Universidade Federal de Ouro Preto
Escola de Minas
CECAU - Colegiado do Curso de
Engenharia de Controle e Automação

Douglas Xavier de Souza

Projeto e implementação de um controlador de velocidade de um motor CC utilizando inteligência artificial

Monografia de Graduação

Ouro Preto, 2023

Douglas Xavier de Souza

Projeto e implementação de um controlador de velocidade de um motor CC utilizando inteligência artificial

Trabalho apresentado ao Colegiado do Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para a obtenção do Grau de Engenheira(o) de Controle e Automação.

Universidade Federal de Ouro Preto

Orientador: Profa. Adrielle de C. Santana, Dr.

Ouro Preto
2023

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

S729p Souza, Douglas Xavier de.
Projeto e implementação de um controlador de velocidade de um motor CC utilizando inteligência artificial. [manuscrito] / Douglas Xavier de Souza. - 2023.
65 f.: il.: color., gráf., tab..

Orientadora: Profa. Dra. Adrielle de Carvalho Santana.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Controle e Automação .

1. Motor de Corrente Contínua (Motor CC). 2. Inteligência artificial. 3. Redes neurais (Computação). 4. Microcontroladores. 5. Velocidade - Controle. I. Santana, Adrielle de Carvalho. II. Universidade Federal de Ouro Preto. III. Título.

CDU 681.3

Bibliotecário(a) Responsável: Maristela Sanches Lima Mesquita - CRB-1716

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
ESCOLA DE MINAS
DEPARTAMENTO DE ENGENHARIA CONTROLE E AUTOMACAO

FOLHA DE APROVAÇÃO

Douglas Xavier de Souza

Projeto e Implementação de um Controlador de Velocidade de um Motor CC Utilizando Inteligência Artificial

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como requisito parcial para obtenção do título de bacharel em Engenharia de Controle e Automação

Aprovada em 06 de março de 2023

Membros da banca

Dra. Adrielle de Carvalho Santana - Orientadora (Universidade Federal de Ouro Preto)
Dr. Agnaldo José da Rocha Reis - Convidado (Universidade Federal de Ouro Preto)
Dr. Paulo Marcos de Barros Monteiro - Convidado (Universidade Federal de Ouro Preto)

Adrielle de Carvalho Santana, orientadora do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 08/03/2023

Documento assinado eletronicamente por **Adrielle de Carvalho Santana**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 08/03/2023, às 10:37, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0484332** e o código CRC **16555C1A**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35402-163
Telefone: 3135591533 - www.ufop.br

Agradecimentos

Agradeço a Deus por ser o guia em todos os momentos e por sempre estar presente em minha vida. Agradeço à UFOP e a Escola de Minas por me proporcionar uma excelente formação acadêmica e profissional.

Quero agradecer especialmente à minha orientadora Adrielle por toda a paciência, dedicação e apoio incansável durante todo o processo. A todos os professores e mestres que participaram desta jornada, agradeço pelas valiosas contribuições e ensinamentos.

Gostaria de agradecer aos meus amigos Anderson, Gabriela e Wesley por estarem sempre presentes e me apoiando durante todo o percurso. Além disso, gostaria de agradecer a todos os que fizeram parte desta jornada, por terem contribuído de alguma forma para a conclusão dessa etapa.

Por fim, gostaria de agradecer a minha família, especialmente a meu pai, Altamir, minha mãe, Lilian, e a minha namorada, Wanuza, por todo o amor e apoio incondicional ao longo de toda a minha vida e, especialmente, durante a realização deste trabalho. Agradeço de coração a todos vocês por terem feito parte desta história.

"A persistência é o menor caminho do êxito" (Charles Chaplin)

Resumo

O motor elétrico de corrente contínua (CC) é importante em diversos setores da indústria, portanto, controlar sua velocidade é essencial para determinados processos industriais. O controle clássico muitas vezes consegue controlar inúmeras plantas, porém quando a complexidade do sistema aumenta os mesmos não são tão eficazes. Com isso, métodos de controle modernos vêm ganhando cada vez mais espaço por conseguir controlar plantas com uma complexidade mais alta. Assim, a inteligência artificial é uma grande aliada para esses métodos. Este trabalho tem como objetivo projetar um circuito de acionamento e controle de um motor CC por meio de um microcontrolador, aplicar técnicas de controle clássico e moderno com o uso de redes neurais e, por fim, analisar e comparar os resultados de tais métodos mostrando pontos positivos e negativos dos mesmos. Os objetivos foram alcançados e foi possível evidenciar pontos positivos e negativos dos 3 controladores utilizados: o PID (proporcional, integral e derivativo), o controlador por modelo preditivo (do inglês *Model predictive control* (MPC)) e o controlador proposto baseado em rede neural. O controlador PID apresentou bons resultados, estabilizando-se rapidamente em todos os cenários, mas com um *overshoot* médio de 18%, além disso, o sinal de controle deste controlador sofreu muito ruído em consequência do sinal de erro do sistema. O MPC também obteve bons resultados, porém em alguns casos apresentou *overshoot* e tempo de estabilização maior, sendo assim, sua principal vantagem é conseguir controlar plantas mais complexas, inserir limites na variável de controle e garantir um sinal de controle menos ruidoso. Em contrapartida, sua aplicação requer um custo computacional mais elevado. O controlador baseado em redes neurais obteve resultados menos satisfatórios, com erro e tempo de estabilização maiores. No entanto, demonstrou ser capaz de controlar plantas mais complexas com um custo computacional menor que controladores robustos, evidenciando a relevância de se continuar investigando seu desempenho em trabalhos futuros.

Palavras-chaves: motor de corrente contínua; controle clássico; controle moderno; inteligência artificial; redes neurais; microcontrolador; controle de velocidade.

Abstract

The direct current motors are very important in many sectors of the industry, therefore, it's essencial to control your speed in determinated industrial processes. The classic control often manages to control countless plants but when the complexity of the system increases they are not as effective. With this being said, modern control methods are gaining more and more space for being able to control plants with elevated complexity. Consequently, the artificial intelligence is a great ally to this methods. This work aims to design a DC motor drive and control circuit through a microcontroller, apply modern control techniques with the use of neural networks, classical control and, in the end, analyze and compare the results of such methods comparing positive and negative points of them. The objectives were somehow achieved and it was possible to highlight the positives and negatives of the 3 controllers used: the PID, the MPC and the standard controller based on a neural network. The PID controller showed good results, quickly stabilizing in all scenarios, but with an average overshoot of 18%, furthermore, the control signal of this controller suffered a lot of noise as a result of the system error signal. The MPC also obtained good results, but in some cases it presented overshoot and longer stabilization time, therefore, its main advantage is being able to control more complex plants, inserting limits in the control variable and ensure a less noisy control signal, on the other hand, its application requires a higher computational cost. The controller based on neural networks obtained less satisfactory results, with higher error and stabilization time. However, it proved to be able to control more complex plants with a lower computational cost than robust controllers, evidencing the relevance of continuing to investigate its performance in future works.

Key-words: direct current motor; classic control; modern control; artificial intelligence; neural networks; microcontroller; speed control.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

CC	Corrente contínua
PID	Proporcional, integral e derivativo
MPC	*Model predictive control*
SISO	*Single-input and single-output*
MIMO	*Multiple input multiple output*
PWM	*Pulse Width Modulation*
CI	Circuito integrado
PI	Proporcional e integral
NMPC	*Nonlinear Model Predictive Control*
MLP	*Multi Layered Perceptron*
MSE	*Mean squared error*

Sumário

1 Introdução

1.1 Contextualização

O controle de sistemas, sejam eles físicos, sociais, econômicos ou biológicos, sempre foi de interesse para o ser humano, pois controlando esses sistemas é possível obter uma saída desejada que possibilitará ou facilitará a execução de um processo. Com o avanço tecnológico, aumento do número de máquinas presentes em um processo e com a criação de novos equipamentos, se faz cada vez mais necessário um controle preciso desses processos, a fim de evitar erros que podem ser críticos para a atividade.

Métodos de controle clássicos são excelentes para controle de sistemas lineares, invariantes no tempo e com uma única entrada e única saída (do inglês *single-input and single-output* - SISO). No entanto, muita das vezes tais métodos não conseguem controlar uma planta com uma complexidade maior como, por exemplo, sistemas representados por equações diferenciais, cujo os coeficientes são funções de tempo, ou seja, variantes no tempo, como explicado em Ogata (2010), ou sistemas com múltiplas entradas e saídas (do inglês *multiple input multiple output* - MIMO). Outro problema de alguns controladores clássicos, ressaltado em Santos Neto e Gomes (2010), é sua sintonia, que, em grande parte das vezes, são implementadas inadequadamente.

Além de sistemas lineares, existem também os não lineares que atraem grande interesse da academia. Contudo, para que esses sejam controlados por métodos de controle clássicos seus respectivos modelos devem ser linearizados, o que resulta em uma perda de informações que podem ser importantes para processos que têm uma alta exigência de precisão (DRUMMOND; OLIVEIRA; BAUCHSPIESS, 1999).

Nesse sentido, o controle inteligente vem ganhando cada vez mais espaço na academia e nas indústrias. Métodos de controle que utilizam técnicas de inteligência computacional têm grande vantagem quando comparados ao controle clássico uma vez que conseguem controlar sistemas lineares e não-lineares, variantes e invariantes no tempo além de conseguir estabelecer restrições que, do ponto de vista prático, sempre existem.

Na tentativa de desenvolver um controle mais versátil e robusto pesquisadores utilizam diferentes técnicas de inteligência computacional, entretanto, as mais utilizadas são as redes neurais e a lógica *fuzzy*. Além disso, existem diversos trabalhos que utilizam os controladores híbridos que mesclam duas ou mais técnicas com a finalidade de diminuir as desvantagens de se utilizar uma em específico (BILOBROVEC, 2004). Em alguns casos, utiliza-se também essas técnicas para conseguir uma sintonia de parâmetros em sistemas de controle clássico como o controlador proporcional, integral e derivativo (PID) como

feito em Coelho et al. (2019).

Uma rede neural tem como objetivo chegar a uma saída desejada a partir de entradas escolhidas previamente. Ela faz isso simulando artificialmente o funcionamento das redes de neurônios dos cérebros humanos. Tais redes são formadas basicamente por uma série de neurônios artificiais distribuídos em camadas entre a entrada e saída da rede. A topologia da rede depende, principalmente, do problema que se deseja solucionar. Esses neurônios, resumidamente, são compostos por um conjunto de entradas, um núcleo onde ocorre o processamento das informações de entrada e uma saída que expõe o resultado do processamento (SILVA FERREIRA; CAVALCANTI; ALSINA, 2017). Uma operação importante tanto para as redes neurais como para o cérebro humano é o processo de aprendizado que, no caso das redes artificiais, é chamado de treinamento. Nesse treinamento busca-se atingir pesos para as entradas dos neurônios que garantam o resultado esperado na saída a partir de dados previamente calculados.

Na indústria, a maioria dos processos utilizam como atuador um motor elétrico. O motor de corrente contínua (CC) é, sem dúvidas, muito utilizado nesse contexto. Em muitos processos é necessário se controlar a velocidade de tais motores e para realizar isso é necessário um circuito de acionamento, que usualmente é um circuito *Buck*, e um circuito de controle que é responsável por controlar a tensão a ser enviada para o motor por meio de um sinal de Modulação de Largura de Pulso (do inglês *Pulse Width Modulation* - PWM). Além disso, para fechar a malha de controle utiliza-se como sensor um encoder que é um sensor que tem a capacidade de transformar a posição do eixo em um sinal elétrico digital e a partir desse sinal é possível inferir a posição e velocidade do eixo.

Entendendo os aspectos já citados, é possível perceber que o controle clássico consegue controlar diversas plantas com certa eficácia, porém em algumas outras plantas se faz necessário o uso de outros métodos de controle mais robustos. Visto também a importância do uso de motores na indústria e do controle dos mesmos, este trabalho tem o intuito de projetar um circuito de controle e acionamento de um motor CC e aplicar métodos de controle para controlar sua velocidade e depois compará-los.

1.2 Objetivos

1.2.1 Objetivos Gerais

Projetar e implementar um sistema de acionamento e controle de um motor de corrente contínua e aplicar técnicas de controle clássico e de controle inteligente para controlar sua velocidade, realizando um comparativo entre os desempenhos de cada técnica.

1.2.2 Objetivos Específicos

- Desenvolver um circuito de acionamento de um motor de corrente contínua além de um circuito microcontrolado para implementar o controle e o sensoriamento da velocidade do motor;
- Levantar os parâmetros do motor CC com a finalidade de obter sua função de transferência;
- Aplicar uma técnica de controle inteligente para controlar a velocidade do motor;
- Aplicar um método de controle clássico como o PID para o controle da velocidade do motor;
- Analisar e comparar os resultados das técnicas utilizadas trazendo, no fim, pontos positivos e negativos de cada uma.

1.3 Estrutura do trabalho

Este trabalho está dividido em cinco principais capítulos: o primeiro é a introdução que apresenta a contextualização, justificativa e objetivos do estudo; o segundo é a revisão bibliográfica que apresenta a fundamentação teórica da monografia expondo o embasamento que é necessário para construir a ideia do trabalho; o terceiro capítulo apresenta com detalhes o que foi realizado durante a execução do trabalho com suas respectivas ideias e justificativas; o quarto tem a função de expor todos os resultados obtidos por meio do desenvolvimento explicado no capítulo anterior bem como realizar algumas análises dos mesmos; o quinto e último exibe as conclusões obtidas em todo o decorrer do estudo podendo também sugerir novas vertentes para próximas pesquisas.

2 Referencial teórico

Este capítulo apresenta uma revisão da literatura sobre métodos de controle que utilizam a inteligência computacional, bem como apresenta os conceitos necessários para a compreensão do desenvolvimento deste trabalho.

2.1 Motores elétricos

Um motor elétrico é um instrumento capaz de transformar energia elétrica em energia cinética ou mecânica. Essa energia final é muito importante para indústrias e aparelhos elétricos e eletrônicos pois pode ser utilizada em diversas atividades tais como transporte de cargas, movimentação de produtos, processamento de materiais, entre outras. Outro grande fator que influencia no grande sucesso dos motores elétricos é a energia utilizada para seu acionamento. A energia elétrica é de fácil acesso e transmissão, ademais, corresponde a 43,4% do total da demanda energética brasileira sendo a mais demandada do país (PEREIRA et al., 2005).

2.1.1 Motores de corrente contínua

Motores de corrente contínua são altamente aplicados em sistemas que necessitam de energia mecânica para diversas finalidades. Isso se dá pois eles são baratos, fáceis de controlar e bastante versáteis de modo a funcionar tanto em grandes dispositivos mas, principalmente, por funcionar em pequenos equipamentos que, na maioria das vezes, possuem uma fonte de energia elétrica contínua tais como pilhas e baterias. Dentre os vários exemplos de aplicação estão: telefones celulares (vibracall), drones, trens elétricos, elevadores e prensas.

Para entender o funcionamento de uma máquina de corrente contínua, utiliza-se a chamada máquina elementar que se pode ver na figura 1. Nela é possível ver um campo magnético constante provido por um imã permanente (que fica no estator). Dentro desse campo está o rotor, composto por uma única espira na qual, ao se aplicar uma diferença de potencial, produz uma força F resultando em uma velocidade angular ω. Essa espira é conectada a terminais elétricos (comutador) em contato com as chamadas escovas que são responsáveis por alternar a diferença de potencial na espira, mantendo o sentido da força F e da velocidade ω constante (ZOGHEIB, 2022).

Entretanto, um motor real é composto por um número maior de espiras que traz benefícios. Segundo Zogheib (2022):

> (...) à medida que é adicionado um número maior de espiras e de caminhos em paralelo, mais suave será a troca de polaridade entre os

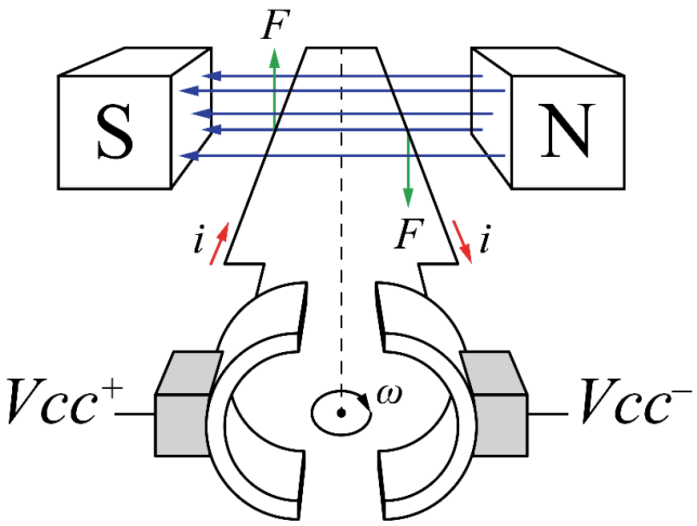

Figura 1 – Máquina de corrente contínua elementar composta por um imã permanente e uma única espira em seu rotor. Fonte: Zogheib (2022).

> terminais e, assim, mais uniforme se apresentará o conjugado útil da máquina.(...)

Sempre é importante que a relação entre a entrada e saída, também conhecida como função de transferência, de um sistema em malha aberta ou fechada seja conhecida, pois, por meio dela, é possível realizar diversas análises e simulações que ajudam a projetar um controlador adequado ao sistema a ser controlado. Logo, é conveniente que se conheça a função de transferência de um motor de corrente contínua ou alguma que evidencie as suas características principais.

Para a modelagem do sistema existem diversos métodos matemáticos e experimentais. Em Carvalho et al. (2015) são utilizados métodos matemáticos e a partir dos parâmetros físicos do motor disponibilizados pela fabricante foi possível modelar o sistema. Outra possível abordagem seria a utilização de experimentos e cálculos matemáticos para a obtenção dos parâmetros necessários para a modelagem. Outro caminho é alcançar a modelagem do sistema por meio de uma modelagem caixa cinza, utilizando métodos empíricos.

No trabalho Reck (2018), o autor chega a conclusão que um modelo de primeira ordem consegue se aproximar da saída de velocidade angular em um motor CC quando os parâmetros são determinados experimentalmente. Isso, somado ao fato de que para obter um modelo de segunda ordem de um motor CC são necessários instrumentos de alta qualidade e custo, por exemplo um osciloscópio, contribuiu para a utilização do método da curva de reação.

O método da curva de reação proposto por Ziegler, Nichols et al. (1942) é um bom

exemplo de modelagem caixa cinza. Em resumo, o método consiste em aplicar um degrau com amplitude de 10% a 20% do fundo de escala ao sistema em malha aberta e a partir da curva de resposta obtida, traçar uma reta tangente ao seu ponto de inflexão, para então, obter os parâmetros atraso de transporte (L), constante de tempo (T) e ganho (K) conforme mostrado na figura 2. Esse método aproxima o sistema da planta a um sistema de primeira ordem com atraso de transporte que tem a função de transferência descrita na equação 2.1.

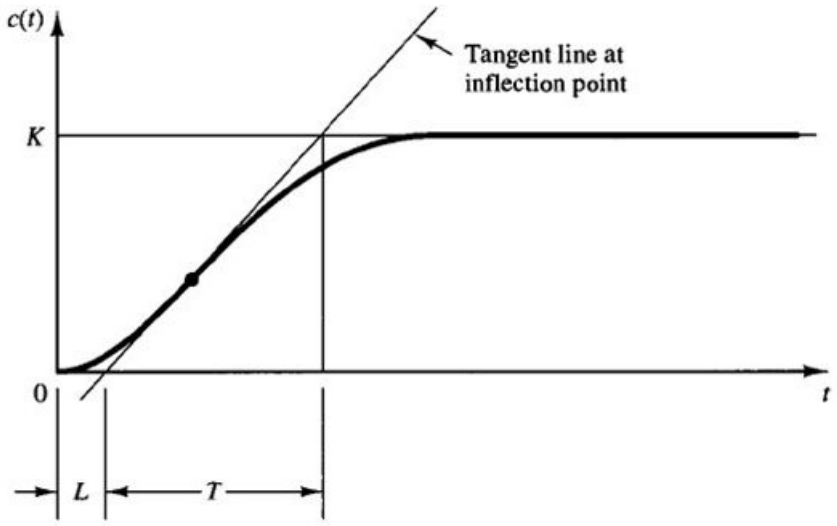

Figura 2 – Método da curva de reação para modelagem do sistema. Fonte: Universidade Gama Filho.

$$G(s) = \frac{Ke^{(-Ls)}}{Ts + 1} \tag{2.1}$$

Neste trabalho será utilizado um mini motor da marca Pololu com tensão nominal de 6V acoplado a um encoder e uma caixa de redução conforme a figura 3. O encoder também é da marca Pololu e tem uma roda com 5 dentes o que gera 20 contagens por revolução do eixo, além disso, ele trabalha com uma tensão nominal de 5V (figuras 4 e 5). A caixa de redução foi utilizada somente para fornecer uma certa resistência ao motor, sendo assim, seu fator de redução é indiferente.

2.2 Microcontroladores

Um microcontrolador pode ser definido facilmente como um componente eletrônico encapsulado em um único circuito integrado (CI) capaz de executar processos lógicos e

[1] https://www.pololu.com/product/2590

Figura 3 – Mini motor Pololu utilizado no trabalho. Fonte: Autoria própria.

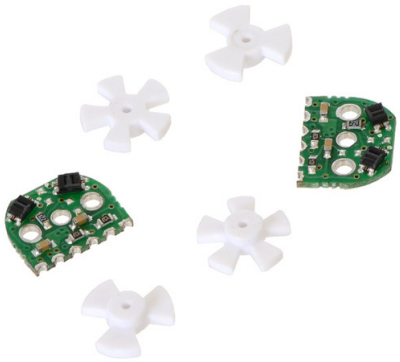

Figura 4 – Encoder óptico utilizado no trabalho. Fonte: Pololu[1].

