UNIVERSIDADE FEDERAL DO TRIÂNGULO MINEIRO
INSTITUTO DE CIÊNCIAS TECNOLÓGICAS E EXATAS
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Renan Pinheiro Soares
Raí Rocha dos Santos

LABORATÓRIO 2: ANÁLISE DA RESPOSTA DE UM CIRCUITO RC AO DEGRAU UNITÁRIO

Uberaba
2024

Renan Pinheiro Soares
Raí Rocha dos Santos

LABORATÓRIO 2: ANÁLISE DA RESPOSTA DE UM CIRCUITO RC AO DEGRAU UNITÁRIO

Relatório apresentado à disciplina Controle de Sistemas Dinâmicos do Curso de Engenharia Mecânica da Universidade Federal do Triângulo Mineiro como requisito parcial de avaliação

Prof. Dr. Fabian Andres Lara Molina

Uberaba
2024

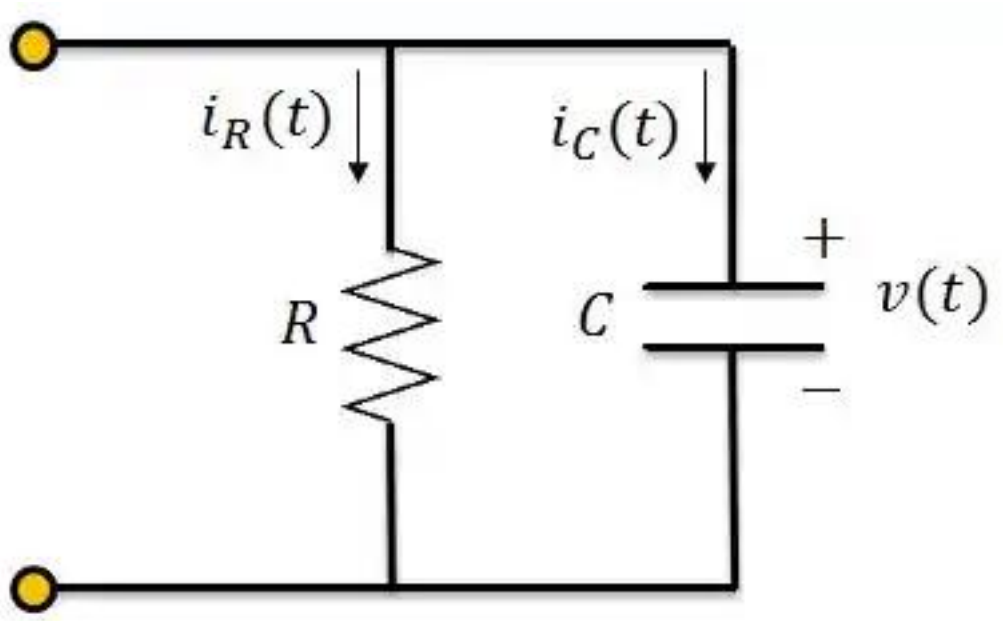
SUMÁRIO

1 INTRODUÇÃO

Um circuito resistor capacitor (Circuito RC) é um circuito eletrônico simples, porém que se disponibiliza de muitas aplicações em projetos e experimentos de engenharia, como por exemplo um circuito de filtro condicionador de sinais, situação onde é mais utilizado. Este circuito é bastante estudado quando se trata de controle de sistemas dinâmicos pois permite fazer analogias com sistemas mecânicos equivalentes.

Durante o presente experimento foi utilizado um circuito RC simples, composto apenas por uma fonte de tensão, um resistor e um capacitor (Figura 1).

Figura 1: Circuito RC



Fonte: https://arquivos.respondeai.com.br/seo-mirror/theory/2023/31a067ce-47d7-4427-961b-cb25e4a91a63.webp

