Centro Universitário do Instituto Mauá de Tecnologia

Escola de Engenharia Mauá

Engenharia de Controle e Automação

Guilherme Nami Bortolozi (RA: 20.00333-0)

Henrique Fortuna Accorinti (RA: 20.00080-4)

Leonardo Oneda Galvani (RA: 20.00196-7)

Matheus Ferreira Palú (RA: 20.00332-3)

Simulação de controle de posição e velocidade de um motor DC com parâmetros interativos e pré-calculados

São Caetano do Sul

2023

RESUMO

O projeto descrito neste relatório abrange as disciplinas de Instrumentação, Microcontroladores e Sistemas Microcontrolados e Sistemas de Controle I, e consiste em desenvolver um simulador interativo para controle de velocidade e posição de um motor acoplado a uma roda de inércia. Este simulador permitirá experimentar e compreender conceitos importantes relacionados a essas disciplinas, como os equipamentos adequados para fazer a instrumentação do sistema, a programação de microcontroladores com o intuito de otimizar a aplicação e a identificação matemática do sistema junto com a proposta de controle do mesmo. Os principais objetivos do projeto incluem a criação de um ambiente virtual para que seja possível aplicar conceitos teóricos aprendidos nas disciplinas de forma prática. Isso envolverá o uso do PIC16F18877 para controlar o motor, bem como a implementação e validação dos sistemas de controle que regulam a velocidade e a posição da roda de inércia. Em resumo, o projeto combina hardware e software para criar um simulador interativo em que é possível definir, através da aplicação feita em Python, qual controlador será simulado.

Palavras-chave: *Microcontrolador*. *Instrumentação*. *Sistemas dinâmicos. Controle. Python*.

ABSTRACT

The project described in this report covers the subjects of Instrumentation, Microcontrollers and Microcontrolled Systems and Control Systems I and consists of developing an interactive simulator for controlling the speed and position of a motor coupled to an inertia wheel. This simulator will make it possible to experiment with and understand important concepts related to these disciplines, such as the appropriate equipment for instrumenting the system, programming microcontrollers in order to optimize the application and the mathematical identification of the system together with the proposal for controlling it. The main objectives of the project include the creation of a virtual environment so that it is possible to apply theoretical concepts learned in the subjects in a practical way. This will involve using the PIC16F18877 to control the motor, as well as implementing and validating the control systems that regulate the speed and position of the flywheel. In summary, the project combines hardware and software to create an interactive simulator in which it is possible to define, through the Python application, which controller will be simulated.

Keywords: Microcontroller. Instrumentation. Dynamic Systems. Control. Python.

.

Sumário

1 INTRODUÇÃO

A integração das disciplinas de Instrumentação, Microcontroladores e Sistemas de Controle I neste projeto técnico representa um marco significativo na abordagem interdisciplinar da engenharia, permitindo uma aplicação prática e completa dos conceitos aprendidos ao longo do curso. Neste relatório, será apresentado o desenvolvimento de um simulador interativo destinado ao controle de velocidade e posição de um motor acoplado a uma roda de inércia. Este simulador não apenas oferece a oportunidade de experimentar e compreender conceitos fundamentais dessas disciplinas, mas também ilustra como a convergência de hardware e software pode criar soluções práticas e educativas.

Este projeto consiste no desenvolvimento de um simulador interativo que controlará a posição angular e velocidade de uma roda de inércia acoplada a um motor DC. O projeto foi desenvolvido utilizando os conhecimentos das matérias estudadas na 4ª Série do curso de engenharia de controle e automação do Instituto Mauá de Tecnologia, sendo estas: Programação Orientada a Objetos e Banco de Dados, Instrumentação, Microcontroladores e Sistemas de Controle. Este simulador permitirá experimentar e compreender conceitos importantes relacionados a essas disciplinas.

Para isso, existirão algumas fases deste projeto, sendo elas: o desenvolvimento do ambiente virtual, que incluirá duas etapas. A primeira parte consiste no controle interativo dos ganhos de um controlador PID que controlará a posição em graus (°) e a velocidade em rotações por minuto (RPM). Na segunda etapa, haverá uma análise do comportamento do sistema com controladores pré-projetados para controle apenas da posição em graus (°), e em ambas as etapas haverá a visualização dos resultados das simulações. Já a segunda fase do projeto consiste na construção do hardware necessário para realizar o requisitado pelo simulador, envolvendo a especificação dos componentes e a estrutura entre eles, e, por fim, a programação para o sistema funcionar de acordo com o especificado pelo usuário do simulador.

1.1 **OBJETIVOS**

a. Desenvolver um ambiente de simulação de malhas de controle fechadas distintas em um mesmo dispositivo.

   O projeto pretende desenvolver uma plataforma de simulação eficaz ao permitir a interatividade e ajuste dos ganhos de controladores PID, e ao permitir a comparação desses com outros tipos de controladores, por meio do controle tanto de posição quanto de velocidade de uma roda de inércia acoplada a um motor DC.

   Essa abordagem possibilita a comparação direta entre diferentes estratégias de controle, fornecendo informações sobre os efeitos específicos de parâmetros como ganho proporcional, integral e derivativo nas respostas do sistema. Essa capacidade de análise comparativa contribui para a otimização do controle desejado, e para o desenvolvimento de uma compreensão mais profunda dos princípios de controle e sua aplicação prática.

b. Aplicação Prática e Interdisciplinaridade:

   O projeto centraliza-se na aplicação prática de conceitos de engenharia, a fim de proporcionar melhor compreensão do conhecimento teórico, além da implementação e testes de conceitos importantes para um profissional da área de Controle e Automação, como identificação matemática dos sistemas, controle PID e programação de microcontroladores.

   Em paralelo, a integração interdisciplinar entre as áreas de Instrumentação, Microcontroladores, Sistemas de Controle e Programação Orientada a Objetos permite o entendimento aprofundado de como essas áreas se relacionam e colaboram em projetos de engenharia complexos. Isso desenvolve o pensamento crítico e desenvolve habilidades que têm como foco resolver problemas e encarar demandas do mundo real, onde as soluções exigem a colaboração entre diferentes áreas do conhecimento.

1.2 **JUSTIFICATIVA**

O desenvolvimento do simulador interativo para controlar a posição angular e a velocidade é uma iniciativa de grande relevância no contexto do curso de engenharia de controle e automação. Este projeto é uma resposta prática ao conhecimento adquirido nas disciplinas da 4ª Série, permitindo que os alunos compreendam como esses conhecimentos se traduzem em soluções reais e promovendo uma compreensão mais profunda dos conceitos e teorias aprendidos.

Além disso, a construção e operação do simulador demandam o desenvolvimento de habilidades técnicas complexas, como lidar com sensores, motores, encoders, microcontroladores, programar controladores de sistemas dinâmicos, interfaces gráficas para a interação humana simplificada, entre outras habilidades técnicas desenvolvidas.

A estrutura do projeto, dividida em fases progressivas, reflete as etapas comuns em projetos de engenharia do mundo real. Conceituar um sistema, simulá-lo e validá-lo é parte integrante do desenvolvimento de um projeto, desde a concepção até a implementação, enfatizando a experimentação prática em um ambiente controlado.

Por fim, o desenvolvimento deste simulador é justificado pelos seus benefícios educacionais e de uso geral, por se tratar de uma ferramenta de controle funcional e eficaz. O simulador tem o potencial de servir como um recurso para estudantes e profissionais, permitindo testes e experimentações virtuais antes da implementação em sistemas reais.

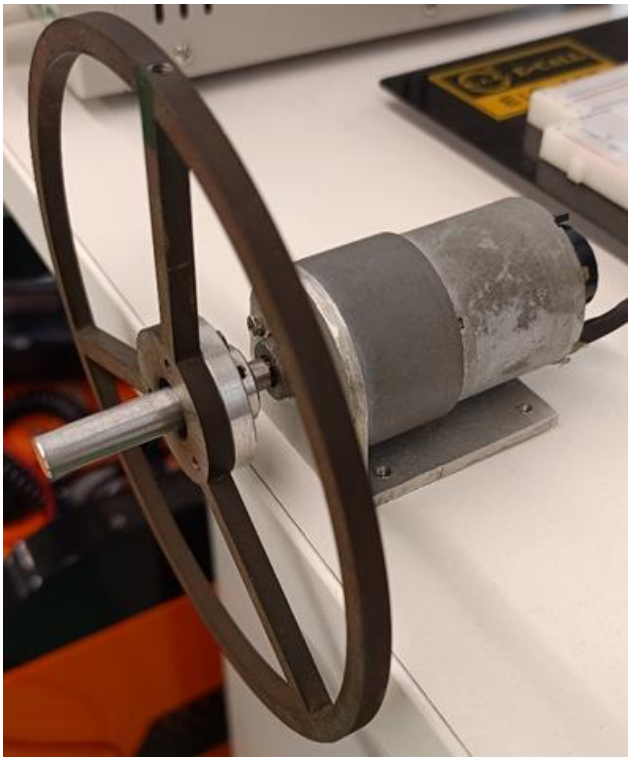
2 FUNDAMENTAÇÃO TEÓRICA

Antes de entrar nos estudos da teoria de controle que envolve o projeto, será abordado o funcionamento das etapas que envolvem o ambiente virtual, os componentes presentes e a lógica entre eles.

2.1 **HARDWARE**

Conforme discutido anteriormente, o componente central do projeto é o motor DC, no qual está acoplada uma roda de inércia ao eixo de saída de uma caixa de redução. A razão para sua inclusão no sistema é a criação de uma carga adicional, proporcionando um maior desafio no controle do motor. No mesmo conjunto do motor, encontramos o rotor do motor acoplado ao eixo de entrada da caixa de redução, bem como um encoder de efeito Hall, conforme ilustrado na imagem a seguir:

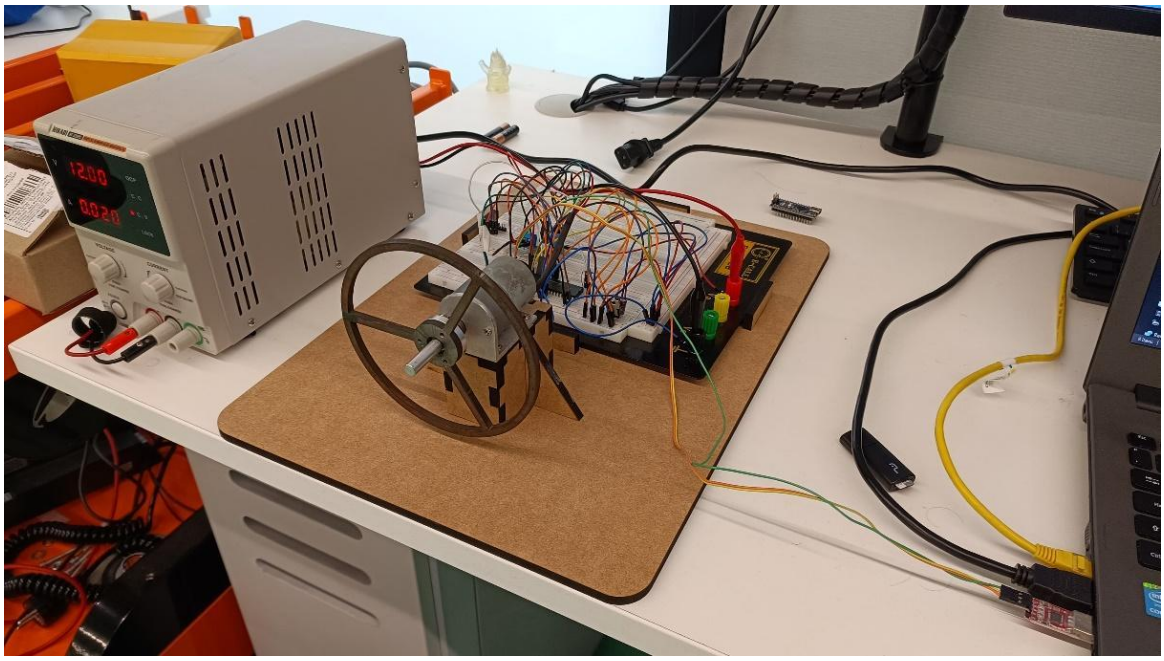
Figura 1 — Motor DC com roda de inércia.



Fonte: Autores (2023).

Porém além do motor deve-se considerar toda a eletrônica que torna possível o uso e controle do sistema, esse conjunto é chamada de planta, sendo a imagem a seguir:

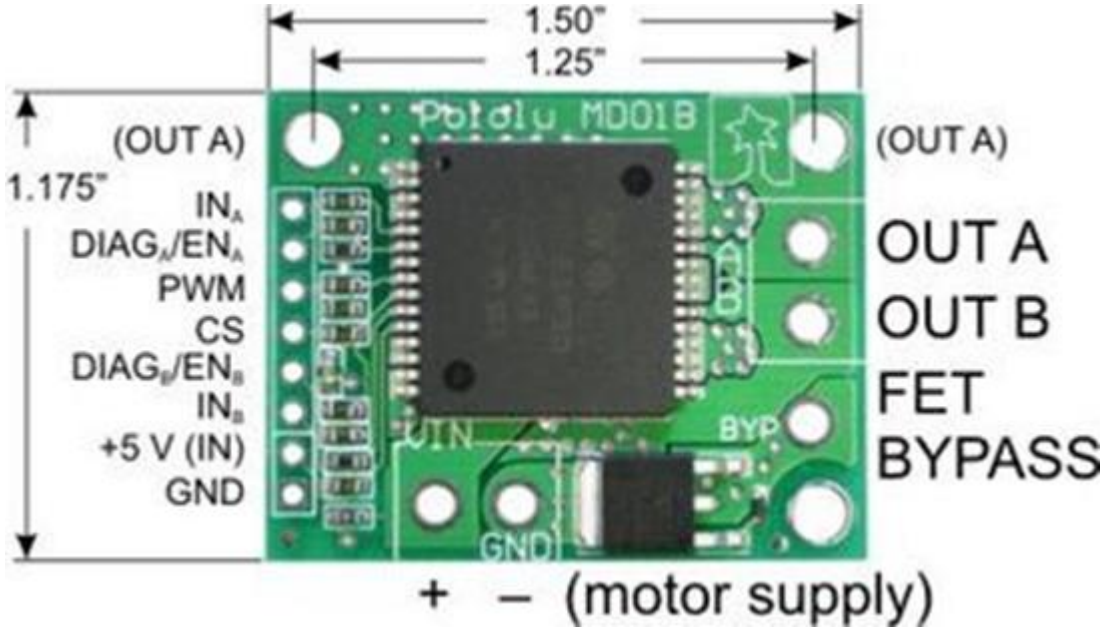
Figura 2 — Planta de Controle Completa



Fonte: Autores (2023)

Quando se trata de controle, a primeira consideração é dada às grandezas analógicas. Por exemplo, foi necessário ajustar a tensão aplicada a um motor devido às suas diferentes velocidades. Para isso, é utilizada uma ponte H, que gerencia a tensão e a direção do motor por meio de dois sinais digitais e um PWM (Modulação por Largura de Pulso). A configuração específica da ponte H empregada no projeto está ilustrada abaixo:

Figura 3 — Ponte H



Fonte: Pololu (2023).

Visto que para controlar a tensão e a direção do motor são necessários pinos digitais e um PWM, foi inserido no projeto um microcontrolador, o PIC16F18877. Ele irá controlar a ponte H enviando os dados de acordo com o controlador implementado. Além de estar conectado na ponte H, o microcontrolador ainda será responsável por contar os pulsos gerados no encoder e identificar, por meio do canal A e B, o sentido de rotação do motor e ao final atribuir ao seu valor corretamente.

Neste momento, o *hardware* já teria capacidade de executar o controle, porém a ideia do projeto é fazer um simulador. Para isso, o *hardware* existente deve ser conectado ao computador. Assim, foi utilizado a comunicação serial entre o microcontrolador e o computador;

2.2 **INSTRUMENTAÇÃO**

Para a parte da disciplina de Instrumentação, além da proposta do projeto, o sensor utilizado foi o próprio encoder de efeito Hall do motor, portanto foi necessário entender como o encoder do motor utilizado funciona e como é feita a sua leitura

Um encoder de efeito Hall é um dispositivo utilizado para medir a posição, velocidade e direção de rotação de um eixo em sistemas mecânicos, particularmente em motores elétricos, como os motores DC (corrente contínua). Este tipo de encoder baseia-se no efeito Hall, que é um fenômeno físico no qual uma tensão elétrica é gerada em um condutor quando este se move através de um campo magnético.