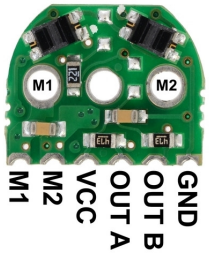

Figura 5 – Pinos do encoder óptico utilizado no trabalho. Fonte: Pololu[1].

operações matemáticas (TORGA, 2016). Os microcontroladores por serem de baixo custo e muito eficazes, são uma excelente opção para programar e controlar aparelhos eletrônicos o que torna sua utilização ainda mais popular. Para serem utilizados devem vir acompanhados de uma placa de circuito impresso que contenha todos os componentes necessários para seu funcionamento ou para maximizar suas próprias funcionalidades.

Placas microcontroladas com código aberto (do inglês *open source*) tornaram-se muito popular nos últimos anos principalmente por causa da plataforma Arduino que junta o *hardware* com um ambiente de desenvolvimento integrado (IDE) bem amigável. Em vista disso, vieram diversas empresas e várias placas com o mesmo objetivo tais como ESP32, Arduino Nano, Arduino UNO, ESP8266 entre diversas outras.

O arduino UNO é uma placa com todos os componentes necessários para o microcontrolador da empresa ATMEL chamado ATmega328P. Trata-se de um sistema de baixo custo e baixa tensão de funcionamento com diversos recursos, sendo estes suficientes para a proposta deste trabalho. A faixa de tensão de operação desse chip é de 2,7V a 5,5V, no entanto, a tensão de operação utilizada na placa é de 5V. Na figura 6 está a descrição dos pinos dessa placa.

2.3 Sistemas de controle

Um sistema pode ser definido de várias maneiras a depender do ponto de vista. De uma maneira geral, um sistema pode ser definido como uma combinação de componentes que atuam juntos para uma finalidade. Do ponto de vista de um engenheiro de controle, um sistema a controlar pode ser um processo físico, parte de um equipamento, parte de um processo com o objetivo de realizar determinado procedimento. Como exemplo podem ser citados sistemas para controlar nível de um líquido, temperatura de um ambiente, posição de um braço robótico, entre diversos outros.

Um sistema de controle pode ser de malha aberta ou de malha fechada. No primeiro, o sinal de saída não tem inflência sobre a ação de controle do sistema e pode ser representado pelo diagrama de blocos mostrado na figura 7. Já o sistema de controle em malha fechada ou com realimentação deve ter uma relação de comparação entre o sinal de saída e o sinal desejado (*setpoint*) por meio de uma diferença, que é o erro, e pode ser representado com o diagrama de blocos exposto na figura 8 (OGATA, 2010). Um sistema em malha fechada, como foi dito em Oliveira, Vargas et al. (1997), nada mais é do que um sistema em malha aberta acrescido de uma realimentação negativa e um comparador. Essa realimentação fornece o valor atual da variável controlada e isso dá ao sistema maior estabilidade e a capacidade de ser regulado mesmo na presença de pertubações ou mudanças em suas características.

[1] https://store.arduino.cc/products/arduino-uno-rev3

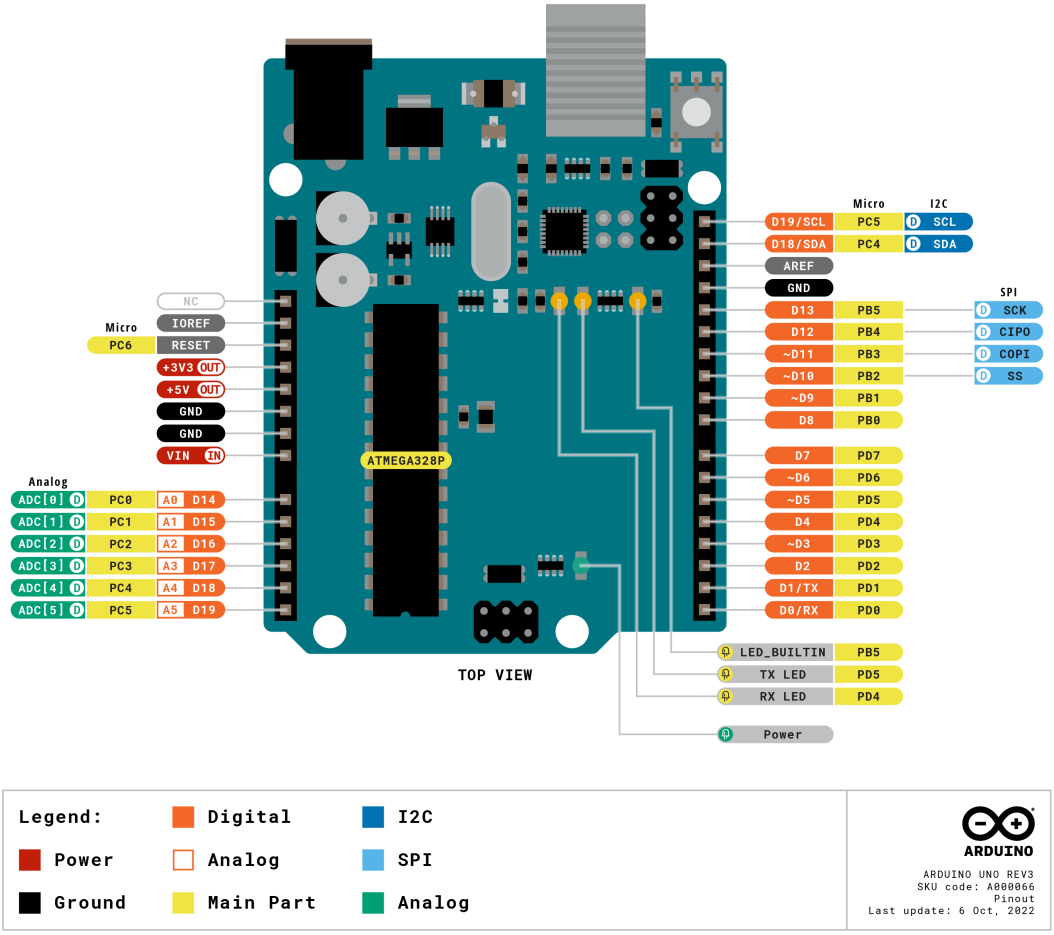

Figura 6 – Descrição dos pinos do microcontrolador Arduino UNO. Fonte: Arduino[1].

Figura 7 – Diagramas de blocos de um sistema em malha aberta ou sem realimentação. Fonte: Autoria própria.

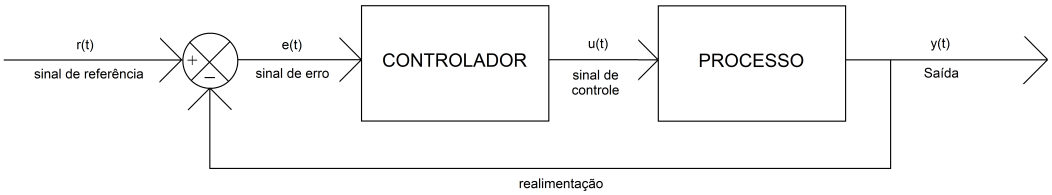

Figura 8 – Diagramas de blocos de um sistema em malha fechada ou com realimentação. Fonte: Autoria própria.

Geralmente, para sistemas que são mais simples e não necessitam de tanta precisão é utilizado um sistema em malha aberta, por exemplo o forno de um fogão a gás em que se coloca a temperatura desejada e ele mantém uma válvula com determinada abertura para chegar àquela temperatura. De maneira geral, nesse exemplo, a temperatura ficará um pouco acima ou um pouco abaixo do desejado, um dos motivos é que, nesse caso, não é levada em consideração a temperatura ambiente que tem influência direta na temperatura interna do forno. Porém, no final essa pequena diferença entre o valor real de temperatura e o valor desejado não faz diferença para o alimento preparado.

Todavia, na grande maioria dos casos, os sistemas a serem controlados exigem certa precisão e buscam o menor erro possível, por isso, são utilizados os sistemas em malha fechada ou com realimentação que garantem maior estabilidade e precisão. Um exemplo simples e bem comum é uma geladeira onde o *setpoint* é ajustado e ela ativa o compressor em plena potência para o resfriar seu compartimento. Assim que ela atinge a temperatura ajustada, ou um pouco menos, o compressor é desligado fazendo com que a temperatura em seu interior aumente. Assim que a temperatura exceda a desejada, ou um pouco mais, o compressor é ligado novamente e esse procedimento se repete ininterruptamente. Para sistemas assim, com realimentação, é necessário um sensor que evidencie ao sistema o valor atual da variável controlada para que assim seja feita uma ação de controle. No caso da geladeira, esse sensor pode ser um termistor (a depender do modelo) que determine a ativação ou não do compressor.

A grande vantagem de um sistema de controle em malha fechada é a capacidade de se adaptar a ruídos ou variações internas de modo a manter o sinal de saída desejado. Isso acontece por causa de sua realimentação. Um sistema em malha aberta não executa a tarefa desejada se houver interferências, ou seja, na prática, esse sistema só é utilizado se a relação entre a entrada e saída for conhecida e não houver nenhum distúrbio interno ou externo (OGATA, 2010). Nesse sentido, o sistema de controle em malha fechada é o mais adequado para a execução desse trabalho, uma vez que em um sistema feito para controlar velocidade de um motor CC não são conhecidas, exatamente, a relação entre a entrada e saída. Além disso, é necessário que o erro seja mínimo (a depender do objetivo)

e, principalmente, o sistema deve ser relativamente insensível a distúrbios (aumento de carga por exemplo) e a variações internas nos parâmetros do sistema. Haja vista todos esses fatores, o sistema em malha fechada vai apresentar um desempenho satisfatório.

2.4 Controlador proporcional, integral e derivativo (PID)

Os controladores PID e suas variações são mais utilizados na indústria e, segundo (OGATA, 2010), o motivo está na aplicabilidade desse controlador na maioria dos sistemas de controle. Outro grande fator se dá no fato de que tais controladores são os mais úteis quando o modelo matemático do sistema é desconhecido resultando em um controle satisfatório.

A ação proporcional de um controlador é basicamente uma amplificação do sinal de erro de modo que quanto maior o erro maior será o sinal de controle aplicado a planta. Sendo assim é importante ressaltar que se utilizado sozinho o controlador proporcional produz um erro residual permanente. A componente integral é responsável por adicionar um polo na origem da função de transferência do controlador o que elimina o erro estacionário e, em contrapartida, aumenta o tempo de acomodação. A ação derivativa responde a taxa de variação do erro resultando em um controlador com maior sensibilidade, no entanto, tal ação pode acentuar o ruído em altas frequências Lourenço (1997). Com a figura 20 é possível ver o exemplo de uma malha de controle com um controlador PID.

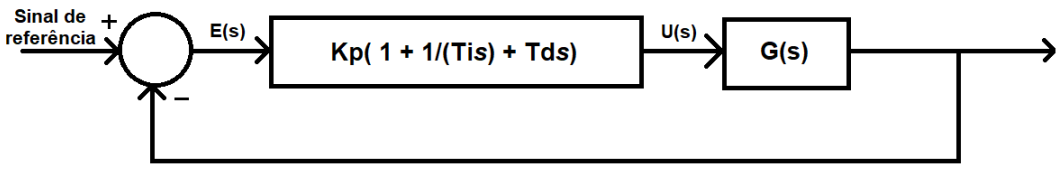

Figura 9 – Exemplo de uma malha de controle com um controlador PID. Adaptado de Lourenço (1997).

Onde Kp é o ganho proporcional, Ti é o tempo integrativo, Td é o tempo derivativo, E é o erro, U é o sinal de controle e G é a função de transferência do sistema.

Kp é a constante que multiplica o erro atuante no sistema. Ti é o tempo para que a saída do integrador atinja o valor Kp para uma entrada unitária, sendo assim $Ki = Kp/Ti$ onde Ki(ganho integral) é a constante que multiplica a integral do erro do sistema. Td é o intervalo de tempo que a ação de controle derivativa antecede a ação de controle proporcional, sendo assim $Kd = Kp * Td$ onde Kd (ganho derivativo) é a constante que multiplica a derivada do erro no sistema (ANGELICO; SCALASSARA; VARGAS, 2023).

Na indústria muitas vezes só o controlador proporcional-integral(PI) é utilizado. A parte derivativa não é utilizada devido as dificuldades normalmente encontradas em sua

utilização em aplicações onde os sinais podem conter ruído de medição (OROZCO; RUIZ, 2003).

Limites das variáveis de controle podem ocorrer em variados sistemas e provocar uma piora no desempenho do sistema que não foi prevista durante o projeto do controlador. Em Bohn e Atherton (1995) os autores discorrem sobre o efeito *windup*. No trabalho, os autores dizem que quando há saturação no atuador, incrementos posteriores no sinal de controle não contribuirão para uma resposta mais rápida do sistema. Logo, a continuação da integração do erro do sistema faz com que o termo integral adquira valores elevados sem qualquer efeito sobre a saída da planta. Com isso, o erro deve ter sinal negativo durante um longo intervalo de tempo para que o termo integral possa ser trazido de volta ao estado estacionário, o que ocasiona um elevado sobre-sinal e um tempo de acomodação relativamente longo.

Para conseguir um desempenho satisfatório e reduzir o efeito citado anteriormente, são desenvolvidas diversas técnicas chamadas *anti-windup*. Em Neto (2005) o autor estuda diferentes técnicas tais como: *Back Calculation*, técnica condicional, técnica da integração condicional, técnica proposta por Visioli, integração limitada e *Feedforward*. A integração condicional consiste em manter a saída da parte integrativa constante enquanto o sinal de controle estiver saturado, caso contrário, essa saída será incrementada normalmente.

Para que o controlador PID funcione adequadamente é necessário realizar sua sintonia. Existem diversas técnicas para isso, desde métodos matemáticos até programas de computador. Em Ziegler, Nichols et al. (1942) os autores chegam em contas matemáticas para calcular os parâmetros do controlador a partir do modelo de primeira ordem do sistema e obtêm resultados satisfatórios. Esse é o método mais utilizado e conhecido. Outro exemplo de método de sintonia é a função pidtune do *software* MATLAB descrita em The MathWorks (2023d) que projeta um controlador PID ajustando seus parâmetros para equilibrar desempenho (tempo de resposta) e robustez (margens de estabilidade) garantindo a estabilidade do sistema.

É válido ressaltar que os valores obtidos nesses métodos não são necessariamente valores ótimos dado que isso depende principalmente do foco do controlador para alcançar o objetivo geral da planta. Sendo assim, depois de obter os valores dos parâmetros por meio de métodos, pode ser feito um ajuste fino a fim de atingir um desempenho desejado. Em Kushwah e Patra (2014), por exemplo, o autor sugere outro método de sintonia a partir do método exposto em Ziegler, Nichols et al. (1942).

2.5 Controlador por modelo preditivo

O controle preditivo baseado em modelo, ou *Model predictive control* (MPC), é um modelo estabelecido para controle restrito. Segundo Carvalho et al. (2015), os controladores MPC

são compostos por uma classe de algoritmos de controle que procuram atingir um sinal de controle ótimo por meio da maximização ou minimização de uma função objetivo baseada no modelo do sistema a ser controlado. Assim, o MPC busca seguir o *setpoint* estabelecido calculando ações para as variáveis manipuladas. Na figura 10 é possível visualizar a estrutura básica de um MPC.

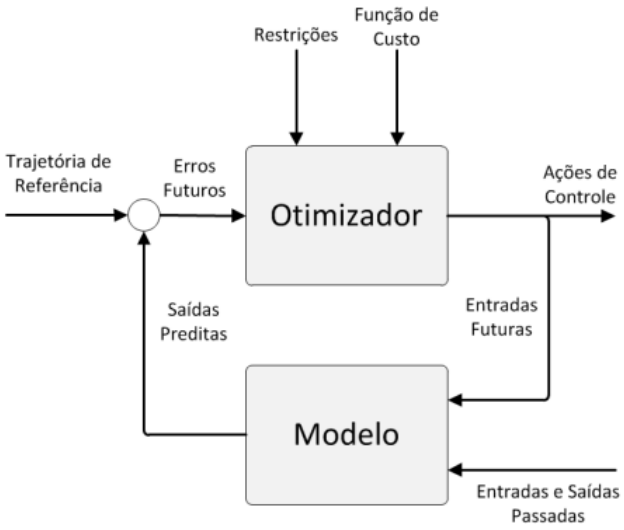

Figura 10 – Estrutura básica de um controlador por modelo preditivo. Fonte: Reis (2018).

Inicialmente os MPC foram criados para controlar processos industriais, porém, hoje eles são empregados em áreas distintas. Isso por causa da sua vasta aplicabilidade. Ele pode ser aplicado em sistemas multivariáveis, não lineares, com restrições de variáveis, entre outros; e sempre obtendo bons resultados (CARVALHO et al., 2015).

Para Serale et al. (2018a):

> (...) A resposta dinâmica das saídas de um sistema é afetada por entradas controladas (ou variáveis manipuladas) e entradas não controladas (ou distúrbios). Um modelo dinâmico do sistema pode capturar tal dinâmica. Posteriormente, o controlador pode explorá-los para fazer previsões da possível resposta futura do sistema em função de futuras entradas controladas e não controladas. O MPC usa essas previsões para selecionar a melhor sequência de futuras variáveis manipuladas, de acordo com um índice de desempenho específico. Este último é definido em uma janela de tempo que começa no tempo atual e abrange um determinado horizonte de previsão no futuro. A melhor sequência é obtida resolvendo um problema de otimização numérica, que também leva em consideração as restrições nas variáveis de entrada e saída que devem ser satisfeitas durante a operação do sistema. (...)

Na figura 11 é possível visualizar o princípio do horizonte de predição utilizado no MPC. Primeiro gráfico refere-se ao instante t, já a segunda o $t + 1$, assim percebe-se que a cada iteração o MPC realiza uma nova otimização calculando o sinal de controle a ser

aplicado. Para Reis (2018), o horizonte de controle é definido como o período estabelecido para o cálculo do conjunto de sinais de controle, ou seja, o tempo gasto para tomada de decisão. Já o horizonte de predição é definido como o período em que o erro entre a variável controlada e sua referência deve ser zero, ou seja, é o período de tempo futuro usado para predizer o comportamento do sistema assim é possível avaliar as consequências das ações planejadas antes que elas sejam aplicadas.

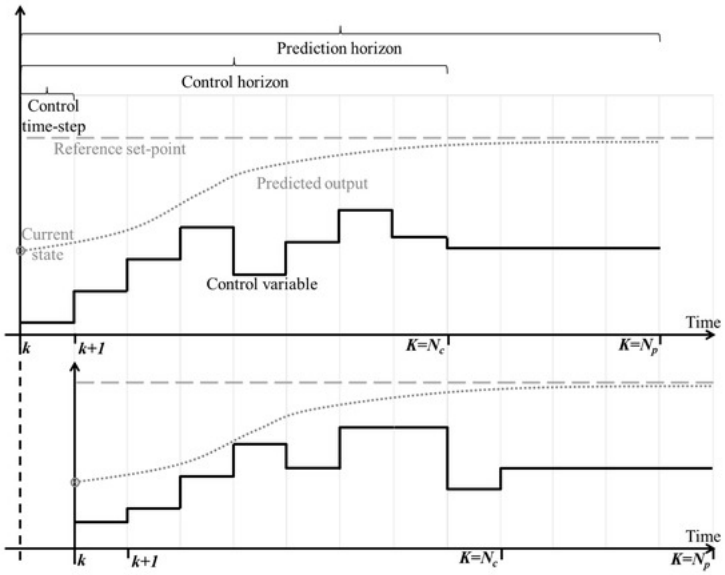

Figura 11 – Princípio do funcionamento do MPC utilizando o horizonte de predição e controle. Fonte: Serale et al. (2018a).

Sendo assim, o principal ponto positivo desse controlador é que ele consegue controlar diversos sistemas com plantas mais complexas. Além disso, ele leva em conta as restrições da planta para solucionar o problema de otimização e estabelecer um sinal de controle ótimo para tais condições. Porém, existem também os pontos negativos como a necessidade de ter o modelo do sistema e, principalmente, o alto custo computacional. No entanto, com o avanço computacional a utilização do MPC vem crescendo exponencialmente (SERALE et al., 2018b).

2.6 Controlador inteligente

2.6.1 Aprendizagem de máquina

A inteligência artificial é baseada na chamada aprendizagem de máquina. Aprendizagem está diretamente relacionada com a obtenção de determinada habilidade para resolução de determinada tarefa. Sendo assim, a aprendizagem de máquina diz respeito a um conjunto de técnicas que conseguem ensinar a computadores como realizar determinada tarefa (OLIVEIRA et al., 2021). Um ponto importante é que com o aprendizado de máquina é

possível melhorar automaticamente o desempenho do sistema por meio da experiência (MITCHELL et al., 1990) assim como ocorre em seres vivos que, com experiência, obtêm melhor desempenho para executar determinada função.

2.6.2 Redes neurais

Redes neurais são máquinas projetadas para modelar a forma que o cérebro processa uma informação e executa alguma função (HAYKIN, 2001). Elas tentam simular o funcionamento do cérebro humano por meio de unidades de processamento(neurônios artificiais) interligadas compondo uma rede com várias camadas. Nesse sentido, segundo Haykin (2001), uma rede neural se assemelha ao cérebro em dois principais aspectos: consegue adquirir conhecimento a partir de seu ambiente por meio do processo de aprendizagem e a força de conexão entre os neurônios (pesos sinápticos) são responsáveis por armazenar o conhecimento.

