



Universidade Federal  
do Rio de Janeiro  

---

Escola Politécnica

## UMA PROPOSTA DE LABORATÓRIO PARA O CONTROLE DE VELOCIDADE DE UM MOTOR CC UTILIZANDO ARDUINO

Fernanda Folly Ferreira

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

Orientadores: Públio Macedo Lima  
Wesley Rodrigues Silveira

Rio de Janeiro  
Março de 2019


UMA PROPOSTA DE LABORATÓRIO PARA O CONTROLE DE  
VELOCIDADE DE UM MOTOR CC UTILIZANDO ARDUINO

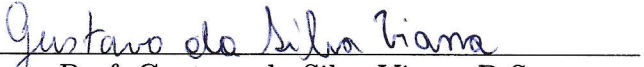
Fernanda Folly Ferreira


PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO  
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA  
POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO  
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU  
DE ENGENHEIRO DE AUTOMAÇÃO.

Examinado por:

  
Wesley Rodrigues Silveira, M.Sc.

  
Públio Macedo de Lima, M.Sc.

  
Prof. Gustavo da Silva Viana, D.Sc.

  
Prof. Marcos Vicente de Brito Moreira, D.Sc.

RIO DE JANEIRO, RJ – BRASIL  
MARÇO DE 2019

Ferreira, Fernanda Folly

Uma proposta de laboratório para o controle de velocidade de um motor CC utilizando Arduino/Fernanda Folly Ferreira. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2019.

XI, 61 p. 29, 7cm.

Orientadores: Públio Macedo Lima

Wesley Rodrigues Silveira

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2019.

Referências Bibliográficas: p. 54 – 54.

1. Controle de velocidade. 2. Arduino. 3. Motor CC.  
I. Lima, Públio Macedo. II. Silveira, Wesley Rodrigues. III. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. IV. Título.

# Agradecimentos

Agradeço, primeiramente, aos meus pais, Claudia e Alexandre, pelo apoio, incentivo e por sempre acreditarem em mim, eu não teria conseguido sem vocês. À minha irmã, Mylena, obrigada pelo carinho e torcida para a conclusão dessa etapa. Agradeço também aos meus avós, Nely, Zilda e Geraldo e aos meus tios e primos, meu muito obrigada.

Gostaria de agradecer ao Renan Calmon, por ter me apoiado, incentivado e ajudado em todos os momentos desde o início da nossa graduação. Você foi o melhor presente que a UFRJ me deu. Obrigada por me ensinar um pouquinho mais a cada dia.

Aos meus amigos de graduação, por me acompanharem ao longo do curso, vocês foram essenciais para minha formação. Obrigada a todos vocês por fazerem essa jornada um pouco mais divertida, em especial à Hannah Zacharias Carol Sales, Thais Silvestre, Laís Mesquita, Maíra Bernardo, Rôb Klér, Pedro Henrique Sevenini, Bruno Valério, Matheus Guedes, Gabriel Marinho, Isabella Quintanilha, Gabriel Loureiro e Adriana Sodré.

Às minhas amigas de escola, Julia, Monique, Larissa, Jaqueline, Juliana, Ester, e Déborah, por estarem comigo desde sempre.

Aos meus amigos do LCA, pela ajuda, incentivo e companhia na conclusão deste trabalho.

Aos meus orientadores, Wesley e Públio, por todas as correções, paciência, compreensão e pelo empenho dedicado à elaboração deste trabalho, muito obrigada.

Gostaria também de agradecer a todos que de alguma forma fizeram parte deste caminho, a todos vocês o meu muito obrigada.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Automação.

## UMA PROPOSTA DE LABORATÓRIO PARA O CONTROLE DE VELOCIDADE DE UM MOTOR CC UTILIZANDO ARDUINO

Fernanda Folly Ferreira

Março/2019

Orientadores: Públio Macedo Lima

Wesley Rodrigues Silveira

Curso: Engenharia de Controle e Automação

Apresenta-se, neste trabalho, o projeto de um controle PI para controlar a velocidade de um motor CC utilizando Arduino e ponte H para substituir o uso de amplificadores de potência nos experimentos feitos na disciplina Laboratório de Controle I do curso de graduação da Engenharia Elétrica.

Primeiramente, é feita a modelagem do motor de corrente contínua de primeira ordem, utilizando os métodos de resposta ao degrau, dos mínimos quadrados, da área e do logaritmo neperiano para encontrar a função de transferência do motor. Em seguida, é implementado um controle proporcional e integral (PI) com código em Arduino acoplado a uma ponte H. Por fim, é feita a validação, verificando se de fato o controle sintonizado está rejeitando perturbações externas.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

## A LABORATORY PROPOSAL FOR CONTROLLING THE SPEED OF A DC MOTOR USING ARDUINO

Fernanda Folly Ferreira

March/2019

Advisors: Públio Macedo Lima

Wesley Rodrigues Silveira

Course: Automation and Control Engineering

In this work, we present the design of a PI controller to control the speed of a DC motor by using Arduino and H bridge to replace the use of power amplifiers in the experiments of the discipline Control Laboratory I of the under graduation course in Electrical Engineering.

First, the first-order model of direct current motor is obtained using the step response, least squares, area and neperian logarithm methods to find the motor transfer function. Then, a Proportional Integral controller (PI) with Arduino code coupled to the H bridge is implemented. Finally, the control code validation is carried out by verifying if external disturbances are rejected.

# Sumário

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introdução</b>                                 | <b>1</b>  |
| 1.1      | Motivação . . . . .                               | 1         |
| 1.2      | Objetivo . . . . .                                | 1         |
| 1.3      | Estrutura . . . . .                               | 2         |
| <b>2</b> | <b>Fundamentação Teórica</b>                      | <b>3</b>  |
| 2.1      | Modelagem do motor CC . . . . .                   | 3         |
| 2.2      | Arduino . . . . .                                 | 7         |
| 2.2.1    | Sinais digitais, analógicos e PWM . . . . .       | 8         |
| 2.3      | Ponte H . . . . .                                 | 11        |
| 2.4      | Amplificador Operacional . . . . .                | 15        |
| 2.5      | Determinação dos parâmetros . . . . .             | 18        |
| 2.5.1    | Mínimos quadrados . . . . .                       | 19        |
| 2.5.2    | Resposta ao degrau . . . . .                      | 20        |
| 2.6      | Controle PI . . . . .                             | 22        |
| 2.6.1    | Ação proporcional . . . . .                       | 23        |
| 2.6.2    | Ação integral . . . . .                           | 23        |
| 2.6.3    | Ação proporcional e integral . . . . .            | 23        |
| 2.6.4    | Especificações no domínio do tempo . . . . .      | 24        |
| <b>3</b> | <b>Método Proposto</b>                            | <b>26</b> |
| 3.1      | Material utilizado . . . . .                      | 26        |
| 3.2      | Montagem . . . . .                                | 26        |
| 3.3      | Aquisição de dados . . . . .                      | 29        |
| <b>4</b> | <b>Resultados e Discussões</b>                    | <b>31</b> |
| 4.1      | Determinação dos parâmetros do motor CC . . . . . | 31        |
| 4.1.1    | Simplificações . . . . .                          | 35        |
| 4.1.2    | Cálculo dos parâmetros . . . . .                  | 37        |
| 4.1.3    | Simulação com os parâmetros encontrados . . . . . | 41        |
| 4.2      | Sintonização do controlador PI . . . . .          | 44        |
| 4.2.1    | Cálculo das variáveis do controle . . . . .       | 44        |

|          |  |           |
|----------|--|-----------|
| 4.2.2    | Implementação do controle PI . . . . .                           | 46        |
| 4.2.3    | Validação do controle . . . . .                                  | 47        |
| <b>5</b> | <b>Conclusões</b>  | <b>53</b> |
|          | <b>Referências Bibliográficas</b>                                | <b>54</b> |
| <b>A</b> | <b>Códigos Utilizados</b>  | <b>55</b> |
| A.1      | Determinação dos parâmetros do motor . . . . .                   | 55        |
| A.1.1    | Código utilizado apenas para descobrir a região linear. . . . .  | 55        |
| A.1.2    | Código utilizado para determinar as constantes do motor. . . . . | 56        |
| A.2      | Controlador PI . . . . .   | 59        |



# Lista de Figuras

|             |  |    |
|-------------|--|----|
| Figura 2.1  | Circuito equivalente de um motor CC. . . . .   | 3  |
| Figura 2.2  | Circuito do Motor CC com tacômetro. . . . .  | 6  |
| Figura 2.3  | Diagrama de blocos do motor CC e tacômetro. . . . .  | 6  |
| Figura 2.4  | Microcontrolador Arduino Mega2560. . . . .   | 7  |
| Figura 2.5  | Exemplos de <i>duty cycle</i> . . . . .  | 9  |
| Figura 2.6  | Esquema do divisor de tensão. . . . .  | 10 |
| Figura 2.7  | Diagrama de blocos do motor CC, tacômetro, divisor de tensão e Arduino. . . . .              | 11 |
| Figura 2.8  | Modelo esquemático da ponte H. . . . .   | 12 |
| Figura 2.9  | Ponte H L298N. . . . .   | 13 |
| Figura 2.10 | Diagrama de blocos do motor CC, tacômetro, divisor de tensão, Arduino e ponte H. . . . .     | 14 |
| Figura 2.11 | Diagrama de blocos do sistema redesenhado. . . . .   | 14 |
| Figura 2.12 | Amplificador operacional. . . . .  | 15 |
| Figura 2.13 | Representação esquemática de um amplificador operacional LF356N. . . . .                     | 16 |
| Figura 2.14 | Amplificadores operacionais. . . . .   | 16 |
| Figura 2.15 | Circuito sem Arduino. . . . .  | 17 |
| Figura 2.16 | Circuito com Arduino e sem seguidor de tensão. . . . .                                       | 18 |
| Figura 2.17 | Circuito com Arduino e com seguidor de tensão. . . . .                                       | 18 |
| Figura 2.18 | Gráfico da resposta ao degrau. . . . .   | 20 |
| Figura 2.19 | Definição do percentual de <i>overshoot</i> e do tempo de acomodação. . . . .                | 25 |
| Figura 3.1  | Esquema do divisor de tensão. . . . .  | 27 |
| Figura 3.2  | Demonstrativo de como montar o divisor de tensão ligado ao amplificador operacional. . . . . | 27 |
| Figura 3.3  | Esquemático da montagem do circuito. . . . .   | 28 |
| Figura 3.4  | Esquema da montagem utilizando a protoboard. . . . .   | 29 |
| Figura 3.5  | Botão para o monitor serial do Arduino. . . . .  | 29 |
| Figura 3.6  | Opções que devem ser escolhidas para exportar os dados para o MATLAB. . . . .                | 30 |
| Figura 4.1  | Não linearidade da ponte H, desconsiderando a zona morta. . . . .                            | 33 |

|             |  |    |
|-------------|--|----|
| Figura 4.2  | Ajuste dos dados da ponte H por uma função polinomial de terceiro grau. . . . .                      | 34 |
| Figura 4.3  | Ajuste dos dados da ponte H por uma função polinomial de sexto grau. . . . .                         | 35 |
| Figura 4.4  | Diagrama de blocos do sistema. . . . .   | 36 |
| Figura 4.5  | Diagrama de blocos do sistema simplificado. . . . .  | 36 |
| Figura 4.6  | Diagrama de blocos do sistema para determinação dos parâmetros. . . . .                              | 37 |
| Figura 4.7  | Região linear de operação do motor. . . . .  | 38 |
| Figura 4.8  | Validação das constantes $K_a$ , $K_t$ e $K$ . . . . .   | 40 |
| Figura 4.9  | Modelo utilizado no Simulink para comparação dos valores simulados e experimentais. . . . .          | 41 |
| Figura 4.10 | Comparação das saídas simuladas do sistema com a saída do Arduino. . . . .                           | 42 |
| Figura 4.11 | Comparação das saídas simuladas do sistema com a saída do Arduino reajusta para a origem. . . . .    | 43 |
| Figura 4.12 | Diagrama de blocos do controle com a planta. . . . .   | 44 |
| Figura 4.13 | Diagrama de blocos do sistema simplificado. . . . .  | 46 |
| Figura 4.14 | Diagrama de blocos com a realimentação, controle e compensações. . . . .                             | 46 |
| Figura 4.15 | Modelo utilizado no Simulink para validação dos valores simulados. . . . .                           | 47 |
| Figura 4.16 | Gráfico da simulação no Simulink. . . . .  | 48 |
| Figura 4.17 | Diagrama de blocos do controle com a planta e com a constante $K_{ff}$ . . . . .                     | 49 |
| Figura 4.18 | Modelo utilizado no Simulink para simulação do controle com a constante $K_{ff}$ adicionada. . . . . | 50 |
| Figura 4.19 | Gráfico da simulação com a constante $K_{ff}$ adicionada. . . . .                                    | 50 |
| Figura 4.20 | Comparação da saída simulada com a saída experimental utilizando o controle PI. . . . .              | 51 |
| Figura 4.21 | Gráfico da tensão de saída do tacômetro em que há perturbações externas do tipo degrau. . . . .      | 52 |

# Lista de Tabelas

|            |  |    |
|------------|--|----|
| Tabela 2.1 | Especificações do Arduino Mega2560. . . . .  | 7  |
| Tabela 2.2 | Especificações da ponte H. . . . .   | 12 |
| Tabela 4.1 | Parâmetros do motor CC, utilizando alimentação de 20 V. . .  | 32 |
| Tabela 4.2 | Valores de $K$ e $\tau$ obtidos pelos métodos da resposta ao degrau,<br>da área e logaritmo neperiano a partir de diferentes amplitudes de<br>degraus. . . . . | 39 |
| Tabela 4.3 | Média e desvio padrão das constantes $K$ e $\tau$ calculadas pelos<br>métodos da resposta ao degrau, da área e do logaritmo neperiano. . .                     | 39 |
| Tabela 4.4 | Valores das constantes $K$ e $\tau$ obtidas pelos métodos utilizados<br>e suas médias. . . . .   | 41 |

# Capítulo 1

## Introdução

### 1.1 Motivação

A disciplina Laboratório de Controle I do curso de graduação da Engenharia Elétrica utiliza, para fins de experimentos, amplificadores de potência, que possuem um custo muito elevado, por volta de U\$ 3000,00 a U\$ 5000,00, dependendo do modelo. Visando substituir esses equipamentos, foi feito um estudo para viabilizar a troca dos amplificadores de potência por componentes de custo mais baixo. Uma das soluções é substituí-los por um conjunto de Arduino e ponte H, em que todos os experimentos possam ser feitos sem a necessidade dos amplificadores. Uma vez que os circuitos utilizados no laboratório são circuitos analógicos, a substituição dos mesmos pelo Arduino também contribui para um laboratório mais moderno.

A escolha pelo uso do Arduino também deve-se ao fato de ser um microcontrolador muito versátil e possuir um extenso suporte online nas comunidades de usuários de Arduino, sendo assim, bem didático para ser aprendido pelos alunos de graduação na disciplina Laboratório de Controle I. Por ser uma plataforma open-source, o Arduino possui um baixo custo, por volta de R\$ 150,00 cada kit.

### 1.2 Objetivo

Este trabalho tem como objetivo fazer com que todos os experimentos feitos na disciplina de Laboratório de Controle I [1], que utilizavam o amplificador de potência, possam ser feitos utilizando somente um Arduino e uma ponte H. O objetivo da disciplina é controlar um motor de corrente contínua (CC), dado que a entrada é um sinal do tipo degrau de referência, modelar o motor e sintonizar o controlador.

O controle utilizado será um controle proporcional - integral (PI). Assim, se houver qualquer perturbação externa no motor, o controle por si só conseguirá compensar e continuar seguindo a referência determinada.

## **1.3 Estrutura**

Este trabalho está organizado da seguinte maneira: No Capítulo 2, é apresentada a base teórica para o entendimento do trabalho, abordando desde os componentes utilizados na montagem, passando pelos métodos de identificação de variáveis, até chegar ao controle que foi implementado.

No Capítulo 3, está todo o método utilizado para execução do trabalho, os materiais que foram utilizados e como foi realizada a montagem do circuito.

No Capítulo 4, é apresentada a determinação dos parâmetros do motor, a sintonia do controle, bem como a validação dos resultados obtidos no trabalho.

No Capítulo 5, são apresentadas as conclusões gerais.

# Capítulo 2

## Fundamentação Teórica

Neste capítulo será apresentada a base teórica para melhor entendimento do trabalho. Começando pelos componentes utilizados na montagem do projeto, passando pelos métodos de determinação das variáveis, e por fim, pelo controle implementado.

### 2.1 Modelagem do motor CC

O motor de corrente contínua é um motor de fácil modelagem e pode ter sua velocidade controlada por mudanças nos níveis da tensão de entrada. Nele, o torque magnético atua sobre um condutor que transporta corrente e a energia é convertida em energia mecânica.

Na figura 2.1, observa-se o modelo do circuito de um motor CC controlado pela armadura, em que  $i_a(t)$  é a corrente de armadura,  $v_a(t)$  é a tensão de armadura,  $\dot{\theta}(t)$  é a velocidade angular do motor,  $J$  é o momento de inércia da carga,  $f$  é o coeficiente de atrito viscoso nos mancais,  $e(t)$  é a força eletromotriz,  $R_a$  é a resistência de armadura e  $L_a$  o indutor, que representam os efeitos do enrolamento de armadura.

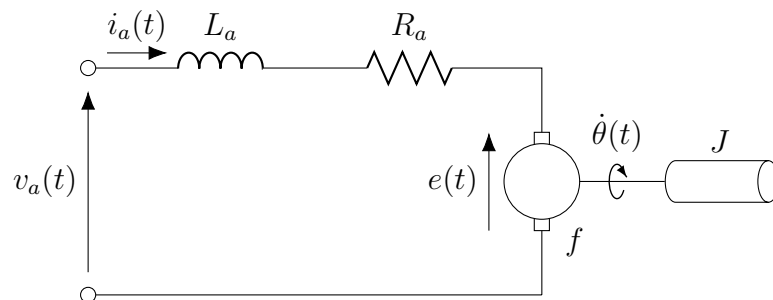


Figura 2.1: Circuito equivalente de um motor CC.

Segundo KUO [2], NISE e DA SILVA [3] e FRANKLIN *et al.* [4], observa-se que o torque produzido pelo motor,  $t_m(t)$ , é proporcional à corrente de armadura,  $i_a(t)$ , isto é:

$$t_m(t) = K_m i_a(t), \quad (2.1)$$

em que  $K_m$  é uma constante de proporcionalidade, chamada de constante de torque do motor.

Aplicando a lei de Kirchhoff ao circuito da armadura, tem-se a seguinte equação:

$$v_a(t) = R_a i_a(t) + L_a \frac{d}{dt} i_a(t) + e(t), \quad (2.2)$$

sendo  $e(t)$  a força eletromotriz, que é proporcional à velocidade angular do motor, podendo ser escrita como:

$$e(t) = K_e \dot{\theta}(t), \quad (2.3)$$

em que  $K_e$  é uma constante de proporcionalidade, chamada de constante de força eletromotriz.

Aplicando a lei de Newton para o movimento rotacional do motor, tem-se:

$$t_m(t) - f\dot{\theta}(t) = J\ddot{\theta}(t), \quad (2.4)$$

Aplicando a transformada de Laplace em ambos os lados das equações 2.1 à 2.4, resulta-se em:

$$\begin{cases} T_m(s) = K_m I_a(s), & (2.5) \end{cases}$$

$$\begin{cases} V_a(s) = R_a I_a(s) + L_a s I_a(s) + E(s), & (2.6) \end{cases}$$

$$\begin{cases} E(s) = K_e s \Theta(s), & (2.7) \end{cases}$$

$$\begin{cases} T_m(s) - s f \Theta(s) = s^2 J \Theta(s). & (2.8) \end{cases}$$

Fazendo as seguintes substituições utilizando as equações 2.5 à 2.8, é possível obter a seguinte função de transferência:

$$\Theta(s) = \frac{E(s)}{K_e s} \implies \Theta(s) = \frac{V_a(s) - (R_a + sL_a)I_a(s)}{K_e s};$$

$$\Theta(s) = \frac{V_a(s) - (R_a + sL_a)\left(\frac{T_m(s)}{K_m}\right)}{K_e s} \implies \Theta(s) = \frac{V_a(s) - (R_a + sL_a)\left(\frac{s f \Theta(s) + s^2 J \Theta(s)}{K_m}\right)}{K_e s};$$

$$\Theta(s) = \frac{K_m}{s[K_e K_m + (R_a + sL_a)(f + sJ)]} V_a(s). \quad (2.9)$$

Substituindo,

$$\begin{cases} \tau_e = \frac{L_a}{R_a}, \text{ em que } \tau_e \text{ é chamada de constante de tempo elétrica;} \\ \tau_m = \frac{J}{f}, \text{ em que } \tau_m \text{ é chamada de constante de tempo mecânica;} \end{cases}$$

na equação 2.9, tem-se:

$$\Theta(s) = \frac{K_m}{s[K_e K_m + fR_a(\tau_m s + 1)(\tau_e s + 1)]} V_a(s). \quad (2.10)$$

Considerando que a indutância de armadura é muito pequena, a constante de tempo elétrica pode ser desprezada frente à constante de tempo mecânica. Então, substituindo  $\tau_e s + 1 \approx 1$ , obtêm-se:

$$\Theta(s) = \frac{K_m}{s[K_e K_m + fR_a(\tau_m s + 1)]} V_a(s); \quad (2.11)$$

$$\Theta(s) = \frac{K_m}{s[K_e K_m + R_a(f + Js)]} V_a(s). \quad (2.12)$$

Substituindo,

$$\begin{cases} \tau = \frac{R_a J}{K_e K_m + f R_a}, \\ K_a = \frac{K_m}{K_e K_m + f R_a}, \end{cases}$$

na equação 2.12, encontra-se:

$$\Theta(s) = \frac{K_a}{s(\tau s + 1)} V_a(s). \quad (2.13)$$

No caso deste trabalho, a função de transferência desejada é entre a entrada do motor,  $V_a(s)$ , e a saída ( $\omega = \dot{\theta}$ ). Logo, a função de transferência, é expressa por:

$$\frac{W(s)}{V_a(s)} = s \frac{\Theta(s)}{V_a(s)}. \quad (2.14)$$

Encontrando, por fim, o seguinte sistema de primeira ordem:

$$\frac{W(s)}{V_a(s)} = \frac{K_a}{\tau s + 1}, \quad (2.15)$$

em que  $K_a$  é o ganho CC do motor e  $\tau$  é a constante de tempo.



