

INSTITUTO FEDERAL

Brasília

Instituto Federal de Brasília

Campus Brasília

Curso Superior de Tecnologia em Sistemas para Internet

RELATÓRIO DE PRÁTICA INTEGRADA DE CIÊNCIA DE DADOS E INTERNET DAS COISAS

Por

***CALEB ALVES FALCÃO, GABRIEL GALVÃO DE OLIVEIRA, PALOMA GINA DE
CAMARGOS E PAULO HENRIQUE RIBEIRO DA SILVA***

Tecnólogo

BRASÍLIA
2023

Lista de Figuras

3.1	Conexão entre o Arduíno UNO R3 e o sensor DHT22	6
3.2	Arduíno Uno R3	7
3.3	Sensor DHT22	7
3.4	Jumper macho-fêmea	8
3.5	Código Arduíno IDE	9
3.6	Código Python	10
3.7	Dados de teste	11
4.1	Gabinete aberto e fechado, respectivamente	13
5.1	Instalação Pymongo	14
5.2	Conexão com o banco de dados	15
5.3	Código gráfico geral	16
5.4	Gráfico de todos os cenários em um gráfico	17
5.5	Gráfico de média de cada cenário	18
5.6	Gráfico da média de cada cenário	19
5.7	Código de gráfico de cada cenário	20
5.8	Gráfico de cada cenário	21

Sumário

1 Descrição do problema	4
2 Objetivos	5
2.1 Em um contexto mais específicos:	5
3 Sprint 1	6
3.1 Materiais e ferramentas	6
3.2 Códigos implementados	8
3.2.1 Arduino	8
3.2.2 Python	9
3.2.3 Resultado dos dados coletados	10
4 Sprint 2	12
4.1 Preparação dos dados	13
5 Sprint 3	14
5.1 Instalação do Pymongo	14
5.2 Conexão com banco de dados	15
5.3 Código do gráfico geral	16
5.4 Gráfico geral	17
5.5 Código do gráfico de média dos cenários	18
5.6 Gráfico das médias de cada cenário	19
5.7 Código para plotagem dos gráficos do computador fechado e aberto	20
5.8 Gráficos do computador fechado e aberto	21
6 Considerações finais	22
Referências	23

1

Descrição do problema

A empresa DELL TECNOLOGIES (2023) afirma que o superaquecimento de um computador é algo a se preocupar, pois pode causar problemas temporários, como reiniciar, desligar ou apresentar problemas de desempenho de modo intermitente, fazendo com que exista possibilidade de ter perda de documentos e trabalhos importantes ao causar corrompimento dos mesmos. Além disso, ao se tornar recorrente, o calor excessivo pode danificar os componentes internos, trazendo prejuízos financeiros.

A principal preocupação aqui é com os componentes que contam com resfriamento passiva como HD, RAM, etc. Segundo o site CANALTECH (2023), "O grande 'perigo' aqui é que esses componentes podem sofrer com o calor de outros (leia-se: CPU e GPU). Um gabinete bem arejado garante um bom funcionamento desses componentes, que funcionam com suas temperaturas ideais. Fora que não chegam a gerar calor o suficiente para influenciar na temperatura do resto, mas estamos pensando em 'casos normais de uso'. Dessa forma o acúmulo de calor em componentes como HD e RAM pode levar a problemas de desempenho, instabilidades e a falhas prematuras.

Dessa forma, pensou-se na ideia de criar um projeto, utilizando Arduíno juntamente com sensor de temperatura e umidade, a fim de identificar o melhor cenário para evitar tal problema e, consequentemente, quaisquer prejuízos relacionados a isto.

2

Objetivos

O objetivo é responder qual a melhor forma de resfriar um ambiente, utilizando o sensor de temperatura e umidade, para analisar como os dados se comportam com diferentes variáveis em que os sensores são expostos.

2.1 Em um contexto mais específicos:

- Analisar à temperatura dentro de um gabinete com hardware ligados ao longo do tempo para identificar possíveis problemas de superaquecimento.
- Variar as situações como colocar o computador sobre estresse
- Coletar e armazenar os dados referente a cada situação
- Realizar análises comparativas de temperatura em difere configurações do gabinete(com a tampa aberta ou fechada) para inferir a eficiência da circulação do ar

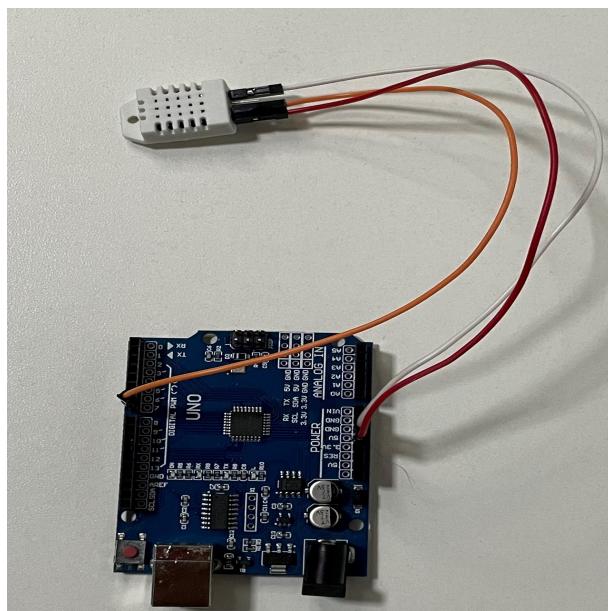
3

Sprint 1

O projeto foi conduzido no Instituto Federal de Brasília (IFB), pelos professores Dr. Fábio Henrique Monteiro Oliveira e Dr. Caio Moura Daoud.

