Eduardo Martins de Carvalho Caixeta
José Humberto Ferreira Ramos Carvalho

Sistema de aquisição de dados de baixo custo utilizando Arduino

Brasil

16 de agosto de 2023

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): Eduardo Martins de Carvalho Caixeta e José Humberto Ferreira Ramos Carvalho

Título do trabalho: Sistema de aquisição de dados de baixo custo utilizando Arduino

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento

[x] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

Casos de embargo:

- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro.

Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 18/08/2023, às 11:31, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13

de novembro de 2020.

Documento assinado eletronicamente por **Eduardo Martins De Carvalho Caixeta**, **Discente**, em 18/08/2023, às 17:38, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Jose Humberto Ferreira Ramos Carvalho**, **Discente**, em 19/08/2023, às 08:15, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3975121** e o código CRC **D757BD82**.

Referência: Processo nº 23070.022358/2023-60 SEI nº 3975121

Eduardo Martins de Carvalho Caixeta
José Humberto Ferreira Ramos Carvalho

Sistema de aquisição de dados de baixo custo utilizando Arduino

Trabalho de conclusão de curso apresentado para a obtenção do título de bacharel em Engenharia Elétrica pela Escola de Engenharia Elétrica, Mecânica e de Computação da Universidade Federal de Goiás.

Universidade Federal de Goiás – UFG

Escola de Engenharia Elétrica, Mecânica e Computação

Orientador: Dr. Álisson Assis Cardoso

Brasil

16 de agosto de 2023

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Caixeta, Eduardo Martins de Carvalho
Sistema de aquisição de dados de baixo custo utilizando Arduino [manuscrito] / Eduardo Martins de Carvalho Caixeta, José Humberto Ferreira Ramos Carvalho. - 2023.
26 f.: il.

Orientador: Prof. Dr. Álisson Assis Cardoso.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Goiás, Escola de Engenharia Elétrica, Mecânica e de Computação (EMC), Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Goiânia, 2023.
Bibliografia. Apêndice.
Inclui gráfico, tabelas, algoritmos.

1. Osciloscópio. 2. Arduíno. 3. Função de transferência. 4. Python. 5. sistemas de controle. I. Carvalho, José Humberto Ferreira Ramos. II. Cardoso, Álisson Assis, orient. III. Título.

CDU 621.3

UNIVERSIDADE FEDERAL DE GOIÁS
ESCOLA DE ENGENHARIA ELÉTRICA, MECÂNICA E DE COMPUTAÇÃO

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

ATA DE AVALIAÇÃO DE PROJETO FINAL

Curso

(x) Eng Elétrica	( ) Eng Mecânica	( ) Eng Computação PFC 1 ( ) PFC 2 ( )

Título do Trabalho

Sistema de aquisição de dados de baixo custo utilizando Arduino

Banca Avaliadora

Membro 1	Prof. Dr. Álisson Assis Cardoso
Membro 2	Dr. Gustavo Souto de Sá e Souza
Membro 3	Prof. Dr. José Wilson Lima Nerys

Discente

Matrícula	Nome
201905573	Eduardo Martins de Carvalho Caixeta
201905597	José Humberto Ferreira Ramos Carvalho

NOTAS										
	Membro 1			**Membro 2**			**Membro 3**			
Matrícula	**NPT**	**NTE**	**NAA**	**NPT**	**NTE**	**NAA**	**NPT**	**NTE**	**NAA**	**Média***
201905573	10	10	10	10	10	10	10	10	10	10
201905597	10	10	10	10	10	10	10	10	10	10

NPT – Nota plano de trabalho;
NTE – Nota do trabalho escrito;
NAA – Nota de apresentação e arguição
Para Eng. Elétrica, Mecânica e PFC2 da Eng. Da Computação: NF = 0,1 x NPT + 0,45xNTE + 0,45 x NAA
Para PFC1 da Eng. Da Computação: NF = 0,3 x NPT + 0,7 x NAA

* A APROVAÇÃO DO(S) ALUNO(S) ESTÁ CONDICIONADA À APRESENTAÇÃO DO TRABALHO FINAL AO ORIENTADOR COM TODAS AS CORREÇÕES SUGERIDAS PELA BANCA.

OBSERVAÇÕES:

Preencher com modificações solicitadas, caso existam. Em caso de reprovação, informar a

justificativa.

_____

_____

_____

_____

_____

Documento assinado eletronicamente por **Gustavo Souto De Sá E Souza**, **Técnico de Laboratório**, em 16/08/2023, às 14:57, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 16/08/2023, às 14:57, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **José Wilson Lima Nerys**, **Professor do Magistério Superior**, em 16/08/2023, às 14:57, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3961068** e o código CRC **1BFE607A**.

Referência: Processo nº 23070.022358/2023-60 SEI nº 3961068

Sistema de Aquisição de Dados de Baixo Custo Utilizando Arduino

CAIXETA. Eduardo, RAMOS. José, *Escola de Engenharia Elétrica Mecânica e de Computação, UFG*

Resumo - Atualmente, os aparelhos residenciais possuem dispositivos elétricos que apresentam propriedades bem definidas. Estas são escopo de estudo de inúmeros acadêmicos de engenharia elétrica. Mesmo que esses aparelhos sejam de fácil manuseio, os estudos dos sistemas e dispositivos que os compõem, necessitam de uma análise aprofundada, e, para realizar essa análise, a utilização de instrumentos de medida são essenciais.

Alguns desses dispositivos são conhecidos, como o multímetro, voltímetro, amperímetro e o osciloscópio. Esses equipamentos são bastante eficientes e apresentam uma gama de utilidades, como também são demasiadamente precisos, apresentando a indicação de um ou mais parâmetros característicos, como valor eficaz, valor de pico e frequência de grandezas como tensão e corrente.

Por conseguinte, percebe-se que a aquisição do osciloscópio é obrigatória para os alunos de engenharia, sendo assim este documento apresenta os princípios de funcionamento do osciloscópio e um modelo básico feito com a utilização de softwares comuns e dispositivos frequentes nos laboratórios e salas de pesquisa das escolas de engenharia. Ademais, o desenvolvimento desse osciloscópio tem o intuito de colaborar com a prática da pesquisa e estudo em laboratório.

Abstract— Currently, most residential appliances have electrical devices that have well-defined properties. These are the scope of study of countless electrical engineering students. Even if these devices are easy to handle, the studies of the systems and devices that compose them require an in-depth analysis, and to carry out this analysis the use of measuring instruments is essential.

Some of these devices are known, such as the multimeter, voltmeter, ammeter and oscilloscope. These devices are very efficient and have a range of utilities, as well as being very precise, showing the indication of one or more characteristic parameters, such as effective value, peak value and frequency of magnitudes such as voltage and current.

Therefore, it is clear that the oscilloscope's acquisitiont is mandatory for engineering students, so this document presents the operating principles of the oscilloscope and a basic model made with the use of common software and devices frequent in laboratories and research rooms. of engineering schools. Furthermore, the development of this oscilloscope is intended to collaborate with the practice of research and study in the laboratory

***Palavras-chave*—osciloscópio, arduíno, função de transferência, circuito RC, Python, servo motor, programação, comunicação serial, taxa de amostragem, conversor A/D,regressão linear, análise de dados, MATLAB, sistemas de controle.**

I. Introdução

Os cursos das áreas científicas, especialmente o curso de engenharia elétrica, necessitam demasiadamente da utilização do osciloscópio como ferramenta fundamental para manutenção e análise de circuitos eletrônicos, sejam eles digitais ou analógicos. Utilizar essa ferramenta é de extrema importância devido às limitações humanas em perceber os fenômenos elétricos nos domínios do tempo e da frequência.

O osciloscópio é como se fosse uma extensão dos sentidos. Para que se possa mensurar a importância do deste aparelho, pode-se valer de uma comparação grosseira com instrumentos utilizados em outras áreas científicas. O osciloscópio é tão útil para um engenheiro, como um microscópio é para o biólogo. Existe uma diferença no sentido de que o microscópio permite a visão de fenômenos biológicos de um micro-universo, já o osciloscópio garante a visualização do universo das variações elétricas.

No entanto, este equipamento ainda possui alto custo tornando-o algo de difícil acesso.

Pensando nisso, o presente trabalho discutirá o desenvolvimento de um osciloscópio didático de baixo custo utilizando um microcontrolador, com conversor A/D e comunicação serial, juntamente com um computador.

Esse osciloscópio terá a capacidade de fazer a leitura de sinais elétricos que variam temporalmente e mostrar esses dados através de uma interface com o usuário para que este possa visualizar, salvar e analisar os sinais da forma que preferir.

A. Objetivos

A proposta do referido projeto se baseia em desenvolver um osciloscópio digital utilizando recurso de baixo custo como um microcontrolador qualquer que contenha:

- Comunicação serial; e
- Conversor A/D.

Como também uma máquina computacional que possa executar o *software* responsável pela comunicação com o microcontrolador com a finalidade de apresentar os dados retirados da análise de sinais elétricos no domínio do tempo.

Desta forma, estabelece-se os seguintes objetivos para o projeto deste osciloscópio:

- Desenvolver uma código de aquisição de dados para o microcontrolador;
- Desenvolver uma interface para a visualização dos dados pelo usuário;
- Permitir a interação do usuário com controle da interface;
- Armazenar informações de dados coletados através de vetores em arquivos *.m*;
- Representar formas de onda no domínio do tempo;

- Desenvolver todo trabalho sem utilizar ferramentas ou equipamentos de alto custo;
- Realizar o teste e validação do osciloscópio projetado através de um circuito RC.
- Obter a função de transferência do servo motor presente no Laboratório de Sistema de Controle;

B. Motivação

A motivação para a realização desse trabalho está relacionada ao desenvolvimento de um osciloscópio de baixo custo para que estudantes e profissionais tenham disponível uma plataforma onde poderão realizar seus experimentos.

Com o osciloscópio proposto neste projeto, o custo inicial será bastante reduzido fazendo sua implementação se tornar viável e acessível a todos.

Este projeto apresentará uma interface a nível de usuário bastante intuitiva e amigável, desta forma mesmo usuários iniciantes não terão problemas para utilizá-lo. Com essas funcionalidades é possível obter leituras e ajustes fidedignos em relação a qualquer outro osciloscópio.

Outro fator interessante do projeto é a possibilidade de salvar os dados lidos a partir da interface gráfica implementada na forma de vetores para que estes sejam tratados nas ferramentas mais utilizadas na análise de dados, como o MATLAB, por exemplo,

Também é necessário justificar aqui que o osciloscópio a ser montado não é um equipamento de desempenho superior aos comerciais. Ele traz o essencial que se espera deste instrumento, que é a visualização da forma de onda no tempo.

Nesse sentido, pode-se afirmar que a principal motivação para o desenvolvimento do projeto é reduzir o custo de um osciloscópio capaz de realizar medições didáticas com custo mínimo e que seja capaz de alcançar os usuários através da facilidade de utilização.

C. Estrutura de Trabalho

O presente Projeto Final de Curso (PFC) apresenta a realização de um projeto de desenvolvimento de um osciloscópio utilizando um microcontrolador como peça central. O escopo abrange desde o desenvolvimento do software de comunicação entre o microcontrolador e o sistema de aquisição de dados até a análise de circuitos e sistemas por meio das medições realizadas.

I. Introdução
  A) Objetivos do Trabalho
  B) Motivação

II. Revisão Bibliográfica
  A) Osciloscópio padrão
    1) Osciloscópio analógico
    2) Osciloscópio digital
  B) Conversor A/D
    1) Conversor por aproximações sucessivas
  C) Taxa de amostragem
  D) Comunicação serial
  E) Circuito RC
  F) Função de transferência
  G) Rergressão linear para casos exponenciais
  H) Servo-motor

III. Materiais e métodos
  A) Equipamentos e Softwares
  B) Desenvolvimento da interface para o usuário
  C) Circuito RC de teste
  D) Servo motor do laboratório
  E) Leitura e análise de dados

IV. Resultados
  A) Circuito RC de teste
    1) Medições com Osciloscópio padrão
    2) Medições com Osciloscópio-Arduíno
    3) Análise de dados de medição do circuito RC
      3.1) Função da curva
      3.2) Constante de tempo
      3.3) Análise da Taxa de amostragem
      3.4) Validação do projeto
  B) Kit servo-motor
    1) Medições com Osciloscópio padrão
    2) Medições com Osciloscópio-Arduíno
    3) Análise de dados do servo motor
    4) Função de transferência do servo-motor

V. Conclusão

Referências Bibliográficas

APÊNDICE A
APÊNDICE B
APÊNDICE C

II. Revisão Bibliográfica

Esta seção traz conceitos que foram importantes ao longo do desenvolvimento do projeto. O conjunto de informações inseridas nesta seção traz ao leitor, uma base para que possa compreender os passos que foram seguidos durante o trabalho.

A. Osciloscópio Padrão

1) Histórico do osciloscópio

Em 1897, o físico alemão Karl Ferdinand Braun construiu o tubo de raios catódicos (TRC), em inglês (CRT) *cathode ray tube*, que consistia em um dispositivo que produz imagens quando uma quantidade de elétrons incide sobre a tela recoberta por fósforo, como mostra a Figura 1.

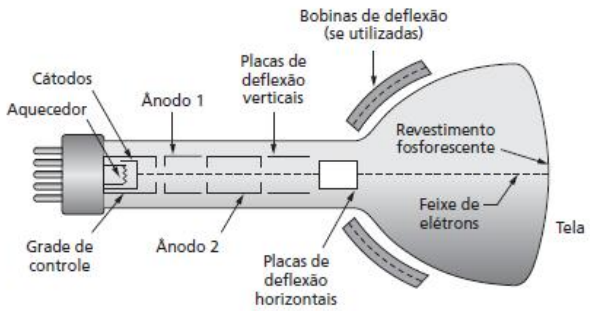

Figura 1: Esquema de um tubo de raios catódicos.

O feixe de elétrons, momentos antes de atingir a tela, poderia ser submetido a um campo elétrico ou magnético causando uma mudança de trajetória, o que possibilita a projeção de diversas imagens na tela. O tubo de raios catódicos foi uma invenção de grande valia para a eletrônica, uma vez que os monitores dos computadores, televisões e osciloscópios foram produzidos partindo de seu princípio de funcionamento.

Desta forma, pode-se afirmar que o tubo de raios catódicos foi o primeiro osciloscópio a ser invetado.

2) Osciloscópio Analógico

Nos osciloscópios analógicos o sinal é produzido através de feixes de elétrons que incidem sob a tela fosforescente.

O sinal de tensão adquirido passa primeiramente pelo "sistema vertical" do osciloscópio. Esse sistema possui duas placas defletoras na horizontal, mas que permite o desvio do feixe de elétrons na vertical. No"sistema vertical", dependendo de como a escala é ajustada, o sinal terá a amplitude reduzida ou aumentada através de circuitos atenuadores e amplificadores. Em seguida o sinal passa pelo "sistema horizontal" do osciloscópio, que permitirá a visualização do sinal no domínio do tempo, isso é conseguido através de duas placas defletoras localizadas na vertical, mas que permite o desvio do feixe de elétrons na horizontal (ALVES, 1998).

3) Osciloscópio Digital

Assim como no osciloscópio analógico, haverá um sistema responsável por efetuar o deslocamento do sinal na vertical. O que muda é a forma como o osciloscópio digital faz isso, pois agora não haverá as placas defletoras, mas sim, um circuito responsável por efetuar o ajuste da amplitude do sinal. O osciloscópio digital também possui um sistema responsável por fazer a aquisição do sinal, de forma que o sistema "recolherá" amostras do sinal que está sendo medido e converterá o valor analógico de cada amostra em um valor digital.

O sistema horizontal possui um dispositivo chamado "*sample clock*". Ele é responsável por determinar a frequência com que o conversor A/D realizará a aquisição das amostras e a conversão das mesmas. Após ter concluído a conversão, as amostras são armazenadas na memória para posteriormente serem mostradas na tela. Cada amostra possuirá um conjunto de bits que o representará, e cada amostra será um ponto, que no final quando todos esses pontos forem juntados, formará o sinal que está sendo medido.

O funcionamento deste tipo de osciloscópio se assemelha a um quebra cabeça, onde as amostras representam as peças, e o sinal formado a partir de todas essas amostras, representa o quebra cabeça montado por completo, ou seja, representa o resultado final.

O conjunto de amostras é chamado de registro, e o osciloscópio digital, assim como no analógico, terá de ter um sistema de sincronismo. No caso do osciloscópio digital o sistema responsável pelo sincronismo determinará o início e o fim do registro, de forma que será definido o número de amostras necessárias e, consequentemente se terá um registro, que será armazenado na memória para posteriormente ser apresentado na tela.

Os osciloscópios digitais realizam a aquisição do sinal que está sendo medido, realiza a conversão, grava na memória, e posteriormente converte para um sinal analógico.

Após o armazenamento do registro na memória, ele é apresentado na tela LCD para que o operador do equipamento possa enxergar o sinal. Com isso pode-se entrar no método de amostragem, que nada mais é que a forma com que o osciloscópio faz a aquisição das amostras.