Para medir a velocidade angular do motor é necessário o uso de um tacômetro, conforme mostrado na figura 2.2. O tacômetro é um gerador CC de baixa potência, cuja tensão gerada é proporcional à velocidade do eixo. Logo, a tensão nos terminais do tacômetro,  $v_t(t)$ , pode ser escrita como:

$$v_t(t) = K_t \omega(t), \quad (2.16)$$

em que  $K_t$  é a constante de ganho do tacômetro.

A relação entre  $V_t(s)$  e  $V_a(s)$ , é expressa por:

$$\frac{V_t(s)}{V_a(s)} = \frac{K_a K_t}{\tau s + 1}. \quad (2.17)$$

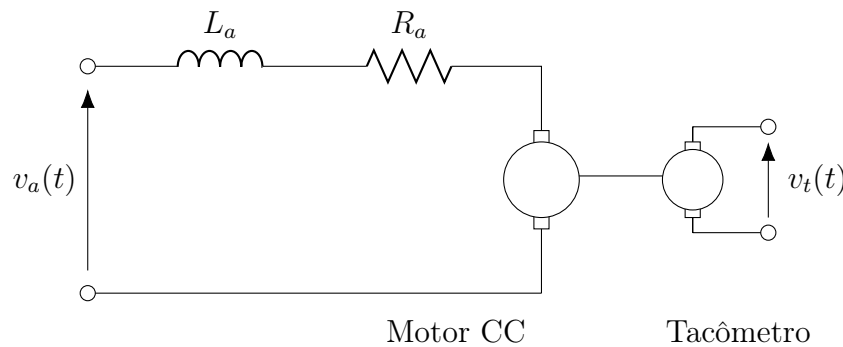


Figura 2.2: Circuito do Motor CC com tacômetro.

Na figura 2.3 é apresentado o diagrama de blocos do motor e do tacômetro.

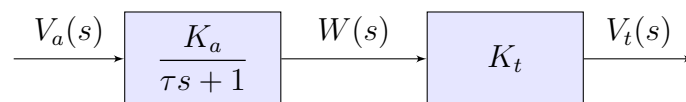


Figura 2.3: Diagrama de blocos do motor CC e tacômetro.

É importante ressaltar que o modelo matemático do motor CC dado pela equação 2.17, considera a planta linear. Entretanto, isso não é sempre verdade para todos os valores de  $v_a(t)$ , pois existem os efeitos da zona morta e da saturação do motor; a zona morta é quando valores muito baixos de tensão de entrada não são capazes de vencer o atrito nos mancais do motor; já a saturação é quando, a partir de determinada tensão, diferentes valores na tensão de entrada não alteram mais a velocidade do motor. Portanto, é importante obter a faixa de valores para qual o sistema é linear, para que todos os cálculos para determinação das constantes do motor sejam feitos dentro dessa região.

## 2.2 Arduino

Segundo seu site oficial [5], o Arduino é uma plataforma eletrônica *open-source* baseada em um *hardware* e *software* fáceis de serem utilizados. As placas de Arduino são capazes de ler entradas (*inputs*) e gerar saídas (*outputs*), para isso, deve ser informado o que deve ser feito, enviando um conjunto de instruções para o microcontrolador. Para isso, utiliza-se a linguagem de programação Arduino, muito similar à linguagem C++, e o *software* Arduino (IDE).

O Arduino utilizado neste trabalho é o Arduino Mega2560, mostrado na figura 2.4. Outros modelos de Arduinos, como o Arduino UNO, por exemplo, também poderiam ser utilizados.

Na tabela 2.1, são mostrados as principais especificações do Arduino Mega2560.

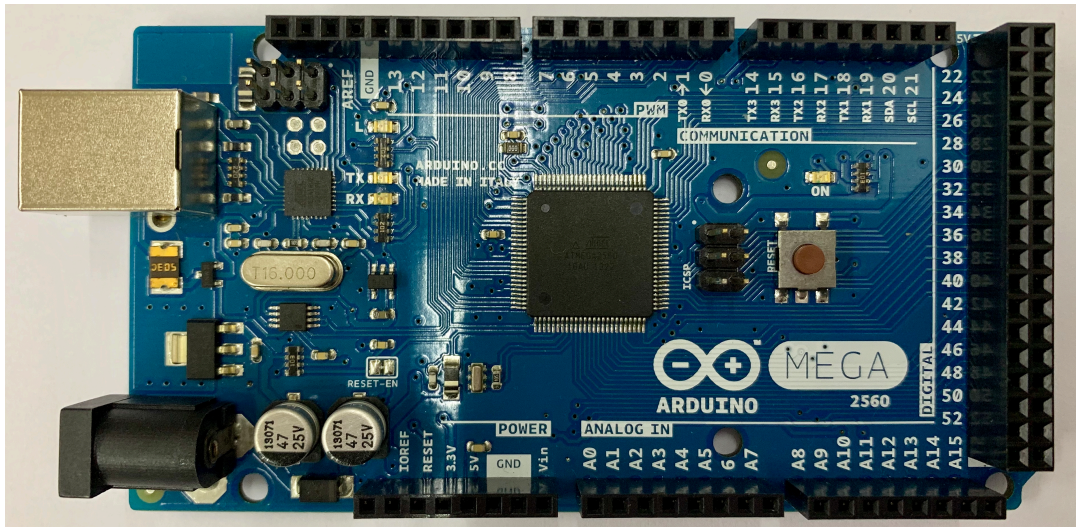


Figura 2.4: Microcontrolador Arduino Mega2560.

Tabela 2.1: Especificações do Arduino Mega2560.

| Especificações              |                              |
|-----------------------------|------------------------------|
| Tensão de operação          | 5 V                          |
| Corrente de operação máxima | 50 mA                        |
| Pinos analógicos            | 16                           |
| Pinos digitais              | 54, sendo 14 para saídas PWM |
| Memória flash               | 256 KB                       |
| Dimensões                   | (101,52 x 53,3) mm           |
| Peso                        | 37 g                         |

A programação do Arduino é feita em basicamente três etapas, a declaração de variáveis, o *setup* e o *loop*:

(i) Na primeira etapa, declaram-se todas as variáveis e bibliotecas que serão utilizadas ao longo do programa, e também, é o local onde são definidas todas as funções.

(ii) O *Setup* é a segunda etapa a ser chamada quando executa-se o programa, e é utilizada para determinar quais variáveis são de entradas e quais são de saídas. Essa etapa é executada uma única vez, no início do programa.

(iii) O *Loop* é a última etapa e inicia-se logo após o *setup*, rodando repetidamente até o término ou pausa do programa.

## 2.2.1 Sinais digitais, analógicos e PWM

Segundo BLUM [6], existem dois grupos principais de sinais que o Arduino pode ler e escrever: sinais digitais (+5 V ou 0 V) e sinais analógicos (tensões entre 0 V e +5 V).

Um sinal digital no Arduino pode estar apenas em dois estados, nível lógico **alto**, quando o pino está em 5 V, ou nível lógico **baixo**, quando o pino está em 0 V ou GND. Existem 54 portas digitais no Arduino Mega2560 (pinos 0 - 53) e cada uma delas pode ser configurada como entradas ou saídas.

Já os sinais analógicos no Arduino podem ser quaisquer valores entre 0 V e +5 V. Existem 16 portas analógicas presentes no Arduino Mega2560 (pinos A0 - A15). Como no Arduino tudo é processado de forma digital, é necessário converter os sinais analógicos em digitais. Essa conversão é feita diretamente na placa, com conversores já embutidos nela. Tais conversores possuem uma resolução de 10 bits para um intervalo de tensão entre 0 V e 5 V. Sendo assim, se for aplicado 0 V à entrada, será visto um valor analógico de 0; ao passo que se for aplicado um valor de 5 V à entrada, será visto um valor analógico de 1023; e qualquer valor intermediário, será proporcional à entrada. Por exemplo, 2,93 V aplicados à entrada, resultará em um valor analógico de 600, pois segue a regra de proporção:

$$\begin{array}{rcl} 5 \text{ V} & \text{---} & 1023 \\ 2,93 \text{ V} & \text{---} & x. \end{array}$$

Sendo assim, tem-se que  $x = 600$ .

Apesar de haver perda de informação na transformação de 0 - 5 V para 0 - 1023 bits, para este trabalho essa conversão não influenciará, pois o erro estaria apenas na terceira casa decimal, uma vez que a leitura é  $5 \text{ V}/1024 = 4.9 \text{ mV}$  por unidade de medida, sendo praticamente desprezível.

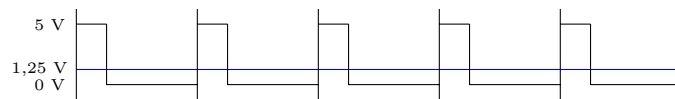
Já o sinal chamado de modulação de largura de pulso (*Pulse Width Modulation* - PWM), que não é uma saída analógica, mas sim o equivalente digital a uma saída analógica, consiste na geração de uma onda quadrada, na qual, controla-se a porcentagem de tempo em que a onda permanece em nível lógico alto. Essa porcentagem é chamada de ciclo de trabalho (*duty cycle*), que segundo WARREN *et al.* [7], é definida como a razão entre o tempo em que o sinal permanece na tensão máxima (5 V) sobre o tempo total de oscilação, como apresentado na equação 2.18. Sua alteração provoca uma mudança no valor médio da onda, como pode ser visto na figura 2.5, em que a linha azul é referente ao valor médio da onda. Das 54 portas digitais presentes no Arduino Mega2560, 14 são destinadas ao uso do PWM.

Diferentemente do sinal analógico, o PWM possui um registrador de 8 bits, sendo assim, seu valor vai de 0 até 255, em que 0 representa 0% de *duty cycle* e 255 representa 100%. Então, se for desejado um valor médio de 0 V na saída, acarretará em um *duty cycle* de 0%, como visto na figura 2.5a e se for desejado 5 V na saída, o *duty cycle* será 100%, e deve-se implementar no código o valor de 255 referente aos 100% de *duty cycle*, como visto na figura 2.5d. Mas se forem aplicados valores intermediários na saída, o *duty cycle* será proporcional, como mostrado nas figuras 2.5b e 2.5c. Sempre seguindo a regra de proporção:

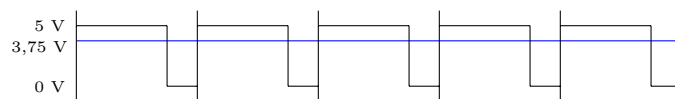
| Duty cycle | — | Tensão | — | Valor analógico |
|------------|---|--------|---|-----------------|
| 0%         | — | 0 V    | — | 0               |
| 100%       | — | 5 V    | — | 255             |



(a) 0% *duty cycle*.



(b) 25% *duty cycle*.



(c) 75% *duty cycle*.



(d) 100% *duty cycle*.

Figura 2.5: Exemplos de *duty cycle*.

O *duty cycle* pode ser calculado pela seguinte equação:

$$DutyCycle(\%) = \frac{\text{Tempo em nível lógico alto}}{\text{Tempo total de oscilação}} \times 100\%. \quad (2.18)$$

Em relação à entrada do Arduino, é importante ressaltar que o Arduino suporta uma tensão de entrada menor ou igual à 5 V, por isso um divisor de tensão é necessário à entrada do Arduino quando tensões maiores forem utilizadas. O divisor consiste em dois resistores em série ( $R_1$  e  $R_2$ ) com o intuito de não deixar o Arduino receber uma tensão maior do que 5 V, para não acarretar possíveis danos ao mesmo. O cálculo do divisor de tensão utilizado foi feito pensando em uma tensão de até 20 V. Uma vez que a tensão de entrada do motor para os experimentos é de 0 a 20 V, e por consequência o tacômetro fornece uma tensão  $V_t(s)$  de até 20 V. Sendo assim, tem-se que a tensão de saída do divisor de tensão precisa ser  $\frac{1}{4}$  da entrada, para ter certeza que não chegará mais de 5 V ao Arduino.

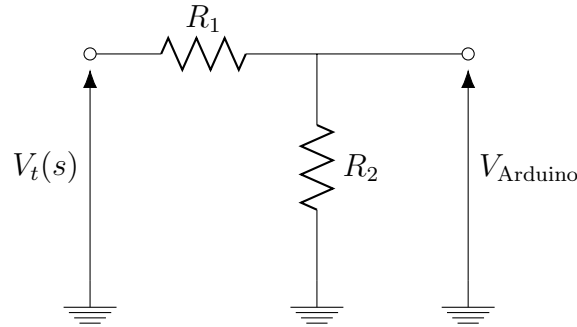


Figura 2.6: Esquema do divisor de tensão.

De acordo com a figura 2.6, tem-se que:

$$V_{\text{Arduino}} = \frac{R_2}{R_1 + R_2} V_t(s), \quad (2.19)$$

e escolhendo que a tensão de entrada no Arduino seja  $\frac{1}{4}$  da tensão de saída do tacômetro, encontra-se:

$$V_{\text{Arduino}} = \frac{V_t(s)}{4}. \quad (2.20)$$

Substituindo a equação 2.20 na equação 2.19, obtêm-se:

$$4R_2 V_t(s) = V_t(s)(R_1 + R_2), \quad (2.21)$$

$$3R_2 = R_1. \quad (2.22)$$

Sendo assim, para que a tensão de entrada do Arduino não seja maior do que 5 V, é preciso que as resistências do divisor de tensão sejam uma três vezes maior que a outra, como visto na equação 2.22.

Com o uso do divisor de tensão e do Arduino acoplados ao motor CC e ao tacômetro, tem-se agora um novo diagrama de blocos do circuito, visto na figura 2.7. A entrada do Arduino será um valor em bits de 0 - 1023, pois usa-se uma porta analógica para ler a tensão de saída do tacômetro, e a saída será um sinal PWM de 0 a 5 V, baseando-se no *duty cycle* que vai de 0 à 255.

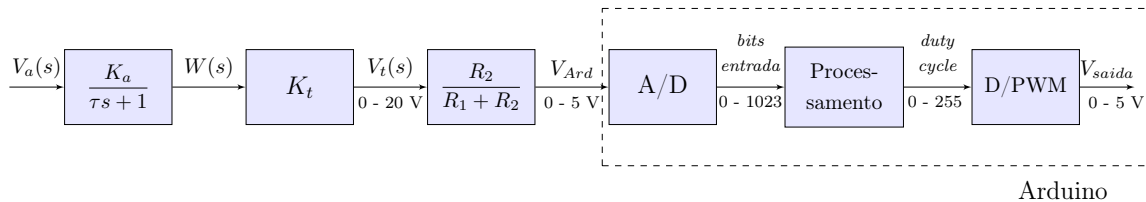


Figura 2.7: Diagrama de blocos do motor CC, tacômetro, divisor de tensão e Arduino.

Para fazer a conversão da tensão de entrada para bits, deve-se apenas usar a regra de três vista anteriormente:

$$\begin{array}{rcl} 5 \text{ V} & \text{---} & 1023 \\ x & \text{---} & \text{Entrada do Arduino.} \end{array}$$

Enquanto que para fazer a conversão do *duty cycle* para a saída PWM, basta utilizar a regra de três a seguir:

$$\begin{array}{rcl} 5 \text{ V} & \text{---} & 255 \\ x & \text{---} & \text{Saída PWM do Arduino.} \end{array}$$

## 2.3 Ponte H

Para o funcionamento do motor CC utilizado neste trabalho, a demanda de corrente é superior à que as portas do Arduino conseguem fornecer, sendo assim, não se deve ligar o motor diretamente às portas do Arduino, pois assim que o motor demandar uma corrente acima de 50 mA (máxima corrente fornecida pela placa), acarretará danos às portas do Arduino.

Para solucionar esse problema, devido à correntes superiores à capacidade do Arduino, poderia ser utilizado transistores, porém não é possível controlar o sentido de giro do motor, caso fosse necessário. Para inverter o sentido de giro deve-se inverter a polaridade da alimentação do motor. O transistor só seria suficiente para ligar e desligar o motor.

Neste trabalho, é utilizado o circuito conhecido como ponte H, mostrado na figura 2.8, que nada mais é que um arranjo de quatro transistores. Este circuito é uma boa solução por ser capaz de acionar o motor, controlando não apenas seu sentido de rotação, como também sua velocidade.

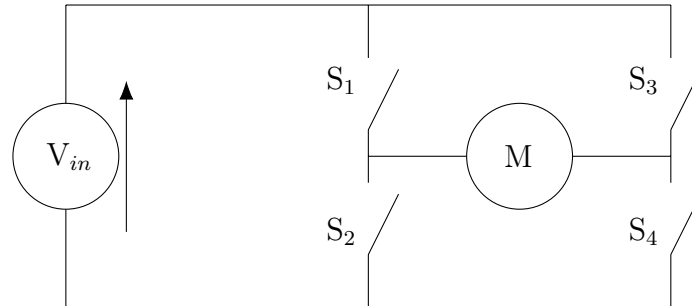


Figura 2.8: Modelo esquemático da ponte H.

De acordo com o esquema apresentado na figura 2.8, as chaves da ponte H fecham em pares na diagonal. Então, se a chave  $S_1$  fecha,  $S_4$  também fechará junto e  $S_3$  e  $S_2$  ficarão abertas, fazendo com que a corrente passe por  $S_1$  e  $S_4$ , girando o motor em um sentido. Se  $S_3$  e  $S_2$  fecharem e  $S_1$  e  $S_4$  abrirem, a corrente passará por aquelas chaves, fazendo o motor girar no outro sentido.

As especificações referente ao circuito da ponte H estão apresentadas na tabela 2.2 e o circuito físico utilizado neste trabalho foi o L298N, visto na figura 2.9. Este circuito é capaz de alimentar dois motores simultaneamente, mas neste trabalho, será utilizado apenas um motor.

Tabela 2.2: Especificações da ponte H.

| <b>Especificações</b>       |                             |
|-----------------------------|-----------------------------|
| Tensão de operação          | 6 V ~ 35 V                  |
| Corrente de operação máxima | 2 A por canal ou 4 A máxima |
| Tensão lógica               | 5 V                         |
| Limites de temperatura      | -20 °C a +135 °C            |
| Potência máxima             | 25 W                        |
| Dimensões                   | (43 x 43 x 27) mm           |
| Peso                        | 30 g                        |

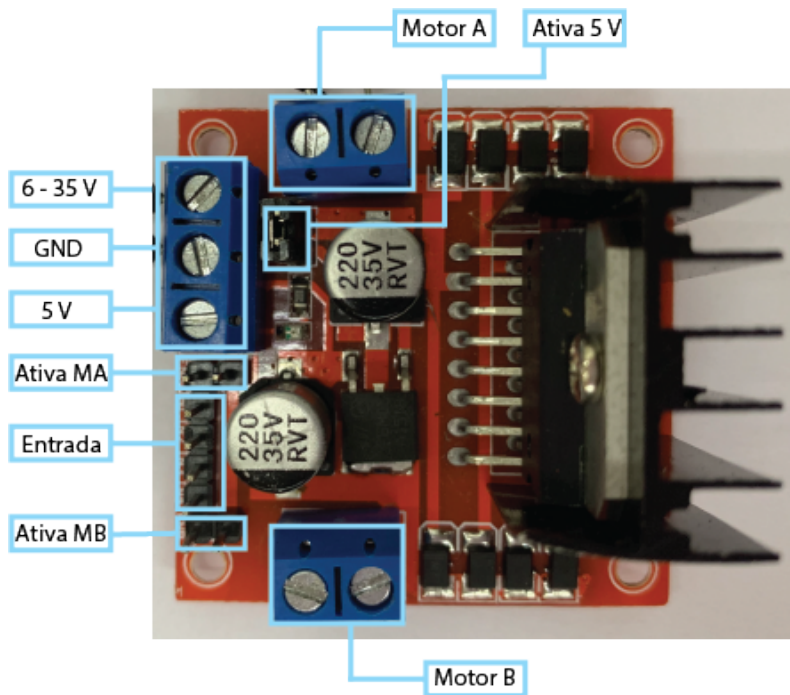


Figura 2.9: Ponte H L298N.

