UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Rafael Luís do Nascimento Santos

Projeto e Fabricação de um Aeropêndulo Reconfigurável para Ensino de Conceitos Relacionados ao Controle de Sistemas Dinâmicos.

Recife
2024

Rafael Luís do Nascimento Santos

Projeto e Fabricação de um Aeropêndulo Reconfigurável para Ensino de Conceitos Relacionados ao Controle de Sistemas Dinâmicos.

Monografia submetida ao Departamento de Engenharia Mecânica, da Universidade Federal de Pernambuco - UFPE, para conclusão do curso de Graduação em Engenharia Mecânica

Orientador: Pedro Manuel Gonzalez del Foyo

Recife
2024

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Santos, Rafael Luís do Nascimento.
Projeto e fabricação de um aeropêndulo reconfigurável para ensino de conceitos relacionados ao controle de sistemas dinâmicos / Rafael Luís do Nascimento Santos. - Recife, 2024.
118p : il., tab.

Orientador(a): Pedro Manuel Gonzalez del Foyo
(Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, , 2024.
Inclui referências, apêndices.

1. Plataforma didática. 2. Ensino do controle. 3. Bancada Aeropêndulo. I. Foyo, Pedro Manuel Gonzalez del. (Orientação). II. Título.

620 CDD (22.ed.)

Rafael Luís do Nascimento Santos

Projeto e Fabricação de um Aeropêndulo Reconfigurável para Ensino de Conceitos Relacionados ao Controle de Sistemas Dinâmicos.

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia Mecânica da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Engenheiro Mecânico

Aprovado em: 01/07/2024

BANCA EXAMINADORA

_____

Prof. Pedro Manuel Gonzalez del Foyo (Orientador)

Universidade Federal de Pernambuco

_____

Prof. João Paulo Cerquinho Cajueiro (Examinador Interno)

Universidade Federal de Pernambuco

_____

Prof. Eduardo José Novaes Menezes (Examinador Externo)

Universidade Federal de Pernambuco

RESUMO

Nos cursos de Engenharia Mecânica, as disciplinas que ensinam conceitos relacionados ao controle de sistemas dinâmicos estão presentes na maioria dos currículos. E muitas vezes o ensino desses conceitos de forma teórica possui um grau de abstração elevado, sendo necessária a complementação com um ensino prático. O presente trabalho apresenta o desenvolvimento de uma bancada didática para uso no ensino de conceitos realizados ao controle de sistemas dinâmicos. Embora já exista uma bancada usada para o ensino, algumas modificações eram necessárias para aumentar a robustez e atender algumas especificações relacionadas ao desempenho mecânico e facilidade de execução dos experimentos. O sistema escolhido para a nova bancada foi mantido como o aeropêndulo, que já havia sido usado anteriormente. Essa escolha se deve a sua simplicidade, e ao comportamento do sistema mecânico que pode apresentar variabilidade dependendo de parâmetros mecânicos, que são facilmente modificáveis. O foco das melhorias foi permitir a variabilidade do sistema permitindo a obtenção de diferentes modelos mecânicos facilmente, a partir da montagem e desmontagem de componentes. Essa intercambiabilidade também permite o uso de diferentes motores, podendo alterar também as características eletrônicas do sistema. Além disso, foi desenvolvida uma interface que permitisse aos alunos realizar todos os experimentos necessários para testar os conhecimentos no ensino de controle. Ela tem o objetivo de facilitar a extração de dados de um sistema físico para interpretação dos dados temporais e identificação de sistemas. Por fim, foram realizados os testes de integração das funcionalidades do sistema, exemplificando o procedimento de identificação de uma configuração de aeropêndulo e mostrando alguns dos conceitos que devem ser analisados pelos alunos no decorrer das práticas desenvolvidas na disciplina.

Palavras-chave: Plataforma didática, Ensino do controle, Bancada Aeropêndulo.

ABSTRACT

In Mechanical Engineering courses, subjects that teach concepts related to the control of dynamic systems are present in most curricula. And often the teaching of these concepts in a theoretical way has a high degree of abstraction, requiring complementation with practical teaching. This work presents the development of a teaching bench for use in teaching concepts related to the control of dynamic systems. Although a bench was already used for teaching, some modifications were necessary to increase robustness and meet some specifications related to mechanical performance and ease of carrying out experiments. The system chosen for the new bench was maintained as the aeropendulum, which had already been used previously. This choice is due to its simplicity, and the behavior of the mechanical system, which may present variability depending on mechanical parameters, which are easily modifiable. The improvements allow different mechanical models to be obtained easily by assembling and disassembling components. This interchangeability also allows the use of different motors, and can also change the electronic characteristics of the system. Furthermore, an interface was developed that enabled students to carry out all the experiments necessary to test their knowledge in the concepts of control. It aims to facilitate the extraction of data from a physical system for interpretation of temporal data and system identification. Finally, integration tests of the system's functionalities were carried out, exemplifying the procedure for identifying an aeropendulum configuration and showing some of the concepts that must be analyzed by students during the practices developed in the discipline.

Keywords: Didactic platform, Control teaching, Aeropendulum Bench.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AeE	Agir e Esperar
CC	Corrente Contínua
CEV	Controle Eletrônico de Velocidade
CNE	Conselho Nacional de Educação
CSV	Comma Separated Values
GMP	Grupo Motopropulsor
LARC	Laboratório de Automação Robótica e Controle
MCCSE	Motor CC Sem Escovas
MLP	Modulação por Largura de Pulso
PCI	Placa de Circuito Impresso
PID	Proporcional Integrativo Derivativo
SBPA	Sinal Binário Pseudo Aleatório
UFPE	Universidade Federal de Pernambuco

LISTA DE SÍMBOLOS

g	Aceleração Gravitacional	m/s^2
θ	Ângulo da Haste	rad
C_D	Coeficiente de Arrasto	
c	Coeficiente de Fricção do Aeropêndulo	$N.M.s/rad$
B_v	Coeficiente de Fricção do Motor	
C_L	Coeficiente de Sustentação	
K_d	Coeficiente Derivativo	
K_i	Coeficiente Integrativo	
K_p	Coeficiente Proporcional	
λ	Constante de Tempo	
K_T	Constante de Torque	
K_v	Constante Elétrica	
K_e	Constante Elétrica Eletromotriz	
K_g	Constante Geométrica	
i	Corrente Elétrica	A
e_a	Força Contra Eletromotriz	V
F_t	Força De Empuxo	N
L	Indutância	T
J_P	Momento de Inércia do Aeropêndulo	$kg.m^2$
J_m	Momento de Inércia do Motor	$kg.m^2$
T_s	Período De Amostragem	s
R	Resistência Elétrica	Ω
u	Tensão Elétrica	V
T_e	Torque Eletromagnético	N

ω	Velocidade Angular	rad/s
V_r	Velocidade Linear de rotação	m/s

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

Segundo as Diretrizes Curriculares Nacionais para cursos de Graduação em Engenharia, definidas pelo CNE (CNE, 2002), os mesmos devem conter um mínimo de 15% da sua carga horária para disciplinas profissionalizantes, dentre as quais, se encontra o Controle de Sistemas Dinâmicos. Nos cursos de Engenharia Mecânica, as disciplinas que envolvem estes conhecimentos e habilidades estão na grade curricular de disciplinas obrigatórias em 9 das 10 instituições de ensino superior brasileiras melhor avaliadas[1].

Estas disciplinas contêm em suas ementas os conteúdos relacionados modelagem de sistemas, análise da estabilidade, as especificações da resposta em regime transiente e permanente e técnicas de desenvolvimento de controladores em malha fechada.

No curso de Engenharia Mecânica da UFPE, duas disciplinas da grade curricular obrigatória são dedicadas ao ensino dos conceitos relacionados a Teoria de Controle. A disciplina ME443 Engenharia de Controle 1 onde são estudados estes conceitos de forma teórica e ME472 Laboratório de Automação e Controle onde é realizada uma abordagem prática.

O efeito do uso de atividades práticas para o aprendizado é bem conhecido por melhorar o desempenho do ensino e a fixação de conhecimento. Segundo (AGUADO, 2009), ao aplicar esse conceito no ensino prático de pesquisa, os estudantes melhoraram sua disposição e habilidades de aprendizagem de forma significativa, tornando o ensino muito mais estimulante.

As atividades práticas desenvolvidas na disciplina ME472 são realizadas em turmas de no máximo 8 alunos, onde os alunos se dividem em grupos de até 4 para executarem as atividades práticas nas bancadas didáticas. As bancadas didáticas usadas são aeropêndulos desenvolvidos em Gonçalves (2018) durante a realização de seu trabalho de conclusão de curso.

Outras bancadas similares foram desenvolvidas no LARC, como as baseadas no

[1] Foram revisados os PPCs dos cursos de Engenharia Mecânica das 10 mais bem colocadas de acordo com o ranking Scimago (2024).

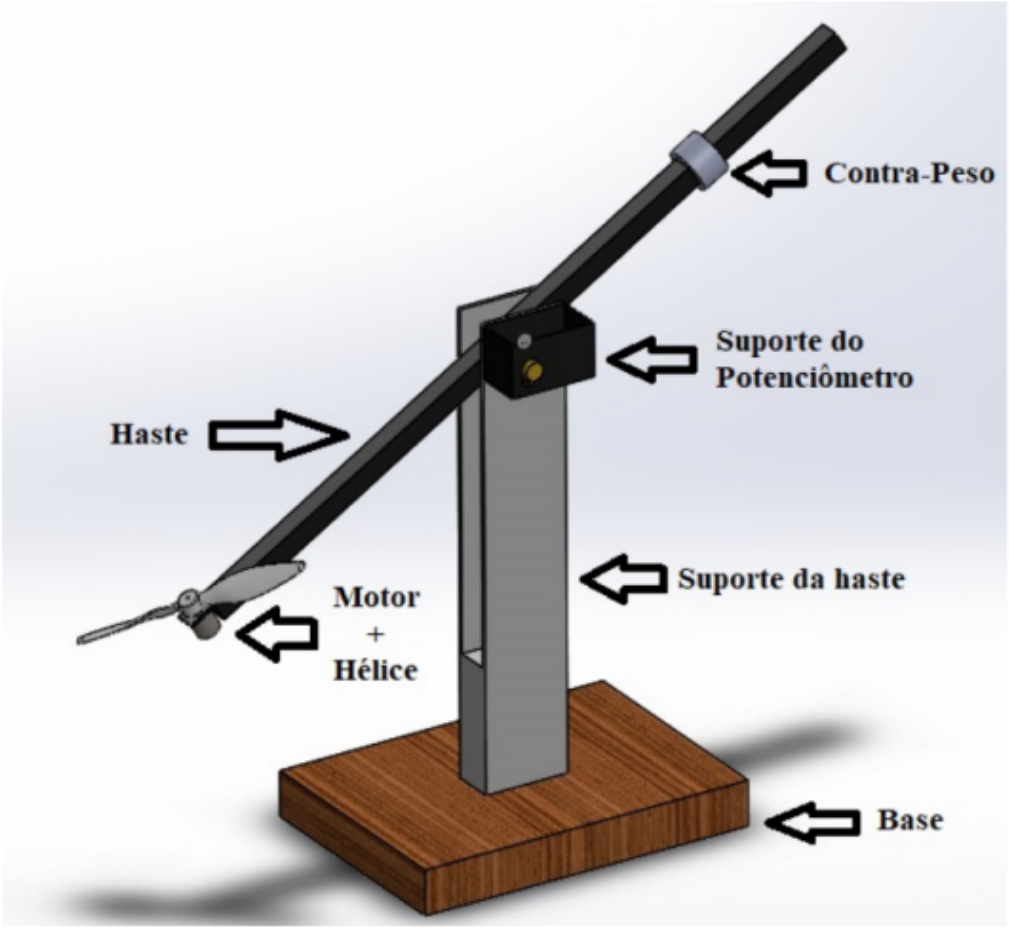
sistema com pêndulo com volante de inércia (VASCONCELOS *et al.*, 2018), (CARDOSO *et al.*, 2022) embora seu uso esteja ligado a temas um pouco mais complexos de controle, vistos numa disciplina eletiva.

Do ponto de vista do ensino dos conceitos básicos da teoria de controle, o aeropêndulo é uma escolha natural devido a sua simplicidade, associada aos poucos componentes necessários para o funcionamento, e ao comportamento do sistema mecânico, que pode ser sub-amortecido, amortecido ou sobre-amortecido dependendo de parâmetros mecânicos facilmente modificáveis. Tanto é assim que existem vários trabalhos mostrando o uso de este sistema no ensino desta área de estudo (ENIKOV; CAMPA, 2012a), (GONÇALVES, 2018), (NETO *et al.*, 2023).

1.2 Justificativa

A Figura 1 mostra o aeropêndulo desenvolvido como trabalho de conclusão de curso de um aluno, disponível no LARC para as práticas de controle.

Figura 1 – Modelo de um aeropêndulo



Fonte: Gonçalves (2018)

Embora a bancada desenvolvida tenha funcionado desde 2019, algumas dificuldades têm sido identificadas através dos relatórios de monitória de vários períodos (SOUZA; FOYO, 2022) (ROCHA; FOYO, 2022) (SANTOS; FOYO, 2023). Dentre os problemas estão o ruído devido ao uso do potenciômetro para medição e a dificuldade no uso da pacote Simulink Real-Time do MATLAB devido ao travamento do computador durante a realização dos experimentos.

Além das dificuldades da bancada em si, a análise dos relatórios também mostrou que a modelagem analítica do sistema era a parte na qual os alunos apresentavam as maiores dificuldades, em especial na etapa de modelar a força de empuxo e sua relação com a tensão aplicada ao motor.

A bancada atual permite a variação da função de transferência do sistema apenas através da mudando o local do contrapeso, gerando funções de transferência diferentes. Contudo, a parte elétrica era fixa e invariável, não permitindo a exploração total de modelos possíveis.

Em vista de todos os fatores mencionados, chegou-se a conclusão de que deve ser projetado uma nova bancada que além de consertar as imperfeições da anterior, também incorporar a variabilidade para montar e desmontar para que os alunos possam obter as propriedades elétricas como também mecânicas. Permitindo a análise de um modelo analítico através desses parâmetros, para que assim acrescente a capacidade de aprendizado dos alunos.

Sendo assim, optou-se pelo projeto e construção de um protótipo de aeropêndulo para ser usado no ensino do controle, objetivando melhorar a base de bancadas didáticas disponíveis no LARC para o ensino desta disciplina. Uma característica fundamental é a capacidade de montagem e desmontagem, para obter sistemas de aeropêndulos com características mecânicas diferentes. Desta forma as equipes poderão desenvolver seus trabalhos em duplas, sendo que cada uma delas terá um sistema com uma dinâmica diferente.

A parte eletrônica foi concebida para ser única em todas as bancadas, devido aos custos envolvidos e necessidade de usar uma plataforma de programação única e escalável. Desta forma o processo de montagem e desmontagem foi padronizado e uma interface foi desenvolvida para facilitar a realização dos propostos na disciplina.

1.3 Objetivos

1.3.1 Objetivo Geral

O objetivo geral é projetar uma bancada de aeropêndulo reconfigurável para ensino dos conceitos relacionados ao controle de sistemas dinâmicos.

1.3.2 Objetivos Específicos

1. Realizar levantamento dos principais componentes eletromecânicos para utilização no projeto;
2. Projetar e testar arquitetura elétrica desenvolvida;
3. Projetar estrutura mecânica que suporta os componentes eletrônicos necessários para o funcionamento da bancada;
4. Desenvolver e programar interface do usuário para realizar as funcionalidades exigidas para o ensino;
5. Realizar testes de integração do sistema e verificar se as funcionalidades foram satisfeitas.

2 FUNDAMENTAÇÃO TEÓRICA

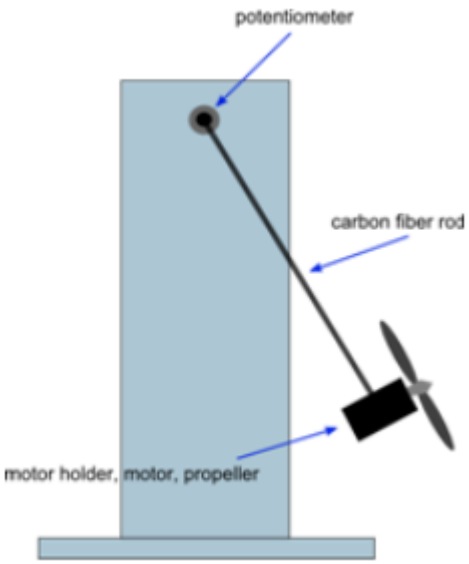
Neste capítulo são revisados conceitos teóricos necessários para o entendimento do trabalho, envolvendo a modelagem do aeropêndulo e o funcionamento dos principais componentes eletromecânicos utilizados. Por fim, é feito um breve resumo dos conceitos de controle e é apresentado uma discussão sobre alguns trabalhos similares.

2.1 O Aeropêndulo

O aeropêndulo é composto por um grupo motopropulsor, hastes e pode ou não possuir um contrapeso. Em seu contexto didático para a prática do controle, o objetivo é estabelecer uma lei de controle que mantenha um ângulo de referência θ escolhido a partir do controle da tensão do motor.

Um exemplo de pêndulo com haste em fibra de carbono é mostrado na Figura 2. Este sistema se classifica como um sistema de uma entrada (tensão) e uma saída (ângulo), o que o torna mais simples do que sistemas reais que, em geral, englobam múltiplas entradas e saídas.

Figura 2 – Aeropêndulo de fibra de carbono



Fonte: Enikov e Campa (2012b)

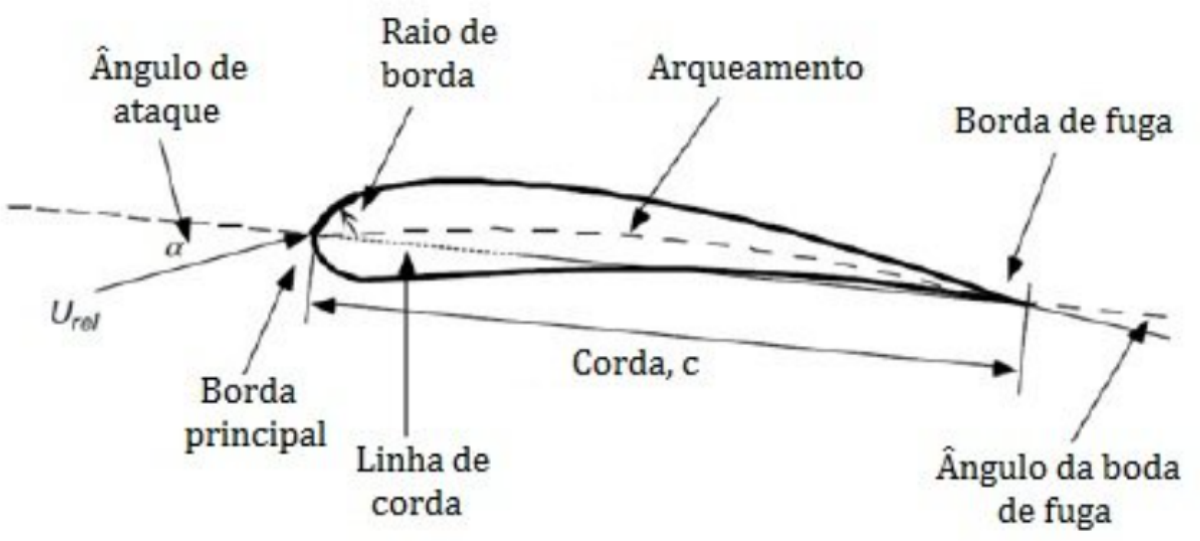
O comportamento angular do aeropêndulo é determinado pelos seus parâmetros geométricos e força de empuxo F_t. Por esse motivo é necessário entender os aspectos que influenciam essa força de atuação e como controlar o conjunto de propulsão de maneira dinâmica.

2.1.1 A força de empuxo

O conjunto de propulsão composto por motor e hélice é responsável por gerar a força de empuxo, utilizada como atuador do sistema. As hélices podem ser descritas como um aerofólio rotativo que gera um movimento do ar, reagindo aerodinamicamente e gerando uma força aerodinâmica resultante (NELSON, 1944). Desse modo, o desempenho do conjunto de atuação também depende da escolha correta da hélice.

A Figura 3 mostra as estruturas de um aerofólio típico. É um formato arqueado que possui uma borda principal arredondada e uma borda de fuga mais fina, a interseção desses pontos forma a corda. O ângulo entre a corda e a velocidade relativa da corrente de ar incidente sobre o aerofólio define o chamado ângulo de ataque. O ângulo de ataque é uma composição do ângulo geométrico da pá e da velocidade resultante, que são diferentes, pois a hélice se movimenta tanto rotacionalmente como translada no espaço (OST, 2015).

Figura 3 – Estrutura de um aerofólio



Fonte: Okita e Moura (2017)

O principal meio de avaliar o desempenho de um aerofólio é através da análise de seus coeficientes de arrasto e sustentação, que, em geral, são determinados numericamente ou por meio de experimentos. Os coeficientes de sustentação C_L e arrasto C_D são definidos como:

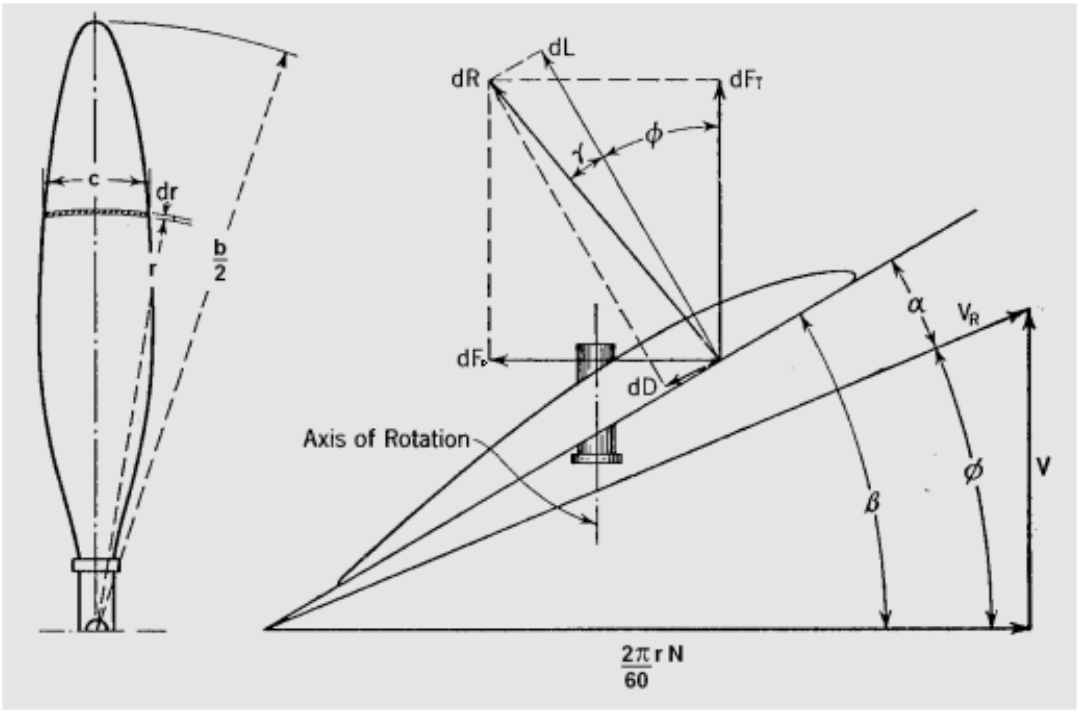
$$C_L = \frac{L}{(1/2)\rho V_r^2 A} \tag{2.1}$$

$$C_D = \frac{D}{(1/2)\rho V_r^2 A} \tag{2.2}$$

Onde L e D são as forças de sustentação e arrasto respectivamente, ρ é a densidade do ar, V_r é a velocidade de rotação e A é uma área de referência determinada.

Uma hélice pode ser descrita através da teoria de elementos de pás. Nessa teoria uma pá é dividida em elementos de pá ao longo de seu comprimento, de espessura dr e largura c conforme a Figura 4.

Figura 4 – Forças em um elemento de pá



Fonte: Modificado de Nelson (1944)

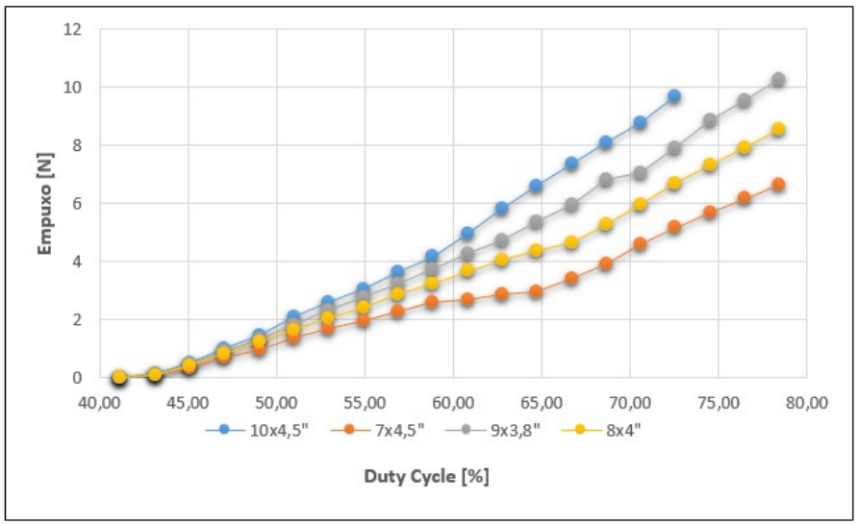
Cada seção tem uma contribuição dF_t e dF_D, cada uma com seu sua própria geometria variando ao longo do comprimento da pá. Dessa forma temos que $C_L = f(r)$ e $C_D = f(r)$. Para determinar as forças de arrasto e sustentação temos:

$$dL = \frac{1}{2} C_L \rho V_r^2 dA = \frac{1}{2} C_L \rho V_r^2 c dr \tag{2.3}$$

$$dD = \frac{1}{2} C_D \rho V_r^2 dA = \frac{1}{2} C_D \rho V_r^2 c dr \tag{2.4}$$

Em Pelke *et al.* (2017) foi analisado o efeito de diferentes geometrias de helices para um mesmo motor. A Figura 5 exemplifica esse comportamento mostrando a relação do empuxo para uma faixa de *Duty Cycle* (percentual que indica uma porcentagem da tensão de alimentação).

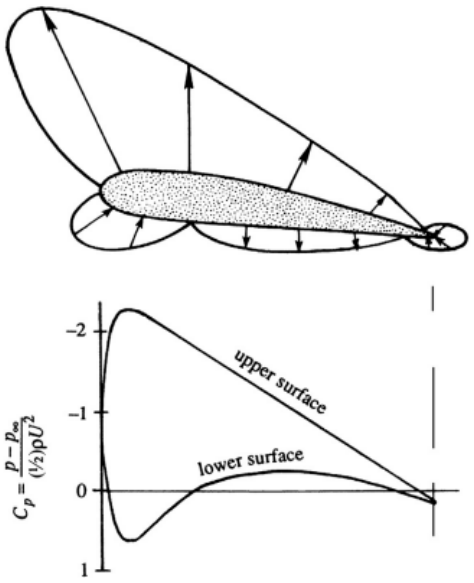
Figura 5 – Comportamento do empuxo com diferentes hélices



Fonte: Pelke *et al.* (2017)

A sustentação é resultado dos diferenciais de pressão que atuam sobre o aerofólio (Figura 6). A superfície inferior possui um coeficiente de pressão superior quando comparado ao componente de pressão negativo gerado na superfície superior. Esse diferencial de pressão é o fator que gera uma componente resultante de força para cima gerando a sustentação do aerofólio, também chamado de empuxo

Figura 6 – Coeficientes de pressão típicos de um aerofólio



Fonte: Kundu *et al.* (2016)

Como visto em 2.1 a força de empuxo F_t é determinada pela velocidade do motor, parâmetros geométricos da hélice, propriedades aerodinâmicas e a densidade do ar. Embora a relação entre força de empuxo e velocidade seja quadrática, é possível linearizar a expressão através da primeira derivada da expansão de Taylor em torno do ponto de operação, uma vez que as propriedades geométricas são constantes e os parâmetros aerodinâmicos podem ser considerados constantes. Considerando a linearização em torno de um ponto de operação com o uso de uma constante que engloba os parâmetros geométricos K_g temos:

$$F_t = K_g \omega \tag{2.5}$$

Como a variável manipulada no controle é a tensão aplicada no motor, é necessário relacionar a tensão aplicada com a força de empuxo. Para isso é necessário definir o comportamento do conjunto de acionamento.

