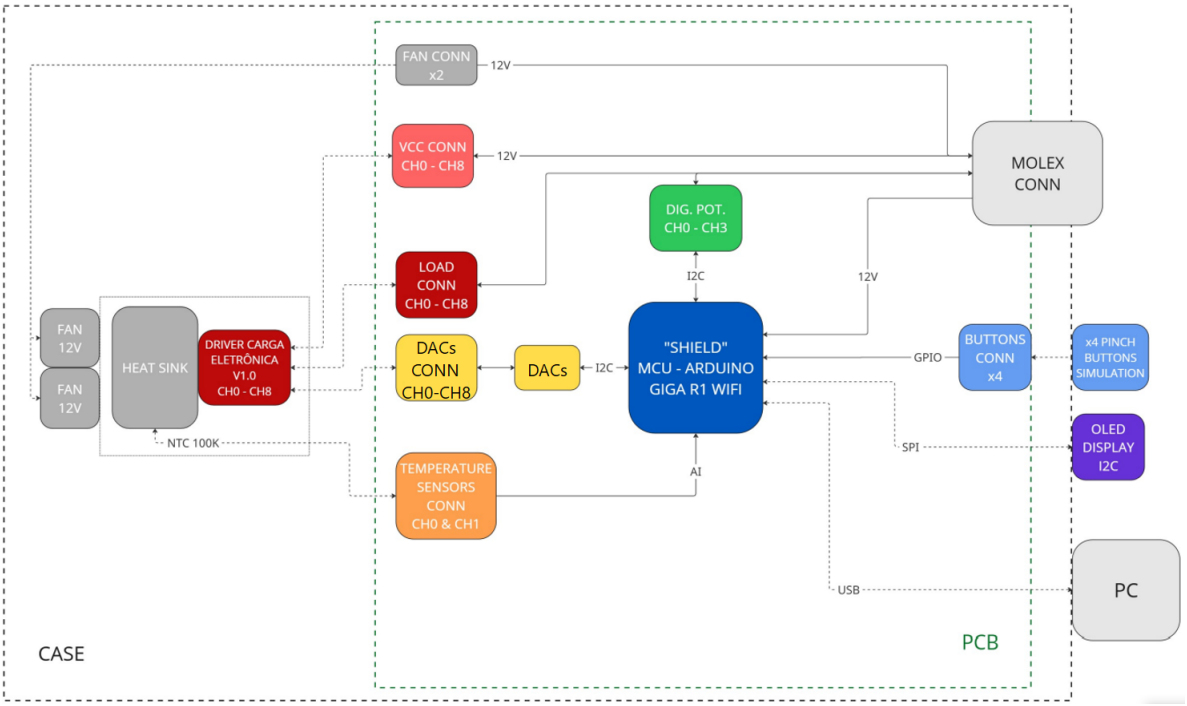
Com base nessas definições iniciais, estabeleceu-se no *software* que os parâmetros configuráveis pelo usuário seriam: corrente de pico inicial, corrente em regime permanente e número de ciclos.

Entretanto, após reuniões com a equipe de desenvolvimento de BCMs da Magneti Marelli, concluiu-se que o pico inicial não teria aplicação prática no projeto e que também não seria necessário implementar uma rampa de crescimento até o pico final. Dessa forma, a especificação da forma de onda foi simplificada. Definiu-se que a placa de desenvolvimento deveria gerar uma onda quadrada, com frequência ajustável entre 500 Hz e 1500 Hz, e capacidade de fornecer corrente de até 10 A.

3.1.3 Arquitetura protótipo

O circuito responsável pela leitura da tensão de entrada e pelo ajuste da corrente da carga eletrônica foi projetado de modo que o valor de corrente seja definido digitalmente pela *MCU*, por meio de um conversor DAC, conforme ilustrado na Figura 13. Dessa forma, o controle da carga é realizado de maneira programática, permitindo ajuste preciso, repetibilidade nos testes e integração com a estratégia de *hardware-in-the-loop* proposta no trabalho.

Figura 13 – Diagrama de blocos do sistema



Fonte: Autoria própria (2025).

A figura apresenta o diagrama em blocos do protótipo desenvolvido, evidenciando a separação física e funcional entre o *case*, onde se encontra o estágio de potência, e a PCB, responsável pelo controle e processamento do sistema. Essa divisão foi adotada com o objetivo de isolar os elementos de alta dissipação térmica da eletrônica sensível de controle, caracterizando uma arquitetura modular.

No *case* está localizado o bloco de potência, composto pelo Driver de Carga Eletrônica V1.0 (CH0–CH8), pelo dissipador de calor, pelas ventoinhas de 12 V e pelo sensor térmico NTC de 100 kΩ. O driver é responsável por controlar a corrente aplicada às cargas, permitindo ajuste individual por canal e viabilizando a simulação programática de motores DC utilizados no desenvolvimento do BCM. Os sinais analógicos provenientes dos DACs definem o valor de corrente conduzido pelos dispositivos de potência.

Devido à operação com correntes de até 10 A, há significativa dissipação de potência nos MOSFETs do estágio de carga. Para garantir operação segura, foi empregado um dissipador de calor com ventilação forçada. O NTC acoplado ao dissipador permite o monitoramento contínuo da temperatura, possibilitando a implementação de proteção térmica via *firmware*.

A PCB concentra o sistema de controle, cujo núcleo é a placa Arduino Giga R1 WiFi. A placa de desenvolvimento executa o *firmware* desenvolvido em C++, sendo responsável pela geração dos sinais de controle, configuração dos DACs, ajuste dos

potenciômetros digitais, leitura dos sensores de temperatura, gerenciamento da interface com o usuário e comunicação com o computador.

Os conversores digital-analógico (DACs CH0–CH8), conectados via barramento I2C, geram os níveis de tensão que determinam a corrente em cada canal da carga eletrônica, estabelecendo a interface entre o domínio digital e o estágio analógico de potência. Os potenciômetros digitais (CH0–CH3), também controlados por I2C, permitem ajustes programáveis de parâmetros do circuito, substituindo ajustes manuais e possibilitando reconfiguração dinâmica.

A interligação entre a PCB e o estágio de potência é realizada por meio dos conectores de carga (CH0–CH8) e de alimentação (VCC CH0–CH8), responsáveis pelo transporte dos sinais de controle e pela distribuição de 12 V aos canais. Sensores de temperatura adicionais podem ser conectados por meio do conector dedicado (CH0 e CH1), sendo lidos pelas entradas analógicas da placa de desenvolvimento.

A interface com o usuário é composta por botões conectados a pinos *GPIO* e por um display OLED com comunicação I2C, utilizado para exibição de parâmetros operacionais, estados do sistema e alarmes térmicos. A alimentação principal do sistema é realizada por um conector do tipo Molex, responsável pela distribuição de 12 V aos diferentes blocos.

A comunicação com o computador é realizada via USB, permitindo configuração de parâmetros, monitoramento em tempo real, atualização de *firmware* e integração com o ambiente de testes, como o LabVIEW.

Essa arquitetura possibilita a simulação programática de motores DC empregados em módulos BCM, com geração de formas de onda controladas entre 500 Hz e 1500 Hz, controle de corrente de até 10 A e monitoramento térmico em tempo real.

