UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Keslley Brito Ramos - 11921EAU002
Matheus Vinicius Martins Cunha - 12011EAU023
Nicole Del Grossi Vieira de Souza - 11911EAU010
Sarah Cristine Fernandes e Silva - 12021EAU010
Yugo Satoshi Nagao da Mota - 11911EAU006
Willy Fernandes - 11821EAU005

RELATÓRIO FINAL:
CONTROLE FEEDFORWARD PARA MOTOR DC

Disciplina: Projeto Interdisciplinar de Controle e Automação (PIEAU)
Professor: Fernando Bento Silva

Uberlândia, 23 de Setembro de 2025

SUMÁRIO

1. OBJETIVO

O objetivo principal deste relatório, e do projeto como um todo, é o desenvolvimento de um sistema de controle para um motor de corrente contínua. Para isso, o documento busca detalhar a análise e a modelagem matemática do motor, as deduções das equações dinâmicas e as análises necessárias para a implementação, que incluem a simulação computacional, a coleta de dados experimentais e a aplicação prática do controle de velocidade.

2. INTRODUÇÃO

Motores de corrente contínua (CC) com excitação independente desempenham um papel crucial em uma vasta gama de sistemas de acionamento, desde aplicações industriais de alta potência até dispositivos de automação de precisão. A capacidade de controlar com elevada precisão a velocidade e o torque desses motores os torna um objeto de estudo fundamental na engenharia elétrica. Este documento apresenta todas as etapas de um projeto de pesquisa e desenvolvimento que buscou aprofundar a compreensão da dinâmica de um motor CC com excitação independente, com o objetivo final de projetar, simular e implementar um sistema de controle de velocidade robusto e eficiente.

Este relatório concentra-se na fundamentação teórica do projeto, bem como na descrição das fases de simulação, validação experimental e implementação prática. São apresentadas as equações diferenciais que modelam o comportamento elétrico e mecânico do motor, seguidas por suas respectivas transformadas de Laplace para análise no domínio da frequência. A partir desses modelos, foram deduzidas as funções de transferência que relacionam as variáveis de entrada e saída do sistema, como a tensão de armadura, o torque de carga, a corrente de armadura e a velocidade mecânica. Essa análise teórica possibilitou a compreensão das características de resposta transitória do motor, incluindo a influência de parâmetros como o fator de amortecimento e a frequência natural não amortecida, cruciais para o projeto de controladores.

A partir da fundamentação teórica, foi desenvolvido um modelo de simulação em ambiente computacional (por exemplo, no MATLAB/Simulink), que permitiu a validação do modelo matemático, a análise da resposta do motor sob diferentes condições e a pré-sintonia dos controladores. Em seguida, procedeu-se à coleta de dados de um sistema real, fornecendo os parâmetros físicos necessários para a

sintonização final e para a comparação com os resultados simulados. Por fim, realizou-se a implementação prática do controle de velocidade do motor, utilizando as análises e simulações como base para assegurar um desempenho otimizado.

Dessa forma, este relatório consolida todas as etapas do projeto, desde a modelagem matemática até a implementação prática do controle, estabelecendo um alicerce teórico e experimental sólido que sustenta o desenvolvimento realizado de forma sistemática e bem fundamentada.

3. DEDUÇÕES MATEMÁTICAS REQUISITADAS

3.1. Equação 4.8: Dedução da Função de Transferência da Malha de Corrente com Controlador PI

A equação (4.8) do material de referência representa a função de transferência da malha de corrente de um motor CC quando um controlador Proporcional-Integral (PI) é adicionado, como mostrado nas Figura 1 e 2. Essa equação é fundamental para entender como o sistema responde a uma referência de corrente.

$$I_a(s) = \frac{K_{pc}s + K_{ic}}{L_a s^2 + (R_a + K_{pc})s + K_{ic}} I^*(s) - \frac{s}{L_a s^2 + (R_a + K_{pc})s + K_{ic}} E(s) \quad (4.8)$$

onde K_{pc} e K_{ic} representam os ganhos proporcionais e integrais do controlador de corrente, respectivamente.

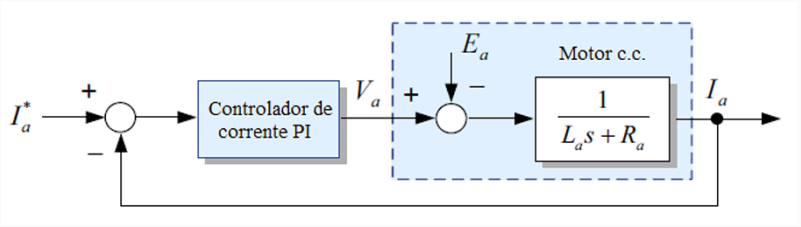

Fig. 25 – Diagrama de blocos do motor c.c. incluindo um controlador de corrente.

***Figura 01**: Função de transferência do controlador e do motor.*

Verifica-se na equação (4.8) que a força contra-eletromotriz do motor c.c. atuará como uma perturbação para o sistema de controle de corrente. Uma forma de eliminar este efeito indesejável é através do controle antecipativo (feedforward), uma vez que a fcem pode ser estimada através da velocidade angular. Quando a perturbação gerada por *E(s)* é compensada como apresentado na Figura 26 (a), o diagrama de blocos se tornará um circuito R-L como apresentado na Figura 26 (b).

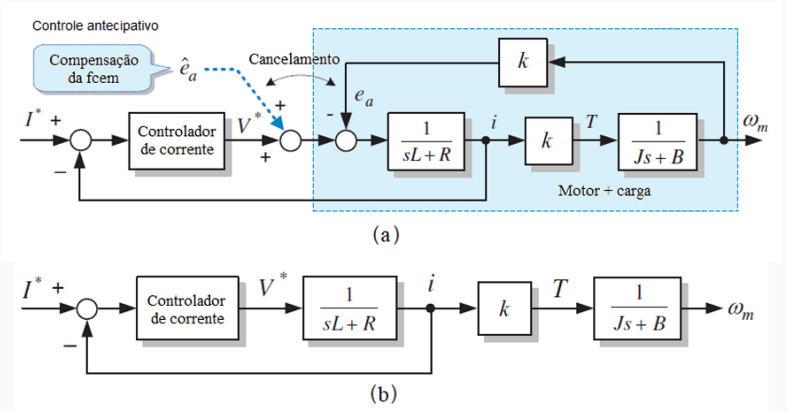

Fig. 26 – Malha do sistema de controle de corrente; (a) Controle antecipativo da *fcem* (B) Malha após compensar o efeito. da *fcem.*

***Figura 02:** Malha do sistema de controle de corrente.*

Para deduzir a equação, consideramos o diagrama de blocos da malha de corrente em malha fechada.

A partir do diagrama, podemos escrever as equações para cada bloco.

A saída do controlador, a tensão de armadura (V_a), é a soma das ações proporcional e integral sobre o erro de corrente. O erro é a diferença entre a corrente de referência (I_a^*) e a corrente de armadura medida (I_a). No domínio de Laplace, a função de transferência do controlador é:

$$G_{PI}(s) = K_{pc} + \frac{K_{ic}}{s}$$

e Motor:

$$\frac{1}{L_a * s + R_a}$$

E o somatório:

$$Motor + G_{PI}(s) = \left(K_{pc} + \frac{K_{ic}}{s} \right) + \frac{1}{L_a * s + R_a}$$

$$= \frac{k_{pc} * s + K_{ic}}{L_a * s^2 + R_a * s}$$

A tensão aplicada no motor, a armadura tem-se:

$$V_a(t) = R_a * I_a(t) + L_a * \frac{di_a(t)}{dt} + e(t)$$

$$V_a(t) = G_{PI}(s) * (I_a(s) - I_a(s))$$

Substituindo os valores:

$$G_{PI}(s) * (I_a(s) - I_a(s)) - E(s) = (L_a * s + R_a) * I_a(s)$$

$$G_{PI}(s) * I_a(s) - G_{PI}(s) I_a(s) - E(s) = (L_a * s + R_a) * I_a(s)$$

$$G_{PI}(s) I_a(s)^{**} - E(s) = (L_a * s + R_a) + G_{PI}(s) * I_a(s)$$

$$\frac{G_{PI}(s) * I_a(s)}{((L_a * s + R_a) + G_{PI}(s))} - \frac{E(s)}{((L_a * s + R_a) + G_{PI}(s))}$$

Onde,

$$G_{PI}(s) = K_{pc} + \frac{K_{ic}}{s}$$

$$G_{PI}(s) = \frac{K_{pc}*s + K_{ic}}{s}$$

E portanto, substituindo, temos:

$$\frac{\frac{K_{pc}*s+K_{ic}}{s} * I_a(s)}{((L_a*s+R_a)+\frac{K_{pc}*s+K_{ic}}{s})} - \frac{E(s)}{((L_a*s+R_a)+\frac{K_{pc}*s+K_{ic}}{s})}$$

$$I_a(s) = \frac{K_{pc}*s+K_{ic}}{L_a*s^2+(R_a+K_{pc})*s+K_{ic}} * I_a^{**}(s) - \frac{s}{L_a*s^2+(R_a+K_{pc})*s+K_{ic}} * E(s)$$

E desta forma, quando aplicamos, E(s) = 0, só existe a entrada $I_a^{**}(s)$, sendo, portanto:

$$I_a(s) = \frac{K_{pc}*s+K_{ic}}{L_a*s^2+(R_a+K_{pc})*s+K_{ic}} * I_a^{**}(s) - 0$$

E para $I_a^{**}(s)$ = 0, tem-se apenas a perturbação como entrada, sendo:

$$I_a(s) = \frac{s}{L_a*s^2+(R_a+K_{pc})*s+K_{ic}} * E(s)$$

3.2. Equação 4.14: Função de Transferência em Malha Fechada e Largura de Banda

Conforme apresentado no material utilizado pelo professor:

A função de transferência de malha aberta $G_c^o(s)$ do sistema apresentado na Figura 27 é dada por:

$$G_c^o(s) = K_{pc}\left(\frac{s+\frac{1}{T_{pi}}}{s}\right).\frac{1}{L_a s+R_a} = K_{pc}\frac{\left(s+\frac{K_{ic}}{K_{pc}}\right)}{s}.\frac{\frac{1}{L_a}}{\left(s+\frac{R_a}{L_a}\right)} \quad (4.10)$$

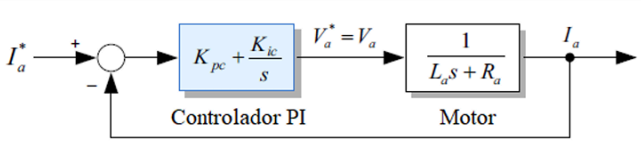

Fig. 27 – Malha do controlador de corrente PI.

***Figura 03**: Função de transferência em malha fechada.*

Da Eq. (4.10), a frequência de cruzamento de ganho ω_{cc} pode ser determinada por

$$|G_c^o(j\omega_{cc})| = \frac{1}{\left|\frac{L_a}{K_{pc}} j\omega_{cc}\right|} = 1 \rightarrow \omega_{cc} = \frac{K_{pc}}{L_a} \quad (4.14)$$

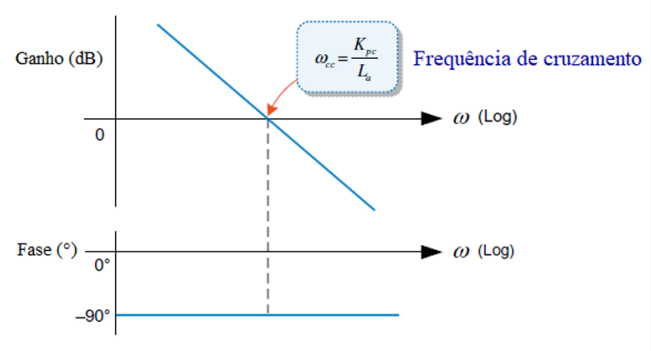

Fig. 28 – Resposta em frequência de malha aberta.