Assim sendo, a aprendizagem é principal etapa para que uma rede neural consiga adquirir e simular o conhecimento humano. Essa etapa é realizada por meio do treinamento da rede, ou seja, para Rauber (2005):

> (...)Isso significa que os graus de liberdade que a rede dispõe, para solucionar a tarefa em consideração, têm que ser adaptados de uma maneira ótima. Normalmente, isso significa que temos que modificar os pesos w_{ij} entre o neurônio i e o neurônio j, segundo um algoritmo. Um conjunto finito T de n exemplos de treino está à nossa disposição para adaptar os pesos durante a fase de treinamento da rede.(...)

2.6.3 Aprendizagem supervisionada

Um dos métodos de aprendizagem é a aprendizagem supervisionada onde no treinamento cada exemplo é composto pela entrada (x) e pela saída (y) desejada, ou seja, o conjunto de treino é composto por n pares de exemplos (x_p,y_p), assim, é possível 'ensinar' a rede neural como obter Y (conjunto de saída) por meio de X (conjunto de entrada). Um bom exemplo desse método é a regressão linear, onde se tem um problema unidimensional com um conjunto de treino que consiste em pares de números reais (x_p,y_p) (ver figura 12). O objetivo é a determinação de w_0 e w_1 da reta $y = w_0 + w_1 x$. Assim o objetivo do algoritmo de aprendizagem é minimizar a diferença entre o valor desejado e o valor que é a saída do sistema dada a mesma entrada (RAUBER, 2005).

2.6.4 Sobreajuste e Subajuste

O algoritmo de aprendizagem busca definir uma função que descreve a relação entre a entrada e saída do sistema. Espera-se então conseguir um modelo final com boa capacidade de generalização, isto é, conseguir saídas coerentes para qualquer entrada, mesmo que ela

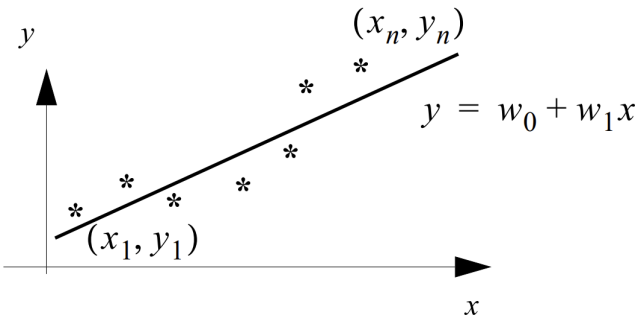

Figura 12 – Regressão linear. Fonte: Rauber (2005).

não esteja no conjunto de treinamento. Para isso, deve-se separar conjuntos de treino e teste aleatoriamente fazendo com que o erro nos dois conjuntos sejam equivalentes. Assim, um bom modelo é quele que consegue reduzir o erro no conjunto de treinamento e a diferença entre os erros dos conjuntos de teste e treinamento (OLIVEIRA et al., 2021).

O subajuste, ou *underfitting*, acontece quando o modelo não consegue reduzir o erro no conjunto de treinamento resultando em um mal desempenho. Já o sobreajuste, ou *overfitting*, ocorre quando o erro do modelo no conjunto de treinamento é baixo, porém no conjunto de teste ele não tem o mesmo resultado, resultando em um mau desempenho em qualquer conjunto que não seja o de treinamento. O *underfitting* ocorre quando não se treina o modelo o suficiente para que o erro no conjunto de teste seja aceitável, já o *overfitting* acontece quando o modelo é excessivamente treinado de modo que fique tão ajustado aos dados de treinamento que resulta em baixa capacidade de generalização.

2.6.5 Rede neural baseada em MPC

Em Oliveira et al. (2021), os autores treinam uma rede perceptron multicamadas(MLP) com base em um controlador por modelo preditivo não linear e consegue obter resultados satisfatórios, pois a rede consegue acompanhar muito bem o modelo original em diferentes circunstâncias porém demanda consideravelmente menos esforço computacional quando comparada ao MPC podendo ser implementada em sistemas embarcados. Com a figura 13 pode-se ver a arquitetura utilizada. Dessa mesma forma, é possível treinar uma rede neural baseada em um MPC obtendo resultados semelhantes, porém reduzindo drasticamente o custo computacional de modo a conseguir aplicar em um microcontrolador.

2.7 Estado da arte

Quando se fala da utilização de inteligência artificial para controlar ou facilitar o controle de um processo, existem diversas vertentes que podem ser utilizadas. Uma delas é utilizar a inteligência computacional para chegar em parâmetros de controladores clássicos bastante

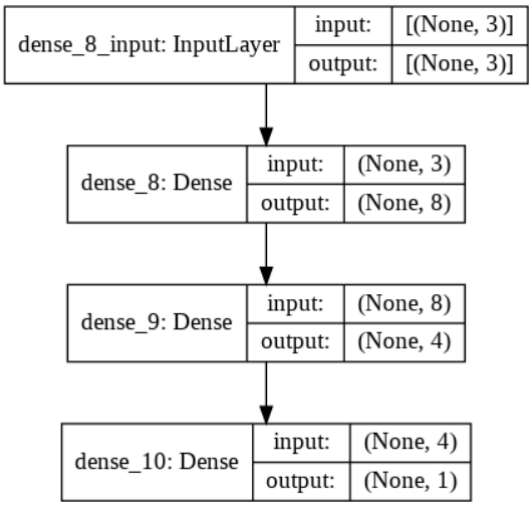

Figura 13 – Arquitetura da rede MLP utilizada no trabalho. Fonte: Oliveira et al. (2021).

famosos e usuais. Em PEREIRA JUNIOR, COSTA e SILVA JÚNIOR (2018) o autor aplica uma rede neural artificial para chegar aos valores das constantes proporcional, integral e derivativa, respectivamente K_p, K_i e K_d de um controlador PID utilizado em um sistema de controle de nível de um destilador de água com alimentação constante. Na figura 14 pode-se ver a representação em diagrama de blocos do sistema de sintonia do controlador PID.

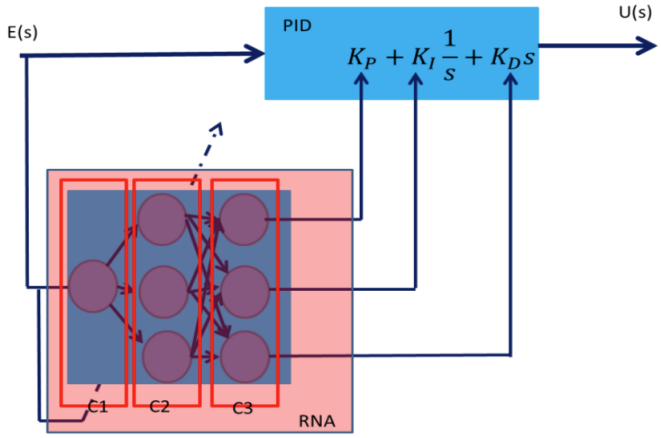

Figura 14 – Representação em diagramas de blocos do sistema inteligente de sintonia do controlador PID. Fonte: PEREIRA JUNIOR, COSTA e SILVA JÚNIOR (2018).

No trabalho, tanto o controlador como a rede neural são alimentados pelo erro do sistema. O controlador utiliza o erro para calcular e aplicar um sinal no sistema, já a rede neural utiliza o sinal de erro para a atualização dos pesos do controlador. A rede utilizada possui 3 camadas com uma estrutura 1-3-3, ou seja, 1 neurônio na camada de entrada

ou unidade sensorial, 3 na camada oculta e 3 na camada de saída sendo que a função de ativação utilizada é uma sigmoide. Assim, a rede neural consegue receber o erro e entregar os valores de K_p, K_i e K_d em sua saída. Os autores chegam em resultados dentro das especificações: o tempo de subida foi 0.00070985s, o tempo de acomodação foi 0.0014s e os parâmetros sintonizados foram K_p=100, K_i=67 e K_d=0. No final, concluem que o método utilizado se mostrou eficiente e, principalmente, levou em conta os parâmetros de projeto, não deixando ocorrer *Overshoot* que é de suma importância para o tanque não transbordar.

Além das redes neurais, outros métodos de inteligência computacional também podem ser empregados para a sintonia de um controlador clássico. Em Coelho et al. (2019) os autores utilizam a lógica *fuzzy* para sintonizar um controlador PID que deve controlar sistemas industriais com aplicação em processo de viradores de vagões por meio da variação da velocidade de rotação dos alimentadores de esteiras de aço. Na maioria das vezes esse controle era feito de forma empírica variando a velocidade de rotação do alimentador o que dependia exclusivamente da perícia do operador.

O controlador PID-*Fuzzy* utilizado é composto por um controlador PID sintonizado pelo método de *Ziegler-Nichols* que, em seguida, tem seus ganhos regulados de maneira online por regras *fuzzy*. Para obter resultados consistentes, os autores realizaram simulações no *software* MATLAB levando em consideração duas plantas com funções de transferência distintas. Para verificar a eficácia do método utilizado foram comparadas as respostas com 2 controladores: um PI sintonizado por tentativa e erro e um PID sintonizado pelo método de *Ziegler-Nichols*. A partir da figura 15 pode-se ver o comportamento dos 3 controladores em uma das plantas em resposta ao degrau. Já na figura 16 é possível visualizar os mesmos 3 controladores com um sinal de 50% de perturbação no valor de referência.

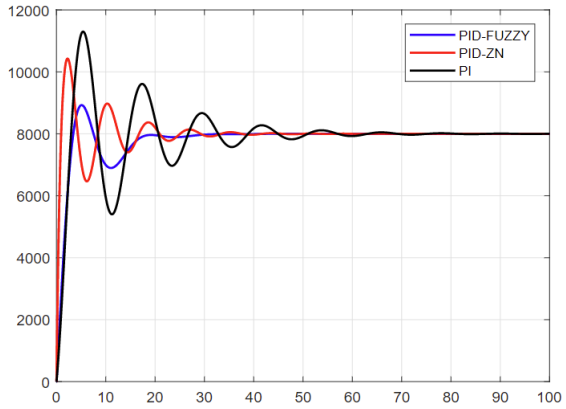

Figura 15 – Resposta ao degrau dos controladores PID-Fuzzy, PID e PI aplicados na mesma planta. Fonte: Coelho et al. (2019).

A partir desses resultados, os autores concluem que o controlador implementado

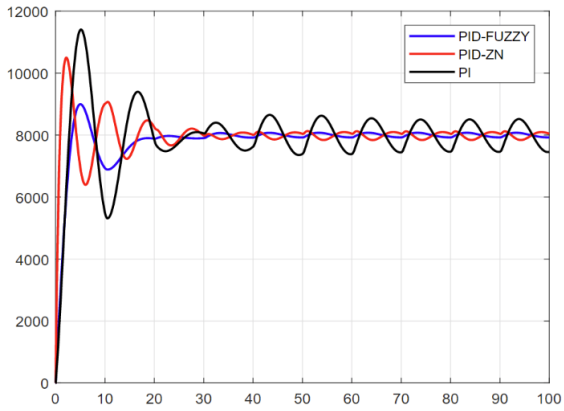

Figura 16 – Resposta ao degrau dos controladores PID-Fuzzy, PID e PI aplicados na mesma planta com um sinal de pertubação de 50% do sinal de referência. Fonte: Coelho et al. (2019).

utilizando a lógica fuzzy obtém melhor desempenho em comparação com os outros PID (sintonizado pelo método de *Ziegler-Nichols*) e PI (sintonizado por tentativa e erro) tanto em respeito a tempo de acomodação como em sinal de *overshoot* levando em consideração o sistema com e sem pertubações.

Em Oliveira et al. (2021), os autors apresentam um procedimento de síntese de um controlador NMPC (*Nonlinear Model Predictive Control*) para um aeropêndulo utilizando redes neurais artificiais. Primeiramente são executadas simulações do sistema com o controlador NMPC em malha fechada, assim, são captados os sinais de referência, controle ótimo e estados do sistema. Com esses dados é executado o treinamento da MLP (*Multi Layered Perceptron*) e depois a simulação e comparação com o controlador NMPC. Finalmente o modelo treinado é incorporado a um microcontrolador. Toda a metodologia é exposta na figura 17.

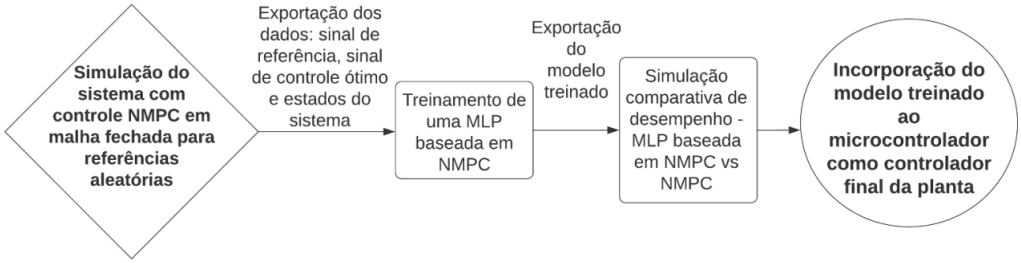

Figura 17 – Proposta da metodologia para a implantação de um controlador com MLP baseada em NMPC. Fonte: Oliveira et al. (2021).

Como forma de comparação os autores executaram simulações e os gráficos da posição do pêndulo $\theta(rad)$, velocidade angular do pêndulo $\frac{d\theta}{dt}(rad/s)$ e dos sinais de controle u são mostrados nas figuras 18, 19 e 20 respectivamente. Como pode-se observar, o controlador treinado acompanha muito bem as dinâmicas em malha fechada de forma muito

semelhante ao controlador NMPC tradicional em todos os sentidos: posição, velocidade e sinais de controle. Dessa forma, os autores concluem que o algoritmo de controle proposto mostrou-se capaz de operar conforme os requisitos desejados em um sistema tipicamente não linear, sendo computacionalmente muito mais simples que o controlador original, a ponto de ser implementado em um microcontrolador de 8bits/16MHz. O controlador proporcionou uma diminuição de até 6000 vezes no tempo de cálculo das ações de controle em comparação com uma estratégia clássica de NMPC. Todos esses resultados são no contexto de simulações. Em relação a aplicação na planta real o autor chega a seguinte conclusão:

> (...)Na aplicação à planta real, o desempenho em malha fechada não foi o mesmo, e isto pode ser explicado devido à impossibilidade de aplicação do controlador NMPC à planta real para uma geração de dados mais fidedigna à realidade.(...)

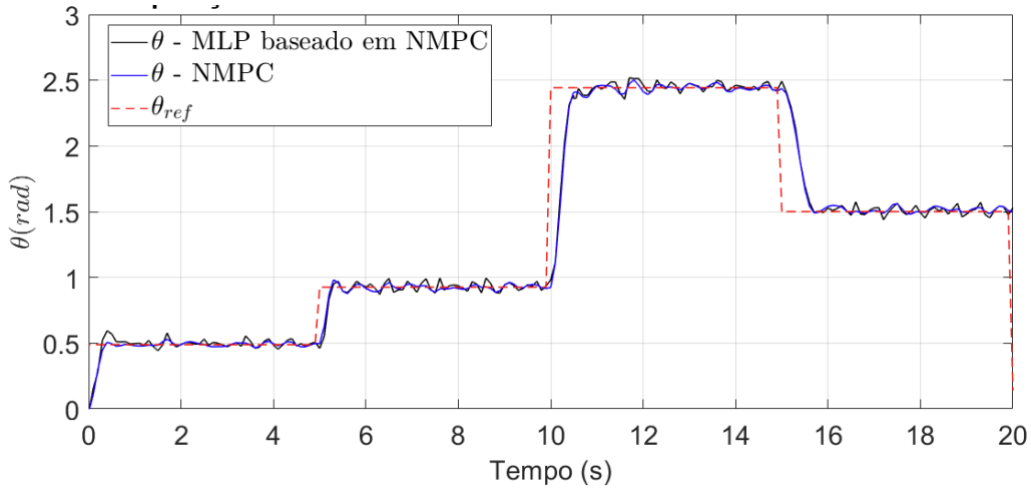

Figura 18 – Comparação entre os controladores MLP baseado em NMPC e NMPC em relação a posição. Fonte: Oliveira et al. (2021).

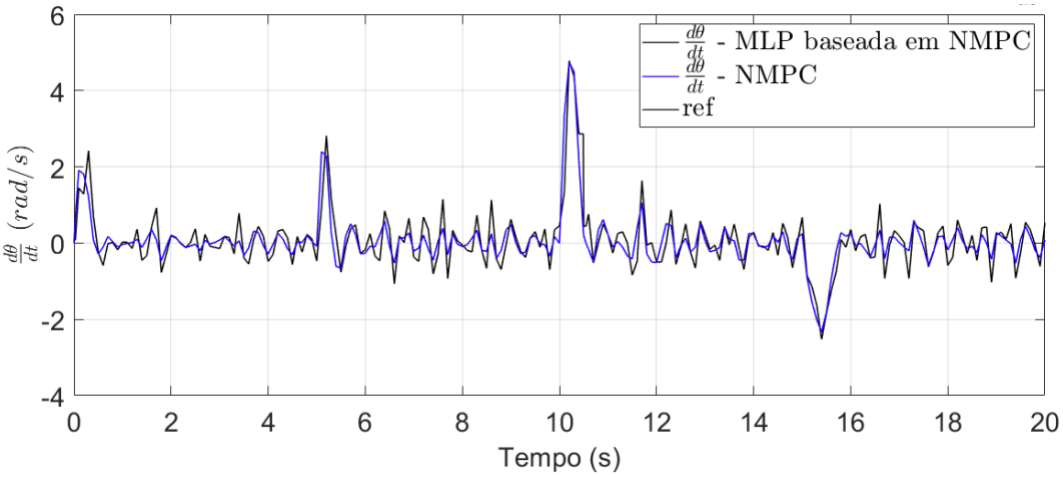

Figura 19 – Comparação entre os controladores MLP baseado em NMPC e NMPC em relação a velocidade angular. Fonte: Oliveira et al. (2021).

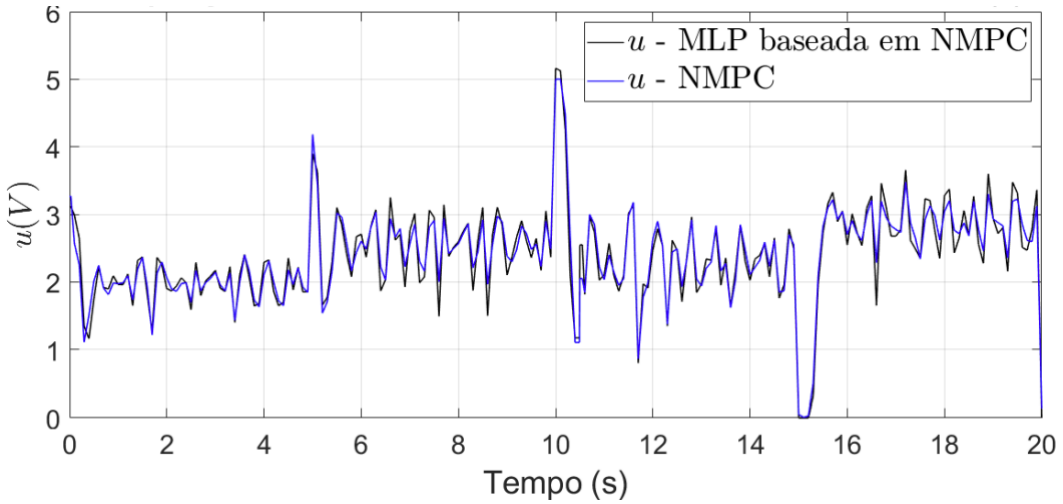

Figura 20 – Comparação entre os controladores MLP baseado em NMPC e NMPC em relação a sinal de controle. Fonte: Oliveira et al. (2021).

3 Desenvolvimento

3.1 Idealização da planta

Depois de realizar a pesquisa bibliográfica, o primeiro passo a ser feito foi idealizar o funcionamento da planta. Sendo assim, a ideia primeiramente é que o microcontrolador capture os dados de velocidade do motor por meio do *encoder*, envie-os para o computador por meio da comunicação serial, o computador calcule o sinal de controle, envie o valor do *duty cycle* ao microcontrolador e, por fim, o microcontrolador aplique o sinal de PWM ao motor. A figura 21 mostra o projeto do funcionamento da planta.

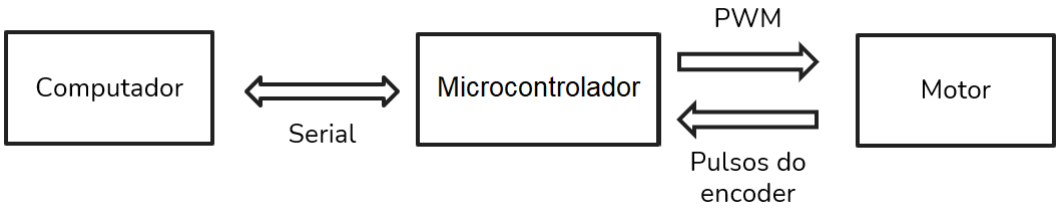

Figura 21 – Planta proposta para o controlar o motor. Fonte: Autoria própria.

3.2 Circuito eletrônico

Agora, a próxima etapa foi desenvolver o circuito eletrônico de acionamento do motor de corrente contínua juntamente com o de captação dos dados de velocidade. Ambos os circuitos são controlados por um microcontrolador. Para isso, foi utilizado o programa Proteus 8 Professional que, além de gerar o esquemático, consegue fazer simulações do circuito implementado. Na figura 22 pode-se ver como o circuito foi idealizado.