2 OBJETIVOS

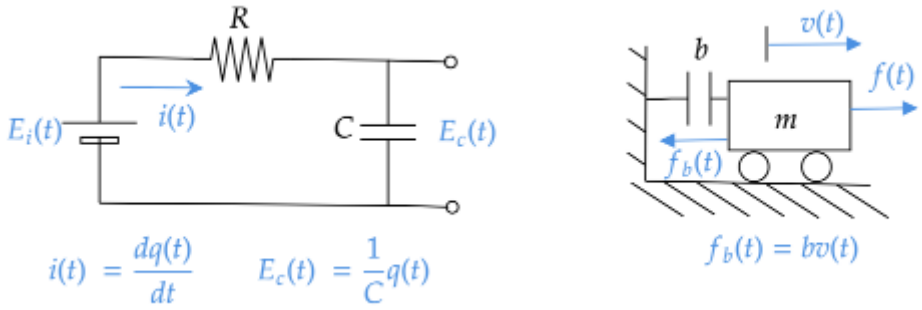
Este experimento teve como finalidade estudar a dinâmica de um circuito RC analisando sua resposta à entrada degrau de 5 V. Em outras palavras, o circuito foi submetido a uma tensão constante de 5 V, para poder então adquirir e mensurar a tensão elétrica de saída no capacitor durante sua carga e descarga. Desta forma também foi possível fazer uma analogia do circuito RC com um sistema massa-amortecedor, para melhor entendimento dos sistemas de controle e de seu funcionamento.

3 METODOLOGIA

3.1 Analogia

Para uma maior clareza sobre o circuito RC foi feita uma analogia com um sistema mecânico massa-amortecedor (Figuras 2 e 3).

Figuras 2 e 3: Circuito RC e sistema massa-amortecedor



Fonte: MOLINA, Fabian. 2024

Utilizando das Leis de Kirchhoff para o circuito RC e substituindo as variáveis pelas Leis de Ohm podemos chegar em uma equação diferencial de primeira ordem (Equação 1) que nos fornece a tensão do capacitor em função da tensão de entrada.

$$\varepsilon_i\,(t) = R * C * \dot{\varepsilon_c}(t) + \varepsilon_c(t) \quad (1)$$

Onde, ε_i é a tensão de entrada no circuito em [V], R é a resistência do resistor, C a capacitância do capacitor e ε_c a tensão no capacitor. A partir da equação foi feita a Transformada de Laplace para obter a função de transferência do sistema (Equação 2), onde foi substituída a tensão de entrada no domínio de Laplace por $E_i(s) = 5u(t)$, onde u(t) é a função

degrau unitário. A partir desse ponto foi feita a transformada inversa de Laplace para se obter a tensão do capacitor em função do tempo (Equação 3).

$$\frac{E_c(s)}{E_i(s)} = \frac{1}{RCs + 1} \tag{2}$$

$$\varepsilon_c(t) = 5 * (1 - e^{-t/RC}) \tag{3}$$

Foi feito então o mesmo passo a passo para o sistema massa-amortecedor utilizando da 2ª Lei de Newton como ponto de partida para se obter a equação do movimento (Equação 4). A partir dessa equação foi então aplicada a transformada de Laplace para se chegar a função de transferência do sistema (Equação 5)

$$f(t) = f_b(t) + \frac{m}{b} * \dot{f}_b(t) \tag{4}$$

$$\frac{F_b(s)}{F(s)} = \frac{1}{m/b * s + 1} \tag{5}$$

A partir das funções de transferência dos dois sistemas (Equações 2 e 5) foi feita a analogia entre eles comparando variável com variável (Figura 4).

Tabela 1: Analogia entre circuito RC e sistema massa-amortecedor

Circuito RC	**Sistema massa-amortecedor**
$\frac{1}{C}$	b
R	m

Fonte: Dos autores, 2024

3.2 Simulação do experimento

Com a intenção de evitar danos aos componentes eletrônicos utilizados no circuito RC, foram realizadas duas simulações previamente ao experimento.

A primeira simulação teve como objetivo apenas simular a resposta do circuito utilizando o IDE GNU Octave, um ambiente de desenvolvimento que trabalha na linguagem de programação Octave.

Com base na Equação 3, foi elaborado um código para calcular e plotar um gráfico da tensão no capacitor durante o processo de carga do mesmo em função do tempo, representado na Figura 4.