No contexto dos encoders de efeito Hall, há geralmente um ímã montado no eixo rotativo do motor. Sensores Hall sensíveis a campos magnéticos são colocados estrategicamente em torno do ímã. Quando o eixo gira, as mudanças no campo magnético são detectadas pelos sensores Hall, e isso gera sinais elétricos proporcionais às variações no campo magnético.

O encoder de efeito Hall, comumente utilizado em motores DC, possui geralmente dois canais de saída chamados Canal A e Canal B. Esses canais são responsáveis pela detecção das mudanças no campo magnético captadas pelos sensores Hall. A relação de fase entre os sinais nos canais A e B é usada para determinar a direção da rotação.

Estes encoders são fundamentais para aplicações onde é necessário monitorar e controlar precisamente o movimento rotativo. Eles são especialmente úteis em sistemas onde a retroalimentação de posição é crucial, como em robótica, automação industrial, sistemas de controle de velocidade, entre outros. A utilização de encoders de efeito Hall contribui para um controle mais preciso e eficiente do movimento rotativo, sendo uma peça chave em muitos sistemas modernos.

Existem diferentes tipos de encoders, e o encoder de efeito Hall é comumente encontrado em motores DC para detectar a rotação do eixo. Ele geralmente possui dois canais de saída, denominados Canal A e Canal B, que são gerados pela mudança do campo magnético detectado pelos sensores Hall. Abaixo está o funcionamento do Canal A e Canal B:

a. Canal A e Canal B: São saídas de sinais digitais que fornecem informações sobre a direção de rotação e o número de pulsos gerados enquanto o eixo do motor está em movimento.
b. Sinais Faseados: Os canais A e B em um encoder de efeito Hall são tipicamente defasados em 90 graus (sinais quadratura). Quando o eixo do motor gira, os sensores Hall detectam as mudanças no campo magnético do ímã e geram pulsos elétricos em

ambos os canais A e B. A relação de tempo entre esses pulsos determina a direção do movimento.

c. Sinais de Contagem: Cada canal produz uma série de pulsos elétricos à medida que o eixo do motor gira. Contar esses pulsos permite calcular a velocidade e a posição angular do motor.
d. Detecção de Direção: A ordem em que os pulsos ocorrem nos canais A e B indica a direção do movimento. Se o Canal A mudar de estado (de alto para baixo ou de baixo para alto) antes do Canal B, o movimento é em uma direção; se for o contrário, é em direção oposta.
e. Decodificação do sinal: A eletrônica do sistema interpreta os pulsos nos canais A e B para determinar a direção e a velocidade do movimento do motor. Com base na contagem de pulsos e na sequência dos sinais, é possível calcular a posição exata do eixo.

O Canal A e o Canal B em um encoder de efeito Hall fornecem informações cruciais para determinar a direção e a velocidade do motor DC. Ao medir os pulsos e interpretar sua sequência, o sistema consegue calcular a posição angular do eixo e entender a direção do movimento.

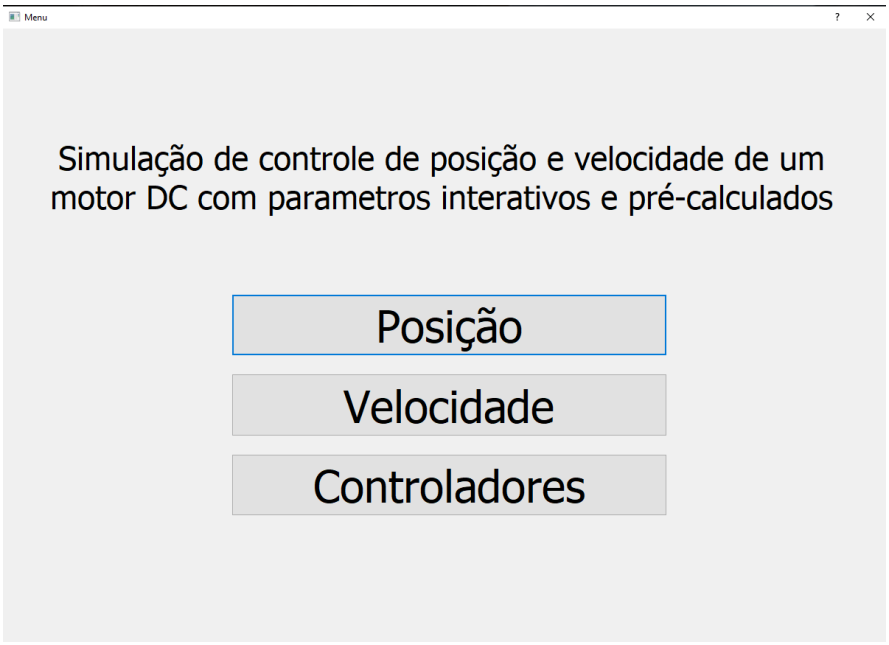
3 DESENVOLVIMENTO

Este capítulo tratará sobre o desenvolvimento das partes comentadas anteriormente com aprofundamento detalhado em cada parte.

3.1 INTERFACE DO USUÁRIO

O processo inicia-se com a interação do usuário por meio de uma interface gráfica. A tela de menu apresenta três botões distintos, cada um destinado a um controle específico: controle de posição, controle de velocidade e comparação entre controladores de avanço de fase, PD e PID. A estrutura, desenvolvida utilizando orientação a objetos na linguagem *Python*, permite a fácil navegação entre as páginas. A tela “Menu” é exibida ao usuário como na figura abaixo:

Figura 4 — Menu da interface.

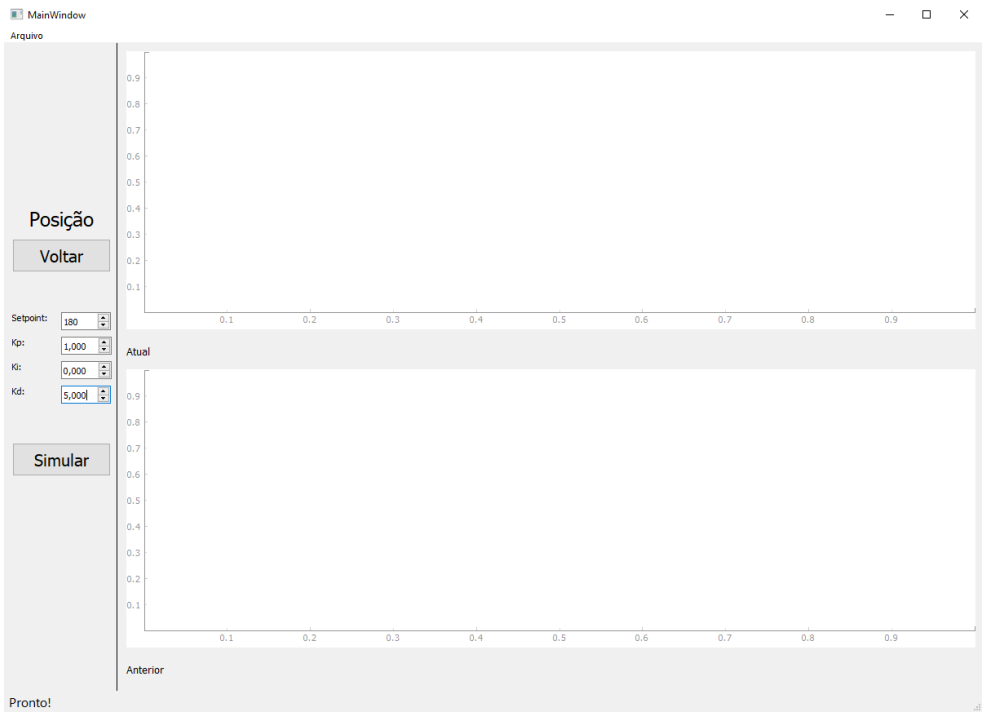


Fonte: Autores (2023)

Nela, as opções “Posição” e “Velocidade” abrem janelas que permitem configurar e visualizar a simulação. Nas duas primeiras páginas, de controle de posição e velocidade, é possível inserir parâmetros como KP, KI e KD para a configuração do controlador PID, além do *setPoint* do sistema. A simulação é iniciada por meio de um botão que envia informações para o microcontrolador por meio da comunicação serial, e posteriormente serão utilizadas para o

controle do motor. As telas de controle PID de posição e velocidade estão apresentadas nas figuras a seguir:

Figura 5 — Ambiente de simulação interativo para a posição.



Fonte: Autores (2023).

Os campos a serem preenchidos se referem aos parâmetros proporcional, integrativo e derivativo de um PID, e o *SetPoint* representa o valor alvo de controle, como detalhado abaixo:

- KP (Proporcional): O parâmetro proporcional controla a resposta proporcional do sistema à diferença entre a saída do sistema e o valor de referência (SetPoint). Um valor maior de KP resulta em uma correção proporcional mais forte, o que significa que a saída do sistema se ajusta mais rapidamente à mudança no SetPoint. Porém, um valor muito alto pode levar a instabilidade ou oscilações indesejadas.
- KI (Integral): O parâmetro integral atua na correção dos erros acumulados ao longo do tempo. Ele é responsável por reduzir o erro em estado estacionário e melhorar a precisão do sistema. Um valor maior de KI aumenta a influência da integral e reduz o erro acumulado, mas também pode levar a respostas lentas e oscilações.
- KD (Derivativo): O parâmetro derivativo controla a resposta do sistema à taxa de mudança da variável controlada. Ele atua para prevenir grandes oscilações e suavizar a resposta do sistema, proporcionando estabilidade. Um KD mais alto ajuda a evitar

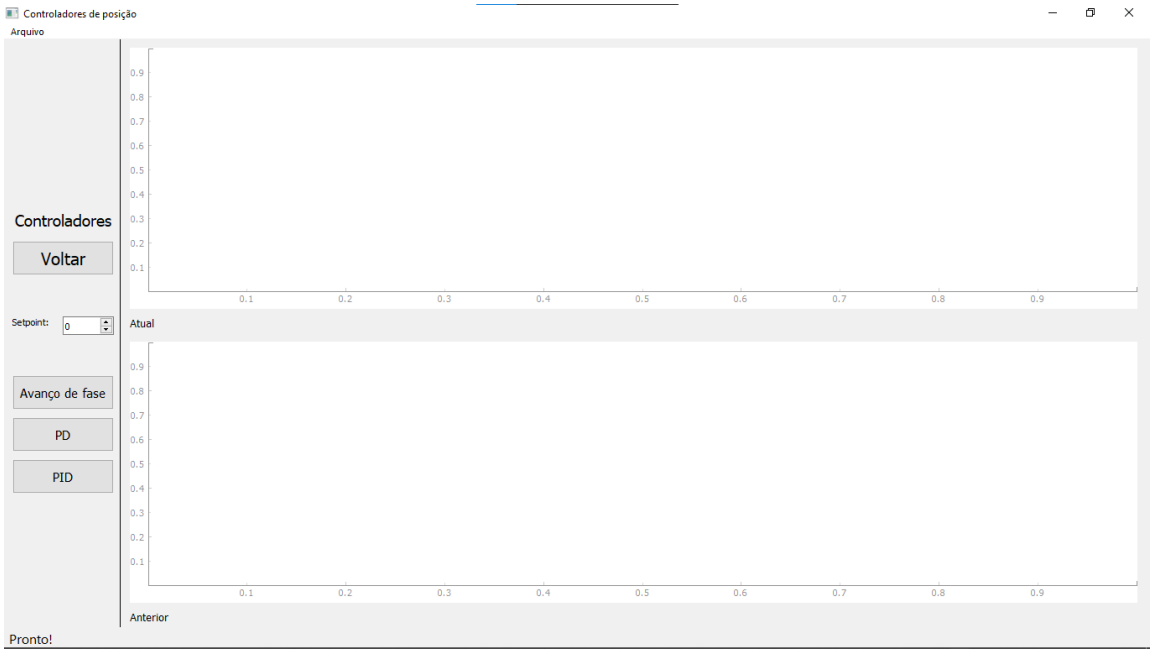
picos e amortecer as oscilações, mas um valor excessivamente alto pode introduzir atrasos na resposta do sistema.

- SetPoint: O SetPoint representa o valor desejado ou a referência para a variável controlada. No contexto de um controle PID, o objetivo é ajustar a saída do sistema para atingir e manter o SetPoint. A malha de controle compara continuamente a saída real do sistema com o SetPoint e ajusta os parâmetros PID para minimizar o erro.

Inserindo os parâmetros KP, KI e KD e SetPoint, é possível simular o sistema. Por meio do botão “Simular”, é gerada uma resposta gráfica do comportamento do conjunto motor/ roda de inércia diante dos parâmetros inseridos. Isso pode ser feito tanto na tela “Posição” quanto na tela “Velocidade”.

Por fim, a tela “Controladores” permite a comparação entre controladores PD, PID e avanço de fase, sendo o primeiro calculado com a ajuda do *software* MatLab e os dois últimos calculados manualmente e detalhados no decorrer do projeto. Essa tela possui apenas um dado de entrada, o *setPoint.* Após preenchido, basta clicar em algum dos botões dos controladores para que o motor seja ensaiado e a resposta apareça graficamente para o usuário na tela:

Figura 6 — Ambiente de simulação dos controladores.



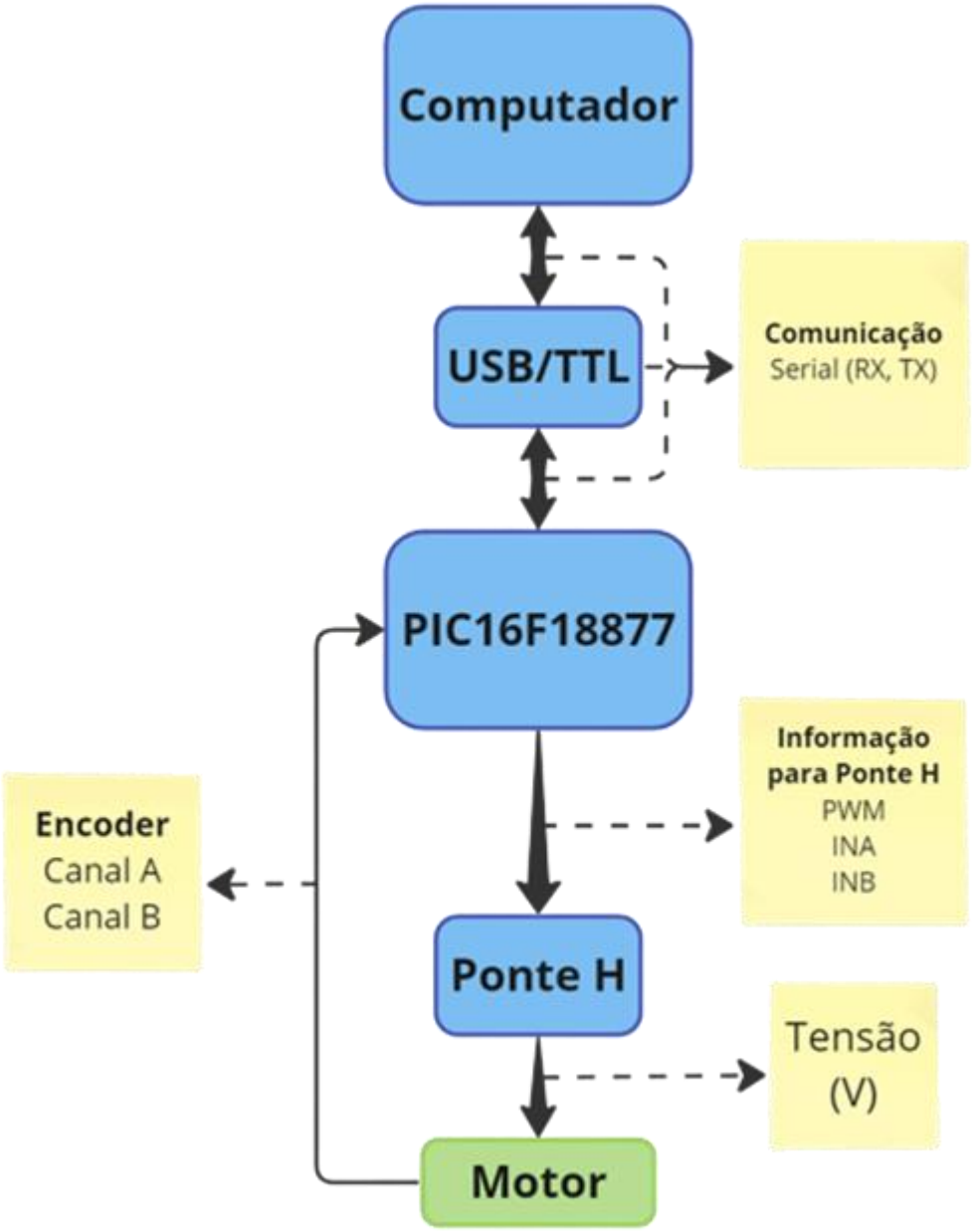
Fonte: Autores (2023).