A placa microcontroladora utilizada foi a UNO R3, responsável por capturar em tempo real os dados do sensor DHT22, que capta a temperatura e umidade em que está alocado. A conexão entre o Arduíno e o computador foi realizada com o cabo USB. A conexão entre os dois componentes eletrônicos foi realizado por 3 *jumpers* macho/fêmea, onde o primeiro pino à esquerda foi conectado à alimentação de 5V, o segundo pino ao pino de entrada de dados(porta 7) e o pino mais à direita ao *Graduated Neutral Density Filter* (GND), conforme mostra a figura 3.1

Figura 3.1 Conexão entre o Arduíno UNO R3 e o sensor DHT22



Fonte: Autoria própria (2023)

3.1 Materiais e ferramentas

Para a montagem do hardware foram adquiridos os seguintes componentes eletrônicos:

- Arduino Uno R3: conforme a própria documentação do ARDUÍNO (2023), o modelo

UNO R3 é uma placa microcontroladora baseada no ATmega328P. Possui 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um ressonador cerâmico de 16 MHz, uma conexão USB, um conector de energia, um conector ICSP e um botão de reset. Ele contém tudo o que é necessário para suportar o microcontrolador; basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC-to-DC ou bateria para começar. ARDUINO® UNO R3 (2020).

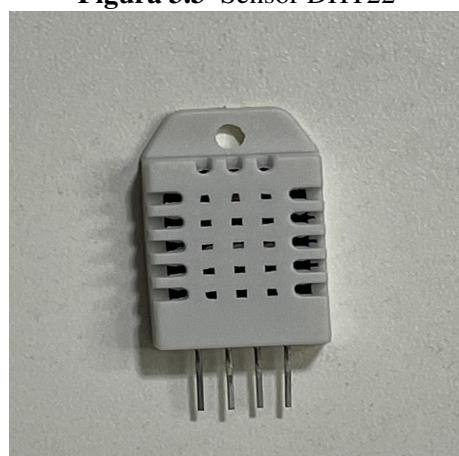
Figura 3.2 Arduíno Uno R3



Fonte: Autoria própria (2023)

- Sensor DHT22: de acordo com CITYOS (2020), é um sensor digital de temperatura e umidade de baixo custo com uma interface digital de fio único. Ele usa um sensor de umidade capacitivo e um termistor para medir o ar ao redor e emite um sinal digital no pino de dados (não são necessários pinos de entrada analógica). A figura 3.3 mostra a que foi utilizada no projeto.

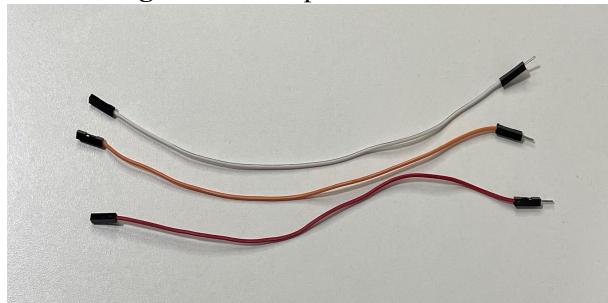
Figura 3.3 Sensor DHT22



Fonte: Autoria própria (2023)

- *Jumper*: são cabos isolados por um material não condutivo e é utilizado para efetuar as conexões entre componentes eletrônicos. Foi utilizado somente 3 cabos, bem como mostra a figura 3.4

Figura 3.4 Jumper macho-fêmea



Fonte: Autoria própria (2023)

- Linguagem de programação Python: Segundo o site do PYTHON (2023) O Python é uma linguagem de tipagem dinâmica, fácil de aprender e também é amplamente utilizada na área de ciência de dados, para desenvolvimento web e servidores. Com a linguagem Python é possível coletar dados e analisar o que facilita a tomada de decisões para gestores.
- Biblioteca Pandas: De acordo com PANDAS BIBLIOTECA (2023) "pandas é uma ferramenta de análise e manipulação de dados de código aberto rápida, poderosa, flexível e fácil de usar, construída sobre a linguagem de programação Python". A biblioteca pandas é capaz de ler dados, fazer análises e a partir disso criar gráficos para compreensão de determinados cenários.

3.2 Códigos implementados

Para programação do Arduino Uno R3 foi utilizado a *Integrated Develop Environment* (IDE) do Arduino versão 2.1.0 que utiliza da linguagem C++, já para a programação da linguagem Python, foi utilizado a IDE do Visual Studio Code. O intervalo entre leituras foi definido um tempo de 2 segundos, onde os dados foram salvos em um arquivo com extensão ".csv".

3.2.1 Arduino

A figura 3.5 representa o código do Arduíno, para fazer a leitura do sensor DHT22 e armazenar os dados no arquivo criado no código em Python, nos cenários especificados.