Para sinais em frequências baixas o osciloscópio não encontra nenhum problema quanto a captura das amostras, de forma que geralmente um osciloscópio irá apresentar um sinal com uma boa qualidade, caso o sinal medido esteja a uma frequência considerada baixa. Porém existe um problema quando o sinal medido está a uma frequência alta, pois a depender da capacidade do osciloscópio em questão, acontecerá do mesmo não consiga capturar a quantidade de amostras necessárias. Sendo assim, a precisão e resolução desse sinal ficarão comprometidas.

B. Conversor A/D

De acordo com MAZIDI, NAIMI e NAIMI (2011) conversores analógicos-digitais são dispositivos mais amplamente utilizados para aquisição de dados. Computadores digitais usam valores binários (discreto), mas no mundo real tudo é analógico. Temperatura, pressão, umidade, e velocidade são alguns exemplos. Uma quantidade física é convertida em um sinal elétrico (tensão, corrente) usando um dispositivo denominado transdutor. Transdutores também são referidos como sensores. Sensores de temperatura, velocidade, pressão, luz entre outros produzem uma saída de tensão (ou corrente). Entretanto, faz-se necessário um conversor analógico-digital (A/D) para traduzir os sinais analógicos em números digitais para que o microcontrolador possa lê-los e processá-los.

Um conversor analógico-digital (A/D) possui *n-bit* de resolução, onde n é um número inteiro, podendo assumir valores como 8, 10, 12, 16 ou mesmo 24 bits. Maiores resoluções do A/D fornecem menor *step size*, que diz respeito à diferença de tensão entre um nível de tensão e o próximo.

Cabe ressaltar que a resolução de A/D não pode ser alterada. Além disto, o tempo de conversão, definido como o tempo em que o A/D converte o sinal analógico para um número digital (binário), é também uma fator de grande impacto para circuitos que dependem desta conversão. Este é ditado pelo *clock* da fonte conectada ao A/D .

A tensão conectada ao pino, juntamente a resolução do A/D, determina o tamanho da etapa (*step size*). Isto é, para 8-bit, o *step size* é igual a 2 elevado a 8, resultando em 256. Desta forma, tem-se que:

$$0000000_2 = 0$$

$$0000001_2 = \frac{V_{ref}}{255}$$

$$0000010_2 = \frac{2V_{ref}}{255}$$

$$\vdots$$

$$11111111_2 = V_{ref}$$

1) Conversor por aproximações sucessivas

Dentre os modelo disponíveis no mercado, destaca-se o conversor analógico-digital por meio de aproximações sucessivas.

Conforme o próprio nome diz, o que diferencia este circuito do anterior é a troca do contador por um registador de aproximações sucessivas, que o torna muito mais rápido, não só reduzindo os tempos de conversão mas uniformizando-os, ou seja, tornando-os iguais independentemente do ponto da escala em que o sinal de entrada se encontre.

O sinal aplicado a entrada é retido pelo circuito de amostragem e retenção, aplicado à entrada do comparador e ao mesmo tempo dispara o circuito de *clock* do setor de conversão digital.

Ao iniciar a conversão, o registador de aproximações sucessivas começa colocando a 1 o bit mais significativo (MSB) da saída, aplicando este sinal no conversor D/A. Se com este procedimento, a tensão aplicada pelo conversor D/A à entrada de referência do comparador for maior que a de entrada, isso será um sinal que o valor que este bit representa é maior que o que se deseja converter.

O comparador informa isso ao registro de aproximações, que então volta o MSB a zero e coloca o bit que o segue imediatamente a 1. Uma nova comparação é feita. Se agora o valor da tensão for menor que a de entrada, este bit é mantido, e testa-se o seguinte, colocando a 1. Se novamente o valor for ultrapassado, o comparador informa isso ao registro e o bit volta a zero passando o seguinte a 1 que é testado.

Quando todos os bits forem testados, tem-se, na saída do registro, um valor binário muito próximo do desejado, dependendo da resolução do circuito. Testando todos os bits desta forma, a conversão se torna muito rápida, já que não será preciso esperar a contagem até o final.

A Figura 2 exemplifica tal esquema na forma de um diagrama para maior compreensão.

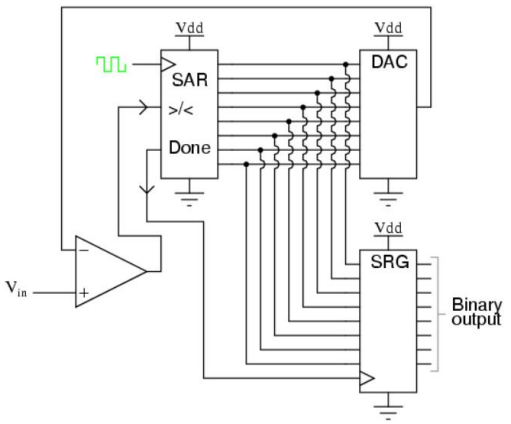

Figura 2: Esquema de um conversor A/D por aproximações sucessivas.

C. Taxa de amostragem

Sabe-se que a conversão do sinal analógico para o digital é realizada por uma sequência de amostras da variação de voltagem do sinal original. Cada amostra é arredondada para o número mais próximo da escala usada e depois convertida em um número digital binário (formado por "uns" e "zeros") para ser armazenado, como mostra as Figuras 3 e 4.

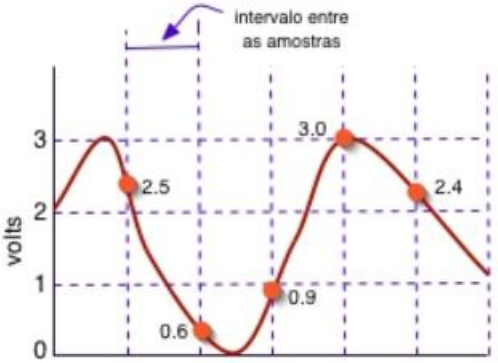

Figura 3: Amostragem de valores analógicos.

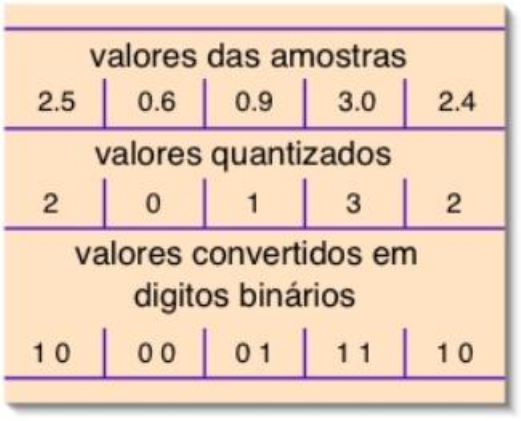

Figura 4: Esquema conversão de valores analógicos.

Da Figura 3, percebe-se que as amostras são medidas em intervalos fixos. Sendo assim, o números de vezes em que se realiza a amostragem em uma unidade de tempo é a taxa de amostragem, geralmente medida em Hertz. Assim, dizer que a taxa de amostragem de áudio em um CD é de 44.100 Hz, significa que a cada segundo de som são tomadas 44.100

medidas da variação de voltagem do sinal. Dessa maneira, quanto maior for a taxa de amostragem, mais precisa é a representação do sinal, porém é necessário que se realize mais medições e que se utilize mais espaço para armazenar esses valores.

A taxa de amostragem afeta diretamente as características dos sinais que podem ser lidos por determinado conversor pois esta dever ser pelo menos duas vezes a maior frequência que se deseja registrar. Valor conhecido na literatura como frequência de Nyquist.

Ao se tentar reproduzir uma frequência maior do que a frequência de Nyquist, ocorre um fenômeno chamado *alising* (ou *foldover*), onde a frequência é "espelhada" ou "rebatida" para uma uma região mais grave do espectro, como mostra a Figura 5.

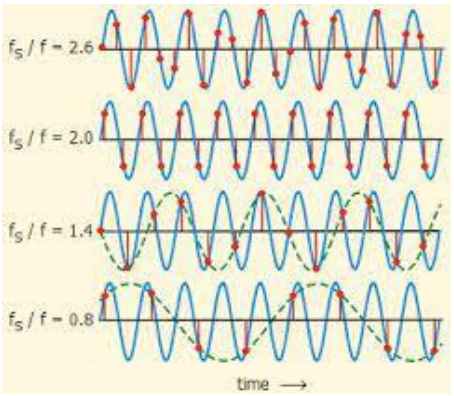

Figura 5: Esquema conversão de valores analógicos.

D. Comunicação Serial Assíncrona

A comunicação serial é um processo de transmissão de dados bit a bit de forma sequencial. Ou seja, um único bit é enviado por vez em um único canal de comunicação (um fio, por exemplo).

Além disso, existe a comunicação paralela, que envia vários bits simultaneamente por canais separados. Cada canal é responsável pela transmissão de um único bit, mas todos eles operam em conjunto. Isso torna a comunicação paralela mais rápida, pois é possível enviar mais informações em uma mesma velocidade.

No entanto, para a realização deste projeto, optou-se pela comunicação serial, apesar de ser mais lenta, devido à sua maior acessibilidade em termos de custo.

A comunicação serial é uma alternativa econômica e viável para transmitir dados entre dispositivos e pode ser dividida em duas categorias: síncrona e assíncrona.

Na comunicação assíncrona, não é utilizado nenhum recurso para sincronizar o dispositivo de envio com o responsável por receber dos dados. Em contraste, na comunicação síncrona, os dispositivos envolvidos possuem um mecanismo direto para sincronizar a leitura e escrita dos dados no canal.

Comunicação serial assíncrona descreve um protocolo de transmissão assíncrono de *n-bits* no qual cada sinal de inicialização (*start*) é enviado previamente para cada *byte*, caractere ou código de palavra. Após o término da transmissão dos bits de informação, um sinal de finalização (*stop*) é enviado.

O sinal de inicialização serve para preparar o mecanismo de recebimento para a recepção e registro dos próximos bits a serem recebidos. Já o de finalização tem o objetivo de preparar o mecanismo de recepção para o próximo sinal. Um tipo comum de sinal de inicialização-finalização é o ASCII sobre RS-232, por exemplo para uso em operação de teletipo.

A comunicação com o terminal é realizada com protocolo assíncrono, como mostrado na Figura 6, onde o primeiro bit (0) informa o início do dados, compostos por 8 bits, logo após é enviado o bit de paridade, que é opcional,, para que seja possível identificar qualquer erro durante a transmissão, e, por último, é enviado o bit de parada, informando o término da transmissão.

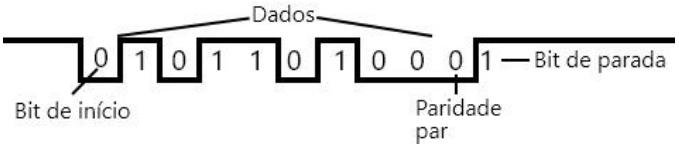

Figura 6: Exemplo de comunicação assíncrona.

Uma das formas de se determinar velocidade de comunicação é em *bit/s*, que, em um sistema de transmissão de dados binários, representam o número de bits transmitidos por segundo.

E. Circuito RC

Os circuitos RC, também conhecidos como circuitos resistor-capacitor, são fundamentais na eletrônica e desempenham um papel crucial em uma variedade de aplicações. Esses circuitos combinam um resistor e um capacitor conectados em série ou paralelo, formando uma rede que apresenta características únicas de resposta a sinais elétricos.

A interação entre o resistor e o capacitor cria uma série de fenômenos interessantes, como carregamento e descarregamento do capacitor, que influenciam a maneira como o circuito responde a diferentes sinais. Devido à capacidade do capacitor de armazenar carga elétrica, os circuitos RC são amplamente utilizados em filtragem de sinais, circuitos temporizadores, atenuadores de sinais, entre outras aplicações.

Desta forma, considerando o circuito RC série, apresentado na Figura 7, busca-se determinar a tensão do capacitor para um instante *t*.

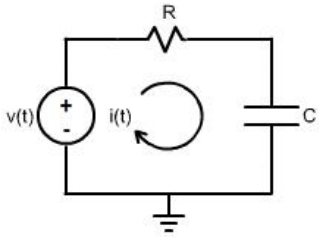

Figura 7: Circuito RC série.

Sabendo que

$$Q = CV_c \tag{1}$$

E que

$$Q = \frac{\mathrm{d}i}{\mathrm{d}t} \tag{2}$$

Tem-se

$$\frac{1}{C} \cdot \frac{\mathrm{d}i}{\mathrm{d}t} = V_c \tag{3}$$

Baseado na Lei de Ohm, pode-se afirmar que

$$V_{cc} = V_r + V_C \tag{4}$$

$$V_{cc} = \frac{\mathrm{d}Q}{\mathrm{d}t}R + \frac{1}{C} \cdot Q \tag{5}$$

$$-\frac{\mathrm{d}Q}{\mathrm{d}t} = \frac{Q}{RC} - \frac{V_{cc}C}{RC} \tag{6}$$

$$-\frac{\mathrm{d}Q}{Q - CV_{cc}} = \frac{\mathrm{d}t}{RC} \tag{7}$$

$$Q = V_{cc}C(1 - e^{-\frac{t}{RC}}) + Q_0 \tag{8}$$

Das equações (1) e (8), pode-se determinar que

$$V_c = V_{cc}(1 - e^{-\frac{t}{RC}}) + V_0 \tag{9}$$

Cuja curva é a de uma exponencial deslocada verticalmente, como apresentado na Figura 8.

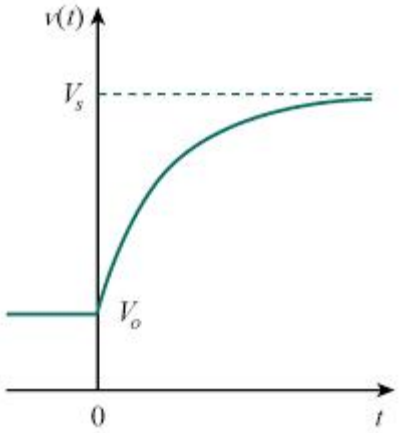

Figura 8: Gráfico de carga do circuito RC.

Onde RC é chamado de constante de tempo do circuito, sendo R a resistência elétrica e C a capacitância do capacitor.

$$\tau = RC \tag{10}$$

A constante de tempo (τ) representa o tempo necessário para que a tensão em um circuito RC alcance aproximadamente 63,2% de seu valor final após uma variação ou perturbação.

Essa constante de tempo é uma característica fundamental do circuito RC e influencia sua resposta a diferentes sinais elétricos. Quando a constante de tempo é pequena, o capacitor carrega ou descarrega rapidamente, resultando em respostas rápidas às variações de entrada. Por outro lado, uma constante de tempo maior indica que o capacitor leva mais tempo para carregar ou descarregar, resultando em respostas mais lentas.

A constante de tempo do circuito RC desempenha um papel essencial em aplicações como filtros de sinais, circuitos temporizadores, osciladores e muito mais. É uma ferramenta valiosa para projetistas e engenheiros, pois permite controlar o comportamento temporal dos circuitos e ajustar a resposta de acordo com as necessidades específicas da aplicação.

F. Função de Transferência

Uma função de transferência no domínio da frequência é uma representação matemática que descreve a relação entre a entrada e a saída de um sistema linear em função da frequência do sinal de entrada. Ela é amplamente utilizada em engenharia, especialmente na análise e projeto de sistemas dinâmicos, como circuitos elétricos, sistemas de controle, filtros, entre outros.

$$H(s) = \frac{Y(s)}{X(s)} \tag{11}$$

Onde *Y(s)* é a transformada de Laplace da saída do sistema (resposta) e *X(s)* é a transformada de Laplace da entrada (estímulo).

A obtenção da função de transferência envolve a aplicação de técnicas de análise de sistemas lineares, que podem incluir a resolução de equações diferenciais, a aplicação de teoremas e propriedades da transformada de Laplace, e a utilização de conceitos como impedância, admitância e resposta em frequência, para modelagem de circuitos elétricos, por exemplo.

Uma vez obtida a função de transferência, ela fornece informações valiosas sobre o comportamento do sistema na frequência, como ganho, fase e resposta em frequência. Essas informações são essenciais para entender e projetar sistemas que atendam aos requisitos específicos de desempenho e estabilidade.

Em resumo, uma função de transferência no domínio da frequência é uma ferramenta poderosa para analisar e projetar sistemas lineares, proporcionando uma visão clara das características do sistema em resposta a diferentes frequências de entrada.

G. Regressão linear para casos exponenciais

Faz-se necessário compreender uma ferramenta matemática poderosa que foi utilizada nesse projeto, trata-se da regressão linear, com ênfase em casos exponenciais.

Os principais objetivos de uma regressão são:

- Permitir construir um modelo matemático que representa atributos (*x* e *y*);
- Determinar *f(x)*, onde *y = f(x)*, em que *f(x)* é a função que relaciona *x* e *y*;

Um caso de regressão linear simples considera que a relação da resposta às variáveis é uma função linear de alguns parâmetros, exemplificado na Figura 10.