De acordo com a figura 2.9, a ponte H possui os seguintes conectores:

- 6 - 35 V: Alimentação da placa com uma tensão de entrada entre 6 e 35 V; para este trabalho foi utilizada uma alimentação de 20 V;
- Motor A: Entradas do motor A.
- Motor B: Entradas do motor B.
- Ativa MA: Responsável pelo controle PWM do motor A e quando jumpeado aciona o motor A com velocidade máxima. Para controlar a velocidade do motor A basta remover o jumper e alimentar o pino com uma tensão entre 0 à 5 V;
- Ativa MB: Responsável pelo controle PWM do motor B e quando jumpeado aciona o motor B com velocidade máxima. Para controlar a velocidade do motor B basta remover o jumper e alimentar o pino com uma tensão entre 0 à 5 V;
- 5 V: Em casos de não haver fonte de alimentação com mais de 6 V pode-se alimentar a placa com 5 V por esta porta;
- Ativa 5 V: Quando jumpeado, a placa utilizará o regulador de tensão integrado para fornecer 5 V (na porta 5 V) quando a porta 6-35 V estiver sendo alimentada por uma tensão entre 6 a 35 V. Neste caso, não se deve alimentar a porta 5 V pois pode danificar os componentes;



- IN1 e IN2: Entradas utilizadas para controlar o sentido do motor A;
- IN3 e IN4: Entradas utilizadas para controlar o sentido do motor B.

Os pinos Motor B, Ativa MB, 5 V, Ativa 5 V, IN3 e IN4, não foram utilizados neste trabalho.

A ponte H, neste trabalho, é alimentada com uma fonte externa (20 V), diferentemente do Arduino, e recebe um sinal de entrada PWM entre 0 V e 5 V provenientes do Arduino na porta Ativa MA. Portanto, a saída da ponte H deveria ser proporcional à sua entrada e também a sua alimentação, porém, é observado que a ponte H possui um comportamento não linear. Como fazer a compensação dessa não linearidade, será apresentado nos capítulos a frente.

Com o uso da ponte H acoplado ao Arduino, motor CC, e ao tacômetro, tem-se agora o novo diagrama de blocos, visto na figura 2.10.

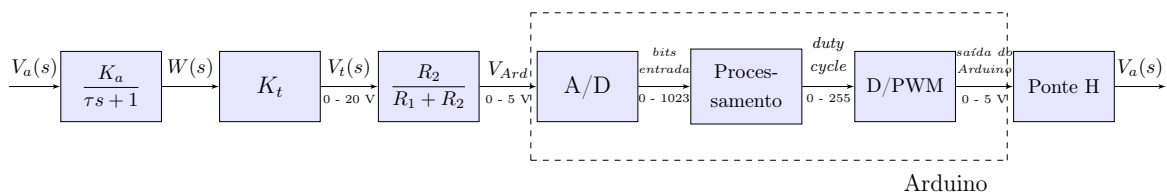


Figura 2.10: Diagrama de blocos do motor CC, tacômetro, divisor de tensão, Arduino e ponte H.

É possível redesenhar a figura 2.10 de modo a ressaltar a utilização do Arduino como o responsável por fornecer a tensão para o sistema, além de ler a tensão do tacômetro e fazer a realimentação do mesmo, quando for acrescido o controle. Tal reformulação pode ser vista na figura 2.11.

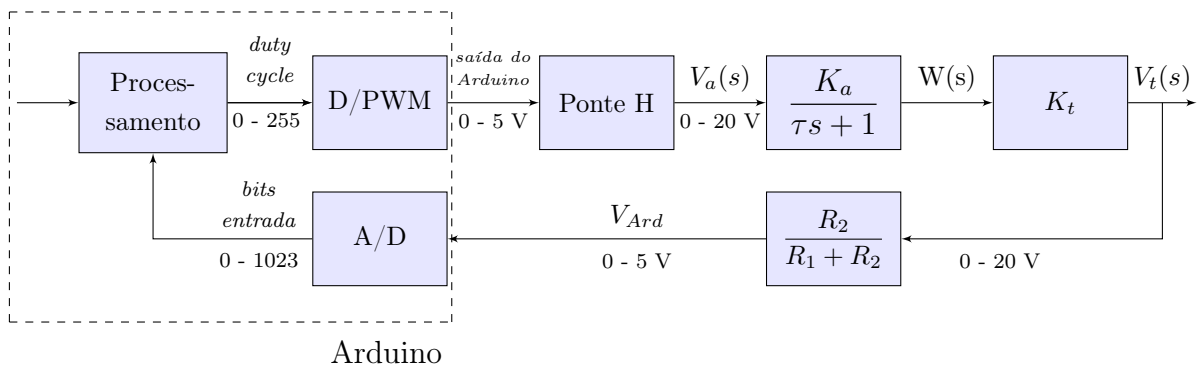


Figura 2.11: Diagrama de blocos do sistema redesenhado.

## 2.4 Amplificador Operacional

Amplificadores operacionais são dispositivos lineares que possuem ganho elevado e são geralmente utilizados para amplificar sinais que se estendem sobre uma ampla faixa de frequências. São muito utilizados em condicionamento de sinal, filtragem, ou para realizar operações matemáticas, como integração, diferenciação, adição e subtração.

Um amplificador operacional possui dois terminais de entrada e um terminal de saída, conforme apresentado na figura 2.12. A entrada  $(-)$  é denominada inversora, enquanto a entrada  $(+)$  é chamada de não-inversora. Isto faz com que esses amplificadores sejam conhecidos como diferenciais, uma vez que a tensão de saída é proporcional à diferença entre as tensões aplicadas nos seus terminais de entrada.

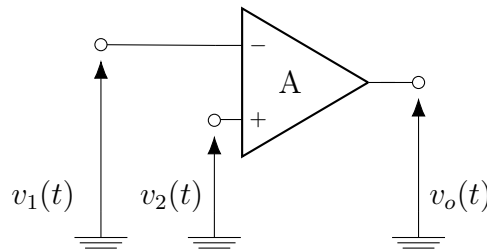


Figura 2.12: Amplificador operacional.

Segundo SEDRA e SMITH [8], utilizando a notação da figura 2.12, essa característica é matematicamente descrita pela equação 2.23.

$$v_0 = A(v_2 - v_1), \quad (2.23)$$

em que  $A$  é o ganho de malha aberta ( $A > 45000$ ), sendo definido pelos transistores internos do amplificador. Idealmente, este ganho é considerado infinito ( $A \rightarrow \infty$ ).

A representação do circuito integrado do amplificador operacional, pode ser visto na figura 2.13. Ele é composto por oito pinos e um chanfro na parte de cima, tal chanfro serve de referência para a numeração dos pinos. É possível observar as respectivas indicações das conexões que devem ser efetuadas em cada um de seus terminais. Nos amplificadores operacionais há a necessidade de tensões externas de polarização ( $+V_{cc}$  e  $-V_{cc}$ ), sendo que neste trabalho é utilizada uma tensão de  $\pm 20V$ .

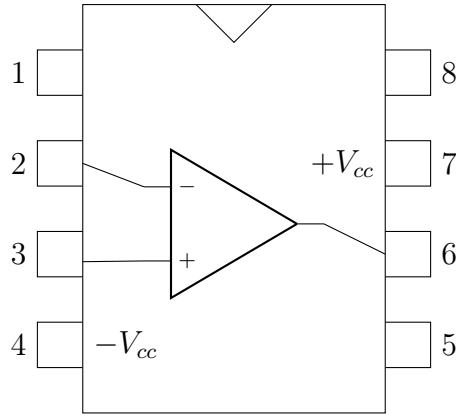


Figura 2.13: Representação esquemática de um amplificador operacional LF356N.

A configuração de amplificador operacional utilizada no trabalho foi a configuração não-inversora, apresentada na figura 2.14a. Nela, o sinal de entrada  $e_i(t)$  é aplicado diretamente ao terminal não inversor do amplificador, enquanto que o terminal inversor é conectado nos terminais de  $R_i$  e  $R_f$ , sendo o outro terminal de  $R_i$  conectado ao terra e o outro terminal de  $R_f$  conectado à saída.

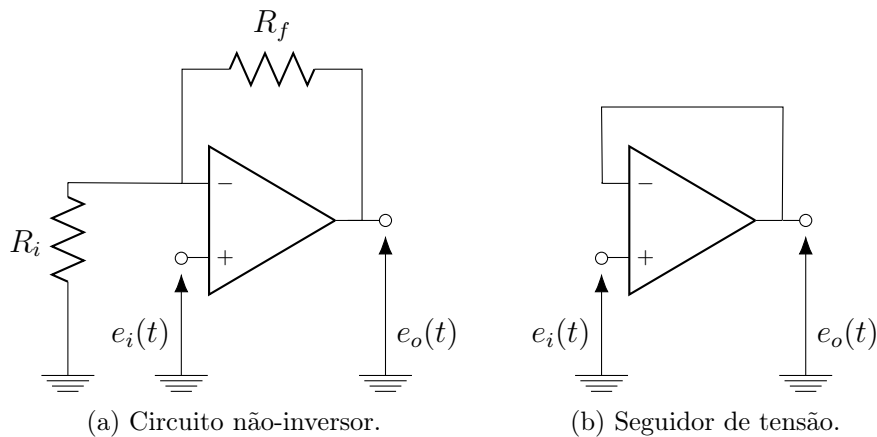


Figura 2.14: Amplificadores operacionais.

Como a corrente de entrada inversora do amplificador da figura 2.14a é nula, a parte composta por  $R_f$  e  $R_i$  atua como um divisor de tensão no caminho da realimentação negativa, fazendo com que uma fração da tensão de saída apareça no terminal de entrada inversora, apresentando o seguinte ganho:

$$e_i(t) = \frac{R_i}{R_i + R_f} e_o(t) \Rightarrow e_o(t) = \frac{R_f + R_i}{R_i} e_i(t). \quad (2.24)$$

Observa-se ainda que, fazendo  $R_f \rightarrow 0$  (curto-circuito) e  $R_i \rightarrow \infty$  (circuito-aberto), conforme mostrado na figura 2.14b, tem-se que o ganho do amplificador será igual a 1. Nesse caso, o amplificador atua como um **seguidor de tensão**, isto é, a tensão de saída segue exatamente a tensão de entrada.

A configuração de um seguidor de tensão é largamente utilizada para promover isolamento entre os sinais da fonte e da carga, evitando assim interações indesejáveis.

Neste trabalho, o amplificador operacional utilizado é o LF356N, como seguidor de tensão, no intuito de não deixar a impedância de entrada do Arduino interferir na tensão de saída do tacômetro pós divisor de tensão ( $V_{ard}$ , vide figura 2.11). Pois se o Arduino for conectado diretamente a saída do divisor de tensão, a impedância do Arduino acarretaria numa mudança na tensão de saída.

Na figura 2.15 é possível observar um esquema do circuito em que o Arduino ainda não foi acoplado, assim a tensão de saída é:

$$V_{saída} = \frac{R_2}{R_1 + R_2} V_{tacômetro}, \quad (2.25)$$

em que,  $R_1$  e  $R_2$  são resistências utilizadas somente para dividir a tensão de saída do tacômetro.

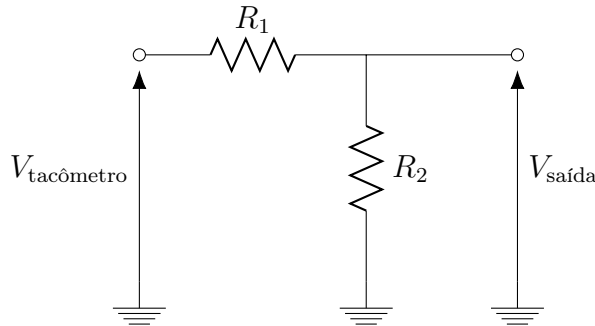


Figura 2.15: Circuito sem Arduino.

Quando acopla-se o Arduino ao circuito, como visto na figura 2.16, em que a impedância de entrada do Arduino é representada como  $R_{arduino}$ , a tensão de saída, isto é, a tensão que de fato será lida pela porta analógica do Arduino, não continuará a mesma. A nova tensão de saída será de:

$$V_{saída} = \frac{R_2 // R_{arduino}}{R_1 + R_2 // R_{arduino}} V_{tacômetro}. \quad (2.26)$$

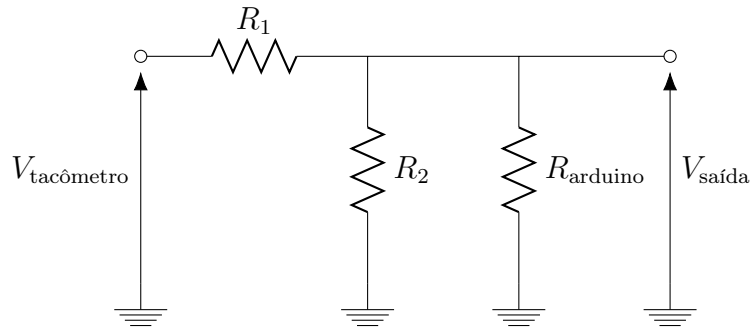


Figura 2.16: Circuito com Arduino e sem seguidor de tensão.

Para não deixar a impedância do Arduino interferir na tensão de saída, é utilizado o seguidor de tensão entre a saída do divisor de tensão e o Arduino, mostrado na figura 2.17. Assim, tem-se certeza que a impedância de entrada do Arduino não influenciará na saída do divisor de tensão, pois o amplificador pode ser considerado como tendo impedância de entrada infinita, logo na configuração seguidor de tensão, a saída segue exatamente a entrada, não deixando o Arduino influenciar na saída do circuito. Assim, a tensão de saída passa a ser novamente igual a equação 2.25.

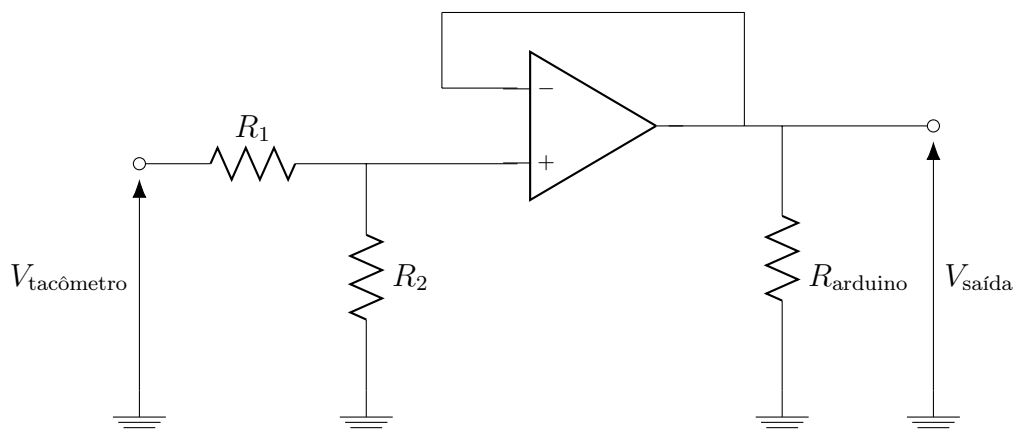


Figura 2.17: Circuito com Arduino e com seguidor de tensão.

Com o amplificador operacional sendo utilizado como seguidor de tensão, não há influências do amplificador com ganhos ao circuito. Sendo assim, o diagrama de blocos continua sendo o mesmo da figura 2.11.

## 2.5 Determinação dos parâmetros

A função de transferência utilizada neste trabalho para a determinação dos parâmetros considera a planta como o motor CC mais o tacômetro, como pode ser visto na equação 2.27, em que  $K = K_a K_t$ .

$$G(s) = \frac{V_t(s)}{V_a(s)} = \frac{K}{\tau s + 1}, \quad (2.27)$$

Os ganhos relativos ao Arduino, ponte H e divisor de tensão, serão compensados, e serão apresentados no capítulo 4.

### 2.5.1 Mínimos quadrados

Quando um sistema do tipo:

$$A\underline{x} = \underline{b}, \quad (2.28)$$

no qual  $A$  possui dimensões  $m \times n$ , sendo  $m > n$ , frequentemente não há solução, pois  $\underline{b}$ , muito possivelmente, não pertencerá ao espaço formado pelas colunas de  $A$ . Segundo STRANG [9], pode-se utilizar aproximação por mínimos quadrados para encontrar um  $\underline{x}$  que minimize o erro,  $\underline{e} = \underline{b} - A\underline{x}$ .

Multiplicando ambos os lados da equação 2.28 por  $A^T$ , encontra-se:

$$A^T A \underline{x} = A^T \underline{b}. \quad (2.29)$$

$$\underline{x} = (A^T A)^{-1} A^T \underline{b}. \quad (2.30)$$

Se as colunas de  $A$  forem linearmente independentes, logo  $A^T A$  é invertível e  $\underline{x}$  pode ser calculado, pois agora  $A^T \underline{b}$  faz parte do espaço gerado pelas colunas da matriz  $A^T A$ .

No caso deste trabalho, usa-se o método dos mínimos quadrados para funções lineares e com  $y(0) = 0$ , logo a reta que melhor ajusta os pontos é do tipo  $y = ax$ , sendo assim, o vetor  $\underline{x}$ , da equação 2.28, na verdade é um escalar. A matriz  $A$  será um vetor coluna, assim como o vetor  $\underline{b}$ , logo a equação 2.30 pode ser reescrita como:

$$x = \frac{A^T \underline{b}}{A^T A}. \quad (2.31)$$

Assim, a partir da equação 2.31, é possível encontrar os parâmetros do motor:

- $K_a$ : Quando a variável independente for a entrada em tensão do motor e a variável dependente for a saída do tacômetro em  $rad/s$ ,

$$K_a = \frac{v_a(t)^T * w(t)}{v_a(t)^T * v_a(t)}; \quad (2.32)$$

- $K_t$ : Quando a variável independente for a saída do tacômetro em  $rad/s$ , e a variável dependente for a saída do tacômetro em tensão,

$$K_t = \frac{w(t)^T * v_t(t)}{w(t)^T * w(t)}; \quad (2.33)$$

- $K$ : Quando a variável independente for a entrada em tensão do motor e a variável dependente for a saída do tacômetro em tensão. Também pode ser calculado multiplicando os dois outros parâmetros acima,

$$K = \frac{v_a(t)^T * v_t(t)}{v_a(t)^T * v_a(t)} \quad \text{ou} \quad K = K_a K_t. \quad (2.34)$$

Em que  $v_a(t)$ ,  $w(t)$  e  $v_t(t)$  são vetores coluna com os valores experimentais.

## 2.5.2 Resposta ao degrau

Aplica-se um degrau de amplitude  $A$  à função de transferência do motor, como visto na figura 2.18, obtendo a seguinte resposta:

$$y(t) = KA(1 - e^{-t/\tau}), t \geq 0. \quad (2.35)$$

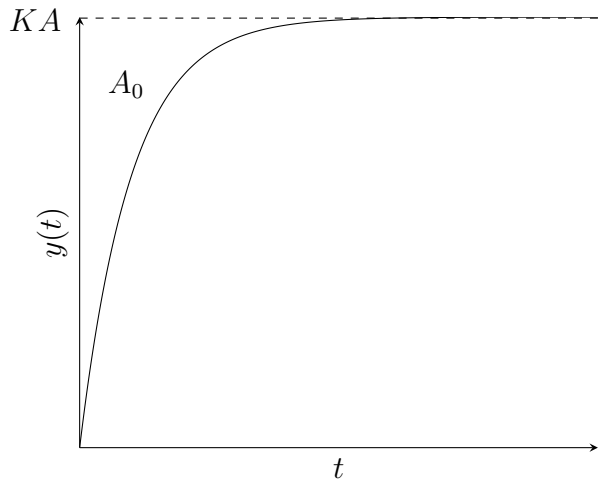


Figura 2.18: Gráfico da resposta ao degrau.