Figura 3.5 Código Arduíno IDE

```

arduino_iot.ino
1
2 #include <DHT_U.h>
3
4 #include <DHT.h>
5 #include <SD.h>
6
7
8 #define DHTPIN 7           // Pino do sensor conectado
9 #define DHTTYPE DHT22       // Definindo tipo do sensor DHT
10
11 DHT dht(DHTPIN, DHTTYPE); // Inicia o objeto DHT
12
13 void setup() {
14     Serial.begin(9600);
15     dht.begin();
16 }
17
18 void loop() {
19     delay(2000); // Aguarda 2 segundos
20
21     float temperatura = dht.readTemperature(); // Lê a temperatura em Celsius
22     float umidade = dht.readHumidity(); // Lê a umidade
23
24     if (isnan(temperatura) || isnan(umidade)) {
25         Serial.println("Falha ao ler o sensor DHT!");
26         return;
27     }
28
29     // Mostra os dados no monitor serial
30     Serial.print("Temperatura: ");
31     Serial.print(temperatura);
32     Serial.print(" C");
33     Serial.print("\tUmidade: ");
34     Serial.print(umidade);
35     Serial.println(" %");
36
37     // Salva os dados no arquivo CSV
38     File dataFile = SD.open("dados_pc_fechado_muito_uso_3.csv", FILE_WRITE);
39     if (dataFile) {
40         dataFile.print(temperatura);
41         dataFile.print(",");
42         dataFile.println(umidade);
43         dataFile.close();
44     } else {
45         Serial.println("Falha ao abrir o arquivo dados_pc_fechado_muito_uso_3.csv");
46     }
47 }

```

Fonte: Autoria própria (2023)

3.2.2 Python

A figura 3.6 representa o código utilizado para configurar as conexões com o Arduíno, como a porta onde o cabo USB está conectado e a velocidade de comunicação. Além disso, cria o arquivo ".csv", determina o título das colunas, verifica se os dados são válidos e salva os dados no arquivo ".csv".

Figura 3.6 Código Python


```

dht_logger.py ×
C: > Users > paulo > Documents > projeto > projeto > dht_logger.py > ...
1 import serial
2 import csv
3
4 print("Arquivo em execução.")
5
6 # Configurando conexão serial
7 porta_serial = 'COM3' # Porta serial conectada no computador
8 baud_rate = 9600 # Velocidade de comunicação usada no arduino
9
10 # Configurando arquivo CSV
11 arquivo_csv = 'dados_pc_fechado_muito_uso.csv'
12
13 # Inicia conexão serial
14 serial_connection = serial.Serial(porta_serial, baud_rate)
15
16 # Cria o arquivo CSV e escreve o título das colunas "temperatura" e "Umidade"
17 with open(arquivo_csv, 'w', newline='') as file:
18     writer = csv.writer(file)
19     writer.writerow(['Temperatura (°C)', 'Umidade (%)'])
20
21 while True:
22     # Lê uma linha da conexão serial
23     line = serial_connection.readline().decode().strip()
24
25     # Verifica se existe dados válidos na linha
26     if line.startswith('Temperatura:'):
27         parts = line.split()
28         if len(parts) >= 4:
29             temperatura = parts[1] # Acessa a temperatura na posição 1
30             umidade = parts[4] # Acessa a umidade na posição 4
31
32             # Salva os dados no arquivo CSV
33             with open(arquivo_csv, 'a', newline='') as file:
34                 writer = csv.writer(file)
35                 writer.writerow([temperatura, umidade])
36         else:
37             print("Não contém valores suficientes na linha:", line)
38

```

Fonte: Autoria própria (2023)

3.2.3 Resultado dos dados coletados

Com a inicialização dos programas Arduino IDE e aplicação em Python, foi capaz de armazenar os dados no arquivo de extensão ".csv", após encerrar os programas, é possível visualizar os dados coletados por meio de uma planilha, conforme é mostrado na figura 3.7.

Figura 3.7 Dados de teste

The screenshot shows a LibreOffice Calc spreadsheet titled "dados.csv - LibreOffice Calc". The menu bar includes "Arquivo", "Editar", "Exibir", "Inserir", "Formatar", and "E". The toolbar includes icons for file operations, font selection, and cell styling. The formula bar shows "B29" and "59.30". The table has three columns: A (Temperatura (°C)), B (Umidade (%)), and C. Column A contains values from 25.70 to 25.20. Column B contains values from 58.70 to 59.30. Column C is empty. Row 29 is selected, showing the value 59.30 in cell B29.

	A	B	C
1	Temperatura (°C)	Umidade (%)	
2	25.70	58.70	
3	25.70	58.80	
4	25.70	59.30	
5	25.80	59.20	
6	25.80	58.80	
7	25.70	58.80	
8	25.70	59.00	
9	25.70	59.40	
10	25.70	59.60	
11	25.70	59.40	
12	25.70	59.50	
13	25.60	59.80	
14	25.60	58.80	
15	25.60	58.50	
16	25.50	58.60	
17	25.50	58.50	
18	25.50	58.70	
19	25.50	59.20	
20	25.40	58.80	
21	25.40	58.50	
22	25.40	58.60	
23	25.40	58.70	
24	25.40	58.70	
25	25.30	59.10	
26	25.30	59.20	
27	25.30	59.20	
28	25.20	59.20	
29	25.20	59.30	

Fonte: Autoria própria (2023)

4

Sprint 2

Os dados de temperatura e umidade do ar foram coletados de 10 a 16 de junho de 2023, em diferentes horários durante tais dias, com duração de duas horas em cada coleta. Os cenários utilizados para coleta no referido período foram:

- Gabinete **fechado com pouca utilização** de recursos do computador, como, por exemplo: vídeos no YouTube, poucas guias abertas no Google Chrome e alguns minutos de música no Spotify.
- Gabinete **fechado com muita utilização**, como, por exemplo: muitas guias abertas no Google Chrome, vídeos no YouTube, ChatGPT em execução e rodando dois jogos, Roblox Ploxburg e The Sims.
- Gabinete **aberto com pouca utilização**, como, por exemplo: alguns minutos de música no Spotify, poucas guias abertas no Google Chrome e vídeos no YouTube.
- Gabinete **aberto com muita utilização**, como, por exemplo: muitas guias abertas no Google Chrome, vídeos no YouTube, ChatGPT em execução e rodando dois jogos, Roblox Ploxburg e The Sims.