O circuito é composto por um *encoder* que trabalha como sensor de velocidade e é responsável por fechar a malha de controle. Esse *encoder* está acoplado ao motor de 6V que será controlado. Além disso, existe a ponte H responsável pelo acionamento do motor por meio de um PWM, os pinos desse circuito estão sendo mostrados na figura 23. Finalmente, para controlar tudo isso, ou seja, acionar o motor e captar os dados do sensor, existe o microcontrolador que, como se pode visualizar na figura 22, é um Arduino Uno.

Com o circuito idealizado a próxima etapa foi montar o circuito em uma *protoboard* de 400 pinos. Pode-se observar nas figuras 24 e 25.

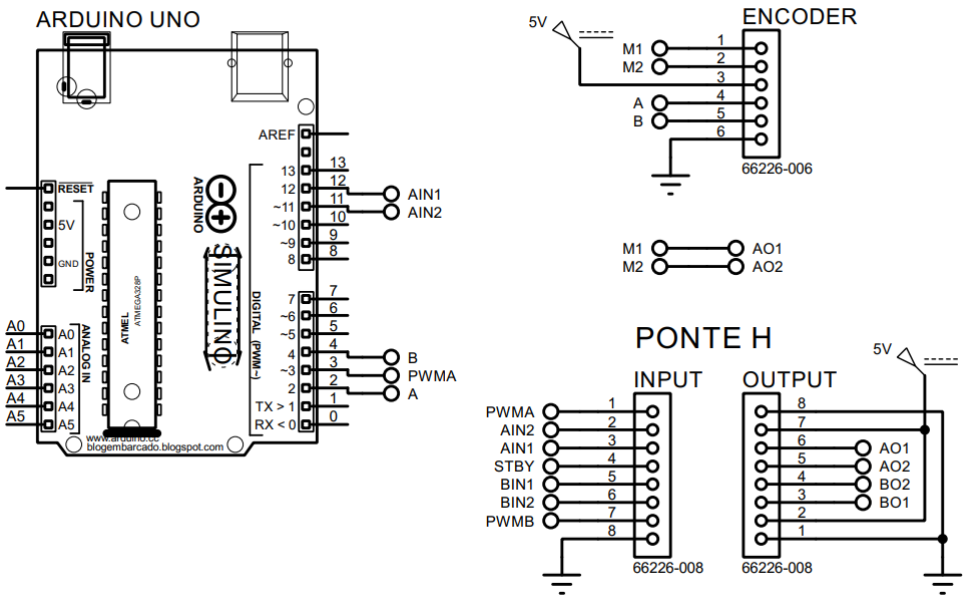

Figura 22 – Circuito proposto para o controle do motor, feito no *software* Proteus 8 Professional. Fonte: Autoria própria.

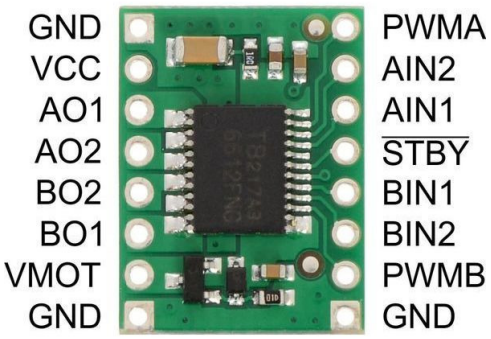

Figura 23 – Ponte H utilizada para o acionamento do motor. Fonte: Pololu[1].

3.3 Código do microcontrolador

Em seguida, foi realizada a conexão do circuito com o computador. A IDE do Arduino foi utilizada para iniciar a programação do microcontrolador. O código implementado é responsável por captar os dados de velocidade do motor em uma frequência de amostragem pré-determinada e enviá-los para o computador por meio da comunicação serial. Além disso, o código também é responsável pela comunicação inversa, ou seja, do computador para a planta. Assim, quando o computador enviar algum dado, o Arduino consegue receber as mensagens e aplicá-las no ciclo de trabalho *(duty cicle)* do PWM.

Segundo Santana (2022), o tempo de amostragem deve ser, pelo menos, 8 vezes menor que o tempo de subida do sistema para sistemas subamortecidos, ou seja, ao aplicar um degrau em malha aberta, o sistema deve captar dados, ao menos, 8 vezes durante o

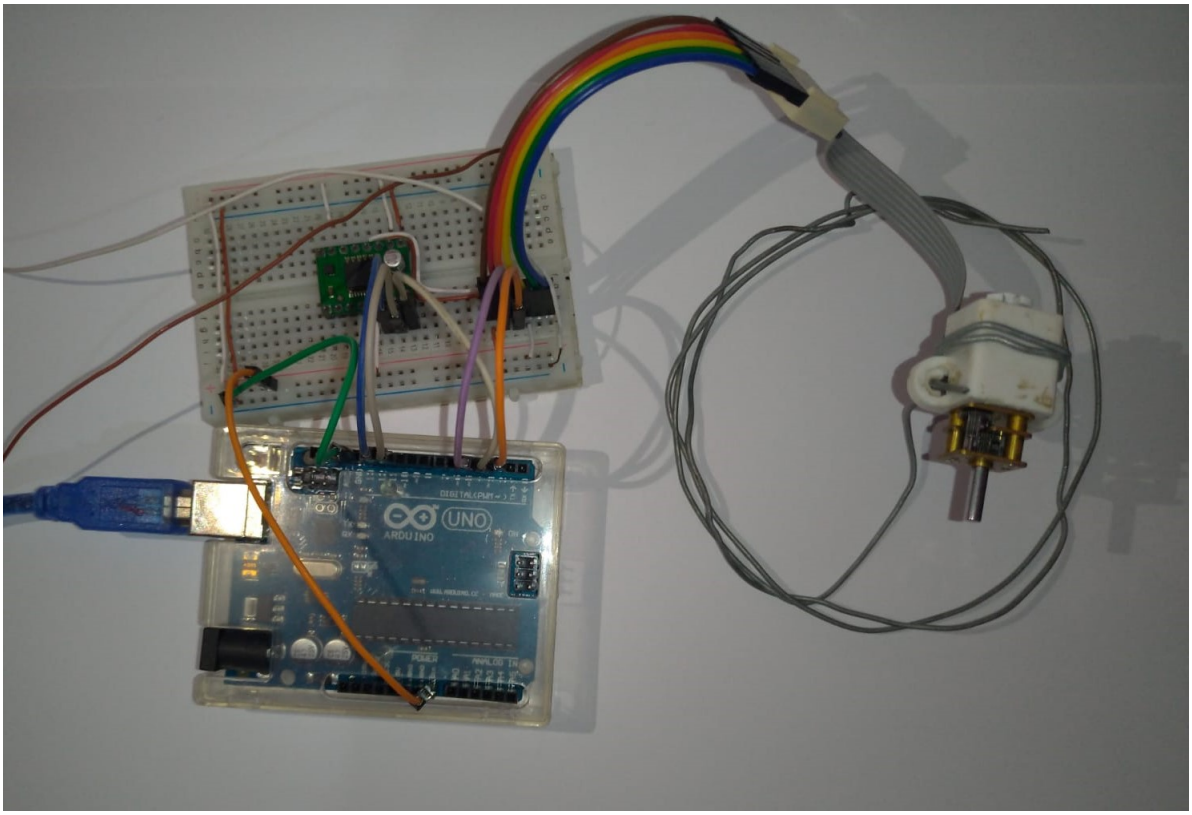

Figura 24 – Circuito montado na bancada. Fonte: Autoria própria.

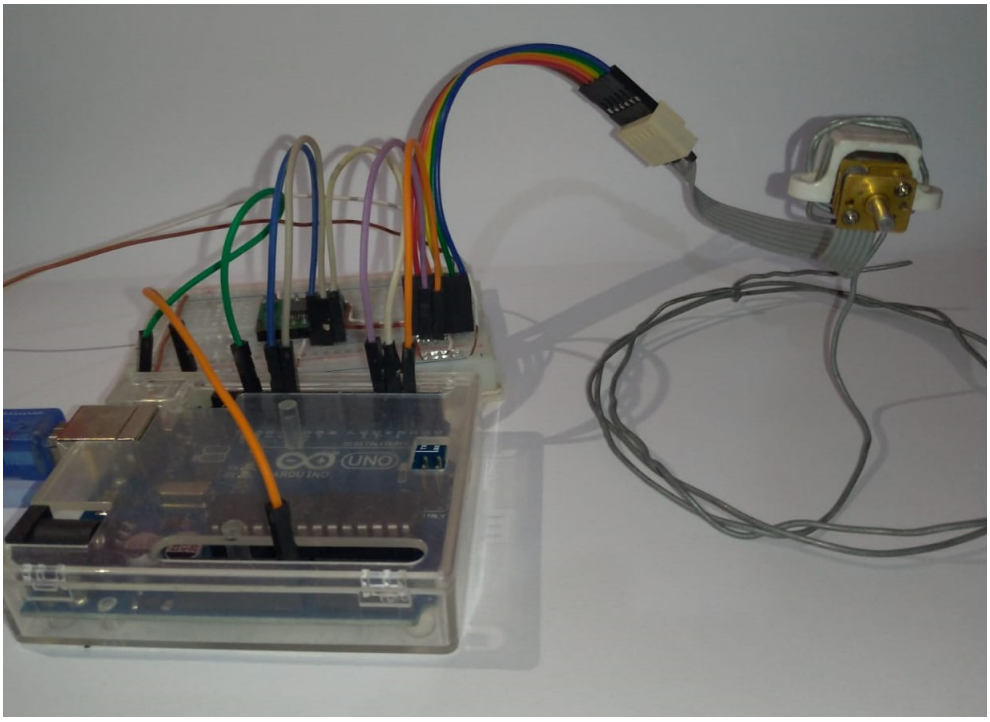

Figura 25 – Circuito montado na bancada em outro ângulo. Fonte: Autoria própria.

tempo de subida do sistema que é o tempo do sistema partir de 10% e atingir 90% do valor de estabilização.Dado isso, de maneira empírica foi escolhida uma frequência de amostragem igual a 100Hz. Vale ressaltar que a etapa periódica do código, ou seja, a etapa de captação da velocidade e envio para o computador, é realizada por meio de uma interrupção por estouro do *timer* o que garante a frequência desejada de acordo com o cristal do próprio microcontrolador.

3.4 Modelagem da planta: método da curva de reação

A função de transferência de um sistema é muito importante para o projeto de um controlador. Com ela consegue-se realizar algumas análises, simulações e até sintonia de controladores. Sendo assim, foi necessário obter a função de transferência da planta. Para isso, foi utilizado um método caixa cinza proposto em Ziegler, Nichols et al. (1942): o método da curva de reação. Como dito no capítulo de referencial teórico esse método consiste em aplicar um degrau com amplitude de 10% a 20% do fundo de escala da variável controlada no sistema em malha aberta e com a curva obtida consegue-se aproximar o sistema a um sistema de primeira ordem com atraso de transporte.

Com a planta montada, para fazer o experimento foi necessário desenvolver um programa no *software* MATLAB que enviasse o valor do *duty cycle* do PWM para o microcontrolador e captasse os dados da resposta do sistema. Como o PWM do arduino é de 8 bits, a variável controlada varia de 0 a 255 o que corresponde de 0V a 5V aplicados ao motor. Assim, primeiramente foi enviado um valor de 100 (1,96V) para que o motor começasse a girar e assim o experimento não considerasse a faixa de tensão em que o motor não gira. Com o motor girando foi aplicado um degrau de aproximadamente 15% do fundo de escala, ou seja, o sinal aplicado foi de 100 (1,96V) a 138 (2,73V) e assim foram captados os dados de velocidade. Na figura 26 pode-se ver o degrau que foi aplicado a planta e na figura 27 pode-se ver a resposta do sistema ao degrau aplicado.

O próximo passo foi achar o ponto de inflexão da curva de resposta, para isso bastou derivar a curva amostrada calculando a diferença entre a amostra n e a $n-1$ em todos os pontos, e, assim, obter o tempo em que a derivada atinge seu maior valor. Com o tempo x em mãos foi possível obter a velocidade y e assim saber a posição exata do ponto de inflexão no plano cartesiano (representado pelo círculo vermelho na figura 28). Com o valor da derivada e o ponto de inflexão da curva, é possível traçar a reta tangente a curva (representada pela linha verde na figura 28) por meio de uma equação de primeiro grau. Agora é possível obter os parâmetros atraso de transporte (L), constante de tempo (T) e ganho (K) da função de transferência.

O parâmetro L é a diferença de tempo entre a aplicação do degrau e o início da resposta do sistema. De maneira prática o valor de L foi calculado pela diferença do

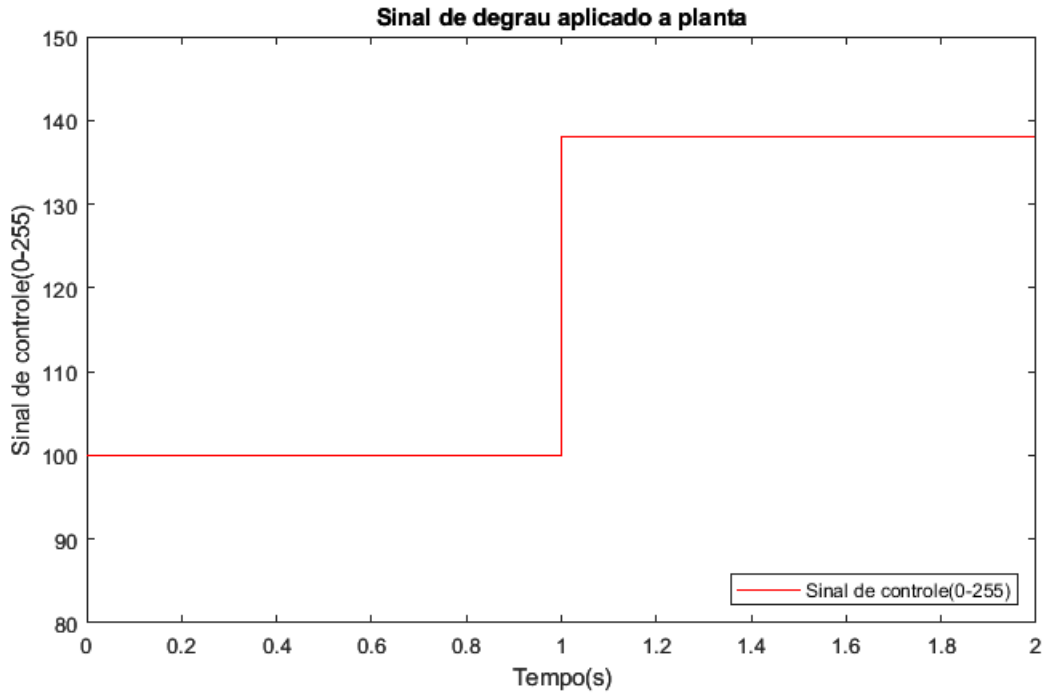

Figura 26 – Degrau aplicado ao *duty cycle* do PWM empregado no motor. Fonte: Autoria própria.

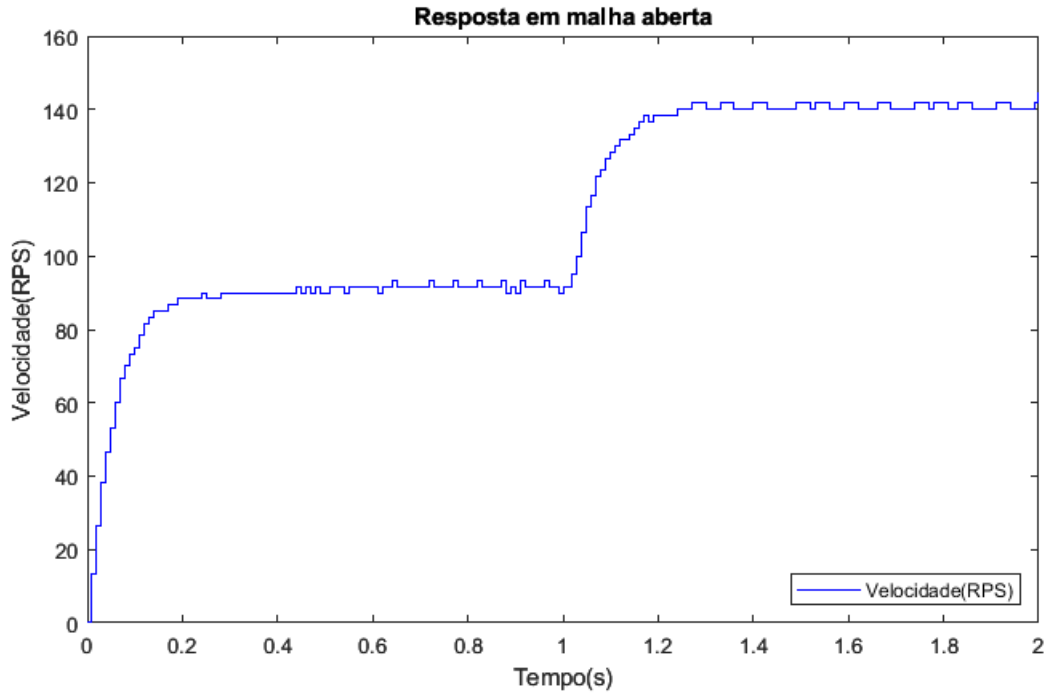

Figura 27 – Resposta da velocidade do motor com a aplicação do degrau. Fonte: Autoria própria.

tempo entre a interseção da linha tangente com a velocidade de partida do experimento e o momento em que o degrau é aplicado. O valor de L obtido foi de aproximadamente 0,0176s. O parâmetro K foi calculado pela divisão da diferença entre a velocidade de estabilização e a de partida por 38 que foi o valor do degrau aplicado no PWM, o valor obtido foi de aproximadamente 1,293. O parâmetro T foi calculado pela diferença de tempo entre a interseção da linha tangente com a velocidade de estabilização e a interseção da linha tangente com a velocidade de partida, o valor obtido foi de aproximadamente 0,0737s.

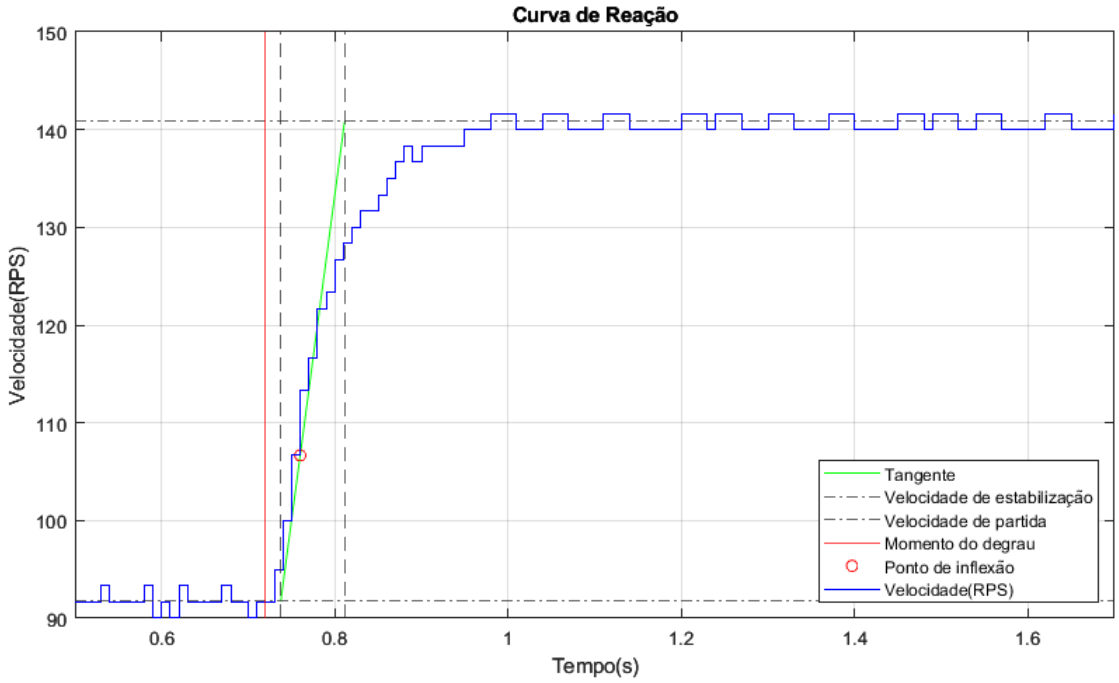

Figura 28 – Curva de reação do sistema com alguns dados importantes para a modelagem. Fonte: Autoria própria.

Como forma de comparação a curva de resposta do sistema real foi normalizada matematicamente para que se consiga comparar a resposta a um degrau do sistema modelado (tanto o contínuo como o discreto). Dessa forma, foi possível comparar os gráficos como mostra a figura 29. Pode-se analisar que o modelo obtido acompanhou bem o resultado obtido na planta real mostrando que o método da curva de reação obteve um bom resultado.