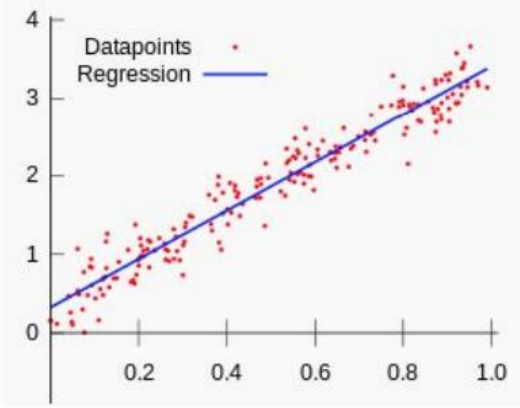

Figura 10: Regressão linear simples.

O modelo de regressão linear são frequentemente ajustados usando a abordagem dos mínimos quadrados.

Modelos de regressão linear não costumam ser válidos para fins de extrapolação, ou seja, predizer um valor fora do domínio dos dados. Aproximadamente, 95% dos resíduos estarão no intervalo de um desvio padrão da média.

De modo geral, a abordagem a ser utilizada para resolver a regressão linear é obter os parâmetros da curva é representada por:

$$\begin{pmatrix} y_1 \\ y_2 \\ \dots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \dots & x_{1p} \\ 1 & x_{21} & \dots & x_{2p} \\ \dots & \dots & \dots & \dots \\ 1 & x_{n1} & \dots & x_{np} \end{pmatrix} * \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix} \quad (12)$$

O método dos mínimos quadrados pode ser facilmente adaptado para o caso exponencial, usando logaritmos neperianos, onde

$$y = be^{ax} + c \quad (13)$$

$$y' = ln(y - c) \quad (14)$$

Desta forma, a equação que modela a relação entre uma variável independente e uma variável dependente será dada por:

$$y' = ax + b' \quad (15)$$

Onde

$$a = \frac{n\sum_{i=1}^{n} x_i y_i' - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i'}{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \quad (16)$$

$$b' = y' - ax \quad (17)$$

$$b = e^{y'-ax} \quad (18)$$

H. Servo Motores

Um dos dispositivos utilizados nesse projeto é o servo motor. Sendo assim, é necessário conhecer o objetivo do seu uso e como funciona. Um servo motor, exemplificado na Figura 11, é um atuador rotativo ou linear que garante o controle, velocidade e precisão em aplicações de controle de posição em malha fechada.

O servo motor é projetado com pequeno diâmetro e longo comprimento do rotor se diferenciando dos motores convencionais. Ele trabalha com servomecanismo que usa o *feedback* de posição para controlar a velocidade e a posição final do motor. Internamente, esse dispositivo combina um motor com um circuito de realimentação, um controlador e outros circuitos complementares. E, usando um codificador ou sensor de velocidade (*encoder*), ele possui a função de fornecer o *feedback* de velocidade e posição.


Figura 11: Servo motor.

O sinal de realimentação por sua vez é comparado com a posição de comando de entrada (posição desejada do motor correspondente a uma carga) e produz o sinal de erro igual a diferença entre eles.

O sinal de erro disponível na saída do detector não é suficiente para acionar o motor. Assim, o detector de erro alimenta um servo amplificador que eleva a tensão e o nível de potência do sinal de erro e então gira o eixo do motor para a posição desejada.

III. MATERIAIS E MÉTODOS

A. Equipamentos e Softwares

Para o desenvolvimento do osciloscópio projetado, é necessário um microcontrolador capaz de realizar comunicação serial assíncrona e, que ao mesmo tempo, possua um conversor A/D.

Desta forma, optou-se pela microcontrolador Arduíno UNO R3, com microchip ATmega328 que possui um conversor A/D por aproximações sucessivas de 10 bits e com transmissão serial de 8 bits de dados, apresentado na Figura 12.

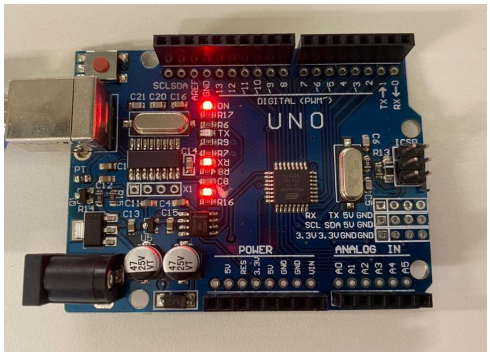

Figura 12: Arduíno Uno R3 utilizado no projeto

A fim de realizar a leitura de dois canais distintos, as portas analógicas A0 e A1 irão receber a entrada analógica da tensão que serão convertidas para sinais digitais. Logo após, serão enviadas pela função *Serial.print()* até o computador, conectado ao Arduíno pelo cabo USB.

Para isto, com a ajuda do *software* Arduino IDE 2.1.1, que pode ser gratuitamente encontrado na internet, gravou-se o código do APÊNDICE A no microcontrolador.

Além disto, visando estimar a função de transferência de um servo motor, é necessário o uso do MATLAB, desenvolvido pela MathWorks. Apesar de ser utilizado neste projeto, o MATLAB não é um software essencial para o manuseio do osciloscópio, sendo usado apenas para tratar os dados e estimar a função de transferência.

Desta forma, para que seja possível uma validação do osciloscópio projetado, é preciso que as mesmas medições feitas por este, também sejam efetuadas por um osciloscópio digital de alto custo. Para tais medições utilizou-se o dispositivo do Laboratório de Sistemas de Controle, apresentado na Figura 13.

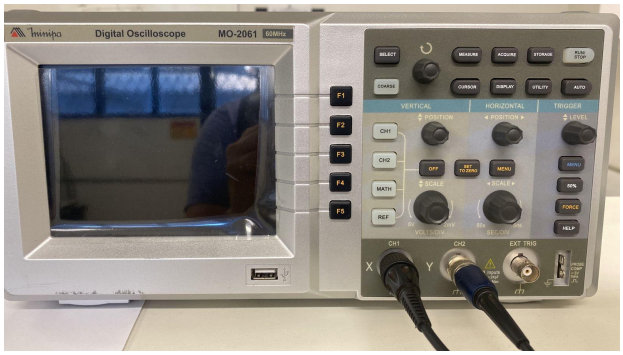

Figura 13: Osciloscópio padrão de alto custo.

B. Desenvolvimento da interface para o usuário

Para máquina computacional que irá receber os dados em sinais digitais, desenvolveu-se o código do APÊNDICE B o qual é responsável por realizar a leitura e impressão dos pontos nos gráficos.

O *software* desenvolvido visa uma fácil interpretação por parte do usuário. Sendo assim, estabeleceu-se uma barra de menu, apresentada na Figura 14, onde é possível controlar toda entrada e salvamento de dados.

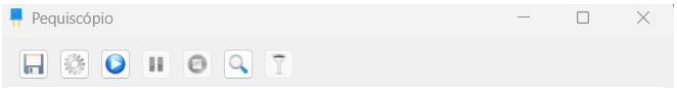

Figura 14: Barra de menu do *software* de recebimento de dados.

Na Figura 14, é possível observar sete botões. Sendo eles, da esquerda para direita:

- Salvar: responsável por salvar os dados lidos pelos canais na forma de arquivo *.m* e uma imagem, formato *.png*, com o gráfico de cada canal
- Configuração: cujo objetivo é permitir que o usuário altere parâmetros gráficos (escala, títulos, etc) e da comunicação serial, como mostra a Figura 15.

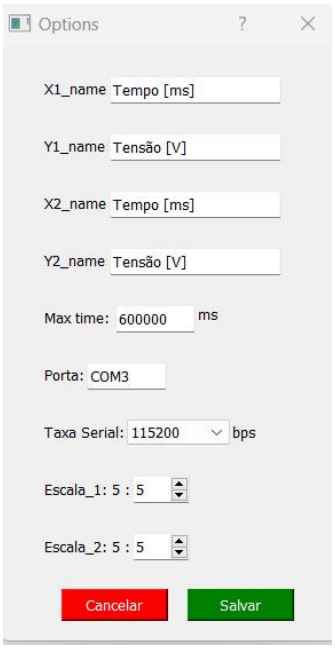

Figura 15: Janela de configurações.

- Play: responsável por efetuar a conexão com o microcontrolador através da porta informada na janela de configurações. Além disto, uma vez que a conexão é bem sucedida, a aquisição de dados e geração dos gráficos é iniciada.
- Pause: onde é possível pausar, por um momento desejado, a aquisição de dados e atualização dos gráficos.
- Stop: responsável por interromper o recebimento de dados via serial.
- Zoom: cujo objetivo é ampliar determinada sessão do gráfico a fim de uma melhor análise.
- Filtrar: responsável por eliminar os pontos fora da área ampliada pelo botão Zoom e inserir a origem do eixo horizontal no primeiro ponto dentro da área limitada.

Mais além, para facilitar a visualização dos dados e manter-se semelhante ao osciloscópio tradicional, os vetores de dados recebido são atualizados e plotados em tempo real no gráficos dos canais de leitura.

Desta forma, ocupando quase toda janela do *software*, como mostra a Figura 16, os gráficos de canais se assemelham à tela de um osciloscópio.

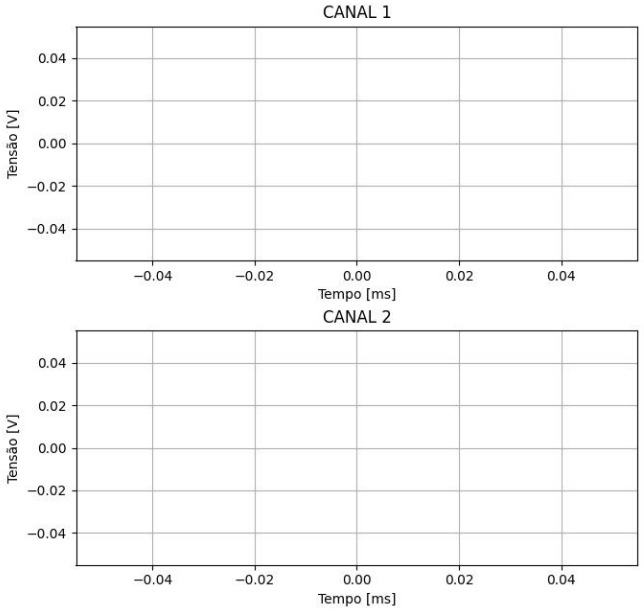

Figura 16: Canais do osciloscópio projetado.

C. *Circuito RC de teste*

Após o desenvolvimento da interface gráfica e da programação do microcontrolador, o osciloscópio de baixa custo já pode ser utilizado.

Contudo, é necessário verificar a eficiência do dispositivo projetado. Para isto, nada melhor do que realizar a leitura de um simples circuito resistor-capacitor e obter a equação da tensão do capacitor deste.

A escolha deste circuito se deu devido a sua facilidade de montagem, aquisição de componentes e simplicidade em se obter a equação de tensão sobre o capacitor através da regressão linear para casos exponenciais.

Devido a quantidade de teste a serem realizados, optou-se pela montagem de um circuito que permitisse a carga e descarga do capacitor através de *push-buttons*, como mostra a Figura 17.

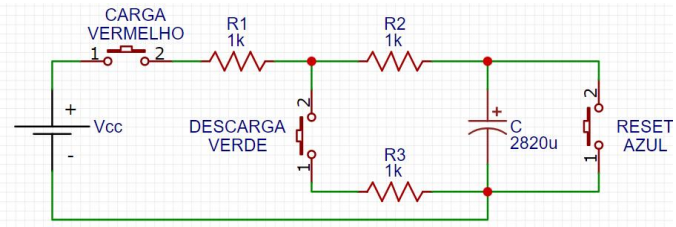

Figura 17: Esquemático do circuito RC de teste.

Ao analisar o o esquemático, pode-se afirmar que não há risco curto-circuito na fonte devido a presença do resistor *R1*. No entanto, o botão *RESET*, indicado pela cor azul irá curto-circuitar o capacitor, forçando sua descarga por imediato. Tal ação é prejudicial ao capacitor, e portanto deve ser sempre evitada.

Sobre os componentes, vale ressaltar que não há no mercado um capacitor cuja capacitância seja de 2820 µF, desta forma um associação de capacitores em paralelo é necessária. Sendo assim, optou-se por utilizar 6 capacitores de 470 µF, gerando o valor desejado, como mostra a Figura 18.

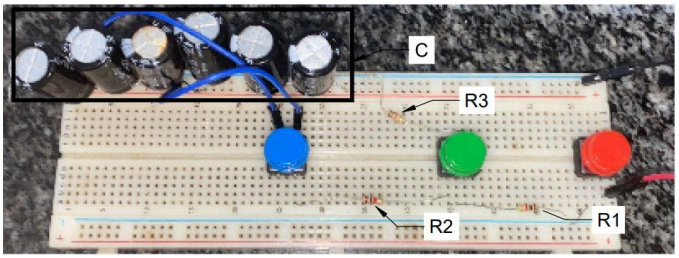

Figura 18: Circuito RC de teste.

Não obstante, o valor de resistência dos resistores *R1*, *R2* e *R3* são iguais a fim de manter a mesma constante de tempo, tanto para carga quanto para descarga, que para o caso, de acordo com a equação (19), será igual a:

$$\tau = RC = 2 * 10^3 * 2820 * 10^{-6} = 5,64s \qquad (19)$$

No entanto, é preciso considerar a incerteza presente entre os valores nominais e os reais dos componentes. Desta forma, com a ajuda de um multímetro, pode-se realizar a medição de resistência e capacitância, apresentadas na Tabela I

.

TABELA I
PARÂMETROS DOS COMPONENTES DO CIRCUITO RC DE TESTE

Componente	Valor teórico	Valor medido
R1	1 kΩ	0,987 Ω
R2	1 kΩ	0,991 Ω
R3	1 kΩ	0,979 Ω
C (único)	470 µF	511 µF
C (banco de 6)	2820 µF	3010 µF
τ (carga)	5,64 s	5,95 s

Mais além, uma vez definido o valor da constate de tempo para a carga do circuito, pode-se obter a equação (20) do circuito bem como o gráfico da curva de carga, como mostra a Figura 19.

$$V_c = 5(1 - e^{-\frac{t}{5,95}}) \qquad (20)$$

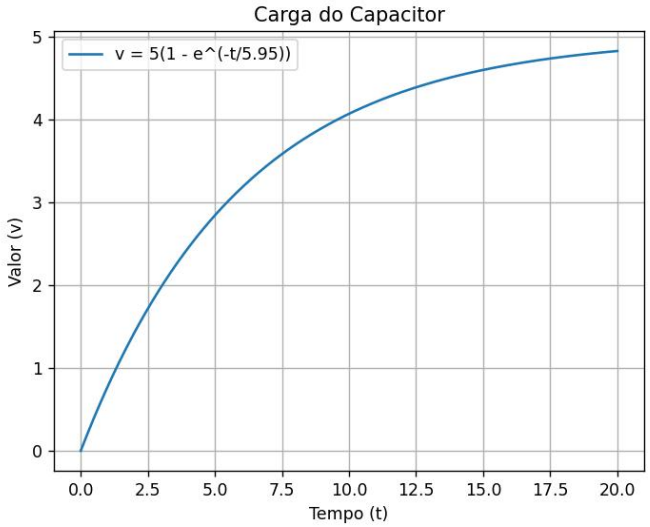

Figura 19: Carga do capacitor do circuito RC de teste.

D. Servo motor do laboratorio

Mais complexo que um simples circuito RC, o servo motor presente no Laboratório de Sistemas de Controle da Universidade Federal de Goiás, apresentado na Figura 20, é de extrema importância para os alunos de engenharia, pois é nele e para ele que são projetados e implantados controladores de malha.

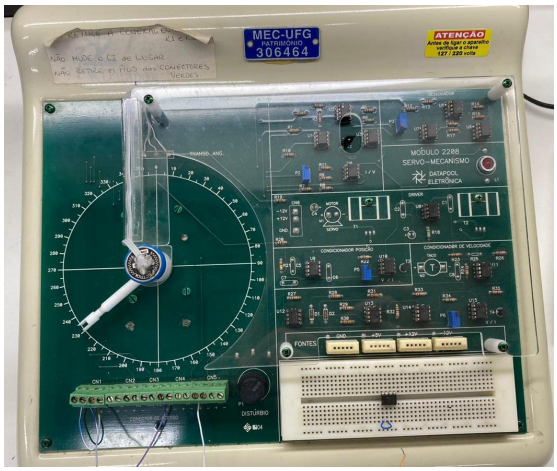

Figura 20: Servo motor do Laboratório de Sistemas de Controle (Sala H7) da Universidade Federal de Goiás - Bancada 01.

Contudo, para se elaborar um controlador é preciso, antes de tudo, conhecer a função de transferência do sistema. Para isto, o fabricante *DataPool* fornece, através do Manual Teórico do Equipamento, a relação, dada abaixo, entre a tensão de entrada e a velocidade do motor, representada pela tensão do pino *Vn* na saída do motor.

$$\frac{V_n}{V_u} = \frac{15,8}{0,058s^2 + 5,51s + 7,97} \qquad (21)$$

De posse da função de transferência do equipamento, obtém-se a resposta ao degrau de alimentação disponível no equipamento, como mostra a Figura 21.