Por fim, é possível salvar um determinado ensaio, a fim de reabri-lo futuramente. Essa função é importante para fazer comparações entre ensaios realizados em diferentes momentos, e registrá-los para análises futuras.

3.2 ESTRUTURA DO PROJETO E COMUNICAÇÃO

Na interação usuário-programa, o sistema pode ser configurado para controle de posição e velocidade de um motor. O diagrama da estrutura do projeto é apresentado na figura a seguir a fim de facilitar a visualização de cada elemento dentro do projeto e como eles estão interconectados:

Figura 7 — Fluxograma.



Fonte: Autores (2023).

A comunicação entre o computador, onde está a interface, e o microcontrolador ocorre por meio de um cabo USB TTL, utilizando o protocolo de comunicação UART.

A comunicação serial UART (Universal Asynchronous Receiver/Transmitter) é um método de comunicação entre dispositivos eletrônicos que envolve a transmissão e recepção de dados em série, um bit de cada vez, através de dois fios: um para transmitir (TX) e outro para receber (RX).

Trata-se de uma comunicação assíncrona, o que significa que não há um sinal de clock compartilhado entre os dispositivos. Em vez disso, os dispositivos precisam concordar com uma taxa de transmissão, chamada de Baud Rate, que determina a velocidade da transmissão. Antes e depois de cada pacote de dados são adicionados de início e de parada para indicar o começo e o final da transmissão.

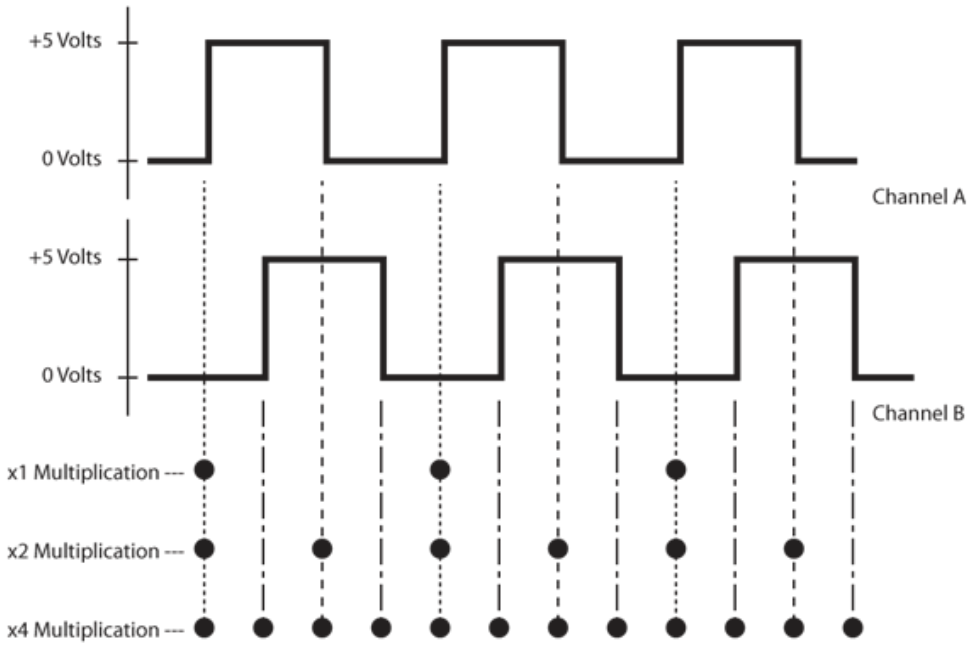
O microcontrolador, ao receber o pacote de informações, realiza a leitura desses dados, processando-os através do código desenvolvido no MPLAB, que contém o software de controle do sistema.

Após o tratamento, um sinal PWM (Pulse Width Modulation) é transmitido para uma ponte H. Comparado a métodos de controle analógicos, o PWM é eficiente em termos de energia, já que o dispositivo está essencialmente ligado ou desligado, minimizando as perdas de potência.

A ponte H, com entradas “INA” e “INB”, controla a direção do motor, determinando o sentido horário ou anti-horário do movimento. O sinal resultante é convertido em tensão (V) e transmitido ao motor para a realização do movimento desejado, seja para controle de posição ou velocidade, conforme as configurações previamente inseridas pelo usuário.

Para fechar a malha de controle, um encoder monitora o motor por meio de dois canais, A e B, fornecendo informações sobre o sentido de rotação. Esses dados são enviados de volta ao microcontrolador, permitindo o controle efetivo na malha fechada. A leitura e interpretação dos sinais transmitidos pelos canais A e B podem ser melhor compreendidas pela imagem abaixo. Por meio da combinação dos tempos de bordas de subida dos dois canais, é possível inferir se o motor gira no sentido horário ou anti-horário:

Figura 8 — Quadratura de um encoder



Fonte: Robotics Business Review (2019).

Assim, o feedback do encoder é integrado ao processo de controle. Isso permite ao PIC enviar comandos ao motor e receber informações constantes sobre seu estado. Com base nesse feedback, o PIC ajusta dinamicamente os sinais PWM enviados à ponte H, otimizando a eficiência e garantindo correções em tempo real para manter a posição desejada do motor.

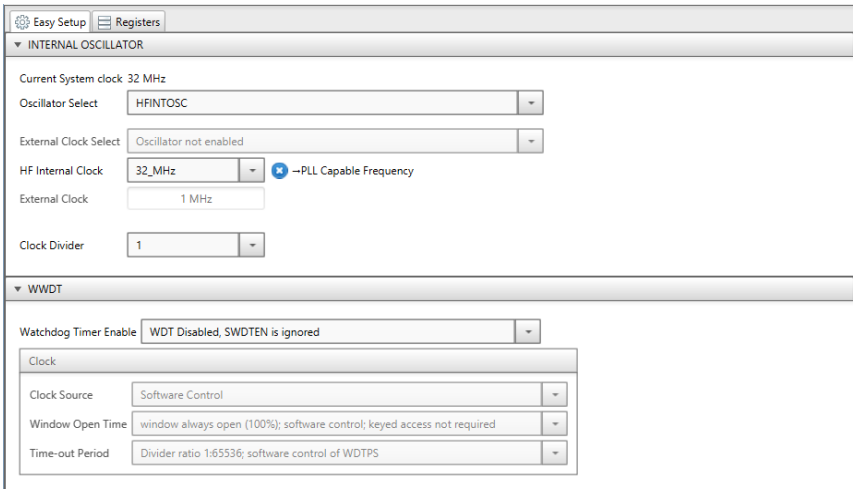
Com o sistema em malha fechada, sua resposta pode ser retransmitida ao computador, para que o usuário visualize graficamente a resposta do sistema diante dos parâmetros inseridos previamente. Destaca-se, portanto, que a comunicação entre o microcontrolador e o computador é bidirecional. As informações processadas pelo microcontrolador são retornadas ao computador, completando o ciclo de comunicação.

3.3 LÓGICA DE PROGRAMAÇÃO DO MICROCONTROLADOR

Antes de começar a desenvolver a programação do microcontrolador será necessário determinar que modelo deste será utilizado, isso depende inteiramente dos requisitos necessários para o projeto. Alguns destes requisitos foram comentados anteriormente, os periféricos do microcontrolador, são estas as saídas digitais e a saída PWM, a comunicação UART e interrupções, tanto externas quanto por timer presente no microcontrolador. Todos estes periféricos estão no PIC estudado nas aulas de Microcontroladores, o PIC16F1619, sendo um microcontrolador muito versátil por também ser compatível com o MCC (MPLAB® Code Configurator) da microchip, porém um dos requisitos do projeto não era contemplado por esse modelo, a memória SRAM dele, era menor do que o necessário para armazenar as informações da simulação, portanto para o projeto tem que ser um PIC com os mesmos periféricos, compatível com o MCC e que tenha a memória maior que 2KB, sendo este o PIC16F18877, atendendo assim todos os requisitos do projeto.

Com o modelo do microcontrolador determinado, a próxima etapa é começar o desenvolvimento do programa. Para isso foram utilizados o MPLAB e o MCC da própria microchip. O primeiro passo é configurar o microcontrolador no MCC e depois os periféricos no mesmo ambiente.

Figura 9 — Configuração do MCC no software MPLAB.

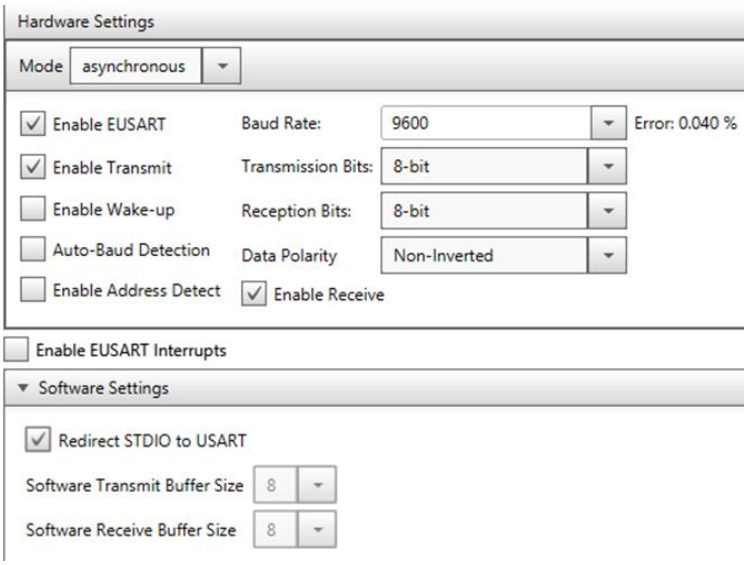


Fonte: Autores (2023).

Nesta parte foi configurado o clock, sendo escolhido o oscilador interno de 32 Mhz sem nenhum divisor e sem watchdog.

O primeiro periférico para ser configurado é o UART, chamado de EUSART *(Enhanced/Addressable Universal Asynchronous Receiver Transceiver)*, para isso basta adicioná-lo e configurar. Foi escolhido um Baud Rate de 9600 por conta do baixo erro e habilitado o uso dos comandos STDIO.

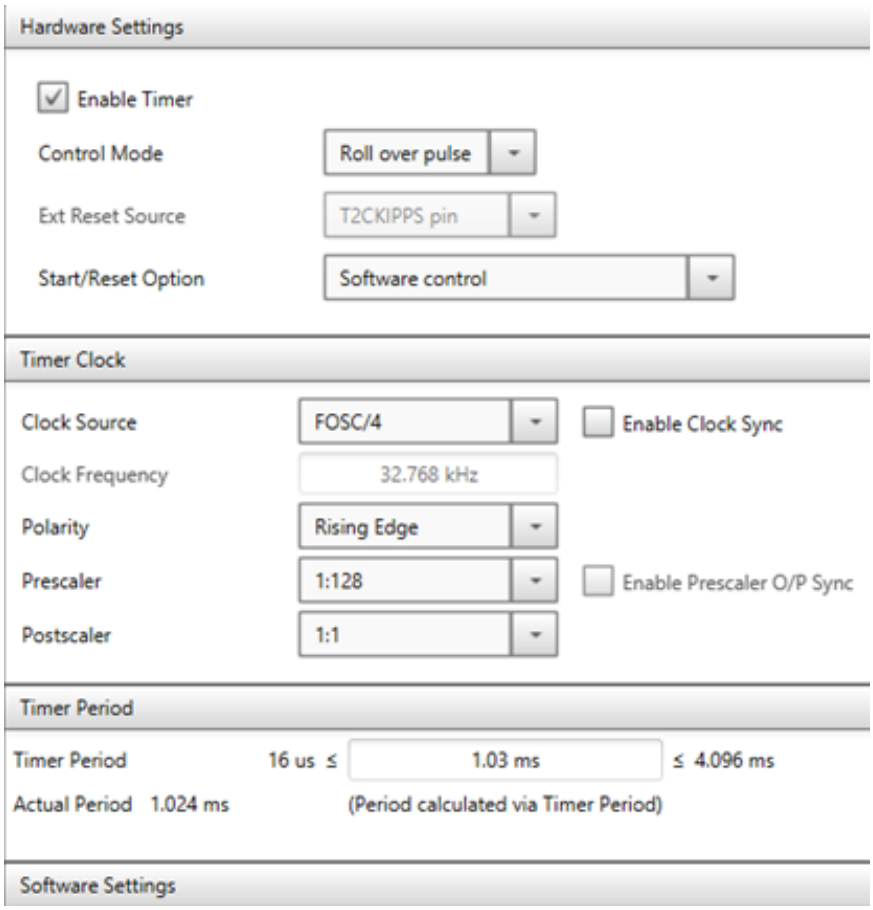
Figura 10 — Configuração do UART.



Fonte: Autores (2023).

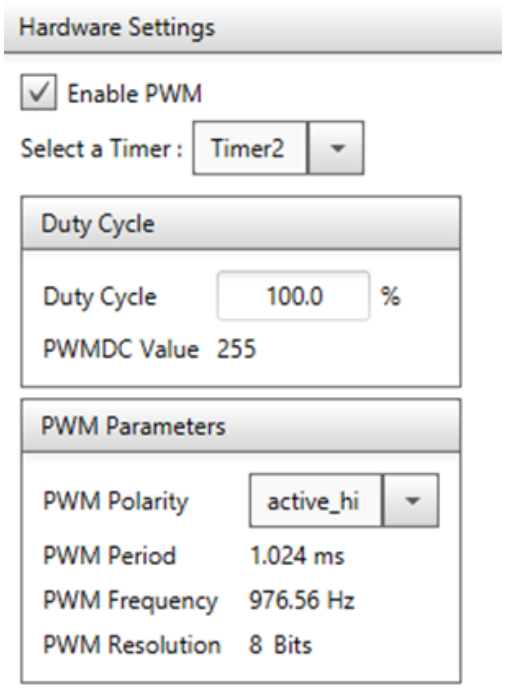
O próximo periférico é a saída PWM, adicionando-o é necessário configurar um timer junto, definindo o seu período para corresponder a frequência do sinal PWM.

Figura 11 — Configuração da saída PWM.



Fonte: Autores (2023).

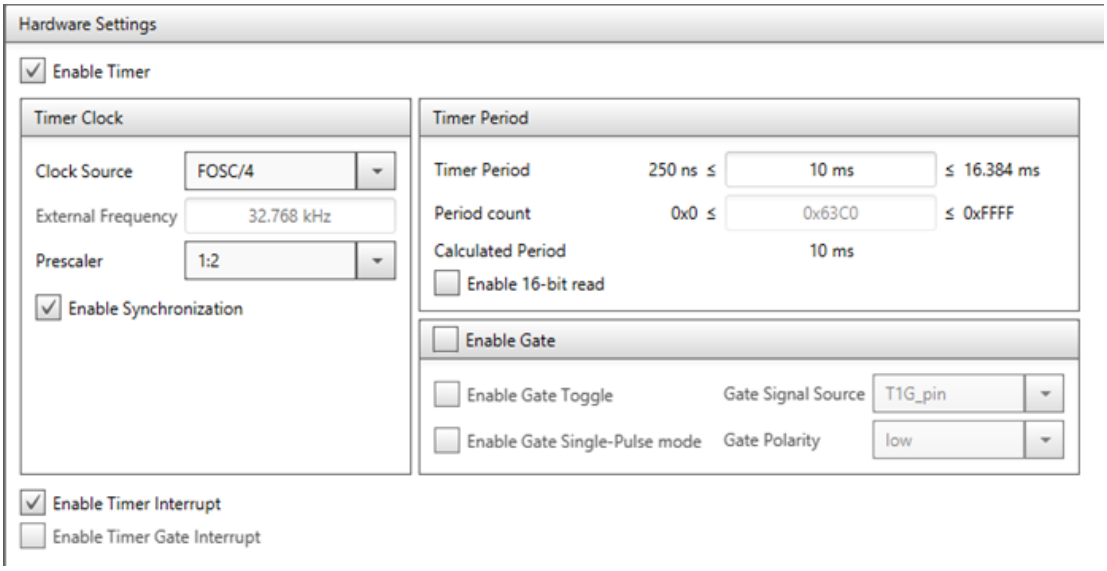
Figura 12 — Frequência do PWM.