3.1.4 Funcionamento das cargas eletrônicas

Com base na fundamentação teórica apresentada na Seção 2.1.1, a carga eletrônica desenvolvida neste trabalho opera em modo de corrente constante, utilizando um MOSFET IRFIZ48GPBF como elemento de controle em região linear e um amplificador operacional para implementar a malha de realimentação negativa.

No circuito implementado (Figura 14), a corrente drenada da fonte sob teste percorre o MOSFET e, em seguida, o resistor de sensoriamento R_{shunt} (R12), cujo valor é de aproximadamente 0,1 Ω. De acordo com a Lei de Ohm,

$$V_{shunt} = I_{load} \cdot R_{shunt} \tag{7}$$

Assim, para uma corrente de 5 A, a queda de tensão no resistor de shunt

é:

$$V_{shunt} = 5 \cdot 0{,}1 = 0{,}5 \text{ V} \tag{8}$$

Essa tensão é aplicada à entrada inversora do amplificador operacional responsável pelo controle de corrente. Na entrada não inversora é aplicada a tensão de referência proveniente do DAC da placa de desenvolvimento, após passar por um estágio *buffer* e por um divisor resistivo composto por dois resistores de 10 kΩ.

O divisor resistivo possui razão:

$$\frac{R_2}{R_1 + R_2} = \frac{10k}{10k + 10k} = \frac{1}{2} \tag{9}$$

Portanto, a tensão de referência efetivamente aplicada ao amplificador operacional é:

$$V_{ref} = \frac{V_{DAC}}{2} \tag{10}$$

A condição de equilíbrio da malha de realimentação ocorre quando:

$$V_{shunt} = V_{ref} \tag{11}$$

Substituindo as expressões anteriores:

$$I_{load} \cdot R_{shunt} = \frac{V_{DAC}}{2} \tag{12}$$

Isolando a corrente:

$$I_{load} = \frac{V_{DAC}}{2 \cdot R_{shunt}} \tag{13}$$

Considerando $R_{shunt} = 0{,}1\ \Omega$, obtém-se:

$$I_{load} = \frac{V_{DAC}}{0{,}2} \tag{14}$$

$$I_{load} = 5 \cdot V_{DAC} \tag{15}$$

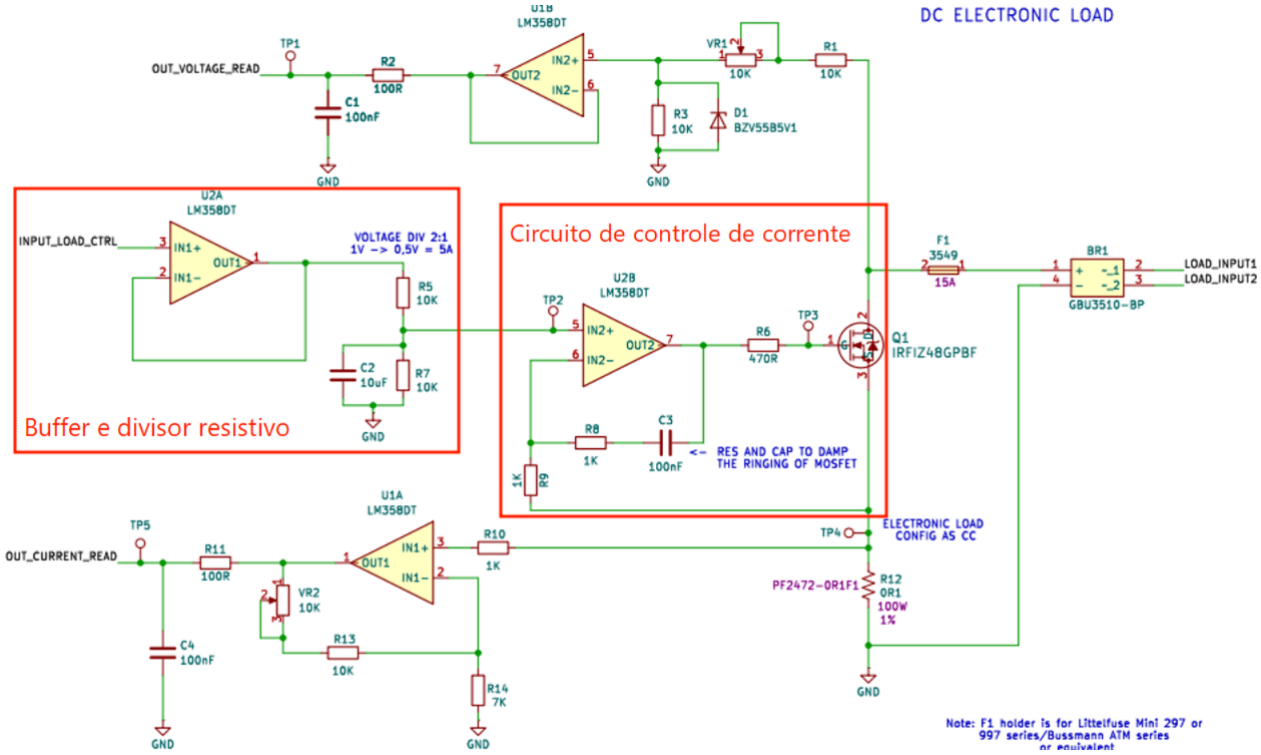
Observa-se, portanto, que o circuito estabelece uma relação linear entre a tensão de saída do DAC e a corrente drenada pela carga eletrônica, com ganho de aproximadamente 5 A/V. Dessa forma, uma tensão de 1 V na saída do DAC resulta em aproximadamente 5 A na carga.

O amplificador operacional ajusta dinamicamente a tensão de gate do MOSFET de modo a manter a igualdade entre V_{shunt} e V_{ref}, compensando variações na

tensão de entrada da fonte sob teste. Assim, mesmo que a tensão da bateria varie entre 11 V e 15 V, a corrente permanece constante.

Dessa forma, o circuito implementa, na prática, o modelo teórico apresentado na literatura, convertendo a tensão de referência digital definida pelo *software* em um valor de corrente controlado por meio de uma malha analógica em tempo real.

Figura 14 – Circuito das cargas eletrônicas



Fonte: Autoria própria (2025).

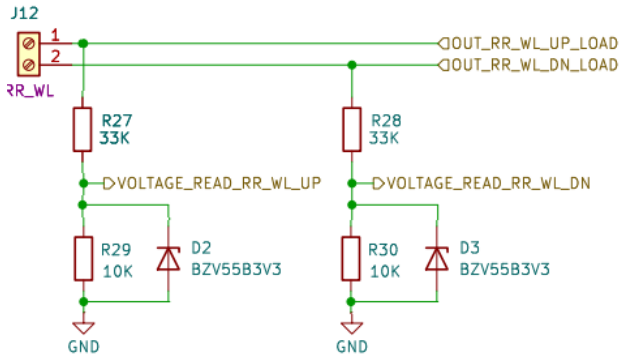
A mudança do sentido do movimento das janelas (que definem o sentido de rotação dos motores DC) é detectada pela inversão do valor de tensão na entrada das cargas eletrônicas. Para garantir que as entradas da placa de desenvolvimento não fossem afetadas, foi utilizado o circuito ilustrado na Figura 15.