A figura 4.1 mostra como ficou o ambiente para coleta das variáveis.

Figura 4.1 Gabinete aberto e fechado, respectivamente



Fonte: Autoria própria (2023)

4.1 Preparação dos dados

1. Não foi necessário realizar a conversão de dados e unidade de medida, pois a biblioteca do sensor já realiza automaticamente;
2. Não foi necessário lidar com dados faltantes, pois a coleta estava sendo realizada ao lado e era verificado periodicamente seu funcionamento;
3. Foi utilizado o aplicativo do clima para verificar se o sensor estava coletando algum dado "fora da curva";
4. Conforme as coletas eram realizadas, foi se adicionando no nome o cenário e o horário que estava sendo realizado;
5. Foram criados vários arquivos csv com os dados coletados, conforme passo anterior;
6. Com os arquivos finalizados, os dados foram encaminhados para o banco de dados MongoDB.

5

Sprint 3

5.1 Instalação do Pymongo

Através do *Pip Installs Packages* (PIP) sistema de gerenciamento de pacotes usado para instalar bibliotecas e módulos em Python, foi capaz de instalar o Pymongo. O Py-
mongo permite a conexão ao banco de dados MongoDB e execute as funções básicas como
CRUD(Create,Read,Update,Delete), a imagem abaixo é mostrado o código para instalação do
Pymongo conforme mostrado na figura 5.1.

Figura 5.1 Instalação Pymongo

```
!pip install pymongo

Collecting pymongo
  Downloading pymongo-4.4.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (648 kB)
    ━━━━━━━━━━━━━━━━ 648.9/648.9 kB 10.2 MB/s eta 0:00:00

Collecting dnspython<3.0.0,>=1.16.0 (from pymongo)
  Downloading dnspython-2.3.0-py3-none-any.whl (283 kB)
    ━━━━━━━━━━━━━━ 283.7/283.7 kB 31.0 MB/s eta 0:00:00

Installing collected packages: dnspython, pymongo
Successfully installed dnspython-2.3.0 pymongo-4.4.0
```

Fonte: Autoria própria (2023)

5.2 Conexão com banco de dados

Com a importação do mongodb e a classe específica *MongoClient* o código abaixo é capaz de fazer a conexão com o banco de dados, que permite manipular dados dentro desse banco de dados.

Figura 5.2 Conexão com o banco de dados

```
from pymongo import MongoClient
from statistics import mean

# Informações de conexão
username = 'root123'
password = 'root123'
host = 'gabinetethermologdb.oblbr1v.mongodb.net'
database = 'GabineteThermoLogDB'

# Cria a string de conexão
uri = f"mongodb+srv://{username}:{password}@{host}/{database}?retryWrites=true&w=majority"

# Conecta ao banco de dados
client = MongoClient(uri)

# Acessa o banco de dados
db = client[database]

# Acessa a coleção desejada
collection = db['dados_pc_fechado_pouco_uso_manhã']

# Imprime a coleção (Teste)
print(collection)

# Recupera os documentos da coleção
documents = collection.find({})

cont = 0
# Testando o acesso aos documentos
print(documents)
media = []
for document in documents:
    media.append(document['Temperatura'])

print(mean(media))
```

Fonte: Autoria própria (2023)

5.3 Código do gráfico geral

Nesta etapa foi feita o código para a plotagem do gráfico geral em forma de linha, conforme mostrado abaixo:

Figura 5.3 Código gráfico geral

```
# Criar uma figura e um conjunto de eixos com tamanho maior
fig, ax = plt.subplots(figsize=(10, 8))

# Plotar as linhas para cada cenário com marcadores maiores e estilos visuais
for colecao, df in dataframes.items():
    ax.plot(df['Temperatura'], label=colecao, marker='o', markersize=3, linestyle='solid', lw=8)

# Configurar os rótulos dos eixos e o título do gráfico com tamanho de fonte maior
ax.set_xlabel('Amostras', fontsize=12)
ax.set_ylabel('Temperatura', fontsize=12)
ax.set_title('Cenários de Temperatura do PC', fontsize=14)
# Padroniza o intervalos da temperatura
ax.set_xlim(24, 40)
ax.set_yticks(range(24, 40, 2))
ax.set_xlim(0,4600, 500)
# Aumentar o tamanho dos marcadores na legenda
handles, labels = ax.get_legend_handles_labels()
ax.legend(handles, labels, loc='lower center', bbox_to_anchor=(0.5, -0.2), ncol=4, fontsize=10)

# Aumentar o tamanho dos rótulos dos eixos e dos números nos eixos
plt.xticks([0, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000])
plt.yticks()
ax.tick_params(labelsize=10)

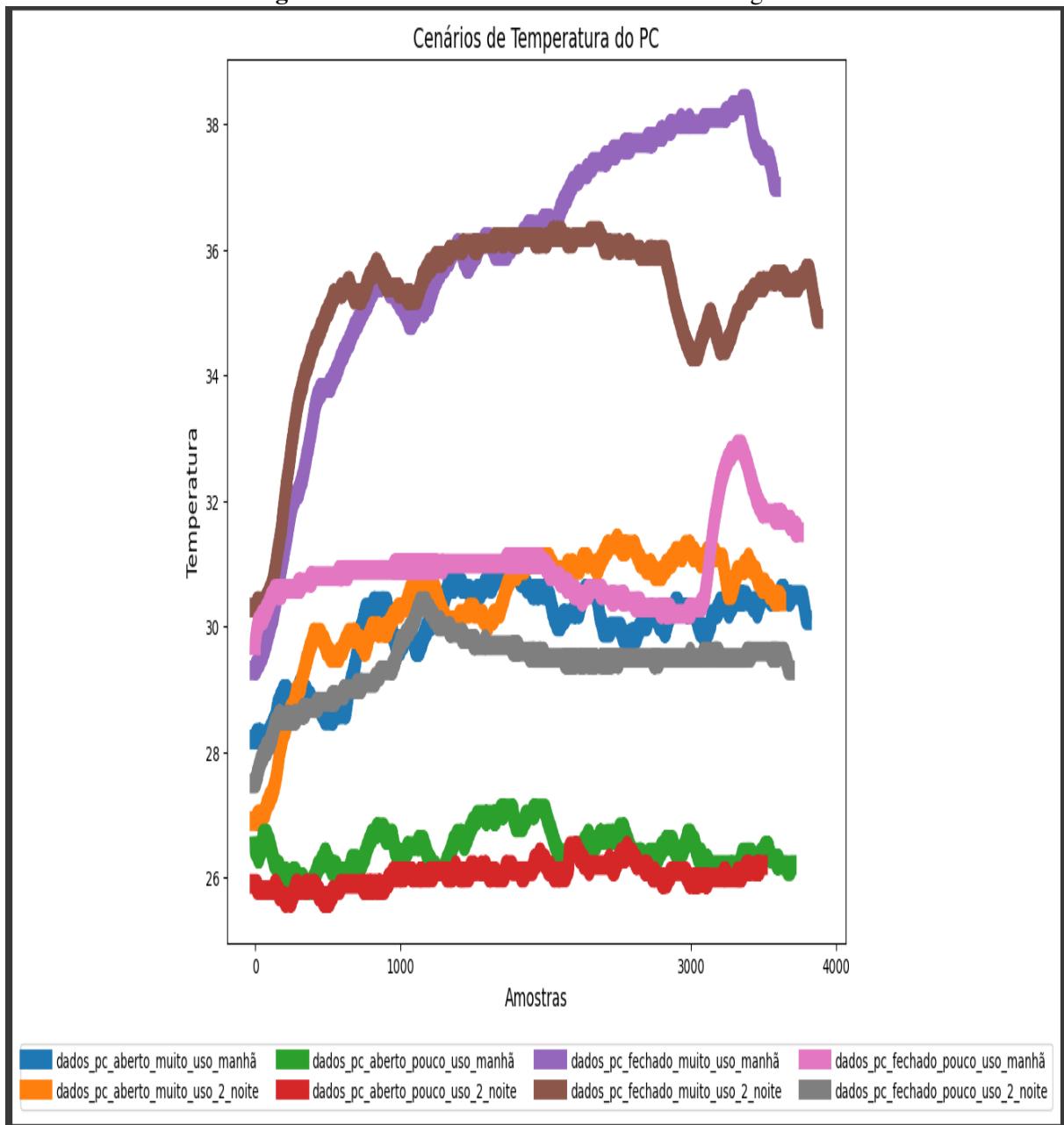
# Salvar o gráfico com tamanho completo
plt.savefig('/content/grafico_temperatura.png', bbox_inches='tight')
plt.show()
```

Fonte: Autoria própria (2023)

5.4 Gráfico geral

O gráfico mostrado na figura 5.4 reflete os cenários de temperatura do computador.

Figura 5.4 Gráfico de todos os cenários em um gráfico



Fonte: Autoria própria (2023)

Conforme a figura 5.4 mostra, percebe-se que há uma diferença significativa de aquecimento em alguns diferentes cenários. Quando se utiliza muitos recursos do computador, juntamente com o gabinete fechado, a temperatura eleva abundantemente. Dessa maneira, foi testado e comprovado que o computador perde muito desempenho neste cenário. Já com recursos utilizados em sua completude, juntamente com o gabinete aberto, se mantém um desempenho agradável, pois, devido à boa ventilação, não há uma elevação significativa de temperatura.

5.5 Código do gráfico de média dos cenários

O código mostrado na figura 5.5 é feito a lista de coleções com o nome das coleções e também um dicionário para mapear nomes originais para nomes abreviados, posteriormente a coleção é transformada em DataFrame e o cálculo da média.