Figura 04: *Frequência de cruzamento.*

No slide 65, temos: Cancelamento de pólos e zeros:

$$\frac{K_{ic}}{K_{pc}} = \frac{R_a}{L_a}$$

$$G_c^0(s) = \frac{1}{\left(\frac{L_a}{K_{pc}}\right) * s} = \frac{K_{pc}}{L_a * s}$$

Transformando para o domínio da frequência, tem-se:

$$\left|G_c^0(jw)\right| = \left|\frac{K_{pc}}{L_a * (jw)}\right|; \; onde \; jw = w$$

$$\left|G_c^0(jw)\right| = \left|\frac{K_{pc}}{L_a * (w)}\right|$$

Dado o ponto de cruzamento de ganho, $\omega_{cc} = \left|G_c^0(j\omega_{cc})\right| = 1$, pela análise gráfica, e portanto,

$$\frac{K_{pc}}{L_a * W_{cc}} = 1$$

E multiplicando cruzado, tem-se:

$$L_a * W_{cc} * 1 = K_{pc}$$

Logo,

$$\omega_{cc} = \frac{K_{pc}}{L_a}$$

3.3. Equação 4.15:

A função de transferência de malha fechada deste sistema é dada por

$$\frac{I_a(s)}{I_a^*(s)} = G_c^c(s) = \frac{G_c^o(s)}{1 + G_c^o(s)} = \frac{1}{\left(\frac{L_a}{K_{pc}}\right)s + 1} = \frac{\omega_{cc}}{s + \omega_{cc}} \qquad (4.15)$$

A Equação (4.15) apresenta a função de transferência em malha fechada para a malha de corrente. A dedução começa com a fórmula geral para um sistema de malha fechada unitária, que relaciona a saída com a entrada através da função de transferência de malha aberta.

$$G_c^c(s) = \frac{G_c^0(s)}{1 + G_c^c(s)}$$

Substituindo a função de transferência de malha aberta, que foi simplificada na Equação (4.12):

$$G_c^c(s) = \frac{\frac{1}{\left(\frac{L_n}{K_{pc}}\right)*s}}{1 + \frac{1}{\left(\frac{L_n}{K_{pc}}\right)*s}}$$

Para simplificarmos a expressão, multiplicamos numerador e denominador por:

$$\left(\frac{L_a}{K_{pc}}\right) * s$$

$$G_c^c(s) = \frac{1}{1 + \left(\frac{L_a}{K_{pc}}\right)*s}$$

E pela equação 4.14, sabemos que:

$$\omega_{cc} = \frac{K_{pc}}{L_a}$$

Fazendo essa substituição na equação da malha fechada:

$$G_c^c(s) = \frac{1}{1 + \left(\frac{L_a}{K_{pc}}\right)*s} = \frac{\omega_{cc}}{s + \omega_{cc}}$$

Essa é a prova da equação (4.15). Ela mostra que a malha de corrente se comporta como um sistema de primeira ordem, com um único pólo em

$$s = -\omega_{cc}$$

o que simplifica bastante a análise e o projeto.

3.4. Equação 4.16:

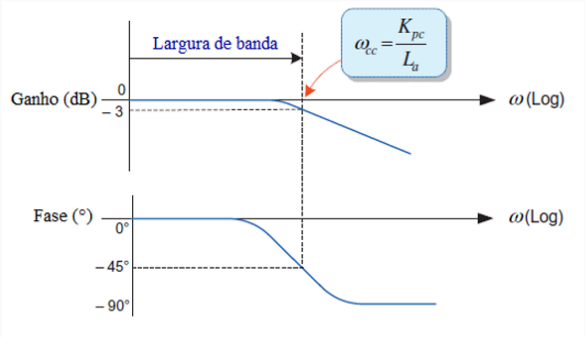

Fig. 29 – Resposta de frequência em malha fechada.

A resposta em frequência da equação (4.15) é apresentada na Figura 29. Verifica-se que a largura de banda deste sistema é dado como ω_{cc}, sendo verificado por:

$$|G_c^c(s)| = \left|\frac{\omega_{cc}}{s+\omega_{cc}}\right| = \frac{1}{\sqrt{2}} \quad (4.16)$$

Assim, **a largura de banda** é igual à **frequência de cruzamento de ganho** da resposta de frequência de malha aberta.

Figura 05: *Resposta em frequência.*

A Equação (4.16) é a prova de que a largura da banda da malha fechada é igual à frequência de cruzamento de ganho. A largura de banda de um sistema de primeira ordem é a frequência onde a magnitude da função de transferência cai. Aplicando o módulo em ambos os lados tem-se

$$\left|G_c^c(s)\right| = \left|\frac{\omega_{cc}}{s+\omega_{cc}}\right| = \frac{\omega_{cc}}{\sqrt{\omega_{cc}^{\,2}+\omega_{cc}^{\,2}}}$$

Para encontrar a largura de banda, igualamos a magnitude a por ω_{cc}, e assim:

$$\left|G_c^c(s)\right| = \frac{\omega_{cc}}{\sqrt{\omega_{cc}^{\,2}+\omega_{cc}^{\,2}}} = \frac{\omega_{cc}}{\sqrt{2*\omega_{cc}^{\,2}}} = \frac{\omega_{cc}}{\omega_{cc}\sqrt{2}} = \frac{1}{\sqrt{2}}$$

E isso prova a Equação (4.16) e confirma que a largura de banda de controle do sistema é igual à frequência de cruzamento de ganho, ou seja,

$$\omega_{cc}$$

3.5. Provar 4.21:

7.6 Projeto do controlador de velocidade

Para que o controle de corrente não exerça qualquer influência no controle de velocidade, $\omega_{cc} = 5\ \omega_{cs}$. Sob esta condição, o ganho do controlador de corrente em torno da frequência de cruzamento ω_{cs} aproxima-se para:

$$G_c^c(s) = \frac{\omega_{cc}}{s + \omega_{cc}} \approx 1 \qquad (4.21)$$

Prova 4.21:

Substituindo s por $j\omega$, tem-se

$$G_c^c(j\omega) = \frac{\omega_{cc}}{j\omega + \omega_{cc}}$$

A premissa do controle em cascata é que a malha interna (corrente) é significativamente mais rápida que a malha externa (velocidade). Isso significa que a frequência de cruzamento de ganho da malha de corrente, ou seja:

Considerando $\omega_{cc} \gg \omega \Rightarrow \left(j\omega + \omega_{cc}\right)$, logo, ω_{cc} é muito maior que as frequências de operação da malha de velocidade, e portanto, é permitindo a seguinte aproximação:

$$\left(j\omega + \omega_{cc}\right) \approx \omega_{cc}$$

$$G_c^c(j\omega) = \frac{\omega_{cc}}{j\omega + \omega_{cc}} \approx \frac{\omega_{cc}}{\omega_{cc}} \approx 1\ .$$

3.6. Provar 4.24:

7.0 Projeto do sistema de controle do motor de *ccei*

7.6 Projeto do controlador de velocidade

O ganho desta função de transferência de malha aberta na frequência ω_{cs} é 0 dB, ou seja,

$$|G_s^o(j\omega_{cs})| = \left|K_{ps}.\frac{k_T}{Jj\omega_{cs}}\right| = 1 \rightarrow \omega_{cs} = \frac{K_{ps}.k_T}{J} \qquad (4.24)$$

Resposta:

$$\left|G_S^O\left(j\omega_{CS}\right)\right| = \left|K_{ps} \cdot \frac{k_T}{Jj\omega_{cs}}\right| = 1$$

Multiplicando ambos os lados por $\left|j\omega_{cs}\right|$, tem-se

$$\left|K_{ps} \cdot \frac{k_T}{Jj\omega_{cs}}\right| \cdot \left|j\omega_{cs}\right| = 1 \cdot \left|j\omega_{cs}\right|$$

$$\Rightarrow \left|\frac{K_{ps} \cdot k_T}{J}\right| = \left|j\omega_{cs}\right|$$

$$\Rightarrow \frac{K_{ps} \cdot k_T}{J} = \omega_{cs}$$

4. METODOLOGIA

A caracterização experimental dos parâmetros elétricos e mecânicos do motor de corrente contínua de imã permanente é um passo fundamental para a construção de um modelo preciso e para o desenvolvimento de um sistema de controle eficaz. A coleta de dados foi realizada no laboratório, seguindo procedimentos específicos para cada tipo de parâmetro, conforme detalhado a seguir.

Para a obtenção dos parâmetros elétricos, nomeadamente a indutância de armadura (L_a) e a resistência de armadura (R_a) foi empregado um ensaio com corrente alternada, conforme descrito na Tabela 1 do roteiro [2] de obtenção dos parâmetros. Este ensaio permite determinar a impedância da armadura, a partir da qual R_a e L_a podem ser calculados com base no fator de potência.

No que concerne aos parâmetros mecânicos, a metodologia envolveu dois ensaios principais. O cálculo aproximado do coeficiente de atrito (B) foi realizado medindo-se a potência elétrica absorvida pelo motor girando a vazio e com velocidade estabilizada. As perdas resistivas foram subtraídas da potência elétrica total para se obter a potência mecânica (P_{mec}) , desprezando as perdas magnéticas no rotor. Com o motor operando a vazio e em velocidade constante, o conjugado mecânico é atribuído unicamente ao atrito nos mancais do motor

$$T_{mec} = B * \omega_m$$

A partir dessa relação, o coeficiente de atrito foi determinado utilizando a potência mecânica e a velocidade angular

$$B = \frac{P_{mec}}{\omega_m^2}$$

E para, a determinação do momento de inércia (J) foi aplicado o método de retardamento. Este ensaio consiste em acelerar o motor até uma velocidade estabilizada (ω_0), desligar a alimentação de tensão e observar a curva de desaceleração da velocidade em função do tempo. A partir da curva de desaceleração, que idealmente segue uma exponencial do tipo:

$$\omega = \omega_0 * e^{-t/t_j}$$

Para a constante de tempo das massas girantes (t_j) é determinada, definida como o tempo necessário para que a velocidade do motor decresça para aproximadamente 37% do valor inicial, ou mais precisamente, pela intersecção da tangente à curva de desaceleração no instante inicial ($\omega = \omega_0$) com o eixo do tempo. Com o valor de t_j e o coeficiente de atrito (B) previamente calculado, o momento de inércia (J) é então obtido pela relação:

$$J = B * t_j$$

4.1. COLETA DE DADOS

Os dados, coletados com auxílio do professor, no dia 24/07/2025 foram anotados conforme as tabelas abaixo.

4.1.1. Obtenção de parâmetros elétricos e mecânicos do motor CC de ímã permanente.

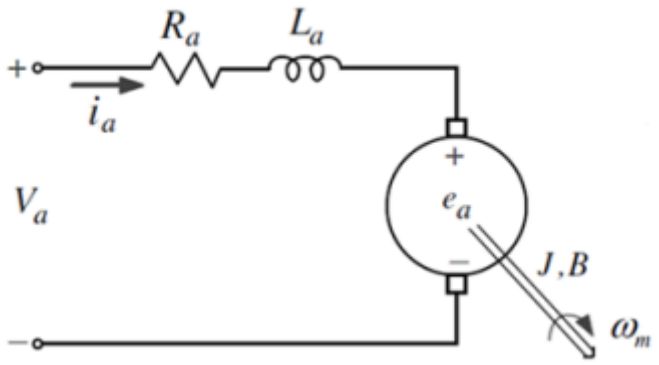

Figura 06: *Diagrama de ligação de um motor de corrente contínua de imã permanente.*

Os parâmetros elétricos da máquina, tais como, a obtenção de L_a (indutância de armadura), e R_a (resistência de armadura), é realizada por meio do ensaio com corrente alternada. Este método é preferível ao ensaio em corrente contínua para a determinação de L_a , pois a indutância só se manifesta em regime de corrente alternada. Ao aplicar uma tensão AC e medir a corrente e o fator de potência, é

possível calcular a impedância de armadura (Z_a), e, a partir de suas componentes real e imaginária, determinar R_a e L_a

Coleta de parâmetros						
Va (V)	**Ia (A)**	**Fp= cosφ**	**φ**	**Za= Va/Ia**	**Ra (Ω)**	**La (mH)**
2,9	0,622	0,898	26	4,662	4,186	5,42
3,7	0,79	0,894	26,5	4,683	4,186	5,54
5,02	1,07	0,881	28,12	4,691	4,183	5,86
6,03	1,27	0,894	26,53	4,748	4,244	5,62
7,4	1,56	0,873	29,1	4,743	4,14	6,11
9,32	1,97	0,875	28,94	4,73	4,138	6,07

Tabela 01: *Dados registrados através do ensaio AC;*

4.1.2. Coleta de Dados de Regime Permanente (Ensaio CC)

Os dados de regime permanente em corrente contínua são cruciais para validar o comportamento teórico do motor e para a obtenção de parâmetros mecânicos. O ensaio foi realizado sob condições de tensão e corrente nominais, com a velocidade estabilizada. A **velocidade nominal (n)**, a tensão nominal (V_n) e a corrente nominal (I_n), são obtidas neste ensaio, permitindo o cálculo do torque nominal:

$$T_{nominal} = \frac{P_{nominal}}{\omega_{nominal}}$$

e da constante do motor (C), onde:

$$C = K_T = K_c$$

É um parâmetro fundamental que liga o domínio elétrico (tensão e corrente) ao domínio mecânico (velocidade e torque).

Ensaio CC						
Vn	**In**	**rpm**	**Ra médio**	**La médio**	**t descida (s)**	**Velocidade**
112	0,56	8650	4,1795	5,77	2,24	3200

Tabela 02: Resultados do ensaio CC registrados.