Desta forma, para proteger o circuito foi utilizado um divisor resistivo formado por R27 (33 kΩ) e R29 (10 kΩ), responsável por reduzir o nível de tensão proveniente do motor para uma faixa compatível com a entrada analógica da *MCU*. De forma equivalente, o ramo inferior é implementado por R28 (33 kΩ) e R30 (10 kΩ), permitindo a leitura diferencial dos sinais de entrada.

Além da atenuação resistiva, foram inseridos diodos Zener BZV55B3V3 (D2 e D3) com tensão de referência de 3,3 V, conectados entre o ponto de leitura e o terra. Esses dispositivos atuam como proteção contra sobretensão, limitando o valor máximo aplicado às entradas da placa e prevenindo danos em caso de transientes ou

inversões abruptas de polaridade.

Figura 15 – Circuito de identificação de sentido de movimento



Fonte: Autoria própria (2025).

3.1.5 Utilização do Arduino Mega 2560

Um dos pontos principais era realizar o controle das cargas eletrônicas e potenciômetros utilizando uma alternativa acessível e de fácil uso. Com isso sendo um dos requisitos principais solicitados pela Magneti Marelli.

Inicialmente foi utilizado um Arduino Mega 2560 para realizar o controle do sistema, mas a escolha desta placa de desenvolvimento se mostrou problemática para o projeto, uma vez que, o mesmo, era incapaz de gerar as formas de onda nas frequências desejadas, devido ao fato da placa suportar até o modo Fast I2C, o que acarretou na mudança para o Arduino Giga R1 WiFi que possui um *pinout* compatível com o Arduino Mega 2560.

Realizando uma primeira análise, ainda usando o Arduino Mega 2560, com apenas uma das cargas eletrônicas, foi obtida uma forma de onda abaixo do aceitável e quando as quatro cargas eram acionadas simultâneamente, a frequência máxima da onda gerada caía drasticamente.

Para verificar a limitação do sistema, foram realizadas medições ao longo do código do *firmware* implementado. Analisando as formas de ondas obtidas fica notável que a velocidade de processamento e a frequência do clock I2C de 400 kHz do Arduino Mega 2560 não são suficientes para o projeto.

Os principais tempos medidos utilizando o NI PXIe-8135 em conjunto com o NI PXI-5152 foram:

- Tempo para atualizar o valor do DAC: 180 us;
- Tempo para atualizar os valores dos potenciômetros: 90 us

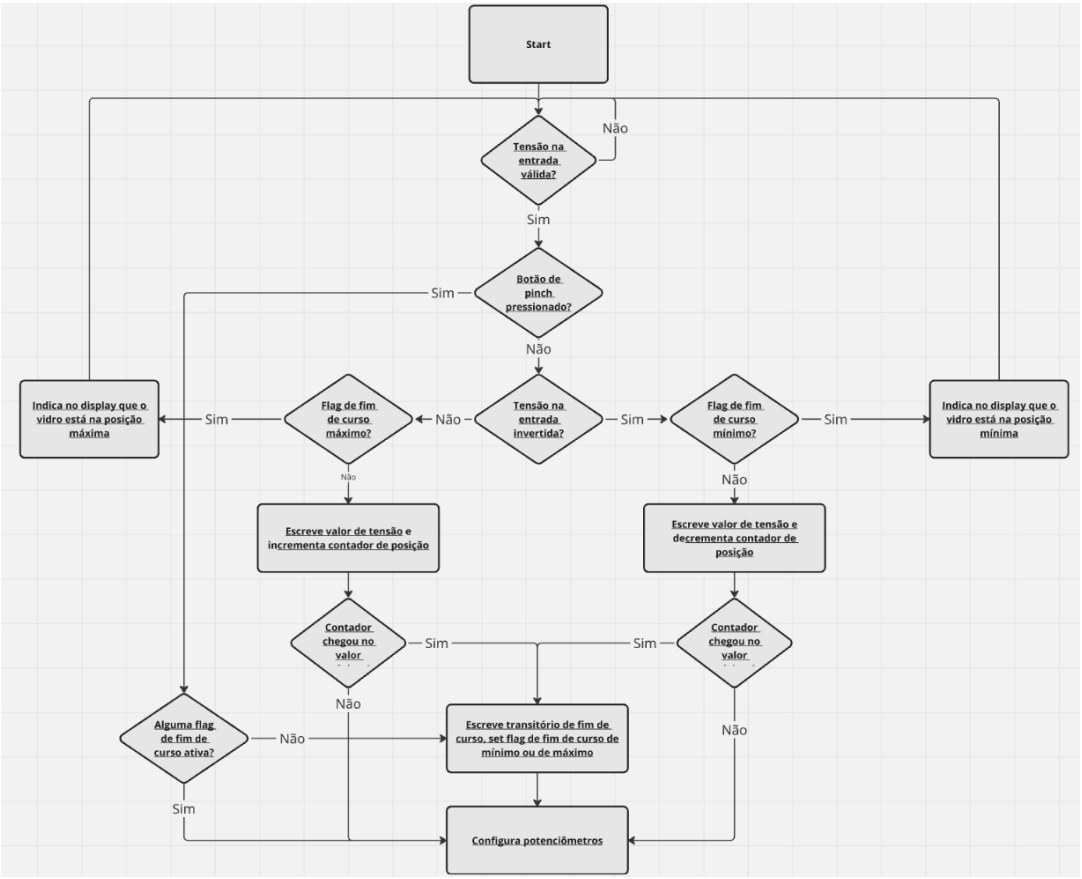
Considerando que o tempo de atualização do DAC (o tempo necessário para a placa enviar o comando e o DAC alterar o seu valor) é 180 us, isso resultaria

em uma onda de no máximo 2770 Hz. Estes valores são apenas para a atualização de um DAC, no sistema atual é necessário a atualização de 4 destes DACs, de forma que a frequência máxima gerada neste cenário seria de 690 Hz.