3.5 Projeto dos controladores

O principal foco desse trabalho é projetar um controlador que utiliza inteligência artificial e outros diferentes controladores para comparar seus resultados na planta. Com a modelagem pronta é possível projetar controladores de várias maneiras diferentes. O *software* MATLAB dispõe de vários recursos para isso. Sendo assim, esse trabalho vai projetar

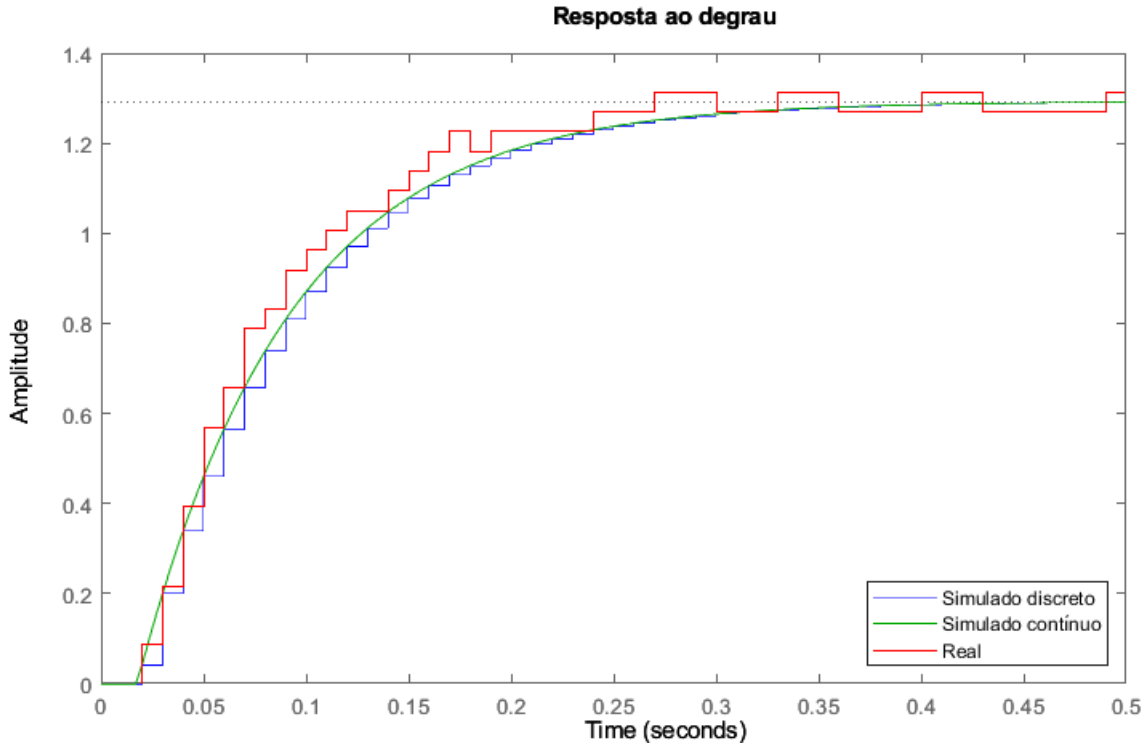

Figura 29 – Comparação da resposta ao degrau do sistema modelado e da planta real. Fonte: Autoria própria.

um controlador PID para comparação, um MPC para comparação e treinamento da rede neural e finalmente um controlador que utiliza rede neural.

3.5.1 Controlador PID

Para projetar o controlador, primeiramente foi necessário passar o modelo contínuo obtido anteriormente para o modelo discreto equivalente com a frequência de amostragem de 100Hz (período 0,01). Para isso, foi utilizada a função *c2d* do *software* MATLAB descrita em The MathWorks (2023b) que discretiza o modelo de sistema dinâmico contínuo utilizando retentor de ordem zero nas entradas e o período de amostragem.

O primeiro controlador a ser projetado foi o PID e para isso foi utilizada a função do *software* MATLAB descrita em The MathWorks (2023d) passando como parâmetros o sistema discretizado e o tipo de controlador desejado, no caso, o PID em sua forma paralela como mostrado na figura 30. Assim os valores obtidos para *kp*, *kd* e *ki* foram respectivamente 1,296, 0 e 21,855. Interessante ressaltar que o próprio método atribuiu 0 para o a constante derivativa resultando em um controlador PI. Lembrando que o controlador deve e será aplicado numa planta como a exposta na figura 8.

Para aplicar o controlador na planta, a cada iteração, ou seja, a cada leitura do sensor, é calculado o sinal de controle a partir do erro, da derivada do erro e da integral do mesmo. O erro é calculado pela diferença entre o *setpoint* e a velocidade captada do motor.

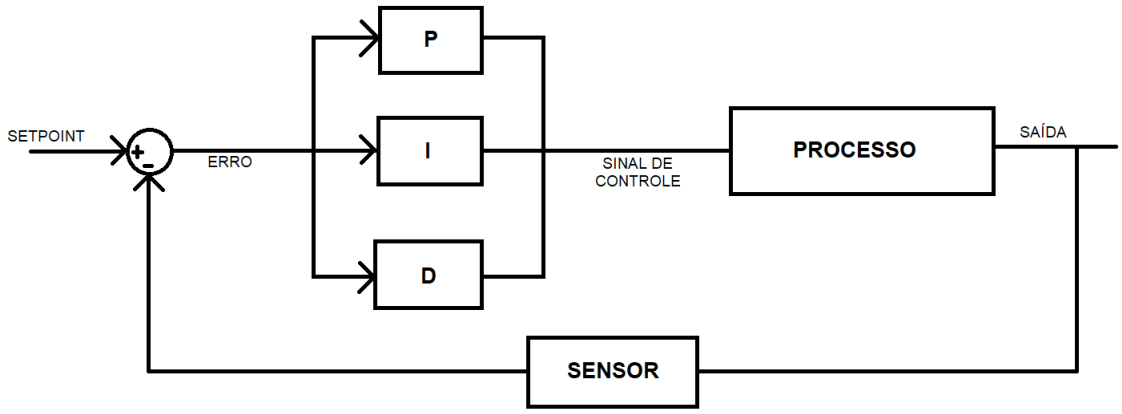

Figura 30 – Diagramas de blocos do sistema com um controlador PID do tipo paralelo. Fonte: Autoria própria.

A derivada do erro é calculada pela multiplicação da diferença entre o erro atual e o da iteração anterior com a frequência de amostragem (100Hz). A integral é simplesmente a soma entre todos os valores de erro calculados. Destaca-se que na parte integrativa, com finalidade de diminuir o efeito *windup*, foi utilizada a técnica condicional limitando o efeito integrativo entre -255 e 255. Por fim, cada valor obtido é multiplicado por sua respectiva constante (*kp*, *kd* e *ki*) e somados resultando no valor do sinal.

Depois de calcular o sinal de controle é necessário garantir que ele está entre 0 e 255, além disso é necessário arredondar o valor obtido para um número inteiro. O motivo de tudo isso é que o *duty cycle* do PWM do Arduino é de 8 bits, sendo assim a variável utilizada no código é *unsigned char* recebendo somente valores inteiros entre 0 e 255. Esse procedimento deverá ser realizado em todos os controladores projetados.

3.5.2 Controlador MPC

A parte mais importante de um controlador por modelo preditivo, como o próprio nome diz é o modelo. Sendo assim, a parte fundamental e mais complexa do controlador já foi efetuada: a modelagem. Para projetar o MPC mais uma vez foi utilizado o *software* MATLAB. O primeiro passo foi passar o modelo de tempo contínuo para espaço de estados. Para isso foi utilizada a função *ss* descrita em The MathWorks (2023f). Depois disso, foi necessário discretizar o modelo por espaço de estados obtido utilizando a mesma função utilizada para o controlador PID, a *c2d*, descrita em The MathWorks (2023b).

Com o modelo em espaço de estados discretizado é possível criar o MPC. Para isso, foi utilizada a função *mpc* descrita em The MathWorks (2023c). Para ela foram passados os seguintes parâmetros: modelo por espaço de estados discretizado, período de amostragem, horizonte de predição e horizonte de controle. O horizonte de predição utilizado foi 10 e esse valor foi obtido de forma empírica da seguinte forma: foram realizados diversos testes

com diferentes valores e aplicados ao motor com diferentes *setpoints*, o valor que obteve o melhor resultado foi selecionado. Para o horizonte de predição foi atribuído o valor 1, ou seja, em todo instante de tempo o controlador calcula os próximos sinais a serem aplicados de acordo com o horizonte de predição, no entanto, o único sinal aplicado é o referente ao instante atual, esse processo é conhecido como estratégia de horizonte móvel (REIS, 2018).

Uma grande vantagem do MPC é a possibilidade de incluir restrições em suas variáveis por se tratar de um problema de otimização. A variável de controle da planta, como já foi dito, varia entre 0 e 255, por isso, é importante que o controlador considere essa informação. Com esse fim, foi alterada a propriedade da variável controlada pelo controlador.

Para aplicar o controlador na planta, a cada iteração foi utilizada a função *mpcmove* descrita em The MathWorks (2023a) que retorna o sinal de controle a ser aplicado e recebe como parâmetros o objeto do controlador MPC, o estado estimado atual, a saída do sistema (no caso, a velocidade atual medida) e o *setpoint*. Depois de obter o sinal de controle, é necessário, da mesma forma que no controlador PID, arredondá-lo e finalmente aplicá-lo à planta.

3.5.3 Controlador por rede neural artificial

3.5.3.1 Aquisição dos dados

Para projetar uma rede neural o principal requisito é ter dados suficientes para seu treinamento. A metodologia utilizada no trabalho foi utilizar como base para o treinamento o MPC, pois esse tipo de controlador tem uma enorme versatilidade sendo capaz de controlar diversas plantas com complexidades distintas. Em contrapartida, o custo computacional é elevado. Nesse sentido, a proposta do trabalho é treinar uma rede que consiga assimilar o comportamento de um MPC, porém com um custo computacional menor.

Como já foi explicado em seções anteriores, o microcontrolador envia ao computador a velocidade medida e recebe dele o sinal de controle. Sendo assim, esses, juntamente com o *setpoint*, são os dados mais importante a serem captados e salvos. A partir desses dados é possível calcular diversos outros como o erro, a derivada do erro, a integral do erro, a derivada da velocidade, entre outros. Assim sendo, esses foram os dados captados e salvos para o posterior treinamento de uma rede que pode ter várias entradas.

Para conseguir tais dados, foi feito um experimento que consistiu em aplicar o MPC na planta durante algum tempo seguindo diversos valores de *setpoint*. O *setpoint* variou de 400 em 400 amostras captadas, ou seja, na frequência utilizada, o sinal de referência variou de 4 em 4 segundos, tempo suficiente para a estabilização da planta com o MPC. Essa variação ocorreu de maneira aleatória seguindo a distribuição uniforme entre 90 e 200 (esses valores foram escolhidos para não atingir os extremos de funcionamento do

motor), para isso foi utilizada a função *randi* do *software* descrita em The MathWorks (2023g). O sinal de referência foi aplicado e os dados captados e salvos. Esse experimento captou 280000 sinais resultando em 700 degraus no sinal de referência e uma duração de aproximadamente 47 minutos.

Os dados foram exportados do *software* MATLAB por meio de um arquivo csv e importados no ambiente de treinamento e aprendizagem de máquina Google Colab, isso foi realizado pois o treinamento de uma rede neural tem uma complexidade computacional grande e com o Google Colab foi possível realizar esse procedimento em máquinas do Google que tem desempenho superior ao do computador que foi executado o MATLAB. Parte do conjunto de dados pode ser visualizada na figura 31.

	sinalDeControle	velocidade	setpoint
0	80	60	180
1	202	60	180
2	203	75	180
3	203	75	180
4	202	140	180
...	...	...	...
279995	172	185	183
279996	172	185	183
279997	172	185	183
279998	172	180	183
279999	172	185	183

280000 rows × 3 columns

Figura 31 – Conjunto de dados captados após o experimento com o controlador MPC. Fonte: Autoria própria.

3.5.3.2 Rede Neural baseada no MPC

Com os dados importados no ambiente, fez-se necessário determinar uma arquitetura robusta, capaz de assimilar o comportamento do MPC com parcimônia. Em Oliveira et al. (2021), os autores projetam uma rede perceptron multicamadas (MLP) que conseguiu sintetizar a lei de controle de um NMPC (*Nonlinear Model Predictive Control*) para o controlador simulado. Sendo assim, foram realizados diversos experimentos para determinar a arquitetura com um desempenho satisfatório a partir da rede exposta no trabalho.

Um ponto importante é que a rede foi treinada para predizer o sinal de controle, ou seja, a saída esperada é o sinal de controle.

Vale ressaltar que por meio de experimentos o otimizador escolhido para o treinamento foi o Nadam (o mesmo utilizado em Oliveira et al. (2021)) com taxa de aprendizagem de 0,01, pois essa combinação foi a que obteve melhor desempenho nos experimentos. A função de perda (em inglês *loss*) é utilizada para mensurar o quão distante está o valor predito do valor desejado e assim ajustar os pesos da rede por meio do otimizador. A função utilizada foi a *mean squared error* (MSE). O *batch size* se refere ao número de amostras que serão utilizadas a cada iteração para fazer a previsão, calcular o *loss* e ajustar os pesos. O valor escolhido foi de 32, pois foi o mesmo utilizado em Oliveira et al. (2021) e obteve um bom desempenho. Os modelos obtidos foram definidos utilizando o Keras (TEAM, 2023) como API.

Depois de alguns testes, a arquitetura escolhida foi semelhante a exposta no trabalho Oliveira et al. (2021)(exposta na figura 13), com algumas alterações. A primeira delas é que a ativação utilizada em todas as camadas foi linear, assim não é preciso regularizar as entradas o que resulta em menos operações para o sistema processar, e isso é de extrema valia dado que a ideia posterior é aplicar a rede em um microcontrolador.

Outra mudança em relação a rede base foram as entradas escolhidas. Para determinar quais seriam essas entradas, foram realizadas diversas experiências que consistiram em treinar uma rede (com a mesma arquitetura, exceto a camada de entrada) por 10 épocas, armazenar o *loss* obtido retornado pela função de treinamento, exportar a rede treinada para o *software* MATLAB, aplicá-la na planta com o motor por meio da função *predict* descrita em The MathWorks (2023e) aplicando degraus no sinal de referência, captar os dados de saída, calcular e salvar o *mean squared error* (MSE) entre o sinal de referência e o sinal de saída. As entradas testadas e os resultados obtidos estão expostos na tabela 1.

Tabela 1 – Tabela com o desempenho de diferentes redes neurais com entradas distintas. Fonte: Autoria própria.

Entradas	*Loss* do treinamento	MSE entre o *setpoint* e a saída
Velocidade, derivada da velocidade, *setpoint*, erro	35.7477	85.9562
Velocidade, derivada da velocidade, *setpoint*	35.7562	256.6625
Velocidade, derivada da velocidade, erro	36.1136	72.6063
Velocidade, *setpoint*	34.4296	58.3937
Velocidade, erro	36.2530	206.5750
Velocidade, *setpoint*, erro	37.3713	51.1375

É possível perceber que o *Loss* obtido no treinamento em todas as entradas testadas foi bem semelhante. Sendo assim, o fator para escolha das entradas foi o MSE entre o

sinal de referência e o sinal de saída. Além disso, apesar de ser treinada para predizer o sinal de controle, o objetivo final da rede é controlar a velocidade do motor, sendo assim, o sinal mais importante para medir sua eficácia é o sinal de saída ou erro do sistema. Por isso, quanto menor o MSE obtido, melhor é o controlador. Desta forma, a arquitetura escolhida foi a com 3 entradas (Velocidade, *setpoint* e erro) conforme é visto na figura 32.

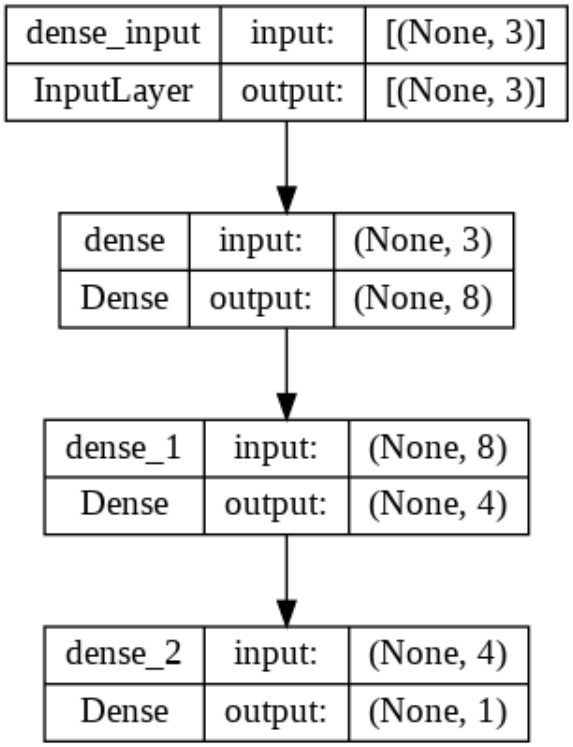

Figura 32 – Arquitetura escolhida para as entradas com melhor desempenho. Fonte: Autoria própria.

Em seguida, com a arquitetura definida, foi executado um treinamento com os mesmos dados, porém, agora, separados em 80% para treino e 20% para teste. Nesse treinamento a taxa de aprendizagem foi de 0,001 e foram executadas 160 épocas, o número de épocas foi escolhido aleatoriamente sendo maior que 10, pois a partir dessa época foi percebido que não há nenhum ganho e não há grandes alterações na perda calculada. O restante dos parâmetros foram mantidos iguais ao experimento anterior. Na figura 33 é possível visualizar o histórico de treinamento para ambos os dados (treinamento e teste), é perceptível que a partir da época 10 não houve nenhum ganho e a perda calculada se manteve próximo ao resultado final. Outro aspecto importante é que a partir desses gráficos constata-se que não ocorreu sobreajuste na rede treinada.

Finalmente, depois de treinada, a rede pôde ser aplicada a planta real. As entradas da rede são calculadas antes de seu processamento, já a saída só necessita do arredondamento para que seja enviada ao microcontrolador e aplicada ao PWM do motor. Tudo isso acontece na frequência estipulada: 100Hz.

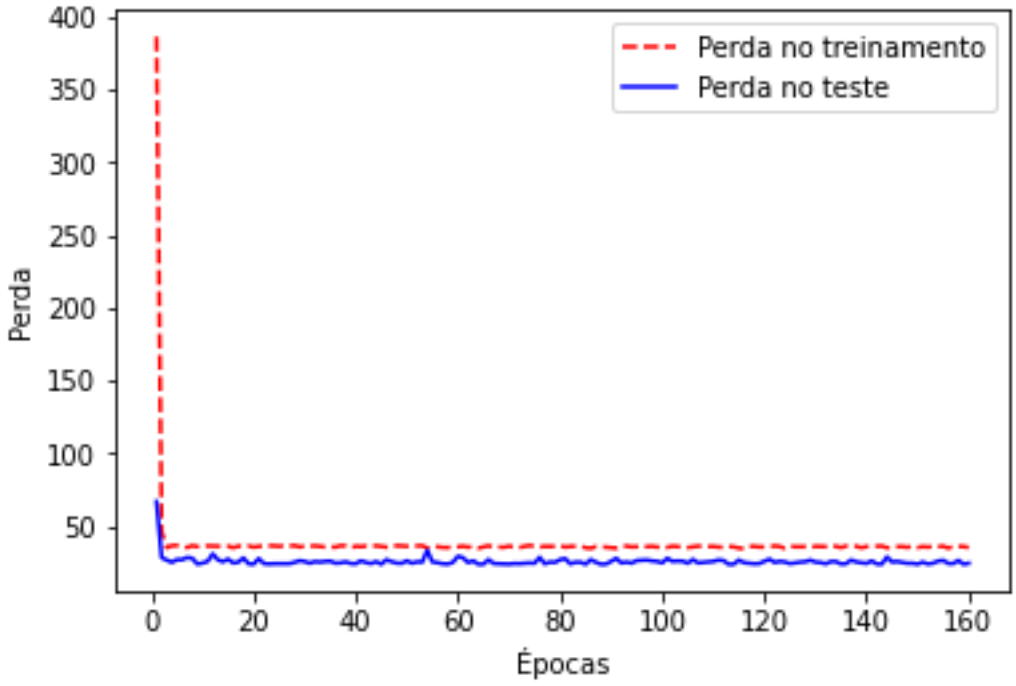

Figura 33 – Histórico de perdas dos dados de treinamento e teste. Fonte: Autoria própria.

4 Resultados e discussão

Para manter um padrão e conseguir comparar o resultado de todos os controladores, foram escolhidos de maneira aleatória 5 números inteiros entre 90 e 200 para serem aplicados como *setpoint* a malha de controle. Na figura 34 é possível visualizar o sinal de referência utilizado.

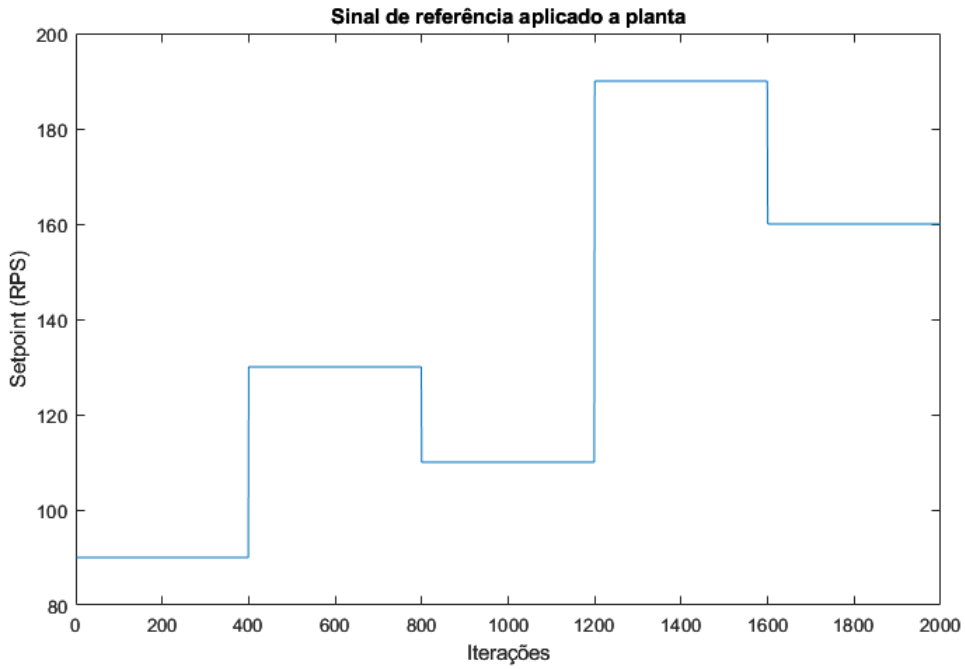

Figura 34 – Sinal de referência aplicado em todos os controladores para comparação. Fonte: Autoria própria.