Para determinar o parâmetro  $K$ , deve-se encontrar o limite de  $t$  tendendo ao infinito para  $y(t)$ :

$$\lim_{t \rightarrow \infty} y(t) = KA = y_{\infty}, \quad (2.36)$$

Logo,

$$K = \frac{y_{\infty}}{A}. \quad (2.37)$$

## Método da área

O método da área é utilizado para determinar o parâmetro  $\tau$  do motor. Segundo ÅSTRÖM e HÄGGLUND [10], por meio da resposta ao degrau, deve-se calcular a área da região  $A_0$  mostrada no gráfico da figura 2.18, encontrando a seguinte equação:

$$A_0 = \int_0^{\infty} [KA - y(t)] dt, \quad (2.38)$$

$$A_0 = -KA\tau e^{-t/\tau} \Big|_0^{\infty}, \quad (2.39)$$

$$A_0 = KA\tau, \quad (2.40)$$

isolando  $\tau$ , tem-se:

$$\tau = \frac{A_0}{KA}. \quad (2.41)$$

Substituindo o valor encontrado na equação 2.36 na equação 2.41, o valor de  $\tau$  é encontrado a partir da seguinte equação:

$$\tau = \frac{A_0}{y_{\infty}}. \quad (2.42)$$

## Método do logaritmo neperiano

O método do logaritmo neperiano também é utilizado para determinar o parâmetro  $\tau$  do motor. Segundo ÅSTRÖM e HÄGGLUND [10], considera-se os instantes de tempo  $t$ , tal que  $y(t) < y_{\infty} = KA$ , em que  $A$  é a amplitude do degrau. Assim, é possível reescrever a equação da resposta ao sistema da seguinte forma:

$$y(t) = KA(1 - e^{-t/\tau}), \quad (2.43)$$

$$y(t) = KA - KAe^{-t/\tau}, \quad (2.44)$$

$$KAe^{-t/\tau} = KA - y(t). \quad (2.45)$$

Aplicando-se o logaritmo natural ( $\ln$ ) em ambos os lados da equação 2.45, encontra-se o seguinte resultado:

$$\ln(KAe^{-t/\tau}) = \ln(KA - y(t)), \quad (2.46)$$

$$\ln KA + (-t/\tau) = \ln(KA - y(t)), \quad (2.47)$$

$$\ln KA - \ln(KA - y(t)) = \frac{t}{\tau}. \quad (2.48)$$



Substituindo  $KA = y_\infty$  na equação 2.48, obtém-se:

$$\frac{t}{\tau} = \ln \left( \frac{y_\infty}{y_\infty - y(t)} \right). \quad (2.49)$$

Podendo ser reescrita, por:

$$at = b. \quad (2.50)$$

Em que:

$$\begin{cases} a = \frac{1}{\tau}, \\ b = \ln \left( \frac{y_\infty}{y_\infty - y(t)} \right). \end{cases}$$

Como na equação 2.50,  $b$  não pode ser dividido por  $t$ , pois  $t$  é um vetor, deve-se multiplicar ambos os lados da equação por  $t^T$ , resultando em:

$$\underline{t}^T \underline{t} a = \underline{t}^T \underline{b}, \quad (2.51)$$

sendo possível, assim, encontrar o valor de  $a$ :

$$a = \frac{\underline{t}^T \underline{b}}{\underline{t}^T \underline{t}}, \quad (2.52)$$

$$a = \frac{\underline{t}^T \underline{b}}{\|\underline{t}\|^2}. \quad (2.53)$$

Assim,  $\tau$  é encontrado a partir da seguinte equação:

$$\tau = \frac{1}{a}. \quad (2.54)$$

## 2.6 Controle PI

O controlador PI (proporcional - integral) é um dos controles mais comumente utilizados tanto no meio acadêmico como na indústria. Com ele é possível avaliar o erro entre a variável controlada e seu valor de referência e, baseando-se nesse erro, o controlador determina um sinal de controle de forma a diminuir tal desvio.

O motivo de ser usado um controle PI neste trabalho, é que usando somente o controle proporcional, existiria um erro no regime estacionário. Tal erro é eliminado se for incluído uma ação integral ao controlador.

### 2.6.1 Ação proporcional

Para um controlador com ação proporcional, segundo FRANKLIN *et al.* [4] e OGATA [11], a relação entre o sinal de saída,  $u(t)$ , e o sinal de erro,  $e(t)$ , é:

$$u(t) = K_p e(t), \quad (2.55)$$

em que  $K_p$  é o ganho proporcional.

No domínio da frequência, utilizando a transformada de Laplace na equação 2.55, tem-se:

$$\frac{U(s)}{E(s)} = K_p. \quad (2.56)$$

O controlador proporcional, é essencialmente um amplificador com ganho ajustável, aumentando a rapidez do sistema.

### 2.6.2 Ação integral

Para um controlador com ação integral, segundo FRANKLIN *et al.* [4] e OGATA [11], a saída do controlador,  $u(t)$ , varia com uma taxa proporcional ao sinal de erro,  $e(t)$ ,

$$\frac{du(t)}{dt} = K_i e(t) \Rightarrow u(t) = K_i \int_0^t e(\lambda) d\lambda, \quad (2.57)$$

em que  $K_i$  é o ganho integral.

No domínio da frequência, utilizando a transformada de Laplace na equação 2.57, tem-se:

$$\frac{U(s)}{E(s)} = \frac{K_i}{s}. \quad (2.58)$$

A ação integral é utilizada para eliminar o erro no regime permanente.

Em OGATA [11], também é mostrado que a ação integral é suficiente para rejeitar perturbações externas do tipo degrau, mantendo o erro no regime permanente nulo.

### 2.6.3 Ação proporcional e integral

O controlador PI possui, segundo FRANKLIN *et al.* [4] e OGATA [11], a seguinte lei de formação:

$$u(t) = K_p e(t) + \frac{K_p}{\tau_i} \int_0^t e(\lambda) d\lambda. \quad (2.59)$$

No domínio da frequência, utilizando a transformada de Laplace na equação 2.59, tem-se: :

$$\frac{U(s)}{E(s)} = K_p \left( 1 + \frac{1}{\tau_i s} \right), \quad (2.60)$$

em que a constante  $\tau_i = \frac{1}{K_i}$  é chamada de tempo integral, que ajusta a ação do controle integral. Se houver uma variação no valor da constante proporcional,  $K_p$ , afeta tanto a parte proporcional quanto a parte integral do controle.

## 2.6.4 Especificações no domínio do tempo

Para determinar as constantes  $K_p$  e  $\tau_i$  da equação 2.60 do controle PI, é preciso que alguns requisitos, associados à resposta temporal do sistema, sejam especificados. De acordo com a figura 2.19, é possível observar dois desses requisitos que serão levados em consideração neste trabalho, o tempo de acomodação ( $ts$ ) e o percentual de *overshoot* (P.O.).

Uma vez escolhidos os valores para o percentual de *overshoot* e para o tempo de acomodação, é possível encontrar as constantes  $\zeta$  e  $\omega_n$ , que são os parâmetros da equação característica para um sistema de segunda ordem sem zeros finitos, apresentada, na equação 2.61.

$$\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}, \quad (2.61)$$

em que  $\omega_n$  é a frequência natural de oscilação e  $\zeta$  é a constante de amortecimento do sistema.

### Percentual de *overshoot*

O percentual de *overshoot* é o valor máximo, em porcentagem, que o sistema supera seu valor final, sendo calculado, segundo FRANKLIN *et al.* [4], pela equação 2.62.

$$\text{P.O.} = \exp\left(-\frac{\zeta\pi}{\sqrt{1-\zeta^2}}\right) \times 100\%. \quad (2.62)$$

### Tempo de acomodação

O tempo de acomodação consiste no tempo necessário para o sistema sair do estado transitório para o regime permanente com um erro de  $\pm 1\%$ , sendo calculado, segundo FRANKLIN *et al.* [4], pela equação 2.63.

$$ts = \frac{4.6}{\zeta\omega_n}. \quad (2.63)$$

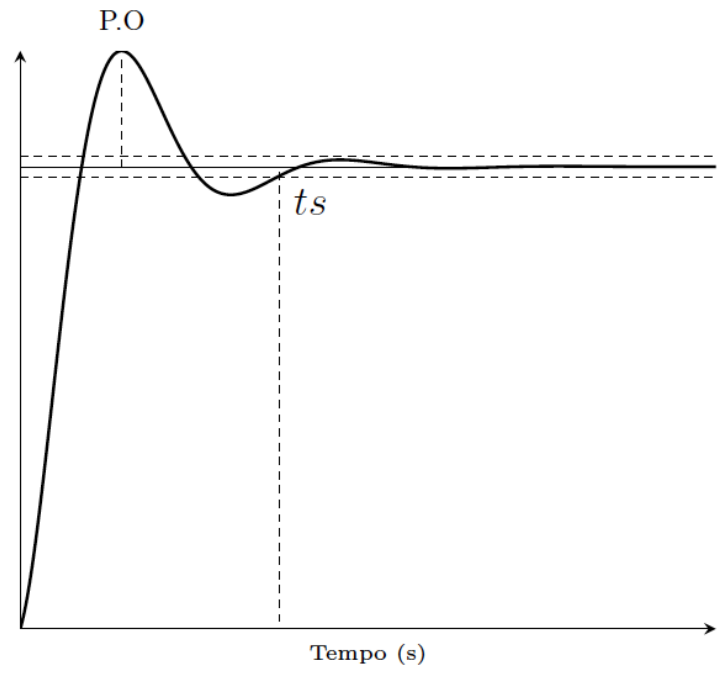


Figura 2.19: Definição do percentual de *overshoot* e do tempo de acomodação.

# Capítulo 3

## Método Proposto

Neste capítulo será explicado todos os passos para a execução do trabalho. Passando pelos materiais necessários para a preparação do circuito e como foi feita toda a montagem em si.

### 3.1 Material utilizado

Os materiais utilizados, foram:

- Arduino Mega2560;
- Ponte H 298N;
- Protoboard;
- Osciloscópio;
- Fonte;
- Multímetro;
- Amplificador Operacional LF356N;
- Motor CC - Potência: 0.1 kW, Tensão: 24 V, Rotação: 1800 rpm, Corrente máxima: 3.8 A;
- Resistor de 330 k $\Omega$ ;
- Resistor de 100 k $\Omega$ .

### 3.2 Montagem

Primeiramente, monta-se na protoboard o divisor de tensão, como apresentado no capítulo 2, apresentado na figura 3.1. Os valores escolhidos para os resistores são  $R_1 = 330 \text{ k}\Omega$  e  $R_2 = 100 \text{ k}\Omega$ .

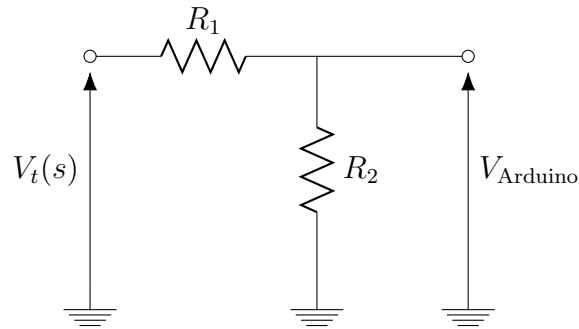


Figura 3.1: Esquema do divisor de tensão.

Com o divisor de tensão montado na protoboard, conecta-se à sua saída a entrada não inversora do amplificador operacional LF356N, como visto na figura 3.2, e a entrada inversora é ligada à saída do amplificador operacional. À saída do amplificador também é conectada uma entrada analógica do Arduino, no caso deste trabalho é utilizada a porta **A0**. Ainda deve-se alimentar o amplificador operacional com uma tensão de  $+20\text{ V}$  e  $-20\text{ V}$  nas respectivas portas  $+V_{cc}$  e  $-V_{cc}$ , através de uma fonte de tensão.

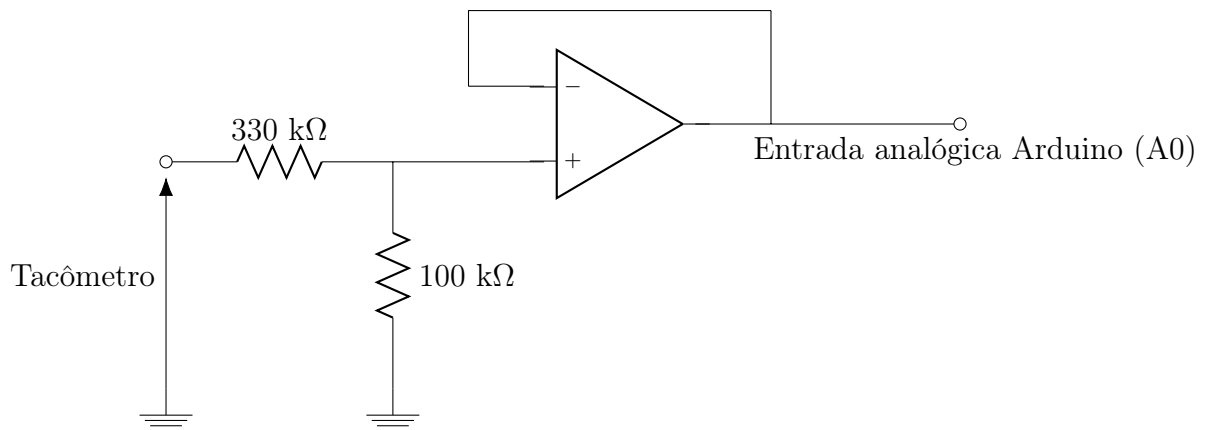


Figura 3.2: Demonstrativo de como montar o divisor de tensão ligado ao amplificador operacional.

Ligada à entrada do divisor de tensão tem-se o terminal positivo do tacômetro, como pode ser observado na figura 3.2, já o terminal negativo do tacômetro é conectado ao terra na protoboard, juntamente com o terra proveniente da fonte.

Alimenta-se o Arduino pelo cabo USB conectado à um computador, e liga-se a porta **GND** do Arduino ao terra na protoboard.

Para fazer a ligação do Arduino com a ponte H, deve-se conectar os pinos de entrada da ponte, neste trabalho foram os pinos **Ativa MA**, **IN1** e **IN2** com os pinos PWM do Arduino, **5**, **6** e **7**, respectivamente.

Para alimentar a ponte H, deve-se alimentá-la com uma tensão de +20 V provenientes da fonte, na porta **6-35 V** da ponte, e o terra no pino **GND** da ponte H.

O motor deve ser conectado à ponte H nas entradas denominadas de **Motor A**, e ainda, deve-se conectar o **GND** da fonte no neutro do motor.

A montagem do circuito pode ser vista no esquemático na figura 3.3, e na figura 3.4, como fica a montagem utilizando a protoboard.

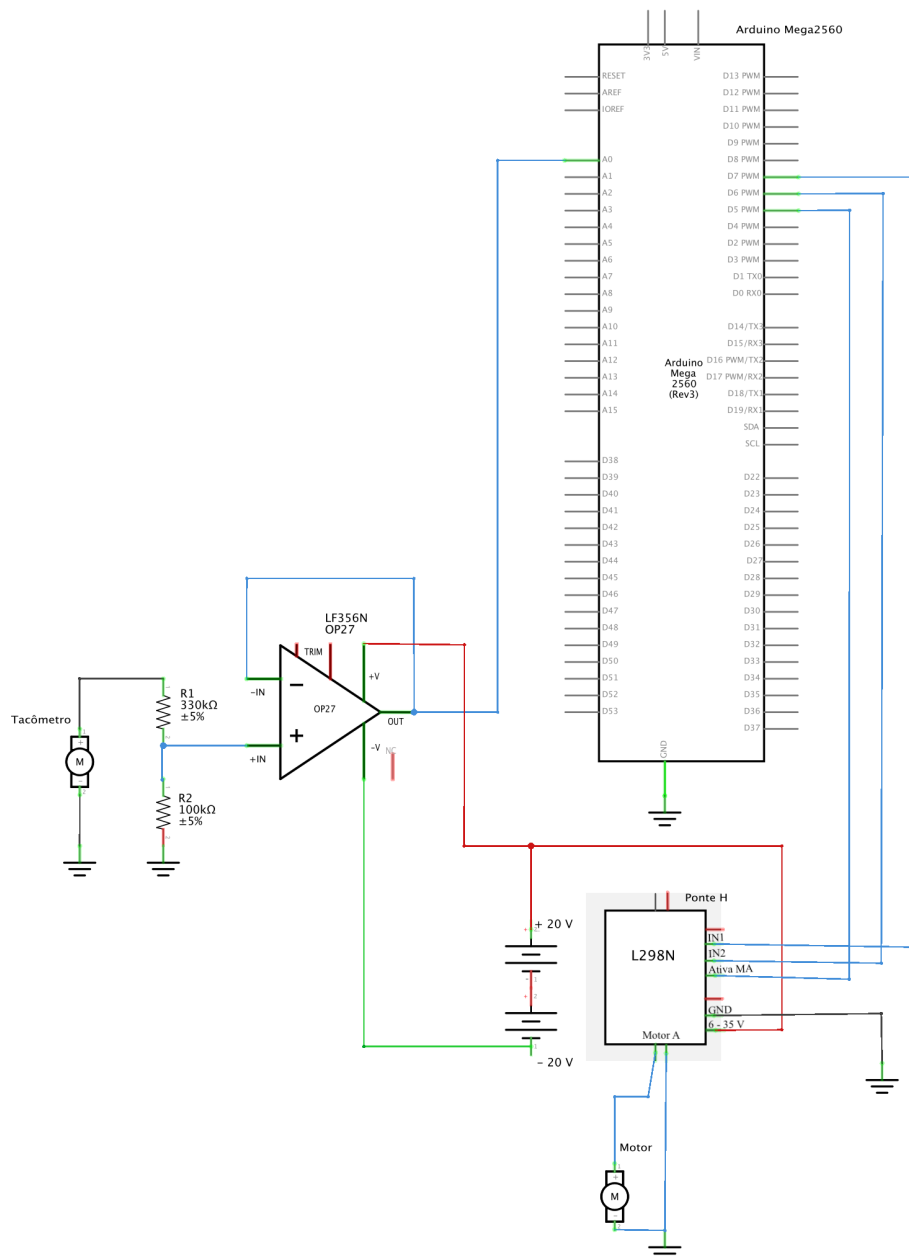


Figura 3.3: Esquemático da montagem do circuito.

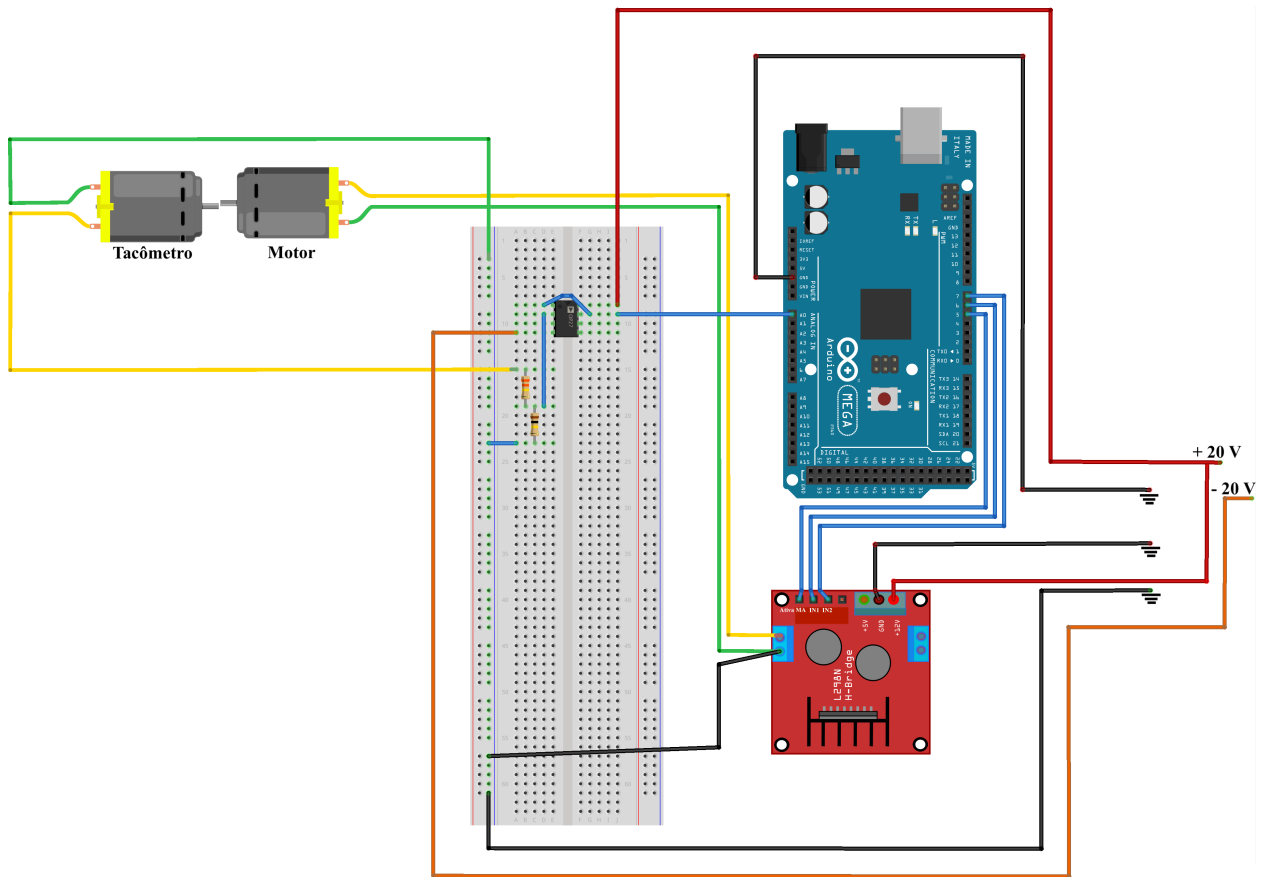


Figura 3.4: Esquema da montagem utilizando a protoboard.