2.1.2 O motor elétrico de corrente contínua sem escovas

O motor CC sem escovas é um motor síncrono controlado por um inversor que é alimentado com corrente contínua, geralmente com o uso de um manipulador[1]. Ele é um motor sem escovas que, por esse motivo, possui menos atrito e realiza a comutação via sensores de efeito hall e ímãs permanentes que em conjunto com o manipulador realiza a comutação de seus polos para gerar o movimento.

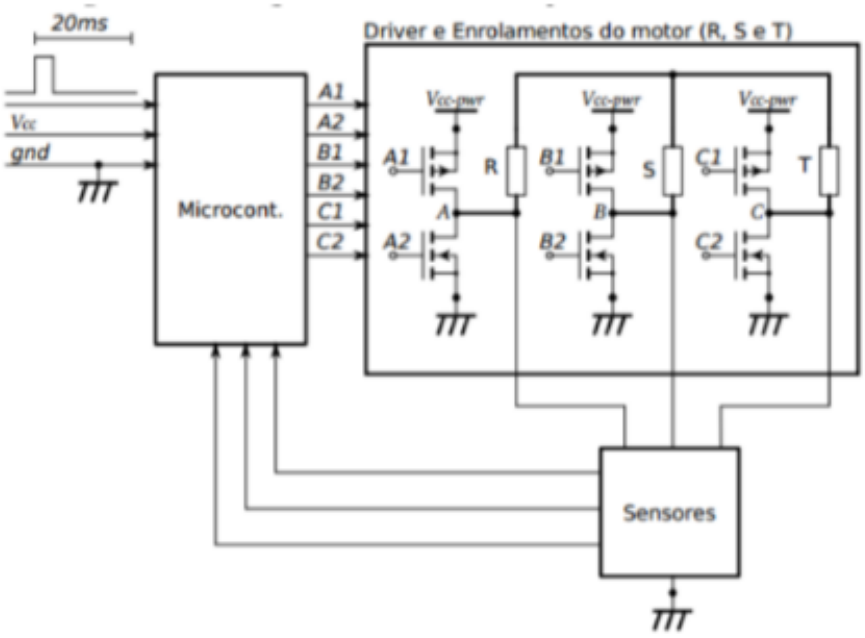
O MCCSE é o mais utilizado ao se tratar de motores de baixa potência. Isso se deve ao fato de ser eficiente, silencioso, confiável, compacto e com um baixo custo de manutenção. Além disso, permite uma redução da interferência magnética e melhor proporção de torque fornecido em relação ao tamanho do motor (OST, 2015).

Os comandos enviados para o MCCSE são comumente enviados por meio de manipuladores, que fazem a interface entre os sinais de controle e o equipamento físico. O manipulador mais comumente utilizado é o chamado CEV (traduzido do inglês Electronic Speed Control). A Figura 7 mostra um esquema simplificado do conjunto CEV - Motor.

Para realizar o controle do motor é necessário o uso de sinais analógicos. Um dos métodos para obter um sinal analógico a partir de saídas digitais é através da modulação por largura de pulso ou MLP (traduzido do inglês Pulse Width Modulation).

[1] Conhecido na literatura inglesa como Driver

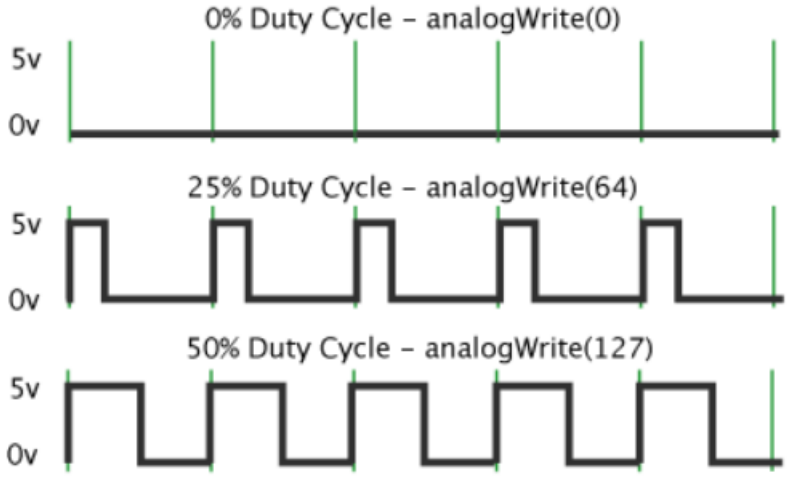
Figura 7 – Diagrama do conjunto CEV e MCCSE



Fonte: Ost (2015)

Essa modulação é realizada por uma sequência 'liga-desliga' em uma onda quadrada para obter uma tensão intermediária, como mostrado na Figura 8. O fator que determina a largura do pulso é o chamado Duty Cycle, que determina o percentual da largura onde a saída fica com o sinal ligado.

Figura 8 – Sinal MLP



Fonte: Arduino (2018)

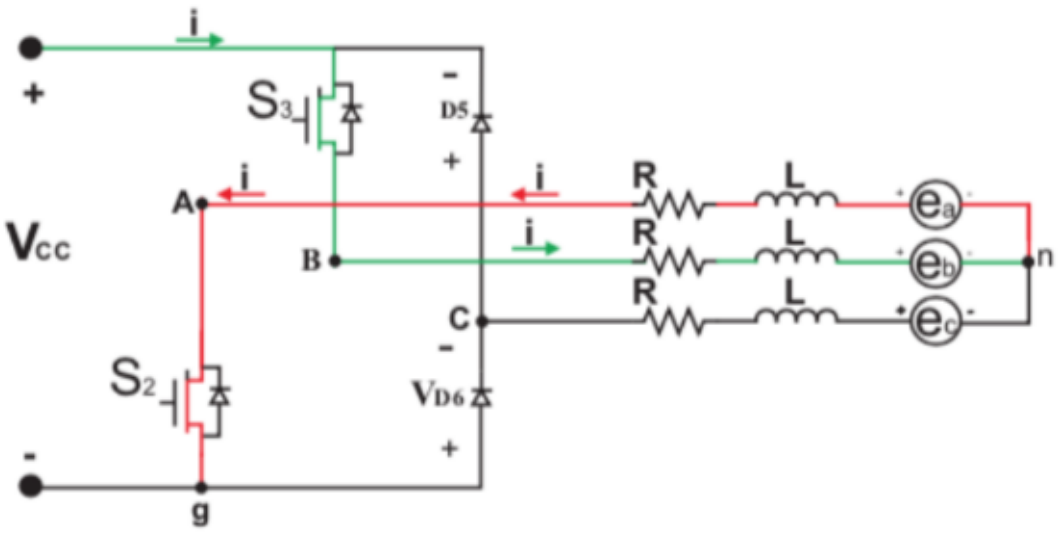
O controle de velocidade do motor com um CEV tradicional é feito mediante um sinal MLP com a frequência de 50Hz, onde o Duty Cycle determina o percentual de V_{cc} aplicado ao motor. O CEV é responsável por gerar, por meio de um inversor trifásico, os sinais para cada uma das três bobinas do motor defasadas de 120º entre si.

Os manipuladores CEV tradicionais também são equipados com uma lógica de segurança, cujo objetivo é impedir inicializações bruscas do motor. Para inicializar o motor é necessário reduzir o sinal de controle enviado para zero, onde em seguida será emitido um sinal sonoro pelo motor que indica que ele está pronto para operar.

Para determinar a função de transferência que relaciona a velocidade do motor MCCSE com o sinal MLP aplicado se pode assumir que em determinado momento o sinal MLP apenas ativa as bobinas A e B, sendo que C está desligado nesse momento. Esta situação estaria representada pelo esquema elétrico da Figura 9.

A tensão V_{BA} é a tensão aplicada ao motor MCCSE nesse momento e equivale a uma porcentagem de V_{cc} definida pela relação $\frac{Ton}{Ton+Toff}$ x 100 a qual denominaremos de u_d e e_a é a força contra eletromotriz resultante do movimento do rotor.

Figura 9 – Circuito Equivalente do MCCSE quando ativados A e B, e C desligado



Fonte: Stella (2022)

$$V_{ba} = u_d = 2Ri + 2L\frac{di}{dt} + e_b - e_a \tag{2.6}$$

$$u_d = 2Ri + 2L\frac{di}{dt} + 2e_a \tag{2.7}$$

$$u_d = Ri + L\frac{di}{dt} + K_e\omega \tag{2.8}$$

Em (2.7) a força contra eletromotriz e_a é considerada proporcional a velocidade de rotação ω e uma constante K_e.

A equação mecânica para o motor de MCCSE para um torque eletromagnético T_e, torque de carga T_L, momento de inércia J_m e coeficiente de amortecimento B_v é:

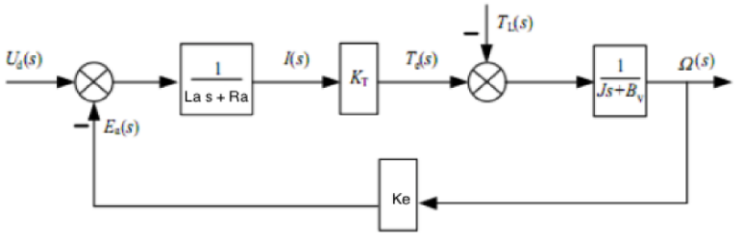
$$T_e - T_L = J_m \frac{d\omega}{dt} + B_v \omega \tag{2.9}$$

A relação entre o torque eletromagnético e a corrente é dada por:

$$T_e = K_T i \tag{2.10}$$

Rearranjando as equações e obtendo a Transformada de Laplace podemos obter o Diagrama de Blocos mostrado na Figura 10:

Figura 10 – Diagrama de blocos do MCCSE com torque de carga T_L



Fonte: Adaptado de Ost (2015)

Tomando a transformada de laplace das equações 2.8, 2.9 e 2.10 e considerando a ausência de torque de carga temos:

$$\frac{T_e(s)}{U(s)} = \frac{K_t(J_m s + B_v)}{J_m L s^2 + J_m R s + (R B_v + K_v K_t)} \tag{2.11}$$

Considerando uma relação linear entre torque e força de empuxo e substituindo em 2.11:

$$F_t = K_g T_e \tag{2.12}$$

$$\frac{F_t(s)}{U(s)} = \frac{K_g K_t(J_m s + B_v)}{J_m L s^2 + J_m R s + (R B_v + K_v K_t)} \tag{2.13}$$

A equação 2.13 também pode ser simplificada considerando que a indutância do circuito equivalente é desprezível em comparação aos outros termos, não tendo influência significativa sobre a dinâmica do sistema. Dessa forma temos:

$$\frac{F_t(s)}{U(s)} = \frac{K_g K_t (J_m s + B_v)}{J_m R s + (R B_v + K_v K_t)} \tag{2.14}$$

Outra forma simplificada de modelar o sistema é considerar que a velocidade do motor é proporcional a tensão aplicada, essa é uma hipótese razoável desde que as constantes de tempo elétricas do motor sejam insignificantes em relação às constantes de tempo envolvidas no sistema mecânico.

$$\omega = K_v u \tag{2.15}$$

$$F_t = K_g K_v u = K u \tag{2.16}$$

$$\frac{F_t(s)}{U(s)} = K \tag{2.17}$$

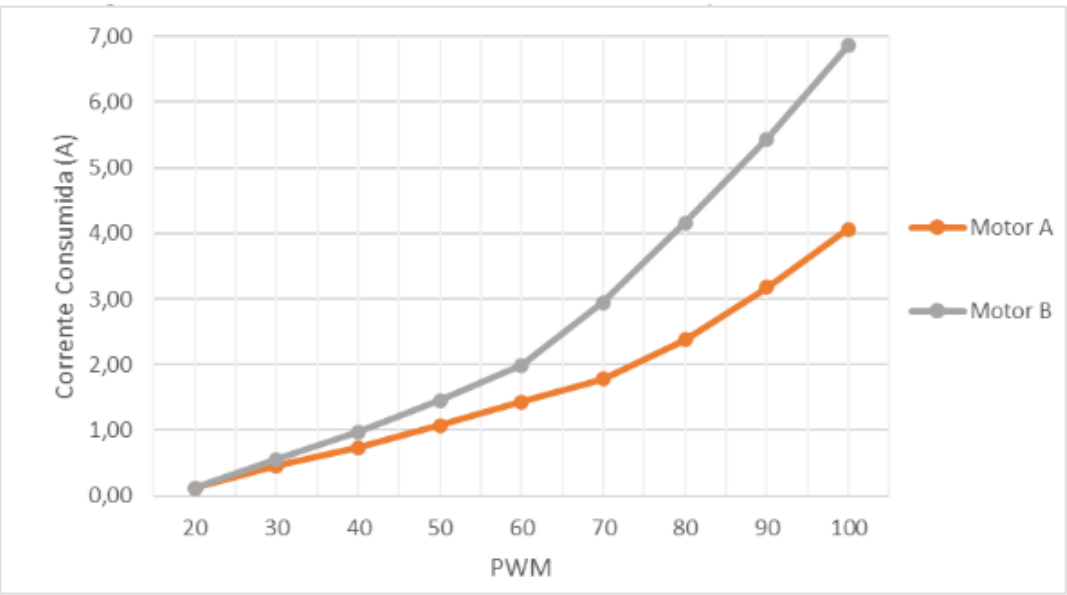
Onde u é a tensão aplicada no motor e as constantes caracterizam a relação do motor podem ser substituídas por uma constante K_v.

Em Dias (2021) dois motores utilizados em uma bancada de teste para drones apresentaram comportamentos diferentes, mesmo sendo do mesmo fabricante. A Figura 11 exemplifica o comportamento da corrente baseado na tensão enviada ao CEV. Essa variabilidade de comportamento torna necessário a análise de cada motor como um equipamento único, independente se possuem o mesmo modelo construtivo de outro motor já analisado.

Essa variação de comportamento influencia diretamente no desempenho do sistema. Como a relação entre corrente e torque é linear, para uma mesma ação de comando é possível ter pontos de equilíbrio diferentes, mesmo tendo todas as outras características do sistema iguais.

Uma maneira de identificar as características do motor é através da realização de experimentos de identificação. Para a determinar a relação empuxo-tensão de um

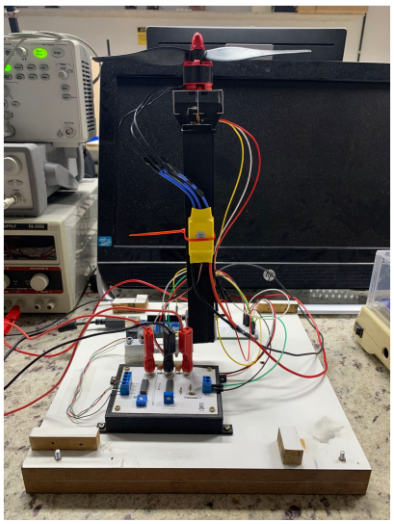
Figura 11 – Comportamento do empuxo com diferentes motores



Fonte: Dias (2021)

determinado conjunto Motor-hélice-CEV Tavares (2022) desenvolveu uma bancada de medição de tração de grupos motorpropulsores de pequeno porte (Figura 12). O objetivo deste trabalho foi identificar as relações empuxo-tensão desses conjuntos em regime permanente. Atualmente a bancada está sendo modificada para também capturar o comportamento transiente através da relação entre tensão e velocidade do motor.

Figura 12 – Bancada de tração vertical



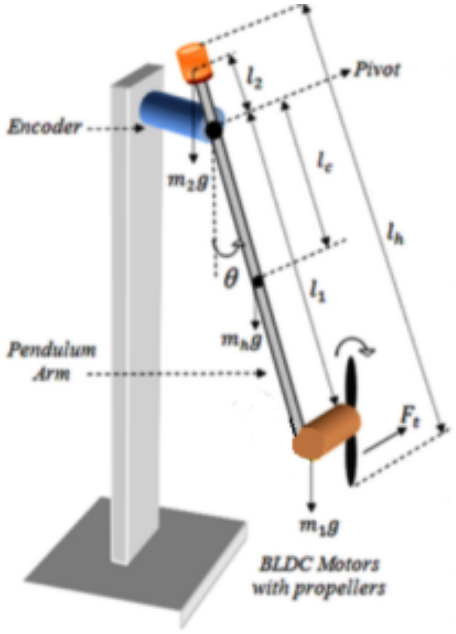
Fonte: Tavares (2022)

Esse desenvolvimento foi uma boa solução para o problema de identificação das diferenças entre conjuntos motopropulsores de diferentes características.

2.1.3 Modelagem do aeropêndulo

Nesta seção foi desenvolvido o modelo analítico do aeropêndulo e o seu comportamento em relação a uma força de acionamento F_t e obtido as respectivas funções de transferência para os modelos considerados.

Figura 13 – Diagrama de forças de um aeropêndulo



Fonte: Saleem *et al.* (2020)

O modelo matemático do sistema mostrado na Figura 13 é obtido através da segunda lei de Newton para sistemas rotativos sendo dado por:

$$J_P\ddot{\theta} = \sum T_i \tag{2.18}$$

$$J_P\ddot{\theta} = m_2gl_2sin\theta + \frac{1}{2}m_{h2}gl_2sin\theta - m_1gl_1sin\theta - \frac{1}{2}m_{h1}gl_1sin\theta - c\dot{\theta} + F_tl_1 \tag{2.19}$$

Onde J é o momento de inércia do sistema em torno do eixo de rotação; F_t é a força do empuxo gerada pelo conjunto motopropulsor; m_1, m_2 e m_h são as massas do conjunto motopropulsor, do contra peso e da haste respectivamente. Uma consideração possível no modelo é que o torque gerado pelo contrapeso é predominante em relação ao da haste, ou seja: $m_2gl_2sin\theta + \frac{1}{2}m_{h2}gl_2sin\theta \approx m_2gl_2sin\theta$. Essa consideração é razoável para hastes finas e será o modelo considerado neste trabalho:

$$J_P\ddot{\theta} = m_2 g l_2 sin\theta - m_1 g l_1 sin\theta - \frac{1}{2} m_{h1} g l_1 sin\theta - c\dot{\theta} + F_t l_1 \tag{2.20}$$

A equação 2.20 é a equação não-linear do movimento de um aeropêndulo submetido a uma força F_t. Sua resposta transitória pode ser obtida através da transformada de laplace da equação 2.20 e considerando $B = m_1 g l_1 + \frac{1}{2} m_h g l_h - m_2 g l_2$ e com a linearização do termo não-linear para pequenos ângulos $sen\theta \approx \theta$ fornece a seguinte equação:

$$\frac{\Theta(s)}{F_t(s)} = \frac{l_1}{s^2 + \frac{c}{J_P} s + \frac{B}{J_P}} \tag{2.21}$$

Um modelo mais simplificado pode ser obtido através da substituição de 2.17 em 2.24, obtendo uma função de transferência de segunda ordem:

$$\frac{\Theta(s)}{U(s)} = \frac{K l_1}{s^2 + \frac{c}{J_P} s + \frac{B}{J_P}} \tag{2.22}$$

Considerando o modelo de primeira ordem da equação 2.14 temos um sistema equivalente de terceira ordem:

$$\frac{\Theta(s)}{U(s)} = \frac{l_1 \left[K_e K_t (J_m s + Bv)\right]}{\left[J_m R s + (R B_v + K_v K_t)\right] (s^2 + \frac{c}{J_P} s + \frac{B}{J_P})} \tag{2.23}$$

Considerando o modelo desenvolvido através da explicitação do modelo do MCCSE, podemos definir um modelo de quarta ordem a partir da substituição da equação 2.13 em 2.21:

$$\frac{\Theta(s)}{U(s)} = \frac{l_1 \left[K_e K_t (J_m s + Bv)\right]}{\left[J_m L s^2 + J_m R s + (R B_v + K_v K_t)\right] (s^2 + \frac{c}{J_P} s + \frac{B}{J_P})} \tag{2.24}$$

2.2 A plataforma Arduino

A Arduino é uma plataforma eletrônica de código aberto de baixo custo com foco em facilidade de integração entre hardware e software para desenvolvimento de projetos

eletrônicos (ARDUINO, 2018). Existem diversas placas com capacidades de processamento diferentes, entre elas a Arduino Uno (Figura 32), sendo uma das mais comuns para prototipagem de sistemas eletrônicos.

Figura 14 – Placa Arduino Uno

Especificações do Arduino Uno	
Tensão de operação	5V
Corrente máxima por pino	40mA
Frequência de operação	16MHz
Entradas/Saídas digitais	14
Entradas analógicas	6
Suporte a comunicação serial	UART, SPI,

Fonte: Arduino (2018)

A Arduino Uno é uma placa de desenvolvimento, baseada no microcontrolador ATmega328P, inicialmente projetado para permitir que usuários de diversas áreas de conhecimento possam implementar projetos eletrônicos de maneira simplificada.

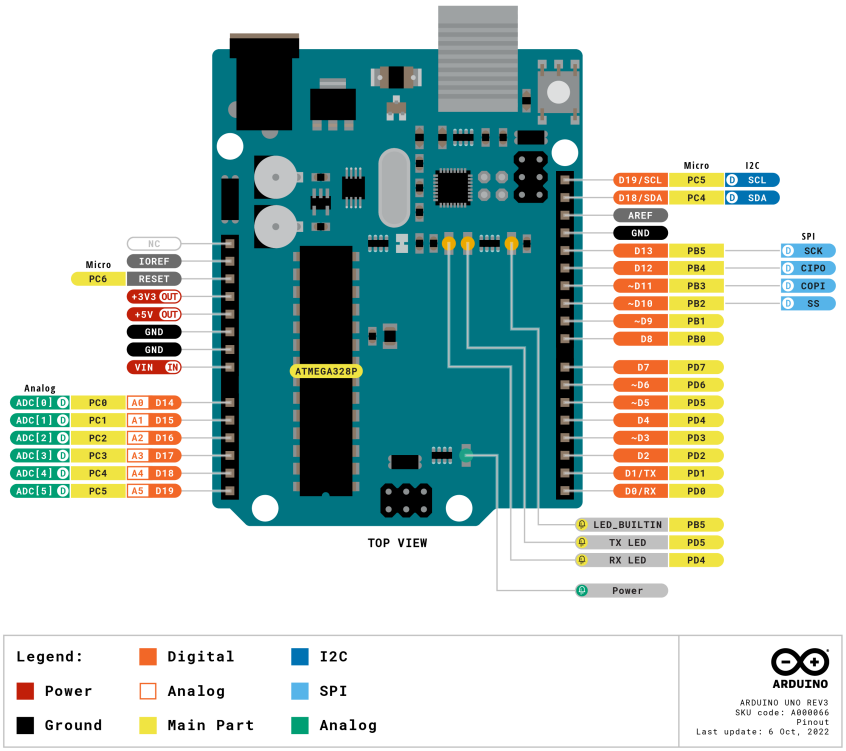
A Arduino possui um ambiente de desenvolvimento que permite a programação das entradas e saídas de sinal utilizando uma variante da linguagem C++, exibido na Figura 15.

Figura 15 – Interface de desenvolvimento da Arduino

Fonte: Arduino (2018)

A Arduino conta com 14 entradas/saídas digitais, onde 6 dessas podem ser utilizadas com modulação por largura de pulso e 6 entradas analógicas (Figura 16). Além disso, é capaz de comunicação serial e interrupção. A faixa de tensão de operação da Arduino para alimentação de componentes é de 0 a 5V, sendo suficiente para o controle da maioria dos componentes utilizados para projetos eletrônicos.

Figura 16 – Entradas e saídas da Arduino Uno



Fonte: Arduino (2018)

2.3 O ensino do controle de sistemas dinâmicos

O ensino do controle de sistemas dinâmicos visa ensinar a modelagem de sistemas, análise da estabilidade, especificações da resposta em regime transiente e permanente e técnicas de desenvolvimento de controladores em malha fechada. As principais bibliografias utilizadas têm em seus principais capítulos esse conteúdo, são estas Ogata (2011), Dorf e Bishop (2008), Nise (2000) e Kuo e Golnaraghi (2002). Nessas bibliografias, seu conteúdo aborda todos esses tópicos de maneira semelhante. A seguir é apresentado um breve resumo dos assuntos abordados nesta disciplina.

2.3.1 Modelagem de sistemas dinâmicos

O processo de modelagem ocorre através das equações diferenciais que regem seus comportamentos para identificar uma representação do sistema. As representações mais comuns são a função de transferência, diagrama de blocos e representação

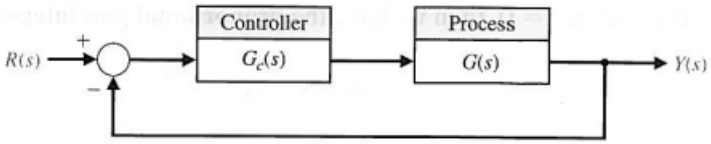
em espaço de estados. Os principais sistemas considerados no ensino do controle são os sistemas mecânicos, elétricos, de nível de líquidos, térmicos, pneumáticos e hidráulicos.

Um meio de classificar sistemas é em relação à sua ordem, determinada pelo número de polos presentes no sistema. Os sistemas estudados são os de primeira ordem, segunda ordem e ordem superior.

Em geral, os sistemas podem ser classificados como sistemas de malha aberta e malha fechada. Um sistema de malha aberta é aquele onde é enviado um sinal e não há comparação e realimentação do sistema, como, por exemplo, um sistema baseado em um temporizador que executa uma ação após um certo tempo. Já um sistema em malha fechada faz o uso da realimentação do sinal, onde normalmente na presença do controlador, é gerado um sinal de atuação baseado na diferença entre o valor de referência e o valor atual da variável de processo.

O diagrama de blocos de um sistema de malha fechada é mostrado na Figura 17, que exemplifica um valor de referência $R(s)$ subtraído do valor atual $Y(s)$ para determinação do erro $E(s)$. O controlador atua sobre o erro do sistema e envia um sinal para o atuador, que age sobre o processo de modo a minimizar o erro.

Figura 17 – Diagrama de blocos de sistema em malha fechada



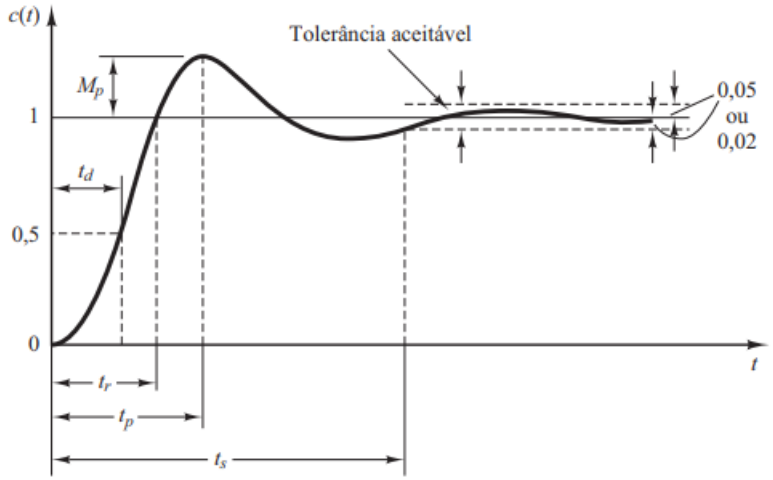
Fonte: Dorf e Bishop (2008)

2.3.2 Análise da resposta temporal

A análise temporal de sistemas é divida em duas partes, o regime transitório e o regime permanente, que começa uma vez que o sistema atinge o ponto de equilíbrio.

As especificações comuns para a resposta transiente de um sistema são diversas, entre as principais temos o sobressinal M_p, tempo de subida t_r, tempo de acomodação t_s. Estes podem ser facilmente identificados na resposta típica de um sistema a uma entrada degrau unitário mostrado na Figura 18. Em sistemas sobreamortecidos, o tempo de subida é tipicamente o tempo necessário para o sistema partir de 10% a 90% de seu valor em estado estacionário (OGATA, 2011).

