



**INSTITUTO FEDERAL**

Brasília  
Campus Brasília

**TECNOLOGIA EM SISTEMAS PARA INTERNET**

**Claudio Lucas de Oliveira Franco**

**Jean Pereira Ribeiro**

**Marcos Vinicius Lira de Oliveira**

**Rafael Souza Nunes**

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

**Brasília - DF**

**23/12/2022**

# Sumário

<b>1. Objetivos</b>	<b>3</b>
<b>2. Descrição do problema</b>	<b>4</b>
<b>3. Desenvolvimento</b>	<b>5</b>
3.1 Código implementado	12
<b>4. Considerações finais</b>	<b>22</b>
<b>Referências</b>	<b>23</b>

# 1. Objetivos

O objetivo deste relatório é descrever o processo de desenvolvimento da Prática Integrada de Introdução a Ciência de Dados e Internet das Coisas. O projeto escolhido pelo grupo foi o da “Caixa para controle de umidade de filamento para impressora 3D”.

Para a sprint 1, os objetivos foram: montar o hardware e configurá-lo para funcionar adequadamente. Após a finalização dessa sprint, iniciou-se a sprint 2, em que os objetivos eram: definir o contexto de aplicação, coletar e preparar os dados. Em relação a sprint 3, os objetivos eram: resolver problemas encontrados nas sprints anteriores, armazenar os dados no MongoDB e analisar os dados.

## 2. Descrição do problema

Em relação ao projeto escolhido, o problema envolve a umidade no filamento que resulta em problemas de impressão e perdas do material. Para resolver esse problema, é necessário evitar ou remover a umidade no material para que se tenha uma impressão adequada (TECNOCUBO, 2022).

Nesse sentido, o contexto de aplicação desse projeto é orientar o leitor a realizar boas praticas visando eludir problemas ao iniciar uma impressão.

Os materiais geralmente utilizados para impressão 3D absorvem de forma natural a umidade. Os mais utilizados são o filamento PLA (ácido polilático), filamento Nylon, filamento ABS (acrilonitrila butadieno estireno) entre outros. O que pode acarretar em alguns problemas como uma peça quebradiça, camadas não uniformes, rompimento do filamento ou entupimento do bico.

Figura 1 – A esquerda um vaso impresso em PLA com umidade e à direita com o mesmo material em perfeito estado.



Fonte: 3DLab, 2016.

Temos algumas soluções disponíveis para resolver este problema, entretanto a mais viável é o armazenamento em ambiente adequado, inserindo ao nosso contexto vamos utilizar uma caixa de plástico com desumidificador e algumas fontes de calor para diminuir a umidade dentro da caixa e para armazenar de forma correta o filamento visando garantir uma vida útil maior aos materiais e consequentemente constituindo impressões com melhores resultados.

### 3. Desenvolvimento

Para a realização do projeto, seguimos o passo a passo mostrado no artigo “Sensores DHT11 e DHT22: Guia Básico dos Sensores de Umidade e Temperatura” (MURTA, 2019).

Os materiais utilizados foram:

- 1 Arduino Uno R3 + Cabo Usb para Arduino
- 1 Protoboard 400 Pontos
- 3 Jumpers – Macho/Macho de 20cm
- 1 Sensor de Umidade e Temperatura DHT11

Os materiais foram adquiridos pela internet. Após a entrega, montamos o hardware de acordo com o Diagrama Fritzing a seguir:

Figura 2 – Diagrama Fritzing demonstrando o esquema de montagem do sensor DHT11 com o Arduino.