Já com o circuito montado e alimentado, acopla-se o osciloscópio ligado ao tacômetro do motor CC, com o intuito de analisar os dados.

### 3.3 Aquisição de dados

A aquisição de dados, é feita através do próprio programa do Arduino, com o Monitor Serial da IDE, mostrado na figura 3.5, em que é possível imprimir as variáveis desejadas, utilizando o seguinte comando;

```

1 Serial.print(X);
2 Serial.print("_");
3 Serial.println(Y);

```

em que X e Y serão as variáveis que aparecerão no monitor serial da IDE.

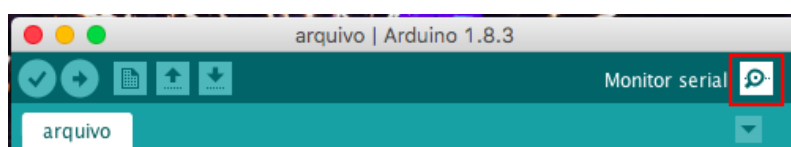


Figura 3.5: Botão para o monitor serial do Arduino.



Com os valores no Monitor Serial, é possível copiá-los para um arquivo de texto, e assim, exportá-los para o MATLAB. Com o arquivo já exportado, deve-se marcar a opção *Delimited* no MATLAB, para passar os dados para formato numérico, e depois importá-los como *Column Vectors*, como visto na figura 3.6. Depois de todos os dados estarem no *workspace* do MATLAB, é possível ser feita a construção de gráficos, comparações e manipulações.

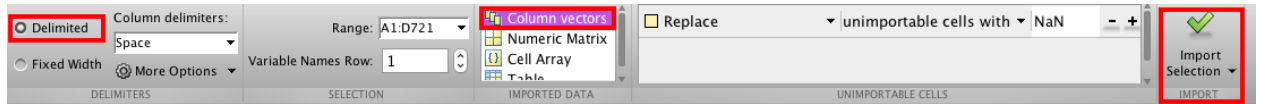


Figura 3.6: Opções que devem ser escolhidas para exportar os dados para o MATLAB.

# Capítulo 4

## Resultados e Discussões

Neste capítulo é apresentado o procedimento experimental necessário para o cálculo dos parâmetros do motor CC e também a sintonização do controle PI, assim como a validação dos resultados obtidos.

### 4.1 Determinação dos parâmetros do motor CC

Como visto nos capítulos anteriores, a função de transferência do motor CC pode ser escrita como sendo de primeira ordem, como na equação 4.1,

$$G(s) = \frac{V_t(s)}{V_a(s)} = \frac{K}{\tau s + 1}, \quad (4.1)$$

em que  $K = K_a K_t$ .

Com todo o circuito montado, como visto anteriormente, é feito o primeiro código, apêndice A.1.1, em linguagem Arduino, em que é utilizada a biblioteca DMPH.h, que facilita a implementação do código para o uso da ponte H.

Utilizando essa biblioteca é necessário apenas instanciar o objeto motor, escolhendo três pinos de conexão como parâmetros,

```
1 #include <DMPH.h>
2 DMPH motor(6, 5, 7);
```

em que 6, 5 e 7 são os pinos PWM referentes ao Arduino.

Para fazer com que o motor se mova, é necessário apenas um comando,

```
1 motor.move(j);
```

em que  $j$  é o valor em bits PWM (0 - 255) que deseja-se que o motor se mova. Para alterar o sentido de rotação, é apenas necessário mudar o sinal do parâmetro.

No programa, são escolhidos valores em bits, entre 0 – 255, e a partir de cada valor escolhido, é feita a medição da tensão de entrada do motor e de saída do tacômetro com um multímetro. Por último, com um tacômetro digital, mede-se a rotação do motor em rpm. A partir desses dados, é feita a tabela 4.1.

Tabela 4.1: Parâmetros do motor CC, utilizando alimentação de 20 V.

| Entrada PWM em bits (0 - 255) | Tensão de Entrada do Motor (V) | Tensão de Saída do Tacômetro (V) | Saída do Tacômetro (rpm) |
|-------------------------------|--------------------------------|----------------------------------|--------------------------|
| 255                           | 18.24                          | 18.94                            | 1213                     |
| 245                           | 17.57                          | 17.91                            | 1158                     |
| 235                           | 17.23                          | 17.61                            | 1138                     |
| 225                           | 16.82                          | 17.16                            | 1107                     |
| 215                           | 16.51                          | 16.89                            | 1084                     |
| 205                           | 16.16                          | 16.50                            | 1065                     |
| 195                           | 15.78                          | 16.08                            | 1036                     |
| 185                           | 15.35                          | 15.50                            | 1002                     |
| 175                           | 14.95                          | 15.11                            | 973.3                    |
| 165                           | 14.43                          | 14.61                            | 935.6                    |
| 155                           | 13.87                          | 14.05                            | 898.4                    |
| 145                           | 13.34                          | 13.53                            | 871.1                    |
| 135                           | 12.67                          | 12.82                            | 826.6                    |
| 125                           | 11.91                          | 11.89                            | 766.3                    |
| 115                           | 11.09                          | 10.99                            | 705.2                    |
| 105                           | 10.20                          | 9.97                             | 638.1                    |
| 95                            | 9.18                           | 8.87                             | 569.7                    |
| 85                            | 8.04                           | 7.57                             | 481.1                    |
| 75                            | 6.81                           | 6.10                             | 387.6                    |
| 65                            | 5.431                          | 4.528                            | 288.9                    |
| 55                            | 3.991                          | 2.811                            | 174.7                    |
| 45                            | 2.670                          | 1.138                            | 77.6                     |
| 35                            | 1.546                          | 0                                | 0                        |
| 25                            | 1.052                          | 0                                | 0                        |
| 15                            | 0.607                          | 0                                | 0                        |
| 5                             | 0.165                          | 0                                | 0                        |

A partir da tabela 4.1, é possível desenhar o gráfico da figura 4.1, em que observa-se que a ponte H não possuiu um comportamento linear, mesmo não levando em consideração a zona morta.

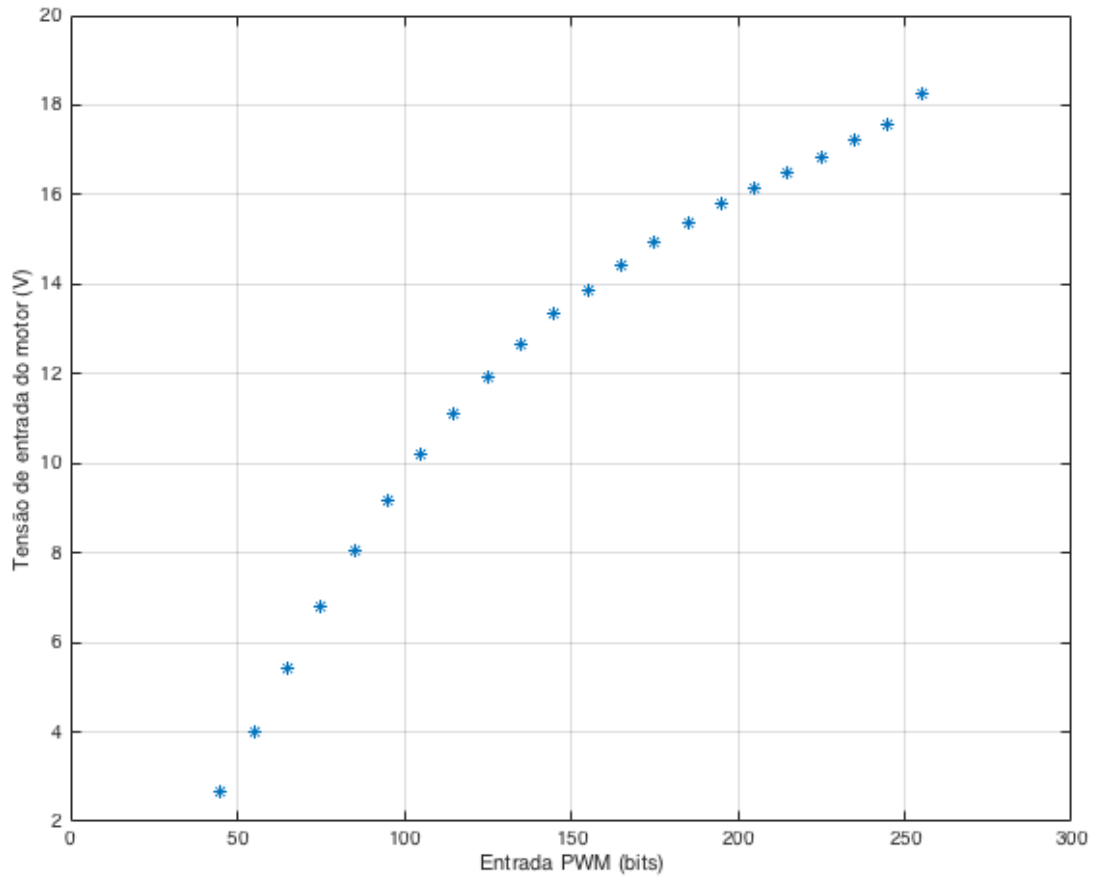


Figura 4.1: Não linearidade da ponte H, desconsiderando a zona morta.

Porém, é possível aproximar tal curva por uma função polinomial, utilizando o seguinte código no Matlab:

```

1 a = polyfit(x, y, N);
2 b = polyval(a, x);
3 plot(x, y, x, b)

```

em que  $x$  e  $y$  são os vetores a serem ajustados, e  $N$  é o grau da função polinomial de ajuste.

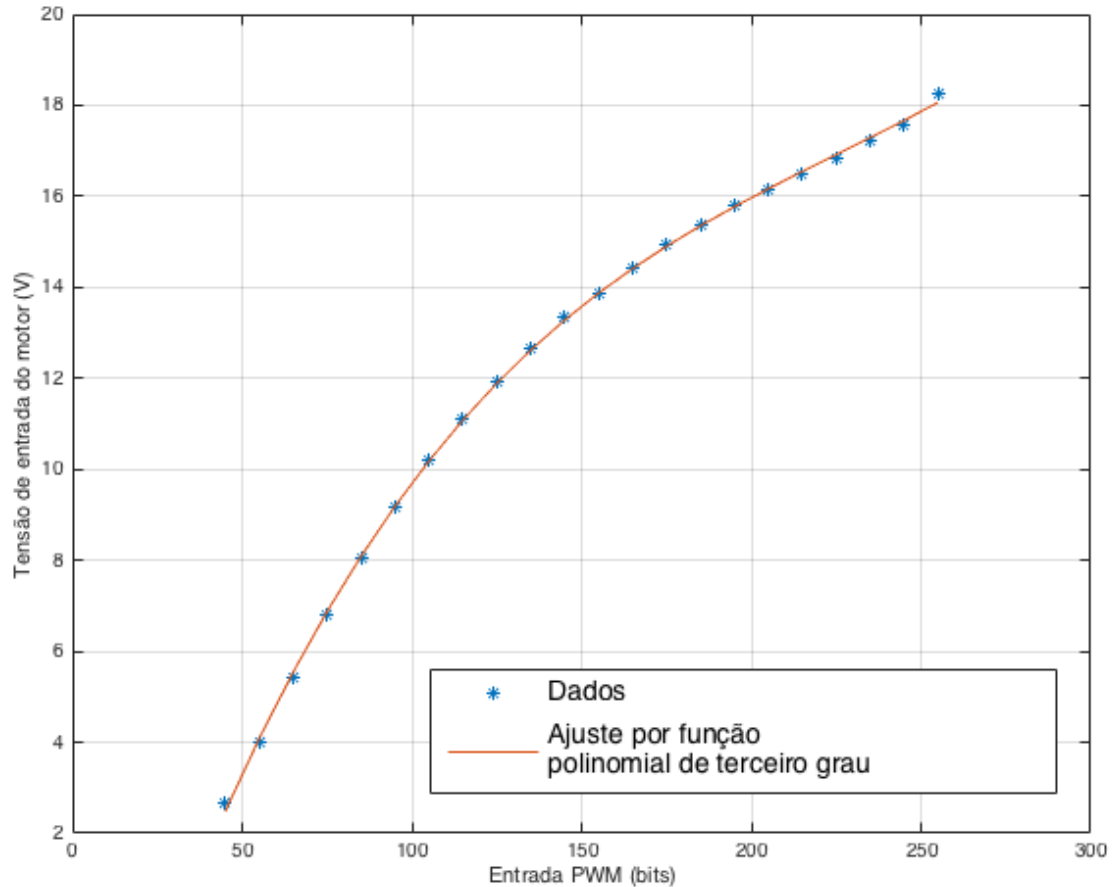


Figura 4.2: Ajuste dos dados da ponte H por uma função polinomial de terceiro grau.

Como visto no gráfico da figura 4.2, a curva foi muito bem ajustada pela seguinte função de terceiro grau:

$$f(x) = 0.000001344577352x^3 - 0.000905270847573x^2 + 0.240201105171526x - 6.616534474464332. \quad (4.2)$$

Tal aproximação por uma função polinomial é importante pois com ela, é possível mensurar a não linearidade empregada pela ponte H, obtendo assim, uma relação entre os bits PWM (0 - 255) fornecidos no Arduino para a tensão (0 - 20 V), que de fato é entregue na entrada do motor. A relação inversa, isto é, de volts para bits PWM, é feita invertendo a ordem da entrada do comando *polyfit* do Matlab, resultando na seguinte equação de sexto grau:

$$g(x) = -0.000252334278442x^6 + 0.015023029522137x^5 - 0.350517081520322x^4 + 4.108721669072144x^3 - 25.10353922037313x^2 + 381.769971910966930x - 56.998334302062169. \quad (4.3)$$

Como visto no gráfico da figura 4.3, a curva foi bem ajustada pela equação de sexto grau 4.3.

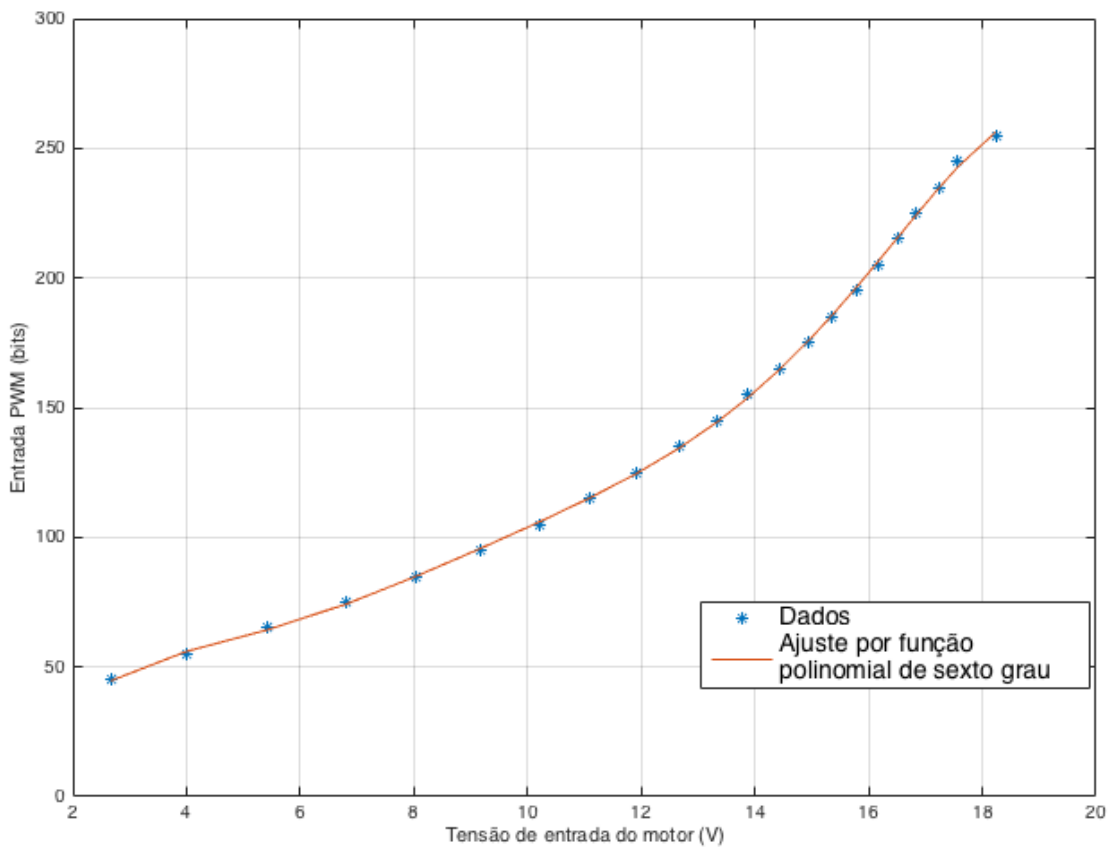


Figura 4.3: Ajuste dos dados da ponte H por uma função polinomial de sexto grau.

Com a transformação da equação 4.3, é possível escolher um valor de tensão de referência na IDE do Arduino, ao invés de se trabalhar com bits PWM, quando deseja-se uma tensão na entrada do motor.

É importante ressaltar que o Arduino trabalha com no máximo seis casas decimais, logo é necessário fazer uma aproximação da equação 4.3 para seis casas decimais quando for utilizada no programa.

#### 4.1.1 Simplificações

Devido ao ganho do divisor de tensão, a entrada A/D do Arduino e a interação não linear entre o *duty cycle* fornecido no Arduino e a saída da ponte H, há uma necessidade de compensação de todos esses efeitos no código. Antes de se propor as compensações, serão contabilizados cada um desses efeitos.

Como visto anteriormente, o diagrama de blocos utilizado neste trabalho, pode ser visto na figura 2.11, e está sendo representado na figura 4.4, porém agora com um destaque para o ganho do divisor de tensão mais a entrada A/D do Arduino, que será considerado um único ganho, daqui para frente, representado como  $K_{Arduino}$ . Outro destaque para a saída D/PWM do Arduino mais a ponte H, denominada a partir desse momento apenas de Ponte, representando o comportamento não linear, mostrado pela figura 4.2. Assim a figura 4.4, pode ser redesenhada como mostra a figura 4.5.

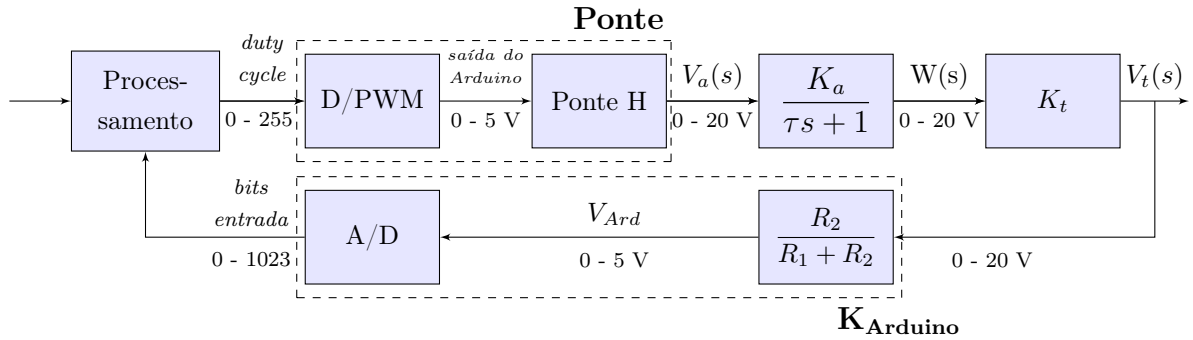


Figura 4.4: Diagrama de blocos do sistema.

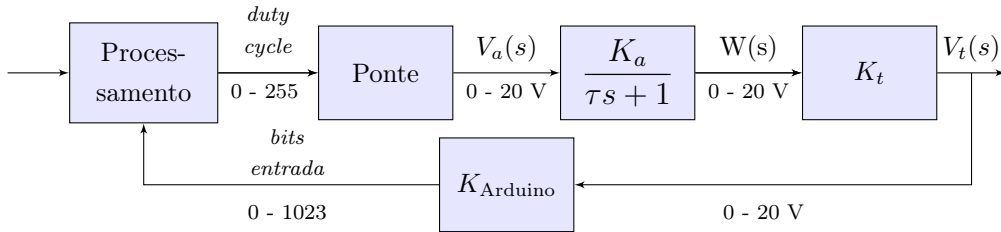


Figura 4.5: Diagrama de blocos do sistema simplificado.