3.1.6 Arquitetura de software e firmware

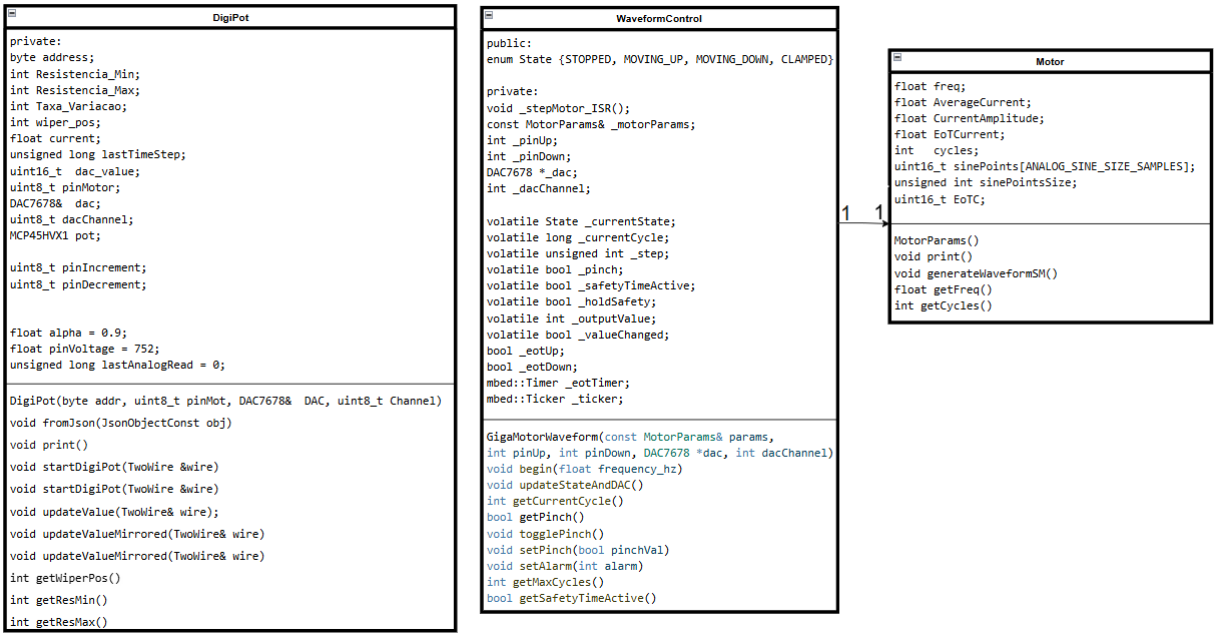
Foi definida um diagrama de funcionamento para realizar o controle da forma de onda da corrente, conforme ilustrado na Figura 16 e no diagrama de classe representado pela Figura 17, com estados que representam a parada do motor, final de curso (*End of Travel*), aperto (*pinch*), que ocorre quando existe alguma coisa travando o motor, e o estado de movimento.

Figura 16 – Diagrama de funcionamento da carga eletrônica



Fonte: Autoria própria (2025).

Figura 17 – Diagrama de classe dos potenciômetros e motores



Fonte: Autoria própria (2025).

Foi definida uma classe para abstração e controle dos potenciômetros digitais, onde cada potenciômetro é um objeto da classe DigiPot. A classe possui métodos para inicialização, atualização e de monitoramento do status do potenciômetro.

A classe Motor, foi implementada para gerar a forma de onda de cada motor com base nos parâmetros configurados pelo usuário, com cada motor sendo um objeto desta classe. A classe também apresenta métodos para verificação da quantidade de ciclos do motor.

Cada objeto da classe Motor é utilizada pela classe WaveformControl para que seja realizado o controle das cargas eletrônicas, a classe implementa uma máquina de estados que se atualiza com base nas tensões de entrada das cargas e dos botões de *pinch*.

Em sequência, foram definidos os formatos de mensagens trocados entre a aplicação LabVIEW e o *firmware*, desta forma definindo um protocolo para que a perda de comunicação fosse devidamente tratada nos dois ambientes, com o *software* indicando para o usuário na interface primária da aplicação e no *firmware* rejeitando mensagens inválidas.

A troca de mensagens entre o protótipo e o *software* LabVIEW é realizada utilizando comunicação serial via USB com um formato e rotina de mensagens pré-definido. As mensagens trocadas são sempre no formato JSON (*JavaScript Object Notation*) pois este se mostra o formato mais adequado para a leitura e interpretação no LabVIEW.

As mensagens foram divididas em dois formatos, o primeiro *handshake*, onde o protótipo envia as informações que estão salvas na sua memória para que o *software* LabVIEW possa apresentar esses dados para o usuário. Enquanto o segundo formato é o de envio de configurações do *software* para o protótipo, onde as configurações enviadas pelo *software* são interpretadas e salvas no protótipo.

O formato do *handshake* e de configuração, desconsiderando os elementos de validação de mensagem, são:

```
Handshake:
{"Payload Type":0,"Payload":"DMM Handshake"}

Configuração:
{"Payload Type":1,"Payload":"{\"FLW\":{\"F\":0,\"A\":0,\"CR\":0,\"CPF\":0,\"NC
    \":0},\"FRW\":{\"F\":0,\"A\":0,\"CR\":0,\"CPF\":0,\"NC\":0},\"RLW\":{\"F\":0,\"
    A\":0,\"CR\":0,\"CPF\":0,\"NC\":0},\"RRW\":{\"F\":0,\"A\":0,\"CR\":0,\"CPF
    \":0,\"NC\":0},\"LMX\":{\"MN\":0,\"MX\":0,\"TV\":0,\"C\":0},\"LMY\":{\"MN
    \":0,\"MX\":0,\"TV\":0,\"C\":0},\"RMX\":{\"MN\":0,\"MX\":0,\"TV\":0,\"C\":0},\"
    RMY\":{\"MN\":0,\"MX\":0,\"TV\":0,\"C\":0}}","HASH":"","ID":"","status_string":
    ""}
```

No protocolo foi especificado um *start byte*, 4 *bytes* para enviar o tamanho do campo de configuração e um CRC32 (*Cyclic redundancy check*) que foi calculado pelo remetente da mensagem, o qual é comparado com o CRC32 que foi calculado pelo destinatário, para validar que a mensagem enviada é a mesma que a mensagem recebida. O padrão é ilustrado na Tabela 2.

Start Byte	Payload Size	Payload	CRC
0xF0	4 bytes	n bytes	4 bytes

Tabela 2 – Estrutura das mensagens

O *software* LabVIEW foi finalizado de forma relativamente rápida, assim que as funcionalidades destinadas ao usuário foram definidas. Entre essas funcionalidades, incluem-se os recursos de salvar e importar configurações, verificar a conexão com a placa de desenvolvimento e disponibilizar algumas ferramentas de apoio para facilitar o uso da interface.

A validação do funcionamento do *software* foi realizada conectando um computador com o *software* instalado ao protótipo via USB e configurando os parâmetros das cargas eletrônicas (motores DC) e dos potenciômetros digitais. Uma configurado, foram utilizados os métodos de monitoramento para verificar que os parâmetros salvos na placa de desenvolvimento foram os mesmos configurados no *software*.

Também foram validadas as funcionalidades de salvar e importar configurações apenas usando o *software*, uma vez que estas funcionalidades estão apenas

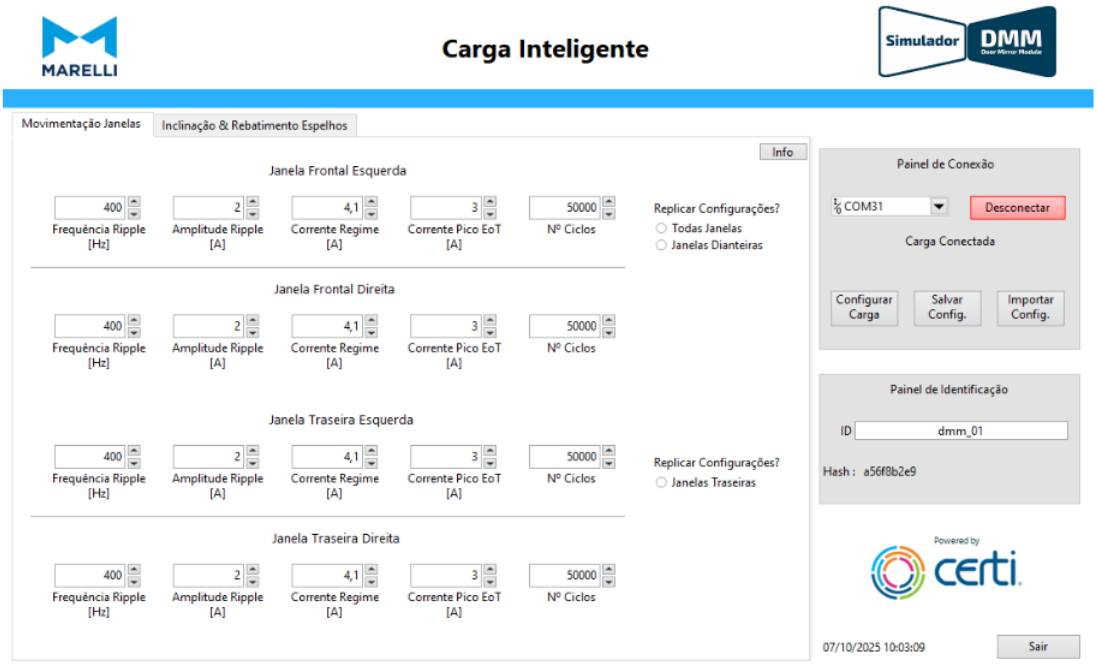
atreladas aos arquivos JSON com as configurações que são interpretados pela interface do usuário.