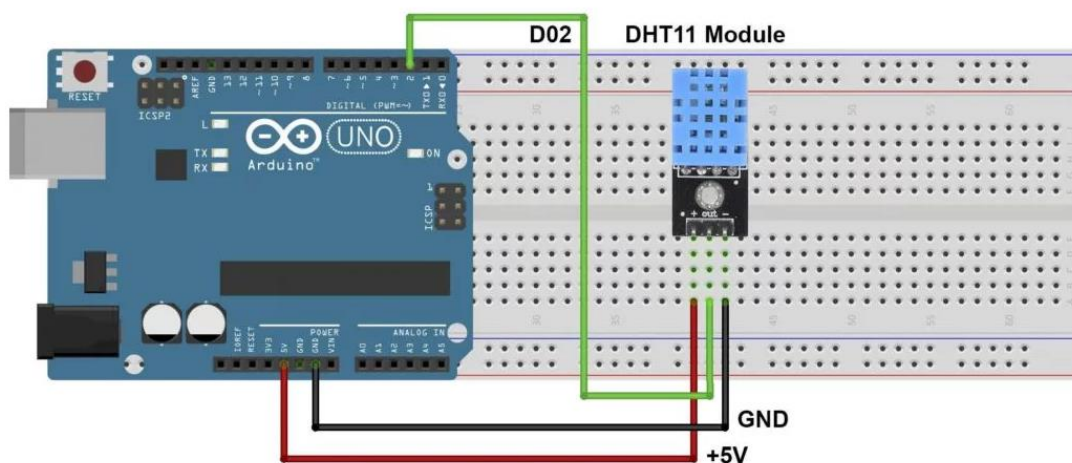
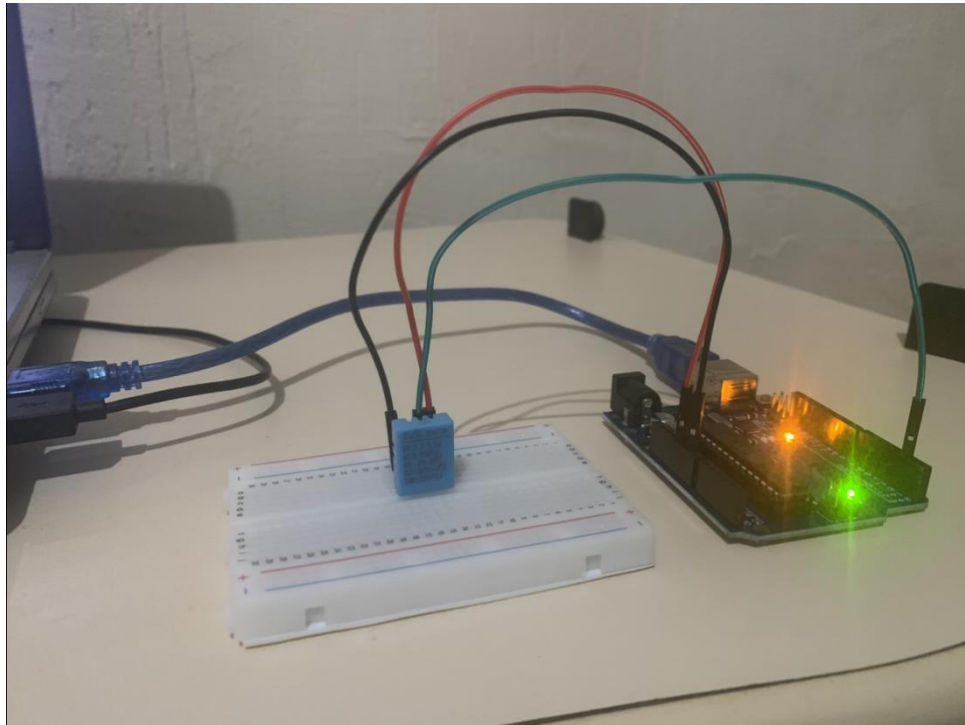


Diagrama Fritzing – Arduino e sensor DHT11

Fonte: Blog Eletrogate, 2009.