Figura 5.5 Gráfico de média de cada cenário

```

import pandas as pd
import matplotlib.pyplot as plt

# Lista de coleções com o nome das coleções
colecoes = ['dados_pc_aberto_muito_uso_manhã', 'dados_pc_aberto_muito_uso_2_noite',
            'dados_pc_aberto_pouco_uso_manhã', 'dados_pc_aberto_pouco_uso_2_noite',
            'dados_pc_fechado_muito_uso_manhã', 'dados_pc_fechado_muito_uso_2_noite',
            'dados_pc_fechado_pouco_uso_manhã', 'dados_pc_fechado_pouco_uso_2_noite']

# Dicionário para mapear nomes originais para nomes abreviados
abreviacoes = {
    'dados_pc_aberto_muito_uso_manhã': 'Aberto, muito uso, manhã',
    'dados_pc_aberto_muito_uso_2_noite': 'Aberto, muito uso, noite',
    'dados_pc_aberto_pouco_uso_manhã': 'Aberto, pouco uso, manhã',
    'dados_pc_aberto_pouco_uso_2_noite': 'Aberto, pouco uso, noite',
    'dados_pc_fechado_muito_uso_manhã': 'Fechado, muito uso, manhã',
    'dados_pc_fechado_muito_uso_2_noite': 'Fechado, muito uso, noite',
    'dados_pc_fechado_pouco_uso_manhã': 'Fechado, pouco uso, manhã',
    'dados_pc_fechado_pouco_uso_2_noite': 'Fechado, pouco uso, noite'
}

# Dicionário para armazenar os DataFrames
dataframes = {}

# Transforma cada coleção em DataFrame
for colecao in colecoes:
    # Acessa a coleção
    collection = db[colecao]
    # Método para acessar os documentos armazenados na coleção
    documents = collection.find({})
    # Cria uma lista com os dados de cada documento
    lista_documentos = [doc for doc in documents]
    # Cria o DataFrame a partir da lista de dicionários
    df = pd.DataFrame(lista_documentos)
    # Armazena o DataFrame no dicionário usando o nome da coleção como chave
    dataframes[colecao] = df

# Calcula a média da temperatura para cada cenário
medias_temperatura = []
for colecao in colecoes:
    media = dataframes[colecao]['Temperatura'].mean()
    medias_temperatura.append(media)

# Configuração das cores para cada cenário
cores = ['#ff77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd', '#8c564b', '#e377c2', '#7f7f7f']

# Cria o gráfico de barras com as médias de temperatura para cada cenário
plt.figure(figsize=(11, 6)) # Tamanho do gráfico ajustado
bars = plt.bar(range(len(colecoes)), medias_temperatura, color=cores)
plt.xticks(range(len(colecoes)), [abreviacoes[colecao] for colecao in colecoes], rotation=45, ha='right') # Rótulos ajustados
plt.xlabel('Cenários')
plt.ylabel('Média de Temperatura')
plt.title('Média de Temperatura para cada Cenário')

# Adicionar rótulos nas barras
for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width() / 2, height, round(height, 2), ha='center', va='bottom')

plt.tight_layout()
plt.savefig('/content/grafico_media_temperatura.png')
plt.show()

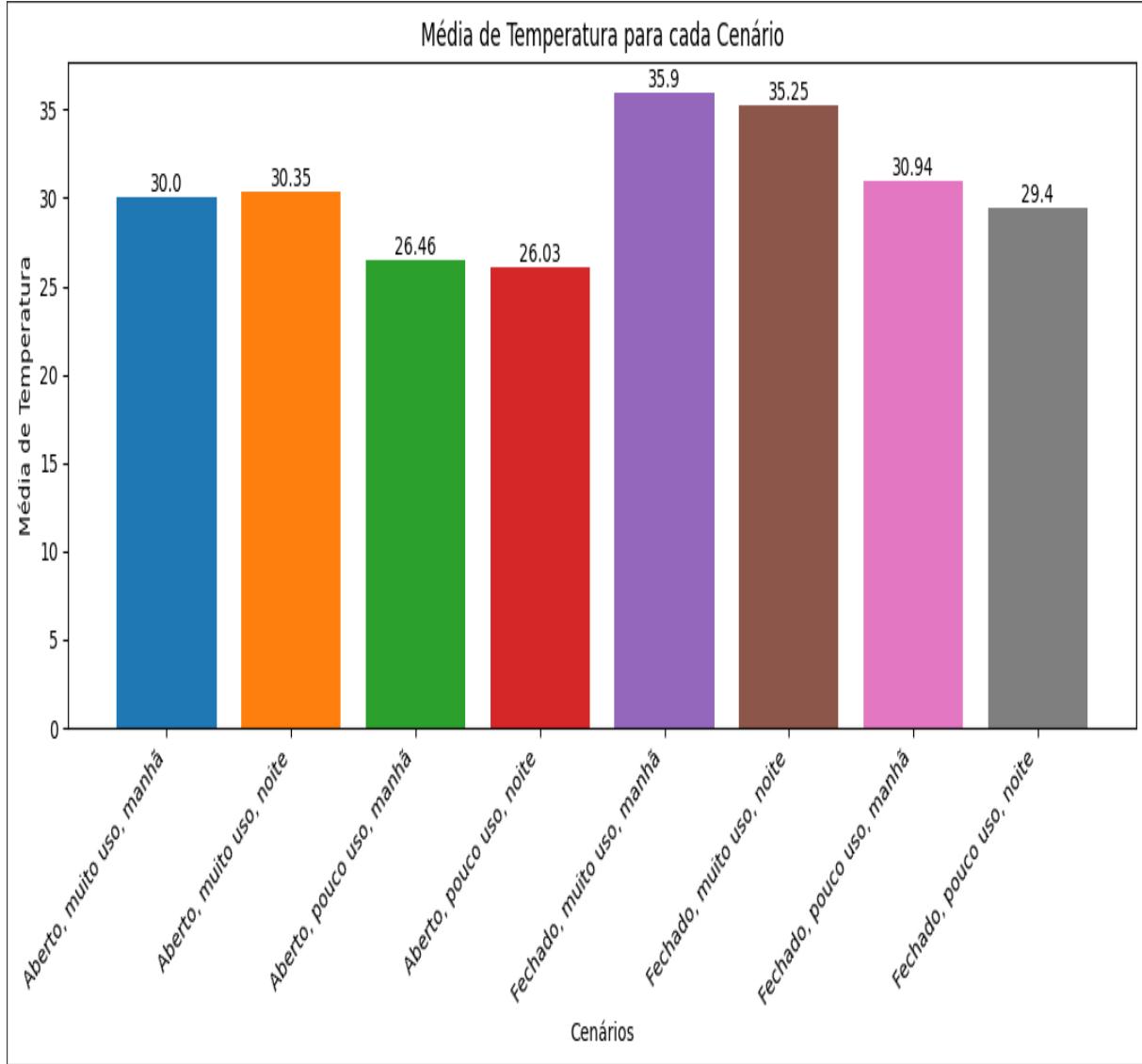
```