Para a estrutura interna do *software* LabVIEW, adotou-se o padrão QMH (*Queued Message Handler*), que permite organizar e gerenciar as ações do usuário por meio de mensagens enfileiradas. Esse padrão também facilita a atualização do *front panel*, garantindo que a interface responda de forma coerente e sincronizada às operações executadas em segundo plano.

Do ponto de vista de interface, o *software* LabVIEW permite que o usuário defina parâmetros de corrente de pico de *End of travel*, amplitude de *ripple*, valor médio da corrente, frequência da forma de onda e número de ciclos do motor, como ilustrado na Figura 18. Além disso a interface conta com uma aba para configuração dos potenciômetros, onde é possível definir os valores máximos, mínimos e a taxa de variação de cada potenciômetro.

Outras funcionalidades foram implementadas, como importar e exportar configurações e configurar todos os motores ou potenciômetros igualmente.

Figura 18 – Front panel do software



Fonte: Autoria própria (2025).

4 Apresentação dos Resultados

Este capítulo apresenta os resultados que foram obtidos das medições das formas de onda, validação das trocas de mensagens entre *software* e *firmware*, validação das funcionalidades de *software* e validação do controle realizado pela placa de desenvolvimento.

As medições para validação da forma de onda de corrente das cargas eletrônicas foram feitas utilizando um NI PXI-5152 em conjunto com o módulo PXIe-8135, como ilustrado pela Figura 19. A validação do protótipo em conjunto com o sistema de controle que viria a ser desenvolvido pela Magneti Marelli não foi possível ser realizado, no entanto, todas as funcionalidades especificadas foram testadas e validadas individualmente.

Figura 19 – Setup de medições

Fonte: Autoria própria (2025).

4.1 Análise e discussão dos resultados

Foi possível realizar as validações de cada um dos sub-sistemas do protótipo, de forma individual, inicialmente foi realizada a validação das funcionalidades presentes na interface do usuário.

O *software* LabVIEW permitiu que o usuário pudesse configurar múltiplos motores e potenciômetros. Além disso, também foram validadas as funcionalidades de importar e exportar configurações, permitindo ao usuário trocar qual veículo ele deseja emular, apenas trocando o arquivo de configuração.

Os testes foram realizados em conjunto do simulador manual, que permite acesso aos sinais gerados pelo protótipo, como demonstrado na Figura 20 sendo protótipo o equipamento à esquerda e o simulador manual a direita.

Figura 20 – Protótipo e sistema de acesso à sinais

Fonte: Autoria própria (2025).

Para este trabalho, o simulador manual serviu apenas como uma interface que externaliza os sinais das cargas eletrônicas e dos potenciômetros digitais.

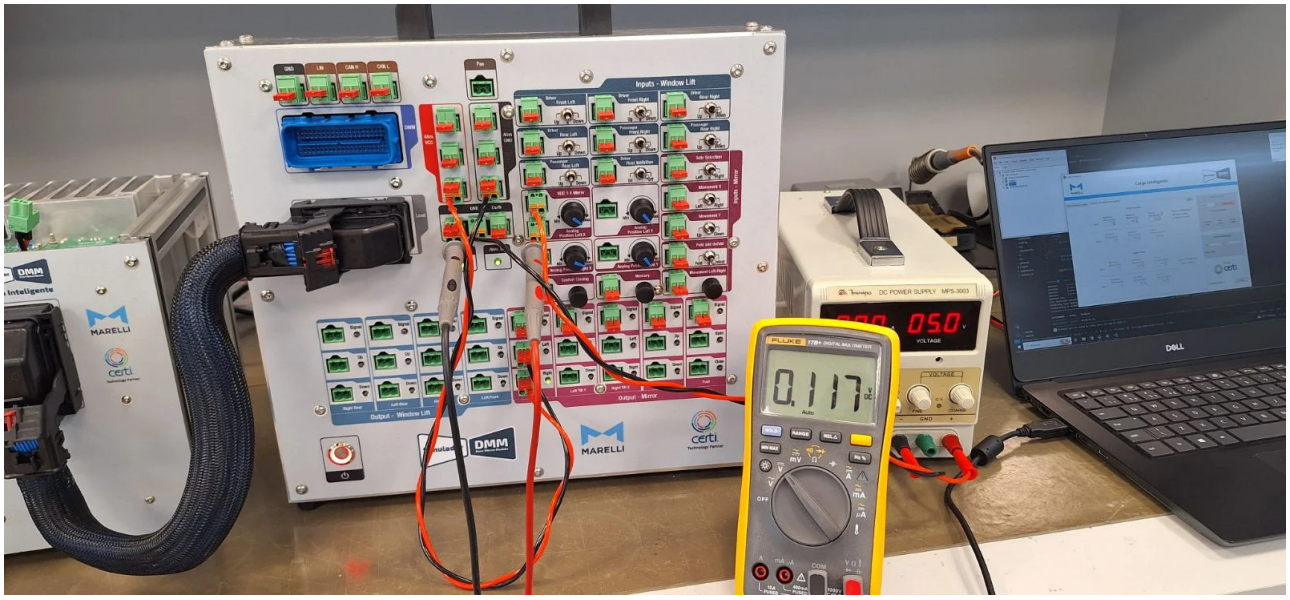
4.1.1 Validação dos potenciômetros digitais

Para verificar o funcionamento correto dos potenciômetros digitais, foi realizado a medição do valor de tensão no cursor do potenciômetro alimentado com 5 V, com o cursor do potenciômetro sendo acessado pelo simulador manual, foram validados os valores mínimos e máximos e o software foi adaptado para condizer com os valores práticos.

Como o MCP45HV51 possui um valor máximo de 5 kΩ, dividido em 8 bits, com um passo de, aproximadamente, 19 Ω por passo.

Para validar o valor mínimo de resistência, foi configurado o valor do cursor do potenciômetro para zero e verificada a queda de tensão no mesmo, como demonstra a Figura 21.