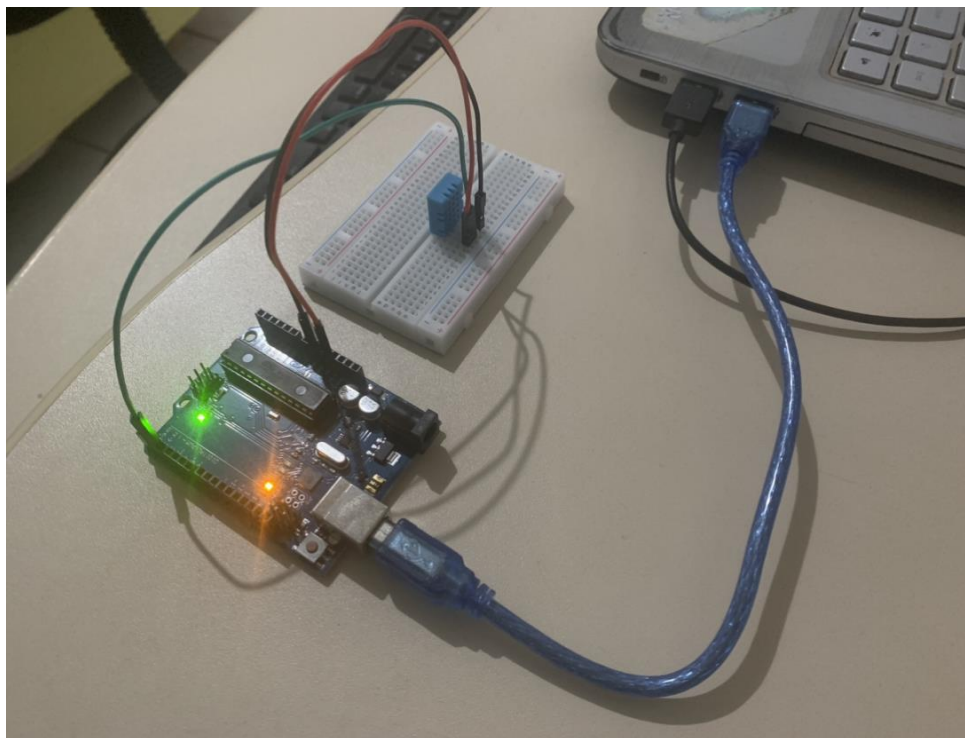
As imagens abaixo mostram o hardware montado:

Figura 3: Hardware montado (visto de frente)



Fonte: Elaborado pelos autores.

Figura 4 – Hardware montado (visto por cima)



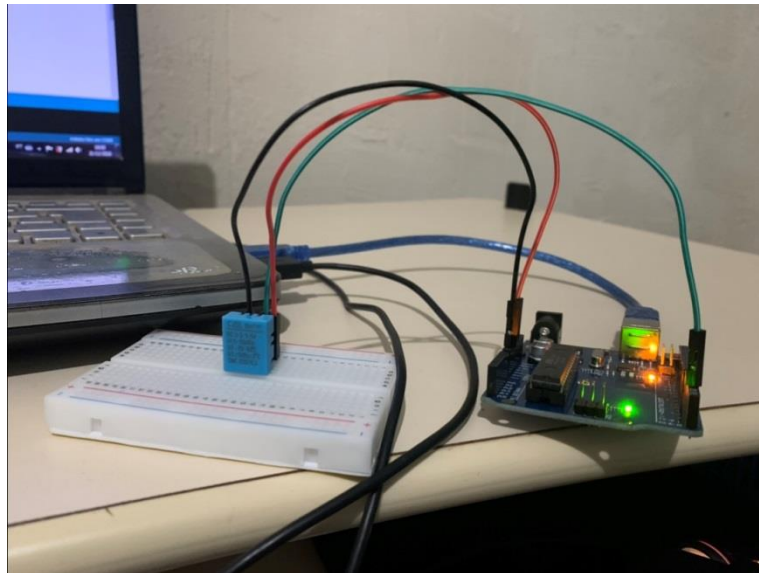
Fonte: Elaborado pelos autores.

Para a etapa de coleta e preparação dos dados, o processo desenvolvido foi o seguinte:

Definimos o contexto e os dados que seriam coletados. Ao entender o problema que teríamos que resolver (evitar a umidade no filamento), decidimos coletar os dados de umidade e temperatura em 3 situações diferentes:

1 - Na coleta 1, os dados coletados vieram de uma sala normal, simulando a situação do filamento ser deixado desprotegido em um cômodo.

Figura 5 – Coleta 1



Fonte: Elaborado pelos autores.