Fonte: Autoria própria (2023)

5.6 Gráfico das médias de cada cenário

O gráfico mostrado na figura 5.6 mostra a média de temperatura nos cenários em que o computador foi submetido.

Figura 5.6 Gráfico da média de cada cenário



Fonte: Autoria própria (2023)

Na figura 5.6, percebe-se um aumento de temperatura média significante da coluna "fechado com muito uso" para "aberto com muito uso", pois com o gabinete fechado, a tendência é não ter muitas áreas de ventilação, consequentemente aumentando a temperatura. Uma solução para tal problema seria a instalação de ventoinhas e coolers eficientes para que o computador pudesse ser melhor ventilado.

5.7 Código para plotagem dos gráficos do computador fechado e aberto

O código mostrado a seguir demonstra como foi feito a plotagem dos gráficos. Foram divididos em computador fechado e aberto, com pouco uso e muito uso, manhã e noite.

Figura 5.7 Código de gráfico de cada cenário

```
# Criar uma figura com uma grade de 2 por 2
fig, axes = plt.subplots(2, 2, figsize=(16, 10))
fig.suptitle('Temperaturas com PC Fechado', fontsize=16)

# Primeiro gráfico (linha 0, coluna 0)
ax1 = axes[0, 0]
ax1.plot(dataframes['dados_pc_fechado_pouco_uso_manhã']['Temperatura'], label='Manhã', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax1.plot(dataframes['dados_pc_fechado_pouco_uso_2_noite']['Temperatura'], label='Noite', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax1.set_xlabel('Amostras')
ax1.set_ylabel('Temperaturas')
ax1.legend(loc='lower right')
ax1.set_title('Pouco Uso')
ax1.set_ylim(24,40)
ax1.set_yticks(range(24, 39, 2))

# Segundo gráfico (linha 0, coluna 1)
ax2 = axes[0, 1]
ax2.plot(dataframes['dados_pc_fechado_muito_uso_manhã']['Temperatura'], label='Manhã', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax2.plot(dataframes['dados_pc_fechado_muito_uso_2_noite']['Temperatura'], label='Noite', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax2.set_xlabel('Amostras')
ax2.set_ylabel('Temperaturas')
ax2.legend(loc='lower right')
ax2.set_title('Muito Uso')
# Padroniza o intervalos da temperatura
ax2.set_ylim(24, 40)
ax2.set_yticks(range(24, 39, 2))
# Segundo Titulo
fig.text(0.5, 0.49, 'Temperaturas com PC Aberto', ha='center', va='center', fontsize=16)

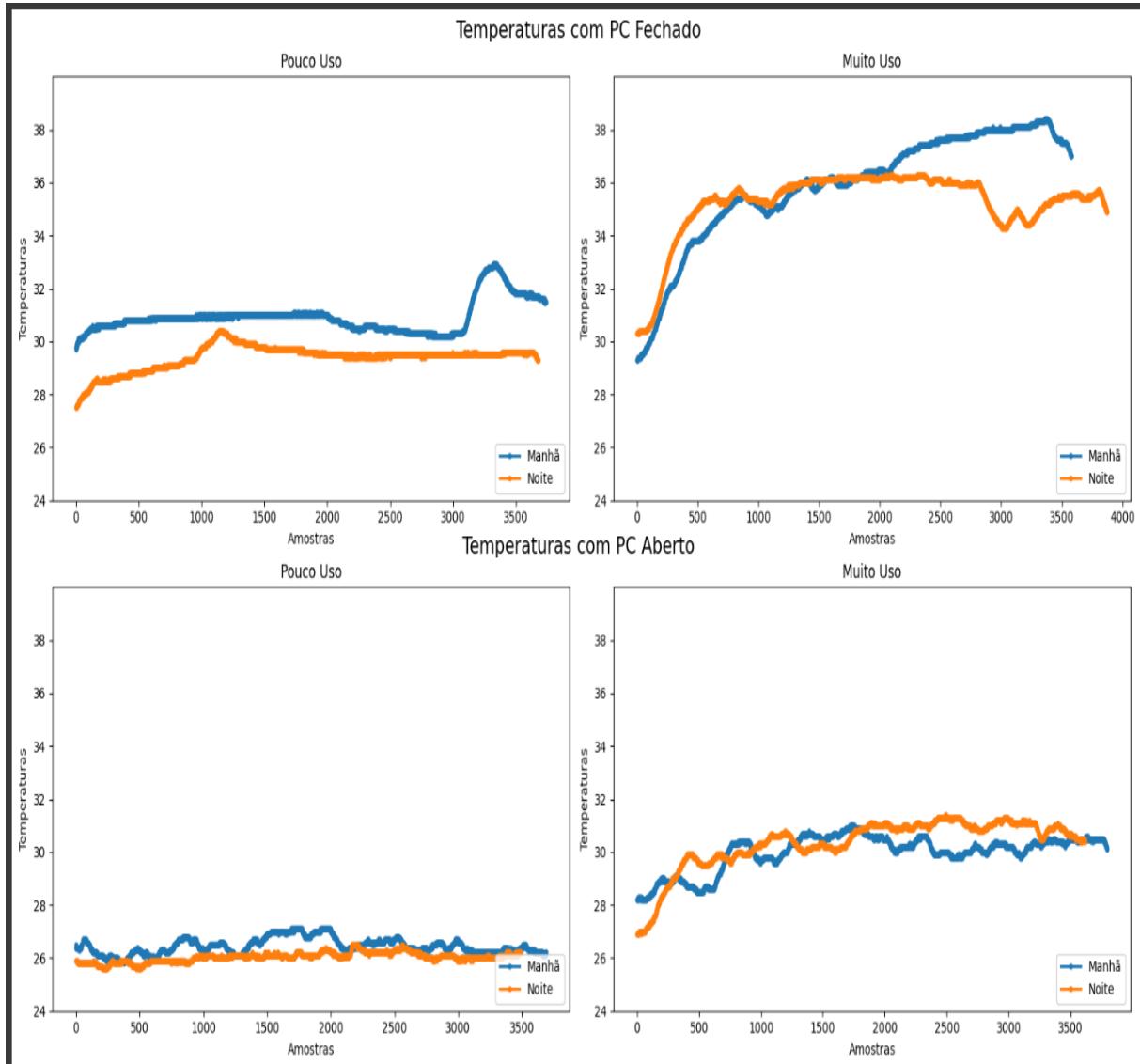
# Terceiro gráfico (linha 1, coluna 0)
ax3 = axes[1, 0]
ax3.plot(dataframes['dados_pc_aberto_pouco_uso_manhã']['Temperatura'], label='Manhã', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax3.plot(dataframes['dados_pc_aberto_pouco_uso_2_noite']['Temperatura'], label='Noite', marker='o', markersize=3, linestyle='solid', lw=2.0)
ax3.set_xlabel('Amostras')
ax3.set_ylabel('Temperaturas')
ax3.set_title('Pouco Uso')
ax3.legend(loc='lower right')
ax3.set_ylim(24, 40)
ax3.set_yticks(range(24, 39, 2))
# Quarto gráfico (linha 1, coluna 1)
ax4 = axes[1, 1]
```