Fonte: Autores (2023).

E o último periférico é o timer que executará a interrupção quando houver o estouro deste, nas configurações é possível determinar o período do estouro, sendo este o valor do tempo de amostragem dos controladores de 10 ms.

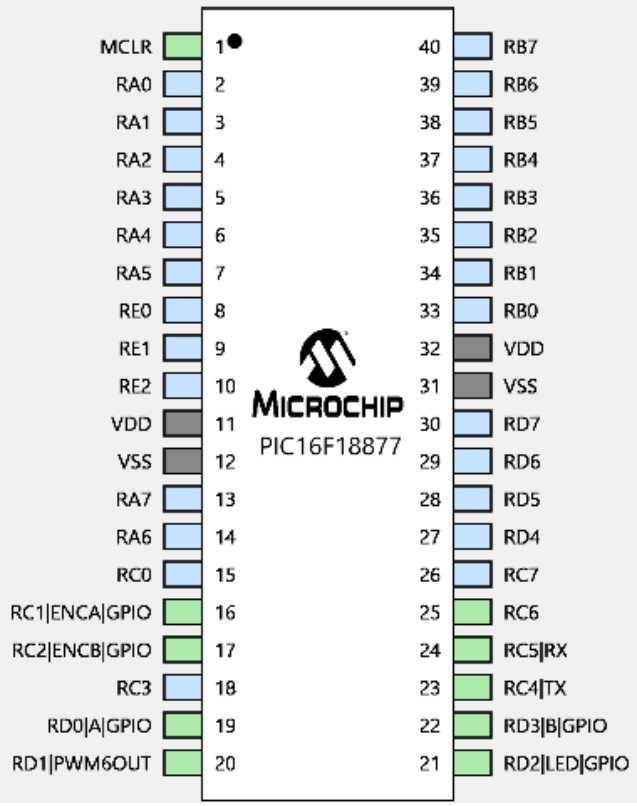
Figura 13 — Configuração do timer.



Fonte: Autores (2023).

E a última parte dentro do MCC é escolher e configurar os pinos utilizados, este são os dos periféricos como o UART e PWM, as saidas digitais para a ponte H (*Outputs)* e os pinos de entrada *(Input)* para o encoder, que devem ser configurados para ter interrupção na borda de subida, *IOC (Interrupr On Changing)*.

Figura 14 — Datasheet PIC16F18877



Fonte: Microchip (2014).

Figura 15 — Pinagem do PIC16F18877 no MPLAB

Easy Setup | Registers

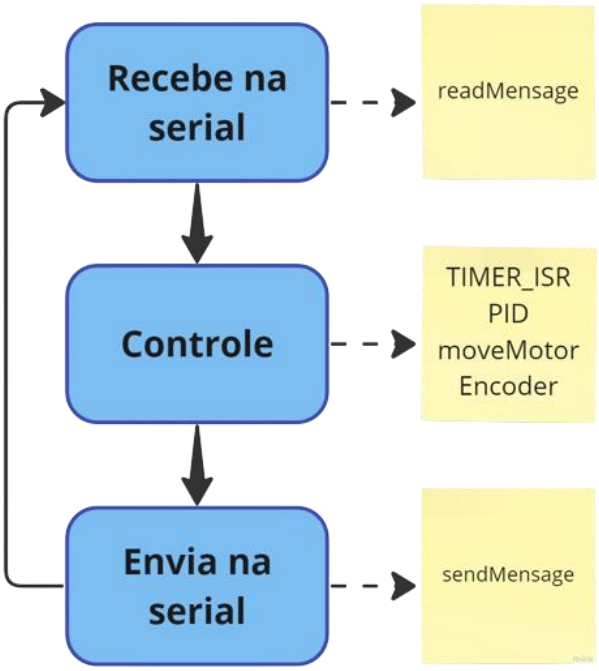
Selected Package : PDIP40

Pin Name	Module	Function	Custom Name	Start High	Analog	Output	WPU	OD	IOC
RC1	Pin Module	GPIO	ENCA	☐	☐	☐	☑	☐	positive
RC2	Pin Module	GPIO	ENCB	☐	☐	☐	☑	☐	positive
RC4	EUSART	TX		☐	☑	☑	☐	☐	none
RC5	EUSART	RX		☐	☐	☐	☐	☐	none
RD0	Pin Module	GPIO	A	☐	☐	☑	☐	☐	
RD1	PWM6	PWM6OUT		☐	☐	☑	☐	☐	
RD2	Pin Module	GPIO	LED	☐	☐	☑	☐	☐	
RD3	Pin Module	GPIO	B	☑	☐	☑	☐	☐	

Fonte: Autores (2023).

Neste instante o microcontrolador tem a capacidade de fazer o que foi planejado, porém para facilitar no desenvolvimento do programa algumas funções foram desenvolvidas seguindo o diagrama lógico:

Figura 16 — Fluxograma do Microcontrolador I.



Fonte: Autores (2023).

O primeiro bloco é responsável por receber na serial as informações vindas do simulador, a mensagem enviada tem um formato padrão contendo o modo, setPoint e se necessário os ganhos separados por vírgulas, então a função readMensage tem que ser capaz de ler toda a mensagem, separar as informações e atribuí las, está sendo implementada a seguir:

```
void readMensage(int *TYPE, long *SETPOINT, float *KP, float *KI,
float *KD){
    int i = 0;
    unsigned char receivedString[30];
    int value[5];
    while(1){
        if(EUSART_is_rx_ready()){
            unsigned char receivedChar = EUSART_Read();
            if(receivedChar == '\n' || receivedChar == '\r'){
                receivedString[i] = '\0';
                break;
            }
            else{
                receivedString[i] = receivedChar;
                i++;
            }
        }
    }
    char *token;
    token = strtok(receivedString,",");
    int j = 0;
    while(token != NULL){
        value[j] = atoi(token);
```

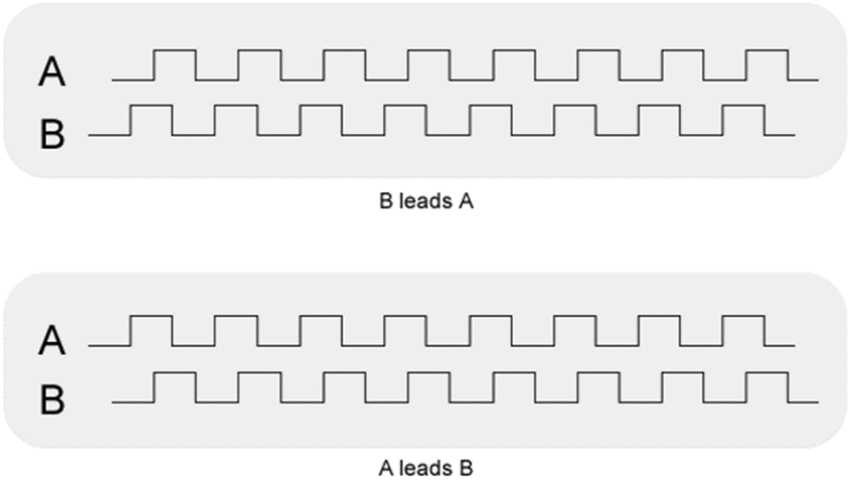
```
        token = strtok(NULL,",");
        j++;
    }
    *TYPE = value[0];
    *SETPOINT = value[1];
    *KP = value[2]/1000.0;
    *KI = value[3]/1000.0;
    *KD = value[4]/1000.0;
    receivedString[0] = '\0';
}
```

Com os dados atribuídos é possível avançar, a próxima etapa é controlar o motor, para isso é necessário desenvolver 3 funções antes, a primeira é função que receberá um valor inteiro, negativo ou positivo, que controlará o sentido e o sinal PWM do motor, além disso controlar a zona morta do motor, a função moveMotor:

```
void moveMotor(int m){
    if(m > 0){
        A_SetHigh();
        B_SetLow();
        m += 30;
//        m = constrain(m,20,255);
    }
    else if (m < 0){
        A_SetLow();
        B_SetHigh();
        m -= 30;
//        m = constrain(m,-255,-20);
    }
    PWM6_LoadDutyValue(abs(m));
}
```

Outra função envolve duas ISR *(interrupt service routine)* dos pinos de interrupção externa do encoder, essas funções foram pensadas no seguinte diagrama para determinar a quantidade e o sentido de rotação, somando e subtraindo um valor da variável.

Figura 17 — Comparação dos canais em ambos os sentidos de rotação



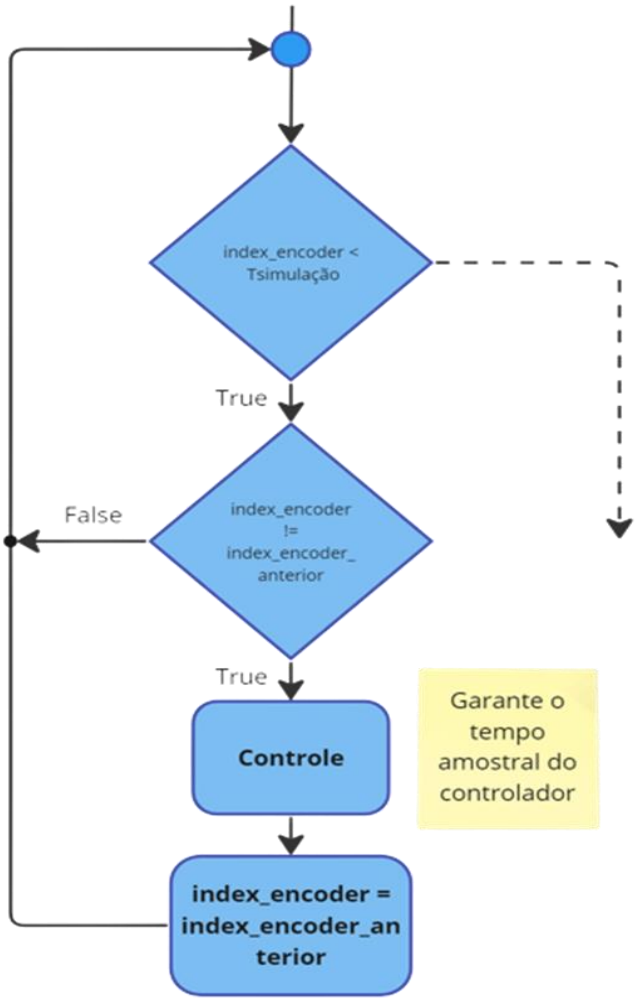
Fonte: Ramprasad Potluri (2011).

```
void ENCA_ISR(void){
    if (ENCB_GetValue() == 1) {
        encoder++;
    }
    else{
        encoder--;
    }
}

void ENCB_ISR(void){
    if (ENCA_GetValue() == 0) {
        encoder++;
    }
    else{
        encoder--;
    }
}
```

Por último é necessário desenvolver outra ISR, porém agora para a interrupção do estouro do timer, essa função tem que ser muito breve portanto não pode conter todos os cálculos dos controladores, pensando nisso foi desenvolvida a seguinte lógica, nela a interrupção “trava” uma outra função do loop, só podendo ocorrer a cada 10 ms.

Figura 18 — Fluxograma do Microcontrolador II.



Fonte: Autores (2023).

Então para isso ser possível basta definir a seguinte função de interrupção assim:

```
void TIMER_ISR(void){
    index_encoder++;
}
```

Neste momento a base para poder controlar a planta está feita, porém os controladores não foram implementados, estes serão implementados na função PID, dependendo do modo lido na serial, no readMensage, determinará qual controlador calculará a saída a partir do erro, usando o valor do encoder e setPoint, aplicando no motor usando moveMotor e ao mesmo tempo salvando o estado atual do motor e atualizando a lógica de travamento da ISR do timer, este trecho foi implementado a seguir:

```
void PID(){
    while(index_encoder < simulationTime/deltaT){
        if(index_encoder != index_encoder_anterior){
            if(type == 1){
                vel = getVelocity();
                error = (float) (setPoint*nPulseTurn/60000.0) - vel;
                correction += kp * error + ki * error_integrativo - kd
* (vel - lastVel);
                error_integrativo += error;
                error_integrativo = constrain(error_integrativo,-
600.0,600.0);
                lastVel = vel;
                Data[index_encoder] = vel * 60000.0 / ((float)
nPulseTurn);
                velPulse_ms = (encoder - lastPulse)/10.0;
                lastPulse = encoder;
            }
            else{
                if(type == 0){
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) -
pos);
                    correction = kp * error + ki * error_integrativo -
kd * (float) (pos - lastPos);
                    error_integrativo += error;
                    error_integrativo = constrain(error_integrativo,-
360.0*nPulseTurn/360.0,360.0*nPulseTurn/360.0);
                    lastPos = pos;
                    Data[index_encoder] = pos * 360.0 / ((float)
nPulseTurn);
                }
                else if(type == 2){//AF
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) -
pos);
                    correction = 0.136 * error - 0.1307 *
error_anterior + 0.9548 * correction;
                    error_anterior = error;
                    Data[index_encoder] = pos * 360.0 / ((float)
nPulseTurn);
                }
                else if(type == 5){//PID
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) -
pos);
                    correction = 1.509 * error - 2.94 * error_anterior
+ 1.431 * error_anterior_anterior + 1.607 * correction_anterior -
0.6065 * correction_anterior_anterior;
                    error_anterior_anterior = error_anterior;
                    error_anterior = error;
                    correction_anterior_anterior =
correction_anterior;
                    correction_anterior = correction;
                    Data[index_encoder] = pos * 360.0 / ((float)
nPulseTurn);
                }
                else if(type == 6){//PD-ML
                    pos = getPosition();
                    error = (float) ((setPoint*nPulseTurn/360.0) -
pos);
```

```
                    correction = 1.772 * error - 1.715 *
error_anterior + 0.6592 * correction;
                    error_anterior = error;
                    Data[index_encoder] = pos * 360.0 / ((float)
nPulseTurn);
                }
                else if(type == 7){
                    correction = setPoint;
                    Data[index_encoder] = getPosition();
                }
            }
            index_encoder_anterior = index_encoder;
            correction = constrain(correction,-255.0,255.0);
            moveMotor((int) correction);
        }
    }
    moveMotor(0);
}
```

Nesse momento o motor está sendo controlado, agora, sendo o último passo, basta enviar na serial todos os dados para exibir na interface, para isso foi utilizado o sendMensage, a qual lê todos os dados armazenados durante o controle e o tempo de cada instante:

```
void sendMensage(){
    for (int index = 0; index < simulationTime/deltaT; index++) {
        if(type == 7){
            printf("%i\n",Data[index]);//Ensaio
        }
        else{
            printf("%i,%i",Data[index],Time[index]);//Posicao
        }

        if (index < simulationTime/deltaT - 1) {
            if(type != 7){
                printf(",");
            }
        }
        __delay_ms(1);
    }
    printf("\n");
}
```

Nisso a programação do microcontrolador foi implementada com sucesso.