Figura 4: Código em Octave para simulação da tensão no capacitor

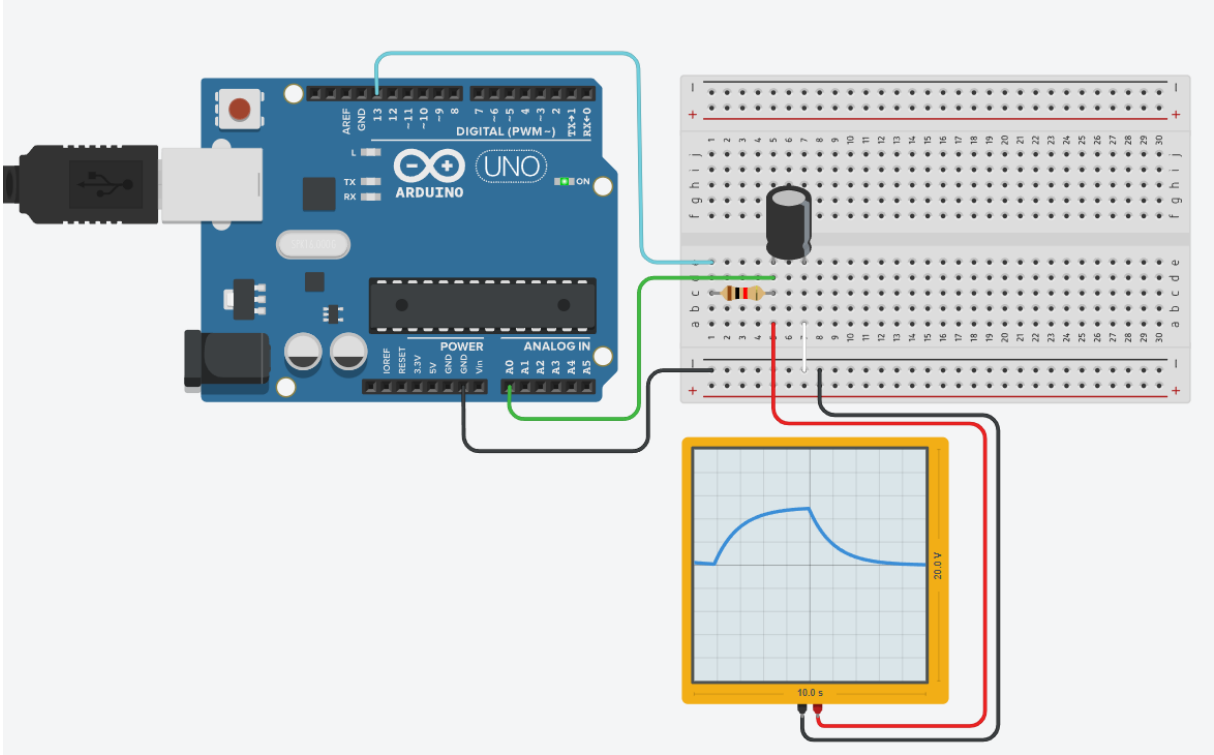
```
% Definindo os parâmetros do circuito
R = 1000;          % Resistência em Ohms (1kΩ)
C = 1000e-6;       % Capacitância em Farads (1000µF)
V0 = 5;            % Amplitude do degrau em Volts

% Definindo o vetor de tempo
t = 0:0.01:5;      % Tempo de 0 a 5 segundos, com intervalos de 0.01s

% Calculando a resposta do capacitor
Ec = V0 * (1 - exp(-t/(R*C)));

% Plotando a resposta
plot(t, Ec, 'b', 'LineWidth', 2);      % Plot em azul com linha mais grossa
xlabel('Tempo [s]');                   % Rótulo do eixo x
ylabel('DDP no Capacitor [V]');       % Rótulo do eixo y
title('Resposta do Circuito RC ao Degrau Unitário de 5V'); % Título do gráfico
grid on;                               % Ativa a grade no gráfico

```

Fonte: Dos autores, 2024

Para esta simulação foram utilizadas a capacitância e resistência iguais aos componentes reais que foram utilizados no experimento.

A segunda simulação foi feita utilizando o software da AutoDesk, ThinkerCAD. Dentro do software é possível montar circuitos elétricos e simulá-los online. O circuito foi montado no software, contendo um Arduino Uno, uma placa de testes, também conhecida como “protoboard”, um osciloscópio para visualização do sinal, um resistor de 1 kΩ e um capacitor de 1000 µF com tensão máxima de trabalho de 16 V. Foi então feito um código na linguagem de programação C++ para implementação no Arduino da simulação no ThinkerCAD, com o objetivo de carregar o capacitor até 4,9 V e então descarrega-lo até 0,1 V. O circuito RC montado no software está representado na Figura 5, onde se pode observar o osciloscópio (Componente em amarelo) mostrando o sinal de tensão de resposta do capacitor, e o código utilizado na Figura 6.