2 - Na coleta 3, os dados coletados vieram de dentro de uma caixa de plástico fechada contendo um desumidificador e uma fonte de calor (notebook), simulando a situação do filamento ser guardado em um ambiente projetado para a diminuição da umidade. O anti mofo é um desumidificador de ambientes, que tem a função de absorver as micropartículas de água do ar, condensando-as e armazenando no próprio recipiente do produto.

Figura 6 – Coleta 2 (imagem 1)

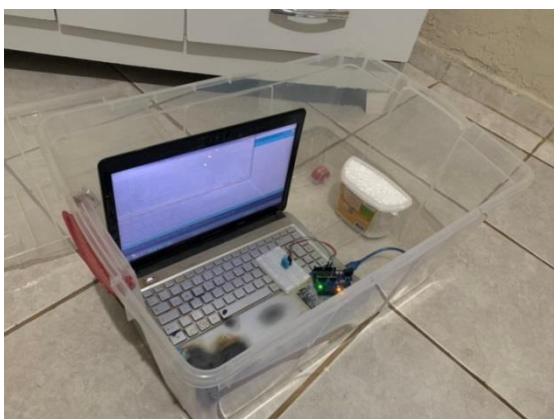
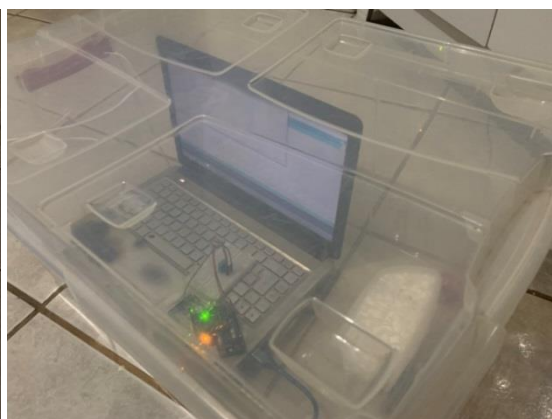


Figura 7 – Coleta 2 (imagem 2)





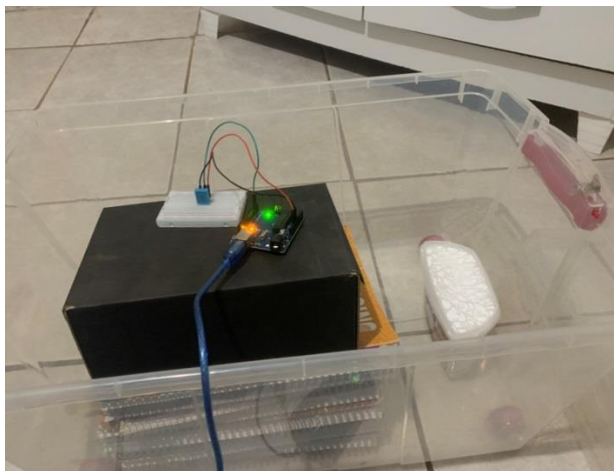
Fonte: Elaborado pelos autores.

Fonte: Elaborado pelos autores.

Na coleta 2, a caixa precisava ser fechada para coletar os dados, então precisamos colocar o sensor junto com a fonte de energia, no caso um notebook. O calor emitido pelo notebook pode ter contribuído para o aumento da temperatura e consequente diminuição da umidade. Pensando na relação entre o aumento da temperatura e diminuição da umidade, as três coletas seguintes foram feitas com o notebook fora da caixa e duas delas usando fontes de calor alternativas, com o objetivo de aumentar a temperatura dentro da caixa.

3 – Na coleta 3, os dados foram coletados de dentro de uma caixa de plástico fechada contendo livros e uma caixa de papelão, que serviram de apoio ao sensor, e um desumidificador. O objetivo era testar se a umidade reduziria apenas com o desumidificador, sem o uso de fontes de calor.

Figura 8 – Coleta 3



Fonte: Elaborado pelos autores.