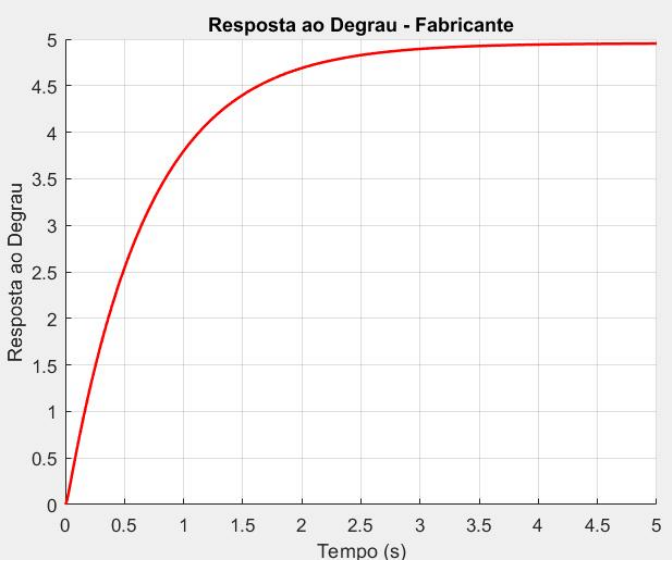

Figura 21: Resposta ao degrau da função de transferência fornecida pela fabricante.

Contudo, está modelagem do servo motor é apenas uma aproximação, de acordo com o fabricante. Sendo assim, é necessário uma análise mais aprofundada para obter uma função de transferência mais precisa para o sistema. Além disto, a calibragem do equipamento também afeta diretamente os coeficientes desta relação entre entrada e saída do sistema.

E. Leitura e análise de dados

Primordialmente, antes de iniciar a leitura de dados para o circuito RC, deve-se primeiramente garantir que o capacitor esteja totalmente descarregado. Para tal, a melhor forma de garantir é causando um curto-circuito em seus terminais, através do botão azul presente na placa.

Vale ressaltar que o curto-circuito de um capacitor gera uma variação abrupta de tensão causando um enorme pico de corrente, além de danificar o componente. Sendo assim, esta operação deve ser evitada sempre que possível.

Uma vez que o circuito está totalmente descarregado, pode-se conectar a placa Arduíno ao sistema. Como apresentado no APÊNDICE A, o pino 4 possui modo de saída com 5 V a fim de alimentar a placa do circuito RC, como mostra a Figura 22.

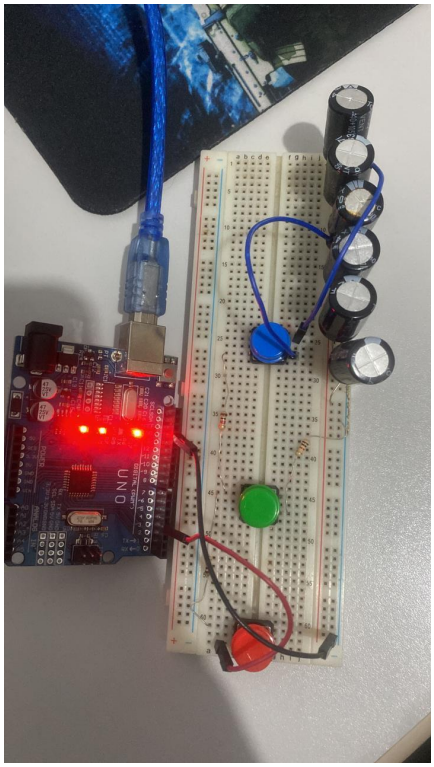

Figura 22: Conexão do microcontrolador com o circuito RC.

A Figura 22 também permite observa que a porta A0 (Canal 1) recebe a informação de tensão no capacitor e a porta A1 (Canal 2) recebe informações sobre o a tensão de entrada do circuito.

Desta forma é possível ler a tensão sobre o capacitor e salvar os valores em um arquivo *.m* que, posteriormente, poderão ser tratados e utilizados como entrada de uma regressão linear a fim de se obter a constante de tempo do circuito.

Já para o kit do servo motor, é necessário realizar a conexão do ponto comum entre o microcontrolador e o equipamento, visto que este é alimentado pelos circuito da instalação predial do Laboratório de Sistemas de Controle, como observado na Figura 23.

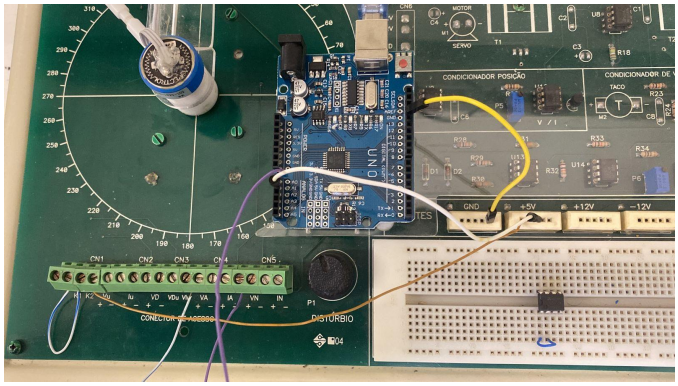

Figura 23: Conexão do microcontrolador com o servo motor

De forma análoga à conexão para o circuito RC, todos o *jumpers* possuem cores distintas, o que permite especificar sua função baseado nesta característica através da Tabela II.

TABELA II
ESPECIFICAÇÕES DOS *JUMPERS* PARA O SERVO MOTOR

Cor	Tipo	Especificação
Amarelo	Referência	GND (PONTO COMUM)
Marrom	Alimentação	Entrada de alimentação do motor
Roxo	Comunicação	Ponto de amostragem da porta A0
Branco	Comunicação	Ponto de amostragem da porta A1

Desta forma, torna-se possível realizar a conexão *software*-microcontrolador através de um computador, como mostra a Figura 24, e iniciar a aquisição de dados.

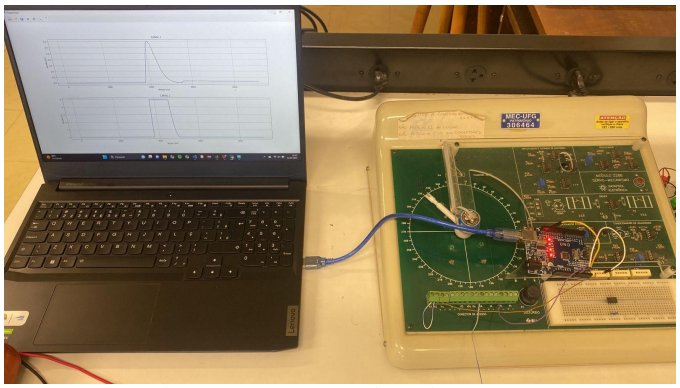

Figura 24: Conexão máquina-arduíno-motor

Uma vez que os dados já foram salvos em arquivos *.m*, é indispensável o uso da função *ARX(v , [z p a])* para se estimar a função de transferência, como mostra o APENDICE C, onde:

- *v*: matriz com valores de entrada e saída do sistema.
- *z*: números de zeros da função;
- *p*: números de pólos da função; e
- *a*: atrasos entre os valores.

Desta forma, basta converter o modelo de saída desta função para o formato desejado, através da função *th2tf()* e então realizar a transição de domínio discreto para contínuo, por meio da função *d2cm()*. Contudo, esta necessita de informações sobre o tempo de amostragem para gera um resultado mais próximo do real.

IV. RESULTADOS

A. *Circuito RC de teste*

Uma vez elaborada toda montagem do software e das conexões necessárias para a comunicação desde com o meio externo, através do Arduíno UNO R3, surge a pergunta: Os dados lidos representam realmente os valores reais?

Para responder a tal indagação, o circuito RC apresentado anteriormente foi usado para realizar um medida de teste do osciloscópio.

1) *Medições com Osciloscópio Padrão*

Inicialmente, ao circuito da Figura 18, aplica-se uma tensão de entrada e, com a ajuda do osciloscópio presente no Laboratório de Sistemas de Controle (Sala H7) da Escola de Engenharia Elétrica, Mecânica e de Computação pertencente a Universidade Federal de Goiás, foi analisa a curva da tensão no capacitores, apresentada na Figura 25.

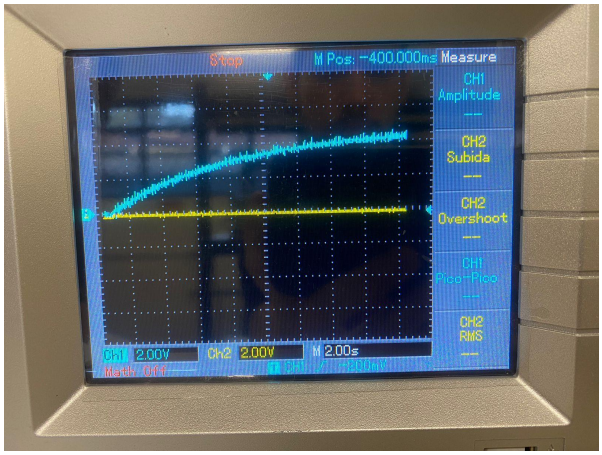

Figura 25: Tensão sobre os capacitores medido com osciloscópio padrão.

Pode-se analisar do gráfico que o tempo de subida é relativamente alto devido a elevada capacitância equivalente da associação encontrada no circuito.

Conhecendo a curva teórica da tensão do capacitor descrita pela equação (20) e apresentada na Figura 19, pode-se afirma que o circuito de teste, apresentado na Figura 18, realmente possui comportamento de circuito RC, logo é possível usá-lo para averiguar a veracidade dos valores lidos com o osciloscópio projetado.

É importante ressaltar a semelhança entre os gráficos das Figuras 19, 21 e 25. Todas apresentam o mesmo formado exponencial característico do circuito RC.

2) *Medições com Osciloscópio-Arduíno*

Para iniciar a leitura da tensão do capacitor com o software, é preciso configurá-lo com a taxa de comunicação serial e demais características encontradas na programação do arduíno como mostra a Figura 26.

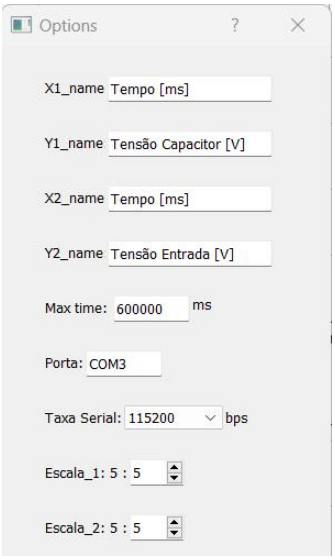

Figura 26: Configuração do software para leitura do circuito RC.

Para este caso em especifico, optou-se por ler a tensão de entrada na porta analógica A1, correspondente ao Canal 2, e a tensão sobre o capacitor na porta A0, correspondente ao Canal 1.

Desta forma, é possível ler valores de entrada e saída facilitando o processo de cálculo de funções de transferência por resposta ao degrau. Contudo, tal método não será abordado nesta etapa visto que há formas mais simplificadas de obter-se os parâmetros do circuito.

Uma vez que todas as conexões já foram realizadas e o arduíno alimentado através do cabo USB conectado na porta COM3 da máquina, inicia-se a leitura dos dados de carga dos capacitores gerando assim os gráficos apresentados nas Figuras 27 e 28.

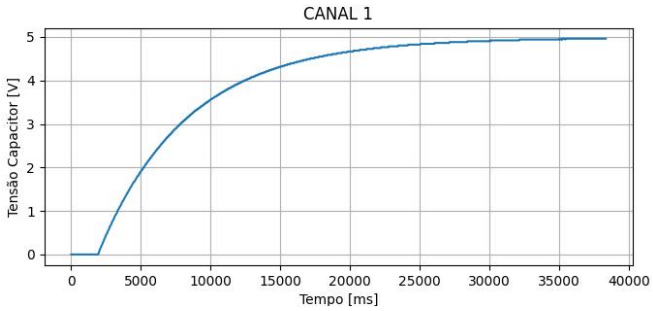

Figura 27: Tensão nos capacitores medida com o *software*.

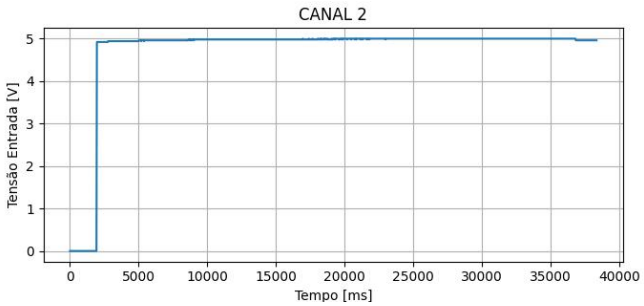

Figura 28: Tensão de entrada medida com o *software*.

É perceptível que a curva em questão apresenta alto nível de semelhança com as curvas apresentadas nas Figuras 19, 21 e 25.

Contudo para que se possa comprovar a eficiência do projeto, é necessário uma análise mais profunda dos dados obtidos com a medição.

3) Análise dos dados de medição do circuito RC

Para análise dos dados obtidos anteriormente, a leitura dos dois canais foram salvas em arquivos de extensão .m através do botão Salvar presente na barra de opções. Contudo, por estar representando apenas o valor de alimentação e pelo metodologia métrica a ser utilizada, o Canal 2 será desconsiderado neste tópico.

Sendo assim, com a ajuda do software MATLAB, pode-se realizar algumas métricas relacionadas ao desempenho do projeto, tais como:

a) Função da curva
b) Constante de tempo
c) Análise da taxa de amostragem

3.1) Função da curva

Como a curva obtida no Canal 1 (tensão nos capacitores) possui formato exponencial, a melhor opção para se obter a função que a gera é através de uma regressão linear para casos exponenciais.

Para isto, é preciso considerar a presença de valores nulos, gerados entre o momento em que se inicia a comunicação software-microcontrolador e o instante onde o *push-button* de carga no circuito é pressionado, que não fazem parte da curva. Desta forma, é necessário eliminá-los do vetor antes de realizar a regressão. Além disto, como já apresentado, não é possível utilizar esta ferramenta matemático quando o pontos são do tipo (x,0).

Além disto, surge um novo problema ao analisar o gráfico da Figura 27. Devido a impossibilidade de expressar a tensão real com sua infinidade de casas decimais, a conversão A/D se mantém em um valor constante por um período de tempo até que a variação na tensão seja tal que o conversor A/D possa observá-la. Este evento gera vários valores de tempo para um mesmo valor de tensão, como demonstrado na Tabela III.

TABELA III
EXEMPLO DE TENSÕES REPETIDAS DURANTE LEITURA

Tensão (V)	Intervalo (s)	Quantidade de pontos
3,4118	8,6139 - 8,7084	3
4,4314	14,1504 - 17,4160	7
4,6471	21,4084 - 21,8942	14
4,8431	28,7112 - 30,0399	32
4,9412	38,2176 - 40,6239	58

É possível observar que conforme a tensão no capacitor se aproxima de 5 V, há pontos com vários valores de tempo para um mesmo valor de v_c. Sendo assim, é conveniente tratar esses valores para que não ocorra esta repetição.

Uma vez resolvido estes empecilhos que poderiam prejudicar a obtenção da função da curva, e com a ajuda do código de implementação da regressão linear, encontra-se os parâmetros para a curva, sendo esta dada por:

$$V_c = 5(1 - e^{-0{,}1104t}) \quad (21)$$

Cujo gráfico obtido pode ser observado na Figura 29 e na Figura 30.

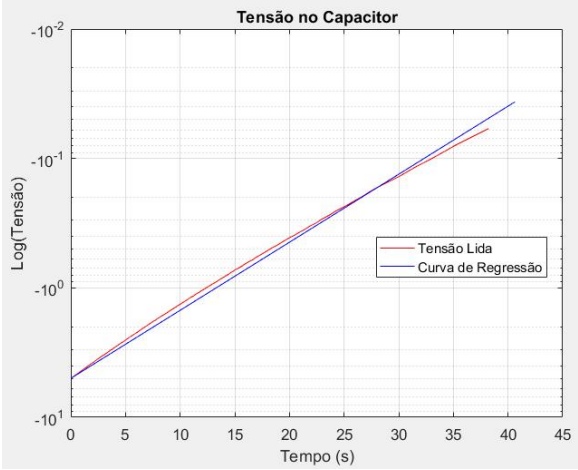

Figura 29: Resultado da regressão linear para o circuito RC em escala semilogarítmica.

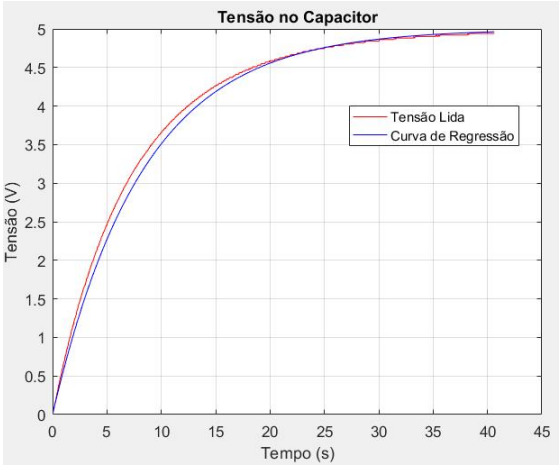

Figura30: Resultado da regressão linear para o circuito RC.