O ganho  $K_{Arduino}$ , consiste tanto no divisor de tensão como no conversor A/D, assim,  $K_{Arduino}$  será dado pela equação 4.4:

$$K_{Arduino} = \frac{1023}{5} \times \frac{R_2}{R_1 + R_2}, \quad (4.4)$$

em que  $R_1$  e  $R_2$  são os valores medidos com um multímetro, pois pode haver discrepâncias entre o valor medido e o especificado pelo fabricante. Neste trabalho, os valores medidos foram  $R_1 = 324.5 \text{ k}\Omega$  e  $R_2 = 101.2 \text{ k}\Omega$ , logo o  $K_{Arduino}$  utilizado foi de:

$$K_{Arduino} = \frac{1023}{5} \times 0.2377. \quad (4.5)$$

O efeito não linear denominado Ponte, consiste tanto na saída D/PWM como na relação entre entrada e saída da ponte H. Assim, o bloco Ponte é caracterizada pela equação de terceiro grau 4.2.

Por fim, para determinar os parâmetros do sistema, o bloco Processamento pode ser expandido nos dois blocos de compensação e a malha pode permanecer aberta, uma vez que deseja-se apenas coletar a tensão do tacômetro.

Os dois blocos de compensação são: Equação não linear e  $K_{\text{Arduino}}^{-1}$ , apresentado na figura 4.6, em que o bloco Equação não linear faz a compensação não linear da ponte H utilizando a equação 4.3, e o  $K_{\text{Arduino}}^{-1}$ , que nada mais é que o inverso do ganho  $K_{\text{Arduino}}$ , faz a compensação do ganho  $K_{\text{Arduino}}$ .

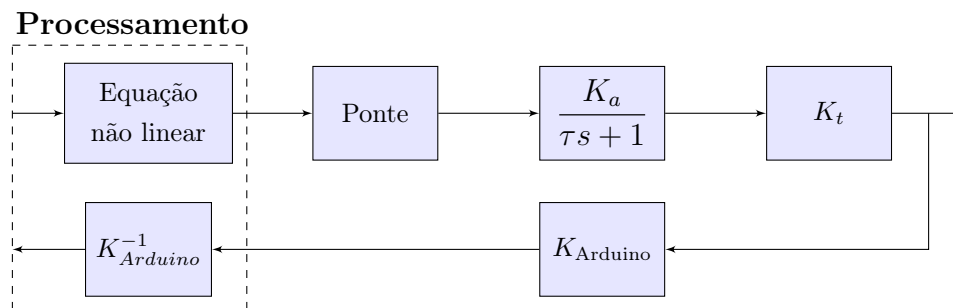


Figura 4.6: Diagrama de blocos do sistema para determinação dos parâmetros.

A Equação não linear é implementada na IDE do Arduino a partir do seguinte código:

```
1 float bits = -0.000252*pow(degrau,6) + 0.015023*pow(
    degrau,5) -0.350517*pow(degrau,4) + 4.108722*pow(
    degrau,3) -25.103539*pow(degrau,2) +81.769972*degrau
    -56.998334;
```

O ganho  $K_{\text{Arduino}}^{-1}$  é implementado na IDE a partir do seguinte código:

```
1 Velocidade = analogRead(Sensor) * (5.0/1023.0) * 4.2065;
```

### 4.1.2 Cálculo dos parâmetros

A partir da tabela 4.1, é possível determinar a região linear de operação do motor, pois o mesmo possui uma zona morta. Determinar o intervalo linear é muito importante, para que todos os dados coletados que serão utilizados no método dos mínimos quadrados, da área e do logaritmo neperiano estejam dentro desta região.



Os valores que compreendem tal região podem ser melhor vistos no gráfico da figura 4.7, em que os valores considerados pertencentes ao intervalo linear vão de aproximadamente 3 V até 18 V da tensão de entrada do motor.

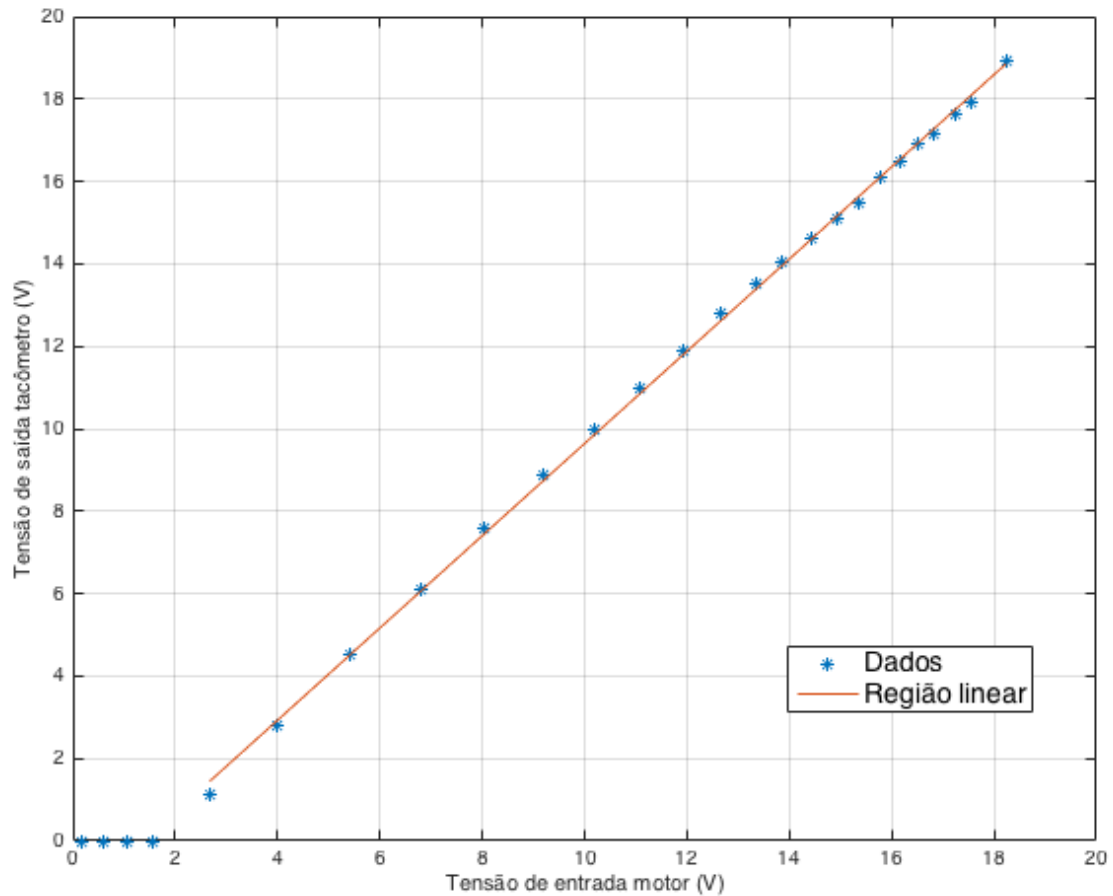


Figura 4.7: Região linear de operação do motor.

Com a região linear definida, um novo código é escrito, apêndice A.1.2, em que escolhe-se degraus de tensão dentro do intervalo linear (3 V - 18 V). Com os resultados obtidos a partir desses experimentos, pode-se utilizar os métodos da resposta ao degrau, da área e do logaritmo neperiano, vistos anteriormente, para o cálculo das constantes do motor.

Primeiramente, são feitos vários experimentos com valores diferentes de degraus, como visto na tabela 4.2.

Tabela 4.2: Valores de  $K$  e  $\tau$  obtidos pelos métodos da resposta ao degrau, da área e logaritmo neperiano a partir de diferentes amplitudes de degraus.

| Amplitude do degrau (V) | Resposta ao degrau (V/V) | Método da área (s) | Método do logaritmo neperiano (s) |
|-------------------------|--------------------------|--------------------|-----------------------------------|
| 7 - 12                  | $K = 1.2155$             | $\tau = 0.3838$    | $\tau = 0.3418$                   |
| 9 - 15                  | $K = 1.2012$             | $\tau = 0.2579$    | $\tau = 0.3700$                   |
| 8 - 14                  | $K = 1.2023$             | $\tau = 0.3716$    | $\tau = 0.3670$                   |
| 8 - 15                  | $K = 1.2349$             | $\tau = 0.2036$    | $\tau = 0.3008$                   |
| 8 - 12                  | $K = 1.1111$             | $\tau = 0.3055$    | $\tau = 0.2784$                   |
| 9 - 13                  | $K = 1.2224$             | $\tau = 0.2850$    | $\tau = 0.2783$                   |
| 6 - 14                  | $K = 1.2282$             | $\tau = 0.3375$    | $\tau = 0.3647$                   |
| 6 - 13                  | $K = 1.1731$             | $\tau = 0.2720$    | $\tau = 0.3716$                   |
| 5 - 12                  | $K = 1.1893$             | $\tau = 0.3038$    | $\tau = 0.3500$                   |
| 5 - 15                  | $K = 1.1723$             | $\tau = 0.2253$    | $\tau = 0.2145$                   |

A partir dos dados da tabela 4.2, é feita a tabela 4.3, em que estão apresentados os valores das médias e seus respectivos desvios padrões das constantes obtidas pelos três métodos.

Tabela 4.3: Média e desvio padrão das constantes  $K$  e  $\tau$  calculadas pelos métodos da resposta ao degrau, da área e do logaritmo neperiano.

|            | Resposta ao degrau  | Método da área      | Método do logaritmo neperiano |
|------------|---------------------|---------------------|-------------------------------|
| $K$ (V/V)  | $1.1950 \pm 0.0366$ | -                   | -                             |
| $\tau$ (s) | -                   | $0.2946 \pm 0.0587$ | $0.3237 \pm 0.0533$           |

A partir dos dados da tabela 4.1, pode-se obter os valores de  $K$ ,  $K_a$  e  $K_t$  pelo método dos mínimos quadrados, visto no capítulo 2. Encontrando assim, os seguintes valores para as constantes:

$$\begin{cases} K_a = 7.673160 \text{ (V/rpm)}, \\ K_t = 0.149160 \text{ (rpm/V)}, \\ K = 1.144496 \text{ (V/V)}, \\ K = K_a \times K_t = 1.144531 \text{ (V/V)}. \end{cases}$$

Sendo importante ressaltar que os valores em *rpm* da saída do tacômetro devem ser transformados para *rad/s* para serem utilizados no método dos mínimos quadrados, sendo apenas necessário a multiplicação por  $(\frac{2\pi}{60})$ .

De acordo com o valores encontrados pelo método dos mínimos quadrados, é possível perceber que houve um erro relativo de aproximadamente  $3.0037 \times 10^{-5}$  entre os resultados de  $K$  calculados.

Para validar os valores encontrados, na figura 4.8 são apresentados os gráficos das constantes  $K_a$ ,  $K_t$  e  $K$ .

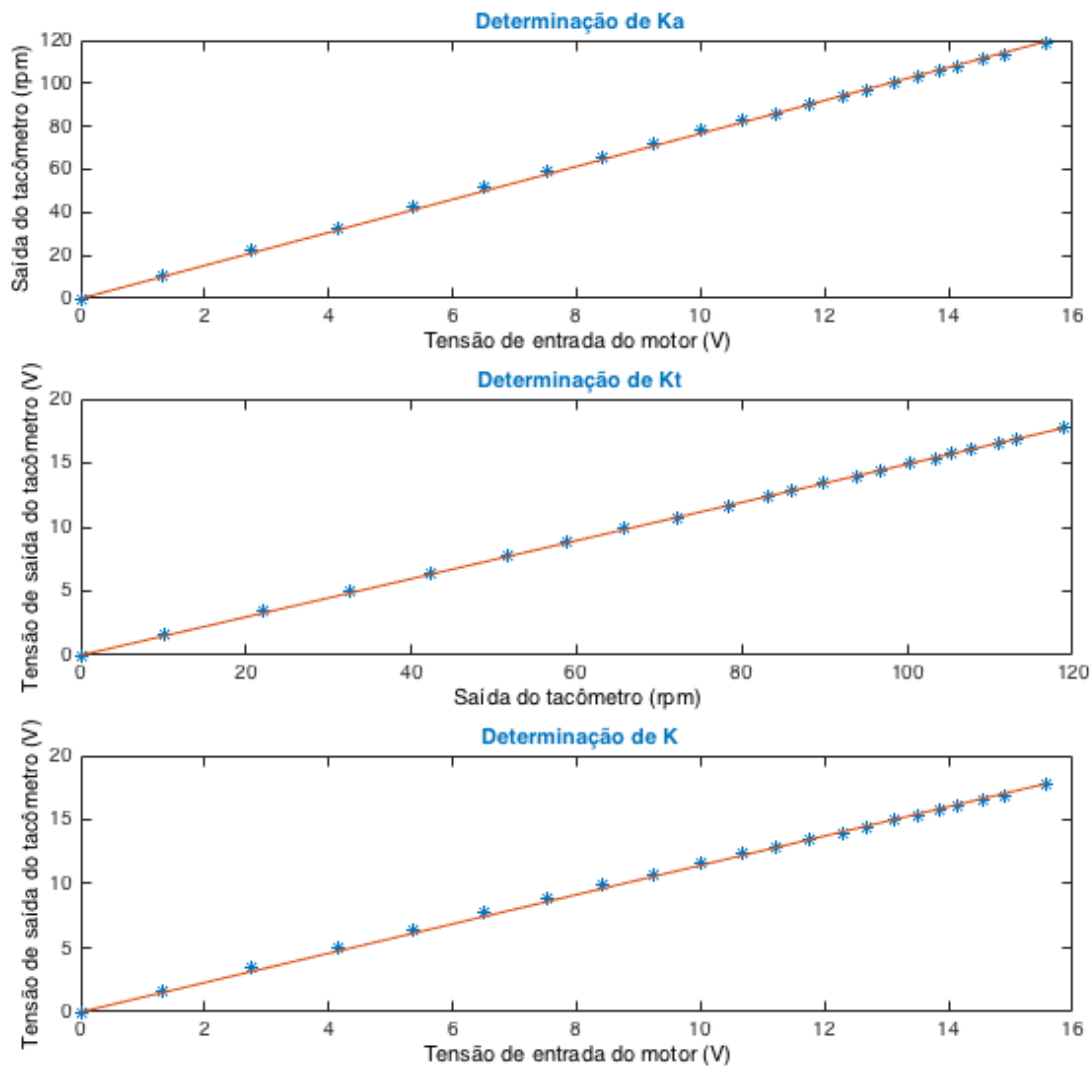


Figura 4.8: Validação das constantes  $K_a$ ,  $K_t$  e  $K$ .

Como pode-se observar nos gráficos da figura 4.8, os valores encontrados para as constantes estão bem ajustados, todos seguindo a reta do ajuste linear.

A partir da tabela 4.3 e dos resultados obtidos pelo método dos mínimos quadrados, é feita a tabela 4.4, em que encontra-se a média de todos os valores de  $K$  e  $\tau$  calculados pelos métodos presentes no trabalho e a média dos quatro métodos juntos.

Tabela 4.4: Valores das constantes  $K$  e  $\tau$  obtidas pelos métodos utilizados e suas médias.

|            | Método dos mínimos quadrados | Resposta ao degrau | Método da área | Método do logaritmo neperiano | Média dos métodos |
|------------|------------------------------|--------------------|----------------|-------------------------------|-------------------|
| $K$ (V/V)  | 1.1445                       | 1.1950             | -              | -                             | 1.1698            |
| $\tau$ (s) | -                            | -                  | 0.2946         | 0.3237                        | 0.3091            |

### 4.1.3 Simulação com os parâmetros encontrados

A partir dos resultados obtidos, é feita uma comparação entre todos os métodos apresentados, utilizando o Simulink do MATLAB, apresentado na figura 4.9, em que é possível comparar os resultados simulados, com a tensão de saída do tacômetro e com o degrau de entrada do sistema.

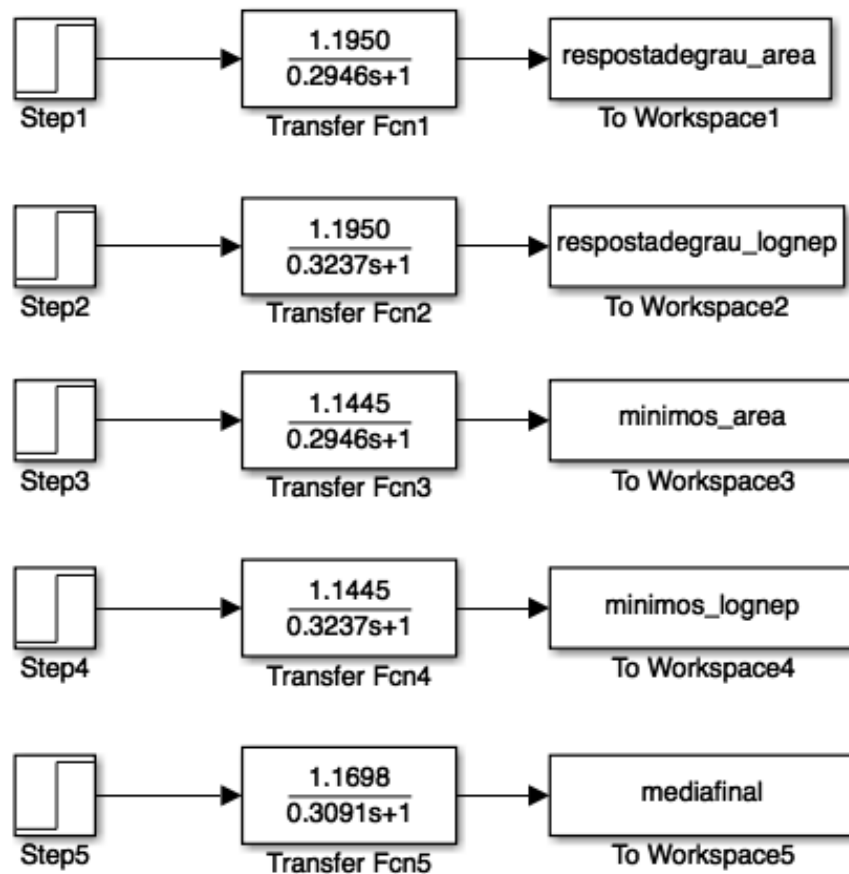


Figura 4.9: Modelo utilizado no Simulink para comparação dos valores simulados e experimentais.

Foi escolhido um degrau de referência dentro da região linear (9 - 13 V), visto no gráfico da figura 4.10. Porém, é possível observar que os sinais simulados não estão correspondendo ao experimental, isso ocorre pelo fato da origem (0,0) não fazer parte da região linear, ocorrendo uma variação na saída do tacômetro. Para compensar essa variação, deve-se passar o gráfico todo para a origem, subtraindo o sinal de saída do tacômetro pela sua média. O gráfico reajustado para a origem pode ser visto na figura 4.11, e nele é possível perceber que a variação foi compensada.

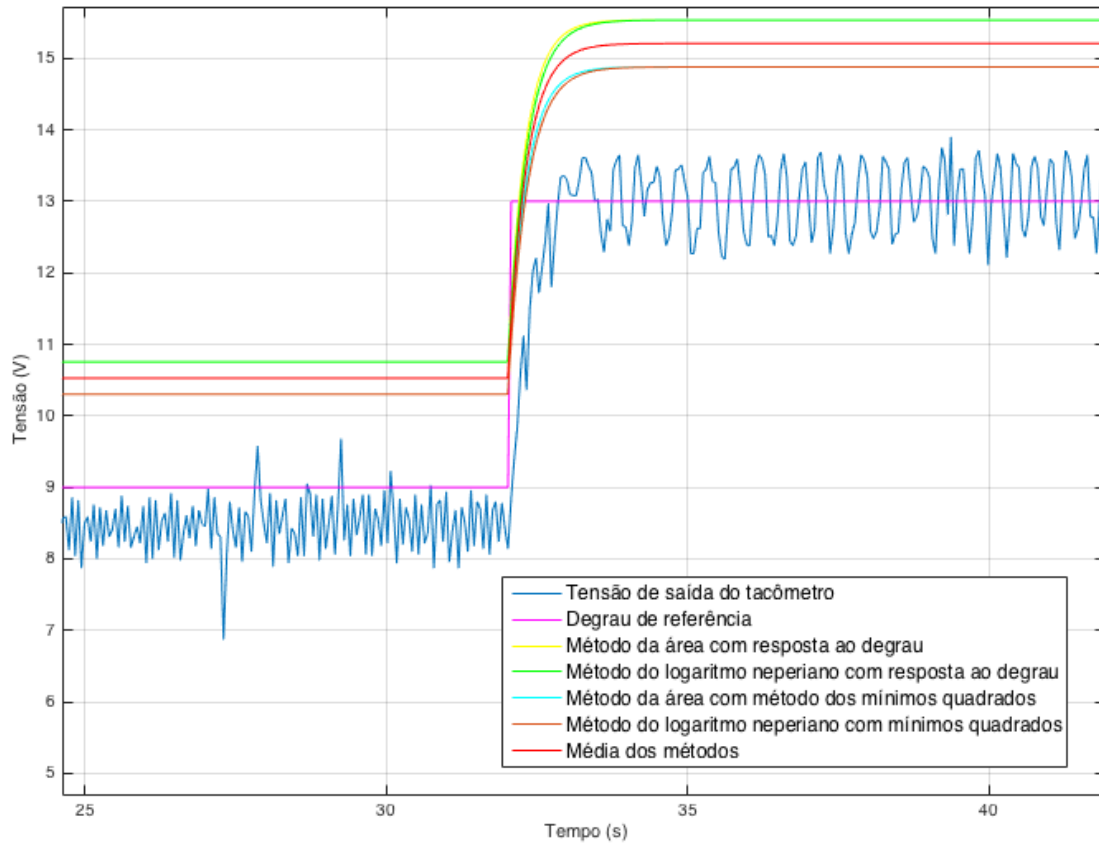


Figura 4.10: Comparação das saídas simuladas do sistema com a saída do Arduino.