Em seguida, foi gerado um sinal de referência que consiste em um só degrau de 100 a 150 como pode-se ver na figura 35.

4.1 Controlador PID

Ao aplicar o controlador PI projetado na malha com o sinal de referência exposto na figura 34 o sistema teve a resposta mostrada na figura 36. Pode-se observar que o sistema comportou-se bem conseguindo se estabilizar em todos os valores do sinal de referência estabelecidos em um tempo inferior ao do degrau aplicado (4 segundos). Em todos os degraus aplicados ocorreu *overshoot*, sendo que a média entre eles foi de aproximadamente 17%. Por fim, o MSE entre a resposta do sistema e o sinal de referência foi de 56,4500 RPS.

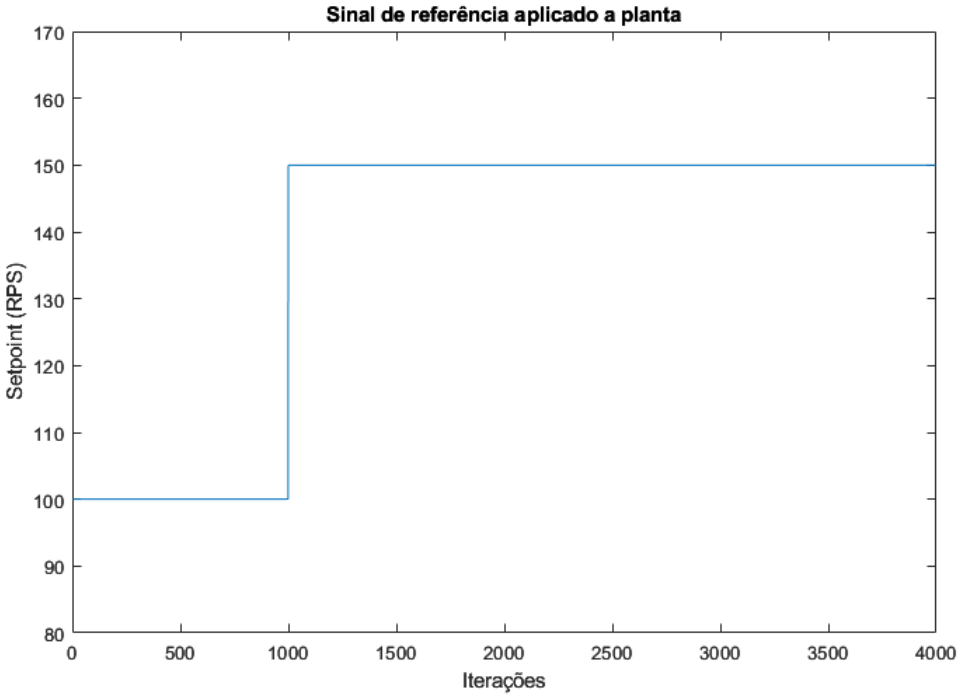

Figura 35 – Degrau no sinal de referência aplicado em todos os controladores para comparação. Fonte: Autoria própria.

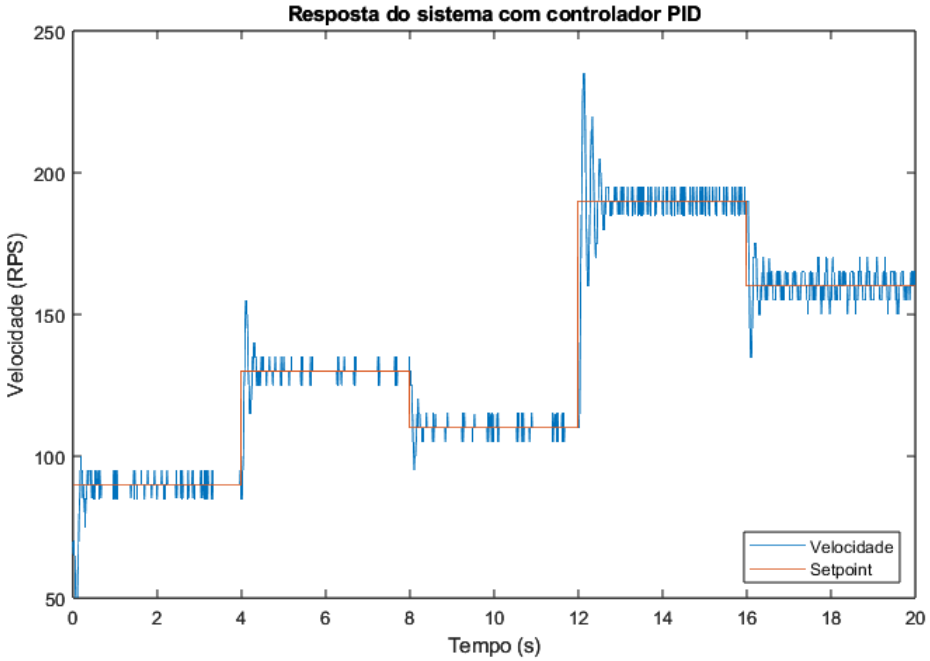

Figura 36 – Resposta do sistema com o controlador PI e *setpoint* estipulado. Fonte: Autoria própria.

Na figura 37 pode-se ver o sinal de erro durante o experimento que permaneceu próximo a zero e teve picos nos momentos dos degraus, conforme o esperado. Na figura 38 é possível visualizar o sinal de controle, ou, nesse caso, o sinal de PWM enviado a ponte H. Nota-se que o sinal varia de acordo com o *setpoint* e é ruidoso devido a leitura do sensor ser ruidosa o que insere um ruído em todos os sinais que dependem dessa leitura como o do erro e, no PID, o de controle.

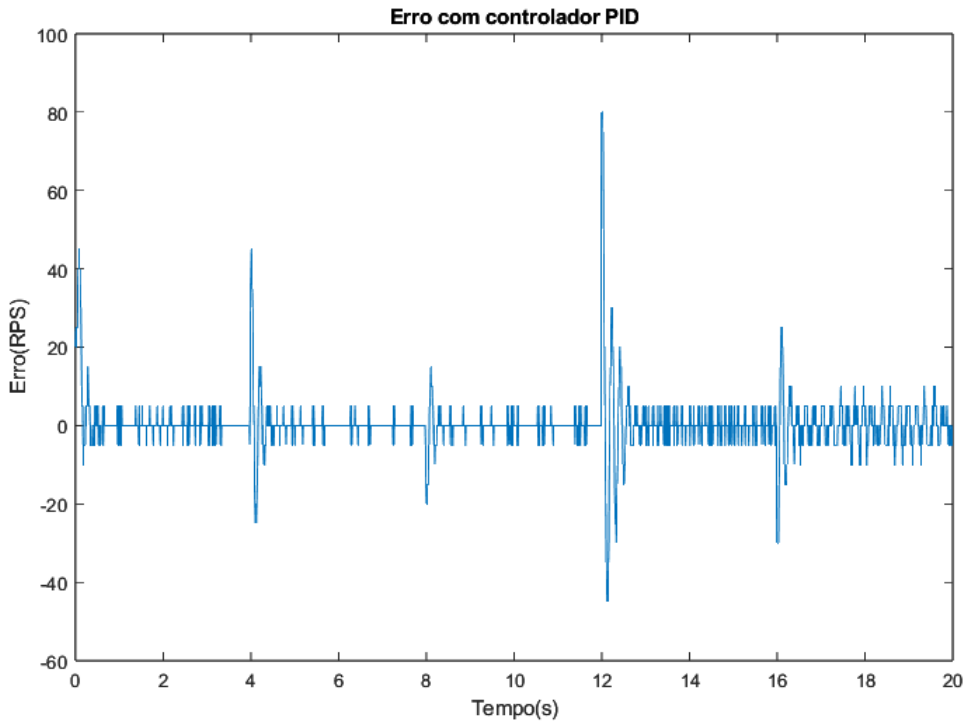

Figura 37 – Sinal de erro do sistema com o controlador PI e *setpoint* estipulado. Fonte: Autoria própria.

Ao aplicar o controlador PI projetado na malha com o sinal de referência com um degrau exposto na figura 35 o sistema teve a resposta mostrada na figura 39. É fácil observar que o sistema se comportou bem, conseguindo se estabilizar de maneira relativamente rápida e com *overshoot* de 20% do sinal de referência, ou seja, com o sinal de referência sendo 150 RPS, a velocidade máxima que o sistema atingiu até se estabilizar foi de 180 RPS. Além disso, o MSE entre a resposta do sistema e o sinal de referência foi de 14,4688 RPS.

Pode-se ver na figura 40 que o sinal de erro comportou-se também, como no experimento anterior, conforme o esperado, permanecendo em torno de zero e tendo picos durante a variação do *setpoint*. A figura 41 refere-se ao sinal de controle aplicado a planta que também se assemelhou muito a experiência anterior, variando de acordo com o *setpoint* e bastante ruidoso pelo mesmo motivo.

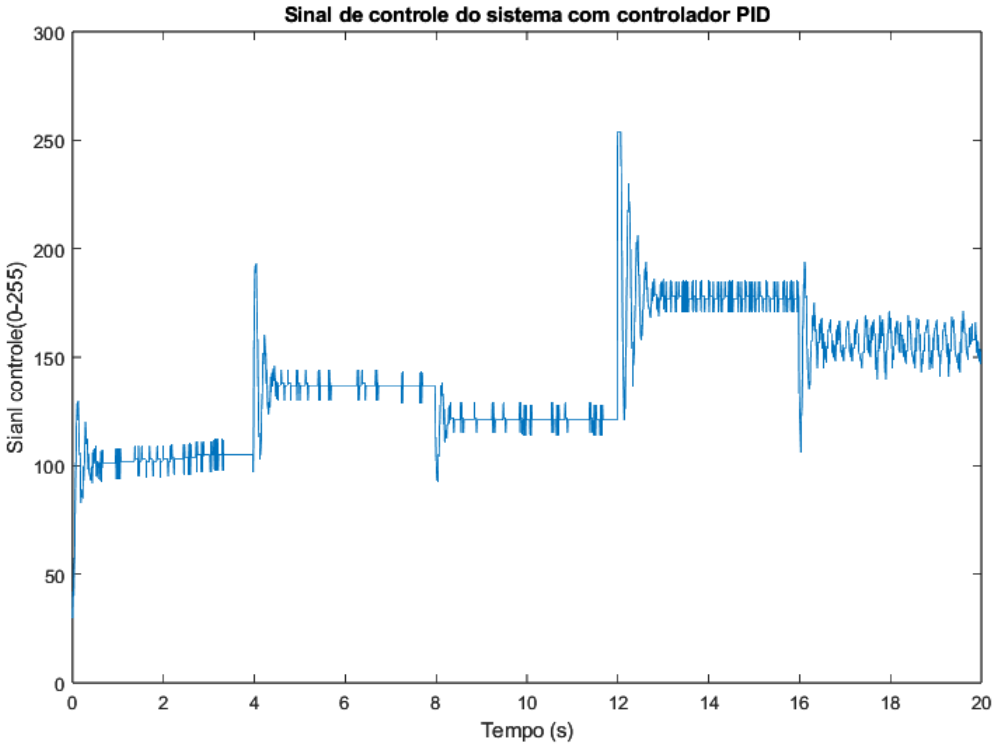

Figura 38 – Sinal de controle enviado ao sistema com o controlador PI com o *setpoint* estipulado. Fonte: Autoria própria.

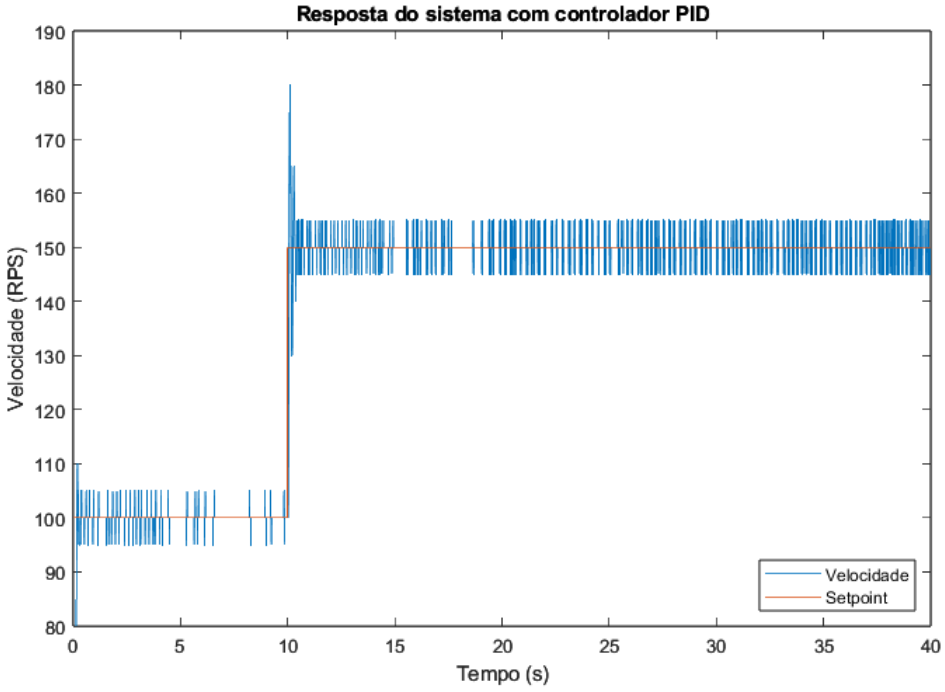

Figura 39 – Resposta do sistema com o controlador PI sendo aplicado um degrau no *setpoint*. Fonte: Autoria própria.

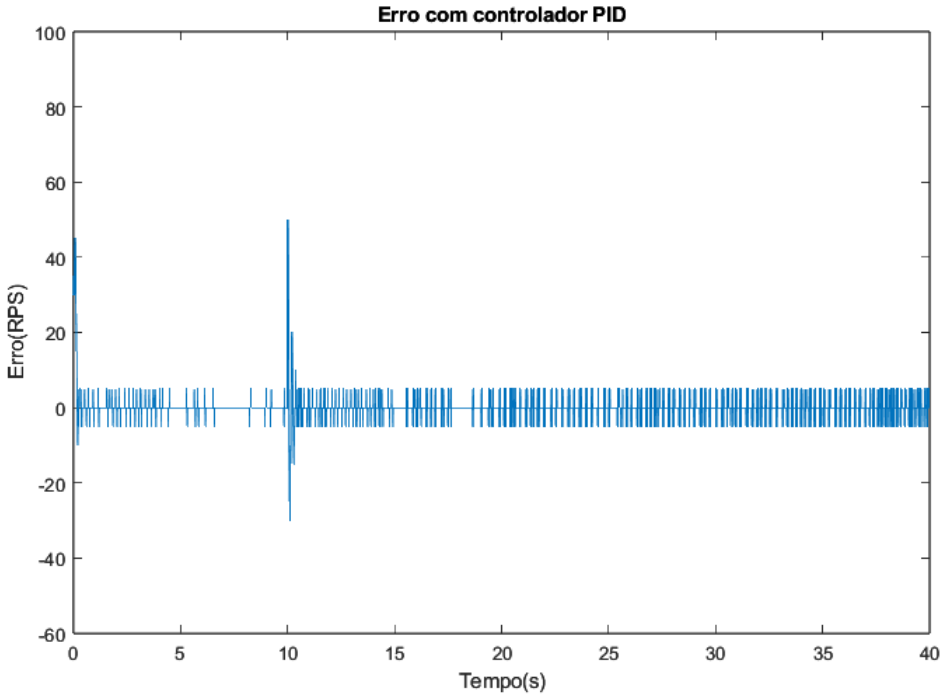

Figura 40 – Sinal de erro do sistema com o controlador PI quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

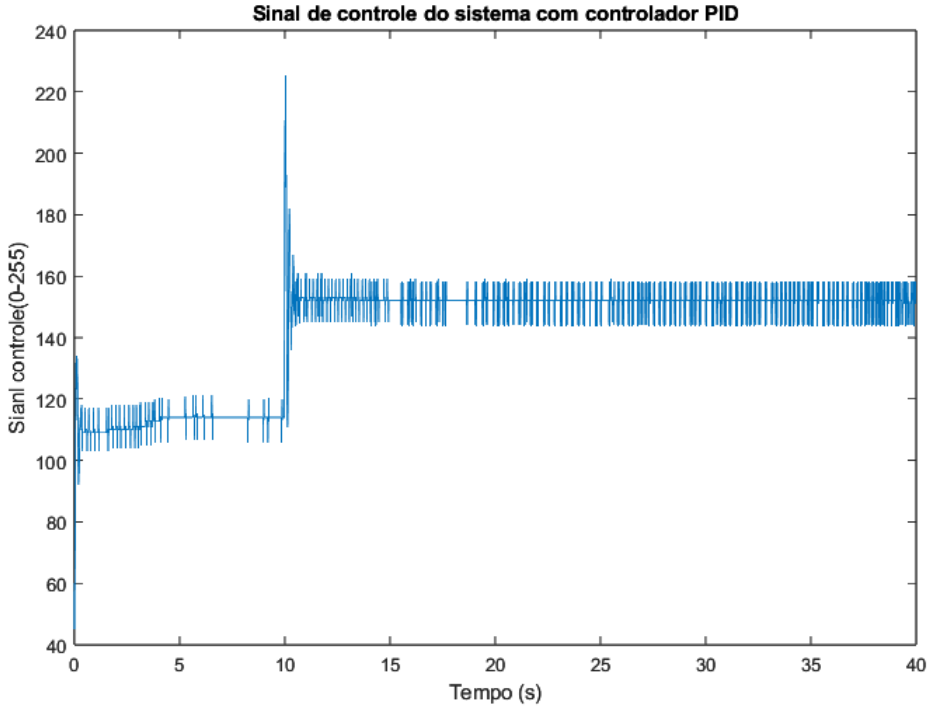

Figura 41 – Sinal de controle enviado ao sistema com o controlador PI quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

4.2 Controlador MPC

Seguindo a mesma ordem de experimentos, foi aplicado o sinal de referência exposto na figura 34 no sistema com o controlador MPC, a resposta do sistema está exposta na figura 42. Percebe-se que para *setpoints* abaixo de 150 o sistema se comportou bem conseguindo se estabilizar de maneira rápida e sem *overshoot*. No entanto quando foram aplicados sinais de referência maiores, no caso 190 e 160, é possível perceber que começa a existir uma dificuldade para a estabilização. Existe um *overshoot* nos dois degraus, o sistema demora mais para se estabilizar e, além disso, a estabilização não garante o erro nulo. Provavelmente isso é consequência de como foi modelado o sistema: primeiramente, foi considerado um sistema de primeira ordem, e além disso, o degrau para o método da curva de reação gerou uma resposta com velocidade abaixo de 150, ou seja, o sistema modelado simula muito bem o sistema real somente nessa faixa de velocidade e, como o MPC foi baseado nesse modelo, ele não opera muito bem longe desse intervalo de velocidade. Por fim, o MSE entre a resposta do sistema e o sinal de referência foi de 38,1750 RPS.

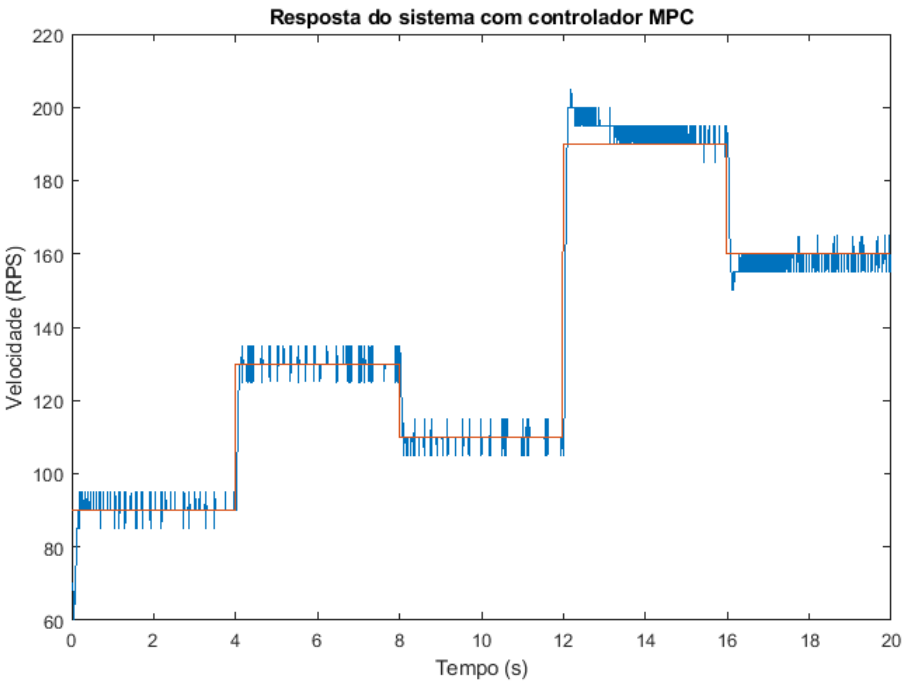

Figura 42 – Resposta do sistema com o controlador MPC e *setpoint* estipulado. Fonte: Autoria própria.