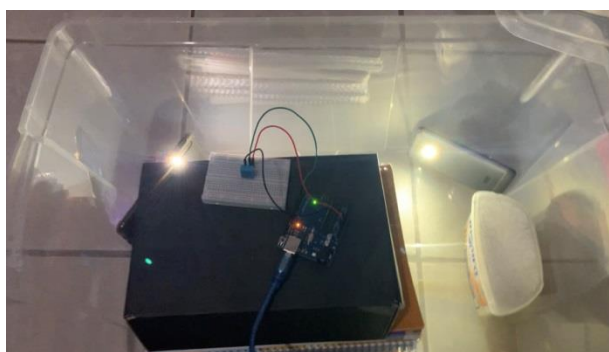
Figura 9 – Desumidificador



Fonte: Elaborado pelos autores.

4 – Na coleta 4, os dados foram coletados de dentro de uma caixa de plástico fechada contendo livros e uma caixa de papelão, que serviram de apoio ao sensor, um desumidificador e 3 lanternas de celulares, que tinham como objetivo aumentar a temperatura dentro da caixa.

Figura 10 – Coleta 4





Fonte: Elaborado pelos autores.

5 – Na coleta 5, os dados coletados vieram de dentro de uma caixa de plástico fechada contendo apenas livros e uma caixa de papelão, que serviram de apoio ao sensor. Nessa coleta, a cada aproximadamente 5 minutos foi ligado um secador de cabelo de 1200 W com ar quente por 1 minuto. O objetivo era aumentar a temperatura e verificar se haveria diminuição na umidade.

Figura 11 – Coleta 5

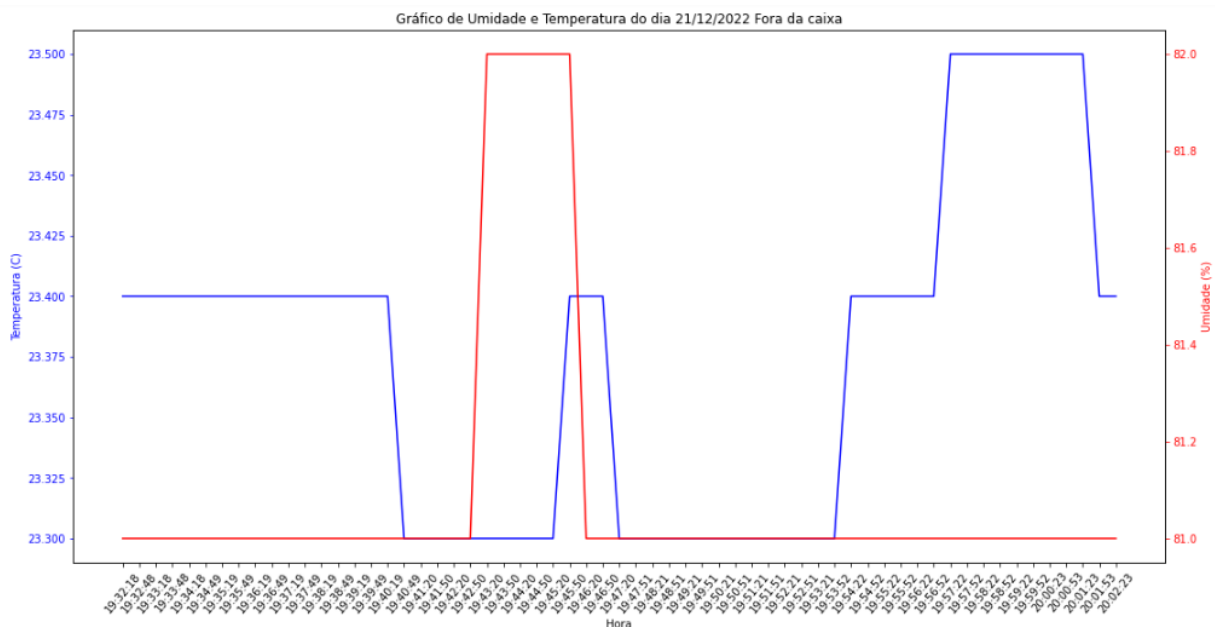


Fonte: Elaborado pelos autores.

Com essas cinco coletas, o nosso objetivo era coletar os dados e verificar se haveria diferença na temperatura e umidade em cada ambiente. Para isso, decidimos coletar a temperatura e umidade de cada ambiente a cada 30 segundos durante 30 minutos.