Figura 5: Circuito RC montado no software de simulação



Fonte: Dos autores, 2024

Figura 6: Código implementado na simulação

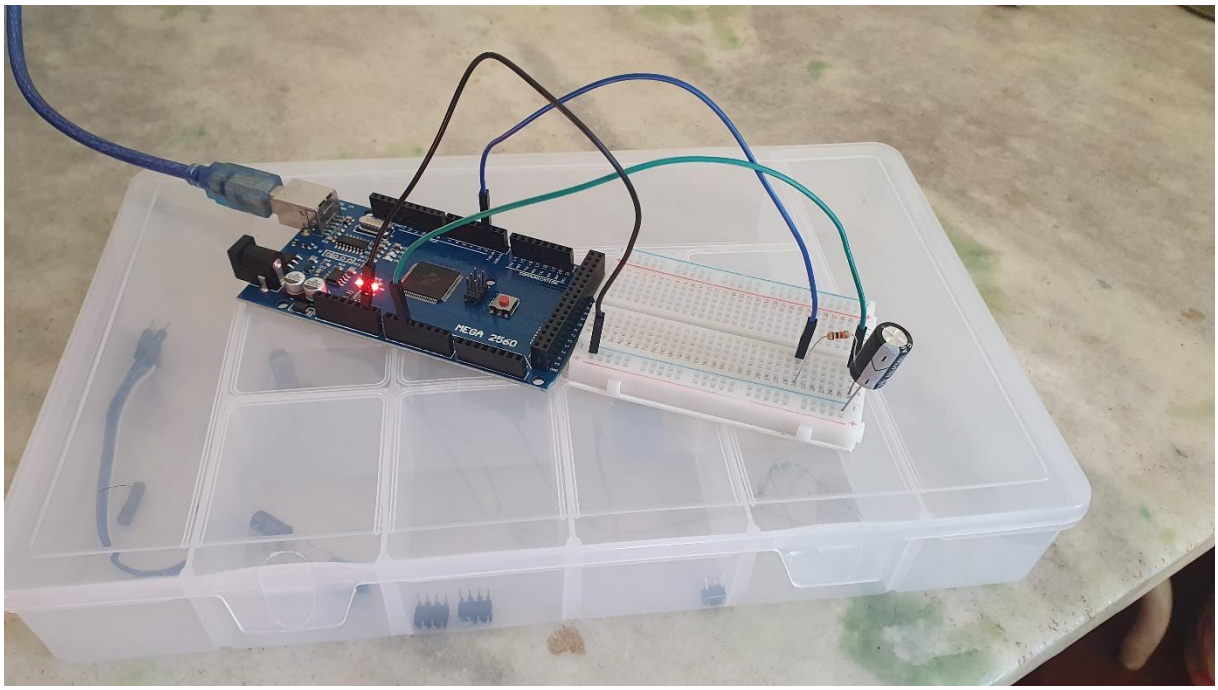
```
// Parâmetros
const int pinControle = 13;   // Pino digital conectado ao resistor que alimenta o capacitor
const int pinMedicao = A0;    // Pino analógico conectado ao terminal positivo do capacitor

// Limiares de tensão (em relação à leitura do ADC)
int limiarCarregado = 1023 * (4.9 / 5.0);  // Aproximadamente 4,9 V
int limiarDescarga = 1023 * (0.1 / 5.0);   // Aproximadamente 0,1 V

void setup() {
  Serial.begin(9600);
  pinMode(pinControle, OUTPUT);
  digitalWrite(pinControle, LOW); // Inicia em LOW
}

void loop() {
  // 1. Carregar o capacitor
  digitalWrite(pinControle, HIGH); // Inicia carregamento
  while (true) {
    int leitura = analogRead(pinMedicao);
    float tensao = (5.0 * leitura) / 1023.0;

    Serial.print("Carregando: ");
    Serial.print(leitura);
    Serial.print(" (");
    Serial.print(tensao);
    Serial.println(" V)");

    if (leitura >= limiarCarregado) {
      // Capacitor considerado carregado
      break;
    }
    delay(50); // Pequeno atraso para leitura [ms]
  }

  // 2. Descarregar o capacitor
  digitalWrite(pinControle, LOW); // Inicia descarga
  while (true) {
    int leitura = analogRead(pinMedicao);
    float tensao = (5.0 * leitura) / 1023.0;

    Serial.print("Descarregando: ");
    Serial.print(leitura);
    Serial.print(" (");
    Serial.print(tensao);
    Serial.println(" V)");

    if (leitura <= limiarDescarga) {
      // Capacitor considerado descarregado
      break;
    }
    delay(50); // Pequeno atraso para leitura [ms]
  }
}