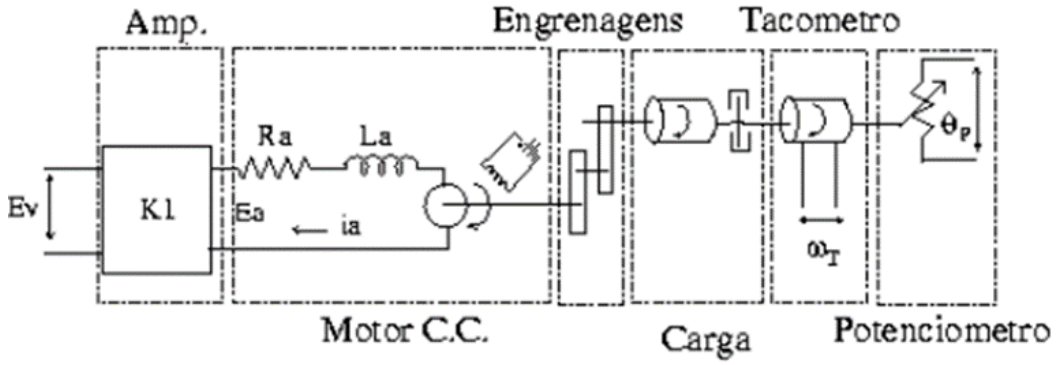
3.4 MODELAGEM DO MOTOR E SIMULAÇÃO

3.4.1 IDENTIFICAÇÃO DO SISTEMA

A primeira etapa para projetar um controlador é saber o que ele deve controlar. Em termos matemáticos, devemos conhecer a função de transferência que será trabalhada. No caso do projeto, deve-se encontrar a função de transferência do sistema motor e da roda de inércia.

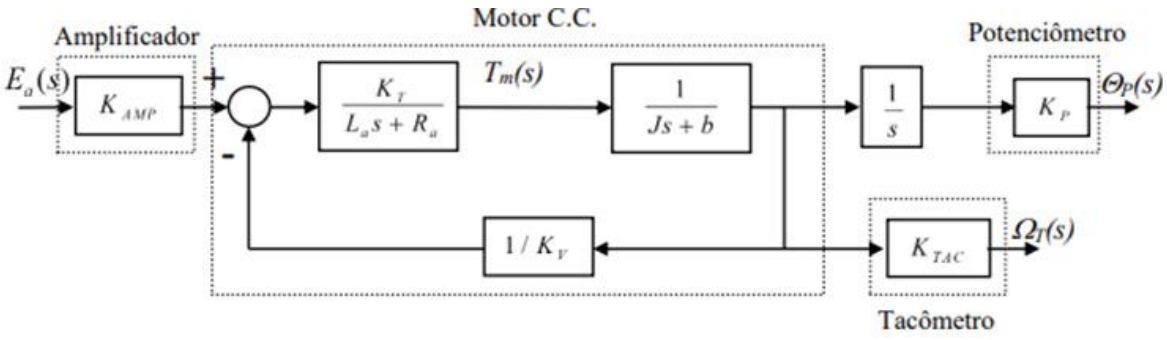
Por se tratar de um motor DC, sua modelagem é bem desenvolvida, sendo possível encontrar diagramas completos, como o mostrado na figura a seguir:

Figura 19 — Diagrama de uma planta com motor CC.



Fonte: Material didático - ECA401 (2022).

Figura 20 — Diagrama de blocos da planta com o motor CC.



Fonte: Material didático - ECA401 (2022).

Ao estudar o modelo e suas simplificações, é possível aproximar o modelo do motor para controle de velocidade em um sistema de primeira ordem, e para controle de posição em um sistema de segunda ordem, descritos a seguir:

$$G_{\omega}(s) = \frac{\Omega_T(s)}{E_V(s)} = \frac{K_w}{Ts + 1}$$

$$G_{\theta} = \frac{\theta_P(s)}{E_V(s)} = \frac{K_{\theta}}{s(Ts + 1)}$$

Há diversas maneiras de adquirir essa função de transferência. Uma delas é por meio da modelagem do motor, utilizando suas características físicas e elétricas para encontrar a função desejada. No entanto, o motor em questão não dispõe das informações necessárias para essa modelagem, tornando esse método inviável para a aplicação em questão. Dessa forma, optou-se pela obtenção da função de transferência através do método de ensaio, ou seja, levantamento da curva do motor. Para esse ensaio, foi decidido que serão realizados 5 degraus com os seguintes valores:

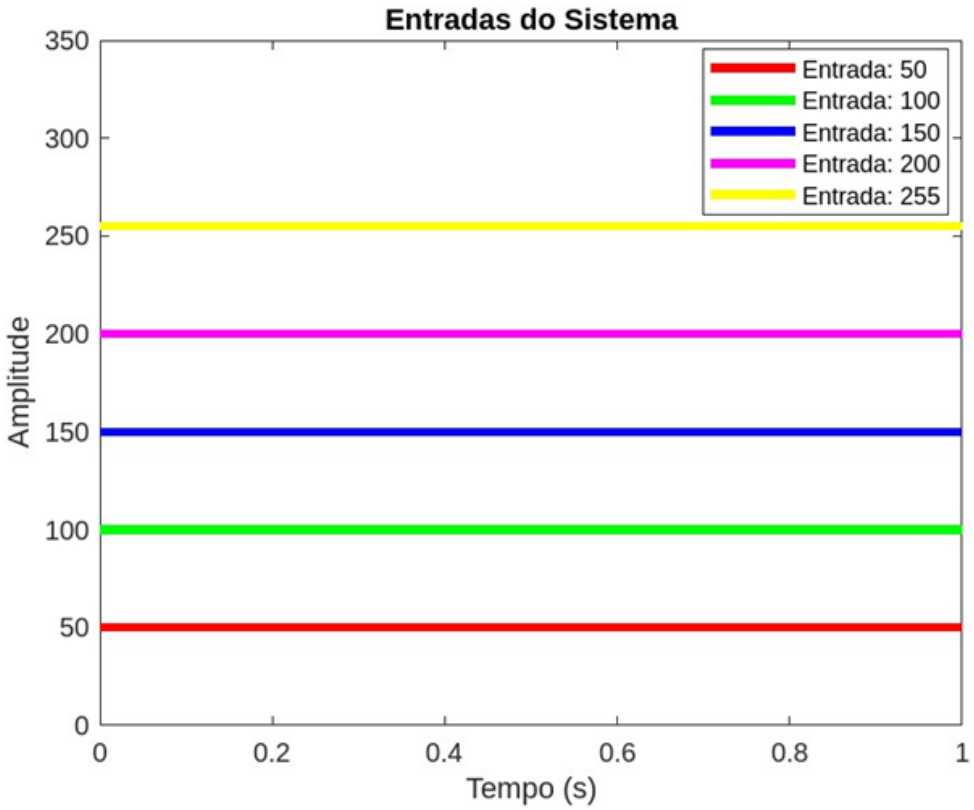
Tabela 1 — Dados de entrada dos ensaios.

PWM (8-bit)	**Porcentagem (%)**
50	19,6%
100	39,2%
150	58,8%
200	78,4%
255	100%

Fonte: Autores (2023).

Com esses dados, é possível definir a entrada do ensaio (U):

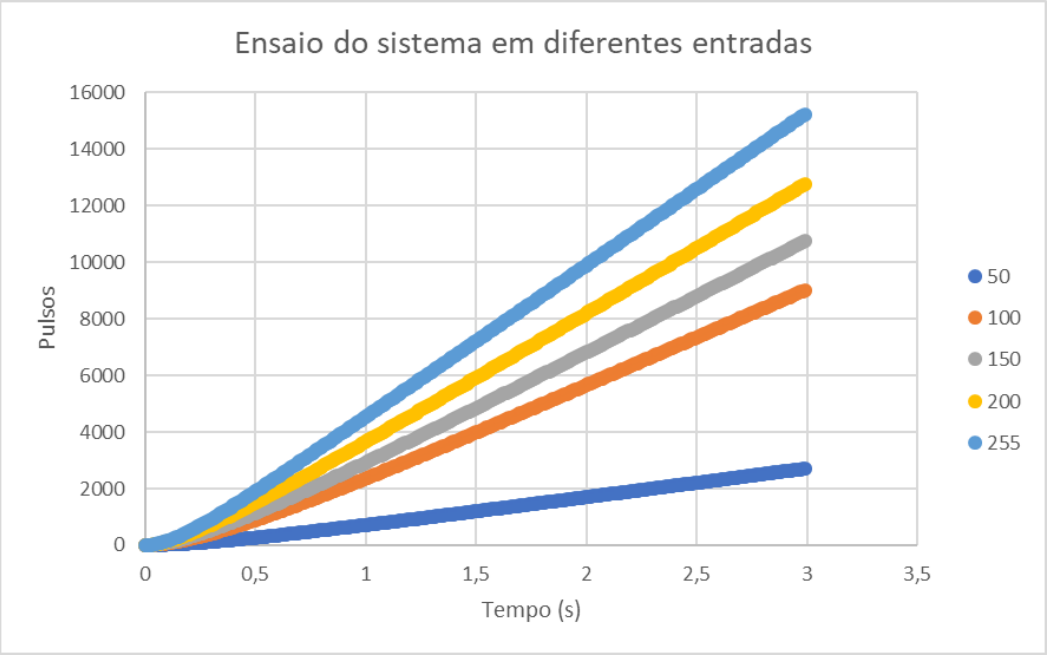
Figura 21 — Gráfico das entradas dos ensaios



Fonte: Autores (2023).

Os valores foram selecionados com base na resolução do PWM do microcontrolador. Uma vez que as entradas estão definidas, o experimento pode ser conduzido. A proposta consistiu em registrar o número de pulsos do encoder ao longo de 3 segundos, período suficiente para estabilizar o sistema. A análise dos dados será abordada de quatro maneiras: duas análises analíticas, utilizando os dados de posição e velocidade, e duas técnicas computacionais, empregando o software MatLab com os mesmos conjuntos de dados. Ao fim dos ensaios, os seguintes dados foram registrados:

Figura 22 — Gráfico dos ensaios de posição.

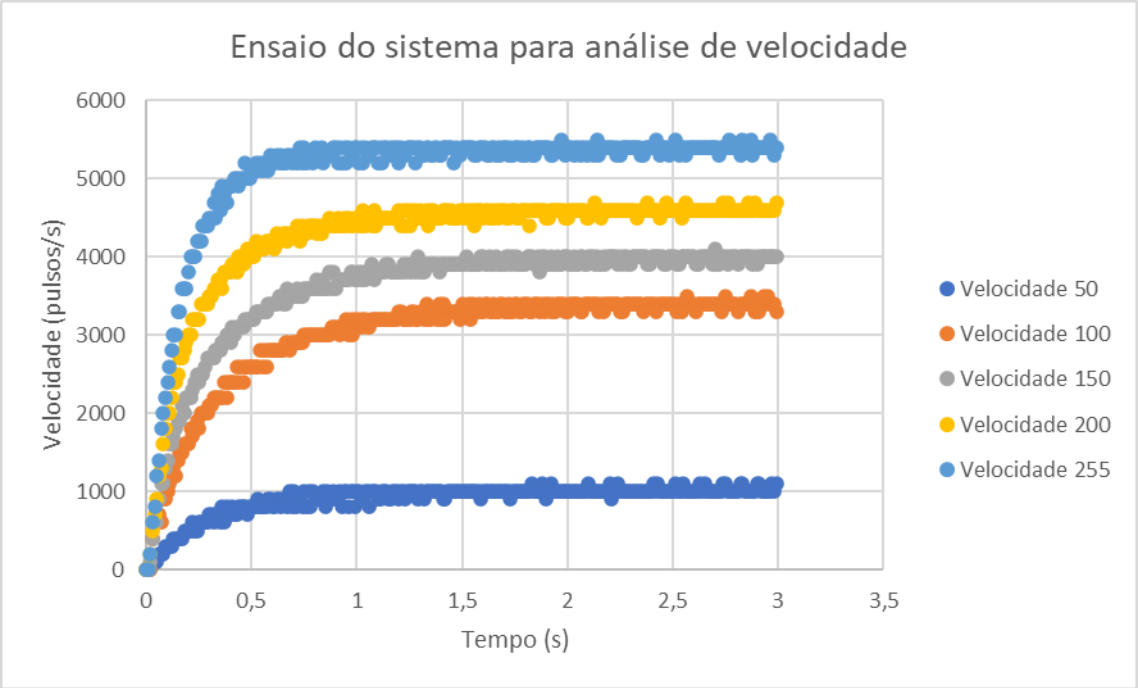


Fonte: Autores (2023).

Com os dados registrados, foram definidas as unidades das grandezas a fim de reduzir o processamento necessário no microcontrolador. Dessa maneira, a unidade de velocidade será de pulsos por segundo (pulsos/s) e posição em pulsos (pulsos).

Começando pela análise da velocidade, os dados foram tratados, chegando ao seguinte gráfico:

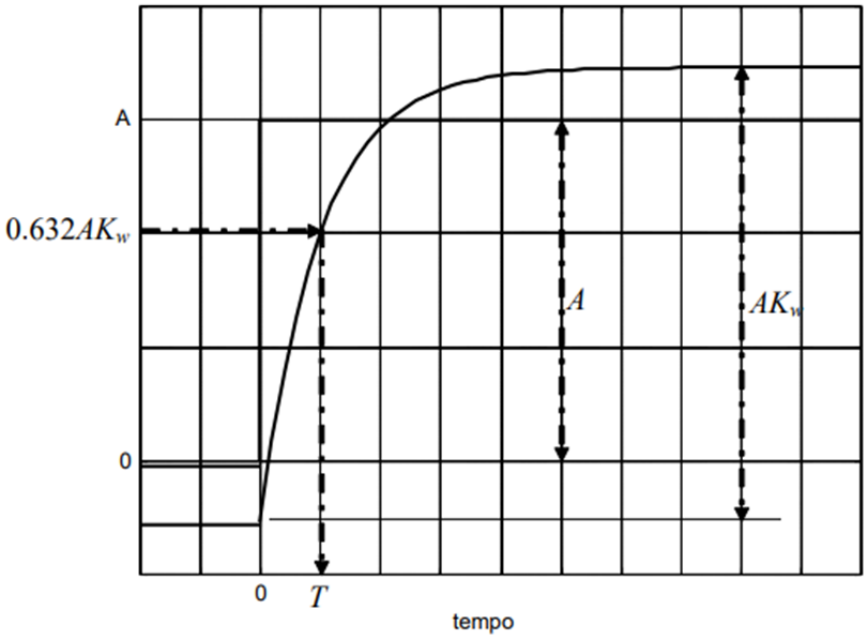
Figura 23 — Gráfico dos ensaios de velocidade.



Fonte: Autores (2023).

Com esses dados, foi feita a análise separada de cada saída, seguindo o modelo:

Figura 24 — Modelo de cálculo para ensaios de velocidade.



Fonte: Material didático - ECA401 (2022).

Usando os dados e os conceitos aprendidos na matéria de Sistemas de Controle I, obtemos as seguintes funções de transferência e seu modelo médio:

Tabela 2 — Funções de transferência da velocidade pelo método analítico

Função de Transferência	Constante de tempo (s)	Ganho
$G_{50} = \frac{20}{0{,}3s + 1}$	0,3	20
$G_{100} = \frac{34}{0{,}3s + 1}$	0,3	34
$G_{150} = \frac{26{,}67}{0{,}27s + 1}$	0,27	26,67
$G_{200} = \frac{23}{0{,}19s + 1}$	0,19	23
$G_{255} = \frac{21{,}176}{0{,}18s + 1}$	0,18	21,176
$G_{M} = \frac{24{,}97}{0{,}248s + 1}$	0,248	24,97

Fonte: Autores (2023).