3.2) Constante de tempo

Uma vez obtida a função através da regressão linear para casos exponenciais, pode-se estimar a tensão no circuito para qualquer instante de tempo. Além disto, obtém-se os parâmetros do circuitos, entre os quais esta a constante de tempo.

Como a equação da tensão do capacitor para um circuito RC qualquer é dada pela equação (9). Desta forma, nota-se, através da função (21) que:

$$\tau = \frac{1}{0,1104} = 9,05798\ s \quad (22)$$

Comparada ao valor téorico de

$$\tau = 5,95\ s \quad (23)$$

Desta forma, o resultado obtido através da regressão apresenta erro percentual de

$$E(\%) = \frac{|\tau_{teórico} - \tau|}{\tau_{teórico}} = \frac{|5,95 - 9,05798|}{5,95} = 52,25\%$$

3.3) Análise da taxa de amostragem

Baseado no código instalado no microcontrolador Arduíno, pode-se afirmar que a taxa de amostragem é restringida pelo *bound-rate* já pré-estabelecido e pela velocidade de conversão do conversor A/D e devido a característica de convergência deste, onde o tempo de conversão varia, não pode-se obter uma taxa de amostragem fixa e pré-definida.

Desta forma, faz-se necessário uma análise estatística dos intervalos de tempo entre valores lido/enviados pelo microcontrolador. Para isto, elaborou-se o gráfico da diferença de tempo entre as amostras, como mostra as Figuras 31 e 32.

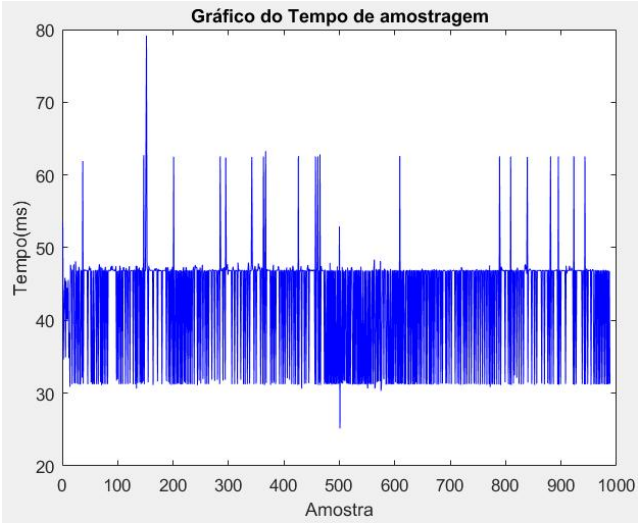

Figura 31: Tempo entre amostras do circuito RC.

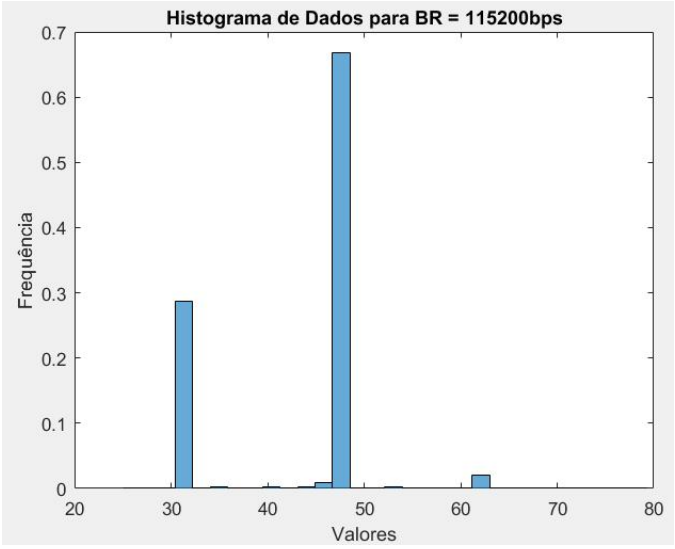

Figura 32: Histograma do tempo entre amostras do circuito RC.

O gráfico da Figura 32 revela que, em sua maior parte, os intervalos de tempo para cada amostra durante todo o experimento mantiveram-se dentro da faixa de 31 a 47 milissegundos para um bound-rate de 115200bps. Algumas exceções podem ser observadas, mas em geral, o comportamento das amostras permaneceu dentro desses limites.

Para uma análise mais detalhada desses intervalos de

amostragem, diversas métricas foram aplicadas e os resultados foram compilados na Tabela IV. As métricas utilizadas incluem a média, moda, mediana, erro padrão, variância, desvio padrão, amplitude, quartis, coeficiente de variação, curtose e assimetria.

Essas métricas oferecem uma análise completa dos intervalos de amostragem obtidos, permitindo uma melhor compreensão do comportamento desses dados. Essas informações são fundamentais para a interpretação dos resultados do experimento e para auxiliar na tomada de decisões em relação ao sistema em estudo.

TABELA IV
MÉTRICAS DOS INTERVALOS DE AMOSTRAGEM PARA BR = 115200 BPS

Métrica	Valor
Média	42,71
Moda	46,88
Mediana	46,87
Variância	60,26
Desvio Padrão	7,76
Erro Padrão	0,25
Amplitude	54,02
Quartis (Q1 ; Q3)	(31,47 ; 46,89)
Coeficiente de variação	18,17%
Curtose	2,85
Assimetria	-0,36

A Tabela IV resume as métricas da análise de intervalos de amostragem, proporcionando uma visão geral dos principais parâmetros e estatísticas relevantes. Essas informações são valiosas para a validação dos resultados do experimento e podem subsidiar futuras otimizações e aprimoramentos no sistema analisado.

Com base nessas análises, pode-se concluir que os intervalos de amostragem mantiveram-se em geral estáveis e adequados para as configurações de *bound-rate* adotadas, mas é importante ficar atento às exceções identificadas e considerá-las no contexto das aplicações do sistema.

Não obstante, para diferentes valores de *bound-rate* surgem diferentes valores nas métricas apresentadas como apresentado nas Figuras 33 a 36 e nas Tabelas V e VI.

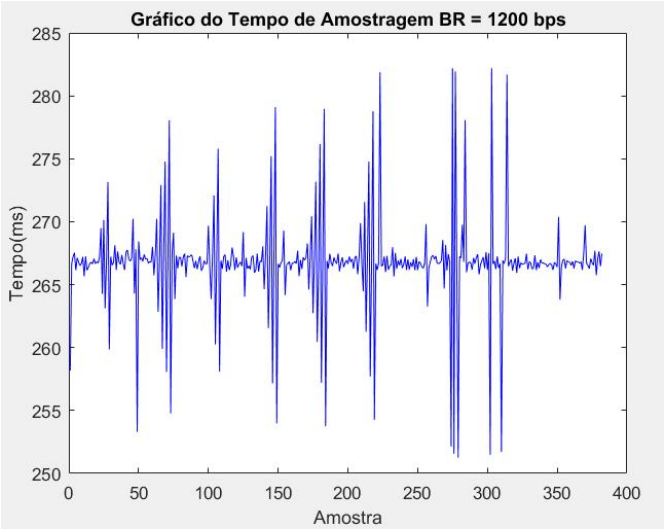

Figura 33: Tempo entre amostras do circuito RC com bound-rate de 1200 bps.

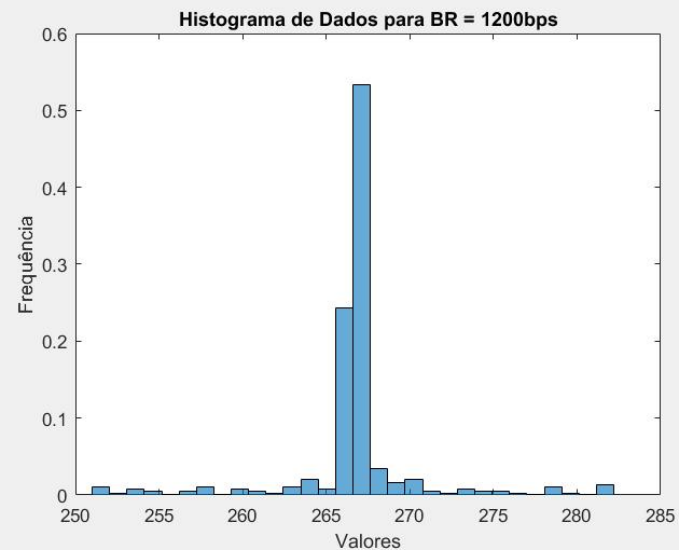

Figura 34: Histograma do tempo entre amostras do circuito RC com *bound-rate* de 1200 bps.

TABELA V
MÉTRICAS DOS INTERVALOS DE AMOSTRAGEM PARA BR =1200 BPS

Métrica	Valor
Média	266,75
Moda	266,75
Mediana	266,78
Variância	14,28
Desvio Padrão	3,78
Erro Padrão	0,19
Amplitude	30,93
Quartis (Q1 ; Q3)	(266,32 ; 267,22)
Coeficiente de variação	1,42%
Curtose	10,96
Assimetria	-0,15

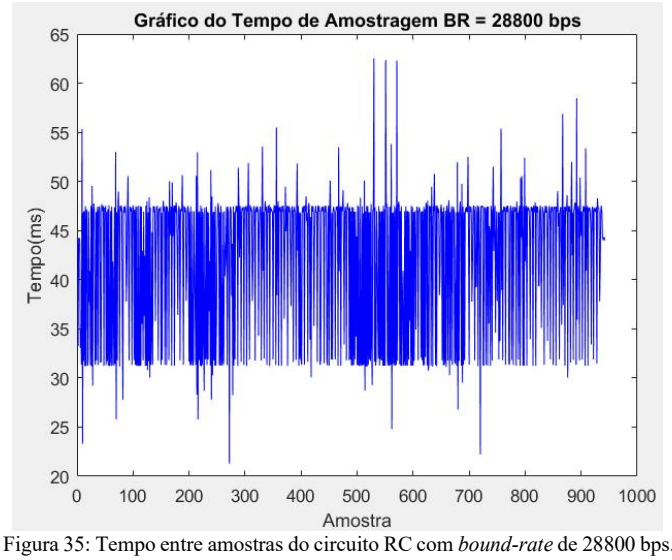

Figura 35: Tempo entre amostras do circuito RC com *bound-rate* de 28800 bps.

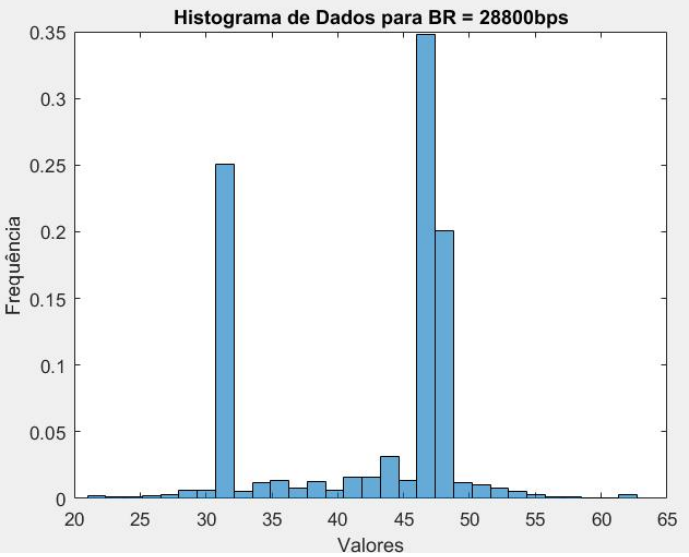

Figura 36: Histograma do tempo entre amostras do circuito RC com *bound-rate* de 28800 bps.

TABELA VI
MÉTRICAS DOS INTERVALOS DE AMOSTRAGEM PARA BR =28800 BPS

Métrica	Valor
Média	42,11
Moda	46,88
Mediana	46,87
Variância	55,36
Desvio Padrão	7,44
Erro Padrão	0,24
Amplitude	41,23
Quartis (Q1 ; Q3)	(31,87 ; 47,4)
Coeficiente de variação	17,67%
Curtose	1,96
Assimetria	-0,61

Os gráficos apresentados acima permitem afirmar que uma maior taxa de comunicação permite maior confiabilidade e uma taxa de amostragem mais uniforme durante a aquisição de dados.

Dessa forma, a análise dos intervalos de amostragem apresenta-se como um passo fundamental na compreensão e avaliação durante a utilização do *software*, contribuindo significativamente para a qualidade e a confiabilidade dos resultados obtidos neste estudo.

3.4) Validação do projeto

Após análise de todos os dados obtidos para o circuito RC de teste, pode-se afirmar com segurança que o projeto de osciloscópio é confiável e estável o suficiente para ser utilizado em casos reais e de necessidade. Os resultados obtidos das métricas e estatísticas revelaram um comportamento consistente dos intervalos de amostragem, mantendo-se dentro dos limites esperados para um *bound-rate* de 115200bps.

As métricas, como a média, a moda, a mediana, o erro padrão, a variância, o desvio padrão, a amplitude, os quartis, o coeficiente de variação, a curtose e a assimetria, ofereceram uma visão abrangente e robusta do desempenho do sistema. A distribuição dos dados próxima a uma curva gaussiana e a baixa variabilidade indicam que o osciloscópio é capaz de fornecer medições precisas e confiáveis.

Essa confiabilidade do osciloscópio permite o avanço com segurança para o próximo passo do estudo, que é usar o projeto para ler a resposta ao degrau de um kit servo motor no Laboratório de Sistemas de Controle. Com base nos dados coletados até o momento, espera-se que o osciloscópio seja uma ferramenta eficiente para capturar e analisar a resposta do servo motor.

B. Kit servo-motor

Como já apresentado anteriormente, o Laboratório de Sistemas de Controle possui em suas bancadas um kit servo motor para que os alunos desenvolvam formas de controlá-lo por meio de circuitos eletrônicos.

Devido a seus objetivos didáticos, o fabricante do servo motor disponibiliza junto ao dispositivo um manual de instruções onde é possível obter a função de transferência entre a tensão de alimentação e a posição do motor. Sendo assim, ao obter a resposta ao degrau do kit servo motor, será possível utilizar o MATLAB para realizar uma análise mais aprofundada e obter a função de transferência do sistema.

Assim, a confiabilidade do osciloscópio comprovada pelo circuito RC de teste, garante a veracidade em sua aplicação prática na leitura da resposta ao degrau do kit servo motor e na obtenção da função de transferência por meio das ferramentas oferecidas pelo MATLAB.

1) Medições com Osciloscópio Padrão

Antes de iniciar as medições de entrada e saída do kit com o osciloscópio Arduíno, foi realizada uma medição preliminar da saída de velocidade servo motor, apresentada na Figura 37, utilizando um osciloscópio padrão disponível na bancada onde o equipamento está localizado.

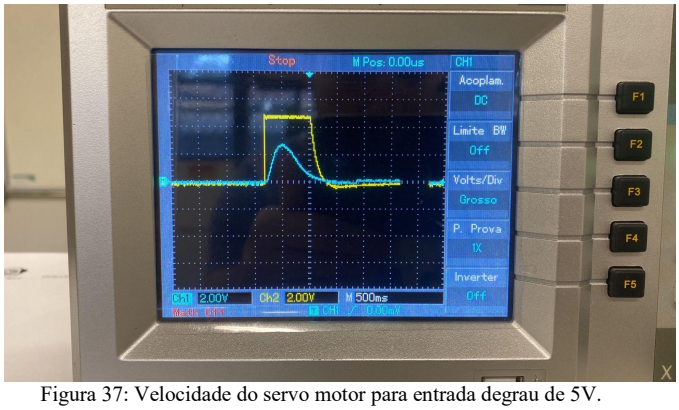

Figura 37: Velocidade do servo motor para entrada degrau de 5V.

Caso a distribuição de pontos fosse disponibilizada por este osciloscópio, seria possível estimar a função de transferência entre as variáveis do sistema, contudo, tal ação só será possível com a leitura de dados através do osciloscópio desenvolvido por este projeto.

2) Medições com Osciloscópio-Arduíno

Para iniciar a leitura da tensão que representa a velocidade do servo motor com o software, é necessário configurá-lo com a taxa de comunicação serial e demais características encontradas na programação do arduíno.

Para manter a taxa de amostragem em uma faixa relativamente constante e de baixa amplitude, optou-se por um *bound-rate* igual a 115200 bps, como mostra a Figura 38.