Figura 18 – Curva de resposta subamortecida a uma entrada degrau unitário



Fonte: Ogata (2011)

Entre as especificações de regime permanente, é possível caracterizar os sistemas em alguns tipos que determinam sua capacidade de seguir determinadas entradas, como, por exemplo, degrau, rampa e parábola. Essa classificação é um critério razoável, pois as entradas reais com frequência podem ser consideradas combinações destas (OGATA, 2011). Para um sistema de controle com realimentação unitária temos:

$$G(s) = \frac{K(T_a s + 1)(T_b s + 1)...(T_m s + 1)}{s^N (T_1 s + 1)(T_2 s + 1)...(T_p s + 1)} \tag{2.25}$$

Onde N indica o tipo do sistema e a quantidade de polos na origem. O tipo é uma classificação diferente da ordem do sistema, conforme N aumenta a precisão do sistema aumenta, porém, a estabilidade é prejudicada. A Figura 19 exemplifica a resposta de sistemas do tipo 0, tipo 1 e tipo 2 para as entradas selecionadas.

Figura 19 – Erro em estado estacionário para os tipos de sistema

	Entrada em degrau $r(t) = 1$	Entrada em rampa $r(t) = t$	Entrada em aceleração $r(t) = \frac{1}{2}t^2$
Sistema do tipo 0	$\frac{1}{1+K}$	∞	∞
Sistema do tipo 1	0	$\frac{1}{K}$	∞
Sistema do tipo 2	0	0	$\frac{1}{K}$

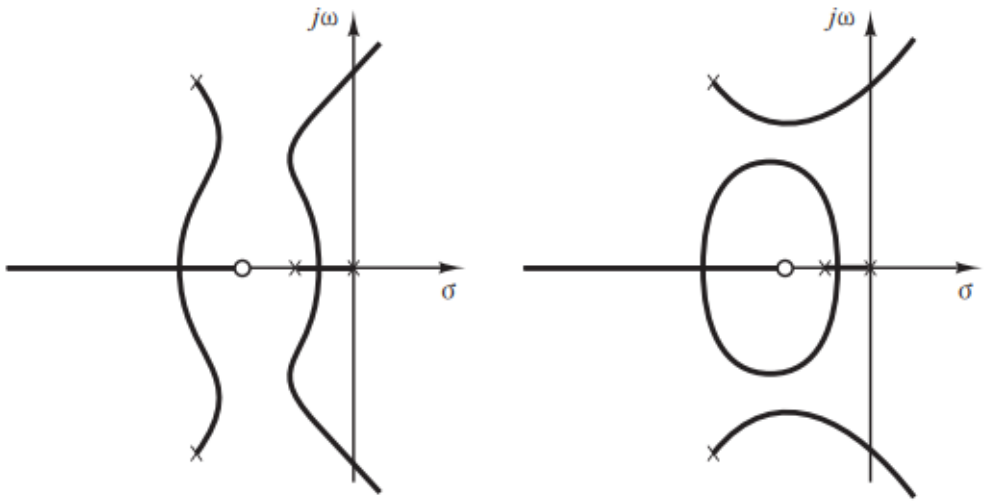
Fonte: Ogata (2011)

2.3.3 Análise da estabilidade

A análise da estabilidade dos sistemas é realizada através do estudo do diagrama do lugar das raízes e através da análise da resposta em frequência, normalmente por meio do uso de diagramas de Bode.

A análise do lugar das raízes permite analisar o comportamento dos polos de um sistema com a variação do ganho em um sistema de malha fechada com ganho K, permitindo uma análise do comportamento usando a função de malha aberta. A Figura 20 mostra o comportamento de alguns sistemas onde é possível ver a variação da posição dos polos do sistema com a alteração do ganho, observando ser possível tornar o sistema instável, como na presença de polos no lado direito do plano S.

Figura 20 – Gráficos do lugar das raízes

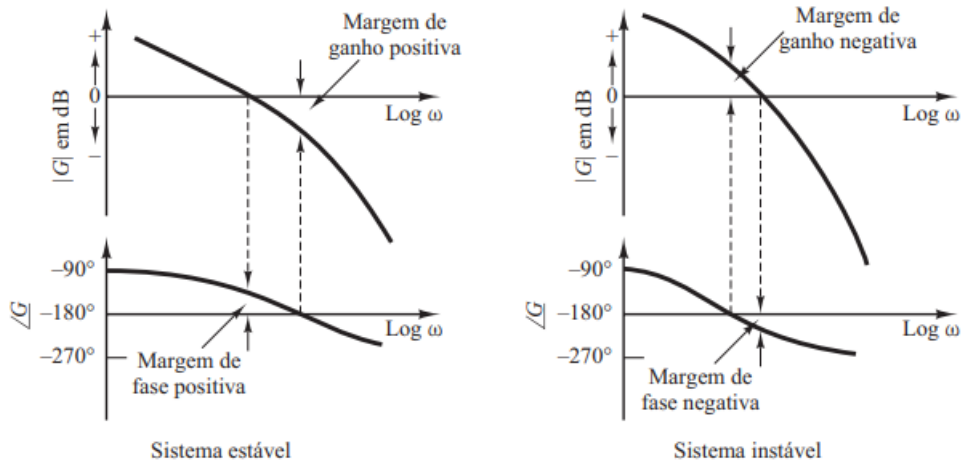


Fonte: Ogata (2011)

Os diagramas de Bode também são utilizados para a análise da estabilidade relativa, que indica o quão próximo o sistema está em relação ao limiar da instabilidade onde o sistema apresentaria uma resposta oscilatória sustentada. Esta estabilidade é determinada através da margem de ganho e margem de fase, mostrada na Figura 21.

Para sistemas de primeira e segunda ordem a margem de ganho é infinita e por esse motivo, teoricamente, não podem ser instáveis. No entanto, sistemas em que são realizadas aproximações na modelagem, onde pequenas constantes de tempo são desconsideradas, não são verdadeiramente sistemas de primeira ou segunda ordem e poderão, sim, se tornar instáveis (OGATA, 2011).

Figura 21 – Diagrama de Bode e as margens de ganho e fase



Fonte: Ogata (2011)

2.3.4 Dispositivos de controle

Os dispositivos de controle são responsáveis por executar uma lei de controle que, utilizando uma ou várias entradas gera uma saída que vai governar a atuação da variável manipulada no sistema. Apesar de existirem muitas técnicas de controle, os alunos estudam principalmente o uso de controladores de propósito geral,sendo estes os controladores proporcional, integral e derivativos (PID) e compensadores de atraso-avanço.

Embora o controlador PID seja simples, ele é um dos mais utilizados para processos industriais sendo usado em cerca de 96% das aplicações de controle (RUBAAI *et al.*, 2008). Ele recebe esse nome devido a suas constantes proporcional, integrativa e derivativa. A função de transferência do controlador é (2.26) e sua resposta temporal é mostrada em (2.27) (DORF; BISHOP, 2008).

$$G(s) = K_P + \frac{K_I}{s} + K_D s \tag{2.26}$$

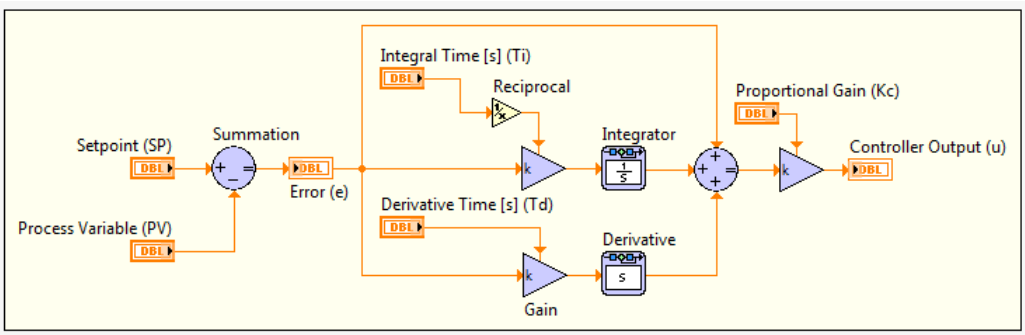
$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt} \tag{2.27}$$

Também é comum a representação dos parâmetros dos controladores PID em função de uma constante proporcional e constantes de tempo T_i e T_d, referentes ao termo integrativo e derivativo respectivamente. Dessa forma a resposta ao tempo tem o formato:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{2.28}$$

O diagrama de blocos de um controlador PID está representado na Figura 22:

Figura 22 – Diagrama de implementação de um controlador PID



Fonte: National Instruments (2024b)

Outro dispositivo comum para aplicação do controle de sistemas é o através do uso de compensadores, que visa alterar o local das raízes pela adição de polos e zeros para obter o comportamento desejado. O compensador tem o formato:

$$G_c = K\frac{T_1 s + 1}{T_2 s + 1} \tag{2.29}$$

Onde o compensador é considerado um compensador de atraso ou avanço dependendo dos valores de T_1 e T_2.

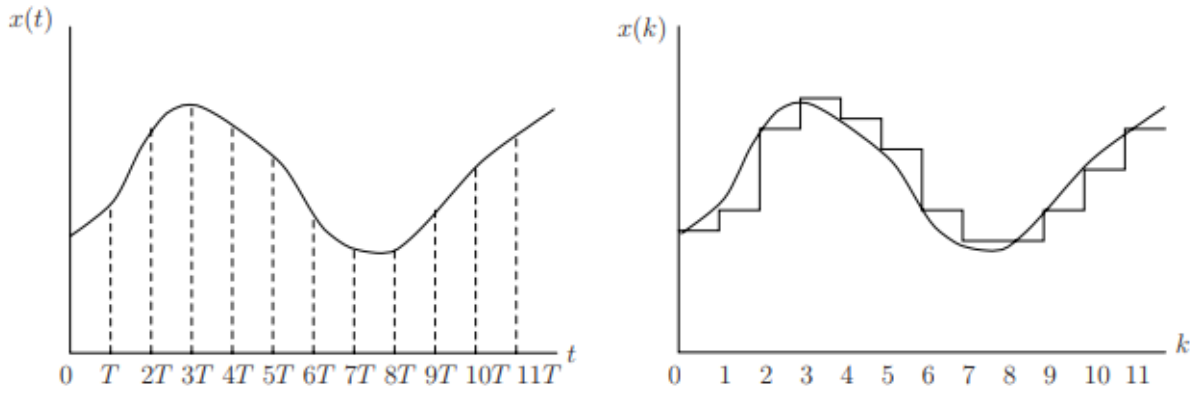
2.4 O uso de computadores no controle de sistemas digitais

Um grande desenvolvimento para o controle de sistemas dinâmicos foi a utilização de sistemas digitais para realizar o controle. Com o avanço da tecnologia e melhoria da capacidade de processamento de computação, se tornou possível realizar a implementação desses algorítimos através de computadores em tempo real. Hoje o controlador digital está presente em praticamente todos os processos, devido a sua versatilidade e facilidade de implementação em comparação ao controle analógico.

Uma das grandes diferenças de um sistema de controle digital é que o fluxo de sinais do sistema não é apenas através das leis que modelam as relações entre as grandezas físicas, mas também usa sensores que transformam a informação física em sinais digitais. Estes sinais são então processados numericamente e enviados a um atuador que interage fisicamente com o processo controlado.

Em um sistema digital, o processamento é realizado em intervalos de tempo conforme o processador utilizado. Por esse motivo é necessário adaptar os métodos de controle utilizados na teoria do contínuo para um sistema com tempo discreto, um exemplo da discretização de um sinal está exposto na Figura 23. Para realizar essa conversão, os sinais analógicos são então quantizados e discretizados, gerando erros e imprecisões.

Figura 23 – Discretização de sinal contínuo



Fonte: Cassandras e Lafortune (2010)

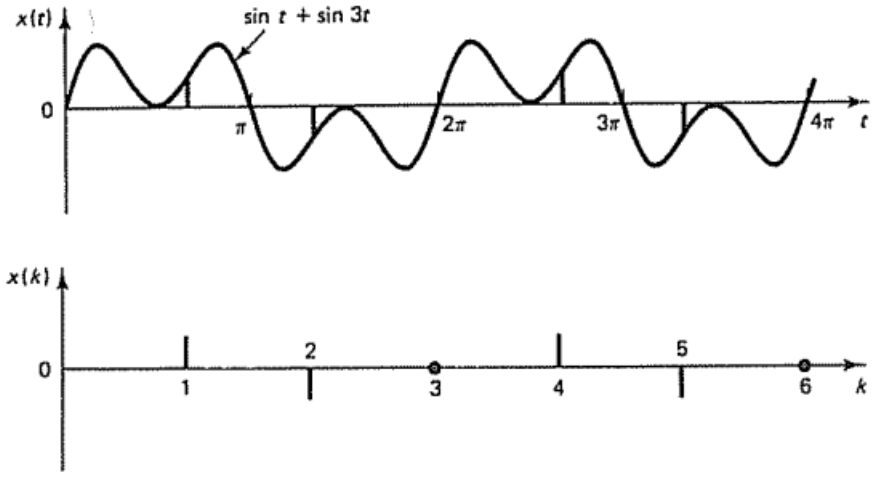
2.4.1 Seleção do período de amostragem

O processo de conversão de sinais contínuos é realizado através da amostragem do sinal em intervalos de tempo regulares, denominado período de amostragem T_a. O período de amostragem é um fator muito importante que possui grande influência na estabilidade do sistema, e está sempre associado a capacidade de processamento do sistema computacional utilizado e o consumo de energia do sistema. A escolha correta desse período depende da característica de cada sistema e sua velocidade.

Conforme o Teorema de Nyquist-Shannon (SHANNON, 1949) a frequência de amostragem ω_s deve ser pelo menos duas vezes a maior frequência ω_1 presente no sistema. Embora o requisito mínimo seja que $\omega_s > 2\omega_1$, para requisitos práticos em um sistema de malha fechada é necessário amostrar em uma frequência muito superior, normalmente entre 10 a 20 vezes ω_s (OGATA, 1987).

Um exemplo é mostrado na Figura 24 que ilustra um sinal senoidal sendo amostrado em uma frequência de $\omega_s = 3$, a qual é inferior a duas vezes a maior frequência desse sinal, levando a uma interpretação incorreta sobre o real sinal, esse efeito é conhecido como 'aliasing'.

Figura 24 – Fenômeno de aliasing em sinais



Fonte: Ogata (1987)

A amostragem do sistema atua como um atraso de transporte, não causando diretamente uma mudança na amplitude do sinal, porém diminuindo a margem de fase e tornando consequentemente o sistema mais próximo de uma região instável e até o levando a instabilidade se o período for muito grande.

Embora o aumento da taxa de amostragem cause um melhor desempenho, isso também acarreta um maior custo para implementação do controlador, devido a maior necessidade de processamento e energia, além do maior uso do atuador, diminuindo o tempo médio entre falhas por tempo de uso. Por esse motivo é preciso selecionar a maior taxa possível que garanta as especificações esperadas.

Existem diversas heurísticas utilizadas para a seleção da frequência de amostragem. É comum utilizar a frequência da largura de banda ω_B no lugar da maior frequência do sistema ω_s.

Uma heurística definida em Seborg *et al.* (2011) sugere como diretriz a seguinte relação:

$$6\omega_B < \omega_s < 25\omega_B \tag{2.30}$$

Franklin *et al.* (2022) sugere uma abordagem semelhante, porém com uma frequência maior, visando suavizar a resposta e obter um melhor controle, definida pela seguinte relação;

$$25\omega_B < \omega_s < 40\omega_B \tag{2.31}$$

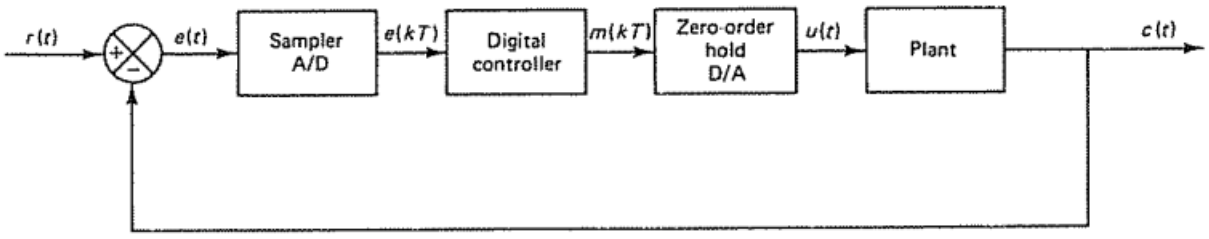
Outro possível método de seleção do período de amostragem é o indicado em Åström e Wittenmark (2011), onde é considerado a resposta no tempo do sistema, considerando a constante de tempo τ_d ou o tempo de acomodação t_s, e é determinado pelas relações:

$$0,01\tau_d < T_a < 0,05\tau_d \tag{2.32}$$

$$\frac{t_s}{15} < T_a < \frac{t_s}{6} \tag{2.33}$$

O diagrama de blocos do controle de um sistema discreto é exibido na Figura 25. E consiste de um conversor Analógico Digital, Controlador digital, conversor digital analógico ou atuador que atua sobre a planta ou sistema.

Figura 25 – Diagrama de blocos de um sistema com controle digital



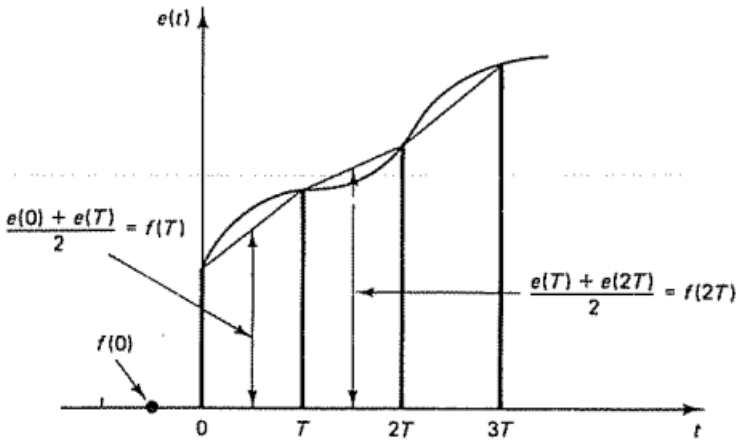
Fonte: Ogata (1987)

2.4.2 O controlador PID digital

O controlador PID digital, diferentemente do contínuo, faz o uso de equações de diferenças obtidas das equações diferenciais derivadas dos modelos físicos como visto na equação 2.28.

A discretização do valor da integral pode ser aproximada de diversas maneiras, uma das mais precisas é a aproximação trapezoidal, que como o nome sugere calcula a área aproximada através de um trapézio, exibida na Figura 26. Da Figura também é perceptível a aproximação para a derivada, $de(k)/dt \approx [e(k) - e(k-1)]/T$

Figura 26 – Discretização de uma integral contínua



Fonte: Ogata (1987)

Substituindo esses fatores na equação 2.28 e considerando tempos discretos onde $t = kT$ temos:

$$u(kT) = K\Big\{e(kT) + \tfrac{T}{T_i}\left[\tfrac{e(0)+e(T)}{2}... + \tfrac{e((k-1)T)+e(kT)}{2}\right] + T_d\tfrac{e(kT)-e((k-1)T)}{T}\Big\} \tag{2.34}$$

$$u(kT) = K\left\{e(kT) + \frac{T}{T_i}\sum_{h=1}^{k}\frac{e((h-1)T)+e(hT)}{2} + \frac{T_d}{T}[e(kT) - e((k-1)T)]\right\} \tag{2.35}$$

Aplicando a transformada Z temos:

$$U(z) = K\left[1 + \frac{T}{2T_i}\frac{1+z^{-1}}{1-z^{-1}} + \frac{T_d}{T}(1-z^{-1})\right]E(z) \tag{2.36}$$

A equação pode ser reescrita como:

$$U(z) = K\left[1 - \frac{T}{2T_i} + \frac{T}{T_i}\frac{1}{1-z^{-1}} + \frac{T_d}{T}(1-z^{-1})\right]E(z) \tag{2.37}$$

$$U(z) = \left[K_P + \frac{K_i}{1-z^{-1}} + K_D(1-z^{-1})\right]E(z) \tag{2.38}$$

Onde:

$$K_P = K - \frac{KT}{2T_i}$$
$$K_i = \frac{KT}{T_i}$$
$$K_d = \frac{KT_d}{T}$$

Na implementação do controlador digital existem alguns problemas típicos. Um desses problemas é quando a ação integral atinge um valor muito alto, esse fenômeno é conhecido como windup. Isso causa uma sobreposição da constante integrativa sobre as outras, gerando comportamentos indesejados do controlador. Uma solução para este problema é impedir a atualização da ação integral quando o atuador estiver saturado. Ou seja:

$$u(k) = \begin{cases} u_{max}\ para\ u(k) \geq u_{max} \\ u_{min}\ para\ u(k) \leq u_{min} \end{cases} \tag{2.39}$$

Além disso, um ponto importante é o efeito da constante da ação integral na troca do modo de operação de automático para manual, presente no modo de operação desejado neste trabalho. Para isso é necessário limitar o sinal de saída conforme o sinal acumulado até o momento. Dessa maneira temos:

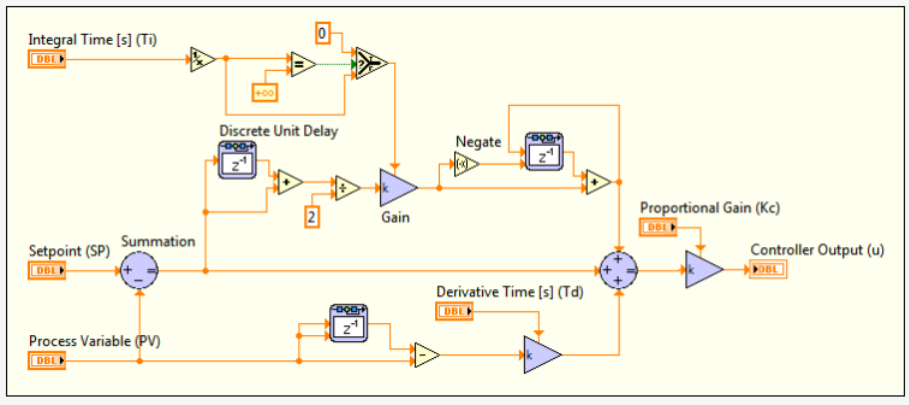
$$|u_p(k) + u_I(k)| > |limite| -> u_I(k) = limite - u_p(k) \tag{2.40}$$

Outro problema típico é a amplificação do ruído da ação derivativa atuando sobre o erro, que pode surgir devido a alterações de setpoint ou por ruídos de alta frequência. Para solucionar esse problema é utilizado a derivada da saída no lugar da derivada do erro. Essa implementação é representada por:

$$u_d(k) = -K\frac{T_d}{T}(PV(k) - PV(k-1)) \tag{2.41}$$

A Figura 27 mostra o diagrama de uma implementação considerando esses métodos de adaptação para um controlador digital.

Figura 27 – Diagrama de implementação de um controlador PID Digital



Fonte: National Instruments (2024a)

2.5 Trabalhos similares

Existem muitos trabalhos na área de controle visando melhorar as técnicas e métodos, em particular os de identificação e controle de sistemas não lineares como o aeropêndulo. Nesta seção serão discutidos alguns destes e quais as contribuições e similaridades com o presente trabalho.

Em Enikov e Campa (2012b) foi desenvolvido um aeropêndulo simplificado, fazendo o uso de fibra de carbono para a haste, com o uso de um potenciômetro para mensuração do ângulo da haste e um motor CC. A plataforma fez uso de comunicação USB com o MATLAB/Simulink Realtime, onde era feito a leitura do ângulo e envio do sinal de comando. Nessa plataforma era possível desenvolver e testar controladores contínuos como o PID e compensadores de atraso e avanço. Foi realizado a análise de estabilidade e verificado os efeitos da linearização de sistemas não lineares e o seu impacto no funcionamento do controle. Essa análise é pertinente, pois a simplificação de sistemas de ordem superior é uma prática comum, e por vezes, seu impacto é desconsiderado. Por fim, realizado a análise do uso desse sistema para aprendizagem de estudantes de engenharia mecânica e foi conduzida uma pesquisa para determinar a qualidade do aprendizado dos alunos.

Em Habib *et al.* (2017) foi desenvolvido um aeropêndulo semelhante ao Enikov e Campa (2012b). Em seu trabalho o foco foi determinar o desempenho de um controle PD (Proporcional e derivativo) que utiliza a técnica agir e esperar (AeE). A estratégia AeE tem o princípio de aguardar o efeito da força de controle em um sistema em tempo discreto antes de enviar um novo sinal de atuação, em geral, esse período de espera é um múltiplo do período de amostragem. Esse comportamento simula atrasos ocorridos em sistemas reais, advindos de atrasos de comunicação, controle remoto, alta

demanda computacional ou atraso humano. Foi analisado o impacto dessa técnica na estabilidade e diminuição de erros em regime permanente, além disso, foi comparado seu desempenho com controladores tradicionais.

Em Gonçalves (2018) foi construído um protótipo de um aeropêndulo para uso como uma bancada didática com foco no controle de sistemas contínuos. Foi utilizado o controle analógico desenvolvido a partir das relações eletrônicas e operações com amplificadores operacionais. Seu objetivo era permitir uma experiência de controle com o uso de controladores contínuos, mas permitindo a versatilidade utilizando potenciômetros digitais e a conexão com o MATLAB Realtime para os experimentos. A integração deste sistema com a plataforma de experimentação e controle dos ganhos digitais e sinais foi feito utilizando uma placa Arduino. Essa bancada foi utilizada como ponto de partida para o presente trabalho sendo utilizada para o ensino no laboratório até o presente momento.

Em Neto *et al.* (2023) foi construído uma bancada para o controle do aeropêndulo, onde foi desenvolvido todo o circuito de condicionamento de sinal, semelhante ao trabalho realizado por Gonçalves (2018), porém com o uso de potenciômetros analógicos. Além disso, a placa desenvolvida permite tanto o uso de controle digital como controle analógico. Em seu trabalho mostra todos os pontos essências da modelagem de sistema, como linearização, atraso e discretização de sistemas. Todos esses pontos são pertinentes para o presente trabalho por tratar dos efeitos das simplificações de modelagem e a influência do período de amostragem no controle. Por fim faz uma análise da bancada como uma ferramenta de aprendizado e seu desempenho em melhorar o ensino da teoria do controle de sistemas dinâmicos.

3 MATERIAIS E MÉTODOS

Neste capítulo é descrito a elaboração dos projetos elétrico e mecânico e as técnicas de fabricação utilizadas. Também é mostrado as funcionalidades da interface gráfica desenvolvida para a operação da bancada e o seu funcionamento básico.

3.1 Seleção dos Principais Componentes Eletromecânicos

Os componentes eletromecânicos foram selecionados observando o baixo custo, a disponibilidade de aquisição do laboratório e sua robustez. Sendo muitos destes já presentes no laboratório ou adquiridos com alguma verba fornecida pela universidade ou por recursos próprios.Além disso, esses componentes também já são familiares pois estão presentes em práticas do laboratório no decorrer do curso.

Para satisfazer os requisitos elétricos do sistema ele deve ser capaz de atender as seguintes especificações:

1. Utilizar atuador robusto de baixo custo na geração de forca de empuxo;
2. Diminuir ruído de medição do ângulo em relação à bancada anterior (GONÇALVES, 2018);
3. Permitir a escolha de funcionamento entre malha aberta e fechada;
4. Inserir proteção para forçar o desligamento ao atingir limite mecânico

Para o acionamento foi decidido utilizar um MCCSE e CEV, pois apresenta todos os benefícios citados anteriormente em relação a custo e confiabilidade. Além disso esse conjunto é favorecido por apresentar desempenho satisfatório em outros trabalhos realizados no laboratório, como, por exemplo, o de Tavares (2022), além de estar prontamente disponível para uso no laboratório. O motor e CEV são mostrados nas Figuras 28 e 29 respectivamente.