Usando outro método de obtenção, pelo *software* MatLab e sua parte de identificação de sistemas, chegamos nas seguintes funções:

Tabela 3 — Funções de transferência da velocidade pelo software

Função de Transferência	Constante de tempo (s)	Ganho
$G_{50} = \frac{20{,}128}{0{,}321s + 1}$	0,321	20,128
$G_{100} = \frac{33{,}697}{0{,}353s + 1}$	0,353	33,697
$G_{150} = \frac{26{,}26}{0{,}295s + 1}$	0,295	26,26
$G_{200} = \frac{22{,}776}{0{,}21s + 1}$	0,21	22,776
$G_{255} = \frac{21{,}04}{0{,}16s + 1}$	0,16	21,04
$G_{M} = \frac{24{,}7802}{0{,}2678s + 1}$	0,2678	24,7802

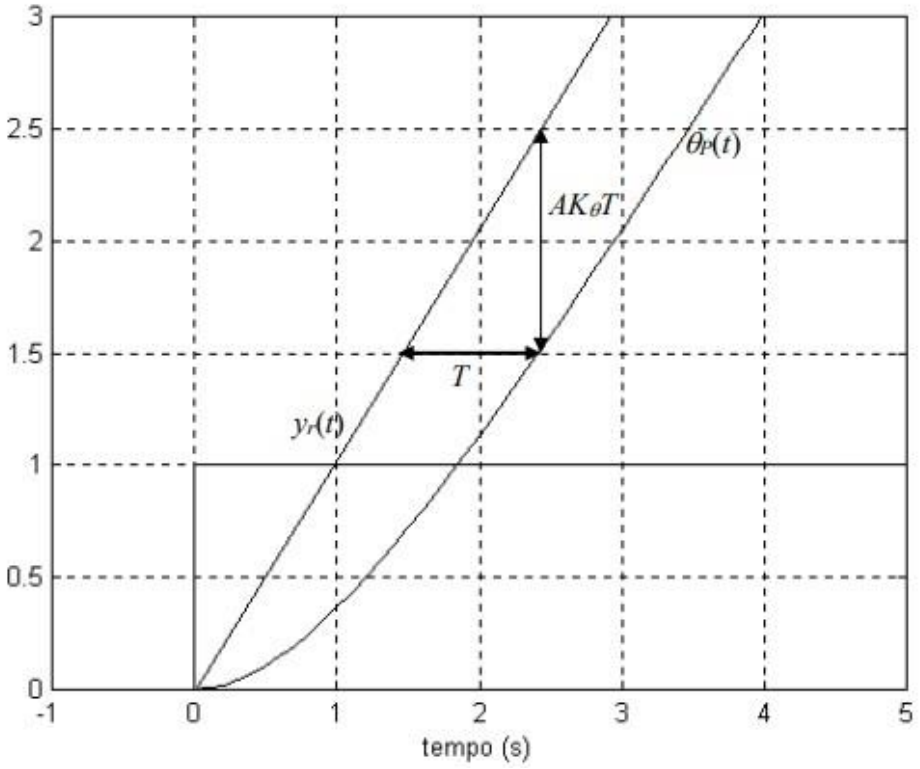
Fonte: Autores (2023).

Ao se analisar os resultados obtidos, percebe-se que as funções de transferência médias dos dois métodos exibem parâmetros próximos, com um erro percentual de 0,76% para o ganho e 7,4% para a constante de tempo. Dessa forma, torna-se possível considerar um modelo médio por meio da análise da velocidade.

$$G(s) = \frac{24{,}8751}{0{,}2579s + 1}$$

Além da análise dos dados de velocidade para a estimativa da função de transferência, propõe-se, neste projeto, uma abordagem adicional para obtenção da curva característica. Os dados de posição do motor serão utilizados, permitindo a comparação posterior dos resultados obtidos por ambas as metodologias. O processo seguirá uma abordagem semelhante ao realizado para a velocidade, seguido pela análise no MatLab, conforme o modelo apresentado abaixo:

Figura 25 — Modelo de cálculo para ensaios de posição



Fonte: Material didático - ECA401 (2022).

Tabela 4 — Funções de transferência da posição pelo método analítico e software.

Analítica			*Software*
Função de transferência	Cte. De tempo	Ganho	Função de transferência
$G_{50} = \frac{19,93}{s(0,3s+1)}$	0,3	19,93	$G_{50} = \frac{63,67}{s^2+3,342s+4,64*10^{-9}}$
$G_{100} = \frac{33,68}{s(0,31s+1)}$	0,31	33,68	$G_{100} = \frac{88,53}{s^2+2,748s+1,368*10^{-9}}$
$G_{150} = \frac{26,17}{s(0,27s+1)}$	0,27	26,17	$G_{150} = \frac{74,97}{s^2+2,978s+2,315*10^{-7}}$
$G_{200} = \frac{23,4}{s(0,2s+1)}$	0,2	23,4	$G_{200} = \frac{97,9}{s^2+4,43s+2,003*10^{-9}}$
$G_{255} = \frac{20,78}{s(0,16s+1)}$	0,16	20,78	$G_{255} = \frac{135,7}{s^2+6,521s+4,108*10^{-10}}$
$G_M = \frac{24,794}{s(0,248s+1)}$	0,248	24,794	$G_M = \frac{92,154}{s^2+4,002s+4,8*10^{-8}}$

Fonte: Autores (2023).

A primeira conclusão importante observada a partir da comparação dos métodos é a função de transferência sem o integrador livre obtida usando as ferramentas do MatLab, diferente do modelo simplificado obtido analiticamente. O modelo sem o integrador livre será utilizado para o cálculo analítico do controlador PID, que será abordado futuramente.

A segunda conclusão possível é: considerando que a função de transferência da velocidade integrada resulta na função de transferência da posição para a mesma entrada, chegamos com funções de transferência semelhantes, com um erro da constante de tempo em 3,8% e no ganho em 0,33%.

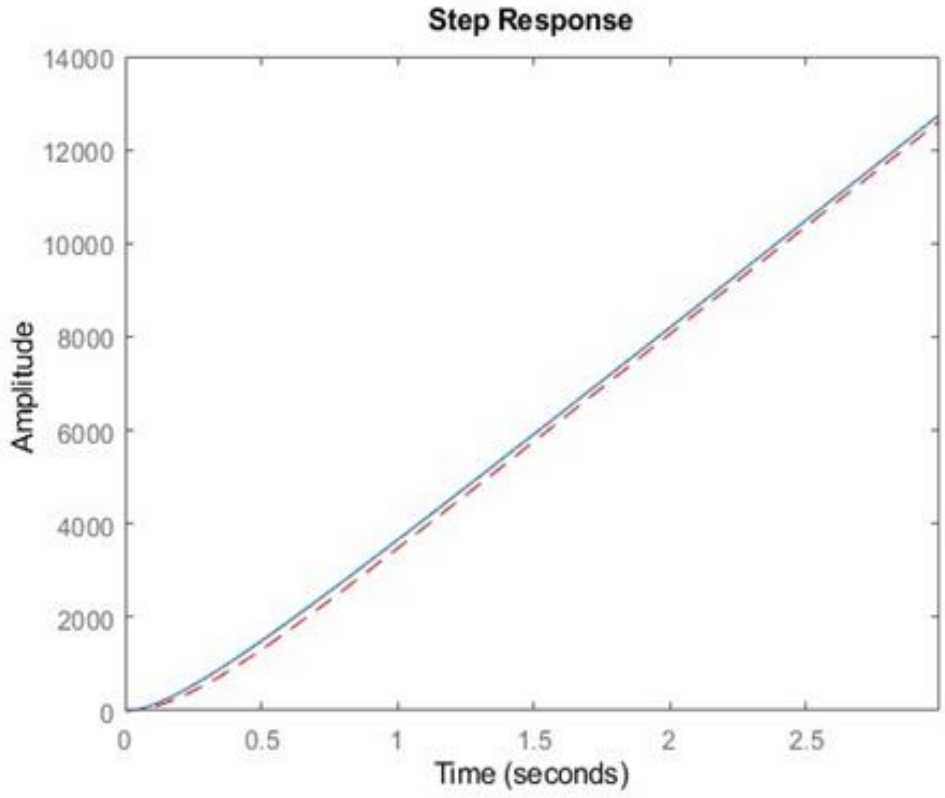
Com todos os dados obtidos e calculados é possível obter um modelo médio do motor estudado:

$$G(s) = \frac{24,8345}{s(0,2579s + 1)}$$

3.4.2 VALIDAÇÃO DO SISTEMA

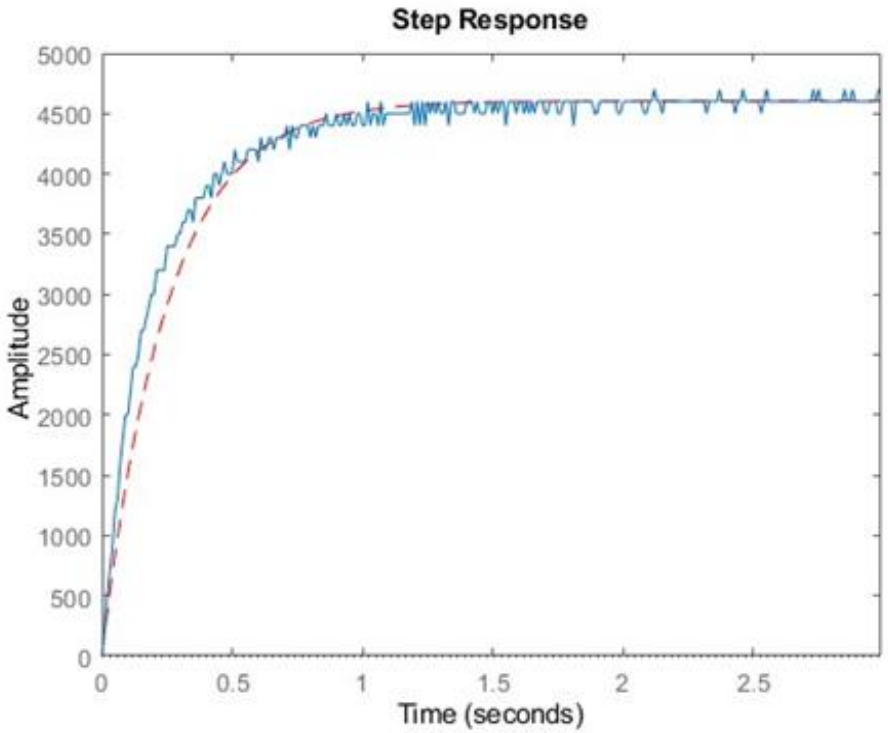
Com a obtenção da função de transferência que descreve o sistema de forma matemática, é possível realizar uma comparação entre a resposta simulada e a resposta real do sistema. A seguir, apresentam-se alguns testes que incluem as curvas correspondentes às simulações e à execução prática.

Figura 26 — Comparação dos modelos de posição e real.



Fonte: Autores (2023).

Figura 27 — Comparação os modelos de velocidade e real.



Fonte: Autores (2023).

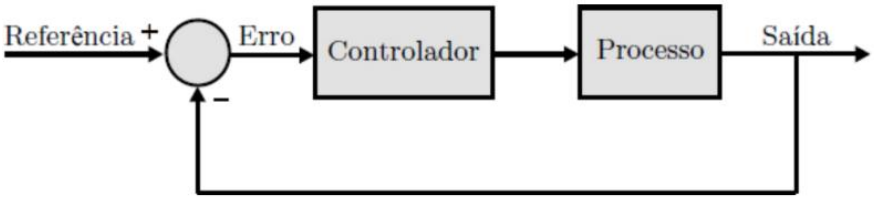
A curva azul representa o que o motor realmente realizou, e em vermelho tracejado está a simulação pela função de transferência obtida nos ensaios.

3.4.3 PROPOSTA DE CONTROLE DO SISTEMA

O projeto incorpora uma seção de controle interativo fundamentada em um controlador PID, além da seção dedicada aos controladores projetados. Em relação a esta última, o propósito primordial consiste na comparação entre os controladores calculados e aqueles projetados com o auxílio do MatLab. Estes incluem um controlador de avanço de fase, um controlador PID e um controlador PD, sendo este último concebido no MatLab.

Primeiramente, todos os controladores devem seguir o mesmo diagrama de blocos:

Figura 28 — Diagrama de blocos



Fonte: Autores (2023).

O primeiro controlador projetado será o avanço de fase para o controle da posição do motor. Para isso, foram determinados os requisitos do projeto: um tempo de assentamento ($T_S(2\%)$) de 1 segundo e um sobressinal ($M_{pt}(\%)$) de 1%. Sabendo disso, é possível começar o projeto do controlador. Sabendo que:

$$G_c(s) = \frac{k_c(s + z_c)}{s + p_c}, z_c < p_c$$

O primeiro passo para projetar o controle é definir os pólos desejados:

$$\left.\begin{aligned} \zeta &= \sqrt{\frac{ln^2 M_p}{\pi^2 + ln^2 M_p}} = \sqrt{\frac{ln^2 0{,}01}{\pi^2 + ln^2 0{,}01}} = 0{,}826 \\ T_s(2\%) &= \frac{4}{\sigma} = \frac{4}{\zeta\omega_n} \rightarrow 1 = \frac{4}{0{,}826 * \omega_n} \therefore \omega_n = 4{,}842\ rad/s \\ &= -4 \pm 2{,}7293j \end{aligned}\right\} p_d = -\zeta\omega_n \pm \omega_n\sqrt{1 - \zeta^2}j$$

Depois de calcular os pólos desejados, é possível começar o cálculo do controlador. O primeiro passo é assumir o zero do controlador igual a parte real do pólo desejado, ou seja, $z_c = 4$. Portanto, temos:

$$G_c(s) = \frac{k_c(s + 4)}{s + p_c}$$

O próximo passo é utilizar o critério de fase para encontrar o valor de p_c:

$$\angle\left(\left(\frac{K_c * (s+4)}{s + 8{,}46}\right) * \left(\frac{24{,}8345}{s * (0{,}2579s + 1)}\right)\right)_{s = -4 + 2{,}7293j} = -180 \rightarrow \therefore p_c = 8{,}46$$

Com o pólo do controlador calculado, encontra-se o ganho do controlador, k_c . Para isso, será usado o critério do módulo:

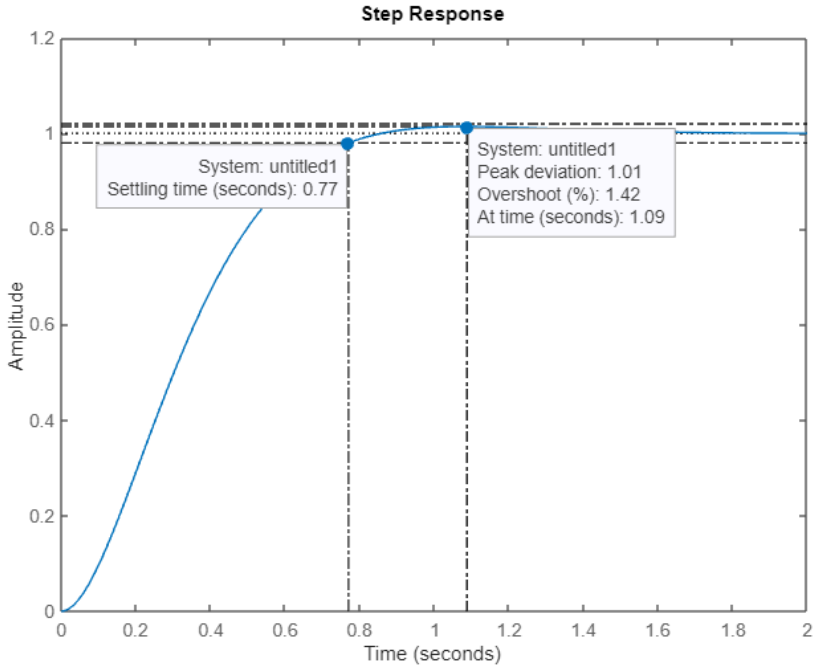
$$\left|\left(\frac{K_c * (s+4)}{s + 8{,}46}\right) * \left(\frac{24{,}8345}{s * (0{,}2579s + 1)}\right)\right|_{s = -4 + 2{,}7293j} = 1 \rightarrow \therefore k_c = 0{,}2632$$

Depois de obter seus parâmetros, foi obtido o seguinte controlador:

$$G_c(s) = \frac{0{,}2632(s+4)}{s + 8{,}46}$$

Para avaliar se o controlador calculado atende aos parâmetros do projeto, é possível simular sua resposta no MATLAB.

Figura 29 — Resposta do sistema controlado



Fonte: Autores (2023).