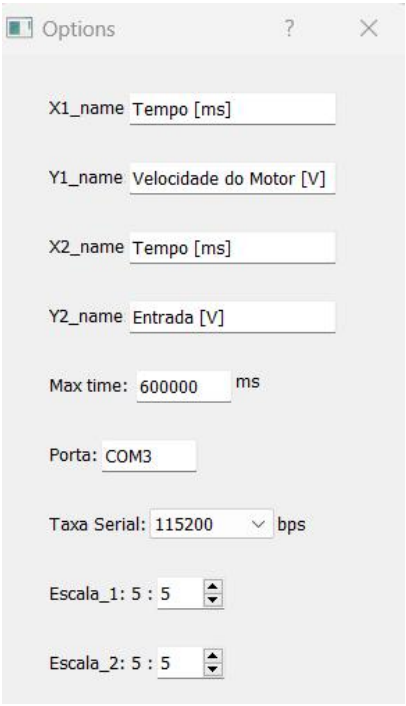

Figura 38: Configuração do software para leitura do servo motor

.

Através da Tabela III e da Figura 23, percebe-se que a tensão de entrada, na forma de degrau, será observado na porta analógica A1, corresponde ao Canal 2, enquanto a tensão de velocidade do motor (Vn) será observada na porta analógica A0, correspondente ao Canal 1.

Divergente da análise de teste utilizando o circuito RC, para o caso do motor, a aquisição de dados referentes ao degrau unitário de entrada é essencial para que seja possível a obtenção da função de transferência desejada.

Uma vez que todas as conexões já foram realizadas e o arduíno alimentado através do cabo USB conectado na porta COM3 da máquina, como apresenta a Figura 24.

A leitura dos dados é iniciada por meio do microcontrolador, e em seguida, um degrau é gerado na entrada do sistema, conforme as Figuras 39 e 40.

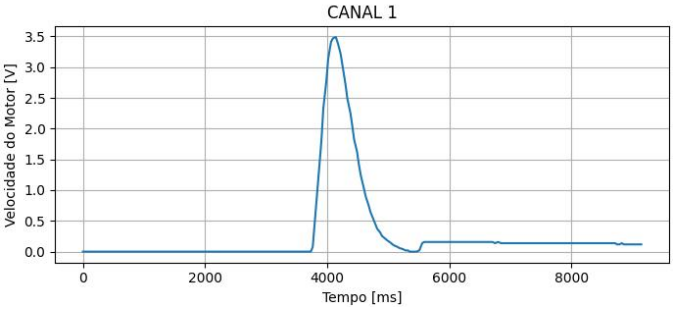

Figura 39: Velocidade do servo motor a uma entrada degrau.

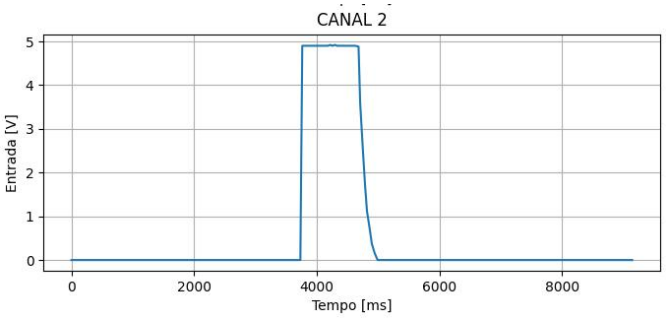

Figura 40: Entrada degrau de entrada do servo-motor.

A Figura 40 ilustra a aplicação de uma entrada degrau de curta duração no sistema do motor. Essa escolha de entrada, semelhante a um impulso, é adotada para capturar rapidamente a resposta do sistema após uma mudança abrupta no sinal de entrada. Esse tipo de entrada é fundamental para identificar e modelar sistemas de resposta rápida, como é o caso, permitindo uma análise precisa e detalhada de suas características dinâmicas em um intervalo específico de tempo. Além disso, o método utilizado pelo modelo de aquisição da função de transferência fornecido pelo fabricante pode requerer a aplicação de um degrau curto para garantir a obtenção de dados confiáveis sobre a resposta do sistema. As Figuras 39 e 40 representam, assim, um ponto crucial no processo de obtenção da função de transferência, abrindo caminho para uma análise mais aprofundada e precisa da dinâmica do sistema do motor.

Desta forma, com base nos dados de leitura obtidos, é possível realizar uma análise para estimar a função de transferência.

3) Análise de dados do servo motor

Com o objetivo de obter a relação entre entrada e saída, o MATLAB fornece a função ARX. Contudo, esta poderosa ferramenta de identificação de sistemas irá analisar todos os pontos obtidos durante a aquisição de dados. Desta forma, assim como ocorreu no circuito RC de teste, os resultados da leitura do kit possuem valores iniciais e finais nulos. Logo é preciso eliminá-los antes de utilização a função desejada.

Com este objetivo em mente e utilizando os recursos de Zoom e Filtro na interface gráfica do projeto, é possível realizar um tratamento preliminar dos dados coletados, permitindo isolar o intervalo desejado da saída e, consequentemente, da entrada correspondente, como mostra a Figura 41 e 42. Essas funcionalidades possibilitam uma análise mais precisa e detalhada do comportamento do sistema, tornando a visualização e seleção de trechos específicos dos dados mais eficientes.

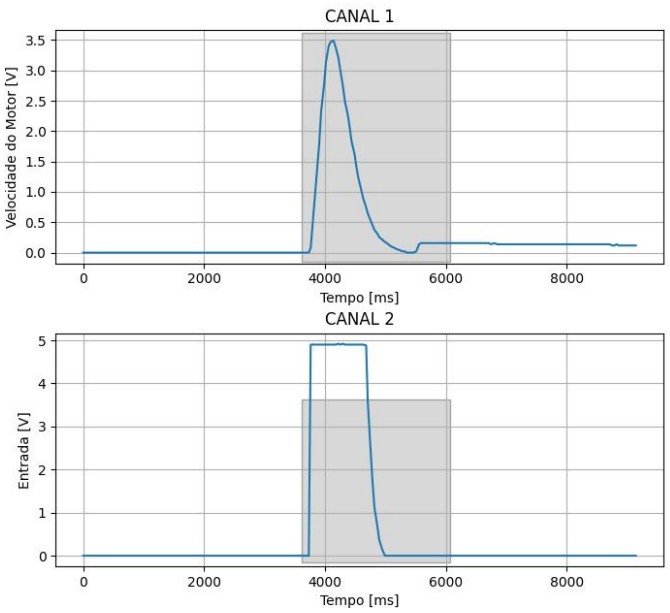

Figura 41: Seleção da faixa de zoom da leitura do servo motor.

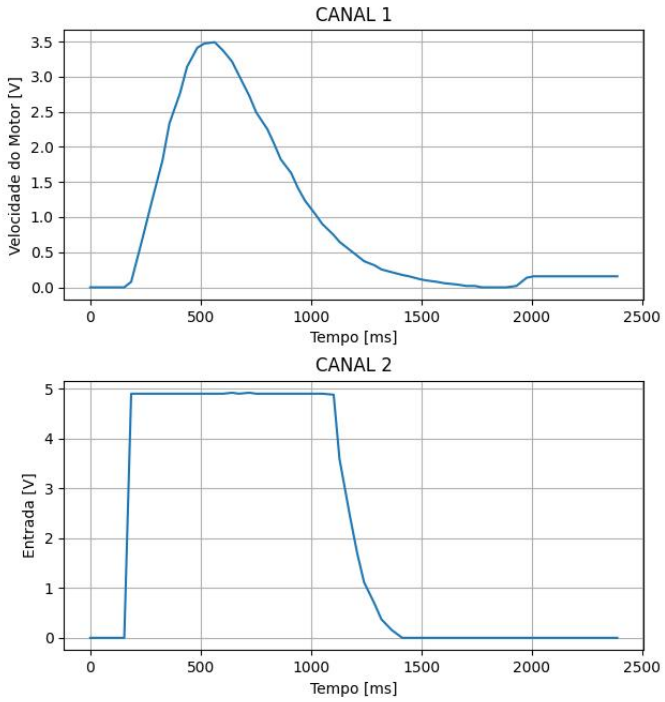

Figura 42: Dados após filtragem da caixa de seleção do zoom..

O primeiro passo para a análise dos dados obtidos com o osciloscópio projetado é sempre a observação da taxa de amostragem, visto a alta importância e impacto que esta possui sobre todo o sistema.

Devido a presença de dois canais distintos, é necessário gerar dois gráficos para avaliar as características das duas taxas de amostragem como mostra as Figuras 43 a 46.

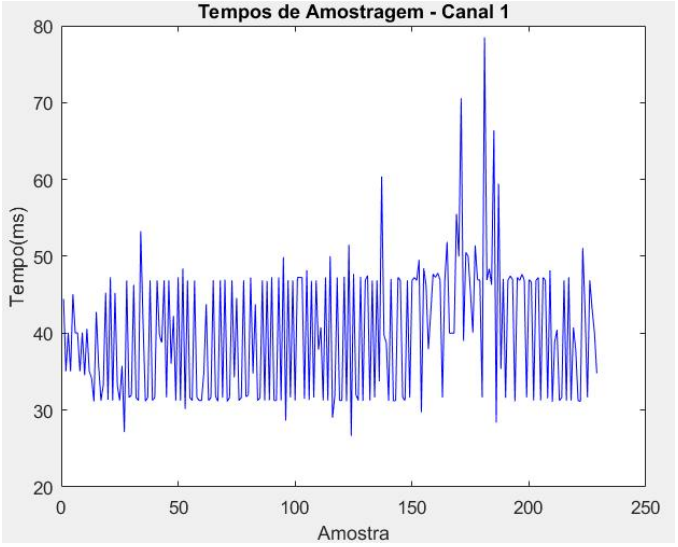

Figura 43: Tempo entre amostras do canal 1 para o servo motor.

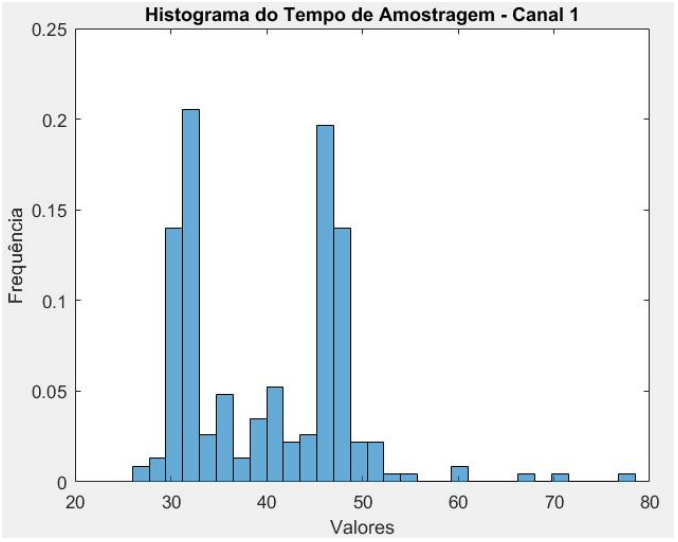

Figura 44: Histograma do tempo entre amostras do canal 1 para o motor.

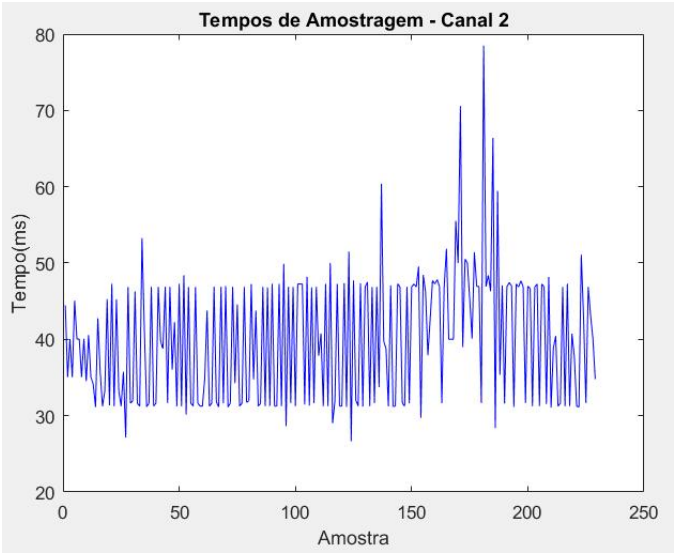

Figura 45: Tempo entre amostras do canal 2 para o servo motor

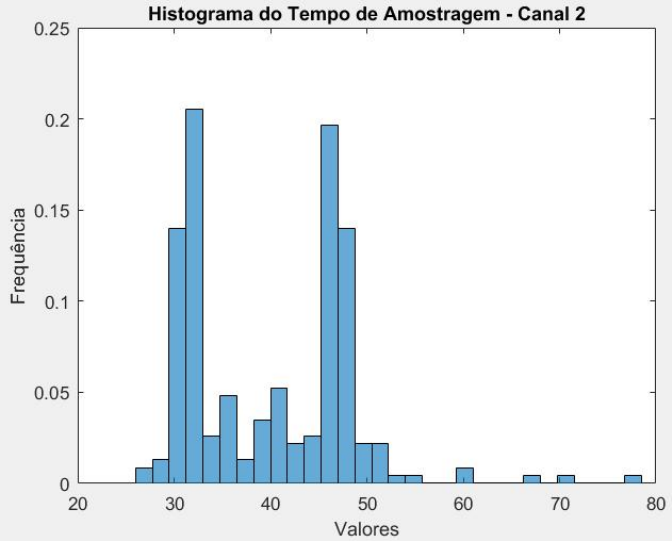

Figura 46: Histograma do tempo entre amostras do canal 2 para o motor.

As Figuras 43 e 45 demonstram o mesmo gráficos, isto ocorre devido a alta taxa de comunicação adotada, tornando o atraso causado pelo envio dos *bits* serem praticamente desprezíveis, restando somente o atraso do conversor A/D juntamente com o atraso causado pela preferência de tarefas executadas pelo computador usado durante a medição.

Sendo assim, os dados métricos estatísticos são apresentados na Tabela VII.

TABELA VII
MÉTRICAS DOS INTERVALOS DE AMOSTRAGEM (SERVO-MOTOR)

Métrica	Valor
Média	40,07
Moda	47,27
Mediana	40,05
Variância	72,79
Desvio Padrão	8,53
Erro Padrão	0,56
Amplitude	51,87
Quartis (Q1 ; Q3)	(31,62 ; 46,89)
Coeficiente de variação	21,29%
Curtose	4,01
Assimetria	0,71

Ao comparar as Tabelas V e VII, pode-se constatar que a distribuição dos intervalos de amostragem sofreu leves alterações durante a medição da velocidade do motor, excepcionalmente em relação a assimetria. Apesar disto, ainda é possível dizer que o osciloscópio projeto é capaz de realizar a leitura de tais dados com uma boa precisão e a função de transferência a ser obtida através deste é uma boa aproximação do modelo real do servo motor.

4) Função de transferência do servo motor

De posse agora de todos os dados de entrada e saída do sistema e com o auxílio da função *ARX* do MATLAB, a estimativa da função de transferência entre as variáveis torna-se possível.

O primeiro passo para a aquisição desta relação, como mostra o APÊNDICE C, é gerar uma matriz *v* com os valores de entrada e saída para que este possa ser inserido na função *ARX*. Além disto, escolheu-se o vetor *[2 2 1]* como sendo os parâmetros desejados na estimativa da função de transferência, obtendo a função de transferência:

$$G(s) = \frac{V_n(s)}{V_u(s)} = \frac{15,55s + 214,6}{s^2 + 41,72s + 139,6} \qquad (24)$$

Sabendo que a função transferência fornecida pelo fabricante do servo motor é igual a:

$$G_0(s) = \frac{V_n(s)}{V_u(s)} = \frac{15,8}{0,058s^2 + 5,51s + 7,97} \qquad (25)$$

Percebe-se uma grande diferença entre os coeficientes de *G(s)* e *G₀(s)*. Desta forma, a normalização da função estimada pelo MATLAB pelo coeficiente de s^0 do denominador é uma das melhores opções para resolver tal empecilho.

$$G_n(s) = \frac{\frac{7,97}{139,6}}{\frac{7,97}{139,6}} G(s) = \frac{0,8882s + 12,26}{0,05711s^2 + 2,382s + 7,97} \qquad (26)$$

Agora, é possível perceber uma melhor aproximação entre os coeficientes duas funções como mostra a Tabela VIII.

TABELA VIII
ERRO ENTRE OS COEFICIENTE DAS FUNÇÕES DE TRANSFERÊNCIA.

Coeficiente	Valor Fabricante	Valor Estimado	Erro Relativo (%)
N_a	0	0,8882	-
N_b	15,8	12,26	21,41
D_a	0,058	0,05711	0,089
D_b	5,51	2,382	56,77
D_c	7,97	7,97	0

Outra forma de observar esta diferença entre as funções é através da resposta destas ao degrau de amplitude 5 juntamente com o valor real lido com o osciloscópio projetado, como mostra a Figura 47.