Para a escolha do sensor, foi considerado a substituição do potenciômetro, pois a bancada atual utiliza um potenciômetro para medição, o que gera ruídos é frágil a torques externos. Dessa maneira se optou pela utilização de um encoder capaz de suportar esforços e possuir melhor qualidade do sinal. A Em Cardoso *et al.* (2022) foi utilizado um encoder de efeito hall que apresentou um desempenho satisfatório e boa resistência mecânica. Por esse motivo este encoder que é mais resistente a torques

Figura 28 – Motor CC sem escovas A2212

Especificações do Motor	
Tensão de operação	2 a 12V
Potência máxima	109W
Velocidade nominal	930 RPM
Corrente máxima	9,8A

Fonte: AliExpress (2024b)

Figura 29 – Driver CEV HW30A

Especificações do Manipulador	
Tensão de operação	7.4 a 14.8V
Frequência PWM	50Hz
Velocidade máxima	210.000 RPM
Corrente máxima	30A

Fonte: MakerHero (2024)

externos e com um sinal com menos ruídos foi selecionado. O sensor é mostrado na Figura 30.

Figura 30 – Encoder P3022-V1-CW360

Especificações do Encoder	
Tensão de operação	5V
Tensão de saída	0 a 5V
Resolução	0.088º
Carga Radial Suportada	10N
Carga Axial Suportada	5N

Fonte: AliExpress (2024a)

Para o atendimento do terceiro requisito, foi incluída uma chave alavanca para definição do modo de operação entre malha aberta e fechada. A alavanca é alimentada pela própria Arduino e possui dois estados, determinando se está no modo de malha aberta ou fechada. Como complemento, foi utilizado um potenciômetro para que operador altere o valor de maneira dinâmica.

O último componente eletrônico selecionado foi um sensor de impacto que estava

disponível no laboratório para interrupção do funcionamento do sistema caso ele atinja um deslocamento muito elevado, exibido na Figura 31

Figura 31 – Sensor de impacto YL-99

Especificações Sensor	
Tensão de operação	3 a 12V
Tensão de saída	0 ou 5V

Fonte: ielectrony (2024)

A Arduino será utilizada para a integração do sistema, devido a facilidade de integração dos componentes selecionados e programação, além da compatibilidade de tensão entre eles.

Figura 32 – Placa Arduino Uno

Especificações do Arduino Uno	
Tensão de operação	5V
Corrente máxima por pino	40mA
Frequência de operação	16MHz
Entradas/Saídas digitais	14
Entradas analógicas	6
Suporte a comunicação serial	UART, SPI,

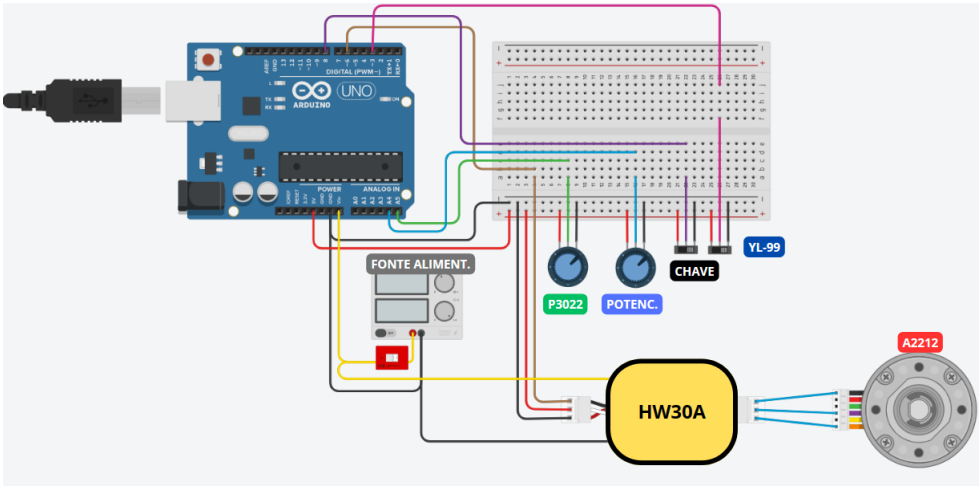
Fonte: Arduino (2018)

3.2 Projeto do circuito elétrico

Para elaboração do projeto elétrico da bancada foi utilizado a plataforma Arduino pela simplicidade, custo e facilidade de conexão com os componentes selecionados. A Figura 33 mostra a arquitetura desenvolvida para compor as funcionalidades do aeropendulo.

As entradas e saídas digitais e os pinos utilizados para comunicação com a Arduino estão exibidos na Tabela 1.

Figura 33 – Diagrama das conexões elétricas



Fonte: Próprio Autor

Tabela 1 – Tabela com entradas e saídas utilizadas

Componente	Tipo	Pino
Encoder P3022	Entrada analógica	A4
Potênciometro	Entrada analógica	A5
Fim de curso YL-99	Entrada digital	3
ESC HW30A	Saída analógica	6
Chave	Entrada digital	8

Fonte: Próprio autor.

3.2.1 Testes de integração da arquitetura elétrica

Para verificar o funcionamento adequado de todos os componentes foram utilizadas simples rotinas para o teste de cada um dos componentes, utilizando seções do programa para testar diferentes funcionalidades.

Motor Elétrico Sem Escovas e CEV

Para o teste do motor sem escovas foi utilizado o potenciômetro para variar o valor enviado ao controle do motor dinamicamente. Foi utilizado a biblioteca Servo.h para enviar o sinal na frequência adequada. O sinal deve ser mapeado da faixa de valores de leitura da Arduino de 0 a 1023 para a faixa de envio para a biblioteca de servomotores de 0 a 180.

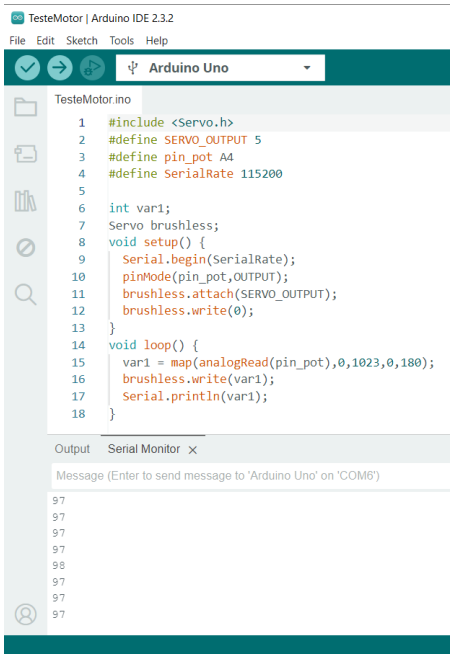
Nos testes foi necessário sempre ao inicializar o sistema, retornar o potenciômetro para enviar um sinal nulo para inicialização do motor para cumprir com o requerimento do CEV, como mencionado anteriormente.

```
#include <Servo.h>
```

```
#define SERVO_OUTPUT 5
#define pin_pot A4
#define SerialRate 115200
int var1;
Servo brushless;
void setup() {
    Serial.begin(SerialRate);
    pinMode(pin_pot,OUTPUT);
    brushless.attach(SERVO_OUTPUT);
    brushless.write(0);
}
void loop() {
    var1 = map(analogRead(pin_pot),0,1023,0,180);
    brushless.write(var1);
    Serial.println(var1);
}
```

Foi verificado se o acionamento do motor estava conforme a intensidade esperada na Figura 34.

Figura 34 – Teste de verificação do motor



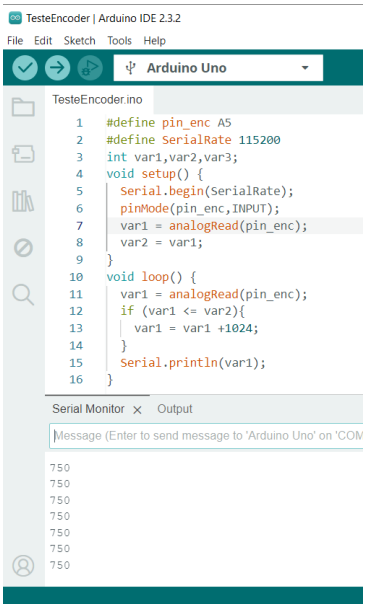
Fonte: Próprio Autor

Encoder

Para a verificação do funcionamento do encoder foi utilizado a rotina de teste abaixo. A Figura 35 mostra a execução do teste, que foi comparado com o resultado esperado fisicamente.

```
#define pin_enc A5
#define SerialRate 115200
int var1,var2,var3;
void setup() {
    Serial.begin(SerialRate);
    pinMode(pin_enc,INPUT);
    var1 = analogRead(pin_enc);
    var2 = var1;
}
void loop() {
    var1 = analogRead(pin_enc);
    if (var1 <= var2){
        var1 = var1 +1024;
    }
    Serial.println(var1);
}
```

Figura 35 – Teste de verificação do encoder

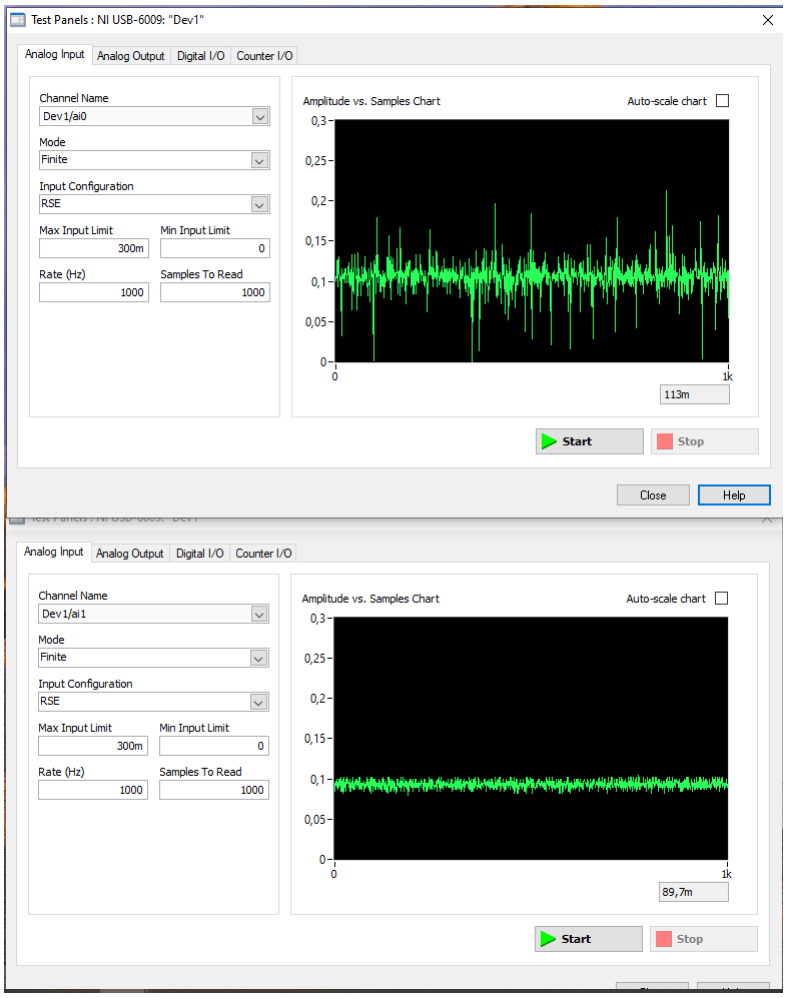


Fonte: Próprio Autor

Semelhante ao potenciômetro, o encoder é alimentado com uma tensão de 0 a 5V e emite como sinal de saída uma fração do sinal em função do ângulo. Desta maneira o teste de funcionamento envolve a verificação do sinal com a alteração do ângulo, porém, o sinal deve se adaptar a diversas posições de instalação por se tratar de um encoder multi-voltas, calculando o sinal correto para uma volta maior que 360º.

Como parte das especificações, foi prevista uma necessidade de redução de ruído na medição do ângulo. Para testar se a especificação foi atingida foi realizado um experimento utilizando a placa NI USB-6009. Na Figura 36 é possível observar na mesma escala o ruído gerado no canal 0 e canal 1, amostrados a uma taxa de $1kHz$, referente à bancada antiga e à bancada nova, respectivamente, onde é possível observar a redução substancial do ruído.

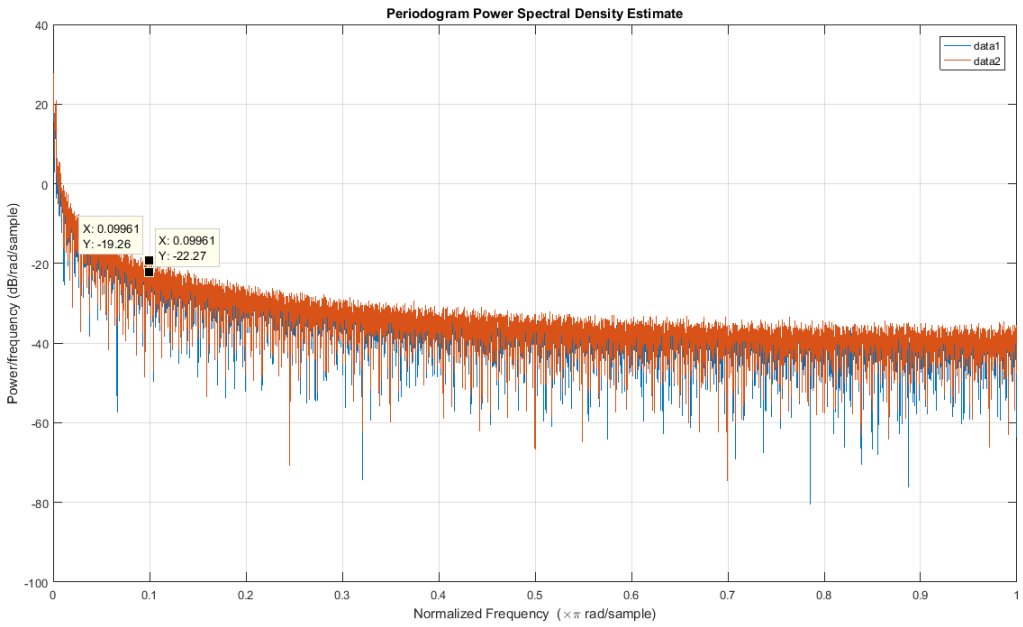
Figura 36 – Amostra da aquisição do ângulo de potenciômetro e encoder



Fonte: Próprio Autor

Para avaliar a melhora quantitativamente, foi realizado um experimento comparando o ruído do encoder e de um potenciômetro novo, a Figura 37 mostra uma redução de $3dB$ ao se utilizar o encoder.

Figura 37 – Periodograma do encoder e potenciômetro novo



Fonte: Próprio Autor

A grande diferença de desempenho entre o potenciômetro usado e o novo é causada pelo desgaste devido ao atrito e diminuindo a longevidade da bancada. Ao se usar um encoder de efeito hall, também é esperado que esse fator não seja impactante, uma vez que este não possui atrito de contato devido a seu princípio de medição.

Sistema de proteção da bancada

O teste do sensor de impacto foi a verificação do funcionamento da interrupção em seu acionamento, para cumprir com sua função de proteção.

```
#define pin_interrupt 3
#define pin_pot A4
#define SerialRate 115200
int var1;
bool Interrupted = 0;
void setup() {
    Serial.begin(SerialRate);
    pinMode(pin_pot,INPUT);
    attachInterrupt(digitalPinToInterrupt(pin_interrupt), stop, RISING);
}
void loop() {
```

```
    if(Interrupted == 1){
        break;
    }
    var1 = analogRead(pin_pot);
    Serial.println(var1);
}
void stop() {
    Serial.println("Stop");
    Interrupted = 1;
}
```

O teste foi realizado variando o valor do potenciômetro e pressionando manualmente o sensor, vendo assim que o programa é interrompido com sucesso.

Chave Seletora

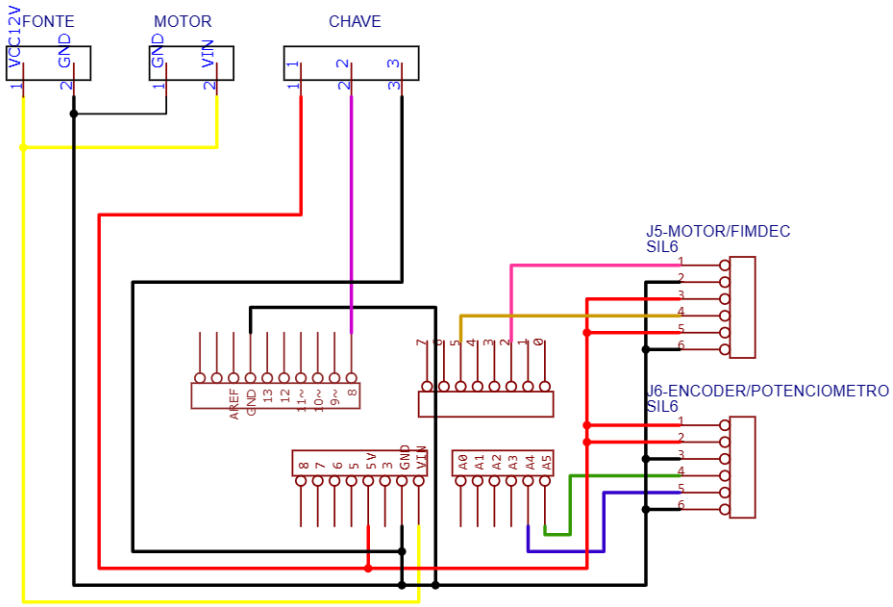
Para o teste da chave seletora foi verificado apenas se ao alterar o estado da alavanca era obtido valores de 0 ou 1, através da leitura de um pino de entrada digital.

```
#define pin_chave 8
#define SerialRate 115200
int var1;
void setup() {
    Serial.begin(SerialRate);
    pinMode(pin_chave,INPUT);
}
void loop() {
    var1 = analogRead(pin_chave);
    Serial.println(var1);
}
```

3.2.2 Fabricação de Placa Impressa

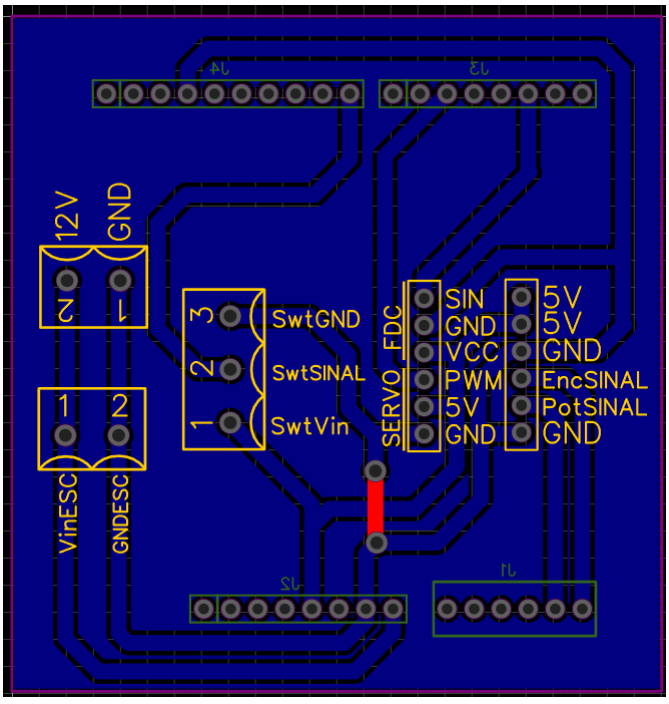
Concluídos os testes de verificação do acionamento correto dos componentes, uma placa impressa foi desenvolvida para distribuir os diferentes sinais dos conectores da Arduino para os componentes eletrônicos e o painel de acionamento. O circuito desenvolvido é exibido na Figura 38 e o projeto da placa impressa resultante na Figura 39.

Figura 38 – Circuito desenvolvido para PCI



Fonte: Próprio Autor

Figura 39 – Circuito Impresso desenvolvido no software Easy EDA



Fonte: Próprio Autor

3.3 Projeto Mecânico

O projeto mecânico é constituído pela definição e fabricação das peças que devem compor a estrutura e suportar os componentes e o painel da bancada. Devido à versatilidade de geometria, a impressão 3D foi utilizada para atender os requisitos geométricos necessários. Nesta seção é elucidado os fatores considerados na fabricação de cada

uma das peças desenvolvidas.

Para satisfazer os requisitos mecânicos do sistema ele deve ser capaz de atender as seguintes especificações:

1. Utilizar encaixe intercambiável desenvolvido em Tavares (2022) para fixação do motor;
2. Garantir a rigidez das hastes;
3. Permitir a passagem de fios pelo interior da estrutura da bancada, ligando diretamente com a placa de controle;
4. Possuir partes mecânicas intercambiáveis para reconfiguração das hastes para obtenção de diferentes modelos mecânicos a partir do mesmo projeto elétrico.
5. Utilizar método de fabricação acessível e replicável

A bancada desenvolvida por Gonçalves (2018), foi considerada como ponto de partida a partir do qual foram consideradas modificações, pois os materiais usados e as proporções se mostraram adequadas para uma bancada didática. Sendo estes o perfil de alumínio central de 70x25mm e as hastes de alumínio de 15x15mm, por serem materiais resistentes e esteticamente agradáveis.

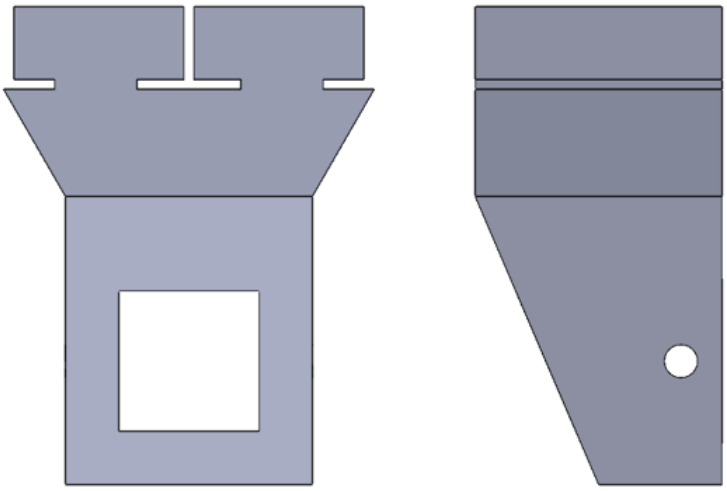
A base é composta por três placas de madeira com espessura de 20mm intercaladas com o perfil de alumínio, permitindo a fixação do perfil e a passagem de fios. Também possui um espaço vazio destinado ao encaixe da caixa da placa de controle. Um parâmetro definidor para a escolha do tamanho do perfil foi o tamanho do encoder e rolamentos que ficam inseridos no perfil. A profundidade do rasgo foi determinada pelo ângulo desejado da haste me repouso.

A impressão 3D foi utilizada para a fabricação do restante das peças, pois era acessível e prontamente disponível, sendo também já utilizada em outros projetos do laboratório.

Para o atendimento da primeira especificação foi desenvolvido uma adaptação da peça criada em Tavares (2022) para o suporte do motor. Dessa maneira é facilitada a troca do motor e a realização de experimentos utilizando ambas as bancadas de empuxo quanto a bancada de identificação do sistema. Por esse motivo foi projetada uma peça que possui o mesmo encaixe que permite acoplar em hastes de alumínio

com dimensões de perfil definidas e a passagem de fios. A peça também contém um furo para fixação da haste de maneira justa, evitando folgas. A peça está exibida na Figura 40

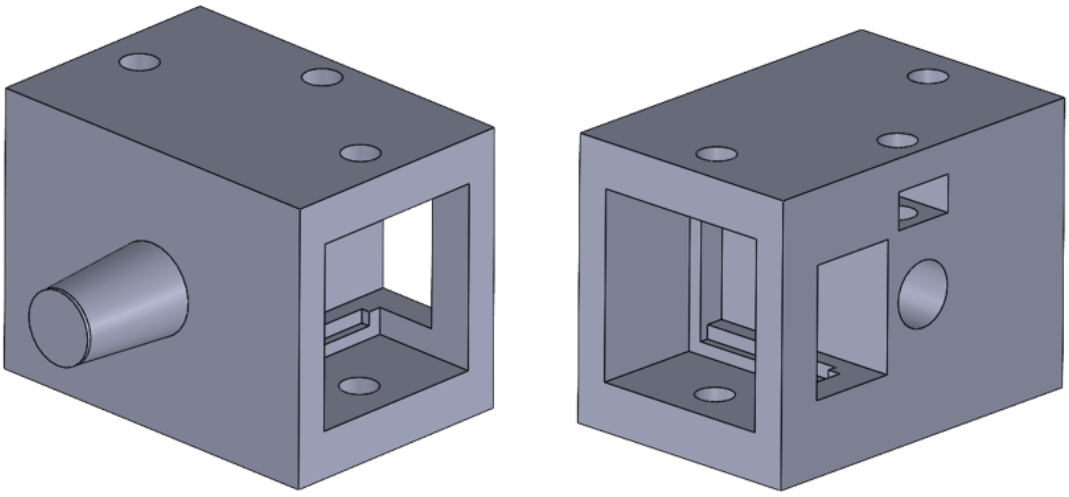
Figura 40 – Vistas do encaixe do motor



Fonte: Próprio Autor

Para a fixação das hastes foi desenvolvido um encaixe central, que fica localizado no eixo de rotação do aeropêndulo, exibido na Figura 41. Esta peça permite o encaixe das hastes e fixação com parafuso, passagem de fios e encaixe do eixo do encoder que conta com um parafuso para eliminar folgas na medição do ângulo. Também foi adicionado um eixo para aumentar a estabilidade, este eixo ficará apoiado em um rolamento oposto ao encoder. As dimensões da peça foram determinadas pelo eixo do encoder, tamanho dos perfis utilizados e o rolamento disponibilizado.

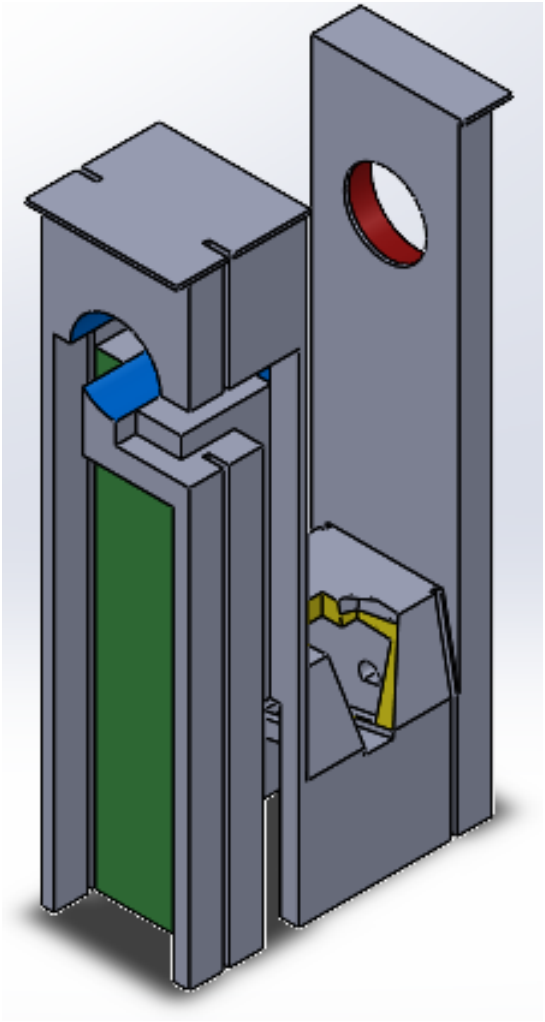
Figura 41 – Vistas do encaixe central



Fonte: Próprio Autor

A Figura 42 mostra a peça central, utilizada como suporte para diversos componentes. Em verde está o local para o posicionamento do CEV, em azul está o local para o posicionamento do encoder, em vermelho está o local para o posicionamento do rolamento e por fim em amarelo a entrada para o posicionamento do fim de curso.

Figura 42 – Peça central inserida no perfil de alumínio

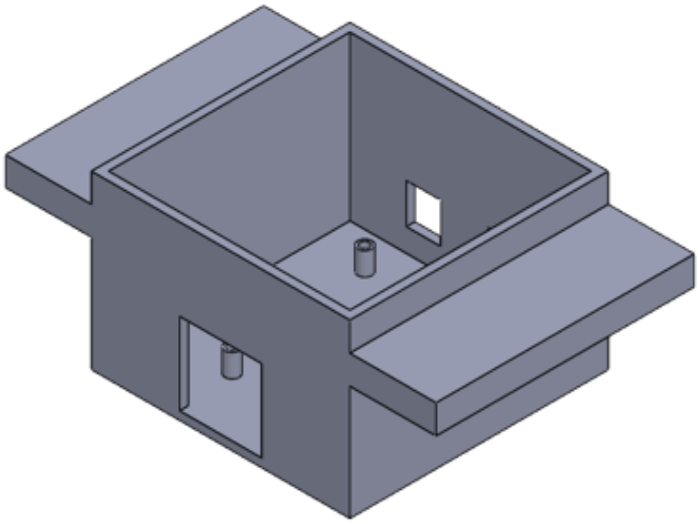


Fonte: Próprio Autor

A peça central foi projetada para conter tanto o rolamento do contra eixo do encaixe central e encoder, além de permitir a passagem de fios das hastes através do perfil. Esta peça também possui uma rampa que define uma angulação em repouso de cerca de 22º. Por fim, deve possuir um espaço para o fim de curso no outro sentido de rotação.