Na figura 43 é possível perceber que o erro durante a maior parte do experimento permaneceu muito próximo a zero e teve picos nos momentos dos degraus. No entanto, o erro durante os degraus acima de 150 (a partir do segundo 12) manteve-se levemente diferente de zero tendo picos também nos momentos de mudança no sinal de referência. Pode-se perceber que o sinal de controle, exposto na figura 44, acompanha o sinal de referência tendo picos nos momentos de alteração, conforme o esperado. Uma grande

diferença entre o controlador PID e o MPC é que o sinal de controle no MPC é muito menos ruidoso e se estabiliza rápido. Essa rejeição a ruídos pode ser interessante em diversos sistemas em que o ruído é um problema.

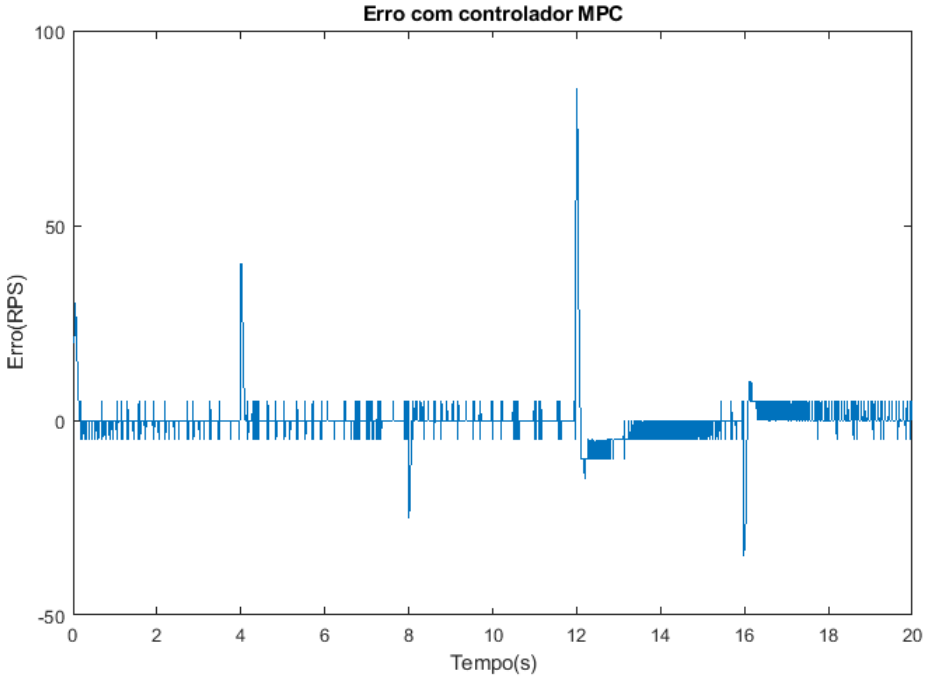

Figura 43 – Sinal de erro do sistema com o controlador MPC e *setpoint* estipulado. Fonte: Autoria própria.

Ao aplicar o controlador MPC projetado na malha com o sinal de referência com um degrau exposto na figura 35 o sistema teve a resposta ilustrada na figura 45. É fácil observar que o sistema se comportou bem, conseguindo se estabilizar de maneira rápida e sem *overshoot*. Além disso, o MSE entre a resposta do sistema e o sinal de referência foi de 10,9313 RPS.

Na figura 46 pode-se ver que o sinal de erro durante o experimento permaneceu muito próximo a zero e teve pico no momento de alteração do *setpoint*, conforme o esperado. É possível visualizar na figura 47 o sinal de controle enviado ao sistema. Pode-se perceber que, assim como no experimento anterior, o sinal é bem menos ruidoso quando comparado ao controlador PI.

4.3 Rede neural baseada no MPC

A rede treinada obteve um valor final de *loss* com os dados de treinamento igual a 35,1603, já com os dados de teste ou validação esse valor foi de 24,0770 confirmando que não houve sobreajuste ou subajuste no treinamento. Como a arquitetura escolhida prediz o sinal de controle que deve ser enviado como PWM ao motor, é necessário analisar e comparar a saída da rede e a saída do controlador MPC com o qual ela foi treinada considerando

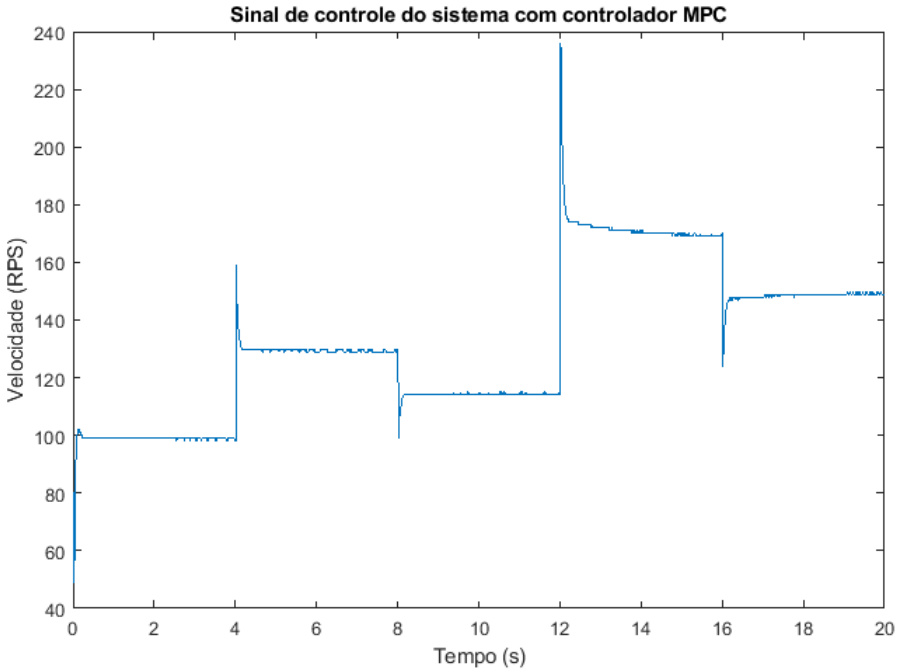

Figura 44 – Sinal de controle enviado ao sistema com o controlador MPC com o *setpoint* estipulado. Fonte: Autoria própria.

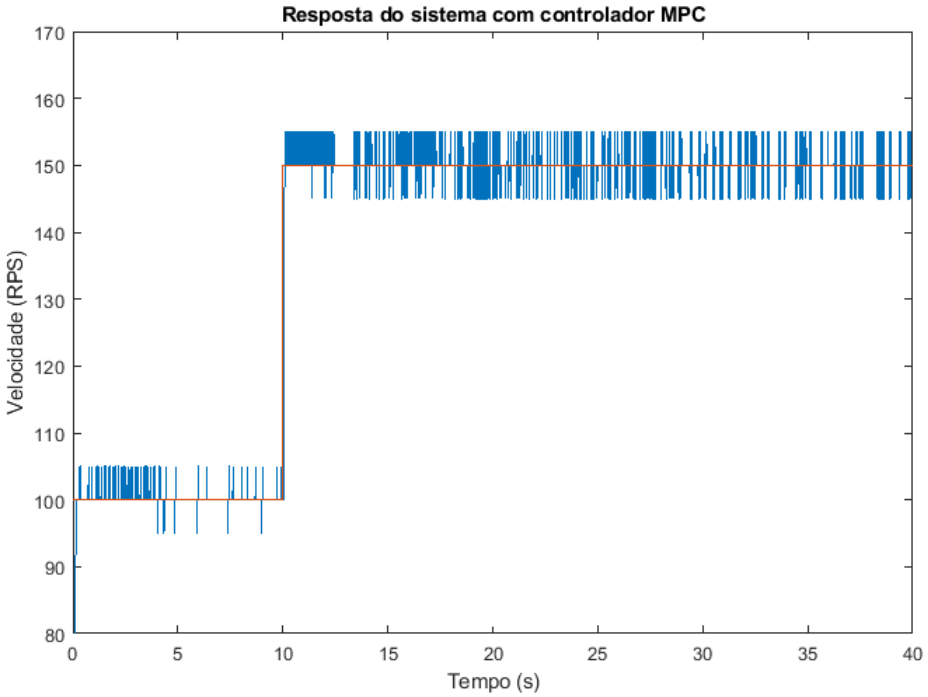

Figura 45 – Resposta do sistema com o controlador MPC sendo aplicado um degrau no *setpoint*. Fonte: Autoria própria.

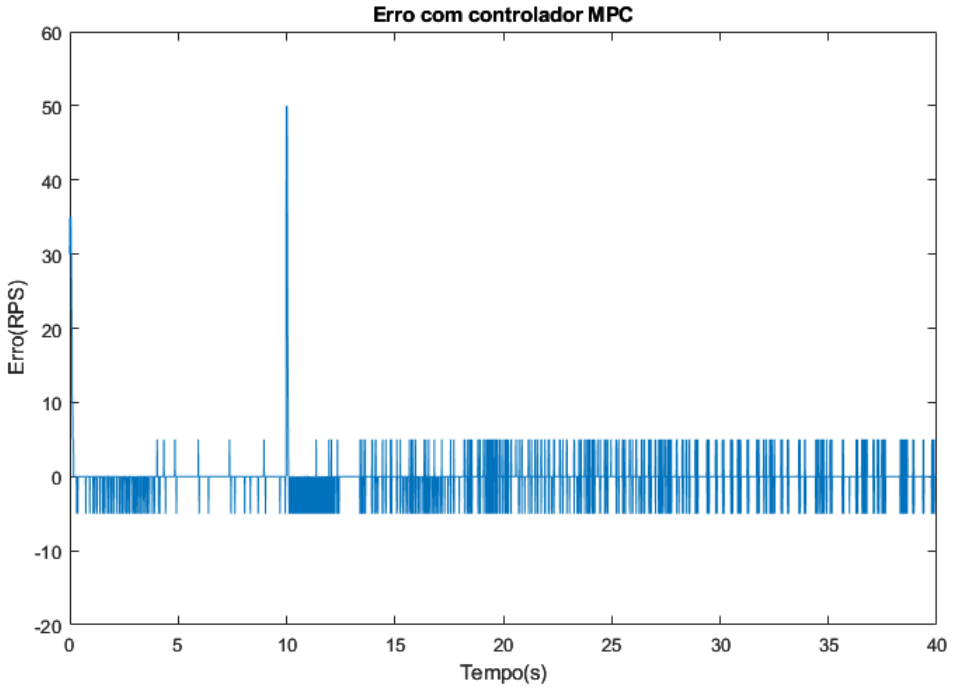

Figura 46 – Sinal de erro do sistema com o controlador MPC quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

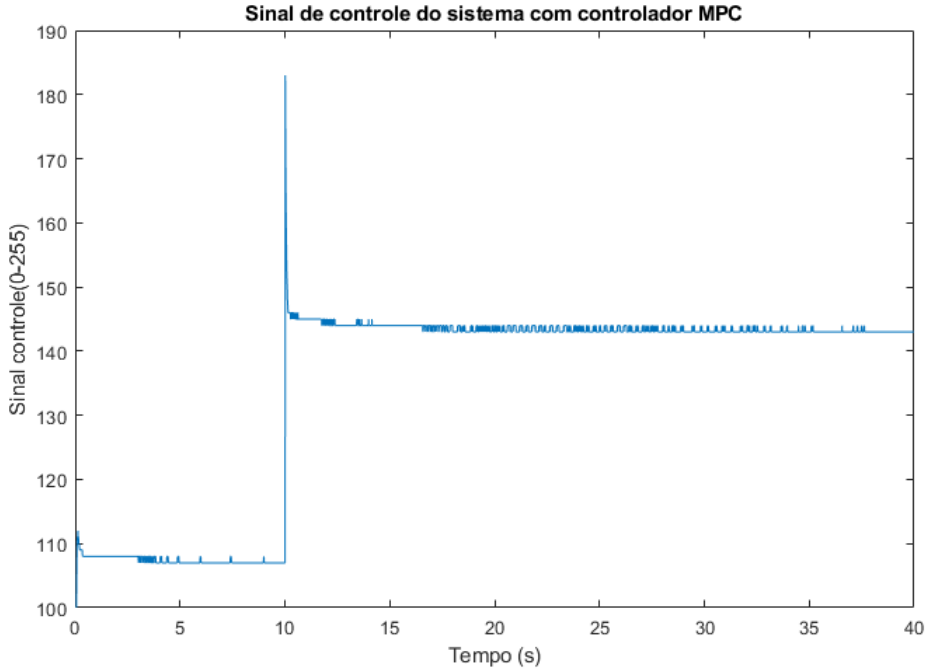

Figura 47 – Sinal de controle enviado ao sistema com o controlador MPC quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

os mesmos dados de entrada. Na figura 48 estão expostas ambas as saídas. É possível perceber que a rede prediz valores pouco abaixo dos esperados e que nos momentos de mudança do *setpoint* a rede não acompanha muito bem o sinal do controlador. Assim, espera-se que, ao aplicar a rede no sistema, a velocidade se estabilize um pouco abaixo do *setpoint* e que o sistema não trate muito bem os períodos de transição do sinal de referência podendo acarretar um tempo de acomodação maior.

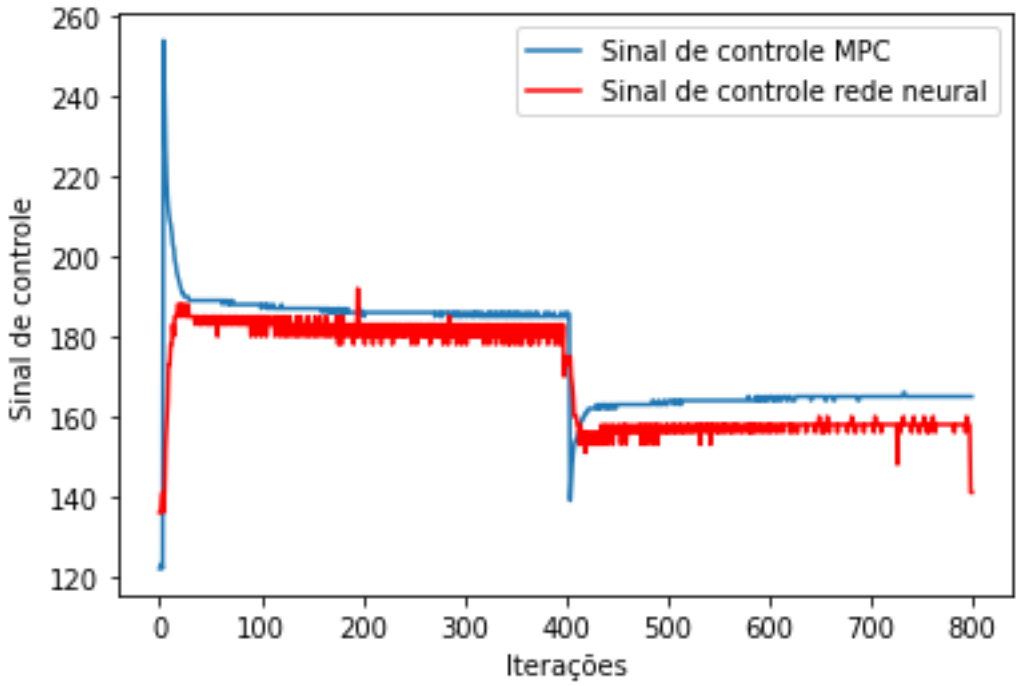

Figura 48 – Sinais de controle do controlador MPC e da rede neural baseada nele, com os mesmos dados de entrada. Fonte: Autoria própria.

Ao aplicar a rede neural treinada na malha com o sinal de referência exposto na figura 34 o sistema teve a resposta mostrada na figura 49. Percebe-se que a resposta do sistema fica acima do *setpoint* em todos os valores testados diferente do esperado, além disso, nos valores de 90 e 190 o erro é maior, e isso pode ser consequência também do que acontece com o MPC. É possível ver que o tempo de acomodação em relação ao PI e o MPC é bem superior, resultado esperado visto que nos momentos de mudança do *setpoint* a rede não acompanha muito bem o sinal do controlador MPC. Por fim, o MSE entre a resposta do sistema e o sinal de referência foi de 113,5125 RPS o que indica que ainda há margem para melhoria no projeto do controlador.

Na figura 50 é possível perceber que o erro durante a maior parte do experimento não fica em torno de zero e teve picos nos momentos dos degraus. O erro durante os degraus de 90 e 190 manteve-se mais alto que nos degraus de 110, 130 e 160. Pode-se perceber que o sinal de controle, exposto na figura 51, acompanha o sinal de referência e não tem picos nos momentos de alteração, conforme o esperado o que ocasiona um maior

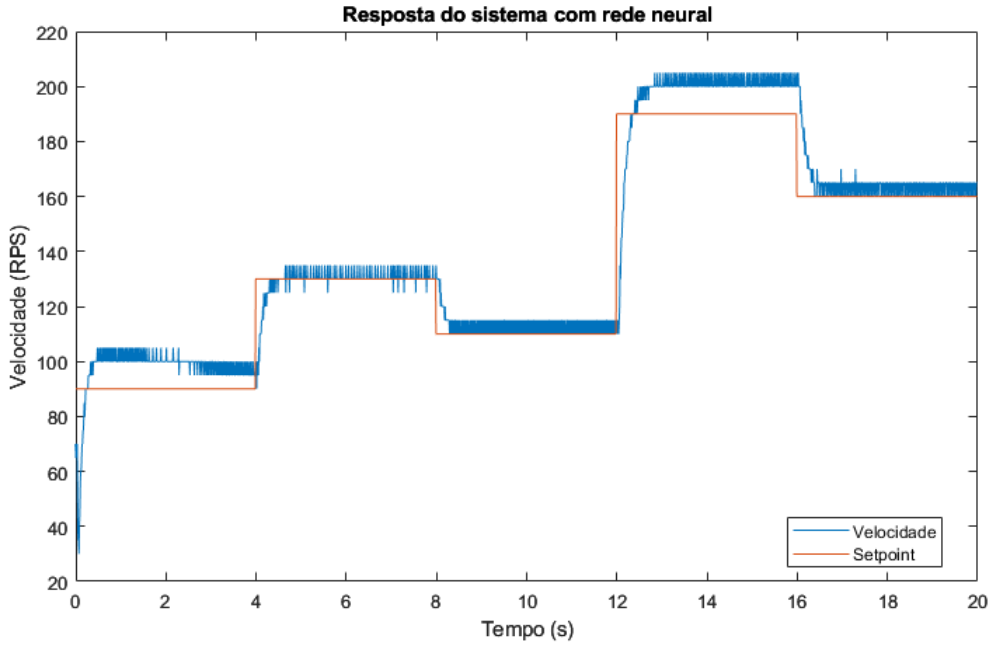

Figura 49 – Resposta do sistema com o controlador por rede neural e *setpoint* estipulado. Fonte: Autoria própria.

tempo de acomodação. No entanto, assim com o sinal de controle no MPC, o sinal é muito menos ruidoso. Essa rejeição a ruídos veio com o aprendizado a partir do MPC e pode ser interessante em diversos sistemas em que o ruído é um problema.

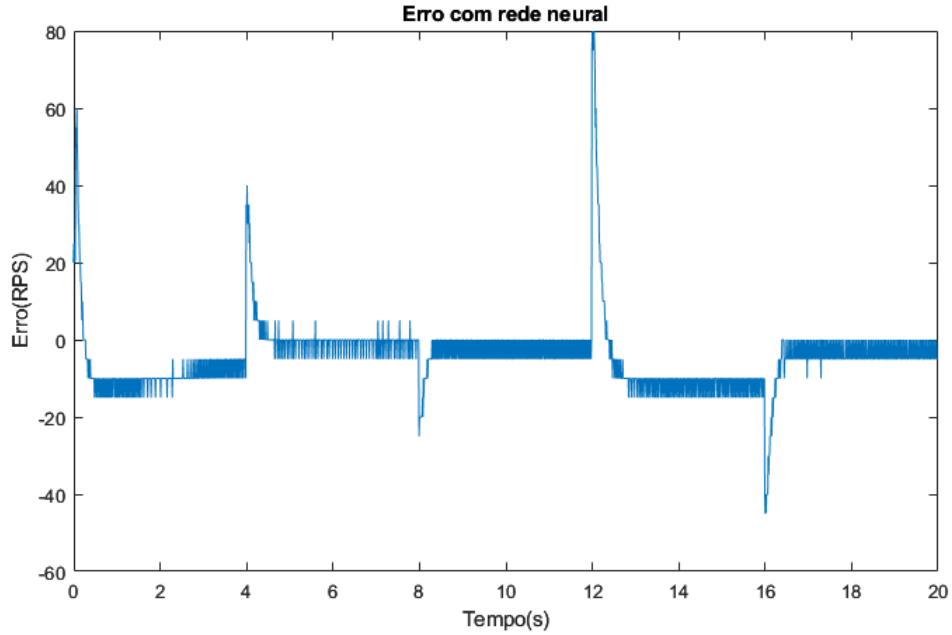

Figura 50 – Sinal de erro do controlador baseado em rede neural e *setpoint* estipulado. Fonte: Autoria própria.