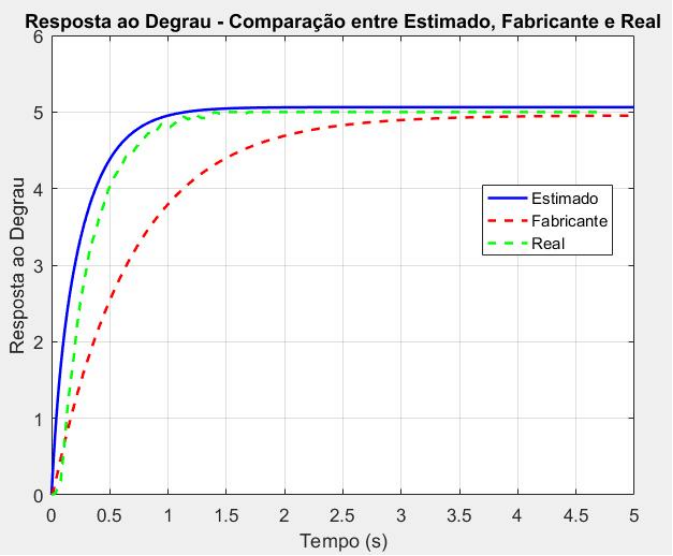

Figura 47: Respostas do servo motor.

Da Figura 47, pode-se assumir que a função de transferência

obtida através do osciloscópio projeto é significativamente mais próxima da realidade do que a modelagem fornecida pelo fabricante.

V. Conclusão

No decorrer deste Projeto Final de Curso, foi possível projetar, elaborar, validar e aplicar um osciloscópio de baixo custo e com ótima precisão de leitura de dados. Além disto, desenvolveu-se também um *software* de comunicação serial com o microcontrolador através da conexão deste com o computador a ser utilizado durante a medição, onde os dados são evitados serialmente e exibidos no gráfico de seu respectivo canal. Desta forma, alcançou-se o objetivo de apresentar, em tempo real, a forma de onda lida em cada canal.

Este dispositivo representa uma inovação na área acadêmica, uma vez que não há nenhum outro igual disponível na literatura. Vale ressaltar que há aqueles que realizam a amostragem, transmissão e visualização dos dados, porém não possuem as possibilidades de manuseio aqui desenvolvidas.

Nesse sentido, a ferramenta desenvolvida por este projeto torna-se inestimável tanto para a área acadêmica quanto para a aplicação prático em sistemas eletrônicos e de controle.

Através da medição da constante de tempo de um circuito RC, foi possível validar a eficiência do projeto, permitindo que os resultados de aplicações práticas possam ser utilizados em análise mais aprofundadas, como, por exemplo, a medição e obtenção da função de transferência do servo motor presente no Laboratório da Sistemas de Controle da Universidade Federal de Goiás, analisado no decorrer do projeto.

Para este caso, conclui-se que a relação entre variáveis disponibilizada pelo fabricante, apesar de próxima, não representa fielmente a saída do sistema. Desta forma, pode-se afirma, com base nas análises realizadas, que para as próximas aulas experimentais utilizando o servo motor do laboratório, melhores resultados poderão ser obtidos ao se considerar a função de transferência obtida através deste projeto. Com base nisto, será possível melhores didática e resultados para as aplicações realizadas pelos discentes da universidade.

Sendo assim, o desenvolvimento deste osciloscópio de baixo custo reforça a importância da instrumentação precisa da engenharia, principalmente na elétrica, onde não é possível determinar visualmente a presença de tensão e/ou corrente. Mais além, ressalta a ampla área de atuação de microcontroladores, em especial o Arduíno utilizado neste projeto, permitindo a realização de boas taxas de amostragem, ótima precisão de dados e excelente eficiência quanto a comunicação serial.

Portanto, este projeto serve como um exemplo concreto de como a teoria e a prática podem ser integradas de maneira eficiente e produtiva, enriquecendo a formação dos envolvidos e contribuindo para o progresso do conhecimento e da aplicação da engenharia no mundo real, além de fornecer um equipamento cujo valor de mercado é extremamente elevado, por um preço acessível aos necessitados.

Referência Bibliográfica

[1] SEDRA, S.; SMITH, K.. Microeletrônica. 4ª. Edição, Pearson Makron Books, São Paulo, Brasil, 2005.

[2] TOCCI, Ronald J.; WIDMER, Neal S.; MOSS, Gregory L..Sistemas Digitais: Princípios e Aplicações. 10ª ed. São Paulo: Pearson, 2007. 830 p.

[3] Ogata, K. - Modern Control Engineering, Prentice-Hall, 4th ed., 2001.

[4] PEREIRA, Fábio. Microcontroladores PIC: Programação em C. São Paulo:

[5] Érica, 2003. 360 pLATHI, B. P. – Sinais e Sistemas Lineares – 2ª Edição, Bookman. (Livro Texto)

[6] BRAIN Marshall, O tubo de raio catódico. Disponível em: <http://tecnologia.hsw.uol.com.br/televisao3.htm>. Acesso em 18 de julho de 2023.

[7] Luiz E. S. Oliviera, Regressão, Notas de Aulas, DInf / UFPR, 2017.

[8] FREGNI, E.; SARAIVA, A. M. Engenharia do Projeto Lógico Digital: Conceitos e Prática. Editora Edgard Blücher, 1995.

[9] BANZI, Massimo. Getting Started with Arduino. California: O'Reilly Media/Make, 2011.

[10] BRAGA, Newton C. Osciloscópio: Primeiros Passos. 1 ed. São Paulo: Newton C. Braga., 2014.

[11] MAZIDI, Muhammad A.; NAIMI, Sarmad; NAIMI, Sepehr. The AVR MICROCONTROLLER AND EMBEDDED SYSTEM: Using assembly and c. New Jersey: Prentice Hall, 2011.

Apêndice a - Código do microcontrolador

```
int analogPin0 = A0;  // Pino analógico utilizado
int analogPin1 = A1;  // Pino analógico utilizado
void setup() {
  Serial.begin(115200);  // Inicia a comunicação serial1
  pinMode(4, OUTPUT);
  digitalWrite(4,1);
}

void loop() {
  // Lê o valor analógico do pino A0
  int analogValue0 = analogRead(analogPin0);

  // Mapeia a faixa de valores de tensão (0-1023) para a faixa
de byte (0-255)

  byte byteValue0 = map(analogValue0, 0, 1023, 0, 255);
```

```
  // Lê o valor analógico do pino A0
  int analogValue1 = analogRead(analogPin1);

  // Mapeia a faixa de valores de tensão (0-1023) para a faixa
de byte (0-255)
  byte byteValue1 = map(analogValue1, 0, 1023, 0, 255);

  // Envia o valor convertido para um byte pela porta serial
  Serial.write(byteValue0);


  // Envia o valor convertido para um byte pela porta serial

  Serial.write(byteValue1);
}
```