A caixa desenvolvida para encapsular a Arduino foi desenvolvida com ressaltos para permitir o encaixe na base e possui espaço para parafusos para fixação da Arduino. A peça está exibida na Figura 43

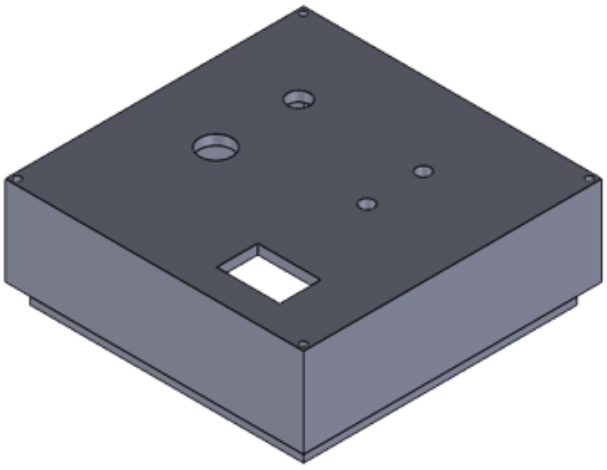
Figura 43 – Caixa para Arduino



Fonte: Próprio Autor

A tampa da caixa foi desenvolvida possuindo espaços para comportar os conectores de alimentação, o potenciômetro, a chave de energia da plataforma e uma chave para as funcionalidades de malha aberta e fechada. A tampa é exibida na Figura 44.

Figura 44 – Tampa da caixa para Arduino



Fonte: Próprio Autor

A Tabela 2 mostra o custo dos principais componentes para fabricação da bancada, incluindo o valor total gasto e o custo unitário por banca, e a Figura 45 mostra o projeto concluído.

Tabela 2 – Custo dos principais componentes

Componente	Custo para 4 bancadas	Custo unitário
Base de madeira	R$ 100,00	R$ 25,00
Conjunto Motor ESC	R$ 162,72	R$ 40,68
Encoder	R$ 350,44	R$ 87,61
Arduino UNO	R$ 207,00	R$ 51,75
Fim de curso	R$ 67,00	R$ 16,75
Placa Impressa	R$ 421,23	R$ 105,28
Total	R$ 1.308,39	R$ 327,09

Fonte: Próprio Autor

Figura 45 – Projeto mecânico concluído

Fonte: Próprio Autor

3.4 Desenvolvimento de Interface Gráfica para Operação da Bancada

Nesta seção é apresentado o funcionamento da interface gráfica desenvolvida para operação da bancada. Esta foi desenvolvida visando ser uma maneira rápida de realizar um experimento e extrair as informações da resposta temporal, já que o MATLAB é utilizado nas atividades práticas como ferramenta de análise. A operação da interface não exige necessidade de um conhecimento de ferramentas específicas, apenas os parâmetros dos experimentos que se quer realizar.

Está plataforma é utilizada para o ensino da modelagem de sistemas, observação do comportamento em malha aberta e identificação experimental. Também permite a sintonia de um controlador proporcional para análise de estabilidade e testes de

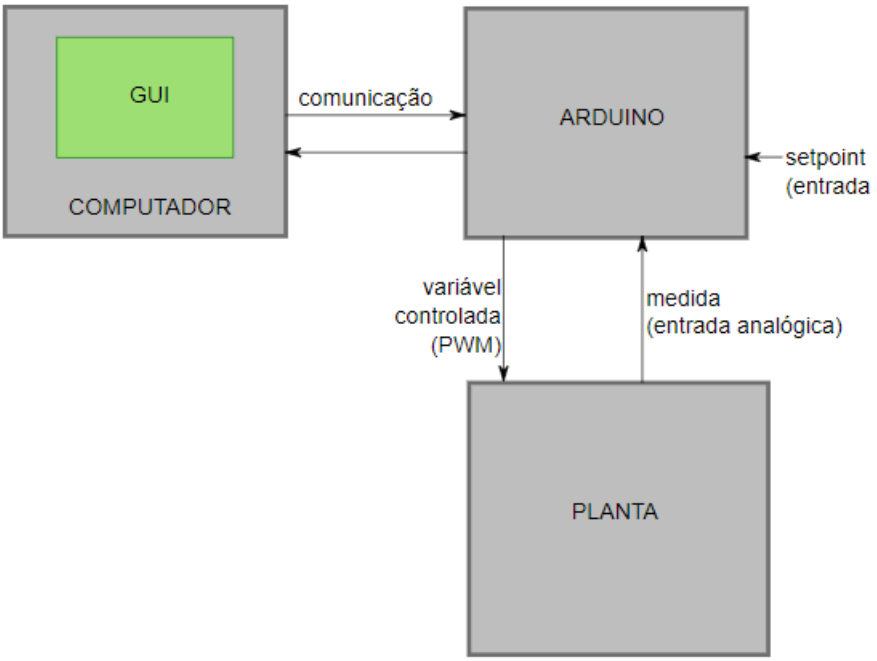
sintonia de controladores PID e compensadores.

Para satisfazer os requisitos necessários a interface deve ser capaz de atender as seguintes especificações:

1. Possuir a funcionalidade de realizar experimentos com sinais de entrada do tipo degrau, senoidal e sinal pseudoaleatório (SBPA);
2. Permitir o teste de controladores PID e compensadores em malha fechada;
3. Possibilitar alteração do período de amostragem;
4. Permitir a exportação dos dados do experimento em um formato padronizado;

O expGUI é uma interface gráfica desenvolvida por CAJUEIRO (2024), em python, que se comunica com a Arduino que condiciona os sinais físicos e controle dos atuadores durante um experimento. A Figura 46 exemplifica o fluxo de informação.

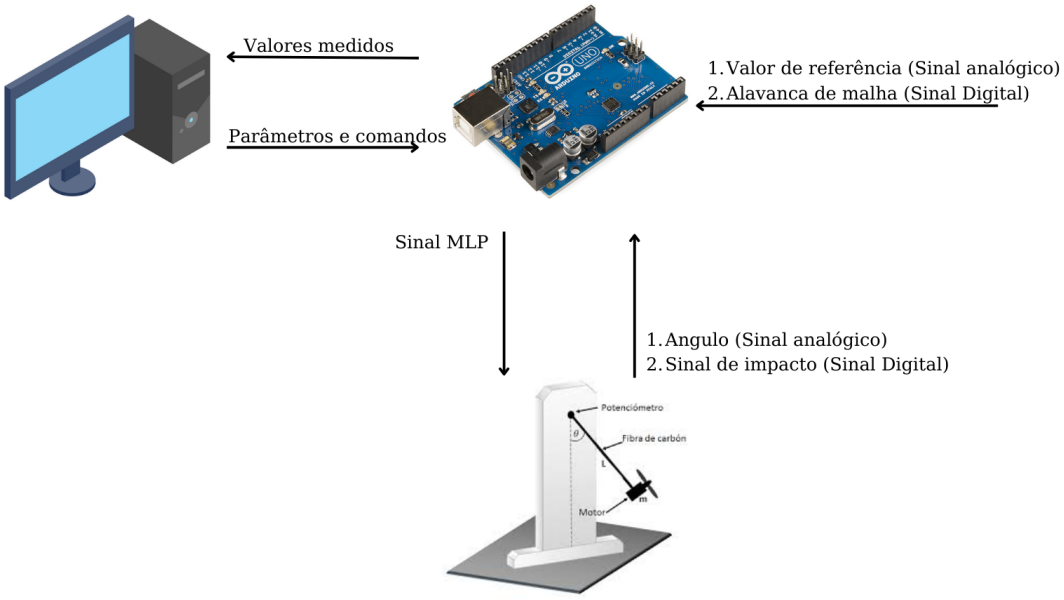
Figura 46 – Fluxo de informação expGUI



Fonte: CAJUEIRO (2024)

Neste trabalho essa interface foi modificada, com uma alteração de interface e programação de algumas especificidades relacionada ao funcionamento do aeropêndulo. A Figura 47 mostra o fluxo de informaçãoconsiderado para a bancada.

Figura 47 – Fluxo de informação bancada



Fonte: Próprio Autor

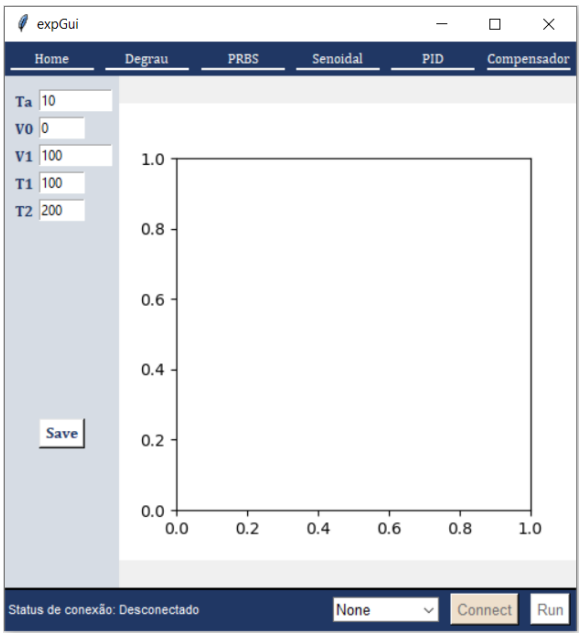
O desenvolvimento dessa interface pretende facilitar a integração entre realização de experimentos e obtenção dos dados de saída para diversas entradas, além da possibilidade de testar controladores tanto em malha aberta quanto fechada. Os experimentos disponíveis são:

- Resposta em malha aberta ao degrau
- Resposta em malha aberta a um sinal senoidal
- Resposta em malha aberta a um sinal binário pseudo aleatório (SBPA)
- Resposta em malha fechada para um valor de referência v_{sp}

A Figura 48 mostra a tela para o experimento de resposta ao degrau. É possível observar os botões para acessar os outros experimentos, os campos de configuração do experimento em que se encontra e o botão de exportar dados.

Para definição da comunicação da Arduino com o expGui foi criado uma biblioteca chamada 'communication.py', que define uma classe 'SerialPort' que possui os métodos

Figura 48 – Interface expGUI para o experimento Degrau



Fonte: Próprio Autor

de conexão serial, definição de parâmetros e leitura de dados e está presente no Apêndice C. Todas as funções de definição de parâmetros de experimento exigem uma resposta da Arduino para confirmação do envio de dados. Após a definição dos parâmetros a interface aguarda uma resposta da Arduino para confirmação da atualização das variáveis.

As funções criadas são exibidas na Tabela 3.

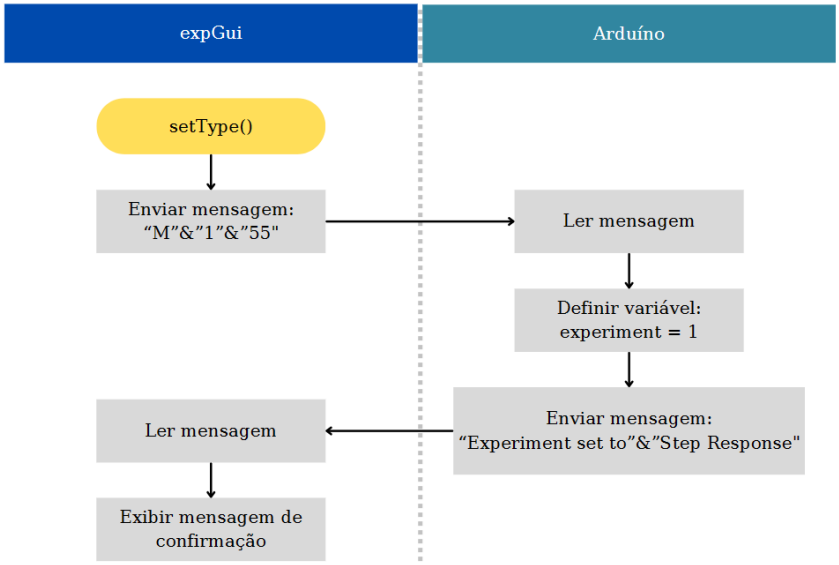
Tabela 3 – Descrição das funções de comunicação do expGui

Função	Descrição	Payload
connect(self)	Conecta a interface e Arduino	-
disconnect(self)	Desconecta a interface e Arduino	-
setType(self,expType)	Define o experimento a ser realizado	M
setTA(self,Ta)	Define o período de amostragem	A
setVotlages(self, Vsp, Vmin, Vmax)	Define as tensões do sistema	V
setSenoidal(self, Freq, Amp)	Define os parâmetros do Senoidal	C
setPRBSTimes(self, Tmin, Tmax)	Define os parâmetros do SBPA	B
setPID(self, Kp, Ki, Kd)	Define os parâmetros do PID	Q
setLeadLag(self, Kp, Tc1, Tc2)	Define os parâmetros do Compensador	L
run(self)	Inicializa o experimento	R
getMeasure(self)	Lê as variáveis V_{in},V_{out} e V_{sp}	E

Fonte: Próprio Autor

O fluxograma de definição do experimento em resposta ao degrau está exibido na Figura 49.

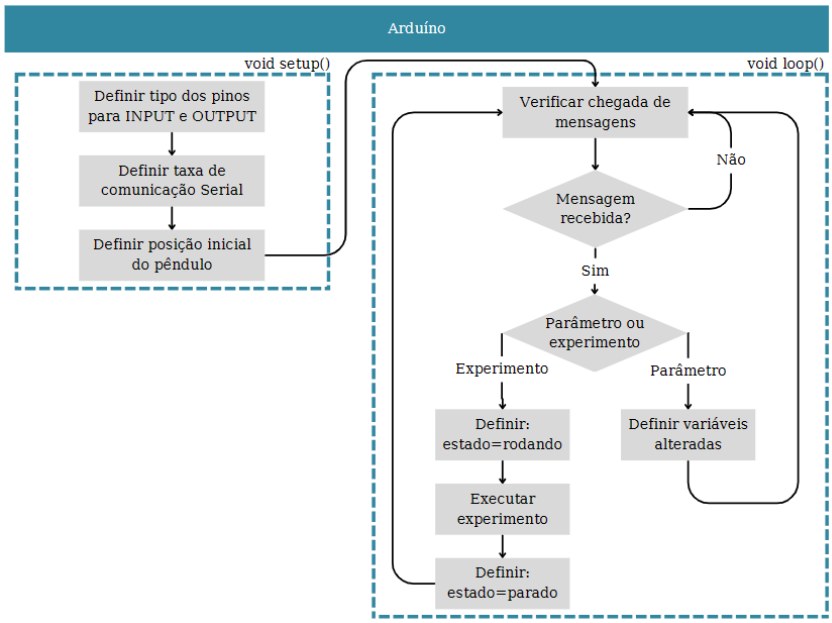
Figura 49 – Fluxograma de definição de experimento



Fonte: Próprio Autor

O código da Arduino para integração do sistema é baseado em aguardar uma mensagem enviada pelo computador para definição dos parâmetros, e em seguida realização do experimento e está presente no Apêndice A. No código de setup temos a definição do estado inicial, variáveis de tempo e a definição do pino do atuador, caso tenha sido definido uma saída do tipo servomotor, utilizará biblioteca de servomotor para controle, caso contrário será utilizado uma saída MLP genérica. O fluxograma geral da Arduino é exibido na Figura 50

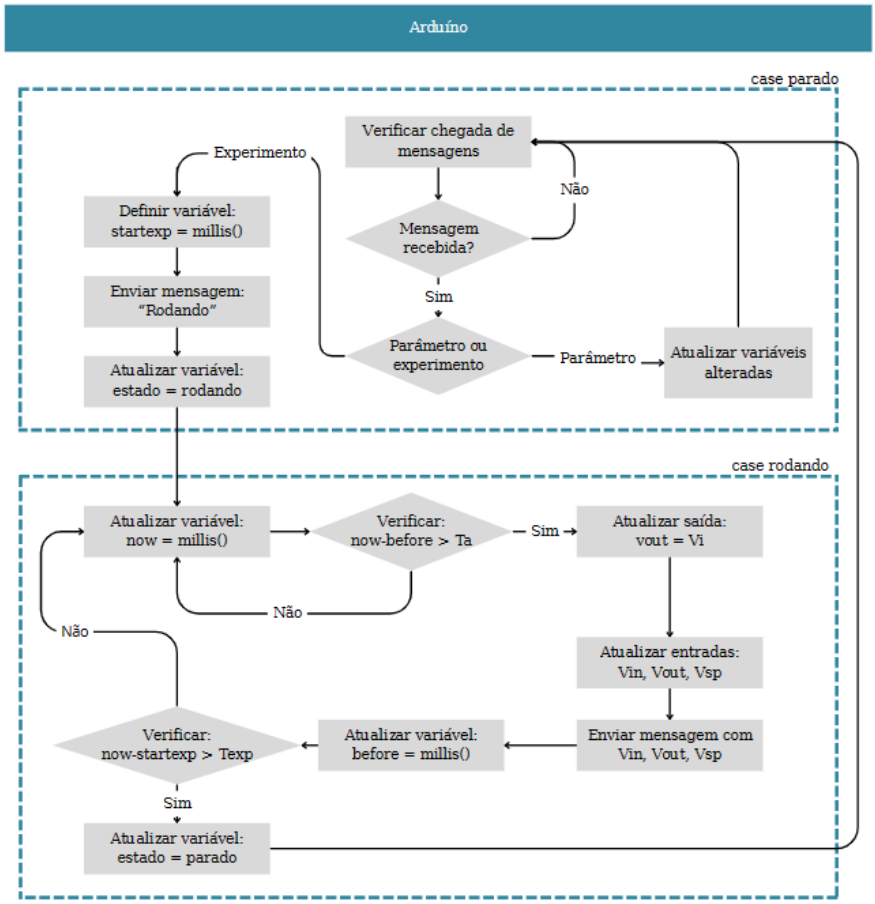
Figura 50 – Fluxograma de funcionamento da Arduino



Fonte: Próprio Autor

Ao executar um experimento, a interface realiza primeiro a atualização de todas as variáveis pertinentes àquele experimento e em seguida envia o sinal para iniciar. Durante o funcionamento de um experimento, o Arduino tem o seu comportamento baseado no modo selecionado, calculando o sinal de atuação e obtendo os sinais apenas para cada múltiplo do período de amostragem. Todo o fluxograma de um experimento está exibido na Figura 51.

Figura 51 – Fluxograma de funcionamento de um experimento



Fonte: Próprio Autor

Durante o experimento o Arduino responde com T_2/T_a pacotes, um para cada ponto do experimento. Cada pacote é composto por 7 bytes da forma mostrada na Tabela 4, onde MSB representa o byte mais significativo e LSB o byte menos significativo. Essa informação é então traduzida na interface gráfica e armazenada em um array no expGui, através da função 'getMeasure'.

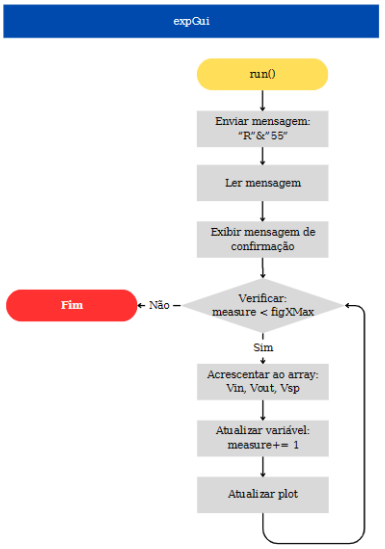
Tabela 4 – Resposta do Arduino

Byte	1	2	3	4	5	6	7
Valor	'E'	vout	vin(MSB)	vin(LSB)	vsp(MSB)	vsp(LSB)	EOP

Fonte: Próprio autor

O fluxograma de execução do expGui durante um experimento está exibido na Figura 52 e presente no Apêndice B.

Figura 52 – Fluxograma de funcionamento da interface



Fonte: Próprio Autor

Os parâmetros de cada experimento são mostrados na Tabela 5. Todos os experimentos de malha aberta funcionam a partir da divisão do experimento em dois períodos. O período de 0 a T_1, que recebe a tensão V_0, utilizado para estabilizar o aeropêndulo em torno do ponto de linearização θ_{ss}. O segundo período de T_1 a T_2 que varia conforme o experimento.

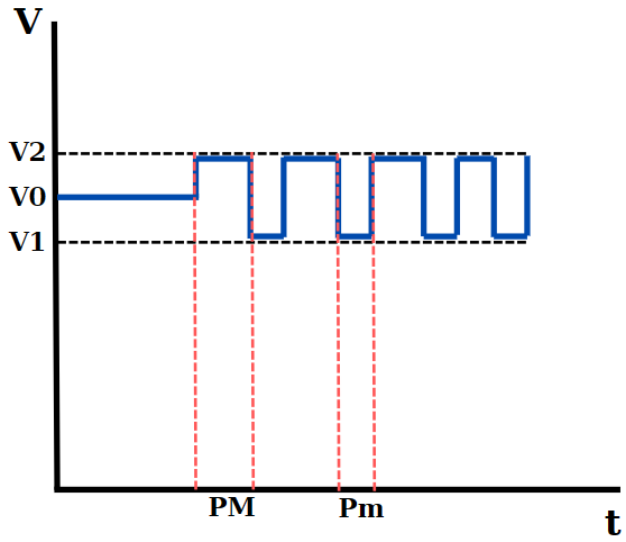
Tabela 5 – Variáveis utilizadas na interface

Variável	Descrição
T_a	Período de amostragem
T_1	Tempo de início do segundo período
T_2	Tempo total do experimento
V_0	Tensão inicial do experimento
V_1	Tensão final do experimento
V_2	Tensão máxima do PRBS
P_m	Período mínimo do PRBS
P_M	Período máximo do PRBS
F_1	Frequência do sinal senoidal
A_1	Amplitude do sinal senoidal
K_P	Constante proporcional
K_i	Constante integral
K_D	Constante derivativa
T_{c1}	Constante de tempo 1
T_{c2}	Constante de tempo 2

Fonte: Próprio Autor

O segundo período do experimento para o sinal degrau corresponde apenas na alteração da tensão V_0 para a tensão V_1. No experimento SBPA representa a variação da tensão de V_1 para V_2 com um período de P_m ou P_M, exibido na Figura 53. E no experimento senoidal apresenta um sinal centrado em V_0 variando com frequência F_1 e amplitude A_1.

Figura 53 – Exemplo de sinal SBPA



Fonte: Próprio Autor

4 DISCUSSÕES DE RESULTADOS

Este capítulo apresenta os testes de integração realizados, garantindo que todas as funcionalidades presentes nos objetivos sejam satisfeitas. Como uma maneira de exibir todo o funcionamento da bancada, foi realizado um experimento de identificação de uma configuração de aeropêndulo e testado seu desempenho com a presença de um controlador PID.

4.1 Testes de integração

Os testes de integração realizados serão realizados por meio de uma sequência de experimentos para garantir o funcionamento as funcionalidades principais da bancada, tais como funcionamento, garantia de dados corretos e testes da interface e sua operação.

Os teste realizados foram conduzidos com a placa Arduino conectada a um computador e a tensão de alimentação de 10V para o motor.

4.1.1 Medição do ângulo

Para garantir a medição correta do ângulo foi realizado uma comparação entre ângulo registrado no programa e ângulos conhecidos, como o de repouso e 90º.

Para evitar ajustes trabalhosos referentes ao posicionamento do sensor, foi realizado um simples cálculo baseado na angulação em repouso do aeropêndulo e considerando que o sinal do encoder, na posição de instalação, é inversamente proporcional ao ângulo θ. Foi considerado então uma equação linear do tipo:

$$\theta = aV_{enc} + b \tag{4.1}$$

Considerando que a angulação inicial em repouso é 22º, temos então e que o termo linear a é faixa de valores de θ (0 a 360º) dividido pela faixa de valores de V_{enc} (0 a 1024):

$$b = \frac{360}{1024} V_{encI} + 22 \tag{4.2}$$

Uma vez determinado o valor de b temos o valor inicial do ângulo.

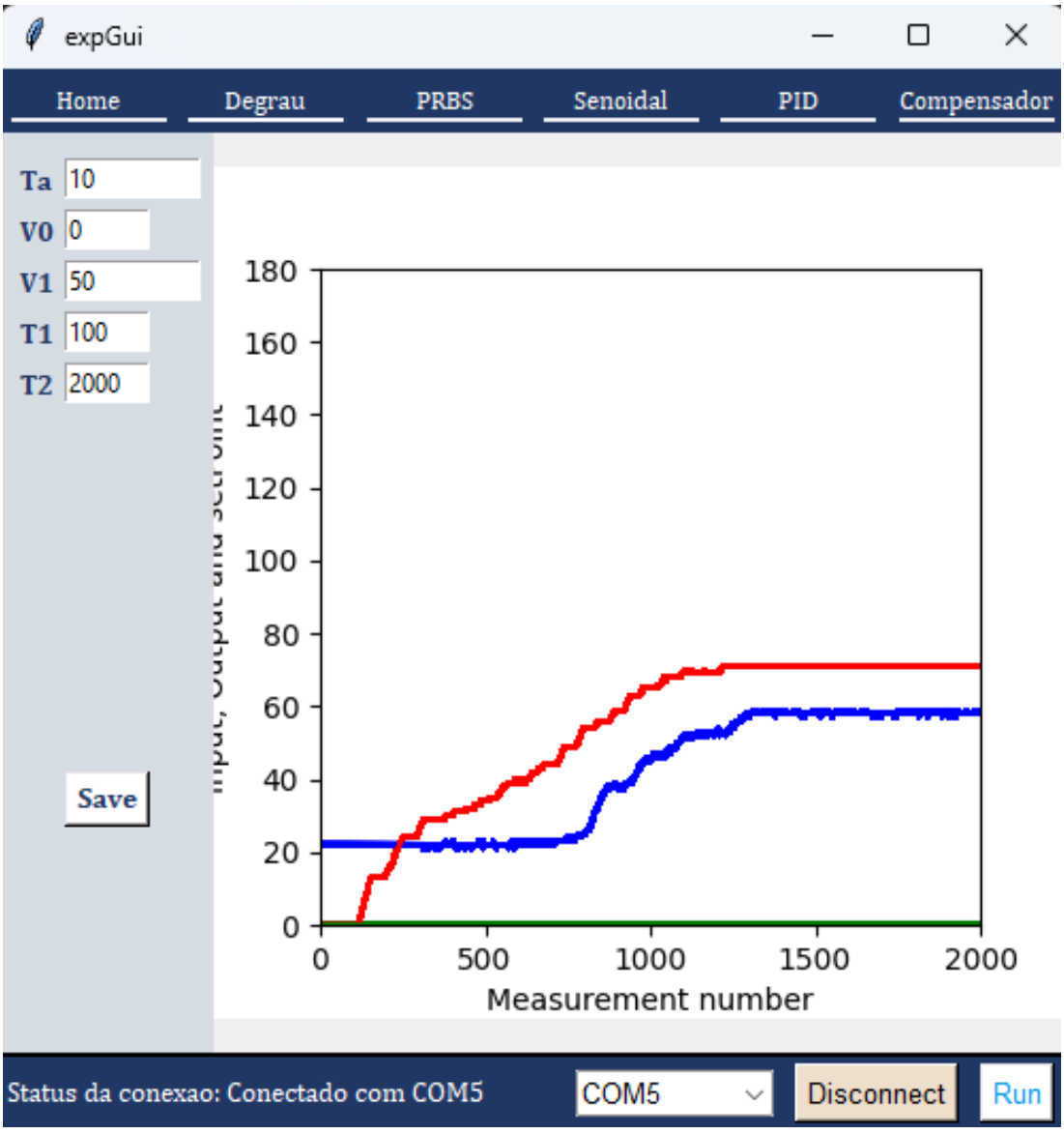
4.2 Experimento de identificação

Esta seção exemplifica o procedimento de identificação que deve ser realizado na bancada e também mostra o desenvolvimento de um controlador PID e um compensador e o desempenho na utilização da bancada.