Para a determinação dos parâmetros mecânicos, como o coeficiente de atrito (B) e o momento de inércia (J) , foi utilizado o ensaio de retardamento. Este método é fundamental porque ele isola as dinâmicas mecânicas do sistema. O ensaio de atrito (B) é baseado na medição da potência mecânica **a vazio** (P_{mec}), , que é atribuída unicamente ao atrito nos mancais quando o motor opera a velocidade constante. O ensaio de retardamento para a inércia (J) utiliza a curva de desaceleração do motor após a interrupção da alimentação. A constante de tempo das massas girantes (t_j) é obtida a partir desta curva:

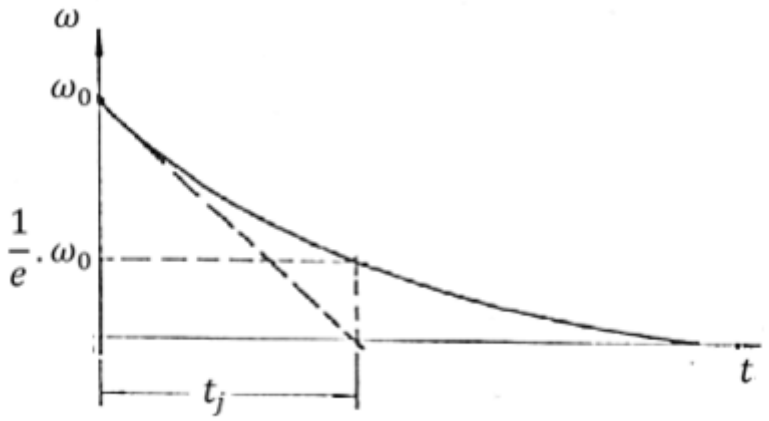

Figura 07: *Curva de desaceleração do motor.*

E o momento de inércia (J) é então calculado pela relação:

$$J = B\,x\,t_j$$

Este procedimento é considerado mais preciso pois a constante de tempo do sistema se altera, pois B aumenta com a queda de velocidade, deformando a curva exponencial ideal de desaceleração. A qual foi construída análise através da tabela:

Velocidade(rad/s)	Tempo(s)
	0
0	

Tabela 03: *Tabela para preenchimento dos dados, fornecida pelo professor Fernando Bento Silva.*

E como resultado, após a análise dos parâmetros, foram obtidos:

B	J	T (N.m)	c	Ea (V)
$7,48 \times 10^{-5}$	$1,676 \times 10^{-4}$	$0,0678$	$0,121$	$109,66$

Tabela 04: *resultado dos parâmetros mecânicos do ensaio a vazio.*

5. SIMULAÇÃO

Para validar o comportamento dinâmico e o desempenho do sistema de controle de velocidade de um motor de corrente contínua com excitação independente, foi desenvolvida uma simulação computacional no ambiente MATLAB/Simulink. O modelo implementado representa a planta do motor, o sistema de controle em malha fechada e as condições de operação. A modelagem matemática do motor foi baseada nas equações diferenciais que descrevem o circuito de armadura, a geração da força contra-eletromotriz (FCEM), a relação torque-corrente e a dinâmica mecânica do eixo, conforme discutido na seção de modelagem.

$$V_a = R_a \cdot i_a + \frac{L_a di_a}{dt} + e_a$$

Onde:

- V_a = tensão de armadura [V]
- R_a = resistência da armadura [Ω]
- i_a = corrente da armadura [A]
- L_a = indutância da armadura [H]
- e_a = força contra-eletromotriz [V]

$$T_e = J\frac{d\omega}{dt} + B\omega_m + T_L$$

Onde:

- T_e = torque eletromagnético [N·m]
- J = momento de inércia do rotor [kg·m²]
- ω_m = velocidade angular mecânica [rad/s]

- B = coeficiente de atrito viscoso [N·m·s]
- T_L = torque de carga [N·m]

No Simulink, esses relacionamentos foram traduzidos em blocos interconectados que reproduzem a resposta do sistema frente a variações na referência de velocidade e no torque de carga. A simulação contempla:

- Controlador de corrente (malha interna): implementado na forma PI, projetado para garantir rápida resposta na regulação da corrente de armadura e compensar a FCEM através de uma ação feedforward.
- Controlador de velocidade (malha externa): também do tipo PI, responsável por ajustar a referência de corrente para o controlador interno de modo a reduzir o erro entre a velocidade medida e a referência (setpoint).
- Conversor PWM: bloco que simula a modulação por largura de pulso aplicada à armadura do motor, convertendo o sinal de controle em tensão aplicada.

Modelo do motor: sub-bloco que representa as equações elétricas e mecânicas, incluindo parâmetros como resistência da armadura R_a, a indutância La, o momento de inércia, o atrito viscoso B, e constante de torque/FCEM. Blocos de medição e visualização: indicadores analógicos (velocímetro e torquímetro), gráficos de velocidade e corrente ao longo do tempo.

Os parâmetros utilizados na simulação foram definidos conforme especificações do motor e do projeto de controle, sendo:

- $V_{nominal} = 112V$;
- $I_{nominal} = 0,56A$;
- $rpm_{nominal} = 8650\,rpm$;
- $R_a = 4,1795\Omega$;
- $L_a = 5,77 \cdot 10^{-3}H$;
- $J = 1,676 \cdot 10^{-4}kg.m^2$;
- $B = 7,45 \cdot 10^{-5}N.m.s.$

A frequência de corte da malha de corrente foi definida como $f_c = 5kHz$, resultando nos ganhos calculados para os controladores.

A simulação foi executada para diferentes valores de setpoint de velocidade e torque de carga, permitindo observar a resposta transitória, o regime permanente e a atuação dos controladores na rejeição de perturbações. Os resultados obtidos serão analisados na seção seguinte.

5.1. Dados MathLive Script

O script em MATLAB, cujos parâmetros físicos do motor foram obtidos por ensaios laboratoriais, foi utilizado para projetar os ganhos dos controladores PI de corrente e de velocidade, elementos essenciais para o sistema de controle em cascata.

A partir do projeto contínuo, a lógica de controle foi então discretizada, um passo fundamental para sua posterior implementação em um microcontrolador. Durante este processo, o script também calculou os coeficientes **b0** e **b1** para os controladores de corrente e de velocidade, preparando o código para a forma incremental que seria utilizada na programação

5.1.1. Script em MATLAB

```
clc;
clear all;
%% ======================
%  Parâmetros do motor
%% ======================
V_nominal = 112;                  % Tensão nominal [V]
I_nominal = 0.56;                 % Corrente nominal [A]
rpm = 8650;                       % Velocidade nominal [rpm]
R_a = 4.1795;                     % Resistência da armadura [Ohm]
L_a = 5.77 * 10^-3;               % Indutância da armadura [H]
B = 7.45 * 10^-5;                       % Coeficiente de atrito viscoso
[N.m.s/rad]
J = 1.676 * 10^-4;                % Momento de inércia [kg.m²]
T = 0.0678;                       % Torque nominal [N.m]
C = 0.121;                          % Constante de torque e f.e.m. [N.m/A
ou V.s/rad]
%% ======================
%  Projeto do controlador de corrente
%% ======================
```

```
fc = 5000;                          % Frequência de corte [Hz]
wcc = 2 * pi * fc / 20;             % Frequência angular [rad/s]
kpc = L_a * wcc;                    % Ganho proporcional corrente
kic = R_a * wcc;                    % Ganho integral corrente
%% ======================
%  Projeto do controlador de velocidade
%% ======================
wcs = wcc / 5;                      % Frequência angular vel [rad/s]
kis = (J * wcs * wcs) / (5 * C);    % Ganho integral velocidade
kps = (J * wcs) / C;                % Ganho proporcional velocidade
%% ======================
%  Discretização (Tustin)
%% ======================
s = tf('s');
% Laço de corrente
Ts_corr = 1/1000;     % período de amostragem da corrente [s] -> 20 us
(50 kHz)
C_corr = kpc + kic/s;
Cz_corr = c2d(C_corr, Ts_corr, 'tustin');
% Laço de velocidade
Ts_vel = 1/100;       % período de amostragem da velocidade [s] -> 0.2 ms
(5 kHz)
C_vel = kps + kis/s;
Cz_vel = c2d(C_vel, Ts_vel, 'tustin');
%% ======================
%  Coeficientes incrementais
%% ======================
% Corrente
b0_corr = kpc + (kic * Ts_corr / 2);
b1_corr = -kpc + (kic * Ts_corr / 2);
% Velocidade
b0_vel = kps + (kis * Ts_vel / 2);
b1_vel = -kps + (kis * Ts_vel / 2);
%% ======================
%  Resultados (impressão)
%% ======================
disp('--- Controlador de Corrente ---')
Cz_corr
fprintf('Forma incremental: u[k] = u[k-1] + b0*e[k] + b1*e[k-1]\n');
fprintf('b0_corr = %.6f, b1_corr = %.6f\n\n', b0_corr, b1_corr);
disp('--- Controlador de Velocidade ---')
Cz_vel
fprintf('Forma incremental: u[k] = u[k-1] + b0*e[k] + b1*e[k-1]\n');
fprintf('b0_vel = %.6f, b1_vel = %.6f\n\n', b0_vel, b1_vel);
```

5.2. Simulink:

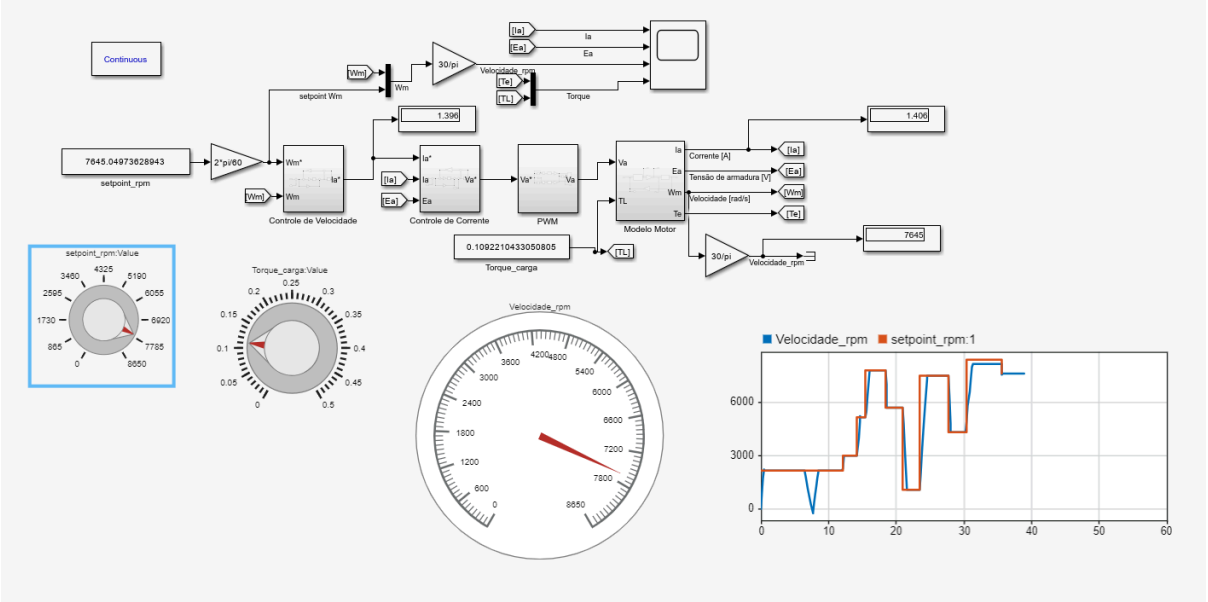

***Figura 08**: Imagem retirada do projeto desenvolvido no Simulink - MatLab*

O diagrama de blocos do Simulink representa um sistema de controle de motor de corrente contínua em **malha fechada** com uma arquitetura **em cascata**. Cada bloco ilustra uma função específica do sistema de controle, conectando a teoria à simulação prática.

Detalhamento dos Blocos

- **Bloco setpoint_rpm**: Este é o sinal de referência para a velocidade do motor em RPM. Ele representa a velocidade desejada pelo usuário e atua como a entrada principal para o sistema de controle.
- **Bloco 2*pi/60**: Este bloco de ganho converte a velocidade de referência de **rotações por minuto (RPM)** para **radianos por segundo (rad/s)**. Esta conversão é necessária porque os cálculos de controle e a modelagem do motor são realizados em unidades do Sistema Internacional (SI). A fórmula utilizada é RPM * 2π / 60.
- **Subsistema Controle de velocidade**: Este é o controlador da malha externa. Ele compara a velocidade de referência com a velocidade real (Wm) e usa um controlador PI para gerar um sinal de referência de corrente (Ia*). Este controlador opera em uma largura de banda mais baixa do que o de corrente, seguindo a lógica do controle em cascata.

- **Subsistema Controle de corrente**: Este é o controlador da malha interna. Ele recebe a referência de corrente (Ia*) e a corrente real (Ia) e utiliza outro controlador PI para gerar um sinal de tensão de referência (Va*). Esta malha é muito mais rápida e tem como objetivo garantir que a corrente siga a referência de forma precisa, compensando as variações da força contra-eletromotriz (Ea) do motor.
- **Subsistema PWM**: Este bloco modela o comportamento de um conversor de potência real (como um chopper de quatro quadrantes) que opera por **Modulação por Largura de Pulso (PWM)**. Ele converte a tensão de referência contínua (Va*) do controlador em um sinal de pulso para acionar o modelo do motor, que simula a tensão aplicada ao motor.
- **Subsistema Modelo Motor**: Este bloco representa a dinâmica do motor de corrente contínua. Ele utiliza os parâmetros elétricos (R_a, L_a) e mecânicos (J, B). As entradas para este bloco são a tensão de armadura (Va) e o torque de carga (TL). Suas saídas são a corrente de armadura (Ia), a velocidade angular (Wm), a força contra-eletromotriz (Ea) e o torque eletromagnético (Te).
- **Blocos de Display**: Os blocos de exibição na parte inferior do diagrama mostram os valores instantâneos das variáveis do sistema, como o setpoint de velocidade e o Torque_carga.
- **Blocos Scope**: Estes blocos permitem a visualização das formas de onda dos sinais ao longo do tempo. Eles são essenciais para analisar a resposta transitória do sistema, verificando se ele atinge a velocidade desejada e como a corrente se comporta durante a partida e sob diferentes cargas.