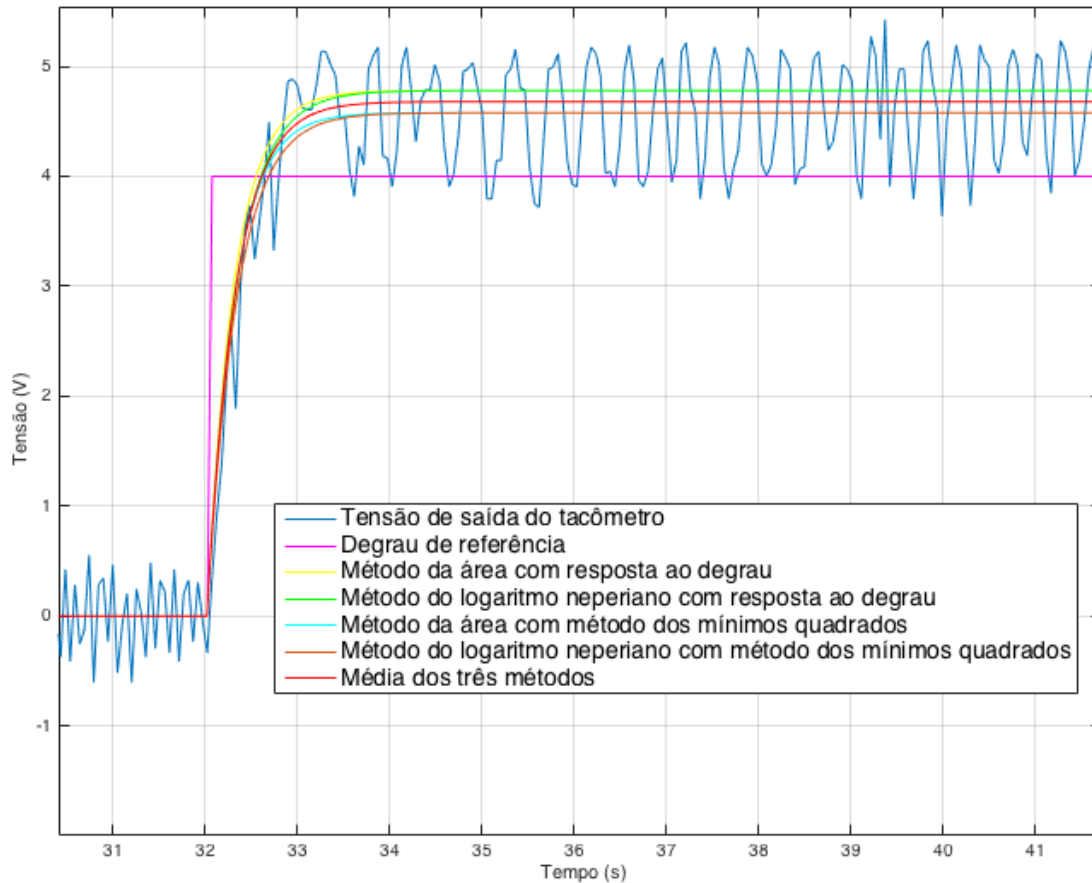


Figura 4.11: Comparação das saídas simuladas do sistema com a saída do Arduino reajusta para a origem.

Portanto, de acordo com o gráfico da figura 4.11, é possível concluir que os valores de  $K$  e  $\tau$  que mais se aproximam da média do sinal de saída do tacômetro, 4.5658 V, (curva azul escura), são os que foram obtidos pelo método da área e dos mínimos quadrados (curva azul claro), que estabiliza aproximadamente no valor de 4.578 V.

Resultando, por fim, na seguinte equação de primeira ordem para o motor mais tacômetro:

$$G(s) = \frac{1.1445}{0.2946s + 1}. \quad (4.6)$$

## 4.2 Sintonização do controlador PI

### 4.2.1 Cálculo das variáveis do controle

A partir dos valores de  $K$  e  $\tau$  encontrados é possível calcular os valores das constantes proporcional e integral do controlador PI.

Para realização dos cálculos das constantes, primeiramente é feita a função de transferência do sistema em malha fechada, a partir do diagrama de blocos mostrado na figura 4.12.

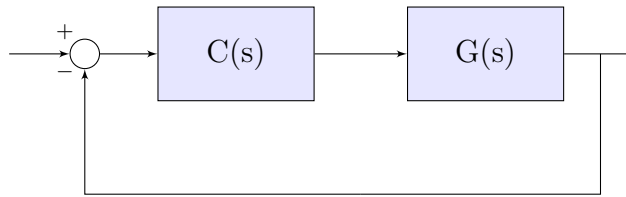


Figura 4.12: Diagrama de blocos do controle com a planta.

Em que o controlador,  $C(s)$ , possui a seguinte função de transferência:

$$C(s) = \frac{K_p \tau_i s + K_p}{\tau_i s}. \quad (4.7)$$

A planta do motor, como visto anteriormente, é um sistema de primeira ordem, que possui a seguinte função de transferência:

$$G(s) = \frac{K}{\tau s + 1}. \quad (4.8)$$

Encontrando a função de transferência do diagrama de blocos visto na figura 4.12, tem-se:

$$H(s) = \frac{C(s)G(s)}{1 + C(s)G(s)}. \quad (4.9)$$

Substituindo as equações 4.7 e 4.8 em 4.9, tem-se:

$$H(s) = \frac{K K_p \tau_i s + K K_p}{s^2 + \left( \frac{1 + K K_p}{\tau} \right) s + \frac{K K_p}{\tau \tau_i}}. \quad (4.10)$$

Como o denominador da equação 4.10 é de segunda ordem, deve-se igualá-lo ao denominador da equação característica para um sistema de segunda ordem sem zeros finitos,  $\frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ , para obter os valores de  $K_p$  e  $K_i$ :

$$\begin{cases} \frac{1 + KK_p}{\tau} = 2\zeta\omega_n, & (4.11) \\ \frac{KK_p}{\tau\tau_i} = \omega_n^2. & (4.12) \end{cases}$$

Assim, os valores das constantes do controlador PI, são:

$$\begin{cases} K_p = \frac{2\zeta\omega_n\tau - 1}{K}, & (4.13) \\ \tau_i = \frac{KK_p}{\tau\omega_n^2}, & (4.14) \\ K_i = \frac{1}{\tau_i}. & (4.15) \end{cases}$$

Para determinar os valores das constantes do controlador, deseja-se que o sistema possua um percentual de *overshoot* (P.O.) de 5% e um tempo de acomodação de 0.7 s, que são as constantes comumente utilizadas na disciplina laboratório de controle I. A partir disso, utilizando a fórmula do percentual de *overshoot* da equação 2.62, é encontrado  $\zeta = 0.69011$ .

Utilizando a fórmula do tempo de acomodação, apresentada na equação 2.63, pode-se substituir o valor de  $\zeta$  encontrado anteriormente e o valor do tempo de acomodação desejado,  $ts = 0.7$  s, resultando em:

$$\omega_n = 9.5223. \quad (4.16)$$

Substituindo  $\omega_n$ ,  $\zeta$ ,  $K$  e  $\tau$ , nas equações 4.13, 4.14 e 4.15, é possível encontrar os valores das constantes do controlador PI:

$$\begin{cases} K_p = 2.5093, \\ \tau_i = 0.1075 \Rightarrow K_i = 9.3014. \end{cases} \quad (4.17)$$

Assim, o controle PI utilizado é:

$$C(s) = 2.5093 \left( 1 + \frac{1}{0.1075s} \right). \quad (4.18)$$



## 4.2.2 Implementação do controle PI

O diagrama de blocos do sistema é apresentado na figura 4.5 e é ilustrado novamente na figura 4.13.

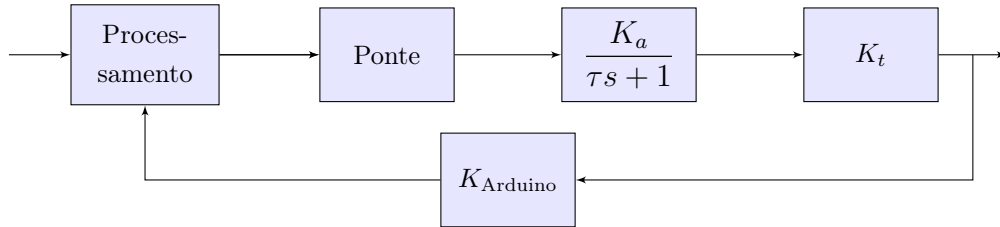


Figura 4.13: Diagrama de blocos do sistema simplificado.

Para representar a implementação do controle PI, o diagrama de blocos da figura 4.13 pode ser redesenhado, como pode ser visto na figura 4.14, em que o bloco Processamento é expandido de modo a representar as compensações, o controlador e a realimentação.

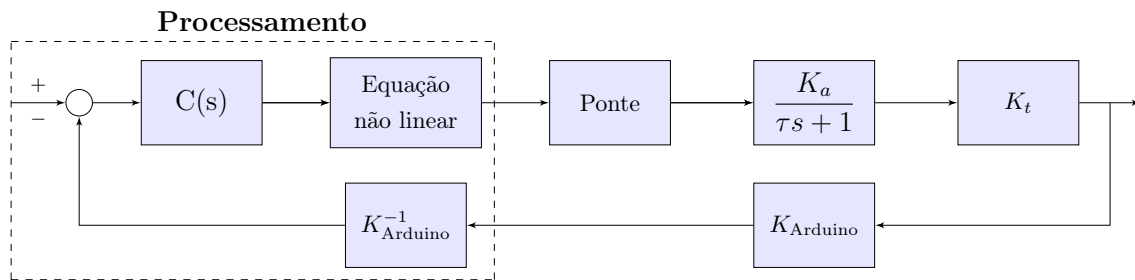


Figura 4.14: Diagrama de blocos com a realimentação, controle e compensações.

Pode-se observar que o diagrama de blocos da figura 4.14 possui a mesma função de transferência do diagrama de blocos da figura 4.12, uma vez que o bloco Equação não linear compensa o bloco Ponte e o  $K_{\text{Arduino}}^{-1}$  compensa o  $K_{\text{Arduino}}$ .

Para a implementação do controle PI na IDE do Arduino, utiliza-se a equação 4.19 do controlador, em que é necessário calcular o erro, que é a diferença entre o degrau de referência e a saída do tacômetro, e multiplicá-lo pela constante  $K_p$ , compondo assim, a parte proporcional do controle. Já na parte integral, é necessário integrar o erro em função do tempo e multiplicá-lo pelas constantes  $K_p$  e  $K_i$ , como visto no código abaixo.

$$c(t) = K_p e(t) + K_p K_i \int_0^t e(\lambda) d\lambda. \quad (4.19)$$

```
1 error = degrau - Velocidade;  
2 P = error * kP;  
3 I = I + (error * kI * kP) * deltaTime;
```

### 4.2.3 Validação do controle

Nesta seção será validado o resultado das constantes do controlador, comprovando assim que o controle de fato funciona de acordo com as especificações escolhidas, P.O. = 5% e  $t_s = 0.7$  s.

É escolhido um degrau dentro da região linear (9 - 13 V), para verificar se o motor está seguindo a referência escolhida. Também é importante o uso de perturbações externas ao motor, para verificar se o motor está rejeitando-as.

Para verificar se as especificações escolhidas estão sendo seguidas, uma simulação é feita utilizando o Simulink, mostrado na figura 4.15 e no gráfico da figura 4.16.

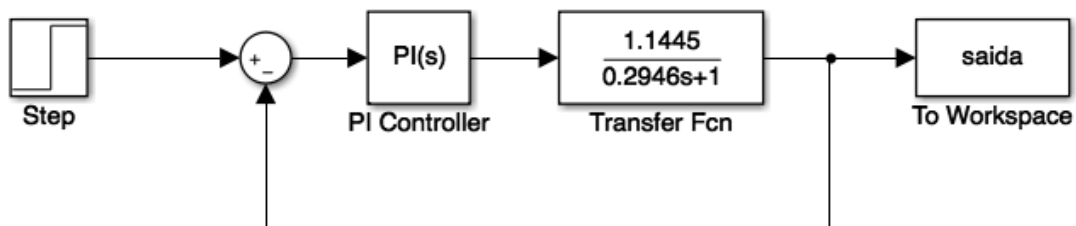


Figura 4.15: Modelo utilizado no Simulink para validação dos valores simulados.

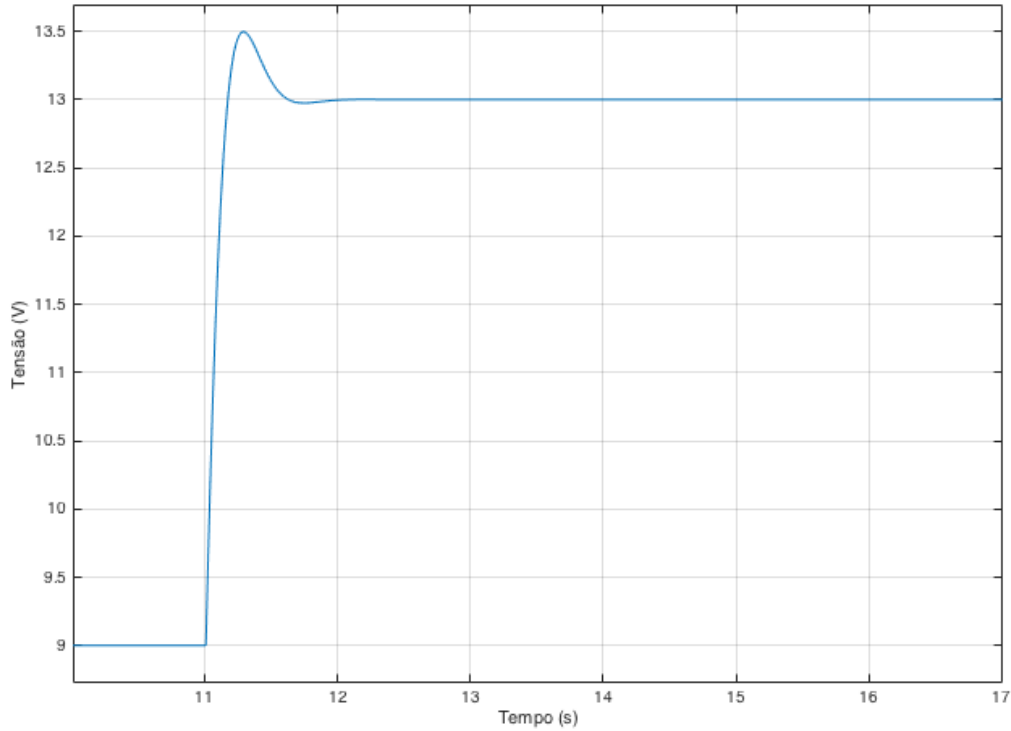


Figura 4.16: Gráfico da simulação no Simulink.

De acordo com o gráfico da figura 4.16, é possível calcular que o percentual de *overshoot* do sistema simulado foi de P.O. = 12.5% e o tempo de acomodação foi  $t_s = 0.5$  s. Logo, as constantes do controle PI ( $K_p$  e  $K_i$ ) calculadas não estão de acordo com as especificações determinadas anteriormente. Isso se deve ao fato de que quando é feita a comparação da função de transferência com a equação característica para um sistema segunda ordem sem zeros finitos para a sintonização do controle, não é levado em conta o zero da função, visto na equação 4.20.

$$H(s) = \frac{KK_p\tau_i s + KK_p}{s^2 + \left(\frac{1 + KK_p}{\tau}\right)s + \frac{KK_p}{\tau\tau_i}} = \frac{KK_p}{s^2 + \left(\frac{1 + KK_p}{\tau}\right)s + \frac{KK_p}{\tau\tau_i}} + \frac{KK_p\tau_i s}{s^2 + \left(\frac{1 + KK_p}{\tau}\right)s + \frac{KK_p}{\tau\tau_i}}. \quad (4.20)$$

Por este motivo, é de se esperar que o sinal simulado não siga exatamente as especificações, apresentando assim um *overshoot* maior e um tempo de acomodação menor, pelo fato do componente com zero acrescentar uma dinâmica não esperada à variável de controle.

Para tentar eliminar o efeito do zero ao sistema e fazer com que o controle fique dentro das especificações previamente determinadas, é adicionada uma constante ( $K_{ff}$ ) ao diagrama de blocos do controle com a planta, como visto na figura 4.17, a adição dessa constante é conhecida na literatura como controle *feedforward*.

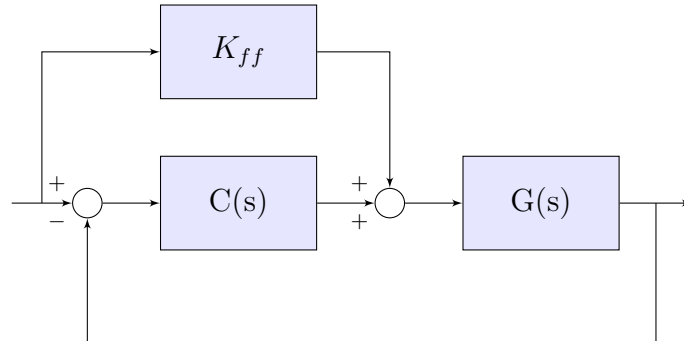


Figura 4.17: Diagrama de blocos do controle com a planta e com a constante  $K_{ff}$ .

Baseado no diagrama de blocos da figura 4.17, é possível encontrar sua função de transferência:

$$H(s) = \frac{KK_p\tau_i s \boxed{(K_p + K_{ff})} + KK_p}{s^2 + \left(\frac{1 + KK_p}{\tau}\right)s + \frac{KK_p}{\tau\tau_i}}. \quad (4.21)$$

A partir da função de transferência da equação 4.21, é possível eliminar o zero da função se o fator  $K_p + K_{ff} = 0$ . Logo, é escolhido o valor de  $K_{ff} = -K_p$ , assim, o zero não irá adicionar dinâmica à variável de controle. Como o denominador da função de transferência não foi alterado, os valores  $K_p$ ,  $K_i$  e  $\tau_i$  do controle PI continuam o mesmo.

A partir do novo diagrama de blocos, é feita uma simulação utilizando o Simulink, como visto na figura 4.18, para verificar se com a constante  $K_{ff}$  realmente foi capaz de eliminar a dinâmica do zero ao sistema, como apresentado no gráfico da figura 4.19.

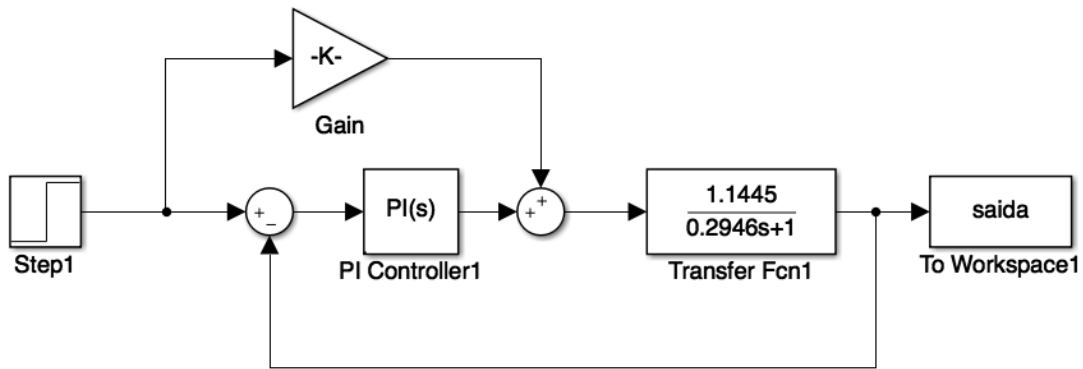


Figura 4.18: Modelo utilizado no Simulink para simulação do controle com a constante  $K_{ff}$  adicionada.