4.2.1 Experimento em malha aberta

Para a identificação do sistema, primeiro foi realizado em malha aberta um experimento para identificar a tensão que faz com que o sistema atinja um ângulo θ_{ss}, o ângulo escolhido foi de 57º. O resultado desse experimento está exibido na Figura 54

Figura 54 – Experimento em malha aberta

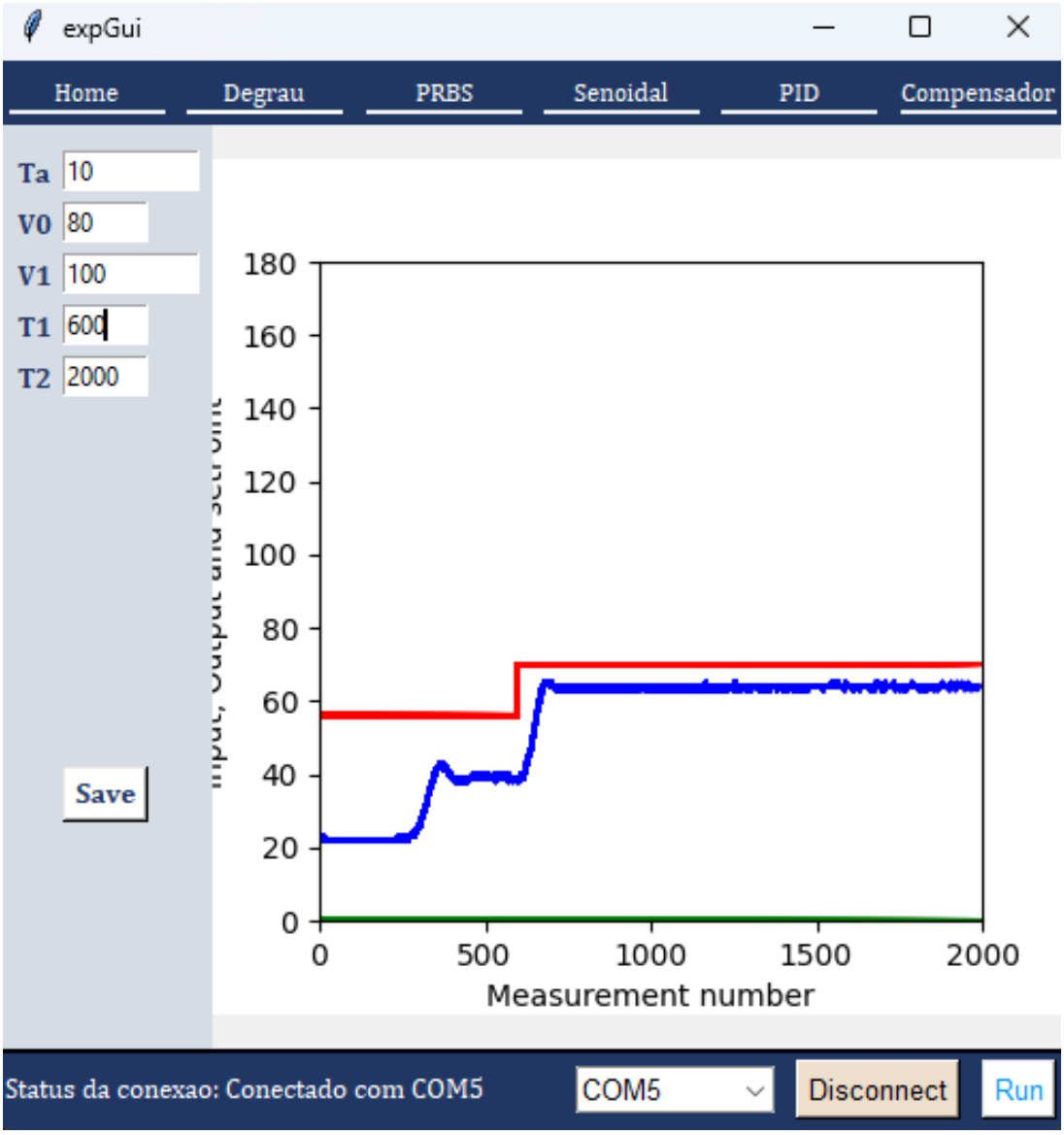


Fonte: Próprio Autor

4.2.2 Experimento Degrau

Em seguida foi obtido a resposta ao degrau desse sistema, para determinar o tempo de subida, valor que será necessário para determinar um dos períodos para o experimento do SBPA. O tempo de subida t_r do sistema foi aproximadamente 4 segundos. A resposta ao degrau pode ser observada na Figura 55

Figura 55 – Experimento Degrau

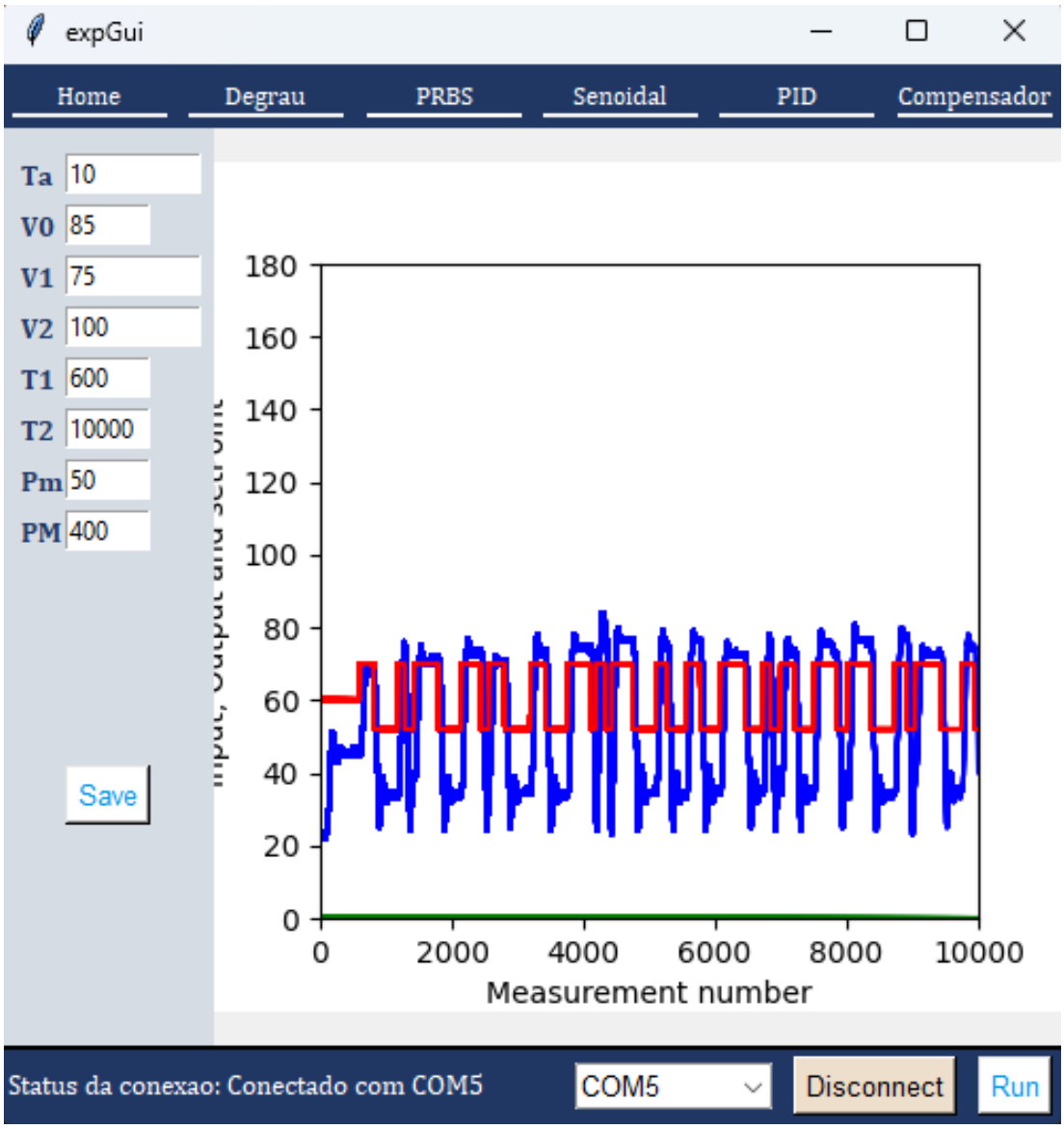


Fonte: Próprio Autor

4.2.3 Experimento SBPA

Em seguida foram realizados alguns experimentos através do sinal SBPA (Figura 56), onde foi obtida a resposta temporal do sistema. Esses experimentos diferem em tempo de período mínimo e duração.

Figura 56 – Experimento SBPA



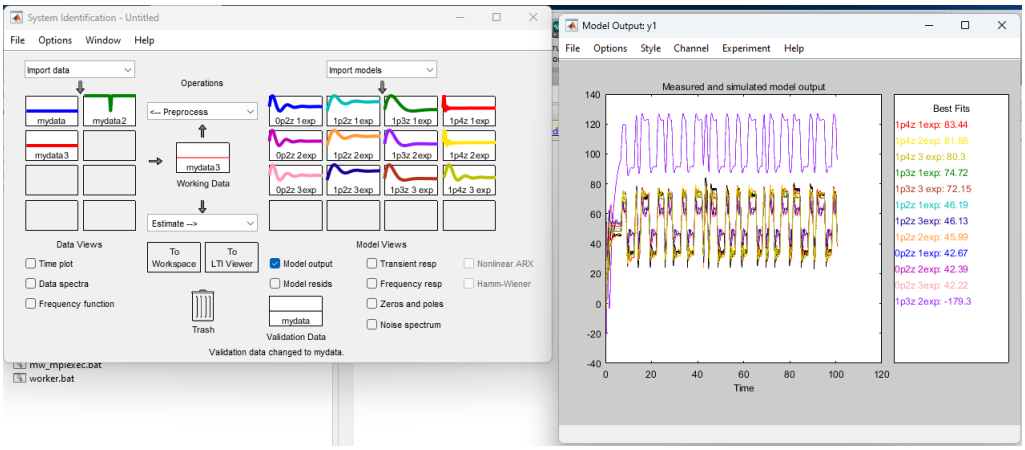
Fonte: Próprio Autor

4.2.4 Identificação do sistema

A partir dos dados obtidos, foi utilizado a toolbox de identificação de sistemas do MATLAB para obter um sistema de que melhor representasse o seu comportamento, mostrado na Figura 57.

No processo de carregamento de dados ao MATLAB é necessário realizar um tratamento inicial que envolve a separação dos dados a partir do tempo T1. Essa separação é necessária pois este será o período a partir da estabilização do sistema no ponto de operação, caso contrário o sistema pode conter muitas não linearidades que foram simplificadas anteriormente.

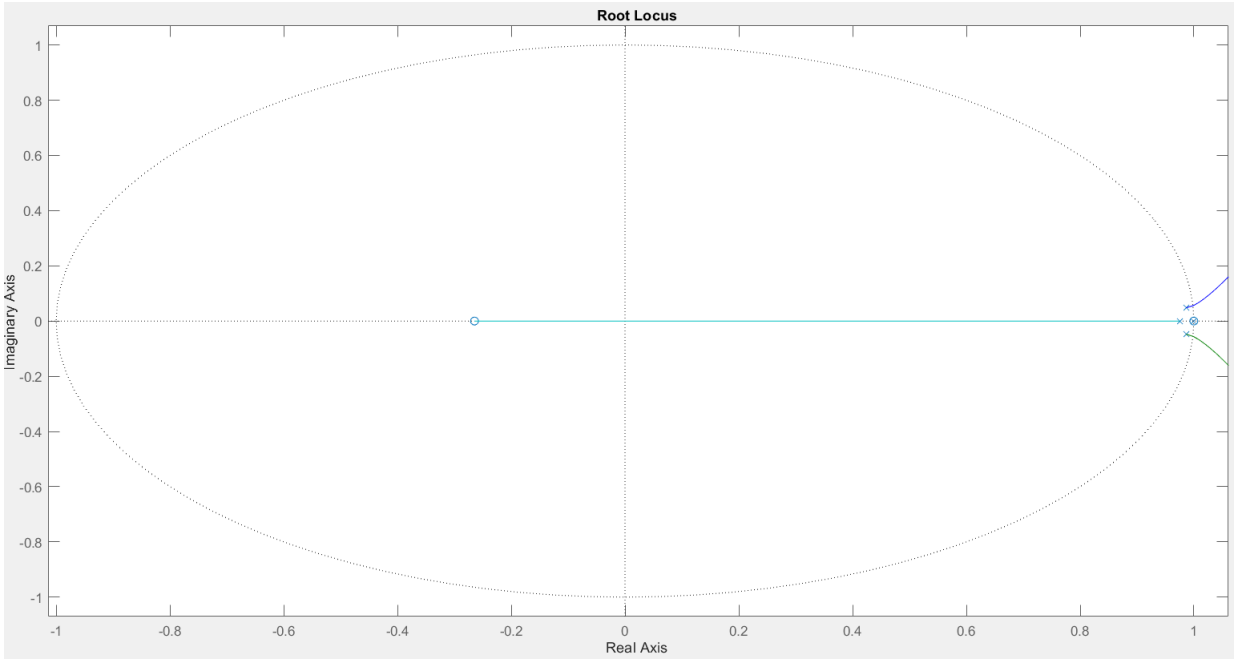
Figura 57 – Identificação de sistemas



Fonte: Próprio Autor

A equação 4.3 e a Figura 58 mostram o modelo do sistema considerado para o teste de controle na próxima seção e o seu lugar das raízes respectivamente.

Figura 58 – Lugar das raízes do modelo



Fonte: Próprio Autor

$$G(s) = \frac{143s + 2.611}{s^4 + 5.044s^3 + 31,65s^2 + 64.77s + 2.92} \tag{4.3}$$

O modelo foi discretizado utilizando a função 'c2d' do MATLAB, considerando o método de discretização padrão de hold de ordem zero e um tempo de amostragem $T_a = 0.01s$. A função de transferência obtida esta exibida na equação 4.4;

$$G(z) = \frac{2.353 \cdot 10^{-5}z^{-1} + 6.94 \cdot 10^{-5}z^{-2} - 6.997 \cdot 10^{-5}z^{-3} - 2.294 \cdot 10^{-5}z^{-4}}{1 - 3.948z^{-1} + 5.846z^{-2} - 3.849z^{-3} + 0.9508z^{-4}} \tag{4.4}$$

4.2.5 Teste de controle

A implementação do controlador foi realizada conforme o sistema exibido na Figura 27.

$$u(k) = u_p(k) + u_i(k) + u_d(k) \tag{4.5}$$

Onde:

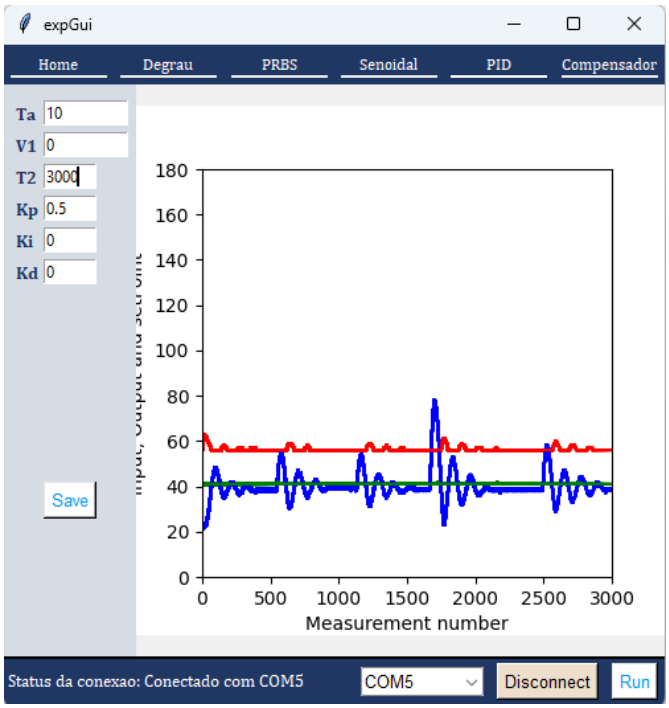
$$u_p(k) = K_p e(k)$$
$$u_i(k) = u_i(k-1) + K_i \frac{e(k) + e(k-1)}{2} T_a$$
$$u_d(k) = \frac{K_d}{T_a}(PV(k) - PV(k-1))$$

A Figura 59 mostra resultado da aplicação de um controlador proporcional com parâmetros $K_p = 0.5$ sob o sistema exibido em 4.4. Neste experimento, distúrbios foram aplicados manualmente para avaliar a capacidade do sistema de retornar ao valor de referência.

Como exemplo de análise foi realizado um teste com um controlador proporcional com um ganho que, de acordo com o lugar das raízes do sistema, o levaria a instabilidade. Foi determinado que o valor de $K = 2$ tornaria o sistema instável. A Figura 60 mostra que o sistema apresentou instabilidade para este ganho.

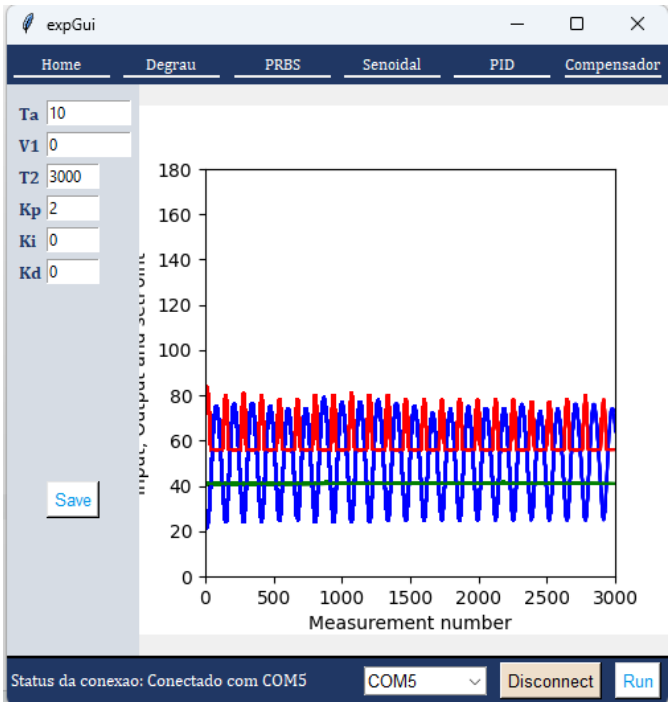
A implementação do compensador de atraso-avanço pode ser implementada através da relação apresentada em Krikelis e Fassois (1984):

Figura 59 – Teste de controlador PID



Fonte: Próprio Autor

Figura 60 – Experimento de controlador proporcional

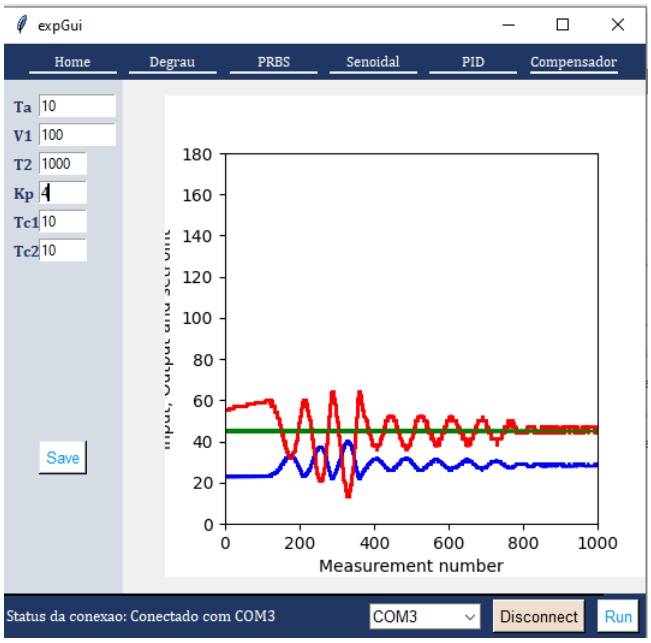


Fonte: Próprio Autor

$$u(k) = \frac{T_2}{T_1 + T_a}u(k-1) + K_p\frac{T_1 + T_a}{T_2 + T_a}e(k) - K_p\frac{T_1}{T_2 + T_a}e(k-1) \tag{4.6}$$

A Figura 61 mostra a realização do controle com um compensador.

Figura 61 – Experimento de teste de um compensador



Fonte: Próprio Autor

4.2.6 Atividades práticas propostas

As atividades propostas para esta bancada são:

1. Modelagem e identificação Experimental de um motor CC sem escovas

   Esta atividade deve ser realizada através da modelagem analítica exibida na seção 2.1.2 e do uso da bancada de tração desenvolvida em Tavares (2022). A bancada possui a capacidade de determinar da tensão com a força de empuxo em regime permanente, e está sendo modificada para possui a funcionalidade de identificar os parâmetros do regime transiente.

2. Montagem e Modelagem analítica do aeropêndulo

   A modelagem analítica consiste na etapa de determinar o modelo do sistema através das relações físicas, de forma semelhante ao procedimento desenvolvido na seção 2.1.3.

3. Identificação Experimental do aeropêndulo

   Essa prática solicita ao aluno que determine uma função de transferência para o modelo, considerando como entrada a tensão e saída o ângulo do aeropêndulo. Uma identificação experimental do sistema foi realizada na seção 4.2.4 e mostra de maneira resumida o procedimento de identificação através dos experimentos

de malha aberta e com sinal pseudo aleatório. Nesse experimento será necessário por parte do aluno determinar os parâmetros adequados do experimento para obter resultados satisfatórios no modelo identificado.

4. Avaliação da performance do controlador proporcional

   Nesta atividade são avaliados os comportamentos das especificações principais de performance citadas na seção 2.3.2. Na seção 4.2.5 foi demonstrado um caso exemplo de como avaliar os resultados de um controlador proporcional e uma das possíveis análises que podem ser realizadas com o uso da bancada.

5. Sintonia de um controlador PID para o aeropêndulo

   Esta prática consiste em dar continuidade a Prática 4, porém desta vez com o uso dos parâmetros integral e derivativo do controlador. Uma análise quantitativa de desempenho pode ser realizada comparando os resultados das especificações com o controlador proporcional utilizado.

6. Projeto de um compensador para o aeropêndulo

   Esta prática é semelhante a Prática 5 porém desta vez com o uso de um compensador. A determinaçaõ das constantes de tempo podem ser determinadas analiticamente ou através de ferramentas como o a Toolbox sisotool do MATLAB.

   Todas as atividades serão realizadas com o auxílio do MATLAB e suas ferramentas, onde os dados serão analisados e modelos serão obtidos.

5 CONCLUSÕES

Neste trabalho foi desenvolvida uma bancada de teste para o ensino prático da teoria de controle para estudantes de Engenharia Mecânica. A bancada foi capaz de realizar experimentos de identificação e o teste de controladores PID e compensadores de atraso-avanço.

As melhorias propostas para a bancada anterior foram executadas, permitindo maior variabilidade do sistema mecânico e elétrico. Adicionalmente a alteração do componente de medição do ângulo, com o uso de um encoder, mostrou redução substancial do ruído presente no sinal.

No desenvolvimento do projeto elétrico foi desenvolvida uma placa impressa para facilitar a montagem do sistema elétrico por meio do uso de conectores. Ademais, a implementação do controle via Arduino permitiu a inclusão de algumas funcionalidades, como o funcionamento de malha aberta e fechada e a inclusão de uma proteção por meio de um sensor de impacto. Além disso a liberdade de programação permite a inclusão de novas funcionalidades caso desejado.

A fabricação das peças através da manufatura aditiva se mostrou satisfatória, porém contou com a tentativa de diversos protótipos falhos até o desenvolvimento das peças que atendiam as dimensões necessárias. Essas falhas se devem a erros de projeto e variações do processo, podendo ser variações do material usado ou falhas dimensionais advindas do processo.

Além disso, foi adaptado uma interface gráfica em python desenvolvida por um professor para experimentos de identificação de sistemas. A interface foi modificada para atender os requisitos específicos desta bancada, além de algumas melhorias que foram realizadas.

Foi executado o experimento de identificação para uma configuração de sistema, mostrando as etapas necessárias para a identificação experimental. Por fim, foram desenvolvidos controladores para executar os testes em malha fechada de um controlador PID e um compensador.

REFERÊNCIAS BIBLIOGRÁFICAS

AGUADO, N. A. Teaching research methods: Learning by doing. *Journal of Public Affairs Education*, Routledge, v. 15, p. 251–260, 2009. Disponível em: <https://doi.org/10.1080/15236803.2009.12001557>.

AliExpress. *Encoder P3022-V1-CW360*. 2024. [Online; accessed 27-April-2024]. Disponível em: <https://pt.aliexpress.com/item/1005005037001141.html>.

AliExpress. *Motor CC Sem Escova*. 2024. [Online; accessed 27-April-2024]. Disponível em: <https://pt.aliexpress.com/i/32611937154.html>.

ARDUINO. *What is Arduino?* 2018.

ÅSTRÖM, K.; WITTENMARK, B. *Computer-Controlled Systems: Theory and Design, Third Edition*. Dover Publications, 2011. (Dover Books on Electrical Engineering). ISBN 9780486486130. Disponível em: <https://books.google.com.br/books?id=9Y6D5vviqMgC>.

CAJUEIRO, J. *expGui*. [S.l.]: GitHub, 2024. <https://github.com/joaopaulo-cerquinhocajueiro/expGui>.

CARDOSO, V. G.; SOUZA, T. F. F. de; FOYO, P. M. G. del. Flywheel inverted pendulum design for evaluation of swing-up energy-based strategies. In: ABCM. *Congresso Brasileiro de Automática*. [S.l.], 2022.

CASSANDRAS, C. G.; LAFORTUNE, S. Introduction to discrete event systems. In: ____. [S.l.: s.n.], 2010. p. 800. ISBN 1441941193.

CNE, C. D. E. S. *Diretrizes Curriculares Nacionais do Curso de Graduação em Engenharia*. [S.l.]: http://portal.mec.gov.br/cne/arquivos/pdf/CES112002.pdf, 2002.

DIAS, I. J. do N. S. A. *Construção de Plataforma de Teste para Algoritmo de Controles de Drone*. [S.l.], 2021.

DORF, R. C.; BISHOP, R. H. *Modern Control Systems*. 11. ed. [S.l.]: Pearson Education, Inc., 2008.

ENIKOV, E. T.; CAMPA, G. Mechatronic aeropendulum: Demonstration of linear and nonlinear feedback control principles with matlab/simulink real-time windows target. *IEEE TRANSACTIONS ON EDUCATION*, v. 55, n. 4, p. 538–545, 2012.

ENIKOV, E. T.; CAMPA, G. Mechatronic aeropendulum: Demonstration of linear and nonlinear feedback control principles with matlab/simulink real-time windows target. *IEEE TRANSACTIONS ON EDUCATION*, v. 55, p. 538–545, 2012.

FRANKLIN, G.; POWELL, J.; WORKMAN, M. *Digital Control of Dynamic Systems-Third Edition*. [S.l.: s.n.], 2022. ISBN ISBN: 0-9791226-3-5 or ISBN13: 978-0-9791226-3-7.

GONÇALVES, S. J. N. *Projeto e Construção de Bancada Didática com Controladores eletrônicos*. [S.l.], 2018.

HABIB, G.; MIKLOS, A.; ENIKOV3, E. T.; STEPAN, G.; REGA, G. Nonlinear model-based parameter estimation and stability analysis of an aero-pendulum subject to digital delayed control. *International Journal of Dynamic and Control*, n. 5, p. 629–643, 2017.

ielectrony. *YL-99 Impact Switch Module*. 2024. [Online; accessed 27-April-2024]. Disponível em: <https://ielectrony.com/en/product/yl-99-impact-switch-module-2/>.

KRIKELIS, N. J.; FASSOIS, S. D. Microprocessor implementation of pid controllers and lead-lag compensators. *IEEE Transactions on Industrial Electronics*, IE-31, p. 79–85, 1984. Disponível em: <https://api.semanticscholar.org/CorpusID:42118103>.