Fonte: Autoria própria (2023)

5.8 Gráficos do computador fechado e aberto

O gráfico mostrado a seguir foram divididos em computador fechado e aberto, com pouco uso e muito uso, manhã e noite.

Figura 5.8 Gráfico de cada cenário



Fonte: Autoria própria (2023)

De acordo com a figura 5.8, percebe-se que os recursos do computador utilizados com pouca e muita frequência, desde que o gabinete esteja aberto, não há uma elevação significativa de temperatura, devido à boa ventilação que os componentes estão submetidos. Contudo, com o gabinete fechado, submetido ao calor que ocorre devido ao forte poder de processamento, as temperaturas são altamente elevadas, acarretando mau funcionamento dos recursos do computador.

6

Considerações finais

Diante do exposto, percebe-se que um computador bem arejado, com o ar bem distribuído, é essencial para o bom funcionamento do mesmo, pois o calor em excesso, pode diminuir o desempenho ou até mesmo danificar alguns componentes. Assim sendo, o projeto teve um foco maior com gamers, onde se deposita um bom número de pessoas e de máquinas um tanto quanto potentes, que exigem um bom poder de processamento.

Além disso, com os dados coletados e analisados, notou-se que em alguns cenários houve uma alta elevação da temperatura. Com isso, houve uma queda significativa de desempenho do computador, devido sua alta temperatura, alguns programas e recurso foram prejudicados, acarretando avisos de alta temperatura e travamento de softwares.

Desse modo, apesar do projeto ser apenas um protótipo, obtiveram-se bons resultados e, por meio de pesquisas, tem-se um público alvo muito grande, devido ao vasto mercado gamer, máquinas cada vez mais poderosas estão presentes atualmente, exigindo um bom resfriamento dos componentes para que se obtenha bons desempenhos para os gamers.

Referências

ARDUINO® UNO R3. Acessado em 17 de Julho. Disponível em:
<https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf>.

ARDUÍNO. Acessado em 17 de Julho. Disponível em:
<https://docs.arduino.cc/hardware/uno-rev3>.

CANALTECH. Acessado em 18 de Julho. Disponível em :
<https://canaltech.com.br/hardware/qual-e-a-temperatura-ideal-dos-componentes-do-pc/>.

CITYOS. Acessado em 17 de Julho. Disponível em:
<https://cityos-air.readme.io/docs/4-dht22-digital-temperature-humidity-sensor>.

DELL Technologies. Acessado em 18 de Julho. Disponível em:
<https://www.dell.com/support/kbdoc/pt-br/000130867/como-solucionar-um-problema-t%C3%A9rmico-de-superaquecimento-ou-de-desligamento-em-um-computador-dell: :text=Eventualmente%2C%20o%20calor%20excessivo%20pode,remover%20o%20excesso%20de%20>

PANDAS Biblioteca. Acessado em 18 de Julho. Disponível em : <https://pandas.pydata.org/>.

PYTHON. Acessado em 18 de Julho. Disponível em : <https://www.python.org/>.