Figura 21 – Teste do valor mínimo do potenciômetro digital

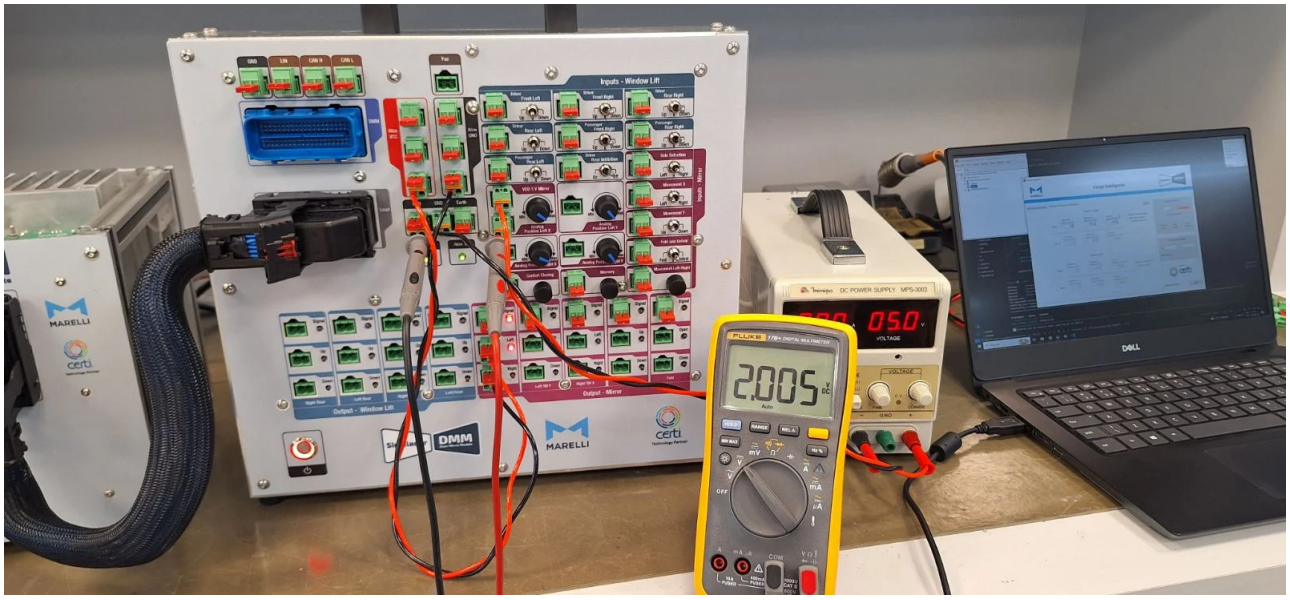


Fonte: Autoria própria (2025).

Com uma queda de tensão de 117 mV no cursor, foi definida a tensão mínima de operação do software, o valor medido condiz com o valor observado na folha de dados do fabricante. O tempo de resposta não foi um parâmetro de importância devido ao fato de que não existe necessidade de um tempo de resposta pequeno na aplicação real.

Em seguida, o potenciômetro foi acionado até o valor máximo definido por software para validar se é possível atingir uma queda de tensão próxima dos 2 V, valor referente ao potenciômetro utilizado nos circuitos da Marelli.

Figura 22 – Teste do valor máximo do potenciômetro digital



Fonte: Autoria própria (2025).

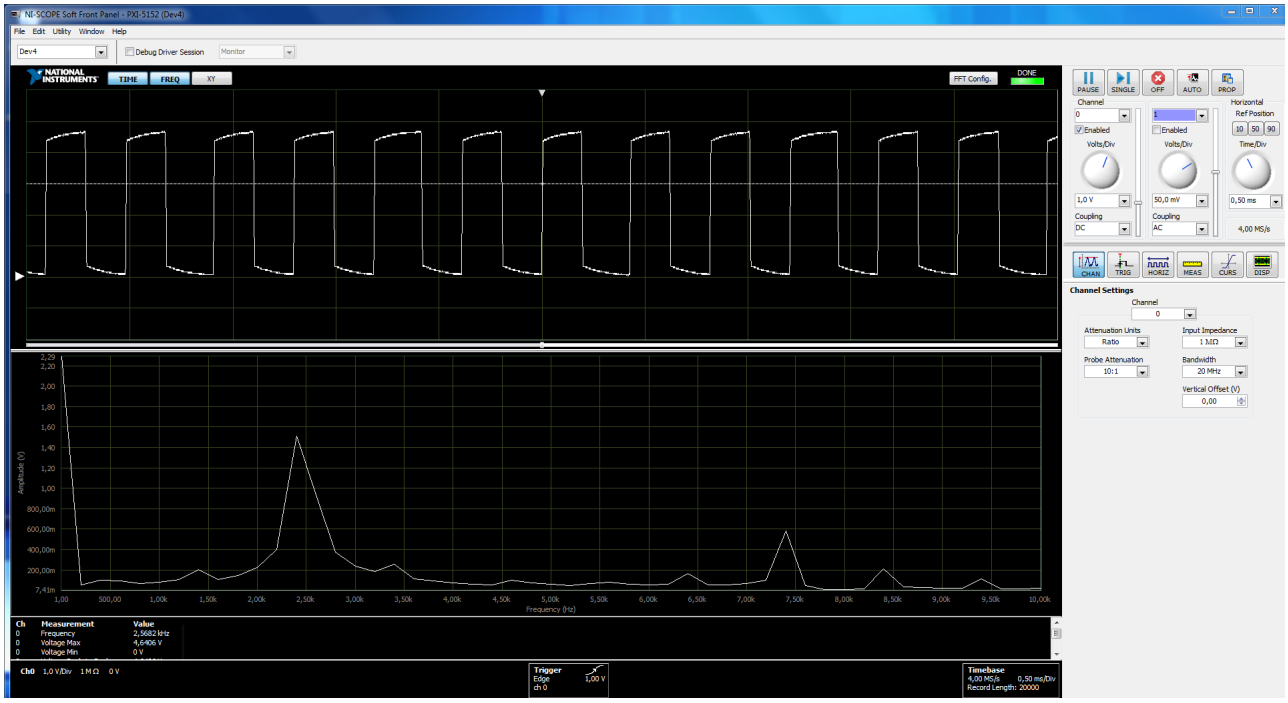
A Figura 21 apresenta a queda de tensão no potenciômetro no valor máximo. Com o potenciômetro acionado no valor máximo, foi medido um valor de tensão de 2,005 V, valor muito próximo do desejado.

4.1.2 Validação da forma de onda gerada

Os testes foram realizados verificando a geração de onda na saída do DAC que gera a forma de onda com uma frequência de I2C de 1 MHz. O teste foi realizado verificando a saída do DAC 7678 com os valores gerados pelo Arduino Giga R1 utilizando uma NI PXI-5152 em conjunto com uma NI PXIe-8135.

No teste sem controle de tempo, foi obtido uma onda quadrada variando numa faixa de frequência entre 2.4 KHz e 2.6 KHz nos quatro canais referentes aos motores, como ilustrado na Figura 23. As medições foram realizadas na saída do DAC que está conectado à placa de desenvolvimento, sendo este o sinal de entrada dos circuitos das cargas eletrônicas.