KUNDU, P. K.; COHEN, I. M.; DOWLING, D. R. Chapter 14 - aerodynamics. In: ____. Sixth edition. Academic Press, 2016. p. 773–817. ISBN 978-0-12-405935-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780124059351000149>.

KUO, B. C.; GOLNARAGHI, F. *Automatic Control Systems*. 8th. ed. USA: John Wiley & Sons, Inc., 2002. ISBN 0471134767.

MakerHero. *ESC 30A Brushless com Bec Interno 2A/5V Aeromodelos*. 2024. [Online; accessed 27-April-2024]. Disponível em: <https://www.makerhero.com/produto/esc-30a-brushless-com-bec-interno-2a5v-aeromodelos/>.

National Instruments. *Implementing the PID Algorithm with the PID VIs*. 2024. [Online; accessed 19-May-2024]. Disponível em: <https://www.ni.com/docs/en-US/bundle/labview/page/implementing-the-pid-algorithm-with-the-pid-vis.html>.

National Instruments. *PID Algorithms*. 2024. [Online; accessed 19-May-2024]. Disponível em: <https://www.ni.com/docs/en-US/bundle/labview/page/pid-algorithms.html>.

NELSON, W. C. *Airplane Propeller Principles*. [S.l.]: Jonh Wiley Sons Inc., 1944.

NETO, R. C.; TRINDADE, F. L. A.; MARQUES, B. R. A.; AZEVEDO, G. M. S.; BARBOSA, E. J.; BARBOSA, E. A. O. An aeropendulum-based didactic platform for the learning of control engineering. *Journal of Control, Automation and Electrical Systems*, n. 34, 2023.

NISE, N. S. *Control Systems Engineering*. 3rd. ed. USA: John Wiley & Sons, Inc., 2000. ISBN 0471366013.

OGATA, K. *Discrete-Time Control Systems*. Australia, Sydney: Prentice Hall, 1987.

OGATA, K. *Engenharia de controle moderno*. 5. ed. [S.l.]: Pearson Education do Brasil, 2011.

OKITA, W.; MOURA, L. *Simulação Númerica do Desempenho Aerodinâmico de Aerogeradores de Eixo Horizontal*. Tese (Doutorado), 01 2017.

OST, A. *Modelagem Matemática do Conjunto ESC-Motor-Hélice de um VANT Utilizando Identificação de Sistemas*. Dissertação (Mestrado) — Universidade Regional do Noroeste do Estado do Rio Grande do Sul, 2015.

PELKE, E.; REIMBOLD, M. M. P.; SILVA, J. V. D. C.; SAUER, C. Plataforma experimental para mediÇÃo do empuxo do sistema de propulsÃo de veÍculos multirrotores1 experimental platform for the measurement of the thrust the propulsion system of multirotor vehicles. In: . [S.l.: s.n.], 2017.

ROCHA, J. V. B. da; FOYO, P. M. G. del. *Relatório de Monitoria da disciplina ME472 Laboratório de Automação e Controle, 2022.1*. [S.l.], 2022.

RUBAAI, A.; CASTRO-SITIRICHE, M.; OFOLI, A. Design and implementation of parallel fuzzy pid controller for high-performance brushless motor drives: An integrated environment for rapid control prototyping. *Industry Applications, IEEE Transactions on*, v. 44, p. 1090 – 1098, 08 2008.

SALEEM, O.; RIZWAN, M.; ZEB, A.; ALI, A.; SALEEM, M. Online adaptive pid tracking control of an aero-pendulum using pso-scaled fuzzy gain adjustment mechanism. *Soft Computing*, v. 24, 07 2020.

SANTOS, M. P. de C.; FOYO, P. M. G. del. *Relatório de Monitoria da disciplina ME472 Laboratório de Automação e Controle, 2022.2*. [S.l.], 2023.

SCIMAGO, I. R. *University Rankings*. 2024.

SEBORG, D. E.; MELLICHAMP, D. A.; EDGAR, T. F. *Process Dynamics and Control*. Third. John Wiley & Sons, 2011. (Wylie Series in Chemical Engineering). ISBN 9780470646106. Disponível em: <http://www.worldcat.org/isbn/9780470646106>.

SHANNON, C. Communication in the presence of noise. *Proceedings of the IRE*, Institute of Electrical and Electronics Engineers (IEEE), v. 37, n. 1, p. 10–21, jan 1949. Disponível em: <https://doi.org/10.1109/jrproc.1949.232969>.

SOUZA, T. F. F. de; FOYO pedro M. G. del. *Relatório de Monitoria da disciplina ME472 Laboratório de Automação e Controle, 2021.2*. [S.l.], 2022.

STELLA, F. S. *Modelagem e Controle de Motor Sem escovas Utilizando Filtro Estendido de Kalman para Estimação da Velocidade e Posição*. [S.l.], 2022.

TAVARES, R. T. *Projeto e fabricação de Bancada para Medição de Tração de Grupos Motopropulsores de pequeno Porte*. [S.l.], 2022.

VASCONCELOS, J. R. C.; GONZÁLEZ, E. M. A.; FOYO, P. M. G. del. Design and control of a flywheel inverted pendulum system. In: ABCM. *Congresso Brasileiro de Automática*. [S.l.], 2018. v. 1.

APÊNDICE A – CÓDIGO DO ARDUINO

```
//#define SERVO_OUTPUT
#include <Servo.h>
#define SERVO_OUTPUT 5
#ifdef SERVO_OUTPUT
// Brushless motor
Servo brushless;
#endif

//time variable
unsigned long previousMillis = 0; // stores the last time LED was updated
const long interval = 1000; // interval at which to blink (milliseconds)

// PID variables
#define outMax 255.0
#define outMin 0.0

double Setpoint, Input, Output, Offset;
double Kp=1.0, Ki=0.05, Kd=0.25;
double Integral = 0.0;
double lastInput = 0.0;
double lastInputI = 0.0;
int vin,vsp,someInt,vsin;
float Ta = 10;
float fTa = 10/1000;

double yAnt = 0.0; // y[n-1]
double eAnt = 0.0; // e[n-1]
double fT1;
double fT2;
double kyAnt; // (T2/(T2 + Ta))
double ke;   // (Kp*(T1 + Ta)/(T2 + Ta))
double keAnt; // (Kp*T1/(T2+Ta))



double PIDCompute(){

  double error = Setpoint - Input;
```

```
  double proportional = Kp * error;

  Integral += lastInputI+Ki*Ta/1000*(error+eAnt)/2;
  if(Integral > outMax)
    Integral = outMax;
  else if(Integral < outMin)
    Integral = outMin;

  double derivativo = Kd*(vin - yAnt)/(Ta/1000);

  double PIDOut = proportional + Integral + derivativo + Offset;
  if(PIDOut > outMax){
    PIDOut = outMax;
    Integral = outMax-proportional;
  }
  else if(PIDOut < outMin){
    PIDOut = outMin;
  }
  lastInputI = Integral;
  lastInput = Input;
  eAnt = error;
  yAnt = vin;
  return PIDOut;
}

///////// Lead Lag variables and computation ///////

// Using implementation described in (Krikelis, Fassois, 1984)
// y[n] = (T2/(T2 + Ta))*y[n-1] + (Kp*(T1 + Ta)/(T2 + Ta))*e[n] - (Kp*T1/(T2+Ta)
    )*e[n-1]

// Many variables already defined for PID
// #define outMax 255.0
// #define outMin 0.0
// double Setpoint, Input, Output;
//double Kp=1.0, Ki=0.05, Kd=0.25;


double leadLagCompute(){
  double error = Setpoint - Input;
  double leadLag = kyAnt*yAnt + ke*error - keAnt*eAnt + Offset;
```

```
  if(leadLag > outMax)
    leadLag = outMax;
  else if(leadLag < outMin)
    leadLag = outMin;
  yAnt = leadLag;
  eAnt = error;
  return leadLag;
}

//#define PI 3.141
#define SerialRate 115200


#define pin_vout 5
#define pin_vin A5
#define pin_sp A4
#define pin_interrupt 3
#define pin_malha 8

byte input_buffer[16];

enum estado_t {parado, rodando, erro}estado;
enum experiment_t {step, senoidal, prbs, pid, compensator}experiment;

int V1 = 100;
int V2 = 250;
int V0 = 175;
int sinFreq;
int sinAmp;
int ang;
double timesin;
float b = 1;
float a = 0.3539;

int T0 = 50;
int T1 = 200;
int Tmin = 10;
int Tmax = 50;

int expNumber = 0;
bool Interrupted = 0;
```

```
int malha = 1;
int prbsPulseCounter;
int prbsStep = random(0,2);
int currentState = LOW;
int previousState = LOW;
int var1,curr_enc,init_enc;
char vout;

unsigned long int now, before, startexp;

void setup() {
  pinMode(13,OUTPUT);
  pinMode(pin_malha,INPUT);
  pinMode(pin_sp,INPUT);
  pinMode(pin_vin,INPUT);
  pinMode(pin_interrupt, INPUT);
  attachInterrupt(digitalPinToInterrupt(pin_interrupt), stop, RISING);
  // put your setup code here, to run once:
  Serial.begin(SerialRate);
  Serial.setTimeout(200);
  Serial.readBytesUntil(byte(55),input_buffer,16);
  estado = parado;
  experiment = step;
  before = millis();
  #ifdef SERVO_OUTPUT
  // brushless motor controlled by pin_vout through an ESC
  brushless.attach(SERVO_OUTPUT);
  brushless.write(0); //initialize the signal to 1000
  #endif
  #ifndef SERVO_OUTPUT
  pinMode(pin_vout,OUTPUT);
  #endif
  curr_enc = analogRead(pin_vin);
  init_enc = curr_enc;
  b = 360 + 22 + 360.0*curr_enc/1024;
}

void loop() {
  // put your main code here, to run repeatedly:
  switch(estado){
  case parado:
```

```
if(Interrupted == 1){
  break;
}
curr_enc = map(analogRead(pin_sp),0,1023,0,180);
brushless.write(curr_enc);
curr_enc = analogRead(pin_vin);
if (curr_enc <= init_enc+2){
  curr_enc = curr_enc +1024;
}
ang = b - 360.0*curr_enc/1024;

delay(10);
currentState = digitalRead(8);
if (currentState == HIGH && previousState == LOW) {
  previousState = currentState;
  Serial.print("SM:\t"); //status malha fechada
  Serial.println(1);
}else if (currentState == LOW && previousState == HIGH){
  previousState = currentState;
  Serial.print("SM:\t"); //status malha aberta
  Serial.println(0);
}



digitalWrite(13,0);
if (Serial.available()){
  // digitalWrite(13,1);
  Serial.readBytesUntil(byte(55),input_buffer,16);
  // Serial.println(input_buffer[0]);
  switch(input_buffer[0]){
    case 'r': // Run the experiment
    case 'R':
      Serial.println("Rodando7");
      startexp = millis();
      delay(10);
      setExperiment();
      break;
    case 'v': // set the voltages
    case 'V':
      // V0 is the output value that the experiment starts
      V0 = int(input_buffer[1]) * 256 + int(input_buffer[2]);
```

```
  // V1 is the step value for the step or one of the values for the PRBS
  // It also serves as the offset of the PID experiment
  V1 = int(input_buffer[3]) * 256 + int(input_buffer[4]);
  // V2 is the other value for the PRBS
  V2  = int(input_buffer[5]) * 256 + int(input_buffer[6]);
  Serial.print("V0, V1, V2:\t");
  Serial.print(V0);
  Serial.print('\t');
  Serial.print(V1);
  Serial.print('\t');
  Serial.println(V2);
break;
  case 't': // Set the general time constants
  case 'T':
  //           ---------------
  // __________/
  // 0        T0           T1
      T0 = int(input_buffer[1]) * 256 + int(input_buffer[2]);
      T1 = int(input_buffer[3]) * 256 + int(input_buffer[4]);
      Serial.print("T0, T1:\t");
      Serial.print(T0);
      Serial.print('\t');
      Serial.println(T1);
  break;
case 'a': // Set the time constants for the PRBS signal
case 'A':
  // Ta is the sampling period
  Ta = int(input_buffer[1]) * 256 + int(input_buffer[2]);
      Serial.print("Ta:\t");
      Serial.print(Ta);
  break;
case 'c': // Set the variables for Sin signal
case 'C':
  // Start the motor
  sinFreq = int(input_buffer[1]) * 256 + int(input_buffer[2]);
  sinAmp = int(input_buffer[3]) * 256 + int(input_buffer[4]);
  Serial.print("Freq, Amp:\t");
  Serial.print(sinFreq);
  Serial.print('\t');
  Serial.println(sinAmp);
  break;
```

```
case 'b': // Set the time constants for the PRBS signal
case 'B':
  // Tmin is the minimum length of a PRBS pulse
  Tmin = int(input_buffer[1]) * 256 + int(input_buffer[2]);
  // Tmax is the maximum length of a PRBS pulse
  Tmax = int(input_buffer[3]) * 256 + int(input_buffer[4]);
      Serial.print("Tmin, Tmax (PRBS):\t");
      Serial.print(Tmin);
      Serial.print('\t');
      Serial.println(Tmax);
  break;
case 'm': // define the experiment type
case 'M':
  switch(input_buffer[1]){
    case 0:
    experiment = step;
    Serial.println("Experiment set to\tStep response");
    break;
    case 1:
      experiment = prbs;
    Serial.println("Experiment set to\tPRBS response");
    break;
    case 2:
    experiment = senoidal;
    Serial.println("Experiment set to\tSenoidal");
    break;
    case 3:
    experiment = pid;
    Serial.println("Experiment set to\tPID control");
    break;
    case 4:
      experiment = compensator;
    Serial.println("Experiment set to\tcompensator control");
    break;
    default:
    experiment = step;
  }
  break;
case 'q': // Set the PID parameters
case 'Q':
  byte * bKp = (byte *) &Kp;
```

```
  bKp[0] = input_buffer[1];
  bKp[1] = input_buffer[2];
  bKp[2] = input_buffer[3];
  bKp[3] = input_buffer[4];
  byte * bKi = (byte *) &Ki;
  bKi[0] = input_buffer[5];
  bKi[1] = input_buffer[6];
  bKi[2] = input_buffer[7];
  bKi[3] = input_buffer[8];
  byte * bKd = (byte *) &Kd;
  bKd[0] = input_buffer[9];
  bKd[1] = input_buffer[10];
  bKd[2] = input_buffer[11];
  bKd[3] = input_buffer[12];
  Serial.print("Kp, Ki, Kd:\t");
  Serial.print(Kp);
  Serial.print('\t');
  Serial.print(Ki);
  Serial.print('\t');
  Serial.println(Kd);
  break;
case 'l': // Set the lead/lag parameters
case 'L':
  byte * bKll = (byte *) &Kp;
  bKll[0] = input_buffer[1];
  bKll[1] = input_buffer[2];
  bKll[2] = input_buffer[3];
  bKll[3] = input_buffer[4];
  byte * bT1 = (byte *) &fT1;
  bT1[0] = input_buffer[5];
  bT1[1] = input_buffer[6];
  bT1[2] = input_buffer[7];
  bT1[3] = input_buffer[8];
  byte * bT2 = (byte *) &fT2;
  bT2[0] = input_buffer[9];
  bT2[1] = input_buffer[10];
  bT2[2] = input_buffer[11];
  bT2[3] = input_buffer[12];
  // Adjust the leadLag specific constants
  float fTa = float(Ta)/1000;
  kyAnt = (fT2/(fT2 + fTa));
```

```
        ke = (Kp*(fT1 + fTa)/(fT2 + fTa));
        keAnt = (Kp*fT1/(fT2+fTa));

            Serial.print("Kp, Tc1, Tc2:\t");
            Serial.print(Kp);
            Serial.print('\t');
            Serial.print(fT1);
            Serial.print('\t');
            Serial.println(fT2);
        break;
      default: break;
    }
  }
  break;
case rodando:
  if(Interrupted == 1){
    break;
  }
  digitalWrite(13,1);
  now = millis();
  if((now-before)>=Ta){
    switch(experiment){
      case step:
        if(malha==1){
          //vout = expNumber<T0?V0:V1;
          if ((before-startexp)/10<T0){
            vout = map(V0,0,255,0,180);
          }
          else{
            vout = map(V1,0,255,0,180);
          }
          #ifndef SERVO_OUTPUT
          analogWrite(pin_vout,vout);
          #endif
          #ifdef SERVO_OUTPUT
          brushless.write(vout);
          //brushless.write(map(vout,0,255,0,180));
          #endif
        }else{
          var1 = map(analogRead(pin_sp),0,1023,0,180);
          brushless.write(var1);
```

```
    vout = var1;
  }

  vin = analogRead(pin_vin);
  ang = b - 360.0*vin/1017;
  break;
case senoidal:
  //vout = expNumber<T0?V0:V1;
  now = millis();
  //timesin = (now-startexp)/1000;
  if ((before-startexp)/10<T0){
    vout = map(V0,0,255,0,180);
    brushless.write(vout);
  }else{
  //vsin = sin((timesin/sinFreq)*2*3.14)*3*sinAmp+V0;
  vsin = V0 + sin(timesin/100*2*PI*sinFreq/2)*sinAmp;

  vout = map(long(vsin),0,255,0,180);

  }
  #ifndef SERVO_OUTPUT
  analogWrite(pin_vout,vout);
  #endif
  #ifdef SERVO_OUTPUT
  brushless.write(vout);
  #endif
  vin = analogRead(pin_vin);
  ang = b - 360.0*vin/1017;
  //vout = sin(timesin/100*2*PI)*30+40;
  break;
case prbs:
  if((before-startexp)/10<T0){
    vout = map(V0,0,255,0,180);
  } else {
    if(prbsPulseCounter<=expNumber){
      prbsPulseCounter = expNumber + random(Tmin,Tmax);
      prbsStep = !prbsStep;
    }
    if(prbsStep){
      vout=map(V1,0,255,0,180);
    }else{
```

```
      vout=map(V2,0,255,0,180);
    }
    //vout = prbsStep?V1:V2;
  }
  #ifndef SERVO_OUTPUT
  analogWrite(pin_vout,vout);
  #endif
  #ifdef SERVO_OUTPUT
  //brushless.write(map(vout,0,255,0,180));
  brushless.write(vout);
  #endif
  vin = analogRead(pin_vin);
  ang = b - 360.0*vin/1017;
  break;
case pid:
  //Kp = 1;
  vsp = analogRead(pin_sp);
  vsp = map(vsp,0,1023,0,360);
  Setpoint = (double)vsp;
  curr_enc = analogRead(pin_vin);
  if (curr_enc <= init_enc+2){
    curr_enc = curr_enc +1024;
  }
  ang = b - 360.0*curr_enc/1024;
  Input = (double)ang ;
  Offset = (double)V1;
  Output = PIDCompute();
  someInt = (int)Output;
  vout = map(someInt,0,255,0,180);
  #ifndef SERVO_OUTPUT
  analogWrite(pin_vout,vout);
  #endif
  #ifdef SERVO_OUTPUT
  brushless.write(vout);
  #endif
  break;
case compensator:
  vsp = analogRead(pin_sp);
  vsp = map(vsp,0,1023,0,360);
  Setpoint = (double)vsp;
  curr_enc = analogRead(pin_vin);
```

```
        if (curr_enc <= init_enc+2){
          curr_enc = curr_enc +1024;
        }
        ang = b - 360.0*curr_enc/1024;
        Input = (double)ang;
        Offset = (double)V1;
        Output = leadLagCompute();
        someInt = (int)Output;
        vout = map(someInt,0,255,0,180);
        #ifndef SERVO_OUTPUT
        analogWrite(pin_vout,vout);
        #endif
        #ifdef SERVO_OUTPUT
        brushless.write(map(vout,0,255,0,180));
        #endif
        break;
      default: break;
    }
    curr_enc = analogRead(pin_vin);
    if (curr_enc <= init_enc+2){
      curr_enc = curr_enc +1024;
    }
    ang = b - 360.0*curr_enc/1024;
    Serial.print('E');
    Serial.write(byte(ang>>8));
    Serial.write(byte(ang%256));
    Serial.write(byte(vout));
    Serial.write(byte(vsp>>8));
    Serial.write(byte(vsp%256));
    Serial.write(55);
    Serial.println();
    before +=Ta;
    timesin = (now-startexp)/10;
    if(++expNumber>T1){
      estado = parado;
      brushless.write(0);
      analogWrite(pin_vout,0); // return to 0
    }
  }
  break;
case erro:
```

```
  digitalWrite(13,1);
  delay(50);
  digitalWrite(13,0);
  delay(50);
  }
}

void stop() {
  Serial.println("Stop");
  brushless.write(0);
  Interrupted = 1;
}

void setExperiment(){
  expNumber = 0;
  prbsPulseCounter = 0;
  estado = rodando;
  before = millis();
}
// AMDG
```