```

Fonte: Dos autores, 2024

3.3 Montagem do circuito físico

O circuito RC real foi montado utilizando os mesmos componentes da simulação no ThinkerCAD, com exceção do Arduino, por qual foi substituído por um Arduino Mega 2650, e do osciloscópio, que não foi utilizado para visualização do sinal. O circuito montado é mostrado na Figura 7.

Figura 7: Circuito RC montado



Fonte: Dos autores, 2024

Com o circuito montado, outros códigos foram feitos para aquisição do sinal de resposta. Primeiro um código em C++ foi feito utilizando o Arduino IDE para carregar e descarregar o capacitor, assim como feito na simulação do ThinkerCAD (Figura 8). Como o Arduino não consegue plotar gráficos, foi preciso utilizar outros códigos, porém desta vez utilizando a linguagem de programação Python 3.12 por questão de escolha dos autores. Foram então desenvolvidos dois códigos em Python, o primeiro para fazer a leitura e importação dos dados adquiridos pelo Arduino e salvá-los em um arquivo de extensão “.cvs” (Figura 9). E o segundo código em Python foi feito apenas para importar esse arquivo criado e plotar um gráfico dos resultados de carga e descarga do capacitor (Figura 10). Com os códigos prontos, foi feito o “upload” do código, para carga e descarga do capacitor, para o Arduino. Logo em seguida, foram executados os códigos em python para importação dos dados e criação do diagrama de tensão no capacitor por tempo.

Figura 8: Código utilizado para carga e descarga do capacitor no Arduino

```
// Parâmetros do circuito
const int pinControle = 2;   // Pino digital (PWM) conectado ao resistor
const int pinMedicao = A2;   // Pino analógico conectado ao capacitor

// Limiares de tensão (em relação ao ADC)
int limiarCarregado = 1023 * (4.9 / 5.0);  // Aproximadamente 4,9 V
int limiarDescarga = 1023 * (0.1 / 5.0);   // Aproximadamente 0,1 V

int ciclos = 0;  // Contador de ciclos (carregamento e descarregamento)

void setup() {
  Serial.begin(9600);           // Inicia a comunicação serial
  pinMode(pinControle, OUTPUT); // Configura o pino de controle como saída
  digitalWrite(pinControle, LOW); // Inicia o pino em LOW (capacitor descarregado)
  Serial.println("tempo(ms),tensao(V),estado"); // Cabeçalho dos dados para CSV
}

void loop() {
  if (ciclos >= 2) { // Finaliza após dois ciclos completos
    Serial.println("Experimento finalizado.");
    while (1); // Loop infinito para parar o programa
  }

  // Carregar o capacitor
  unsigned long tempoInicial = millis();
  digitalWrite(pinControle, HIGH); // Configura o pino para HIGH (5V)
  while (true) {
    int leitura = analogRead(pinMedicao);
    float tensao = (5.0 * leitura) / 1023.0;
    unsigned long tempoAtual = millis() - tempoInicial;

    // Envia os dados para o Serial Monitor
    Serial.print(tempoAtual);
    Serial.print(",");
    Serial.print(tensao);
    Serial.println(",Carregando");

    if (leitura >= limiarCarregado) { // Verifica se atingiu 4,9 V
      break;
    }
    delay(50); // Atraso para suavizar as leituras
  }

  // Descarregar o capacitor
  tempoInicial = millis();
  digitalWrite(pinControle, LOW); // Configura o pino para LOW (0V)
  while (true) {
    int leitura = analogRead(pinMedicao);
    float tensao = (5.0 * leitura) / 1023.0;
    unsigned long tempoAtual = millis() - tempoInicial;

    // Envia os dados para o Serial Monitor
    Serial.print(tempoAtual);
    Serial.print(",");
    Serial.print(tensao);
    Serial.println(",Descarregando");

    if (leitura <= limiarDescarga) { // Verifica se atingiu 0,1 V
      break;
    }
    delay(50); // Atraso para suavizar as leituras
  }

  ciclos++; // Incrementa o contador de ciclos
}
```