5.2.1. Malha de Controle de Velocidade

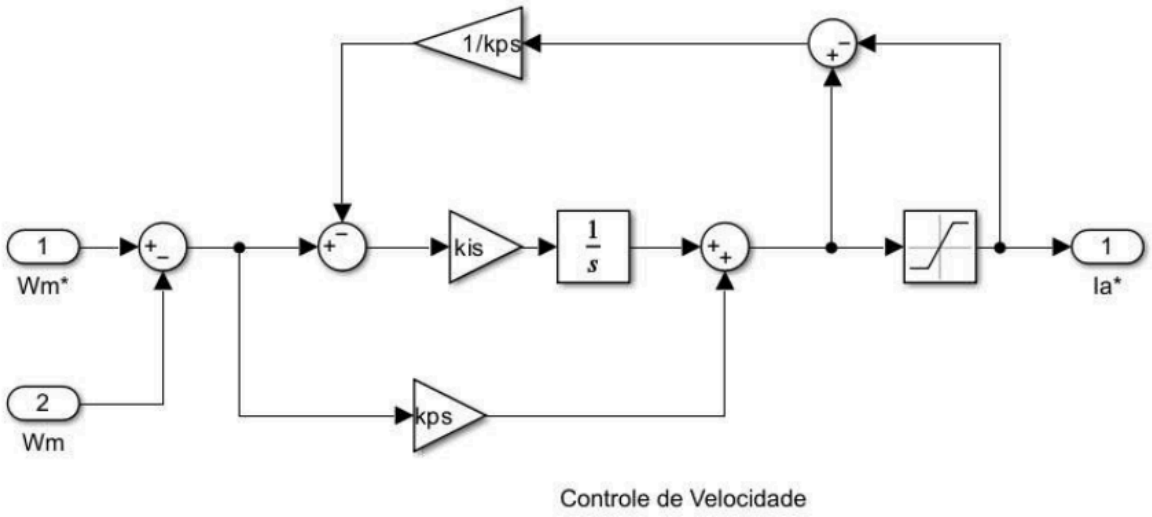

Figura 09: *Bloco de Controle de Velocidade*

O subsistema "Controle de Velocidade" atua como a **malha de controle externa** no arranjo em cascata. Sua principal função é garantir que a velocidade real do motor (ω_m) siga com precisão a velocidade de referência desejada (ω^*_m). Este subsistema é crucial para o desempenho geral do sistema, pois determina quão bem o motor mantém sua velocidade sob diferentes condições de carga e referências.

Detalhando cada componente e o fluxo do sinal neste bloco:

- Entrada de Referência (ω^*_m): Representada pelo terminal 1 no lado esquerdo, esta é a **velocidade angular desejada** para o motor. No seu modelo Simulink de nível superior, este sinal viria do seu setpoint_rpm após a conversão para rad/s.
- Sinal de Feedback (ω_m): Representado pelo terminal 2 no lado esquerdo, este é o feedback da velocidade angular real do motor, obtido do subsistema "Modelo Motor". É o que informa ao controlador a velocidade atual do motor.

- **Comparador de Erro (+ -)**: O primeiro círculo com os sinais + e - é um somador que calcula o erro de velocidade (e_{ω}). Este erro é a diferença entre a velocidade de referência e a velocidade medida:

$$e_{\omega} = \omega^{*}_{m} - \omega_{m}$$

Se o erro for positivo, o motor precisa acelerar; se for negativo, precisa desacelerar.

- Controlador PI (Proporcional-Integral): O erro de velocidade é então realimentado no controlador PI. Este controlador é composto por dois ramos paralelos:
  - Ramo Proporcional (k_{ps}): O bloco kps (ganho proporcional) multiplica o erro instantâneo de velocidade por um valor constante (o ganho k_{ps}). Este termo proporciona uma resposta rápida a variações no erro. Quanto maior k_{ps}, mais agressiva é a correção.
  - Ramo Integral (k_{is} e $\frac{1}{s}$): O bloco k_{is} (ganho integral) e o bloco 1/s (integrador) atuam na integral do erro de velocidade. Este termo é fundamental para eliminar o erro de regime permanente. Ele acumula o erro ao longo do tempo, garantindo que mesmo um pequeno erro persistente seja eventualmente corrigido.
  - Somador Final do PI: A saída dos ramos proporcional e integral é somada para gerar o sinal de controle principal do PI. Este sinal, antes da limitação, representa o que seria a referência de corrente (I^{*}_{a}) ideal para corrigir o erro de velocidade.
- Anti-Windup ($\frac{1}{k_{ps}}$ e Somadores): Este é um circuito de realimentação inteligente projetado para mitigar o fenômeno de windup integral.
  - **Saturador**: O bloco de saturação representa as **limitações físicas** do sistema. A saída do controlador de velocidade (que é a referência de corrente para a malha interna) não pode ser

infinita; ela é limitada pela capacidade máxima de corrente que o motor ou o inversor pode fornecer.

- **Detecção de Saturação**: Quando a saída do controlador PI excede esses limites (satura), o integrador continua acumulando erro. Isso leva a um valor excessivamente grande no integrador, que causa um atraso significativo na recuperação do sistema (um grande *overshoot* e tempo de assentamento prolongado) uma vez que o erro real comece a diminuir.
- Ação Anti-Windup: O laço de realimentação com o bloco $\frac{1}{k_{ps}}$ e os somadores operam da seguinte forma: ele mede a diferença entre a saída do PI antes e depois da saturação. Se houver saturação, essa diferença é realimentada, através do ganho $\frac{1}{k_{ps}}$, para subtrair do sinal de entrada do integrador ($\frac{1}{s}$). Isso "esvazia" o integrador quando a saída está saturada, evitando que ele acumule valores excessivos e permitindo uma recuperação mais rápida e suave do controle.

- Saída de Referência de Corrente (I_a^*): Representada pelo terminal 1 no lado direito, a saída final deste subsistema é a referência de corrente para a malha interna. Este sinal, limitado pelo saturador e ajustado pelo anti-windup, informa à malha de controle de corrente qual o valor de corrente de armadura que deve ser estabelecido para que o motor atinja a velocidade desejada.

5.2.2. Malha de controle da Corrente:

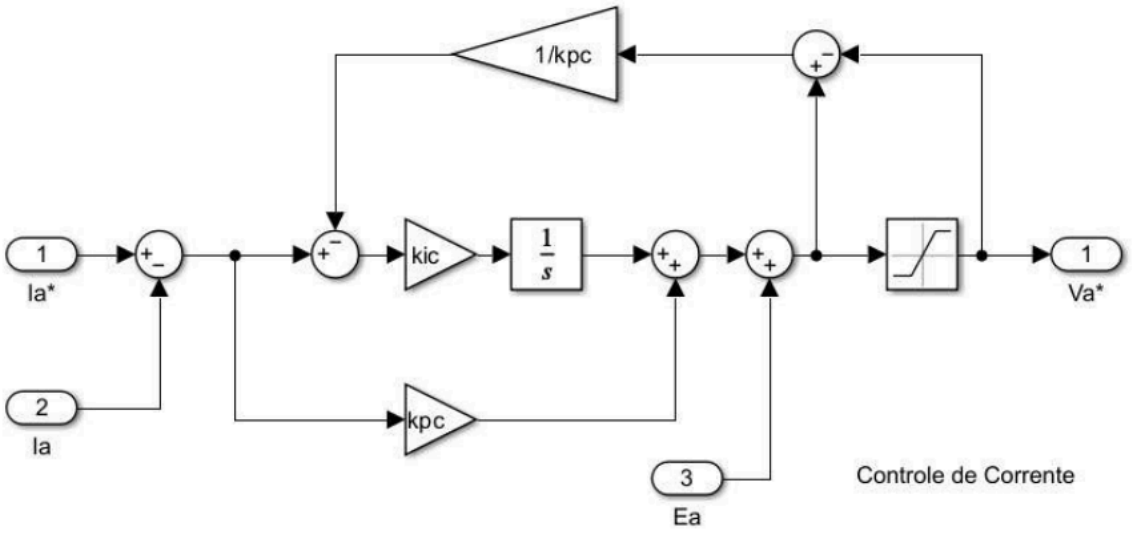

Figura 10: *Bloco de Controle de Corrente*

O subsistema "Controle de Corrente" é o coração da malha interna do seu sistema de controle em cascata. Sua principal função é regular a corrente de armadura (I_a) para que ela siga com precisão e rapidez a referência de corrente (I_a^*). Além disso, ele deve compensar a perturbação da força contra-eletromotriz (E_a) do motor.

Em detalhamento do seu bloco temos:

Entrada de Referência (I_a^*): Representada pelo terminal 1, esta é a **referência de corrente** gerada pelo controlador de velocidade. Este sinal indica o nível de corrente que o controlador de corrente deve estabelecer no motor para atingir a velocidade desejada.

Sinal de Feedback (I_a Representado pelo terminal 2, este é o sinal de feedback da corrente de armadura real do motor. Ele é o que permite ao controlador saber qual a corrente real fluindo no motor.

Comparador de Erro (+ -): O primeiro círculo à esquerda é um somador que calcula o erro de corrente, subtraindo a corrente medida da corrente de referência:

$$e_i = I_a^* - I_a$$

Sendo o erro a ser eliminado pelo controlador.

Controle Antecipativo (Feedforward): O terminal 3 (Ea) representa o sinal de força contra-eletromotriz (fcem) do motor. Este sinal é uma perturbação para a malha de controle de corrente, pois a fcem se opõe à tensão aplicada, afetando a corrente. A sua inclusão no somador com um sinal + (positivo) é uma implementação do controle antecipativo. Ao somar a fcem, o controlador compensa a perturbação de forma proativa, antes que ela cause um erro na corrente. Isso melhora drasticamente a velocidade e a precisão da resposta.

Circuito Anti-Windup: Este é um circuito de realimentação que utiliza o bloco 1/kpc e os somadores. Sua função é prevenir o windup integral, que ocorre quando a saída do controlador PI satura devido às limitações físicas de tensão. O circuito de anti-windup realimenta a diferença entre a saída do PI (limitada pelo saturador) e a saída do somador, corrigindo o valor do integrador ($\frac{1}{s}$), para evitar que ele acumule um erro excessivo. Isso garante uma recuperação mais rápida e sem grandes overshoots quando a saturação termina.

Saturador: O bloco com o limitador no meio do diagrama representa a saturação física da tensão de saída. Na prática, o inversor que aciona o motor possui um limite máximo e mínimo de tensão que pode aplicar, e este bloco simula essa limitação.

Saída de Referência de Tensão (V_a^*): Representada pelo terminal 1 à direita, a saída final deste subsistema é a **tensão de referência** que será enviada para o bloco PWM. Esta tensão, ajustada pelo controlador PI, pela compensação antecipativa e pela saturação, é o sinal que, de fato, comandará o acionamento do motor.

5.2.3. Malha PWM:

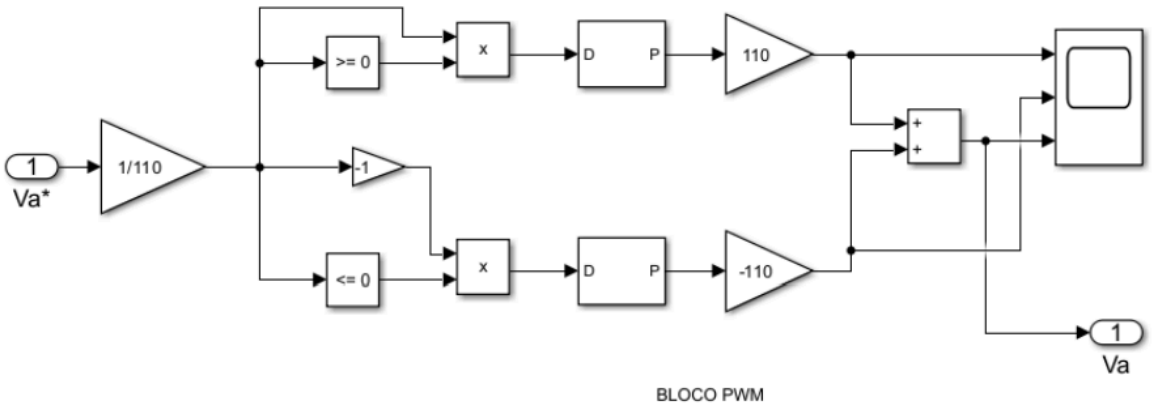

***Figura 11**: Blocos da malha de PWM, disponibilizados pelo professor Fernando Bento Silva.*

O subsistema "BLOCO PWM" é responsável por converter a tensão de referência contínua (V_a^*) gerada pelo controlador de corrente em um sinal de tensão pulsado (V_a) que simula a operação de um inversor de potência (como um chopper de quatro quadrantes). Este processo é essencial, pois os motores CC são acionados na prática por meio de chaveamento de tensão, e não por uma tensão contínua e variável diretamente.