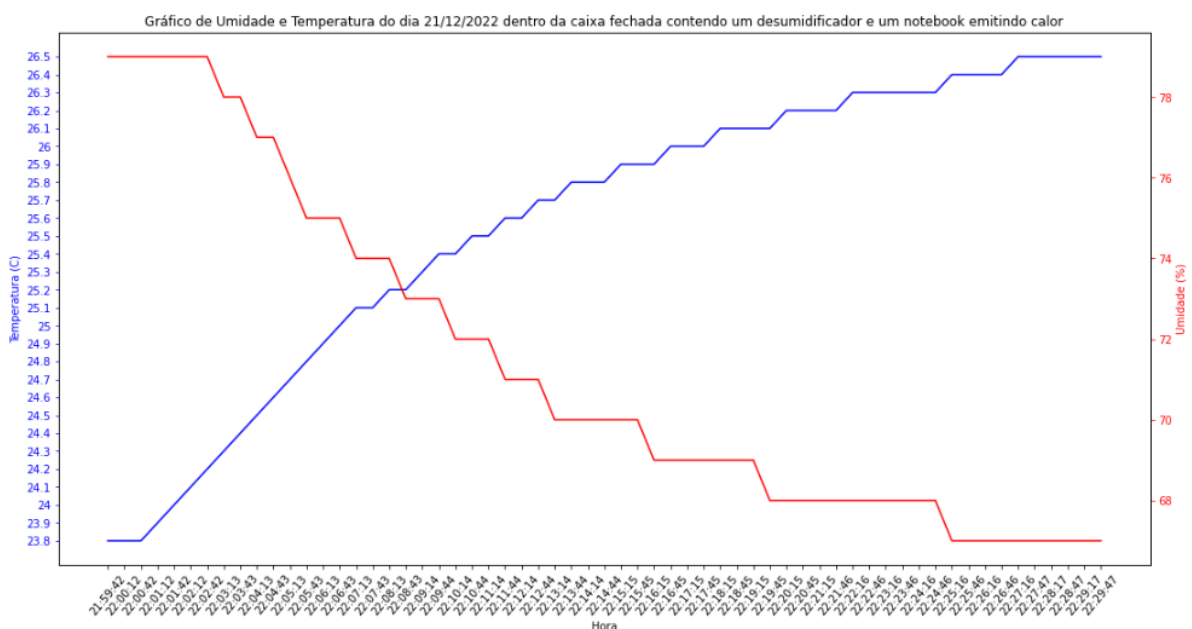
Os resultados foram os seguintes:

Figura 12 – Gráfico da coleta 1:



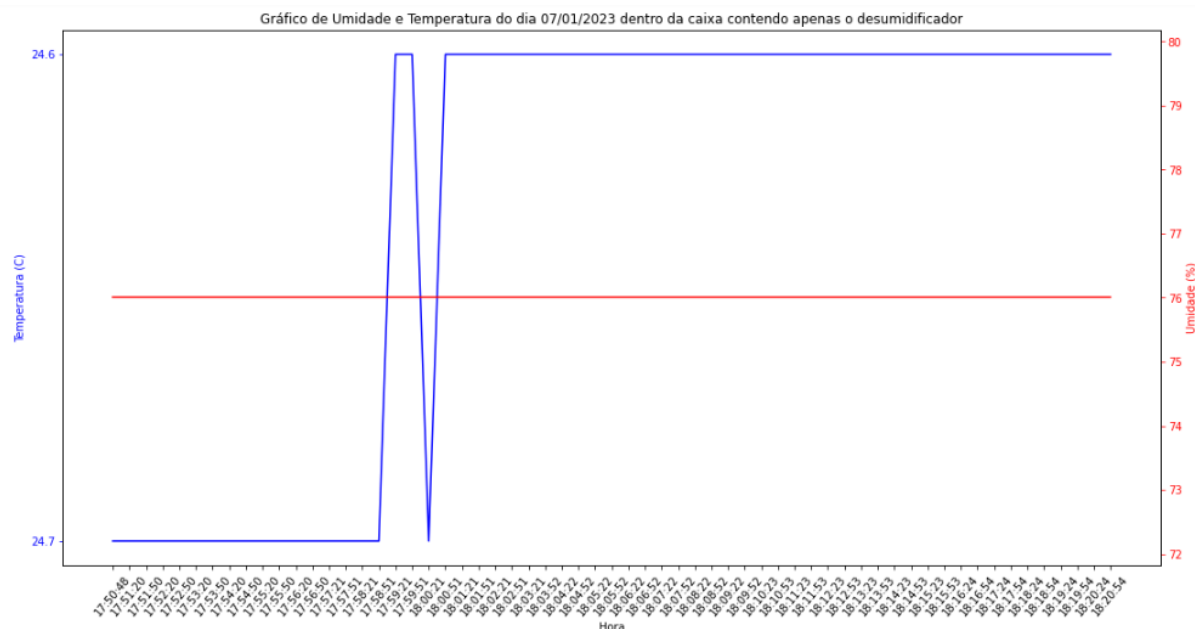
Na coleta 1, a temperatura variou entre 23,3 °C e 23,5 °C e a umidade variou entre 81% e 82%. Ou seja, se manteve estável durante os 30 minutos.