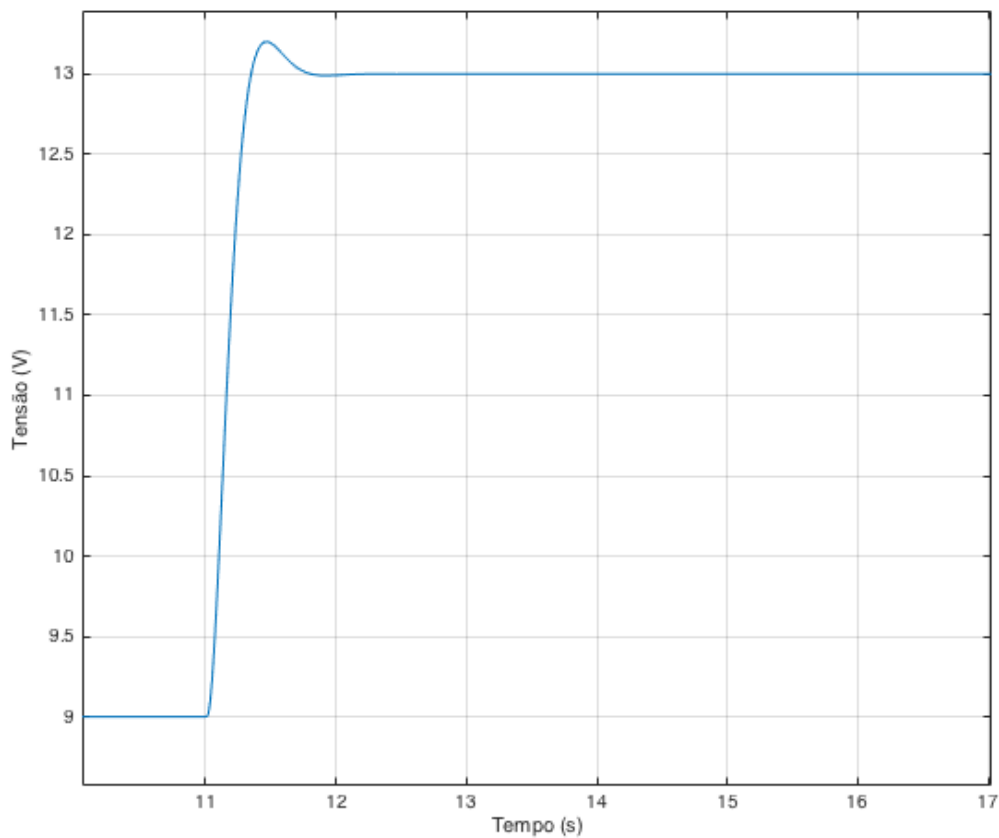


Figura 4.19: Gráfico da simulação com a constante  $K_{ff}$  adicionada.

A partir do gráfico da figura 4.19, é possível calcular o novo percentual de *overshoot* e o novo tempo de acomodação, encontrando;  $P.O = 5\%$  e  $t_s = 0.57$  s. Sendo possível confirmar que de fato a adição da constante  $K_{ff}$  foi capaz de eliminar a ação do zero, fazendo com que o controle fique dentro das especificações determinadas previamente.

Depois de todas as constantes necessárias calculadas, é escrito o último código, apêndice A.2, em que é implementado o controlador PI mais o *feedforward*, baseado no diagrama de blocos da figura 4.17, lembrando de utilizar os bolcos de compensação:  $K_{\text{Arduino}}^{-1}$  e Equação não linear.

Por fim, para validar os resultados obtidos, é feita uma comparação entre a saída simulada com a tensão de saída do tacômetro, apresentada no gráfico da figura 4.20.

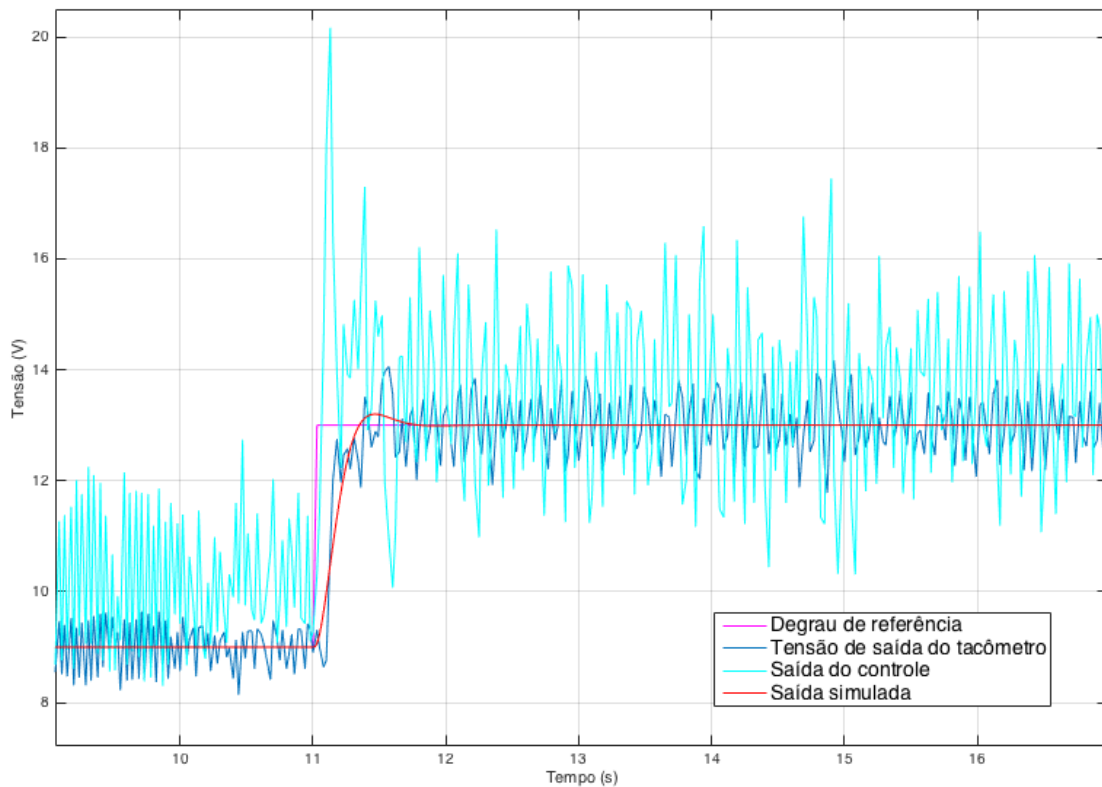


Figura 4.20: Comparação da saída simulada com a saída experimental utilizando o controle PI.

É possível observar, de acordo com o gráfico da figura 4.20, que os valores de  $K_p$  e  $K_i$  encontrados estão de acordo com o esperado, pois a tensão de saída do tacômetro (curva azul), segue o sinal de referência (curva rosa) e se aproxima da saída simulada (curva vermelha). É possível também perceber que a saída do tacômetro apresenta um percentual de *overshoot* P.O.= 10.02%, quando desconsiderado os ruídos inerentes ao próprio sistema e um tempo de acomodação de  $t_s = 0.66$  s. O valor um pouco mais elevado que o pré determinado para o percentual de *overshoot* se deve ao atraso do controle gerado pelo tempo de processamento do próprio Arduino.

O controle também foi capaz de compensar perturbações externas do tipo degrau. De acordo com a figura 4.21, é possível observar que nos intervalos aproximados de  $t = 14$  s a  $t = 18$  s e de  $t = 23.5$  s a  $t = 26$  s há duas perturbações externas, em que ambas foram compensadas pelo controle.

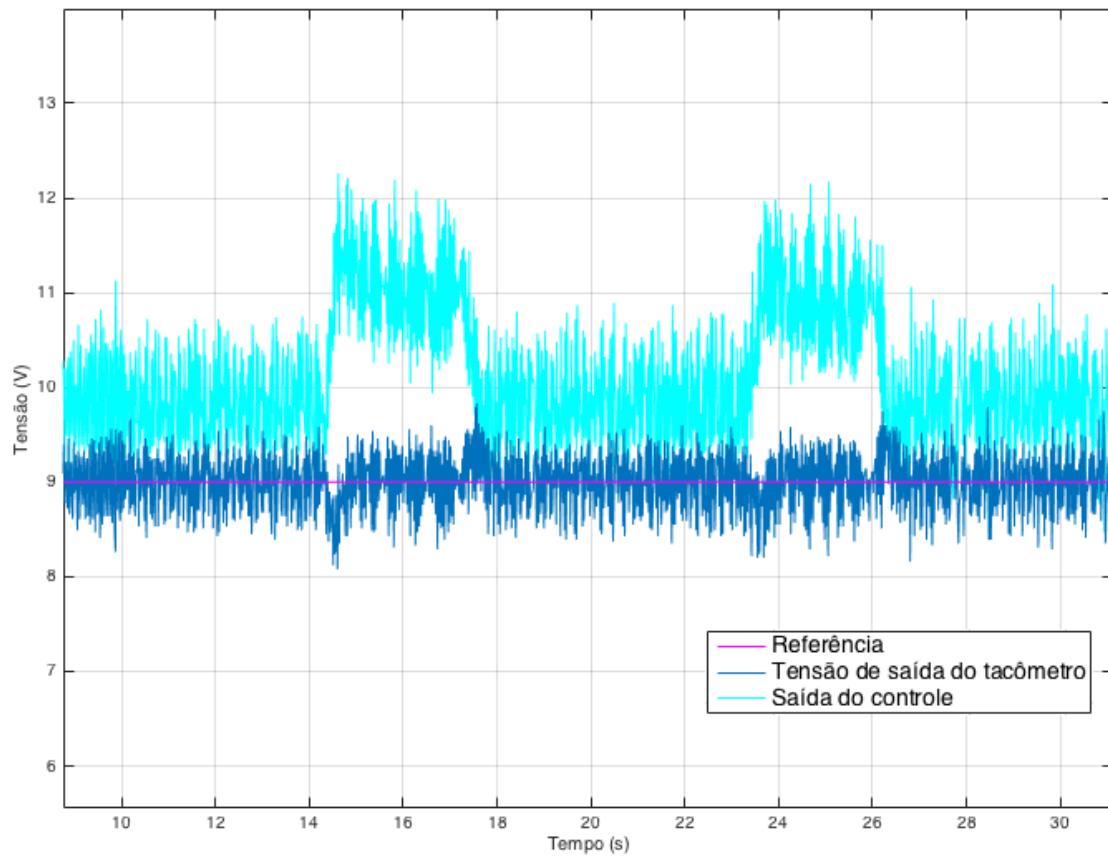


Figura 4.21: Gráfico da tensão de saída do tacômetro em que há perturbações externas do tipo degrau.

# Capítulo 5

## Conclusões

A partir dos resultados obtidos, pode-se observar que com o Arduino acoplado à ponte H é possível determinar os parâmetros do motor CC. Também foi feita uma comparação dos métodos de determinação das variáveis do motor (método dos mínimos quadrados, resposta ao degrau, método da área e método do logaritmo neperiano). Neste trabalho a modelagem feita pelo método da área, juntamente com o método dos mínimos quadrados, foram os que mais se aproximaram dos resultados experimentais, e por isso foram utilizados como modelo para o controle.

Na parte da sintonização do controlador PI, é possível perceber que os resultados não seguem o valor de *overshoot* previamente especificado. Porém, quando é adicionada a constante  $K_{ff}$  ao sistema, observa-se que o controle consegue ficar dentro das especificações de percentual de *overshoot* e de tempo de acomodação, confirmando que o controle *feedforward* foi eficaz de eliminar a dinâmica do zero do sistema. Já na parte de rejeição à perturbações do tipo degrau, o controle calculado apresentou ótimos resultados compensando-as.

Portanto, pode-se concluir que é possível controlar a velocidade do motor CC em questão com Arduino acoplado à ponte H para substituir os amplificadores de potência utilizados na disciplina laboratório de controle I, pois os resultados da modelagem do motor CC e da sintonização do controle apresentaram resultados satisfatórios.

Como trabalhos futuros, o controle pode ser feito através da análise em frequência utilizando o diagrama de Bode implementado a partir do Arduino. Com essa implementação é possível verificar se o Arduino apresenta limitações quando utilizado na análise em frequência.



# Referências Bibliográficas

- [1] BASILIO, J. C., VIANA, G. S., CARVALHO, L. K. “Apostila Laboratório de Sistemas de Controle I”. 2014.
- [2] KUO, B. C. *Automatic control systems*. Prentice Hall PTR, 1987.
- [3] NISE, N. S., DA SILVA, F. R. *Engenharia de sistemas de controle*, v. 3. LTC, 2002.
- [4] FRANKLIN, G. F., POWELL, J. D., EMAMI-NAEINI, A. *Sistemas de controle para engenharia*. Bookman Editora, 2013.
- [5] “Manual do Arduino”. <https://www.arduino.cc/en/Guide/Introduction>. Acessado em: 2018-06-12.
- [6] BLUM, J. *Explorando o Arduino*. Alta Books, 2016.
- [7] WARREN, J.-D., ADAMS, J., MOLLE, H. “Arduino for robotics”. In: *Arduino robotics*, Springer, pp. 51–82, 2011.
- [8] SEDRA, A. S., SMITH, K. C. *Microeletrônica*. Pearson Makron Books, 2005.
- [9] STRANG, G. *Introduction to linear algebra*, v. 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
- [10] ÅSTRÖM, K. J., HÄGGLUND, T. *PID controllers: theory, design, and tuning*, v. 2. Instrument society of America Research Triangle Park, NC, 1995.
- [11] OGATA, K. *Modern control engineering*. Prentice Hall Upper Saddle River, NJ, 2009.

# Apêndice A

## Códigos Utilizados

### A.1 Determinação dos parâmetros do motor

#### A.1.1 Código utilizado apenas para descobrir a região linear.

```
1 #include <DMPH.h> //biblioteca para uso da ponte H
2 DMPH motor(6, 7, 5); // (motor, motor, pino velocidade)
3 //utilizando essa biblioteca, ja e setado o valor analogico
  de HIGH e LOW para os pinos 6 e 7 do motor
4
5 int v = 255; // velocidade em PWM (0 - 255)
6
7 void setup(){
8 }
9
10 void loop() {
11 motor.move(v); //comando para fazer o motor se mover com
  velocidade v
12 delay(50);
13 }
```

## A.1.2 Código utilizado para determinar as constantes do motor.

```
1 #include <DMPH.h> //biblioteca para uso da ponte H
2 #define Sensor A0 //porta analogica que guarda os dados da
   velocidade do motor
3
4 DMPH motor(6, 7, 5); // (motor, motor, pino velocidade)
5 //utilizando essa biblioteca, ja e setado o valor analogico
   de HIGH e LOW para os pinos 6 e 7 do motor
6
7 float degrau_ = 5; //valor em tensao do degrau de
   referencia (0 - 20 V)
8 float degrau = 12; //valor em tensao do degrau de
   referencia (0 - 20 V)
9
10 float bits_ = -0.000252*pow(degrau_, 6) + 0.015023*pow(
   degrau_, 5) -0.350517*pow(degrau_, 4) + 4.108722*pow(
   degrau_, 3) -25.103539*pow(degrau_, 2) +81.769972*degrau_
   -56.998334;
11 //passando o valor de tensao para bits (0 - 255)
12
13 float bits = -0.000252*pow(degrau, 6) + 0.015023*pow(
   degrau, 5) -0.350517*pow(degrau, 4) + 4.108722*pow(
   degrau, 3) -25.103539*pow(degrau, 2) +81.769972*degrau
   -56.998334;
14 //passando o valor de tensao para bits (0 - 255)
15
16 int x = 0; //variavel que guarda a quantidade de loops
17
18 float Bits = bits_;
19 float deg = degrau_;
20
21 double tp = millis();
22
23 float velocidade = 0;
24 float Velocidade = 0;
25
26 void setup() {
```

```

27 Serial.begin(9600);
28 Serial.println("Loop_Tempo(ms)_Degrau_de_entrada(0-20V)_
    Velocidade(0-20V)");
29 }
30
31 void loop() {
32 unsigned long tempo = millis();
33 int t = millis()/1000 -tp; //contador
34
35 Velocidade = analogRead(Sensor) * (5.0/1023.0) * 4.2065;
36 //passa a velocidade de 10bits (0-1023), para tensao (0 - 5
    V) e compensa a parte do divisor de tensao (0 - 20 V)
37
38 Serial.print(x); //loop
39 Serial.print("_");
40 Serial.print(tempo); //tempo em milisegundos
41 Serial.print("_");
42 Serial.print(deg); //degrau de entrada (0 - 20 V)
43 Serial.print("_");
44 Serial.println(Velocidade); //velocidade de saida do motor
    (0 - 20 V)
45 x++;
46 delay(50); //delay para o arduino printar os dados na porta
    serial
47
48 if (t<=10){ //5 segundos com o valor do primeiro degrau
49 motor.move(Bits);
50 }
51 else if (t<=20){ //5 segundos com o valor do segundo degrau
52 Bits = bits;
53 deg = degrau;
54
55 motor.move(Bits);
56 }
57 else{
58 tp = millis()/1000;
59 Bits = bits_;
60 deg = degrau_;
61

```

```
62 }
63 if (x > 1080){ //termina o programa depois de salvar os 720
    loops
64 motor.move(0);
65 exit(0);
66 }
67 }
```

## A.2 Controlador PI

```
1 #include <DMPH.h> //biblioteca para o uso da ponte H
2 #define Sensor A0 //porta analogica que guarda os dados da
   tensao de saida do tacometro
3
4 DMPH motor(6,7,5); // (motor, motor, pino velocidade)
5 //utilizando essa biblioteca, ja e setado o valor analogico
   de HIGH e LOW para os pinos 6 e 7 do motor
6
7 double kP = 2.5093; //valor da variavel proporcional
8 double kI = 9.3014; //valor da variavel integral
9 double kff = -2.5093; //valor da variavel kff
10
11 float degrau_ =9; //valor em tensao do degrau de referencia
   (0 - 20 V)
12 float degrau =13; //valor em tensao do degrau de referencia
   (0 - 20 V)
13
14 int x = 0; //variavel que guarda a quantidade de loops
15
16 float deg = degrau_;
17
18 double tp = millis();
19 float Velocidade = 0.0;
20 float error = 0.0;
21 long lastTime = 0.0;
22 float deltaTime = 0.0;
23 float controle = 0.0;
24 float controlePwm = 0.0;
25 unsigned long tempo;
26
27 double P=0.0;
28 double I=0.0;
29 double KFF =0.0;
30 double pi=0.0;
31
32
33 void setup() {
```

```

34 Serial.begin(9600);
35 Serial.println("CLEARDATA");
36 Serial.println("Tempo(s) _Degrau_de_entrada(0-20V) _
    Velocidade(0-20V)");
37 }
38
39 void loop() {
40 unsigned long tempo = millis();
41 int t = millis()/1000 -tp; //contador
42 Velocidade = analogRead(Sensor) * (5.0/1023.0) * 4.2065;
43 //passa a velocidade de 10bits (0-1023), para tensao (0 - 5
    V) e compensa a parte do divisor de tensao (0 - 20 V)
44
45 Serial.print(tempo/1000.00); //tempo em milisegundos
46 Serial.print ("_");
47 Serial.print(deg); //degrau de entrada (0 - 20 V)
48 Serial.print ("_");
49 Serial.print(Velocidade); //velocidade de saida do motor (0
    - 20 V)
50 Serial.print ("_");
51 Serial.println(controle); //controle PWM (0 - 20 V)
52 x++;
53
54 error = deg - Velocidade;
55 deltaTime = (millis() - lastTime) / 1000.0;
56 lastTime = millis();
57
58 P = error * kP;
59 I = I + (error * kI * kP) * deltaTime;
60 KFF = kff *deg;
61
62 pi = P + I + KFF;
63 controle = pi;
64
65 controlePwm = -0.000252*pow(controle,6) + 0.015023*pow(
    controle,5) -0.350517*pow(controle,4) + 4.108722*pow(
    controle,3) -25.103539*pow(controle,2) +81.769972*
    controle -56.998334;
66 //passando o valor de tensao (0 - 20 V) para bits (0 - 255)

```

```

67
68 if (controlePwm > 255){ //para nao haver saturacao do
    controle Pwm
69 controlePwm = 255;
70 }
71 if (controlePwm <1){ //para nao haver saturacao do controle
    Pwm
72 controlePwm = 1;
73 }
74
75 if (t<=10){ //10 segundos com o valor do primeiro degrau
76 motor.move(controlePwm);
77 }
78 else if (t<= 20){ //10 segundos com o valor do segundo
    degrau
79 deg = degrau;
80 motor.move(controlePwm);
81 }
82 else {
83 tp = millis()/1000;
84 deg = degrau_;
85 }
86 if (tempo > 40000){ //termina o programa depois de 40
    segundos
87 motor.move(0);
88 exit(0);
89 }
90 }

```