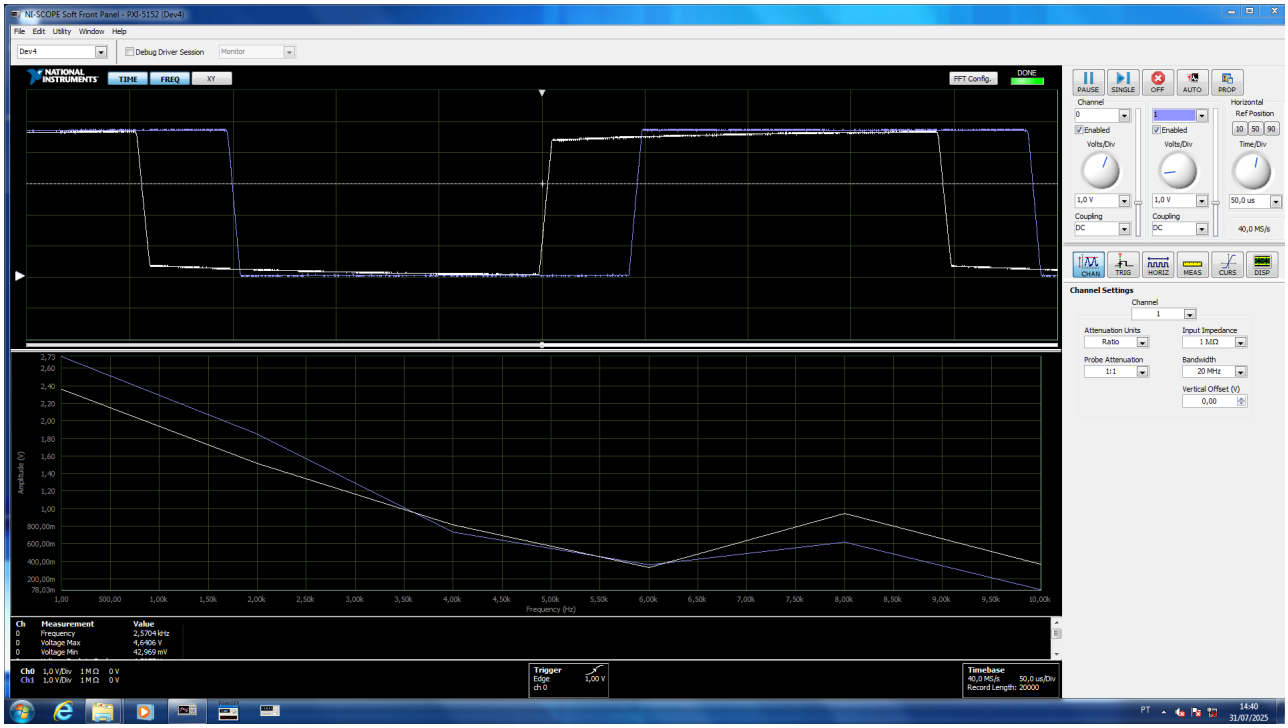
Figura 23 – Saída de controle do DAC7678



Fonte: Autoria própria (2025).

Comparando dois dos quatro sinais de controle gerados, foi possível observar a forma de onda conforme a Figura 24.

Figura 24 – Comparação das saídas de controle do DAC7678

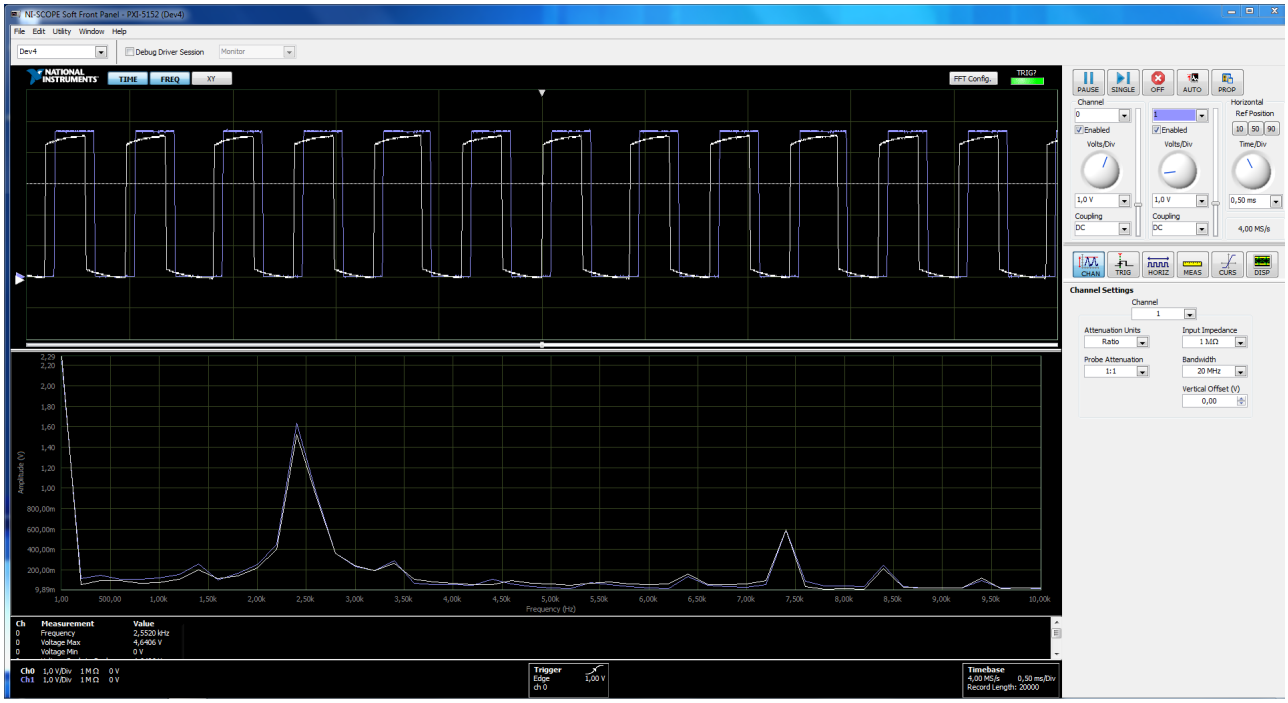


Fonte: Autoria própria (2025).

A partir da resposta apresentada, foi possível observar um atraso de aproximadamente 50 µs entre os canais analisados. Esse valor é compatível com o tempo médio medido para o envio de mensagens no barramento I2C, da ordem de 46 µs, indicando que a maior parte da defasagem observada está associada ao tempo necessário para o processamento e a transmissão das informações pela interface I2C.

Além disso, os sinais de controle apresentaram um leve desvio de frequência. Considerando a média da variação entre os quatro canais, foi identificado um desvio aproximado de 5 Hz.