Detalhando o funcionamento de cada parte deste bloco:

- Entrada de Tensão de Referência (V_a^*): Representada pelo terminal 1 no lado esquerdo, esta é a **tensão de armadura de referência** fornecida pelo controlador de corrente. É um sinal contínuo que indica a amplitude da tensão que o motor deve receber.
- Divisor de Escala (1/110): O primeiro bloco de ganho (1/110) escala a tensão de referência. Como a nossa tensão máxima, do sistema, é 110V (como indicado pelos ganhos de 110 e -110 mais adiante), este bloco normaliza o sinal de referência para uma faixa entre -1 e 1. Por exemplo, se V_a for 110V, a

saída será 1; se for -110V, a saída será -1. Isso é comum em sistemas PWM para trabalhar com uma referência percentual ou normalizada.

- Lógica de Geração de Sinais Positivos e Negativos: O bloco PWM é projetado para simular um inversor de quatro quadrantes, o que significa que ele pode aplicar tensões positivas e negativas ao motor, permitindo operação nos dois sentidos e frenagem. Isso é feito através de dois caminhos paralelos:
    - Caminho Superior (Tensão Positiva):

- Comparador (>= 0): Este bloco verifica se o sinal normalizado é maior ou igual a zero. Se for, ele gera um sinal lógico 1; caso contrário, 0.
- Multiplicador (x): Multiplica o sinal lógico (0 ou 1) pela amplitude do sinal normalizado.
- Bloco D P (PWM): Este bloco, que geralmente representa um gerador de PWM, recebe o sinal modulado (entre 0 e 1) e gera pulsos com largura proporcional à amplitude do sinal de entrada. Quanto maior a amplitude, maior a largura do pulso (duty cycle).
- Ganho (110): Multiplica a saída do gerador PWM pela tensão máxima positiva (110V). Isso resulta em uma tensão pulsada que varia entre 0V e 110V.

    - Caminho Inferior (Tensão Negativa):

- Ganho (-1): Este bloco inverte o sinal normalizado, transformando valores positivos em negativos e vice-versa.
- Comparador (<= 0): Verifica se o sinal invertido é menor ou igual a zero. Se for, gera um sinal lógico 1; caso contrário, 0.
- Multiplicador (x): Multiplica o sinal lógico (0 ou 1) pela amplitude do sinal invertido.
- Bloco D P (PWM): Gera pulsos PWM com base na amplitude do sinal invertido.
- Ganho (-110): Multiplica a saída do gerador PWM pela tensão máxima negativa (-110V). Isso resulta em uma tensão pulsada que varia entre 0V e -110V.

- Somador Final (+ +): As saídas dos caminhos positivo e negativo são somadas. Como apenas um dos caminhos estará ativo por vez (dependendo se a referência é positiva ou negativa), o somador recombina os pulsos PWM, resultando em uma tensão pulsada que pode variar de -110V a 110V.

- Saída de Tensão de Armadura (Va): Representada pelo terminal 1 no lado direito, esta é a tensão pulsada real que será aplicada ao subsistema "Modelo Motor".

- Scope/Visualizador: O bloco no canto superior direito é um Scope ou visualizador, que permite observar a forma de onda da tensão PWM gerada, o que é útil para verificar a operação correta do modulador.

5.2.4. Modelo do MOTOR:

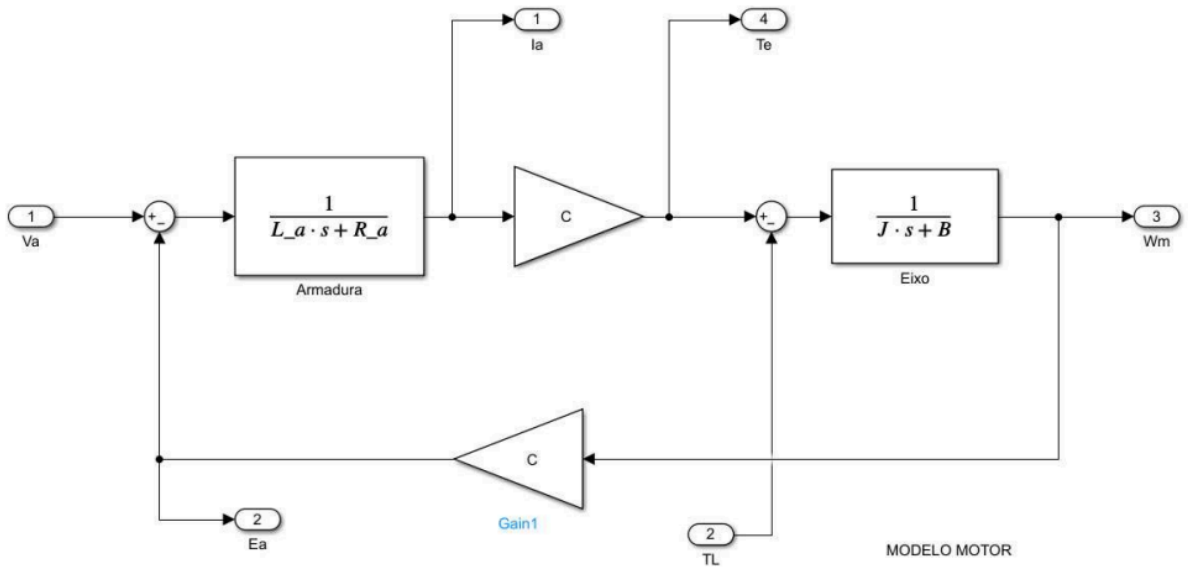

***Figura 12**: Blocos da malha do motor, disponibilizados pelo professor Fernando Bento Silva.*

O subsistema "Modelo Motor" é a representação em diagrama de blocos do comportamento dinâmico do motor de corrente contínua de ímã permanente. Ele

simula como as variáveis de controle (tensão e torque de carga) afetam a corrente e a velocidade do motor. O modelo está dividido em dois domínios interligados: o elétrico (armadura) e o mecânico (eixo).

- Domínio Elétrico (Armadura)

O circuito elétrico do motor é modelado pela equação de tensão de Kirchhoff. No diagrama de blocos, ele é representado pelo primeiro bloco à esquerda:

- Bloco 1/(L_a*s + R_a): Este bloco representa a impedância elétrica da armadura. Ele recebe a tensão líquida aplicada à armadura e, em resposta, determina a corrente. A resistência (R_a) e a indutância (L_a) da armadura são parâmetros cruciais definidos em seu código MATLAB.
- Sinal de Entrada (V_a): É a tensão de armadura que vem do bloco PWM.
- Sinal de Feedback (E_aRepresentado pelo terminal 2 e com um sinal de subtração (−), é a **força contra-eletromotriz (fcem)**. A fcem se opõe à tensão de armadura, atuando como um feedback interno da velocidade do motor.O somador na entrada do bloco de armadura calcula a V_L a que impulsiona a corrente:

$$V_L = V_a - E_a$$

A saída do bloco é a corrente de armadura (I_a), que é um dos principais sinais de feedback para a malha de corrente.

- **Interligação Eletromecânica**

A ligação entre os domínios elétrico e mecânico é feita pela constante do motor (C), que é a constante de torque (k_T), e a constante de fcem (k_e).

- Bloco de Ganho C:

Este bloco converte a corrente de armadura (I_a) em torque eletromagnético (Te), onde a relação é dada por:

$$T_e = C * I_a$$

Este torque é o "motor" do sistema mecânico.

- Bloco de Ganho C (Feedback):

O segundo bloco de ganho C no laço de feedback mecânico é o que gera a força contra-eletromotriz (E_a) a partir da velocidade angular (ω_m), a qual a relação é

$$E_a = C * \omega_m$$

A fcem é realimentada para a entrada da armadura, fechando o circuito eletromecânico.

- **Domínio Mecânico (Eixo)**

A dinâmica do eixo é governada pela equação do movimento, que relaciona o torque líquido com a aceleração.

- Bloco 1/(J*s + B): Este bloco modela a dinâmica mecânica do motor, que é determinada pelo momento de inércia (J) e pelo coeficiente de atrito viscoso (B). Ele recebe o torque líquido e, em resposta, determina a velocidade angular.
- Sinal de Entrada (T_e): O torque eletromagnético, que impulsiona o eixo.
- Sinal de Feedback (T_L): Representado pelo terminal 2 e com um sinal de subtração (-), é o torque de carga que se opõe ao movimento do motor. O somador na entrada do bloco do eixo calcula o torque da carga (:

$$T_L = T_e - T_L$$

A saída deste bloco é a velocidade angular (ω_m), que é a principal variável controlada pelo sistema.

6. MONTAGEM PRÁTICA

Conforme orientação do docente, foi montado um circuito, similar ao apresentado como exemplo, com o objetivo de realizar o controle do motor de corrente contínua por meio de um microcontrolador ESP32. A partir dessa instrução, procederam-se ao cálculo e à definição de todos os componentes necessários, sob supervisão e orientação do professor.

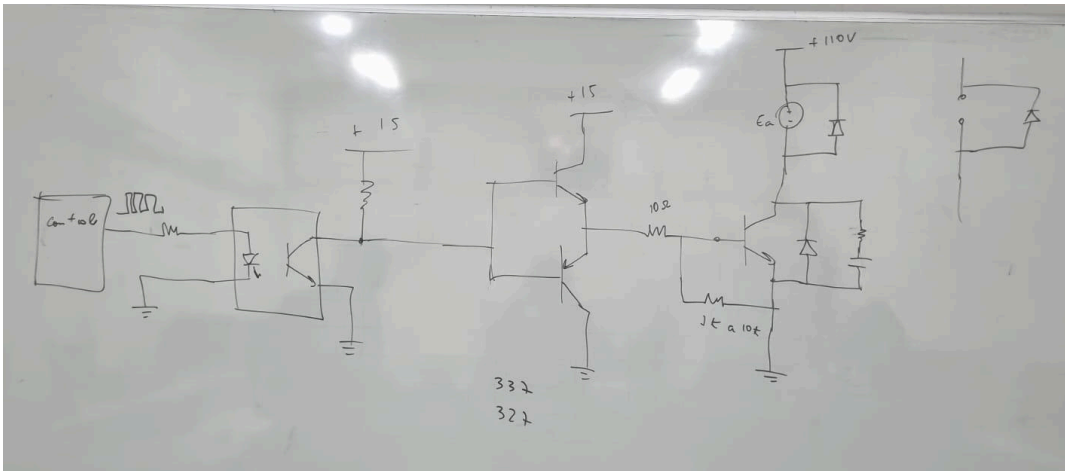

Figura 13: *Foto retirada em sala de aula do quadro branco, do sistema indicado pelo professor Fernando Bento Silva.*

A plataforma *EasyEDA* foi utilizada para a elaboração do esquemático, bem como para o desenvolvimento da placa de circuito impresso (PCB) e da representação 3D. Nos tópicos a seguir, serão detalhadas cada uma das etapas desse processo.

6.1. Materiais/Componentes:

Lista de materiais utilizados em nossa montagem:

- 9 conectores KF301-2P;
- 1 diodo MUR1560;
- 1 MOSFET LSB60R039GT;
- 1 resistor 10 Ω;

- 3 resistores 4,7 kΩ;
- 2 resistores 1 kΩ;
- 1 resistor 10 kΩ;
- 1 resistor de 3.3kΩ;
- 1 resistor de 3300Ω;
- Microcontrolador ESP32 DEVKIT V1;
- 1 transistor BC327;
- 1 transistor BC337;
- 1 amplificador operacional LM358N;
- 1 sensor de corrente ACS712 5A;
- 1 Encoder E6B2-CWZ3E.

6.2. Esquemáticos

O esquemático desenvolvido representa a eletrônica de potência e de controle necessária para o acionamento do motor de corrente contínua, com base no projeto de controle em cascata. Ele integra a placa de microcontrolador ESP32 aos circuitos de acionamento de potência e de sensoriamento. Nos tópicos a seguir, cada seção é detalhada.