Fonte: Dos autores, 2024

Figura 9: Código em Python para importar e salvar os dados

```python
import serial
import csv

# Configurações da porta serial
porta = "COM10"  # Substitua pela sua porta
baud_rate = 9600
ser = serial.Serial(porta, baud_rate)

# Nome do arquivo CSV
arquivo_csv = "dados_capacitor_dois_ciclos.csv"

# Inicializa as listas para contagem de ciclos
ciclos = 0

# Abre o arquivo CSV para salvar os dados
with open(arquivo_csv, mode="w", newline="") as arquivo:
    writer = csv.writer(arquivo)
    writer.writerow(["tempo(ms)", "tensao(V)", "estado"])  # Cabeçalho do arquivo CSV

    print("Capturando dados do Arduino...")

    try:
        while ciclos < 2:  # Captura apenas dois ciclos
            # Lê uma linha da porta serial
            linha = ser.readline().decode().strip()

            if linha.startswith("tempo"):  # Ignora o cabeçalho inicial
                continue

            if "Carregando" in linha or "Descarregando" in linha:
                partes = linha.split(",")
                tempo = int(partes[0])
                tensao = float(partes[1])
                estado = partes[2].strip()

                # Salva os dados no arquivo CSV
                writer.writerow([tempo, tensao, estado])

                # Conta os ciclos de carga e descarga
                if estado == "Descarregando" and tensao <= 0.1:
                    ciclos += 1

    except KeyboardInterrupt:
        print("\nCaptura interrompida pelo usuário.")
    finally:
        ser.close()

print(f"Dados salvos no arquivo: {arquivo_csv}")

```

Fonte: Dos autores, 2024

Figura 10: Código para plotar os dados salvos no arquivo .csv

```python
import pandas as pd
import matplotlib.pyplot as plt

# Nome do arquivo CSV gerado
arquivo_csv = "dados_capacitor_dois_ciclos.csv"

# Carrega os dados do arquivo CSV
dados = pd.read_csv(arquivo_csv)

# Inicializa listas para armazenar os dados contínuos
tempo = []
tensao = []

# Variável para controle do tempo contínuo
tempo_total = 0

# Processa os dados para criar um sinal contínuo
for i, linha in dados.iterrows():
    tensao.append(linha["tensao(V)"])  # Adiciona a tensão à lista
    tempo.append(tempo_total)          # Adiciona o tempo incremental à lista
    tempo_total += 50  # Incremento de 50 ms entre cada leitura

# Plota o gráfico contínuo
plt.figure(figsize=(10, 6))
plt.plot(tempo, tensao, label="Carga e Descarga do Capacitor", color="blue")
plt.title("Gráfico de Carga e Descarga do Capacitor")
plt.xlabel("Tempo (ms)")
plt.ylabel("Tensão (V)")
plt.legend()
plt.grid()
plt.show()