Ao aplicar o controlador baseado na rede neural na malha com o sinal de referência sendo o degrau exposto na figura 35, o sistema teve a resposta ilustrada pela figura

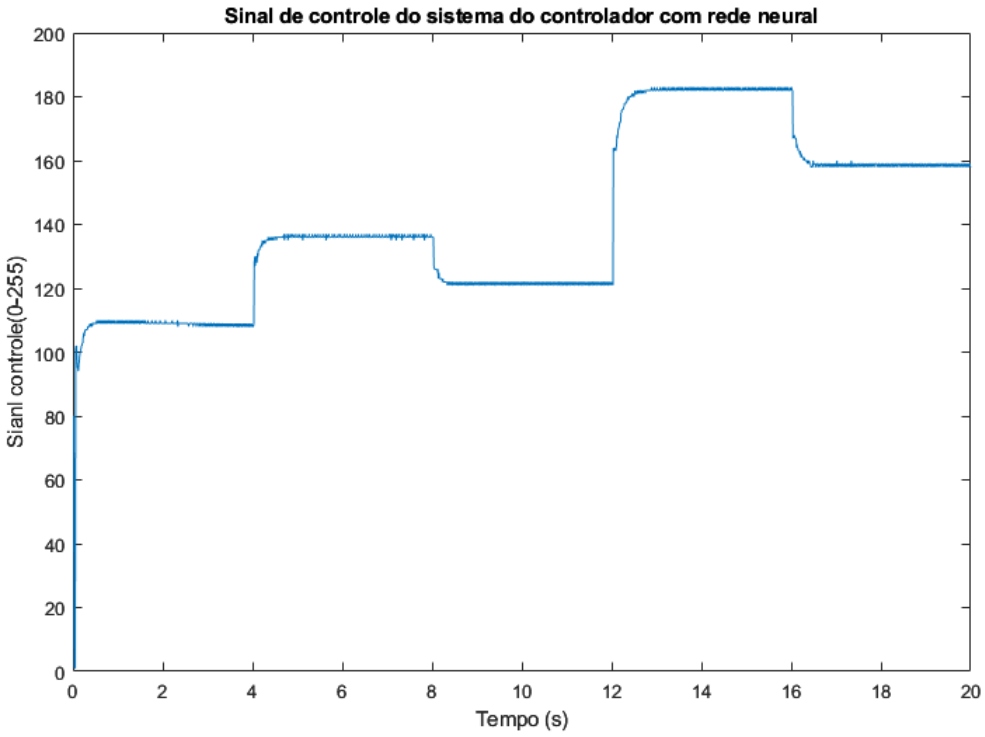

Figura 51 – Sinal de controle enviado ao sistema com o controlador baseado em rede neural com o *setpoint* estipulado. Fonte: Autoria própria.

52. Pode-se perceber que, tanto no primeiro como no segundo degrau, o sistema tentou chegar ao *setpoint* porém ficou abaixo do mesmo sem gerar *overshoot*, provocando um erro positivo. Além disso, é possível analisar que o tempo para a estabilização foi maior quando comparado com os mesmos experimentos realizados no MPC e PI. Por fim, o MSE entre a resposta do sistema e o sinal de referência foi de 40,0250 RPS o que também indica que ainda há margem para melhoria no projeto do controlador.

Na figura 53 pode-se ver que o sinal de erro durante o experimento permaneceu positivo durante a maior parte do experimento e teve pico no momento de alteração do *setpoint*, a depender da precisão e exatidão desejada ainda pode ser viável o uso desse controlador. É possível visualizar na figura 54 o sinal de controle enviado ao sistema. Pode-se perceber que, assim como no experimento anterior, o sinal é bem menos ruidoso quando comparado ao controlador PI.

Todos os *scripts* utilizados no trabalho estão disponíveis no repositório do gitHub.

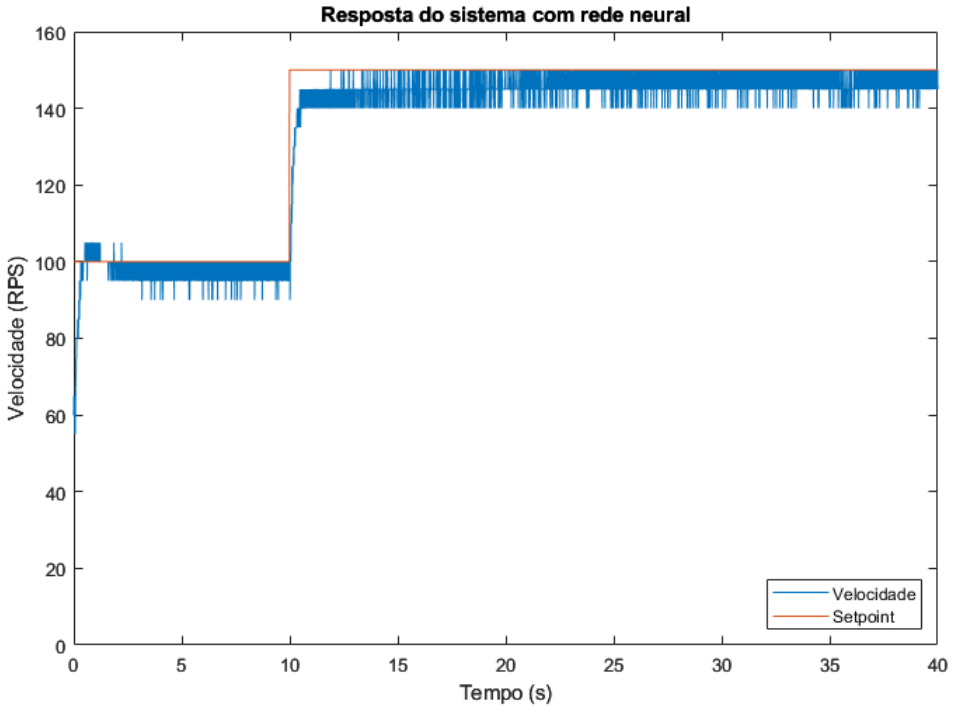

Figura 52 – Resposta do sistema com o controlador baseado na rede neural sendo aplicado um degrau no *setpoint*. Fonte: Autoria própria.

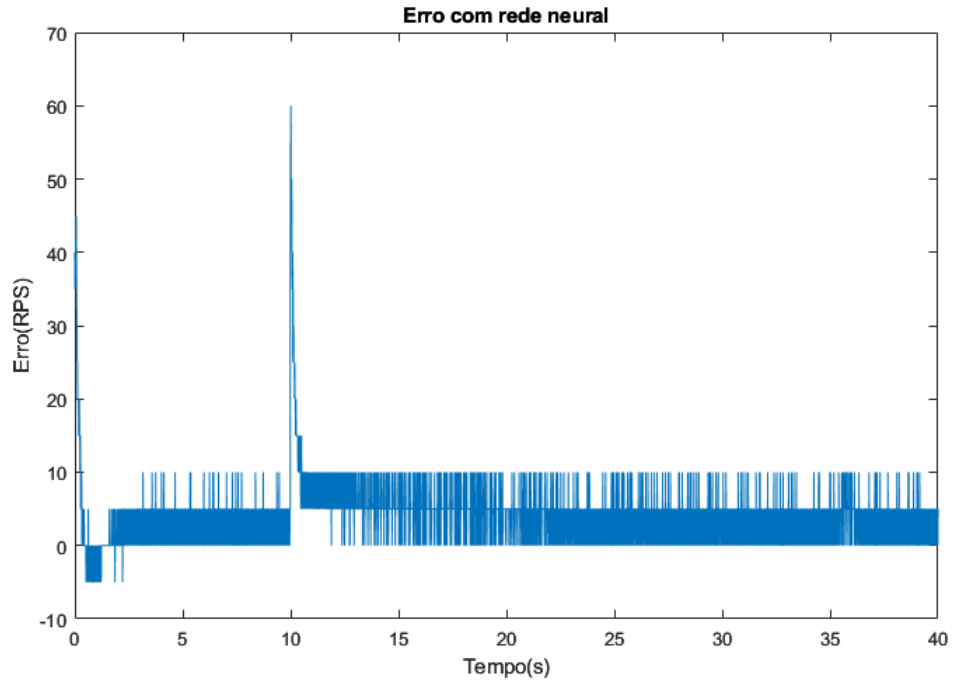

Figura 53 – Sinal de erro do sistema com o controlador MPC quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

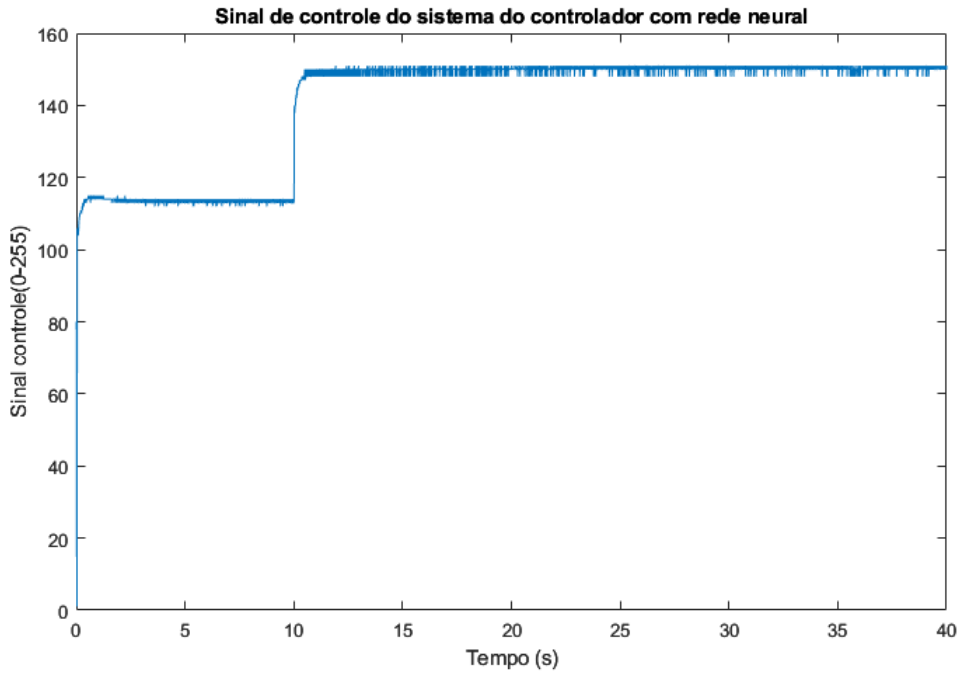

Figura 54 – Sinal de controle enviado ao sistema com o controlador MPC quando foi aplicado um degrau ao *setpoint*. Fonte: Autoria própria.

Considerações finais

O sistema de acionamento e controle do motor de corrente contínua projetado foi comprovado ser válido permitindo a aplicação e validação de diferentes controladores que visam controlar a velocidade do motor. Dessa forma, foi possível aplicar e obter os resultados de uma técnica de controle clássico: o PID, uma de controle ótimo: o MPC, e uma de controle inteligente utilizando redes neurais. Com os resultados também foi possível realizar uma análise comparativa entre os três destacando pontos positivos e negativos de cada técnica.

O controlador PID obteve bons resultados conseguindo se estabilizar em todos os cenários ocorrendo um *overshoot* de, em média, 18%. Por fim, o MSE calculado foi de 56,45 RPS no primeiro e 14,47 RPS no segundo experimento. Sendo assim, o controlador PID se mostrou capaz de ser aplicado em sistemas em que pode ocorrer *overshoot* conseguindo um tempo de resposta aceitável. É válido ressaltar que tal controlador funciona bem para controle de sistemas lineares, invariantes no tempo e com uma única entrada e única saída (do inglês *single-input and single-output* - SISO), no entanto, pode não ser apropriado para sistemas mais complexos.

O controlador MPC também obteve resultados satisfatórios conseguindo estabilização em todos os casos. No entanto, em alguns casos o MPC teve *overshoot* e um tempo de estabilização maior. Isso provavelmente foi consequência de como foi modelado o sistema. Por fim, o MSE no primeiro experimento foi 38,18 RPS e no segundo 10,93 RPS, ambos resultados melhores que o PID. O principal ponto positivo desse controlador é que ele é capaz de controlar diversos sistemas com plantas mais complexas. Em contrapartida, o alto custo computacional é o grande ponto negativo para a aplicação deste controlador.

O controlador por rede neural foi treinado a partir de dados do MPC, por isso trouxe também a variação da eficácia de acordo com a faixa de *setpoint* escolhida. Tal controlador obteve resultados não tão satisfatórios com um erro diferente de 0 e tempo de estabilização maior que o PID e o MPC. No entanto, percebe-se que a rede conseguiu assimilar boa parte do comportamento do MPC mostrando-se capaz de controlar um sistema que não depende de rápida estabilização. Por fim, o MSE no primeiro experimento foi de 113,5125 RPS e no segundo 40,03 RPS, ambos resultados piores que os dois controladores analisados. Foi possível perceber que uma rede neural treinada de maneira correta consegue assimilar o comportamento de um controlador com alta complexidade computacional diminuindo consideravelmente esse custo.

Como sugestões para próximos trabalhos destaca-se: a aplicação desses controladores em sistemas mais complexos, mostrando o comportamento de cada um; a investigação de outras formas de treinamento para redes neurais como, por exemplo, a utilização de ou-

tras entradas, visando melhorar o desempenho do controlador e a aplicação do controlador baseado em rede neural diretamente em um microcontrolador. Além disso, seria interessante avaliar a implementação de controladores híbridos, que combinem as vantagens de diferentes técnicas.

Referências

ANGELICO, B. A.; SCALASSARA, P. R.; VARGAS, A. N. *Princípios de controle*. Acessado em: 26 jan. 2023. 2023. Disponível em: http://paginapessoal.utfpr.edu.br/avargas/courses-1/principios_de_controle/principios_de_controle/principiosCap10.pdf. Citado 1 vez na página 22.

BILOBROVEC, Marcelo et al. Implementação de um sistema de controle inteligente utilizando a lógica fuzzy. *XI SIMPEP, Bauru/Brasil*, p. 42, 2004. Citado 1 vez na página 12.

BOHN, C.; ATHERTON, D. P. An analysis package comparing PID anti-windup strategies. *IEEE Control Systems Magazine*, v. 15, n. 2, p. 34–40, 1995. DOI: 10.1109/37.375281. Citado 1 vez na página 23.

CARVALHO, D. et al. Controlador Preditivo Otimizado Aplicado ao Controle de Velocidade de Motor CC, 2015. Citado 3 vezes nas páginas 16, 23, 24.

COELHO, Bruno França et al. Sintonia online de Controladores PID Sintonizado por Regras Fuzzy em Sistemas Industriais com Aplicaçao em Processo Operacional de Viradores de Vagoes. In: 1. CONGRESSO Brasileiro de Automática-CBA. 2019. v. 1. Citado 2 vezes nas páginas 13, 29, 30.

DRUMMOND, Adriana de C; OLIVEIRA, KC; BAUCHSPIESS, Adolfo. Estudo do Controle de Pêndulo Inverso sobre Carro utilizando Rede Neural de Base Radial. In: IV Congresso Brasileiro de Redes Neurais. 1999. Citado 1 vez na página 12.

HAYKIN, Simon. *Redes neurais: princípios e prática*. Bookman Editora, 2001. Citado 2 vezes na página 26.

KUSHWAH, Manoj; PATRA, Ashish. PID controller tuning using Ziegler-Nichols method for speed control of DC motor. *International Journal of Scientific Engineering and Technology Research*, v. 3, n. 13, p. 2924–2929, 2014. Citado 1 vez na página 23.

LOURENÇO, João. Sintonia de controladores PID. *Escola superior de tecnologia*, 1997. Citado 1 vez na página 22.

MITCHELL, Tom et al. Machine learning. *Annual review of computer science*, Annual Reviews 4139 El Camino Way, PO Box 10139, Palo Alto, CA 94303-0139, USA, v. 4, n. 1, p. 417–433, 1990. Citado 1 vez na página 26.

NETO, Antônio Hadade. *Técnicas anti-windup em estruturas de controle PID, RST e GPC*. 2005. F. 153. Diss. (Mestrado) – Universidade Federal de Santa Catarina, Florianópolis. Citado 1 vez na página 23.

OGATA, Katsuhiko. *Engenharia de controle moderno*. São Paulo: Pearson Education do Brasil, 2010. v. 5. Citado 4 vezes nas páginas 12, 19, 21, 22.

OLIVEIRA, Arthur Dimitri Brito et al. Controlador preditivo não linear embarcado para aeropêndulo utilizando rede neural artificial. Universidade Federal de Campina Grande, 2021. Citado 8 vezes nas páginas 25, 27, 28, 30–32, 42, 43.

OLIVEIRA, Vilma Alves de; VARGAS, Jerson B et al. Laboratório de sistemas de controle. EESC/USP, 1997. Citado 1 vez na página 19.

OROZCO, Orlando Arrieta; RUIZ, Víctor M Alfaro. Sintonización de controladores PI y PID utilizando los criterios integrales IAE e ITAE. *Ingeniería*, v. 13, n. 1-2, p. 31–39, 2003. Citado 1 vez na página 23.

PEREIRA, Ricardo H et al. Geração Distribuída de Energia Elétrica–Aplicação de Motores Bicombustível Diesel/Gás Natural. In: 3º Congresso Brasileiro de P&D em Petróleo e Gás, Salvador–BA. 2005. Citado 1 vez na página 15.

PEREIRA JUNIOR, Claudionor Ferreira; COSTA, Álvaro José da Silva; SILVA JÚNIOR, José Ribamar Ribeiro. Sistema inteligente de sintonia de controlador PID para controle de nível de um destilador de água com alimentação constante. Universidade Federal do Maranhão, 2018. Citado 1 vez na página 28.

RAUBER, Thomas Walter. Redes neurais artificiais. *Universidade Federal do Espírito Santo*, v. 29, 2005. Citado 2 vezes nas páginas 26, 27.

RECK, Rebecca M. Validating DC motor models on the Quanser Qube Servo. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. DYNAMIC Systems and Control Conference. 2018. v. 51906, v002t16a005. Citado 1 vez na página 16.

REIS, Lucas Andery. *Controle preditivo por modelo de um circuito simulado de remoagem de minério de ferro*. 2018. F. 76. Diss. (Mestrado) – Universidade Federal de Ouro Preto, Ouro Preto. Citado 2 vezes nas páginas 24, 25, 41.

SANTANA, Adrielle. *Especificações de Sistemas de Controle*. 2022. Material de aula da matéria teoria de controle 2 do curso de engenharia de controle e automação da UFOP. Citado 1 vez na página 34.

SANTOS NETO, Accacio F dos; GOMES, Francisco José. Controladores PID: introduzindo inteligência computacional no controle industrial. In: XXXVIII COBENGE. 2010. Citado 1 vez na página 12.

SERALE, Gianluca et al. Model Predictive Control (MPC) for Enhancing Building and HVAC System Energy Efficiency: Problem Formulation, Applications and Opportunities. *Energies*, v. 11, n. 3, 2018. ISSN 1996-1073. DOI: 10.3390/en11030631. Disponível em: https://www.mdpi.com/1996-1073/11/3/631. Citado 1 vez nas páginas 24, 25.

SERALE, Gianluca et al. Model predictive control (MPC) for enhancing building and HVAC system energy efficiency: Problem formulation, applications and opportunities. *Energies*, MDPI, v. 11, n. 3, p. 631, 2018. Citado 1 vez na página 25.

SILVA FERREIRA, José Ricardo da; CAVALCANTI, José Homero Feitosa; ALSINA, Pablo Javier. Treinamento de redes neurais artificiais para o controle de motores de corrente contínua. *Revista Traços*, v. 5, n. 9, 2017. Citado 1 vez na página 13.

TEAM, Keras. *Keras documentation: Keras API reference*. Acessado em: 03 jan. 2023. 2023. Disponível em: https://keras.io/api/. Citado 1 vez na página 43.

THE MATHWORKS, Inc. *Compute optimal control action and update controller states*. Acessado em: 26 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/mpc/ref/mpc.mpcmove.html. Citado 1 vez na página 41.

THE MATHWORKS, Inc. *Convert model from continuous to discrete time*. Acessado em: 21 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/ident/ref/lti.c2d.html. Citado 2 vezes nas páginas 39, 40.

THE MATHWORKS, Inc. *Model predictive controller*. Acessado em: 26 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/mpc/ref/mpc.html. Citado 1 vez na página 40.

THE MATHWORKS, Inc. *PID tuning algorithm for linear plant model*. Acessado em: 21 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/control/ref/lti.pidtune.html. Citado 2 vezes nas páginas 23, 39.

THE MATHWORKS, Inc. *Predict responses of linear regression model*. Acessado em: 29 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/stats/linearmodel.predict.html. Citado 1 vez na página 43.

THE MATHWORKS, Inc. *State-space model*. Acessado em: 26 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/control/ref/ss.html. Citado 1 vez na página 40.

THE MATHWORKS, Inc. *Uniformly distributed pseudorandom integers*. Acessado em: 29 dez. 2022. 2023. Disponível em: https://www.mathworks.com/help/matlab/ref/randi.html. Citado 1 vez na página 42.

TORGA, Diego Santana. Desenvolvimento de uma plataforma didática para práticas de controle de velocidade de motor de corrente continua., 2016. Citado 1 vez na página 19.

ZIEGLER, John G; NICHOLS, Nathaniel B et al. Optimum settings for automatic controllers. *trans. ASME*, v. 64, n. 11, 1942. Citado 4 vezes nas páginas 16, 23, 36.

ZOGHEIB, Victor Pizzello. Simulações de acionamento de motor de corrente contínua e de motor de indução trifásico utilizando controladores do tipo PI. Universidade Estadual Paulista (Unesp), 2022. Citado 2 vezes nas páginas 15, 16.