6.2.1. Fonte de Alimentação

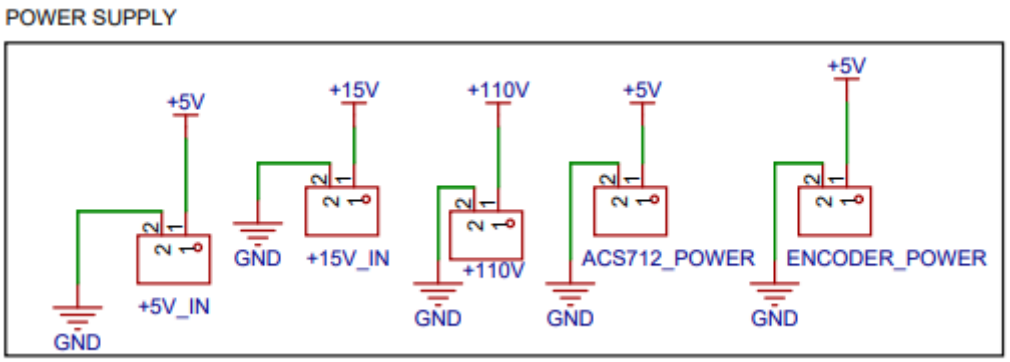

Figura 14: Representação esquemática da alimentação do sistema elaborada no EasyEDA

O circuito utiliza diferentes fontes de tensão para garantir a operação dos componentes de forma isolada e segura:

- **+5V:** Alimenta o microcontrolador ESP32 (+5V_IN) e o sensor de corrente ACS712, sendo fornecida por um circuito de controle (*power supply*).
- **+15V:** Utilizada para a alimentação do circuito de acionamento do motor, garantindo o sinal de controle necessário ao chaveamento dos transistores de potência.
- **+110V:** Fonte principal de alta potência, conectada diretamente à armadura do motor, responsável pelo acionamento efetivo.

6.2.2. Pinagem ESP32

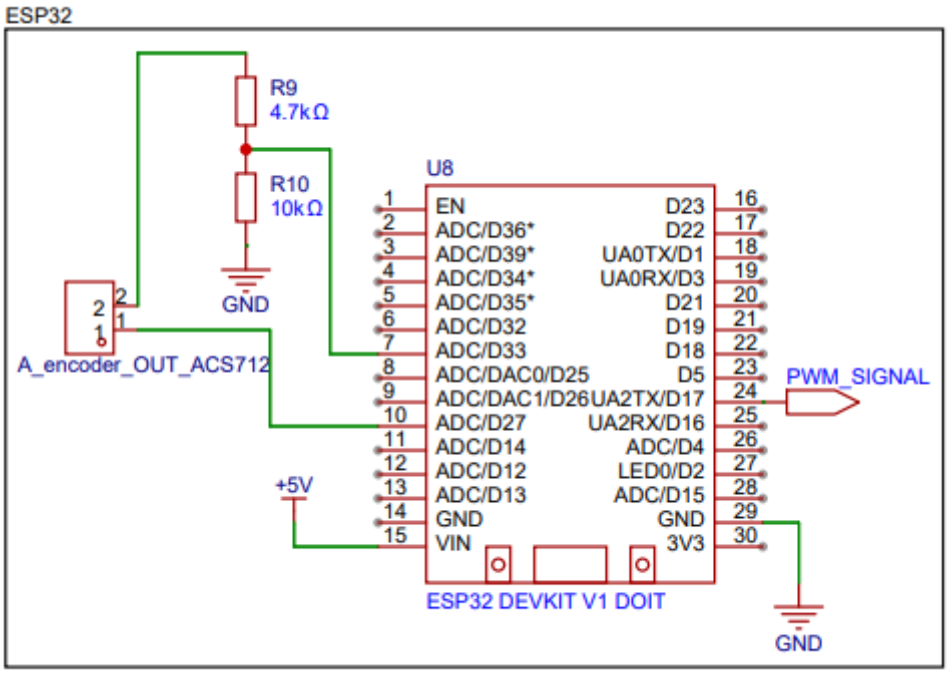

Figura 15: Representação esquemática das conexões do ESP32 elaborada no EasyEDA

O diagrama apresenta a pinagem do módulo **ESP32 DEVKIT V1 DOIT**, que atua como unidade central de processamento do sistema de controle. Cada pino é configurado para uma função específica:

- **Entradas analógicas (ADC):** Diversos pinos, como ADC/D36, ADC/D39, ADC/D34, ADC/D35, ADC/D32 e ADC/D33, são destinados à leitura de sinais analógicos. No circuito implementado, o pino ADC/D36 recebe o sinal de tensão *A_encoder_OUT_ACS712*.

- **Saídas digitais (DXX):** Pinos como D23 e D22 são utilizados para comunicação e controle.
- **Saída PWM**: O pino D25 (ADC/DAC0/D25) gera o sinal PWM (*Pulse Width Modulation*), equivalente ao valor de controle (*Va* da simulação), enviado ao circuito de acionamento de potência.
- **Pinos de alimentação e GND:** Incluem +5V, VIN, 3V3 e GND, responsáveis pela alimentação e referência elétrica da placa.

6.2.3. Circuito de controle de potência:

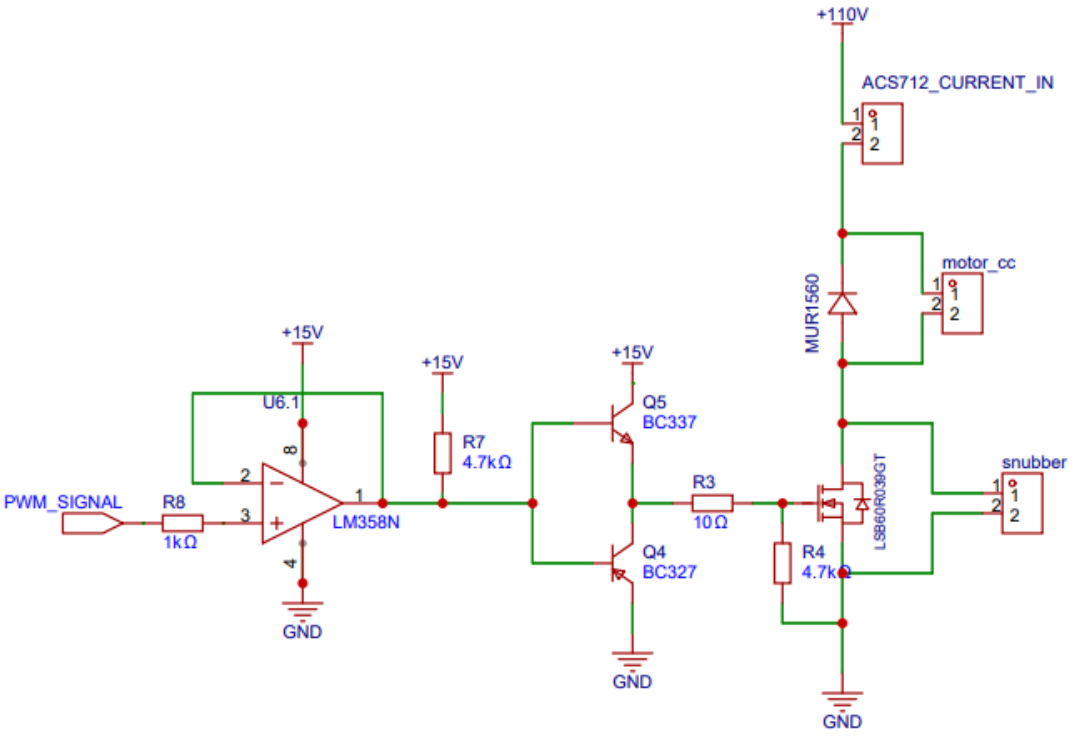

***Figura 16**:* Representação esquemática do circuito de controle de potência elaborada no EasyEDA

Este circuito realiza a interface entre o sinal de baixo nível fornecido pelo ESP32 e a potência de alta tensão necessária para acionar o motor. Ele é composto por um driver de transistores, um sensor de corrente e o próprio motor.

- **Sinal PWM:** Gerado pelo ESP32, é aplicado ao circuito de controle de potência.

- **Driver de transistores (BC337 e BC327):** Utilizados em configuração BJT para amplificação do sinal PWM, garantindo amplitude suficiente para acionar os transistores de potência. Resistores (R7, R8, R4, R3) realizam a polarização e limitação de corrente.
- **Sensor de corrente (ACS712):** Baseado em efeito Hall, mede a corrente de armadura do motor e fornece o sinal de realimentação para a malha de controle.
- **Componentes de potência (MUR1560 e LSB60R039GT):** O diodo MUR1560 e o MOSFET LSB60R039GT controlam o fluxo de corrente da fonte de alta tensão (+110V) para o motor. Os diodos asseguram a condução adequada, enquanto a rede *snubber* e componentes passivos associados protegem o circuito contra surtos e ruídos.

6.2.4. Circuito Completo (Visão Geral)

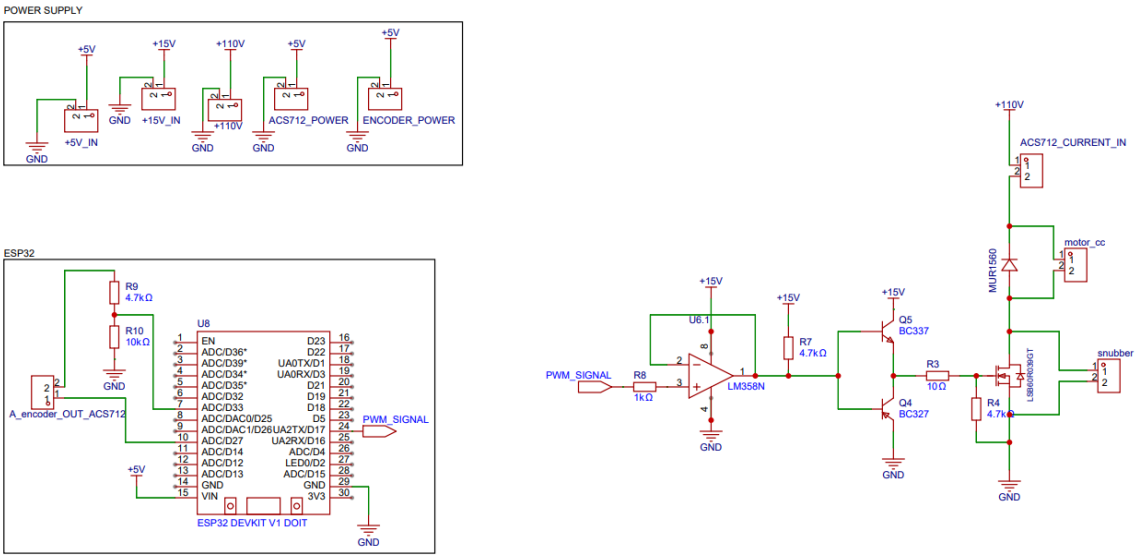

Figura 17: *Representação esquemática do circuito completo elaborada no EasyEDA*

O circuito completo integra todas as seções. A lógica de controle é implementada no ESP32, que recebe o sinal analógico do sensor ACS712 e gera o sinal PWM. Esse sinal é amplificado pelo circuito de acionamento (com auxílio do LM358N) e aplicado aos transistores de potência (BC337, BC327 e LSB60R039GT), responsáveis pela aplicação da tensão de 110 V na armadura do motor. O sensor de

corrente ACS712 fornece a realimentação de corrente ao ESP32, fechando a malha de controle.

6.3. Finalização do protótipo:

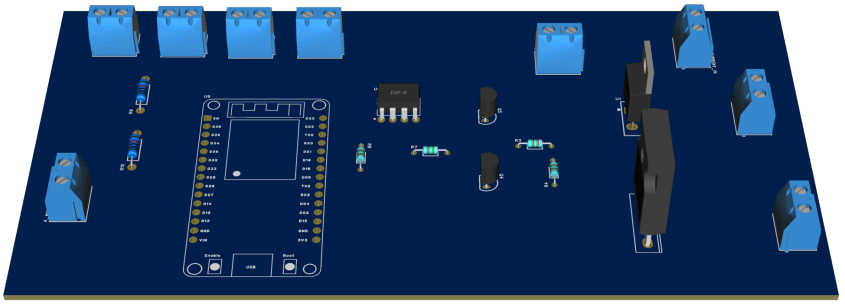

Figura 18: *Visualização CAD 3D da PCB, elaborada no EasyEDA.*

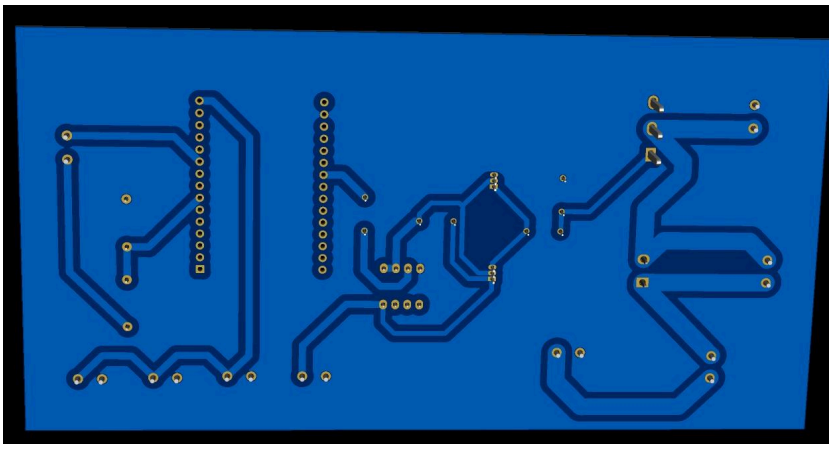

Figura 18: *Trilhagem da PCB, elaborada no EasyEDA.*

Após a modelagem e a análise em 3D, a placa foi produzida e, posteriormente, realizada a fixação dos componentes por meio de soldagem com estanho. Em 28/08/2025, foram conduzidos no laboratório os testes de funcionamento, verificação da placa e validação do código implementado no ESP32.