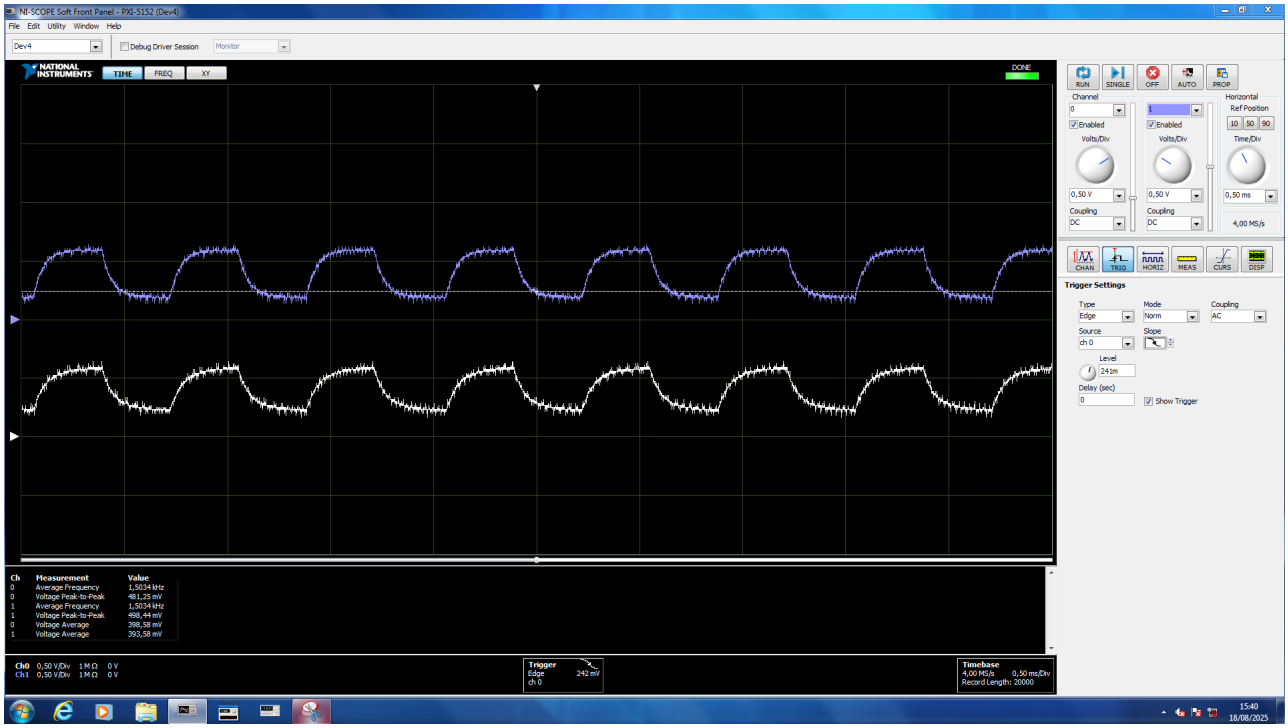
Figura 25 – Sinais de controle



Fonte: Autoria própria (2025).

Analisando o resultado apresentado pela Figura 25, é possível validar a geração da forma de onda que realiza o controle da carga eletrônica.

Figura 26 – Sinais de controle e tensão no resistor shunt



Fonte: Autoria própria (2025).

A figura 26 apresenta a tensão medida diretamente no *gate* do MOSFET e a tensão lida no resistor *shunt* (em branco) com uma configuração de 1500 Hz, 1 A de amplitude, 2 A de regime e 50000 ciclos, o sinal apresenta uma distorção em relação ao sinal de controle da saída da placa de desenvolvimento, mas ainda se mostra válido para a simulação do motor DC.

5 CONSIDERAÇÕES FINAIS

Este trabalho tem como objetivo desenvolver um *firmware* responsável pelo controle de um protótipo destinado à simulação de motores DC e controle de potenciômetros digitais.

Inicialmente foi realizado um estudo a respeito da placa de desenvolvimento, abordagem *hardware-in-the-loop*, cargas eletrônicas e como as mesmas seriam controladas e usadas no desenvolvimento do trabalho. Também foi realizado uma pesquisa a respeito de interface I2C e por fim, como tudo seria integrado usando LabVIEW.

A implementação prática consistiu no desenvolvimento do protótipo capaz de realizar o controle de cargas eletrônicas para gerar uma forma de onda de corrente, ser capaz de receber os parâmetros da forma de onda de uma aplicação feita usando LabVIEW e realizar o controle de potenciômetros digitais usando interface I2C.

Apesar de que algumas das escolhas técnicas deste trabalho tenham impactado negativamente o desenvolvimento do protótipo, como a escolha de uma placa de desenvolvimento de baixo desempenho e de uma interface de comunicação relativamente lenta, este trabalho apresentou bons resultados, realizando o controle das cargas eletrônicas e potenciômetros digitais atingindo as especificações impostas pela Magneti Marelli.

O protótipo foi muito bem recebido pela equipe da Magneti Marelli, sendo validado também na sede em Hortolândia, São Paulo. Onde foi feito a apresentação do projeto e protótipo finalizado para a equipe de P&D eletrônica da Marelli. O conceito do protótipo foi o item de maior interesse da Marelli, comentando que o conceito iria resolver muitos problemas de testes e também na parte física dos *setups*, não precisando mais um conjunto inteiro de portas.

Dessa forma, conclui-se que o trabalho atingiu seus objetivos técnicos e conceituais, demonstrando a viabilidade da abordagem proposta e evidenciando seu potencial de aplicação prática no ambiente industrial automotivo.

5.1 Sugestões para trabalhos futuros

Sugere-se, para trabalhos futuros, adaptar este projeto para realizar a utilização de outra interface para realizar o controle da forma de onda. Também pode-se realizar utilização de uma plataforma de desenvolvimento ou microcontrolador mais simples, uma vez que uma interface mais rápida possa ser utilizada.

Além disso, o desenvolvimento de um ambiente que seja acessível de forma remota para a realização dos testes, com capacidade de leitura dos sinais definidos e alteração de parâmetros sem que o usuário e a carga estejam no mesmo local.

REFERÊNCIAS

ARDUINO. ***GIGA R1 WiFi***. 2025. Acesso em: 3 nov. 2025. Disponível em: https://docs.arduino.cc/hardware/giga-r1-wifi/. 25

KEYSIGHT. ***What Are Electronic Loads?*** 2025. Tradução nossa. Acesso em: 12 nov. 2025. Disponível em: https://www.keysight.com/used/us/en/knowledge/glossary/oscilloscopes/what-are-electronic-loads. 16, 17, 19

MATHWORKS. ***Hardware-in-the-Loop (HIL)***. 2025. Acesso em: 20 nov. 2025. Disponível em: https://www.mathworks.com/discovery/hardware-in-the-loop-hil.html. 13, 25

MATSUSADA. ***Basics of Electronic Loads***. 2025. Tradução nossa. Acesso em: 10 nov. 2025. Disponível em: https://www.matsusada.com/column/electronic_loads_basic.html. 16

Microchip Technology Inc. ***Comparing Digital Potentiometers to Mechanical Potentiometers***. [S.l.], 2000. Acesso em: 3 dez. 2025. Disponível em: https://ww1.microchip.com/downloads/cn/AppNotes/cn_00219.pdf. 30

National Instruments. ***Develop with Graphical Programming in NI LabVIEW***. 2025. Acesso em: 3 dez. 2025. Disponível em: https://www.ni.com/en/shop/labview/develop-graphical-programming.html. 27, 28

WU, J. ***A Basic Guide to I^2C***, *institution = Texas Instruments, type = Application Note, number = SBAA565, year = 2022, url = https://www.ti.com/lit/an/sbaa565/sbaa565.pdf, note = Acesso em: 15 nov. 2025*. [S.l.]. 21