APÊNDICE B – CÓDIGO DA INTERFACE

```
import tkinter as tk
from tkinter import ttk
from tkinter import *
from tkinter.filedialog import asksaveasfilename
import matplotlib
matplotlib.use("TkAgg")
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
import matplotlib.animation as animation
import csv
import time
import communication

# Global variables for the measurement
measurementNumber = -1 # index
measurementArray = [] # index array
inputArray = [] # input array
outputArray = [] # output array
spArray = [] # setpoint array

textinstrucoes = 'Este aplicativo contm as funcionalidades para a realizao dos
    experimentos das prticas de LAC.\n\nPara realizar os experimentos  necessrio
     inicializar o motor. Siga o procedimento abaixo para inicializao\n\n1 -
    Clique no boto abaixo para inicializar a varivel de posio do motor\n2 -
    Ligue a fonte de alimentao do motor\n3 - Clique no boto inicializar motor'

global tVone
global tTa
global tVzero
global tVtwo
global tTone
global tTtwo
global tPmin
global tPmax
global tKp
global tKi
global tKd
global tTc1
```

```
global tTc2
global tFone
global tAone
global ax
global plotCanvas
global fig

expTypeNumber = 0



expTypeNumber = 0 # index for the experiment type
# the experiment type can be:
# experimentTypes = ['Step', 'PRBS', 'PID', 'Compensador', 'Step-PID', 'Step-
    Compensador', 'PRBS-PID', 'PRBS-Compensador']
## so far only the first three are defined
experimentTypes = ['Step', 'PRBS', 'Senoidal', 'PID', 'Compensador']

ard = None # later will be assigned to an SerialPort object, from communication.
    py
#                       can't run an experiment while ard is None


root = tk.Tk()
root.configure(background='#203764')
root.geometry('500x500')
root.title('expGui')

#tVonetext= tk.StringVar()
#tVzerotext= tk.StringVar()
#tVtwotext= tk.StringVar()
#tTonetext= tk.StringVar()
#tTtwotext= tk.StringVar()
#tPmintext= tk.StringVar()
#tPmaxtext= tk.StringVar()
#tKptext= tk.StringVar()
#tKitext= tk.StringVar()
#tKdtext= tk.StringVar()
#tTc1text= tk.StringVar()
#tTc2text= tk.StringVar()
```

```
def setConnButton(a):
        if (comboPort.get() != "None"): # Can connect only if a port is selected
                connButton.configure(state='active')
        else:
                connButton.configure(state='disabled')

# Function to handle the connection with the arduino (or other platform)
    performing the experiment
def connectArduino():
        global ard
        if ard is None: # if there is no connection yet
                ard = communication.SerialPort(comboPort.get()) # get the selected
                     port
                ard.connect() # and connect

                # change the button to disconnect
                connButton.configure(text="Disconnect")
                # and enable the runButton
                conection_status_lb.configure(font=("Cambria",10),text='Status da
                    conexao: Conectado com '+ comboPort.get())
                runButton.configure(state="active")

        else: # If there is a ard object
                if ard.connected: # and it is already connected
                        # then disconnect
                        ard.disconnect()
                        # change the button to connect
                        connButton.configure(text="Connect")
                        # and disable the runButton
                        conection_status_lb.configure(font=("Cambria",10),text='
                            Status da conexao: Desconectado')
                        runButton.configure(state="disabled")

                else: # but if ard is not None but is not connected
                        ard = communication.SerialPort(comboPort.get()) # get the
                            selected port
                        ard.connect() # and connect

                        # change the button to disconnect
                        connButton.configure(text="Disconnect")
```

```
                    # and enable the runButton
                    conection_status_lb.configure(font=("Cambria",10),text='
                        Status da conexao: Conectado com '+ comboPort.get())
                    runButton.configure(state="active")
        # wait half a second to guarantee the connection and the arduino reset
        time.sleep(0.5)

# Execute the experiment
# The experiment type an all parameters are obtained from the GUI inputs - to
    change later
def runExperiment(_ard):

    global measurementArray,inputArray,outputArray,spArray,measurementNumber,
        tTtwo, tTa, tTone, ax,tVtwo
    print("tTone:", tTone)
    ax.clear() # clear the previous plot
    ax.plot([],[],'b')
    ax.plot([],[],'r')
    ax.plot([],[],'g')
    ax.set_xlabel("Measurement number")
    ax.set_ylabel("Input, Output and setPoint")
    ax.set_xlim([0,int(tTtwo.get())/(int(tTa.get())/10)]) # Ttwo defines the
        number of measurements
    ax.set_ylim([0,180])

    _ard.ser.read_all() # flush the serial port
    time.sleep(0.1) # wait for the serial connection to initialize

        # initialize the measurement variables
    measurementNumber = 0
    measurementArray = []
    inputArray = []
    outputArray = []
    spArray = []
    ard = _ard

    # Configure the experiment
    ard.setTA(int(tTa.get()))
    # experiment = comboType.get()
    ard.setType(expTypeNumber) # expTypeNumber is set when the type is selected
    if ((expTypeNumber==1)): # if is PRBS
```

```
                # then set the minimum and maximum step times
            ard.setPRBSTimes(int(tPmin.get()), int(tPmax.get()))
            # set the times
        ard.setTimes(int(tTone.get()), int(tTtwo.get()))
        if ((expTypeNumber==2)): # if is Senoidal
                # then set the minimum and maximum step times
            ard.setSenoidal(int(tFone.get()), int(tAone.get()))

        if ((expTypeNumber==3)): # if is PID
                # set the PID constants
            ard.setPID(float(tKp.get()),float(tKi.get()),float(tKd.get()))
        elif ((expTypeNumber==4)): # if is compensator:
            ard.setLeadLag(float(tKp.get()),float(tTc1.get()),float(tTc2.get()))
            # in all cases set the voltages
        ard.setVoltages(int(tVzero.get()), int(tVone.get()), int(tVtwo.get()))
            # get one measurement for sanity - should return 0
        ard.getMeasure()
            # and run the experiment

        ani.event_source.start()
        ard.run()

def switch(indicator_lb, page):
    global measurementArray,outputArray,inputArray,spArray
    for child in options_fm.winfo_children():
        if isinstance(child,tk.Label):
            child['bg']='SystemButtonFace'

    indicator_lb['bg']='#ffffff'

    for fm in main_fm.winfo_children():
        fm.destroy()
        root.update()

    measurementArray = [] # index array
    inputArray = [] # input array
    outputArray = [] # output array
    spArray = [] # setpoint array
    page()

options_fm=tk.Frame(root,background="#203764")
```

```
home_btn = tk.Button(options_fm,text='Home',font=('Cambria',9),bd = 0,fg='#
    ffffff',background='#203764',activeforeground='#0097e8'
                    ,command=lambda:switch(indicator_lb=home_indicator_lb,page=
                        home_page))
home_btn.place(x=0,y=0,width=83)
home_indicator_lb = tk.Label(options_fm,bg='#ffffff')
home_indicator_lb.place(x=5,y=18,width=73,height=2)


Degrau_btn = tk.Button(options_fm,text='Degrau',name='0',font=('Cambria',9),bd =
     0,fg='#ffffff',background='#203764',activeforeground='#0097e8'
                      ,command=lambda:switch(indicator_lb=Degrau_indicator_lb,
                          page=Degrau_page))
Degrau_btn.place(x=83.3,y=0,width=83)
Degrau_indicator_lb = tk.Label(options_fm)
Degrau_indicator_lb.place(x=88.3,y=18,width=73,height=2)

PRBS_btn = tk.Button(options_fm,text='PRBS',name='1',font=('Cambria',9),bd = 0,
    fg='#ffffff',background='#203764',activeforeground='#0097e8'
                    ,command=lambda:switch(indicator_lb=PRBS_indicator_lb,page=
                        PRBS_page))
PRBS_btn.place(x=83.3*2,y=0,width=83)
PRBS_indicator_lb = tk.Label(options_fm)
PRBS_indicator_lb.place(x=83.3*2+5,y=18,width=73,height=2)

Senoidal_btn = tk.Button(options_fm,text='Senoidal',name='2',font=('Cambria',9),
    bd = 0,fg='#ffffff',background='#203764',activeforeground='#0097e8'
                      ,command=lambda:switch(indicator_lb=Senoidal_indicator_lb,
                          page=Senoidal_page))
Senoidal_btn.place(x=83.3*3,y=0,width=83)
Senoidal_indicator_lb = tk.Label(options_fm)
Senoidal_indicator_lb.place(x=83.3*3+5,y=18,width=73,height=2)

PID_btn = tk.Button(options_fm,text='PID',name='3',font=('Cambria',9),bd = 0,fg
    ='#ffffff',background='#203764',activeforeground='#0097e8'
                    ,command=lambda:switch(indicator_lb=PID_indicator_lb,page=
                        PID_page))
PID_btn.place(x=83.3*4,y=0,width=83)
PID_indicator_lb = tk.Label(options_fm)
PID_indicator_lb.place(x=83.3*4+5,y=18,width=73,height=2)
```

```
Compensador_btn = tk.Button(options_fm,text='Compensador',name='4',font=('
    Cambria',9),bd = 0,fg='#ffffff',background='#203764',activeforeground='#0097
    e8'
                        ,command=lambda:switch(indicator_lb=
                            Compensador_indicator_lb,page=Compensador_page))
Compensador_btn.place(x=83.3*5,y=0,width=83)
Compensador_indicator_lb = tk.Label(options_fm)
Compensador_indicator_lb.place(x=83.3*5+5,y=18,width=73,height=2)

options_fm.pack(pady=5)

options_fm.pack_propagate(False)
options_fm.configure(width=500, height =20)

def home_page():
    home_page_fm = tk.Frame(main_fm,background='white')

    home_page_lb = tk.Label(home_page_fm, text='Instrues',
                        font=('Cambria',25),justify='left',fg='#0097e8',
                            background='white')
    home_page_lb.pack(fill='both')
    home_page_inst = tk.Label(home_page_fm, text=textinstrucoes,
                        font=('Cambria',10),anchor='w',fg='#0097e8',justify='
                            left',wraplength=500,background='white')
    home_page_inst.pack(fill='both')
    home_page_fm.pack(fill='both')
    home_page_fm.configure(width=400, height =400)
    #home_page_startmotorvar=tk.Button(home_page_fm,text='Inicializar Posio',
        command=lambda: ard.setPos1())
    #home_page_startmotorvar.place(x=20,y=300)
    #home_page_startmotor=tk.Button(home_page_fm,text='Inicializar Motor',
        command=lambda: ard.setPos2())
    #home_page_startmotor.place(x=20,y=330)


def validateVzero(P):
        if(P.isNumeric()):
            if int(P)>=0 and int(P)<=255 :
                return True
            else:
```

```
                return False
        else:
            # root.submit.config(state=(NORMAL if P else DISABLED))
            return False

def Degrau_page():

    global tVone,tVzero,tTone,tTtwo,expTypeNumber,tTa,tTtwo, ani, tVtwo
    Degrau_page_fm = tk.Frame(main_fm,background="#D6DCE4")
    expTypeNumber = 0
    #Degrau_page_lb = tk.Label(Degrau_page_fm, text='Degrau Page',
    #                      font=('Arial',25),fg='#0097e8')
    #Degrau_page_lb.pack(pady=80)

    # Ta
    Label(Degrau_page_fm, text='Ta', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=12)
    tTa=Entry(Degrau_page_fm, width = 10)
    tTa.insert(0,str(10))
    tTa.place(x=30, y=12)

    # Vzero
    lVzero = Label(Degrau_page_fm, text='V0', fg='#203764',background="#D6DCE4",
         font=('Cambria', 10, 'bold')).place(x=6, y=36)
    vcmdVzero = Degrau_page_fm.register(validateVzero)
    tVzero = Entry(Degrau_page_fm, width = 6)
    tVzero.insert(0,str(0))
    tVzero.place(x=30, y=36)
        #config_fm.pack(fill=BOTH,side=LEFT)
        #config_fm.configure(width=100, height =400)
    Degrau_page_fm.pack(fill=tk.BOTH,side=LEFT)
    Degrau_page_fm.configure(width=100, height =400)
    # V1
    Label(Degrau_page_fm, text='V1', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=60)
    tVone=Entry(Degrau_page_fm, width = 10)
    tVone.insert(0,str(100))
    tVone.place(x=30, y=60)

    # V2
    #Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
```

```
        place(x=0, y=0)
    tVtwo=Entry(Degrau_page_fm, width = 0)
    tVtwo.insert(0,str(0))



    # T1
    Label(Degrau_page_fm, text='T1', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=84)
    tTone=Entry(Degrau_page_fm,width=6)
    tTone.insert(0,str(100))
    tTone.place(x=30, y=84)
    #T2
    Label(Degrau_page_fm, text='T2', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=108)
    tTtwo=Entry(Degrau_page_fm,width=6)
    tTtwo.insert(0,str(200))
    tTtwo.place(x=30, y=108)

    saveButton = Button(Degrau_page_fm, text='Save',fg='#203764', bg='#ffffff',
        font=('Cambria', 10, 'bold'), command=lambda: saveMeasurement())
    saveButton.place(x=30, y=300)

    Canvas_page()

    ani = animation.FuncAnimation(fig,figAnimate, interval=50)




def PRBS_page():
    global tVone,tVzero,tTone,tTtwo,tTa,tPmin,tPmax,tVtwo, fig, ani
    global expTypeNumber
    expTypeNumber = 1

    PRBS_page_fm = tk.Frame(main_fm,background="#D6DCE4")

    print(expTypeNumber)
    #PRBS_page_lb = tk.Label(PRBS_page_fm, text='PRBS Page',
    #                      font=('Arial',25),fg='#0097e8')
    #PRBS_page_lb.pack(pady=80)
```

```
# Ta
Label(PRBS_page_fm, text='Ta', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=12)
tTa=Entry(PRBS_page_fm, width = 10)
tTa.insert(0,str(10))
tTa.place(x=30, y=12)

# Vzero
lVzero = Label(PRBS_page_fm, text='V0', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=36)
vcmdVzero = PRBS_page_fm.register(validateVzero)
tVzero = Entry(PRBS_page_fm, width = 6)
tVzero.insert(0,str(0))
tVzero.place(x=30, y=36)
    #config_fm.pack(fill=BOTH,side=LEFT)
    #config_fm.configure(width=100, height =400)
PRBS_page_fm.pack(fill=tk.BOTH,side=LEFT)
PRBS_page_fm.configure(width=100, height =400)
# V1
Label(PRBS_page_fm, text='V1', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=60)
tVone=Entry(PRBS_page_fm, width = 10)
tVone.insert(0,str(100))
tVone.place(x=30, y=60)

# V2
Label(PRBS_page_fm, text='V2', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=84)
tVtwo=Entry(PRBS_page_fm, width = 10)
tVtwo.insert(0,str(100))
tVtwo.place(x=30, y=84)

# T1
Label(PRBS_page_fm, text='T1', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=108)
tTone=Entry(PRBS_page_fm,width=6)
tTone.insert(0,str(100))
tTone.place(x=30, y=108)
#T2
Label(PRBS_page_fm, text='T2', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=132)
```

```
    tTtwo=Entry(PRBS_page_fm,width=6)
    tTtwo.insert(0,str(200))
    tTtwo.place(x=30, y=132)

    # Pmin
    Label(PRBS_page_fm, text='Pm', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=156)
    tPmin=Entry(PRBS_page_fm,width=6)
    tPmin.insert(0,str(10))
    tPmin.place(x=30, y=156)

    # Pmax
    Label(PRBS_page_fm, text='PM', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=180)
    tPmax=Entry(PRBS_page_fm,width=6)
    tPmax.insert(0,str(100))
    tPmax.place(x=30, y=180)

    #fig = Figure(figsize=(4, 4), dpi=100)
    #ax = fig.add_subplot(111)
    #plotCanvas = FigureCanvasTkAgg(fig, master=main_fm)
    # plotCanvas.get_tk_widget().pack()
    #plotCanvas.get_tk_widget().pack(side=tk.RIGHT)
    #plotCanvas.draw()
    saveButton = Button(PRBS_page_fm, text='Save',fg='#0097e8', bg='#ffffff',
        font=('arial', 10, 'normal'), command=lambda: saveMeasurement())
    saveButton.place(x=30, y=300)

    Canvas_page()

    ani = animation.FuncAnimation(fig,figAnimate, interval=50)

def Senoidal_page():
    Senoidal_page_fm = tk.Frame(main_fm,background="#D6DCE4")
    global expTypeNumber, tTtwo,tTa,tTone,tVzero,tVone, tFone, tAone, ani,tVtwo
    expTypeNumber = 2
    #Senoidal_page_lb = tk.Label(Senoidal_page_fm, text='Senoidal Page',
    #                      font=('Arial',25),fg='#0097e8')
    #Senoidal_page_lb.pack(pady=80)

    # Ta
```

```
Label(Senoidal_page_fm, text='Ta', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=12)
tTa=Entry(Senoidal_page_fm, width = 10)
tTa.insert(0,str(10))
tTa.place(x=30, y=12)

# Vzero
lVzero = Label(Senoidal_page_fm, text='V0', fg='#203764',background="#D6DCE4
    ", font=('Cambria', 10, 'bold')).place(x=6, y=36)
vcmdVzero = Senoidal_page_fm.register(validateVzero)
tVzero = Entry(Senoidal_page_fm, width = 6)
tVzero.insert(0,str(0))
tVzero.place(x=30, y=36)
    #config_fm.pack(fill=BOTH,side=LEFT)
    #config_fm.configure(width=100, height =400)
Senoidal_page_fm.pack(fill=tk.BOTH,side=LEFT)
Senoidal_page_fm.configure(width=100, height =400)

# V1
#Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tVone=Entry(Senoidal_page_fm, width = 0)
tVone.insert(0,str(0))

# V2
#Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tVtwo=Entry(Senoidal_page_fm, width = 0)
tVtwo.insert(0,str(0))

# F1
Label(Senoidal_page_fm, text='F1', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=60)
tFone=Entry(Senoidal_page_fm, width = 10)
tFone.insert(0,str(100))
tFone.place(x=30, y=60)

# A1
Label(Senoidal_page_fm, text='A1', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=84)
tAone=Entry(Senoidal_page_fm, width = 10)
```

```
    tAone.insert(0,str(100))
    tAone.place(x=30, y=84)

    # T1
    Label(Senoidal_page_fm, text='T1', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=108)
    tTone=Entry(Senoidal_page_fm,width=6)
    tTone.insert(0,str(100))
    tTone.place(x=30, y=108)
    #T2
    Label(Senoidal_page_fm, text='T2', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=132)
    tTtwo=Entry(Senoidal_page_fm,width=6)
    tTtwo.insert(0,str(200))
    tTtwo.place(x=30, y=132)

    #fig = Figure(figsize=(4, 4), dpi=100)
    #ax = fig.add_subplot(111)
    #plotCanvas = FigureCanvasTkAgg(fig, master=main_fm)
    ## plotCanvas.get_tk_widget().pack()
    #plotCanvas.get_tk_widget().pack(side=tk.RIGHT)
    #plotCanvas.draw()

    saveButton = Button(Senoidal_page_fm, text='Save',fg='#0097e8', bg='#ffffff',
         font=('arial', 10, 'normal'), command=lambda: saveMeasurement())
    saveButton.place(x=30, y=300)

    Canvas_page()

    ani = animation.FuncAnimation(fig,figAnimate, interval=50)

def PID_page():
    PID_page_fm = tk.Frame(main_fm,background="#D6DCE4")
    global expTypeNumber, tTtwo,tTa,tTone, tKp, tKd, tKi,tVzero,tVone, ani,
        tVtwo, fig
    expTypeNumber = 3
    #PID_page_lb = tk.Label(PID_page_fm, text='PID Page',
    #                       font=('Arial',25),fg='#0097e8')
    #PID_page_lb.pack(pady=80)

    PID_page_fm.pack(fill=tk.BOTH,side=LEFT)
```

```
PID_page_fm.configure(width=100, height =400)

# Ta
Label(PID_page_fm, text='Ta', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=12)
tTa=Entry(PID_page_fm, width = 10)
tTa.insert(0,str(10))
tTa.place(x=30, y=12)

# V0
#Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tVzero=Entry(PID_page_fm, width = 0)
tVzero.insert(0,str(0))

# V1
Label(PID_page_fm, text='V1', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=36)
tVone=Entry(PID_page_fm, width = 10)
tVone.insert(0,str(100))
tVone.place(x=30, y=36)

#T1
#Label(PID_page_fm, text='T1', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tTone=Entry(PID_page_fm,width=0)
tTone.insert(0,str(0))

#T2
Label(PID_page_fm, text='T2', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=60)
tTtwo=Entry(PID_page_fm,width=6)
tTtwo.insert(0,str(200))
tTtwo.place(x=30, y=60)

# Kp
Label(PID_page_fm, text='Kp', fg='#203764',background="#D6DCE4", font=('
    Cambria', 10, 'bold')).place(x=6, y=84)
tKp=Entry(PID_page_fm,width=6)
tKp.insert(0,str(10))
tKp.place(x=30, y=84)
```

```
    # Ki
    Label(PID_page_fm, text='Ki', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=108)
    tKi=Entry(PID_page_fm,width=6)
    tKi.insert(0,str(100))
    tKi.place(x=30, y=108)

    # Kd
    Label(PID_page_fm, text='Kd', fg='#203764',background="#D6DCE4", font=('
        Cambria', 10, 'bold')).place(x=6, y=132)
    tKd=Entry(PID_page_fm,width=6)
    tKd.insert(0,str(100))
    tKd.place(x=30, y=132)

    # V2
    #Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
        place(x=0, y=0)
    tVtwo=Entry(PID_page_fm, width = 0)
    tVtwo.insert(0,str(0))

    #fig = Figure(figsize=(4, 4), dpi=100)
    #ax = fig.add_subplot(111)
    #plotCanvas = FigureCanvasTkAgg(fig, master=main_fm)
    ## plotCanvas.get_tk_widget().pack()
    #plotCanvas.get_tk_widget().pack(side=tk.RIGHT)
    #plotCanvas.draw()

    saveButton = Button(PID_page_fm, text='Save',fg='#0097e8', bg='#ffffff',
        font=('arial', 10, 'normal'), command=lambda: saveMeasurement())
    saveButton.place(x=30, y=300)

    Canvas_page()

    ani = animation.FuncAnimation(fig,figAnimate, interval=50)


def Compensador_page():
    Compensador_page_fm = tk.Frame(main_fm,background="#D6DCE4")
    global expTypeNumber, tTtwo,tTa,tTone, tKp, tKd, tKi,tVzero,tVone,tTc1,tTc2,
         ani, tVtwo
```

```
expTypeNumber = 4
#Compensador_page_lb = tk.Label(Compensador_page_fm, text='Compensador Page',

#                       font=('Arial',25),fg='#0097e8')
#Compensador_page_lb.pack(pady=80)

Compensador_page_fm.pack(fill=tk.BOTH,side=LEFT)
Compensador_page_fm.configure(width=100, height =400)

# Ta
Label(Compensador_page_fm, text='Ta', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=12)
tTa=Entry(Compensador_page_fm, width = 10)
tTa.insert(0,str(10))
tTa.place(x=30, y=12)

# V0
#Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tVzero=Entry(Compensador_page_fm, width = 0)
tVzero.insert(0,str(0))

# V1
Label(Compensador_page_fm, text='V1', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=36)
tVone=Entry(Compensador_page_fm, width = 10)
tVone.insert(0,str(100))
tVone.place(x=30, y=36)

#T1
#Label(PID_page_fm, text='T1', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tTone=Entry(Compensador_page_fm,width=0)
tTone.insert(0,str(0))

#T2
Label(Compensador_page_fm, text='T2', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=60)
tTtwo=Entry(Compensador_page_fm,width=6)
tTtwo.insert(0,str(200))
```

```
tTtwo.place(x=30, y=60)

# Kp
Label(Compensador_page_fm, text='Kp', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=84)
tKp=Entry(Compensador_page_fm,width=6)
tKp.insert(0,str(10))
tKp.place(x=30, y=84)

# Tc1
Label(Compensador_page_fm, text='Tc1', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=108)
tTc1=Entry(Compensador_page_fm,width=6)
tTc1.insert(0,str(100))
tTc1.place(x=30, y=108)

# Tc2
Label(Compensador_page_fm, text='Tc2', fg='#203764',background="#D6DCE4",
    font=('Cambria', 10, 'bold')).place(x=6, y=132)
tTc2=Entry(Compensador_page_fm,width=6)
tTc2.insert(0,str(100))
tTc2.place(x=30, y=132)

# V2
#Label(PID_page_fm, text='V0', fg='#0097e8', font=('arial', 0, 'normal')).
    place(x=0, y=0)
tVtwo=Entry(Compensador_page_fm, width = 0)
tVtwo.insert(0,str(0))
#fig = Figure(figsize=(4, 4), dpi=100)
#ax = fig.add_subplot(111)
#plotCanvas = FigureCanvasTkAgg(fig, master=main_fm)
## plotCanvas.get_tk_widget().pack()
#plotCanvas.get_tk_widget().pack(side=tk.RIGHT)
#plotCanvas.draw()

saveButton = Button(Compensador_page_fm, text='Save',fg='#0097e8', bg='#
    ffffff', font=('arial', 10, 'normal'), command=lambda: saveMeasurement())
saveButton.place(x=30, y=300)

Canvas_page()
```

```
    ani = animation.FuncAnimation(fig,figAnimate, interval=50)

# Main frame, Conection frame, Config frame
main_fm = tk.Frame(root)
main_fm.pack(fill=tk.BOTH,expand=True)

#config_fm=tk.Frame(main_fm,bg='red')
#config_fm.pack(fill=BOTH,side=LEFT)
#config_fm.configure(width=100, height =400)

conection_fm=tk.Frame(root,background="#203764")
conection_fm.pack(fill=BOTH)
conection_fm.configure(width=500, height =80)
conection_indicator_lb = tk.Label(conection_fm,bg='#000000')
conection_indicator_lb.place(x=0,y=0,width=500,height=2)

conection_status_lb = tk.Label(conection_fm, text='Status de conexo:
    Desconectado',
                        font=('Arial',8),fg='#ffffff',background="#203764")
conection_status_lb.pack(pady=2,side=LEFT)

runButton = Button(conection_fm, text='Run',fg='#0097e8', bg='#ffffff', font=('
    arial', 10, 'normal'), command=lambda: runExperiment(ard))
runButton.pack(padx=5,pady=5,side=RIGHT)
runButton.configure(state="disabled")

# Connect button

connButton = Button(conection_fm, text='Connect', bg='#EEDFCC', font=('arial',
    10, 'normal'), command=connectArduino)
connButton.pack(padx=5,pady=5,side=RIGHT)
connButton.configure(state=DISABLED)

# Arduino config
#Label(conection_fm, text='Arduino Serial port', fg='#0097e8', font=('arial', 8,
     'normal')).pack(pady=2,side=LEFT)
import serial.tools.list_ports
ports = ["None"]
for port in serial.tools.list_ports.comports():
    ports.append(port.name)
comboPort= ttk.Combobox(conection_fm, values=ports, font=('arial', 10, 'normal')
```

```
    , width=10)
comboPort.pack(padx=5,pady=5,side=RIGHT)
comboPort.current(0)
comboPort.bind("<<ComboboxSelected>>",setConnButton)

def Canvas_page():
    global ax, fig
    global plotCanvas
    fig = Figure(figsize=(4, 4), dpi=100)
    ax = fig.add_subplot(111)
    plotCanvas = FigureCanvasTkAgg(fig, master=main_fm)
    #plotCanvas.get_tk_widget().pack()
    plotCanvas.get_tk_widget().pack(side=tk.RIGHT)
    plotCanvas.draw()


def figAnimate(i):
    global measurementNumber, ard,fig, ax, plotCanvas
    global measurementArray,inputArray, outputArray, spArray
        # print("{}.{}".format(type(measurementNumber),type(figXMax)))
    if ard is None:
        ax.clear()
    else:
        figXMax = int(tTtwo.get())/(int(tTa.get())/10.0)
        if ard.connected and (measurementNumber<figXMax):
            while(ard.ser.in_waiting>0):
                measurement = ard.getMeasure()
                if isinstance(measurement, tuple):
                    if measurement[0] == 'meas':
                        measurementArray.append(measurementNumber)
                        inputArray.append(measurement[1])
                        outputArray.append(measurement[2])
                        spArray.append(measurement[3])
                        measurementNumber += 1
                                            # ax.plot(measurementArray,
                                                inputArray, "b") # create the new
                                                 plot
                                            # ax.plot(measurementArray,
                                                outputArray, "r") # create the
                                                new plot
                # else:
```

```
                    # ax.clear()
        ax.plot(measurementArray, inputArray, "b")
        ax.plot(measurementArray, outputArray, "r")
        ax.plot(measurementArray, spArray, "g") # create the new plot
        if measurementNumber>= figXMax:
            ani.event_source.stop()

def saveMeasurement():
        savefile = asksaveasfilename(filetypes=[('csv file','*.csv'),('text file
            ','*.txt')], defaultextension='.csv')
        with open(savefile,'w',newline="") as f:
                write = csv.writer(f)
                write.writerow(['#', 'input', 'output', 'setpoint'])
                for meas,inp,outp,sp in zip(measurementArray,inputArray,
                    outputArray,spArray):
                        write.writerow([meas,inp, outp, sp])


home_page()
#Degrau_page
#Canvas_page()


root.mainloop()
```

APÊNDICE C – CÓDIGO DE COMUNICAÇÃO

```
    import serial
import serial.tools.list_ports
import time
import struct

def listPorts():
     return serial.tools.list_ports.comports()

class SerialPort:
    def __init__(self, portName):
        self.name = portName
        self.connected = False
        self.ser = None

    def connect(self):
        self.ser = serial.Serial(self.name, 115200,timeout=0.1)
        time.sleep(1) # wait for the serial connection to initialize
        self.ser.read_all() # flush the serial port
        time.sleep(0.1) # wait for the serial connection to initialize
        self.connected = True

    def disconnect(self):
        if self.ser == None:
             pass
        else:
            self.ser.close()
            self.connected = False

    def setType(self,expType):
        # Setting voltage levels based on the values of the inputs
        notCompleted = True
        attempts = 3
        typeSetBuffer = b'M' + int(expType).to_bytes(1,'little') + int(55).
            to_bytes(1,'little')
        print("Setting experiment to {}".format(['step','prbs', 'senoidal', 'pid
            ', 'compensator'][expType]))
        while notCompleted:
            attempts -= 1
            self.ser.write(typeSetBuffer)
```

```
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "Experiment set to"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("Experiment type set")

    def setTA(self,Ta):
        notCompleted = True
        attempts = 3
        taBuffer = b'A'+ int(Ta/256).to_bytes(1,'little') + (Ta%256).to_bytes(1,'
            little')+ int(55).to_bytes(1,'little')
        while notCompleted:
            attempts -= 1
            self.ser.write(taBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            print(response)
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "Ta:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)

    def setVoltages(self, Vsp, Vmin, Vmax):
        # Setting voltage levels based on the values of the inputs
        notCompleted = True
        attempts = 3
        voltageSetBuffer = b'V' + int(Vsp/256).to_bytes(1,'little') + (Vsp%256).
            to_bytes(1,'little') + int(Vmin/256).to_bytes(1,'little') + (Vmin
            %256).to_bytes(1,'little') +int(Vmax/256).to_bytes(1,'little') + (
```

```
            Vmax%256).to_bytes(1,'little') + int(55).to_bytes(1,'little')
        print("Setting voltages to {}, {} and {}".format(Vmin,Vmax,Vsp))
        # print(voltageSetBuffer)
        while notCompleted:
            attempts -= 1
            self.ser.write(voltageSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "V0, V1, V2:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("out of voltage set")

    def setTimes(self, T1, T2):
        notCompleted = True
        attempts = 3
        timeSetBuffer = b'T' + int(T1/256).to_bytes(1,'little') + (T1%256).
            to_bytes(1,'little') + int(T2/256).to_bytes(1,'little') + (T2%256).
            to_bytes(1,'little') + int(55).to_bytes(1,'little')
        print("Setting timings to {} and {}".format(T1, T2))
        print(timeSetBuffer)
        while notCompleted:
            attempts -= 1
            self.ser.write(timeSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "T0, T1:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
```

```
        print("out of time set")

    def setSenoidal(self, Freq, Amp):
        notCompleted = True
        attempts = 3
        timeSetBuffer = b'C' + int(Freq/256).to_bytes(1,'little') + (Freq%256).
            to_bytes(1,'little') + int(Amp/256).to_bytes(1,'little') + (Amp%256).
            to_bytes(1,'little') + int(55).to_bytes(1,'little')
        print("Setting frequency to {} and amplitude to {}".format(Freq, Amp))
        print(timeSetBuffer)
        while notCompleted:
            attempts -= 1
            self.ser.write(timeSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "Freq, Amp:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("out of frequency and amp set")

    def setPRBSTimes(self, Tmin, Tmax):
        notCompleted = True
        attempts = 3
        timeSetBuffer = b'B' + int(Tmin/256).to_bytes(1,'little') + (Tmin%256).
            to_bytes(1,'little') + int(Tmax/256).to_bytes(1,'little') + (Tmax
            %256).to_bytes(1,'little') + int(55).to_bytes(1,'little')
        print("Setting PRBS timings to {} and {}".format(Tmin, Tmax))
        print(timeSetBuffer)
        while notCompleted:
            attempts -= 1
            self.ser.write(timeSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
```

```
            else:
                response = response.split(b'\t')
                if(response[0] == "Tmin, Tmax (PRBS):"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("out of time set")

    def setPID(self, Kp, Ki, Kd):
        notCompleted = True
        attempts = 3
        pidSetBuffer = b'Q' + bytearray(struct.pack("f", Kp)) + bytearray(struct.
            pack("f", Ki))+ bytearray(struct.pack("f", Kd)) + int(55).to_bytes(1,'
            little')
        print("Setting PID constants to {}, {} and {}".format(Kp, Ki, Kd))
        print(pidSetBuffer)
        while notCompleted:
            attempts -= 1
            self.ser.write(pidSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "Kp, Ki, Kd:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("out of PID set")

    def setLeadLag(self, Kp, Tc1, Tc2):
        notCompleted = True
        attempts = 3
        llSetBuffer = b'L' + bytearray(struct.pack("f", Kp)) + bytearray(struct.
            pack("f", Tc1))+ bytearray(struct.pack("f", Tc2)) + int(55).to_bytes
            (1,'little')
        print("Setting lead lag constants to {}, {} and {}".format(Kp, Tc1, Tc2))
        print(llSetBuffer)
```

```
        while notCompleted:
            attempts -= 1
            self.ser.write(llSetBuffer)
            time.sleep(0.01)
            response = self.getMeasure()
            if isinstance(response,int):
                notCompleted = True
            else:
                response = response.split(b'\t')
                if(response[0] == "Kp, Tc1, Tc2:"):
                    notCompleted = False
            if attempts == 0:
                notCompleted = False
        print(response)
        print("out of lead lag set")

    def run(self):
        runCommand = b'R7' # 'R' is the command to run the experiment, '7' (55)
            is the EoP
        self.ser.write(runCommand)
        print("Sent run command")
        time.sleep(0.1)

    def getMeasure(self):
        inputPackage = b"I"
        contador = 100
        inputPackage = self.ser.read_until().strip()
        # print("Received: {}".format(inputPackage))
        # print(inputPackage)
        if len(inputPackage)>1:
            if (inputPackage[0] == b'E'[0]) and (inputPackage[-1] == b'7'[0]):
                vInput = int(inputPackage[1])*256 + int(inputPackage[2])
                vOutput = int(inputPackage[3])
                vSp = int(inputPackage[4])*256 + int(inputPackage[5])
                return ('meas',vInput,vOutput,vSp)
            else:
                return inputPackage
        else:
            return(0)
```