7. CÓDIGO DE CONTROLE

7.1. Simulação Eletrônica

A análise do circuito de acionamento do MOSFET evidencia falhas críticas de projeto no estágio de saída *push-pull*, que comprometem seu funcionamento adequado.

O arranjo, composto pelos transistores Q1 e Q2 em configuração de coletor comum, foi projetado para atuar como driver de *gate*, fornecendo alta corrente tanto no acionamento quanto no desligamento do MOSFET. Entretanto, a implementação apresenta um erro fundamental que inviabiliza sua operação como um verdadeiro circuito *push-pull*.

O problema mais grave encontra-se na polarização do transistor PNP (Q2), responsável pela etapa *pull*, ou seja, pela drenagem da corrente do *gate* do MOSFET, possibilitando o desligamento rápido. De acordo com o *datasheet* do LSB60R039GT, a tensão de chaveamento do dispositivo é maior ou igual a 10 V. Contudo, o emissor de Q2 está conectado ao potencial de referência (0 V), o que exigiria uma tensão de base de aproximadamente –0,7 V para sua ativação. Como o amplificador operacional LM358, que o antecede, é alimentado por fonte simples (+15 V e 0 V), ele não é capaz de gerar a tensão negativa necessária, mantendo o transistor Q2 permanentemente em corte.

Dessa forma, o circuito opera apenas em modo *push*, utilizando o transistor NPN (Q1) para fornecer corrente e ligar o MOSFET. O desligamento, por sua vez, ocorre de forma passiva e ineficiente, dependendo exclusivamente do resistor de *pull-down* R3 para descarregar a capacitância do *gate*. Essa lentidão no desligamento é inadequada para aplicações de chaveamento em alta frequência, como no controle PWM, pois mantém o MOSFET na região linear por tempo prolongado, ocasionando elevadas perdas por comutação e aquecimento excessivo.

Adicionalmente, a malha de realimentação do amplificador operacional está incorretamente posicionada, monitorando sua própria saída em vez da saída do estágio *push-pull*. Esse arranjo impede a compensação da queda de tensão base-emissor (V_{BE}) dos transistores, resultando em imprecisão na tensão final aplicada ao *gate*.

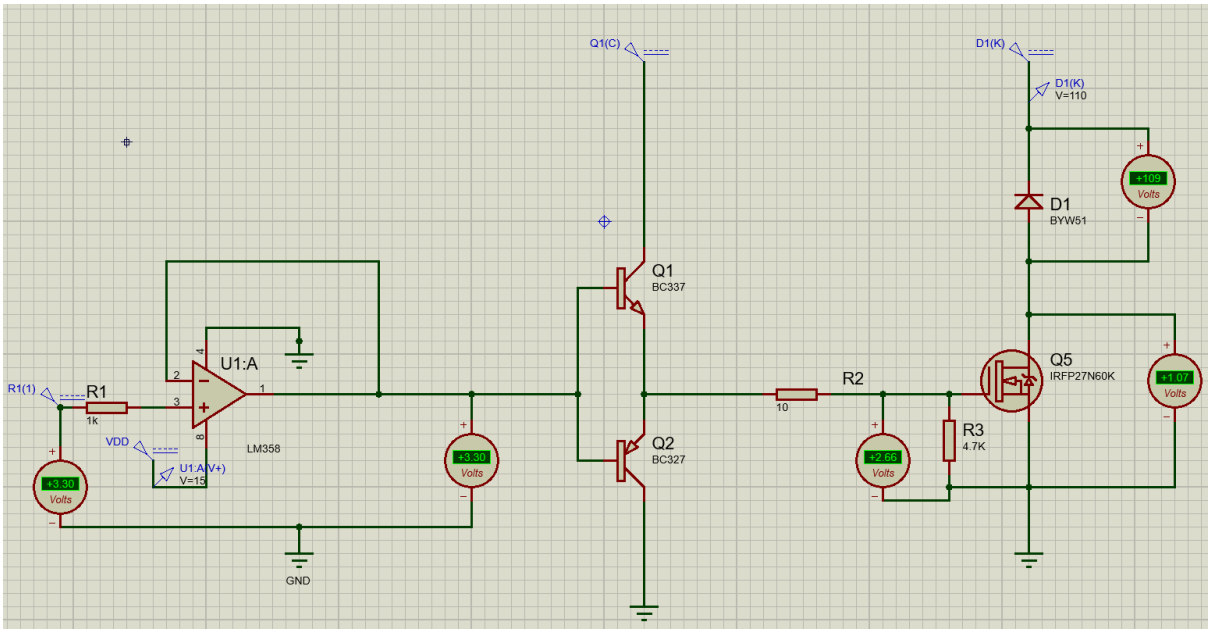

Figura 19: *Simulação de acionamento do gate via circuito push-pull*

Correção Implementada

Para solucionar a limitação da tensão de chaveamento do MOSFET, foi realizada uma modificação na configuração da entrada inversora do LM358, empregando-se um divisor de tensão. A alteração consistiu em modificar a topologia do LM358, passando de seguidor de tensão para amplificador não inversor, utilizando-se os resistores R3 (3,3 kΩ) e R5 (10 kΩ) na malha de realimentação negativa, estabelecendo ganho de tensão.

Essa modificação possibilitou a amplificação do sinal de entrada, de +3,30 V para aproximadamente +13,3 V na saída do amplificador operacional. Consequentemente, a tensão aplicada ao *gate* do MOSFET, após a queda de tensão no transistor Q1, atingiu cerca de +12,6 V — valor suficiente para levar o

MOSFET Q5 à saturação completa, resultando em tensão de dreno-fonte próxima de 0,00 V e assegurando sua operação eficiente como chave totalmente fechada.

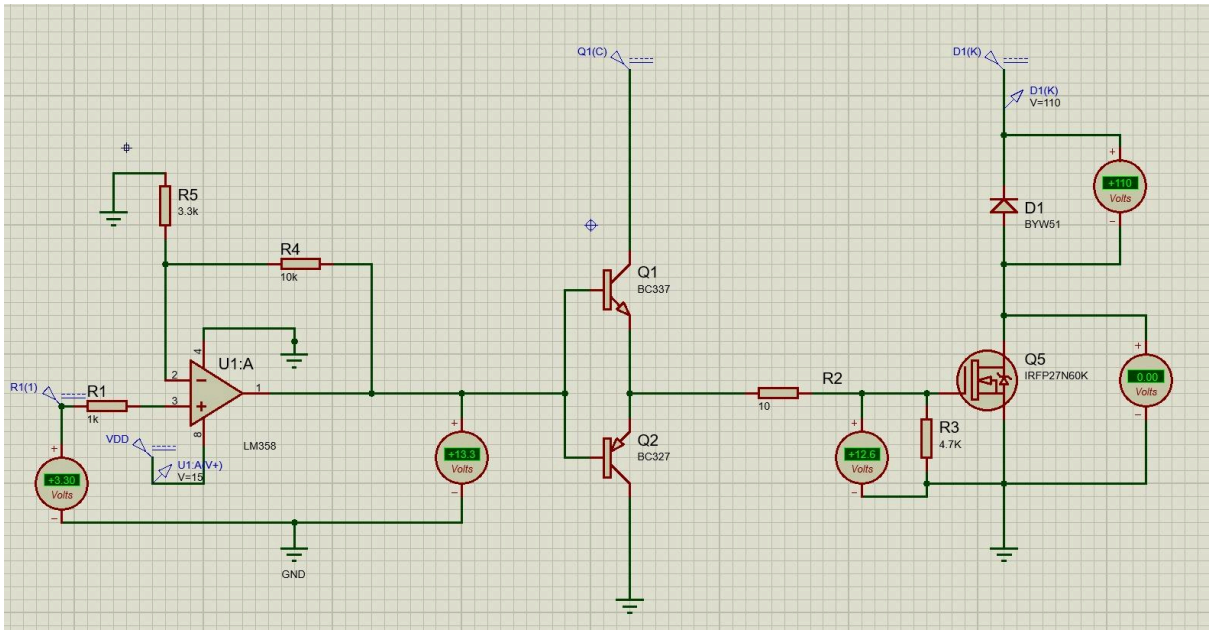

Figura 20: *Simulação eletrônica realizada no Proteus divisor de tensão*

7.2. Código para Recepção e Validação de Dados

Este script em Arduino foi desenvolvido para a validação dos dados brutos provenientes dos sensores de velocidade (encoder) e de corrente (ACS712). Sua função principal é garantir que as medições de ambos os sensores sejam precisas e confiáveis, servindo como uma etapa de pré-requisito para o desenvolvimento do algoritmo de controle em malha fechada.

A medição da velocidade é realizada por meio de uma **interrupção de *hardware*** (countPulse()) que incrementa uma variável a cada pulso detectado pelo encoder. Para o cálculo do RPM, o código utiliza a função millis() em uma abordagem não-bloqueante, calculando a velocidade a cada 100 milissegundos sem interromper outras tarefas. A medição de corrente é feita a partir da leitura analógica do sensor ACS712, com o valor bruto do ADC sendo convertido para amperes. Um filtro de

média móvel (filtroMediamovel()) é aplicado para suavizar a medição e remover ruídos, garantindo um sinal de feedback mais estável.

Os valores de RPM e corrente filtrada são então impressos no monitor serial, permitindo a visualização e a análise em tempo real dos dados. Essa etapa de validação foi crucial para assegurar a integridade dos sinais de entrada antes de sua utilização no algoritmo de controle principal.

```
#include <Arduino.h>

// --- Constantes e Variáveis ---

// Pino do ESP32 onde o pino de saída (OUT) do ACS712 está conectado.
// Use um pino que suporte ADC, como o GPIO 34.
const int pinoSensor = 27;
const int ENCODER_PIN_A = 34;

volatile long pulseCount = 0;

const int PPR = 1024;
unsigned long lastTime = 0;
float rpm = 0.0;
const int calculationInterval = 100;
// A tensão de referência do ADC do ESP32 é de 3.3V.
const float tensaoReferencia = 3.3;


const float sensibilidade = 0.185; // Para o módulo de 5A
//const float sensibilidade = 0.100; // Para o módulo de 20A
// const float sensibilidade = 0.066; // Para o módulo de 30A
int valorADC = 0;
const int numReadings = 200;
int readings[numReadings], readIndex, total;



void IRAM_ATTR countPulse() {
  pulseCount++;
}
```

```
float filtroMediamovel(float newValue){
  total = total - readings[readIndex];
  readings[readIndex] = newValue;
  total = total + readings[readIndex];
  readIndex = readIndex + 1;

  if(readIndex >= numReadings) readIndex = 0;

  return (float)total/numReadings;
}

void setup() {
  // Inicia a comunicação serial a uma taxa de 115200 bauds.
  Serial.begin(115200);

  pinMode(ENCODER_PIN_A, INPUT);
  attachInterrupt(digitalPinToInterrupt(ENCODER_PIN_A), countPulse, RISING);

    lastTime = millis();

  Serial.println("Leitor de Encoder RPM iniciado.");
  Serial.print("PPR configurado: ");
  Serial.println(PPR);

  // Define a resolução do ADC para 12 bits (0-4095), que é o padrão do ESP32.
  // Isso ajuda na precisão da leitura.
  analogReadResolution(12);

  Serial.println("Iniciando leitura do sensor de corrente ACS712...");
  delay(1000); // Pequeno atraso para estabilização.
}

void loop() {
  // Lê o valor analógico bruto do pino do sensor.
  valorADC = analogRead(pinoSensor);

  // Converte o valor lido (0-4095) para tensão (0-3.3V).
  float tensao = (valorADC * tensaoReferencia) / 4095.0;
  // Corrente (A) = (Tensão Medida - Tensão de Offset) / Sensibilidade
  float corrente = (tensao - (2.37)) / sensibilidade;
```

```
  // Imprime os resultados no Monitor Serial.
  /*Serial.print("Valor ADC: ");
  Serial.print(valorADC);

  Serial.print(" | Tensão (V): ");
  Serial.print(tensao, 3); // Imprime a tensão com 3 casas decimais.

  Serial.print(" | Corrente (A): ");
  Serial.print(corrente, 2); // Imprime a corrente com 2 casas decimais.*/

  float corrente_Filtrada = filtroMediamovel(tensao);
  Serial.print(" | corrente filtrada: ");
  Serial.println(corrente_Filtrada); // Imprime a corrente com 3 casas
decimais.

  // Aguarda um pouco antes da próxima leitura.
  delay(50);

    if (millis() - lastTime >= calculationInterval) {

    // --- Início da Seção Crítica ---
    // Desabilita as interrupções temporariamente para ler e zerar a contagem
de pulsos.
    // Isso previne que um pulso chegue bem no meio da leitura, o que causaria
um erro.
    noInterrupts();
    long pulses = pulseCount; // Copia a contagem para uma variável local
    pulseCount = 0;           // Zera o contador para o próximo intervalo
    interrupts();             // Habilita as interrupções novamente o mais
rápido possível
    // --- Fim da Seção Crítica ---

    // Atualiza o tempo da última medição
    lastTime = millis();

    /*
     * Fórmula do RPM:
     * Pulsos por segundo (PPS) = pulses / (calculationInterval / 1000.0)
     * Rotações por segundo (RPS) = PPS / PPR
     * Rotações por minuto (RPM) = RPS * 60
```

```
     *
     * Simplificando:
     * RPM = (pulses / PPR) * (60 * 1000 / calculationInterval)
     */
    rpm = (float)(pulses * 60.0 * 1000.0) / (PPR * calculationInterval);

    // Imprime o resultado no Monitor Serial
    Serial.print("Pulsos no intervalo: ");
    Serial.print(pulses);
    Serial.print(" | Velocidade: ");
    Serial.print(rpm, 2); // Imprime com 2 casas decimais
    Serial.println(" RPM");
  }
}
```