Outro controlador proposto é o PID, mas para este será necessário utilizar outro modelo, aquele que contém dois polos diferentes de 0, como mencionado anteriormente:

$$G_M = \frac{92{,}154}{S^2 + 4{,}002s + 4{,}8 * 10^{-8}} = \frac{92{,}154}{(s+4{,}002)(s + 1{,}2 * 10^{-8})}$$

O controlador PID pode ser escrito como visto abaixo, e ao final do processo os valores de Kp, Ti e Td devem ser definidos.

$$G_{PID} = k_p * \left(1 + \frac{1}{T_i s} + T_D s\right) = k_p \left(\frac{T_i T_D s^2 + T_i s + 1}{T_i s}\right)$$

O primeiro passo é encontrar os valores de Ti e Td a fim de “cancelar” os pólos da planta. Para isso, é necessário normalizar o denominador de G_M:

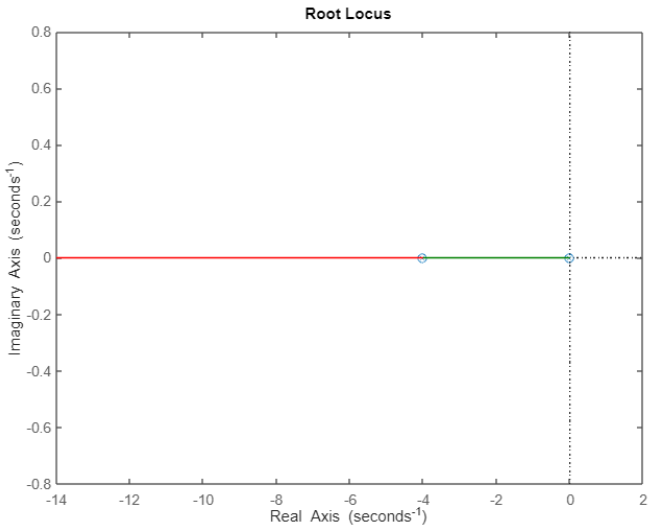
$$s^2 + 4{,}002s + 4{,}8 * 10^{-8} \rightarrow \frac{s^2}{4{,}8 * 10^{-8}} + \frac{4{,}002}{4{,}8 * 10^{-8}}s + 1$$

Com o denominador normalizado é possível estabelecer o seguinte sistema:

$$\begin{cases} T_i T_D = \dfrac{1}{4{,}8 * 10^{-8}} \\ T_i = \dfrac{4{,}002}{4{,}8 * 10^{-8}} \end{cases} \rightarrow \begin{cases} T_D = 0{,}2499 \\ T_i = 83375000 \end{cases}$$

Nesse momento, foram obtidos os valores de Ti e Td, faltando calcular o valor de Kp. Porém, antes disso, deve-se olhar o diagrama do lugar raízes admitindo $k_p = 1$:

Figura 30 — Lugar geométrico das raízes em malha aberta do sistema controlado



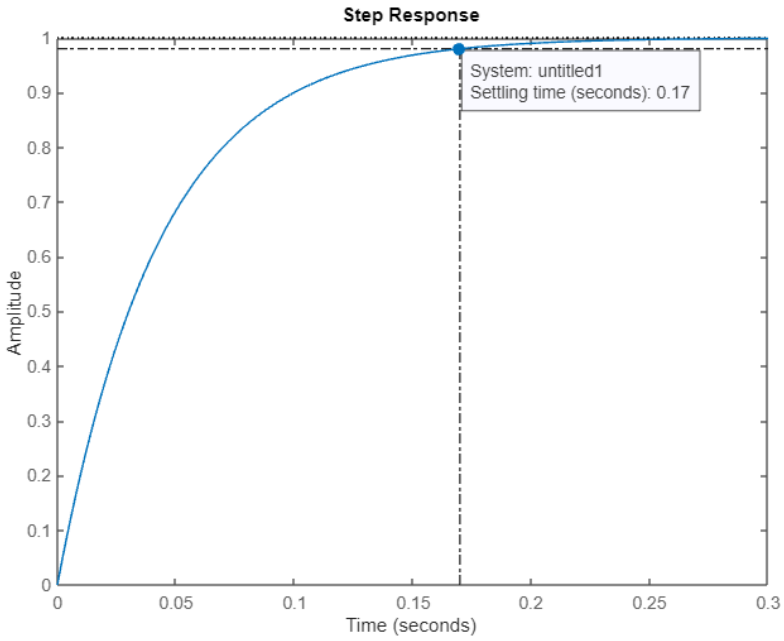
Fonte: Autores (2023).

É possível verificar que, independentemente do valor do k_p , a resposta do sistema nunca será instável e nunca irá oscilar, portanto, a única influência de k_p está no tempo de resposta do sistema. Portanto, k_p = 1 já satisfaz os requisitos do projeto.

Admitindo o controlador projetado, temos o seguinte controlador e a resposta simulada no MatLab:

$$G_{PID} = (1 + \frac{1}{83375000 * s} + 0{,}2499 * s)$$

Figura 31 — Resposta do sistema controlado



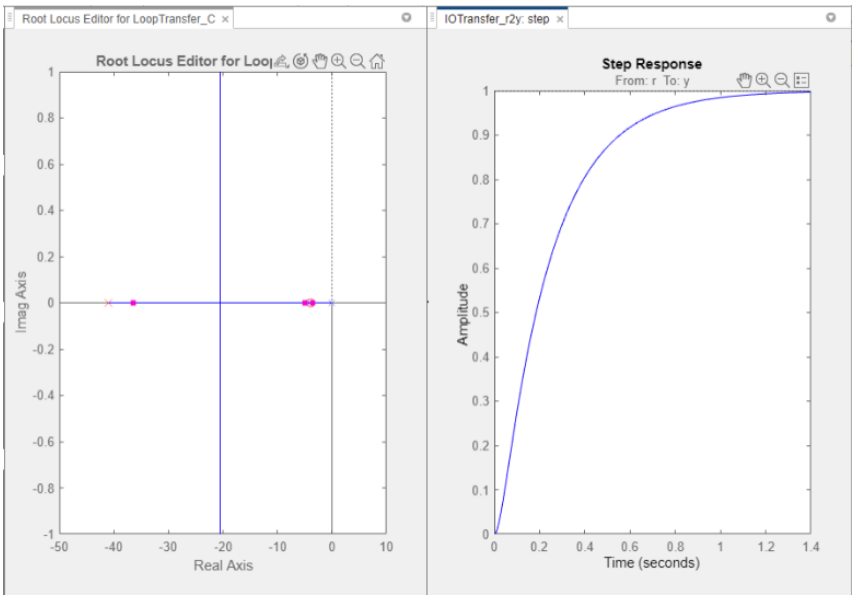
Fonte: Autores (2023).

O resultado obtido é satisfatório para o projeto do controlador. Além dos controladores de avanço de fase e PID mencionados, será apresentada uma comparação com um controlador PD desenvolvido por meio das ferramentas do MatLab, especificamente o RLTOOL.

A otimização do resultado foi realizada de acordo com os requisitos do projeto, considerando a sensibilidade ao ganho do controlador. Os resultados obtidos incluem a configuração específica do controlador e a saída correspondente da planta controlada por esse dispositivo.

$$G_C(s) = 0{,}17016 * \frac{(1 + 0{,}25s)}{(1 + 0{,}024s)}$$

Figura 32 — Ferramenta do MatLab para projetar o controlador

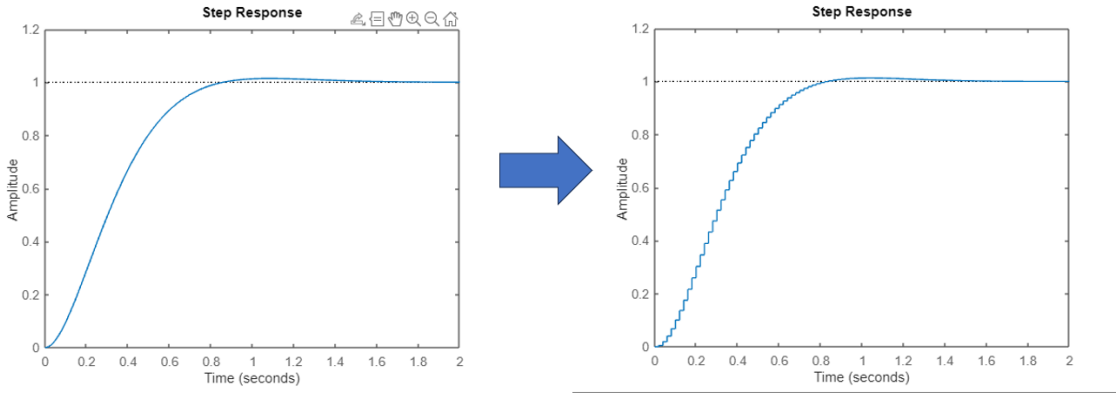


Fonte: Autores (2023).

3.4.4 CONTROLE EMBARCADO

Com o objetivo de integrar um controlador em um microcontrolador ou CLP, por exemplo, torna-se necessário discretizar o controlador projetado em tempo contínuo. Essa discretização implica que o controlador será atualizado em intervalos específicos definidos por T, denominado tempo de amostragem. A seguir, apresenta-se um exemplo de um sistema discretizado:

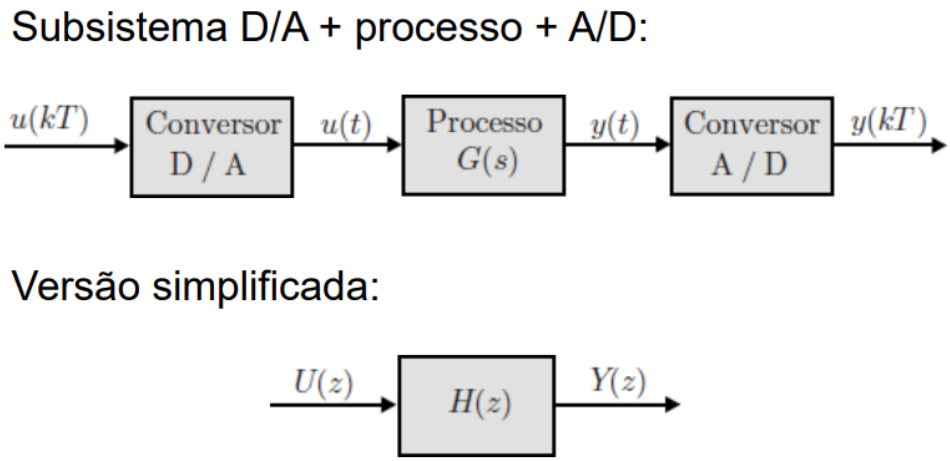
Figura 33 — Representação do sistema discreto



Fonte: Autores (2023).

Para isso, será utilizada a função de transferência do nosso controlador e a discretização será feita usando a transformada Z e a simplificação dos diagramas com os conversores analógicos/digitais e digitais/analógicos, como na imagem e equação abaixo:

Figura 34 — Diagrama de blocos de um sistema discretizado.



Fonte: Autores (2023).

$$\text{com } H(z) = (1 - z^{-1})\, \mathcal{Z}\left[\frac{G(s)}{s}\right]$$

No primeiro momento, o cálculo será demonstrado utilizando o controlador de avanço de fase. Porém, os outros dois controladores serão construídos utilizando a ferramenta do MatLab.

$$H_{af}(z) = (1 - z^{-1})\, \mathcal{Z}\left[\frac{0{,}2632(s + 4)}{s + 8{,}46}{s}\right] = (1 - z^{-1})\, \mathcal{Z}\left[\frac{0{,}2632(s + 4)}{s(s + 8{,}46)}\right]$$

Para realizar a transformada Z foi utilizado o método da expansão em frações parciais:

$$\mathcal{Z}\left[\frac{0{,}2632\ (s\ +\ 4)}{s(s\ +\ 8{,}46)}\right]\ =\ \mathcal{Z}\left[\frac{A}{s}\ +\frac{B}{(s\ +\ 8{,}46)}\right]$$

$$\begin{cases} A = \lim\limits_{s\to 0}\left(\dfrac{0{,}2632(s\ +\ 4)}{s(s+8{,}46)}\right) * s = \dfrac{0{,}2632(0+4)}{(0+8{,}46)} = 0{,}1244 \\ B = \lim\limits_{s\to -8{,}46}\left(\dfrac{0{,}2632(s\ +\ 4)}{s(s+8{,}46)}\right) * (s+8{,}46) = \dfrac{0{,}2632(-8{,}46+4)}{-8{,}46} = 0{,}13876 \end{cases}$$

$$\mathcal{Z}\left[\frac{0{,}2632(s+4)}{s(s+8{,}46)}\right] = \mathcal{Z}\left[\frac{0{,}1244}{s} + \frac{0{,}13876}{(s+8{,}46)}\right] = 0{,}1244 * \frac{z}{z-1} + 0{,}13876 * \frac{z}{z - e^{-8{,}46*T}}$$

$$H_{af}(z) = (1 - z^{-1})\left(0{,}1244 * \frac{z}{z-1} + 0{,}13876 * \frac{z}{z - e^{-8{,}46*T}}\right)$$

$$H_{af}(z) = \left(\frac{z-1}{z}\right)\left(\frac{0{,}1244 * z * \left(z - e^{-8{,}46*T}\right) + 0{,}13876 * z * (z-1)}{(z-1)(z - e^{-8{,}46*T})}\right)$$

$$H_{af}(z) = \frac{0{,}1244 * \left(z - e^{-8{,}46*T}\right) + 0{,}13876 * (z-1)}{(z - e^{-8{,}46*T})}$$

Considerando um tempo de amostragem para os controladores de 10ms, T = 0.01, obteve-se a seguinte função discreta:

$$H_{af}(z) = \frac{0{,}1244 * (z - e^{-8{,}46*0{,}01}) + 0{,}13876 * (z-1)}{(z - e^{-8{,}46*0{,}01})} = \frac{0{,}2632 * z - 0{,}2531}{z - 0{,}9189}$$

Para otimizar o processo foi utilizada a função c2d, que resulta na função já discretizada a partir da função de transferência em tempo contínuo e o tempo amostral. O resultado está descrito nas seguintes funções:

$$H_{PD}\,(z)\ =\ \frac{1{,}772 * z\ -\ 1{,}715}{z\ -\ 0{,}6592}$$

$$H_{PID}\,(z)\ =\ \frac{1{,}509 * z^2\ -\ 2{,}94 * z\ +\ 1{,}431}{z^2\ -\ 1{,}607 * z + 0{,}6065}$$

O próximo passo é encontrar a equação de diferenças utilizando a manipulações algébricas e a propriedade do atraso, $\mathcal{Z}\{u[k\ -1]T\}\ =\ z^{-1}U(z)$. O cálculo será novamente desenvolvido para o controlador de avanço de fase, e os outros dois controladores terão exatamente o mesmo cálculo.

$$\frac{\mathrm{Y(z)}}{\mathrm{U(z)}} = \frac{0{,}2632 * \mathrm{z}\ -\ 0{,}2531}{\mathrm{z}\ -\ 0{,}9189} = \frac{\mathrm{z}(0{,}2632\ -\ 0{,}2531\mathrm{z}^{-1})}{\mathrm{z}(1\ -\ 0{,}9189\mathrm{z}^{-1})}$$

$$\mathrm{Y(Z)}\ -\ 0{,}9189\mathrm{z}^{-1}\mathrm{Y(z)}\ =\ 0{,}2632\,\mathrm{U(z)}\ -\ 0{,}2531\mathrm{z}^{-1}\mathrm{U(z)}$$

$$\mathrm{y_{af}(kT)}\ =\ 0{,}2632\,\mathrm{u[kT]}\ -0{,}2531\mathrm{u}\,[(\mathrm{k}\ -\ 1)\mathrm{T}]\ +\ 0{,}9189\mathrm{y_{af}}[(\mathrm{k}\ -\ 1)\mathrm{T}]$$