APÊNDICE B - CÓDIGO DA INTERFACE COM O USUÁRIO

```
import sys
import os
import serial
import matplotlib.pyplot as plt
from matplotlib.backends.backend_qt5agg import
FigureCanvasQTAgg as FigureCanvas
from matplotlib import patches
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QFileDialog
from datetime import datetime
import serial.serialutil
import numpy as np

port = None
x1_name = "Tempo [ms]"
x1_name_temp = ""
y1_name = "Tensão [V]"
y1_name_temp = ""
x2_name = "Tempo [ms]"
x2_name_temp = ""
y2_name = "Tensão [V]"
y2_name_temp = ""
time_stop = "600000"
time_stop_temp = ""
taxa_serial = 115200
taxa_serial_temp = ""
porta = "COM3"
porta_temp = ""
escala1 = 5
escala_temp1 = ""
escala2 = 5
escala_temp2 = ""
class OptionsDialog(QtWidgets.QDialog):
    def __init__(self, grafico):
        super().__init__(None)
        self.grafico = grafico
        self.setFixedSize(300, 520)
        self.setModal(True)
        self.setWindowTitle("Options")
        # Criar o QLabel "X_name"
        label_x1_name = QtWidgets.QLabel("X1_name:", self)
        label_x1_name.setFixedWidth(55)
        label_x1_name.move(40, 30)
        # Criar o QLineEdit
        text_field_x1 = QtWidgets.QLineEdit(self)
        text_field_x1.setFixedWidth(150)
        text_field_x1.move(label_x1_name.x() +
label_x1_name.width() + 2, label_x1_name.y() - 3)
        # Criar o QLabel "Y_name"
        label_y1_name = QtWidgets.QLabel("Y1_name:", self)
        label_y1_name.setFixedWidth(label_x1_name.width())
        label_y1_name.move(label_x1_name.x(),
label_x1_name.y() + label_x1_name.height() + 20)
        # Criar o QLineEdit
        text_field_y1 = QtWidgets.QLineEdit(self)
        text_field_y1.setFixedWidth(text_field_x1.width())
        text_field_y1.move(label_y1_name.x() +
label_y1_name.width() + 2, label_y1_name.y() - 3)

        # Criar o QLabel "Y_name"
        label_x2_name = QtWidgets.QLabel("X2_name:", self)
        label_x2_name.setFixedWidth(label_x1_name.width())
        label_x2_name.move(label_x1_name.x(),
label_y1_name.y() + label_y1_name.height() + 20)
        # Criar o QLineEdit
        text_field_x2 = QtWidgets.QLineEdit(self)
        text_field_x2.setFixedWidth(text_field_x1.width())
        text_field_x2.move(label_x2_name.x() +
label_x2_name.width() + 2, label_x2_name.y() - 3)

        # Criar o QLabel "Y_name"
        label_y2_name = QtWidgets.QLabel("Y2_name:", self)
        label_y2_name.setFixedWidth(label_x1_name.width())
        label_y2_name.move(label_x1_name.x(),
label_x2_name.y() + label_x2_name.height() + 20)
        # Criar o QLineEdit
        text_field_y2 = QtWidgets.QLineEdit(self)
        text_field_y2.setFixedWidth(text_field_x1.width())
        text_field_y2.move(label_y2_name.x() +
label_y2_name.width() + 2, label_y2_name.y() - 3)
        # Criar o QLabel "Max time"
        label_max_time = QtWidgets.QLabel("Max time:", self)
        label_max_time.setFixedWidth(label_x2_name.width()
+ 5)
        label_max_time.move(label_x2_name.x(),
label_y2_name.y() + label_y2_name.height() + 20)
        # Criar o QLineEdit
        text_field_max_time = QtWidgets.QLineEdit(self)
        text_field_max_time.setFixedWidth(70)
        text_field_max_time.move(label_max_time.x() +
label_max_time.width() + 2, label_max_time.y() - 3)
        text_field_max_time.setValidator(QtGui.QIntValidato
r())
        # Criar o QLabel "ms"
        label_ms = QtWidgets.QLabel("ms", self)
        label_ms.setFixedWidth(label_x2_name.width())
        label_ms.move(text_field_max_time.x() +
text_field_max_time.width() + 2, label_max_time.y() - 3)
        # Criar o QLabel "Porta"
        label_porta = QtWidgets.QLabel("Porta:", self)
```

```
        label_porta.setFixedWidth(35)
        label_porta.move(label_max_time.x(),
label_max_time.y() + label_max_time.height() + 20)
        # Criar o QLineEdit
        text_field_porta = QtWidgets.QLineEdit(self)
        text_field_porta.setFixedWidth(70)
        text_field_porta.move(label_porta.x() +
label_porta.width() + 2, label_porta.y() - 3)
        # Criar o QLabel "Taxa Serial"
       label_taxa_serial = QtWidgets.QLabel("Taxa Serial:",
self)
        label_taxa_serial.setFixedWidth(70)
        label_taxa_serial.move(label_porta.x(),
label_porta.y() + label_porta.height() + 20)
        # Criar o QComboBox
        combo_box_taxa_serial = QtWidgets.QComboBox(self)
        combo_box_taxa_serial.addItems(["300", "1200",
"4800", "9600", "14400", "19200", "28800", "38400", "57600",
"115200"])
        combo_box_taxa_serial.setFixedWidth(90)
       combo_box_taxa_serial.move(label_taxa_serial.x() +
label_taxa_serial.width() + 2, label_taxa_serial.y() - 3)
        # Criar o QLabel "bps"
        label_bps = QtWidgets.QLabel("bps", self)
        label_bps.setFixedWidth(label_x2_name.width())
        label_bps.move(combo_box_taxa_serial.x() +
combo_box_taxa_serial.width() + 2, label_taxa_serial.y())
        # Criar o QLabel "Escala: 5 :"
        label_escala1 = QtWidgets.QLabel("Escala_1: 5 :",
self)
        label_escala1.setFixedWidth(75)
        label_escala1.move(label_porta.x(),
label_taxa_serial.y() + label_taxa_serial.height() + 20)
        # Criar o QSpinBox
        spinner_escala1 = QtWidgets.QSpinBox(self)
        spinner_escala1.setFixedWidth(50)
        spinner_escala1.move(label_escala1.x() +
label_escala1.width() + 2, label_escala1.y() - 4)
        # Criar o QLabel "Escala: 5 :"
        label_escala2 = QtWidgets.QLabel("Escala_2: 5 :",
self)
        label_escala2.setFixedWidth(75)
        label_escala2.move(label_porta.x(),
label_escala1.y() +  label_escala1.height() + 20)
        # Criar o QSpinBox
        spinner_escala2 = QtWidgets.QSpinBox(self)
        spinner_escala2.setFixedWidth(50)
        spinner_escala2.move(label_escala2.x() +
label_escala2.width() + 2, label_escala2.y() - 4)
        # Criar o QPushButton "Cancelar"
       button_cancelar = QtWidgets.QPushButton("Cancelar",
self)
        button_cancelar.move(55, self.height() - 45)
        button_cancelar.setStyleSheet("background-color:
red; color: white")
        button_cancelar.clicked.connect(self.reject)
        # Criar o QPushButton "Salvar"
        button_salvar = QtWidgets.QPushButton("Salvar",
self)
        button_salvar.move(165, self.height() - 45)
        button_salvar.setStyleSheet("background-color:
green; color: white")
        button_salvar.clicked.connect(self.save_values)
        text_field_porta.textChanged.connect(lambda text:
self.convert_text_to_uppercase(text, text_field_porta))
       text_field_max_time.textChanged.connect(lambda text:
self.validate_numeric_input(text, text_field_max_time))
        self.text_field_x1 = text_field_x1
        self.text_field_y1 = text_field_y1
        self.text_field_x2 = text_field_x2
        self.text_field_y2 = text_field_y2
        self.text_field_max_time = text_field_max_time
       self.combo_box_taxa_serial = combo_box_taxa_serial
        self.text_field_porta = text_field_porta
        self.spinner_escala1 = spinner_escala1
        self.spinner_escala2 = spinner_escala2
    def validate_numeric_input(self, text, sender):
        if not text.isnumeric():
           sender.setStyleSheet("background-color: red;")
        else:
           sender.setStyleSheet("")  # Remover o estilo de
fundo
        self.enable_save_button()
    def convert_text_to_uppercase(self, text, sender):
        sender.setText(text.upper())

    def close(self):
        self.close()
    def check_changes(self):
        global taxa_serial, porta
        if taxa_serial !=
int(self.combo_box_taxa_serial.currentText()) or porta !=
self.text_field_porta.text():
            taxa_serial =
int(self.combo_box_taxa_serial.currentText())
            porta = self.text_field_porta.text()
            self.grafico.setup()
    def save_values(self):
       global x1_name, y1_name,x2_name, y2_name, time_stop,
porta, taxa_serial, escala
        self.check_changes()

        x1_name = self.text_field_x1.text()
        y1_name = self.text_field_y1.text()
        x2_name = self.text_field_x2.text()
        y2_name = self.text_field_y2.text()
        self.grafico.ax1.set_xlabel(x1_name)
        self.grafico.ax1.set_ylabel(y1_name)
        self.grafico.ax2.set_xlabel(x2_name)
        self.grafico.ax2.set_ylabel(y2_name)
        self.grafico.canvas.draw()
        time_stop = int(self.text_field_max_time.text())
        taxa_serial =
int(self.combo_box_taxa_serial.currentText())
        porta = self.text_field_porta.text()
        escala1 = int(self.spinner_escala1.value())
        escala2 = int(self.spinner_escala2.value())
        self.accept()
```

```
class ZoomButton(QtWidgets.QAction):
    def __init__(self, canvas):
        super().__init__('Zoom', canvas)
        self.setCheckable(True)
        self.setChecked(False)
        self.canvas = canvas
    def triggered(self, checked=False):
        if checked:
            self.canvas.set_cursor('zoom-in')
            self.canvas.widgetlock(self)
        else:
            self.canvas.set_cursor('default')
            self.canvas.widgetlock.release(self)
class Grafico(QtWidgets.QWidget):
    def __init__(self, parent=None):
        global x_name, y_name, escala, port
        super().__init__(parent)
        self.fristTime = True
        self.lim_xmin = None
        self.lim_xmax = None
        self.lim_ymin = None
        self.lim_ymax = None
        self.fristZoom = True
        self.time_valuesZoom = []
        self.voltage_valuesZoom = []
        self.time_values1 = []
        self.voltage_values1 = []
        self.time_values2 = []
        self.voltage_values2 = []
        # Criação da figura e dos gráficos
        self.fig = plt.figure()
        self.ax1 = self.fig.add_subplot(211)
        self.ax1.set_title("CANAL 1")
        self.ax2 = self.fig.add_subplot(212)
        self.ax2.set_title("CANAL 2")
        self.line1, = self.ax1.plot(self.time_values1,
self.voltage_values1)
        self.line2, = self.ax2.plot(self.time_values2,
self.voltage_values2)
        self.lastState = 0
        self.ax1.set_xlabel(x1_name)
        self.ax1.set_ylabel(y1_name)
        self.ax1.grid(True)
        self.fig.subplots_adjust(hspace=0.3)
        self.ax2.set_xlabel(x2_name)
        self.ax2.set_ylabel(y2_name)
        self.ax2.grid(True)
        # Criação do widget para exibir a figura
        self.canvas = FigureCanvas(self.fig)
        self.zoom_start_cid = None
        self.zoom_end_cid = None
        self.clear_data_check = False

        # Criação dos ícones para os botões
        diretorio_atual =
os.path.dirname(os.path.abspath(__file__))
        icone_iniciar =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_iniciar.png"))
        icone_pausar =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_pausar.png"))
        icone_parar =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_parar.png"))
        icone_opcoes =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_options.png"))
        icone_save =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_save.png"))
        icone_zoom =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_zoom.png"))
        icone_set0 =
QtGui.QIcon(os.path.join(diretorio_atual,
"icone_filtro.png"))
        # Criação dos botões de imagem
        self.botao_opcoes = QtWidgets.QPushButton(self)
        self.botao_opcoes.setIcon(icone_opcoes)
        self.botao_opcoes.setFixedSize(30, 30)
      self.botao_opcoes.clicked.connect(self.open_options
)
        self.botao_iniciar = QtWidgets.QPushButton(self)
        self.botao_iniciar.setIcon(icone_iniciar)
        self.botao_iniciar.setFixedSize(30, 30)
      self.botao_iniciar.clicked.connect(self.start_recei
ving)
        self.botao_pausar = QtWidgets.QPushButton(self)
        self.botao_pausar.setIcon(icone_pausar)
        self.botao_pausar.setFixedSize(30, 30)
      self.botao_pausar.clicked.connect(self.pause_receiv
ing)
        self.botao_parar = QtWidgets.QPushButton(self)
        self.botao_parar.setIcon(icone_parar)
        self.botao_parar.setFixedSize(30, 30)
      self.botao_parar.clicked.connect(self.stop_receivin
g)


        self.botao_save = QtWidgets.QPushButton(self)
        self.botao_save.setIcon(icone_save)
        self.botao_save.setFixedSize(30, 30)
      self.botao_save.clicked.connect(self.save_receiving
)
        self.botao_zoom = QtWidgets.QPushButton(self)
        self.botao_zoom.setIcon(icone_zoom)
        self.botao_zoom.setFixedSize(30, 30)
        self.botao_zoom.setCheckable(True)
        self.botao_zoom.clicked.connect(self.toggle_zoom)

        self.botao_set0 = QtWidgets.QPushButton(self)
        self.botao_set0.setIcon(icone_set0)
        self.botao_set0.setFixedSize(30, 30)
        self.botao_set0.clicked.connect(self.setNewZero)
        # Criação do layout dos botões
        button_layout = QtWidgets.QHBoxLayout()
        button_layout.setContentsMargins(0, 0, 0, 0)
        button_layout.setSpacing(10)
        button_layout.addSpacing(10)
```

```
        button_layout.addWidget(self.botao_save)
        button_layout.addWidget(self.botao_opcoes)
        button_layout.addWidget(self.botao_iniciar)
        button_layout.addWidget(self.botao_pausar)
        button_layout.addWidget(self.botao_parar)
        button_layout.addWidget(self.botao_zoom)
        button_layout.addWidget(self.botao_set0)
        button_layout.addStretch(1)
        # Criação do layout do painel de gráfico
        layout = QtWidgets.QVBoxLayout(self)
        layout.addLayout(button_layout)
        layout.addWidget(self.canvas)
        # Variável para controlar a recepção de dados e
atualização do gráfico
        self.receiving_data = False
        self.start_time = None
        self.last_elapsed_time = 0

        self.botao_iniciar.setEnabled(True)
        self.botao_pausar.setEnabled(False)
        self.botao_parar.setEnabled(False)
        self.botao_zoom.setEnabled(True)
        self.botao_set0.setEnabled(False)
    def update_plot(self):
        self.line1.set_data(self.time_values1,
self.voltage_values1)
        self.ax1.relim()
        self.ax1.autoscale_view()
        self.line2.set_data(self.time_values2,
self.voltage_values2)
        self.ax2.relim()
        self.ax2.autoscale_view()
        self.canvas.draw()
    def open_options(self):
        if not self.receiving_data:
            options_dialog = OptionsDialog(self)
            options_dialog.text_field_x1.setText(x1_name)
            options_dialog.text_field_y1.setText(y1_name)
            options_dialog.text_field_x2.setText(x2_name)
            options_dialog.text_field_y2.setText(y2_name)
         options_dialog.text_field_max_time.setText(str(t
ime_stop))
           options_dialog.text_field_porta.setText(porta)
          options_dialog.combo_box_taxa_serial.setCurrentT
ext(str(taxa_serial))
         options_dialog.spinner_escala1.setValue(int(esca
la1))
         options_dialog.spinner_escala2.setValue(int(esca
la2))
            options_dialog.exec_()
    def start_receiving(self):
        if self.clear_data_check:
            self.clear_data()
            self.clear_data_check = False
        self.botao_iniciar.setEnabled(False)
        self.botao_parar.setEnabled(True)
        self.botao_pausar.setEnabled(True)
        self.botao_zoom.setEnabled(False)
        self.botao_set0.setEnabled(False)
        self.lastState = 0
        if self.fristTime:
            self.fristTime = False
            self.setup()
        if self.receiving_data:
            if self.start_time is None:
                self.start_time = datetime.now()
            self.serial_timer = QtCore.QTimer()
         self.serial_timer.timeout.connect(self.read_seri
al)
            self.serial_timer.start(0)
    def pause_receiving(self):
        self.botao_iniciar.setEnabled(True)
        self.botao_pausar.setEnabled(False)
        self.botao_parar.setEnabled(True)
        self.botao_zoom.setEnabled(True)
        self.lastState = 1
        if self.receiving_data:
            self.receiving_data = False
    def stop_receiving(self):
        self.lastState = 2
        self.botao_iniciar.setEnabled(True)
        self.botao_pausar.setEnabled(False)
        self.botao_parar.setEnabled(False)
        self.botao_zoom.setEnabled(True)
        self.receiving_data = False
        self.clear_data_check = True
    def change_last_4_chars(self, string, new_chars):
        if len(string) >= 4:
            new_string = string[:-4] + new_chars
            return new_string
        else:
            return string
    def save_receiving(self):
        file_dialog = QFileDialog()
        file_pathImage1, _ =
file_dialog.getSaveFileName(self, "Salvar CANAL 1", "",
"Arquivos PNG (*.png)")
       nome_arquivo2,_ = os.path.splitext(file_pathImage1)
        file_dados1 = nome_arquivo2 + " - DADOS.m"

        fig1 = plt.figure()
        ax1 = fig1.add_subplot(111)
        ax1.plot(self.time_values1, self.voltage_values1)
        file_pathImage2, _ =
file_dialog.getSaveFileName(self, "Salvar CANAL 2", "",
"Arquivos PNG (*.png)")
       nome_arquivo2,_ = os.path.splitext(file_pathImage2)
        file_dados2 = nome_arquivo2 + " - DADOS.m"
        fig2 = plt.figure()
        ax2 = fig2.add_subplot(111)
        ax2.plot(self.time_values2, self.voltage_values2)
        fig1.savefig(file_pathImage1)
        fig2.savefig(file_pathImage2)
      self.save(self.voltage_values1,self.time_values1,fi
le_dados1)
      self.save(self.voltage_values2,self.time_values2,fi
le_dados2)
        plt.close(fig2)
```

```
    def save(self,voltages, times,path):
        y1 = np.array(voltages)
        t1 = np.array(times)
        # Formatação dos vetores
       y1_formatted = ' '.join(str(value) for value in y1)
       t1_formatted = ' '.join(str(value) for value in t1)
        # Cria o conteúdo do arquivo .m
        content = f'x = [{t1_formatted}]\n'
        content += f'y = [{y1_formatted}]'
        # Salva o conteúdo no arquivo .m
        with open(path, 'w') as file:
            file.write(content)
    def toggle_zoom(self, checked):
        if checked:
         self.canvas.setCursor(QtGui.QCursor(QtCore.Qt.Cr
ossCursor))

            self.botao_iniciar.setEnabled(False)
            self.botao_parar.setEnabled(False)
            self.botao_save.setEnabled(False)
            self.botao_set0.setEnabled(True)
            self.zoom_start_cid =
self.canvas.mpl_connect('button_press_event',
self.on_zoom_start)
            self.zoom_end_cid =
self.canvas.mpl_connect('button_release_event',
self.on_zoom_end)
        else:
         self.canvas.setCursor(QtGui.QCursor(QtCore.Qt.Ar
rowCursor))
            if self.zoom_start_cid is not None:
            self.canvas.mpl_disconnect(self.zoom_start_ci
d)
                self.zoom_start_cid = None
            if self.zoom_end_cid is not None:
            self.canvas.mpl_disconnect(self.zoom_end_cid)
                self.zoom_end_cid = None

            self.botao_iniciar.setEnabled(True)
            if self.lastState == 1:
                self.botao_parar.setEnabled(True)
            elif self.lastState == 2:
                self.botao_pausar.setEnabled(False)
                self.botao_parar.setEnabled(False)
            self.botao_save.setEnabled(True)
            self.botao_set0.setEnabled(False)
            self.restore_plot_state()
    def restore_plot_state(self):
        self.update_plot()
    def on_zoom_start(self, event):
        if event.button == 1:
            self.zoom_update_cid =
self.canvas.mpl_connect('motion_notify_event',
self.on_zoom_update)
         self.zoom_rect1 = patches.Rectangle((event.xdata,
event.ydata), 0, 0, alpha=0.3, edgecolor='black',
facecolor='gray')
         self.zoom_rect2 = patches.Rectangle((event.xdata,
event.ydata), 0, 0, alpha=0.3, edgecolor='black',
facecolor='gray')
            self.ax1.add_patch(self.zoom_rect1)
            self.ax2.add_patch(self.zoom_rect2)
            self.canvas.draw()
    def on_zoom_update(self, event):
        if event.button == 1:
            dx = event.xdata - self.zoom_rect1.get_x()
            dy = event.ydata - self.zoom_rect1.get_y()
            self.zoom_rect1.set_width(dx)
            self.zoom_rect1.set_height(dy)
            dx = event.xdata - self.zoom_rect2.get_x()
            dy = event.ydata - self.zoom_rect2.get_y()
            self.zoom_rect2.set_width(dx)
            self.zoom_rect2.set_height(dy)
            self.canvas.draw()
    def on_zoom_end(self, event):
       if event.button == 1 and self.zoom_start_cid is not
None:
         self.canvas.mpl_disconnect(self.zoom_update_cid)
            self.zoom_rect1.remove()
            self.zoom_rect2.remove()



            self.time1_valuesZoom = []
            self.voltage1_valuesZoom = []
            self.time2_valuesZoom = []
            self.voltage2_valuesZoom = []
            xmin_zoom = min(event.xdata,
self.zoom_rect1.get_x())
            xmax_zoom = max(event.xdata,
self.zoom_rect1.get_x() + self.zoom_rect1.get_width()) if
self.zoom_rect1.get_width() > 0 else
max(event.xdata,self.zoom_rect1.get_x())
            if xmax_zoom < xmin_zoom:
                aux = xmin_zoom
                xmin_zoom = xmax_zoom
                xmax_zoom = aux
            for i in range(len(self.time_values1)):
                if xmin_zoom <= self.time_values1[i] <=
xmax_zoom:
               self.time1_valuesZoom.append(self.time_val
ues1[i])
               self.voltage1_valuesZoom.append(self.volta
ge_values1[i])
            for i in range(len(self.time_values2)):
                if xmin_zoom <= self.time_values2[i] <=
xmax_zoom:
               self.time2_valuesZoom.append(self.time_val
ues2[i])
               self.voltage2_valuesZoom.append(self.volta
ge_values2[i])
            self.line1.set_data(self.time1_valuesZoom,
self.voltage1_valuesZoom)
            self.line2.set_data(self.time2_valuesZoom,
self.voltage2_valuesZoom)
            self.ax1.relim()
            self.ax2.relim()
            self.ax1.autoscale_view()
            self.ax2.autoscale_view()
            self.canvas.draw()
    def setNewZero(self):
```

```
        limite_inferior, limite_superior =
self.ax1.get_xlim()
        time_filtrado = []
        voltage_filtrado = []
        fristPoint = None
        for valor1, valor2 in zip(self.time_values1,
self.voltage_values1):
          if limite_inferior <= valor1 <= limite_superior:
               if fristPoint == None:
                   fristPoint = valor1
               time_filtrado.append(valor1-fristPoint)
               voltage_filtrado.append(valor2)

        self.time_values1 = time_filtrado
        self.voltage_values1 = voltage_filtrado
        time_filtrado = []
        voltage_filtrado = []
        fristPoint = None
        for valor1, valor2 in zip(self.time_values2,
self.voltage_values2):
          if limite_inferior <= valor1 <= limite_superior:
               if fristPoint == None:
                   fristPoint = valor1
               time_filtrado.append(valor1-fristPoint)
               voltage_filtrado.append(valor2)

        self.time_values2 = time_filtrado
        self.voltage_values2 = voltage_filtrado

        self.update_plot()

    def clear_data(self):
        self.time_values1.clear()
        self.voltage_values1.clear()
        self.time_values2.clear()
        self.voltage_values2.clear()
        self.start_time = None
        self.last_elapsed_time = 0
        self.clear_data_check = False
        self.update_plot()
    def setup(self):
        global port
        if port is not None and port.is_open:
            port.close()
        try:
            # Configurar a porta serial
            port = serial.Serial(porta, taxa_serial)
            # Limpar o buffer da porta serial
            port.reset_input_buffer()

            self.receiving_data = True

        except serial.serialutil.SerialException as e:



            self.botao_iniciar.setEnabled(True)
            self.botao_pausar.setEnabled(False)
            self.botao_parar.setEnabled(False)
            self.fristTime = True
            self.receiving_data = False
            error_dialog = QtWidgets.QMessageBox()
          error_dialog.setIcon(QtWidgets.QMessageBox.Warni
ng)
            error_dialog.setWindowTitle("Erro na conexão
serial")
            error_dialog.setText("Não foi possível
estabelecer a conexão serial.")
            error_dialog.setInformativeText(str(e))
            error_dialog.exec_()
    def read_serial(self):
        global port, escala1, escala2
        if self.receiving_data and port is not None and
self.start_time is not None:
            try:

                port.reset_input_buffer()
                voltage1 = int.from_bytes(port.read(),
byteorder='big', signed=False) * 1 / 255 * escala1
                current_time1 = datetime.now()
                voltage2 = int.from_bytes(port.read(),
byteorder='big', signed=False) * 1 / 255 * escala2
                current_time2 = datetime.now()
                elapsed_time1 = (current_time1 -
self.start_time).total_seconds() * 1000
                self.time_values1.append(elapsed_time1)
                self.voltage_values1.append(voltage1)
                elapsed_time2 = (current_time2 -
self.start_time).total_seconds() * 1000
                self.time_values2.append(elapsed_time2)
                self.voltage_values2.append(voltage2)
                self.update_plot()
                self.last_elapsed_time2 = elapsed_time2
            except serial.serialutil.SerialException as e:
                self.botao_iniciar.setEnabled(True)
                self.botao_pausar.setEnabled(False)
                self.botao_parar.setEnabled(False)
                self.receiving_data = False
                error_dialog = QtWidgets.QMessageBox()
             error_dialog.setIcon(QtWidgets.QMessageBox.Wa
rning)
               error_dialog.setWindowTitle("Erro na conexão
serial")
                error_dialog.setText("Não foi possível
estabelecer a conexão serial.")
                error_dialog.setInformativeText(str(e))
                error_dialog.exec_()
                self.clear_data()

class Janela(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        super().__init__(parent)
        # Criação do painel do gráfico
        self.grafico = Grafico()
        self.setCentralWidget(self.grafico)
        self.showMaximized()
        diretorio_atual =
os.path.dirname(os.path.abspath(__file__))
      self.setWindowIcon(QtGui.QIcon(os.path.join(diretor
io_atual, "icone_frame.png")))
```

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    janela = Janela()
    janela.setWindowTitle("Pequiscópio")

    janela.show()
    sys.exit(app.exec_())
```

APÊNDICE C - CÓDIGO DE ESTIMATIVA DA FUNÇÃO DE TRANSFERÊNCIA

```
v_real = [u_real' y_real'];
modelo = arx(v_real, [2 2 1]);
[Nz, Dz]=th2tf(modelo);
[Ns, Ds]=d2cm(Nz,Dz,media_real,'matched');

%Normalizei o ultimo termo para ficar igual o do livro
num_estimado = 7.97*Ns/Ds(end); %Numerador
den_estimado = 7.97*Ds/Ds(end); %Denominador

sys_estimado = tf(num_estimado,den_estimado);
```