7.3. Implementação do Código de Controle

O código a seguir é a implementação final, em linguagem C++, do sistema de controle PI em cascata, desenvolvido para o microcontrolador ESP32. Ele integra a lógica de controle projetada em MATLAB, utilizando os coeficientes discretizados **b0** e **b1**, com o acionamento e as medições em tempo real do motor. O código é estruturado com um laço principal que opera de forma não-bloqueante, garantindo que as malhas interna (corrente, 1 ms) e externa (velocidade, 10 ms) sejam executadas em suas respectivas frequências, o que é fundamental para a estabilidade e o desempenho do sistema. Por meio da leitura dos sensores de corrente (ACS712) e de velocidade (encoder) e da geração de um sinal PWM, o código regula o motor de forma precisa, validando a teoria de controle em um ambiente prático.

```
#include <Arduino.h>
#include <algorithm>
#include "esp_adc_cal.h"

// Estrutura para calibração do ADC
static esp_adc_cal_characteristics_t *adc_chars; // Objeto para armazenar as
características de calibração

// ----------------- PINOS -----------------
```

```
const int PINO_SENSOR_CORRENTE = 27;
const int PINO_ENCODER_A = 33;
const int PULSOS_POR_ROTACAO = 1024;
// ----------------- Parametro PWM -----------------
const int PINO_PWM = 17;
const int CHANNEL_PWM = 0;
const int FREQUENCIA_PWM = 5000;
const int RESOLUCAO_PWM = 8;
// ----------------- PARAMS SENSOR ACS712 -----------------
const float SENSOR_SENSIBILIDADE = 0.185; // 185 mV/A (5A)
const float SENSOR_OFFSET_VOLTS = 2.5; // será calibrado

// ----------------- FILTROS -----------------
const int numReadings = 200;
float readings[numReadings];
int readIndex = 0;
float total = 0;

float corrente_definitiva = 0.0;

// ----------------- VARIÁVEIS DE CONTROLE -----------------
float rpm = 0.0, rad_s = 0.0;
volatile uint32_t pulsos = 0;
unsigned long tempo_anterior_interno = 0;
unsigned long tempo_anterior_externo = 0;
unsigned long tempo_atual = 0;
const unsigned long tempo_malha_interna = 1000;   // 1 ms
const unsigned long tempo_malha_externa = 10000;  // 10 ms

const float setpoint_velocidade = 5000 * 0.10472;

// Controlador corrente
float d_c = 0.0;
int duty_cycle_c = 0;
float u_c = 0.0, ek_c = 0.0, uk_1c = 0.0, ek_1c = 0.0;
// Controlador velocidade
float d_w = 0.0;
float u_w = 0.0, ek_w = 0.0, uk_1w = 0.0, ek_1w = 0.0;
// Ganhos PI
/*float b0_c = 0.109;
float b1_c = 0.0748*b0_c;
float b0_w = 0.111;
```

```
float b1_w = 0.222*b0_w;*/

// Ganho PI lento e sem overshoot
/*const float b0_c = 0.083;
const float b1_c = -0.082;
const float b0_w = 0.438;
const float b1_w = -0.432;*/

//Ganho PI para de 1ms e 10ms, rápido, mas com overshoot
const float b0_c = 0.112;
const float b1_c = -0.053;
const float b0_w = 0.572;
const float b1_w = -0.296;

float filtroMediamovel(float newValue);
float lerCorrente();
float controlador_corrente(float ref, float med);
float controlador_velocidade(float ref, float med);
void contarPulsos();

void setup() {
  Serial.begin(115200);

  // --- INICIALIZAÇÃO DA CALIBRAÇÃO DO ADC ---
             adc_chars       =       (esp_adc_cal_characteristics_t*)calloc(1,
sizeof(esp_adc_cal_characteristics_t));
    esp_adc_cal_characterize(ADC_UNIT_1,  ADC_ATTEN_DB_11,  ADC_WIDTH_BIT_12,
1100, adc_chars);
  analogSetPinAttenuation(PINO_SENSOR_CORRENTE, ADC_11db);
  analogReadResolution(12);

  pinMode(PINO_ENCODER_A, INPUT);
       attachInterrupt(digitalPinToInterrupt(PINO_ENCODER_A),    contarPulsos,
RISING);
  ledcSetup(CHANNEL_PWM, FREQUENCIA_PWM, RESOLUCAO_PWM);
  ledcAttachPin(PINO_PWM, CHANNEL_PWM);

  ledcWrite(CHANNEL_PWM, 0);

  delay(10000);
}
```

```
void loop() {
  tempo_atual = micros();

  // ---------- Controle Interno (Corrente) ----------
  if (tempo_atual - tempo_anterior_interno >= tempo_malha_interna) {
    tempo_anterior_interno = tempo_atual;
    corrente_definitiva = -(lerCorrente());
    d_c = controlador_corrente(d_w, corrente_definitiva);
    duty_cycle_c = int(d_c);
    ledcWrite(CHANNEL_PWM, 255*duty_cycle_c/110);
  }

  // ---------- Controle Externo (Velocidade) ----------
  if (tempo_atual - tempo_anterior_externo >= tempo_malha_externa) {
    tempo_anterior_externo = tempo_atual;
    noInterrupts();
    uint32_t pulsosMedidos = pulsos;
       rpm  =  (pulsosMedidos  *  60.0  *  1000000.0)  /  (PULSOS_POR_ROTACAO  *
tempo_malha_externa);
    rad_s = rpm * 0.10472;
    pulsos = 0;
    interrupts();

    d_w = controlador_velocidade(setpoint_velocidade, rad_s);

    // Impressão padrão
     Serial.printf("Motor | Corrente: %.3f mA, Vel: %.2f RPM, Ref: %.2f RPM,
PWM: %d",
                  corrente_definitiva*1000,
                  rpm,
                  setpoint_velocidade / 0.10472,
                  255*duty_cycle_c/110);
    Serial.println(" ");

    // Impressões para o Teleplot
    Serial.printf(">rpm: %.2f\n", rpm);
    Serial.printf(">ref: %.2f\n", setpoint_velocidade / 0.10472);
    Serial.printf(">corrente_mA: %.2f\n", corrente_definitiva*1000);
    Serial.printf(">pwm: %d\n", 255*duty_cycle_c/110);
  }
}
```

```
float filtroMediamovel(float newValue){
  total = total - readings[readIndex];
  readings[readIndex] = newValue;
  total = total + readings[readIndex];
  readIndex = readIndex + 1;

  if(readIndex >= numReadings) readIndex = 0;

  return (float)total/numReadings;
}

float lerCorrente() {
  float adcValor = analogRead(PINO_SENSOR_CORRENTE);

  uint32_t tensao_mV = esp_adc_cal_raw_to_voltage(adcValor, adc_chars);
  float tensao = tensao_mV / 1000.0;

  float corrente = (tensao - SENSOR_OFFSET_VOLTS) / SENSOR_SENSIBILIDADE;
  float correnteFiltrada = -(filtroMediamovel(corrente))  - 0.560;
  return correnteFiltrada;
}

// ===================================================================
// ▼▼▼ Controladores ▼▼▼
// ===================================================================
float controlador_corrente(float r, float y) {
  ek_c = r - y;
  u_c = uk_1c + b0_c * ek_c + b1_c * ek_1c;

  if (u_c > 110) u_c = 110;
  else if (u_c < 0) u_c = 0;

  ek_1c = ek_c;
  uk_1c = u_c;
  return u_c;
}

float controlador_velocidade(float r, float y) {
  ek_w = r - y;
  u_w = uk_1w + b0_w * ek_w + b1_w * ek_1w;

  if (u_w > 5) u_w = 5;
```

```
  else if (u_w < 0) u_w = 0;

  ek_1w = ek_w;
  uk_1w = u_w;
  return u_w;
}

void IRAM_ATTR contarPulsos() {
  pulsos++;
}
```

8. RESULTADOS E DISCUSSÕES

A simulação em MATLAB/Simulink e a posterior implementação prática permitiram a análise do desempenho do sistema de controle em cascata sob diferentes condições. Os resultados obtidos validam a metodologia de projeto e demonstram o impacto crucial da sintonia dos controladores e dos tempos de amostragem na resposta do sistema.

Inicialmente, foram realizados testes com tempos de amostragem de 5 ms para a malha interna de corrente e 50 ms para a malha externa de velocidade. Conforme observado na Figura 21, a resposta do motor a um *setpoint* de 5000 RPM apresentou um comportamento dinâmico indesejado, com um **overshoot** significativo e oscilações persistentes tanto na velocidade quanto na corrente. Este resultado indicou que os tempos de amostragem iniciais não eram ideais para garantir a separação de escalas de tempo necessária para um controle em cascata eficiente.

Figura 21: Resposta do sistema com tempos de amostragem de 5 ms (corrente) e 50 ms (velocidade).

Para otimizar o desempenho, os tempos de amostragem foram reduzidos para 1 ms (malha de corrente) e 10 ms (malha de velocidade), conforme as recomendações teóricas. O resultado, ilustrado na Figura 22, demonstrou uma melhoria substancial. A resposta do sistema se tornou mais rápida e precisa, com o motor atingindo o *setpoint* de 5000 RPM com **menor overshoot** e uma **oscilação significativamente reduzida** em regime permanente. A malha de corrente operou de forma estável, confirmando que uma malha interna mais rápida é fundamental para o bom funcionamento do laço de velocidade..

Figura 22: Resposta otimizada com tempos de amostragem de 1 ms (corrente) e 10 ms (velocidade).

Por fim, a capacidade de rejeição de perturbações do controlador foi testada com a aplicação de um torque de carga no eixo do motor. A Figura 23 ilustra a resposta a essa perturbação. O sistema respondeu à carga com um aumento da corrente e do sinal PWM, mostrando que a malha de controle atuou para compensar a diminuição da velocidade. A velocidade do motor estabilizou em um novo patamar, ligeiramente acima do *setpoint*, o que demonstra a capacidade do controlador de atenuar o impacto da perturbação e estabilizar o sistema em uma nova condição de operação.

Figura 23: Resposta do controlador a uma **perturbação de torque**.

9. CONCLUSÃO

O desenvolvimento deste projeto consolidou os fundamentos teóricos do controle de velocidade de um motor de corrente contínua, passando pela modelagem matemática, caracterização experimental, simulação e implementação prática. As simulações validaram a estratégia de controle em cascata, demonstrando que a correta separação entre as malhas interna (corrente) e externa (velocidade) é crucial para um desempenho dinâmico otimizado.

A etapa prática, que permitiu a implementação bem-sucedida do sistema completo, confirmou a eficácia da abordagem em ambiente real. Os resultados experimentais, alinhados com os dados simulados, comprovaram que a malha de corrente operou conforme o esperado, garantindo uma resposta ágil a variações de

setpoint e uma robusta capacidade de rejeição a perturbações de carga. O projeto, portanto, não apenas atingiu seu objetivo de projetar um controlador, mas também validou de forma empírica a teoria de controle em cascata, superando os desafios práticos e demonstrando a transição bem-sucedida da simulação para a realidade.

10. REFERÊNCIAS

[1] Material de Auxílio didático teórico do professor Fernando Bento Silva
A inserir…

[2] Material Roteiro da coleta de dados disponibilizado pelo Fernando Bento Silva

[3] Planilha com nossos dados coletados: https://docs.google.com/spreadsheets/d/18s4svCK2z6HAoDCzgQRuxbDOvBkAeMVQ/edit?usp=sharing&ouid=104103331140277679800&rtpof=true&sd=true

[4] MATHWORKS. *MATLAB R2025a*. Natick, MA: The MathWorks Inc., 2025. Disponível em: https://www.mathworks.com/products/matlab.html. Acesso em: 12 ago. 2025.

[5] MATHWORKS. *Simulink*. Natick, MA: The MathWorks Inc., 2025. Disponível em: https://www.mathworks.com/products/simulink.html. Acesso em: 12 ago. 2025.

[6] UNIVERSIDADE FEDERAL DE UBERLÂNDIA. *Dinâmica e controle de velocidade do motor de corrente contínua com excitação independente*. Uberlândia, MG: Faculdade de Engenharia Elétrica, jun./jul. 2025.