```

Fonte: Dos autores, 2024

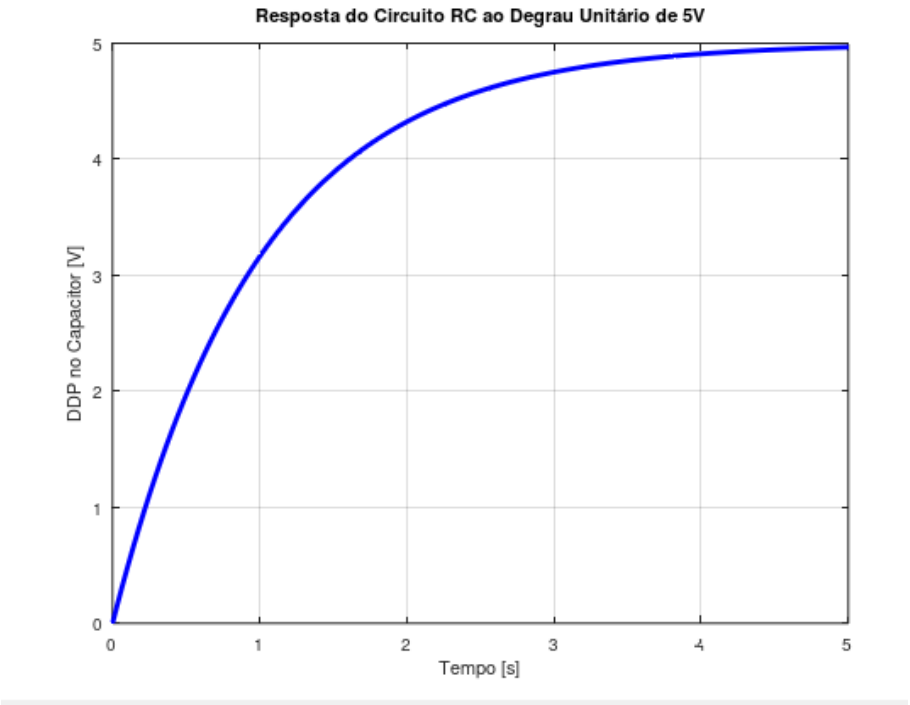
Em seguida foram executados os códigos para coletar e analisar os dados do experimento.

Obs.: Todo o projeto foi salvo em um repositório em [1]

4 RESULTADOS E DISCUSSÃO

Foram adquiridos os dados e plotados em diferentes gráficos para análise. Primeiro pode se observar a simulação utilizando o GNU Octave na Figura 11, onde se encontra o gráfico de tensão por tempo apenas no período de carga do capacitor.

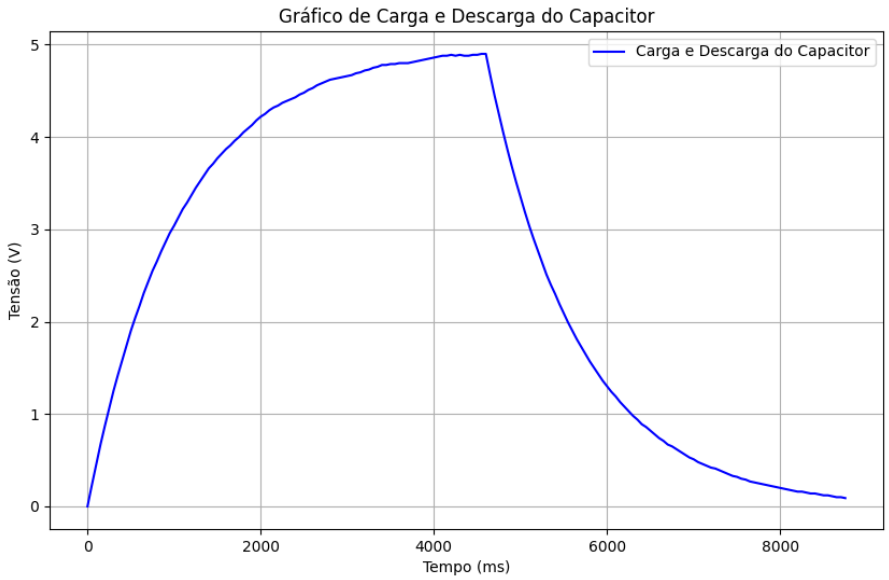
Figura 11: Gráfico gerado por Octave



Fonte: Dos autores, 2024

Podemos observar o comportamento exponencial na carga do capacitor, assim como na Figura 12, onde estão representados os dados coletados no experimento real

Figura 12: Dados de tensão no capacitor coletados na carga e descarga



Fonte: Dos autores, 2024

CONCLUSÕES

Com tudo finalizado, foi possível refletir sobre o que se trata o experimento, pode-se aprender e entender sobre o controle de sistemas e aquisição de dados de sensores deste sistema de controle. Também foi possível criar um melhor entendimento sobre o modelamento das funções de um sistema em um ambiente computacional. O experimento resultou em dados que se aproximam de um modelo exponencial. Fato que é recompensador pois era esse o esperado de acordo com a literatura e estudos do conceito.

REFERÊNCIAS

[1] SOARES, R; SANTOS, R. **Circuito RC – Carga e Descarga.** Repositório no Git Hub, 2024. Disponível em: https://github.com/BodeVelho1911/Circuito-RC. Acesso em: 7 dez. 2024.