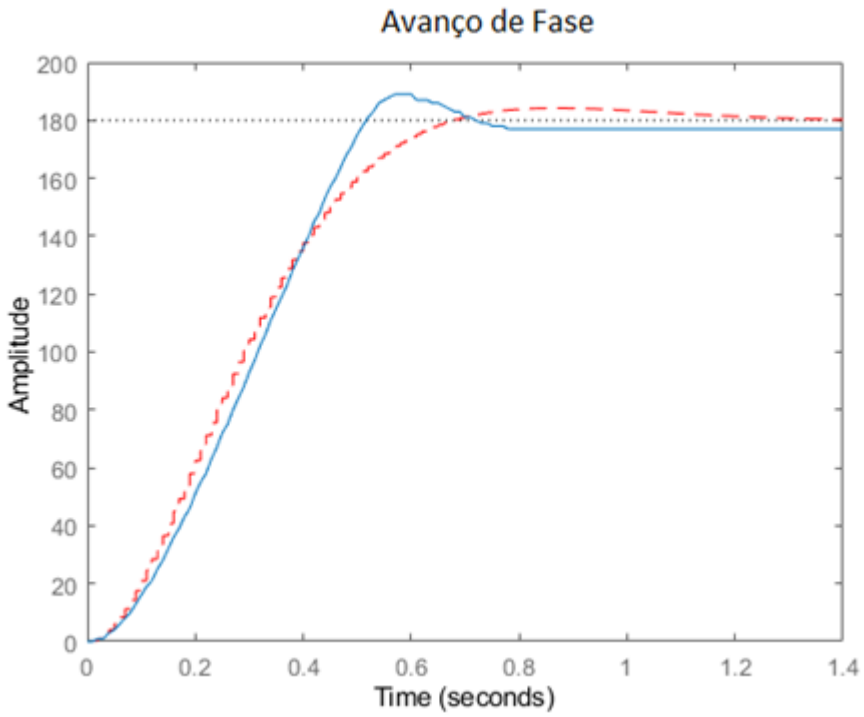
Sendo “u” a entrada, o erro, e “y” a saída do sistema, o PWM, aplicada no motor. Aplicando o mesmo cálculo para os outros controladores achamos as seguintes equações de diferenças:

$$y_{PD}(kT)\ =\ 1{,}772u\,[kT]\ -\ 1{,}715\,u[(k\ -\ 1)T]\ +\ 0{,}6592y_{PD}[(k\ -\ 1)T]$$

$$y_{PID}(kT)\ =\ 1{,}509u[kT]\ -\ 2{,}94u[(k\ -1)T]\ +\ 1{,}431u[(k\ -2)T]\ +\ 1{,}607y_{af}[(k\ -\ 1)T]\ -\ 0{,}6065y_{PID}[(k-2)]$$

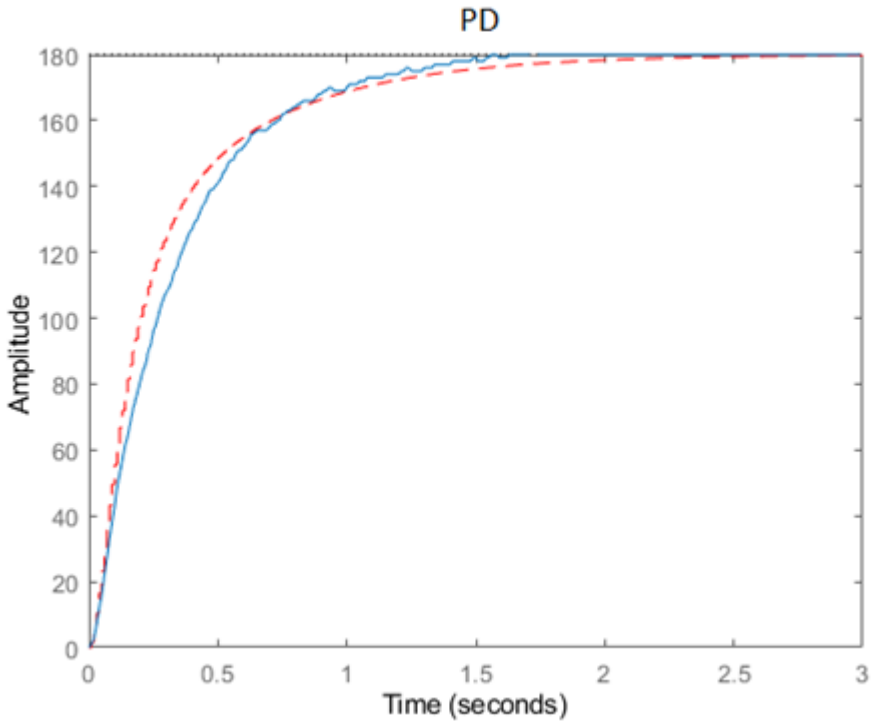
Com essas equações de diferenças é possível implementá-las no microcontrolador e testá-las a fim de compará-las com os resultados simulados, como demonstrado nos seguintes gráficos:

Figura 35 — Comparação do sistema em Avanço de Fase



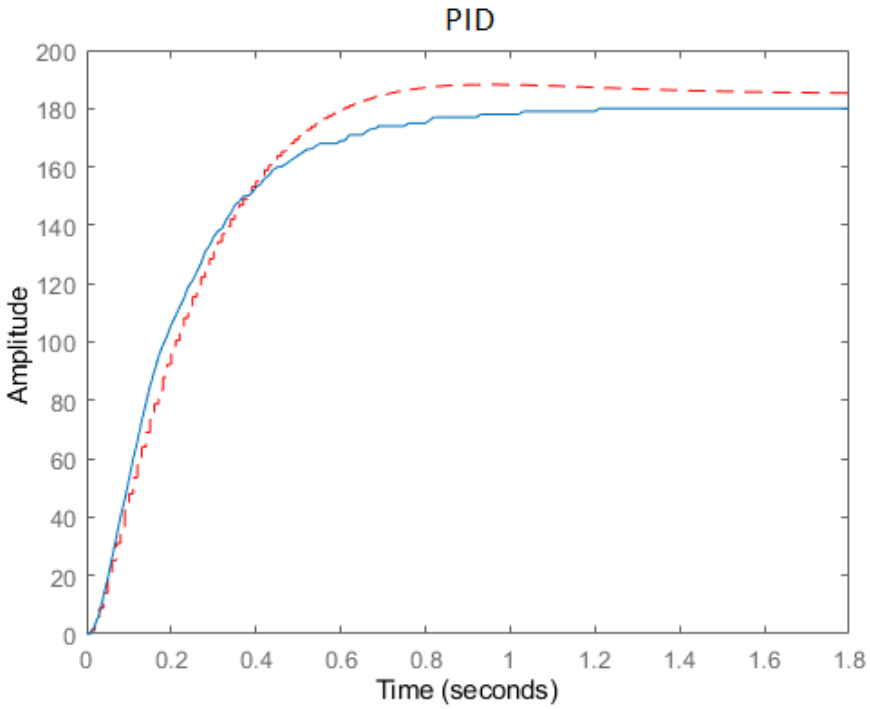
Fonte: Autores (2023).

Figura 36 — Comparação do sistema em Proporcional e Derivativo



Fonte: Autores (2023).

Figura 37 — Comparação do sistema em PID



Fonte: Autores (2023).

A linha vermelha tracejada representa a curva simulada e a linha azul representa a curva do sistema real.

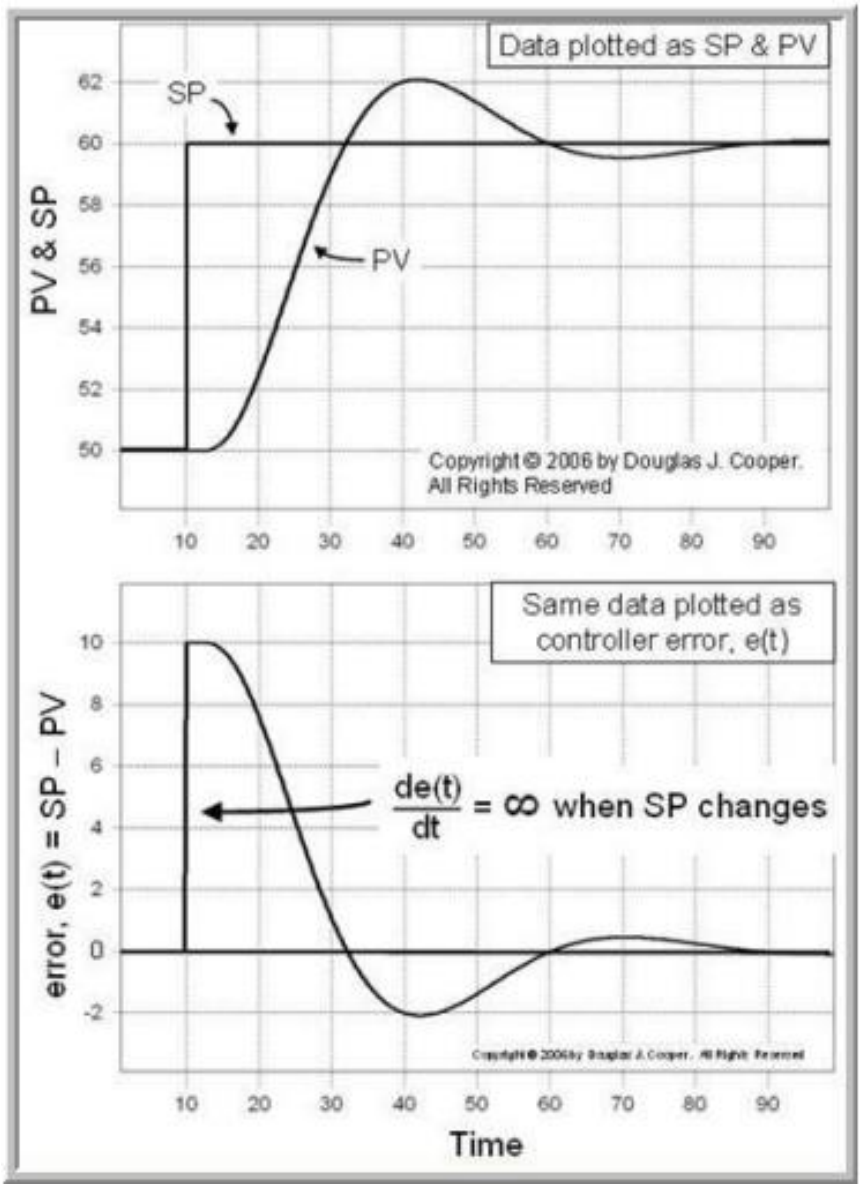
Outro controle discretizado que é importante ser mencionado está compreendido na primeira parte do projeto, no qual os ganhos de um controlador PID são interativos. Nessa parte, existe um controlador discreto implementado da seguinte forma:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t)$$

A componente integrativa é obtida pelo somatório do produto do erro pela variação do tempo (Δt), enquanto a componente derivativa resulta da diferença dividida pelo mesmo intervalo de tempo. Normalmente, essa diferença é calculada entre o erro atual e o anterior, mas essa abordagem pode levar a uma derivada que tende ao infinito. Para contornar esse problema, optou-se pela variação da variável do processo, utilizando a posição em vez da diferença entre os erros. Assim, realiza-se a subtração entre a posição atual e a posição anterior, invertendo

o comportamento da derivada. Para corrigir essa inversão, a parcela da derivada é subtraída em vez de somada, como evidenciado na comparação a seguir:

Figura 38 — Comparação dos métodos



Dessa forma, obtém-se a seguinte função:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)\,dt + K_d \frac{d}{dt} PV(t)$$

4 RESULTADOS E DISCUSSÃO

Ao se observar o projeto já finalizado, constata-se que a aplicação tanto do método analítico quanto do método por software para análise de velocidade ou posição resultou na obtenção de modelos do motor que são bastante próximos, apresentando pequenos desvios, como previamente mencionado. Portanto, ambas as abordagens demonstraram eficácia em atender ao objetivo proposto, que era a obtenção do modelo do motor por meio dos ensaios realizados.

Um ponto crucial identificado nas análises é a presença de disparidades na resposta do sistema, influenciadas por uma variedade de fatores, entre os quais se destaca a folga existente na caixa de redução. Essa folga exerce um impacto significativo na dinâmica do sistema. Considerando a impossibilidade de eliminação completa desse efeito, sugere-se a avaliação e possível adoção de uma caixa de redução alternativa, visando aprimorar a concordância dos resultados obtidos com as expectativas estabelecidas.

Outro aspecto relevante abordado refere-se à zona morta do motor, caracterizada por um intervalo de tensões entre 0V e um valor específico. Nesse intervalo, o motor permanece inerte devido a considerações mecânicas, predominantemente associadas ao atrito estático e momentos. Essa identificação proporciona insights valiosos para o entendimento e aprimoramento do comportamento do sistema em situações específicas.

Apesar das sutis discrepâncias observadas, é notório que o projeto atendeu plenamente às expectativas, permitindo a comparação eficiente da eficiência entre diferentes controladores e métodos. Além disso, proporcionou uma compreensão abrangente dos conceitos abordados nas disciplinas pertinentes, consolidando, assim, o sucesso da pesquisa em alcançar seus objetivos e contribuir para o avanço do conhecimento na área em questão.

5 CONCLUSÕES

O projeto abordado neste relatório representa um significativo avanço na aplicação prática e interdisciplinar dos conceitos aprendidos nas disciplinas de Instrumentação, Microcontroladores, Sistemas de Controle I, Programação Orientada a Objetos e Banco de Dados. A criação de um simulador interativo para controle de velocidade e posição de um motor acoplado a uma roda de inércia é um marco que evidencia a convergência de hardware e software na engenharia, proporcionando uma experiência prática e holística aos estudantes.

Os objetivos primordiais do projeto foram alcançados com sucesso, destacando-se a criação de um ambiente virtual que possibilita a aplicação prática dos conceitos teóricos estudados. O uso do PIC16F18877 para controlar o motor, juntamente com a implementação e validação dos sistemas de controle de velocidade e posição, demonstra a eficácia da abordagem adotada. As fases do projeto, desde o desenvolvimento do ambiente virtual até a construção do hardware, foram estruturadas de maneira lógica e coerente.

O simulador desenvolvido oferece uma plataforma eficaz para experimentação e compreensão de conceitos-chave, como controle PID, identificação matemática do sistema e programação de microcontroladores. A capacidade de simular malhas de controle fechadas distintas em um mesmo dispositivo é uma contribuição valiosa, permitindo a comparação direta entre diferentes estratégias de controle. Essa análise comparativa é crucial para otimizar o desempenho do controle desejado e aprofundar a compreensão dos princípios de controle.

O desenvolvimento deste simulador interativo representa não apenas a aplicação prática de conhecimentos adquiridos, mas também destaca a importância da integração interdisciplinar na formação de profissionais de engenharia. A abordagem adotada não só proporciona uma compreensão mais profunda dos conceitos teóricos, mas também prepara os estudantes para enfrentar desafios do mundo real, onde a colaboração entre diversas áreas de conhecimento é essencial.

6 REFERÊNCIAS

LIN, Paul-l-Hai; HWANG, Sentai; CHOU, J. **Comparison on fuzzy logic and PID controls for a DC motor position controller.** In: Proceedings of 1994 IEEE Industry Applications Society Annual Meeting, Denver, CO, USA, 1994. p. 1930-1935 vol.3. DOI: 10.1109/IAS.1994.377695.

BAR-KANA, I.; FISCHL, R.; KALATA, P. **Direct position plus velocity feedback control of large flexible space structures.** In: IEEE Transactions on Automatic Control, vol. 36, no. 10, Oct. 1991. p. 1186-1188. DOI: 10.1109/9.90232.

MAHMUD, M.; MOTAKABBER, S. M. A.; ALAM, A. H. M. Z.; NORDIN, A. N. **Adaptive PID Controller Using for Speed Control of the BLDC Motor.** In: 2020 IEEE International Conference on Semiconductor Electronics (ICSE), Kuala Lumpur, Malaysia, 2020. p. 168-171. DOI: 10.1109/ICSE49846.2020.9166883.

BALAMURUGAN, S.; UMARANI, A. **Study of Discrete PID Controller for DC Motor Speed Control Using MATLAB.** In: 2020 International Conference on Computing and Information Technology (ICCIT-1441), Tabuk, Saudi Arabia, 2020. p. 1-6. DOI: 10.1109/ICCIT-144147971.2020.9213780.

Pololu - VNH2SP30 Motor Driver Carrier MD01B. Disponível em: <https://www.pololu.com/product/706>. Acesso em: 10 nov. 2023.

HARWANI, B. M. **Qt5 Python GUI Programming Cookbook Building responsive and powerful cross-platform applications with PyQt.** [s.l.] Birmingham ; Mumbai Packt July, 2018.

LGALVANI007. **Projeto Semestral**. Disponível em: <https://github.com/lgalvani007/ProjetoSemestral>. Acesso em: 14 nov. 2023.