Figura 13 – Gráfico da coleta 2:



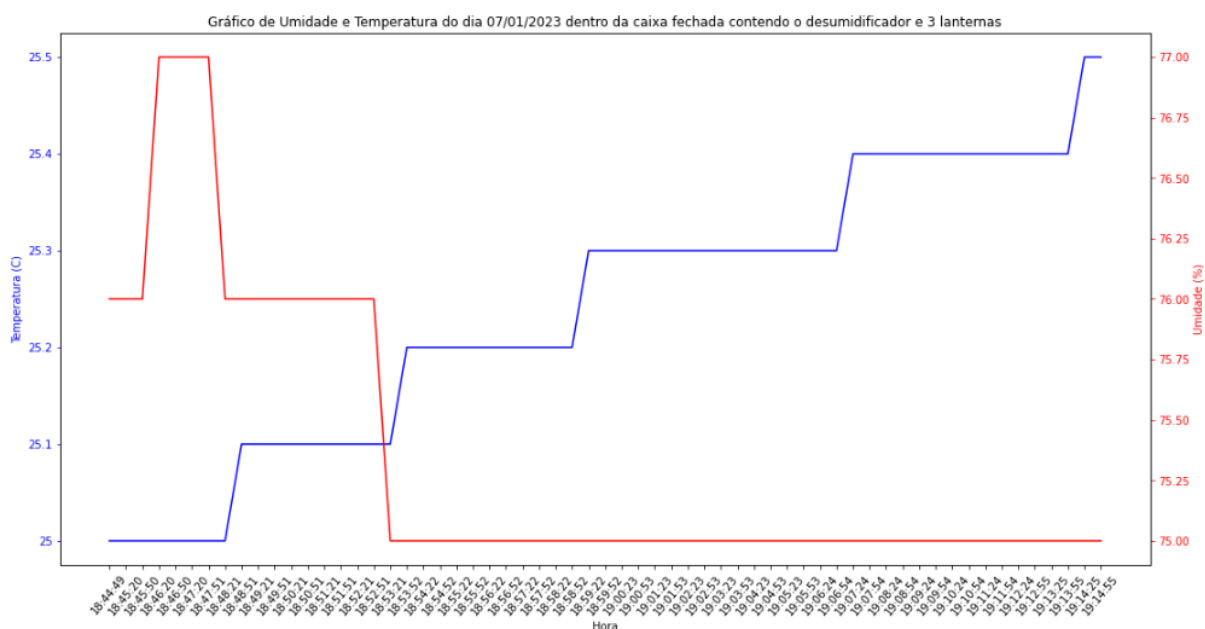
Na coleta 2, a medição de temperatura iniciou-se em 23,8 °C e terminou em 26,5 °C. A umidade inicial foi 79% e terminou em 67%. Ou seja, houve variação nos dados.

Figura 14 – Gráfico da coleta 3:



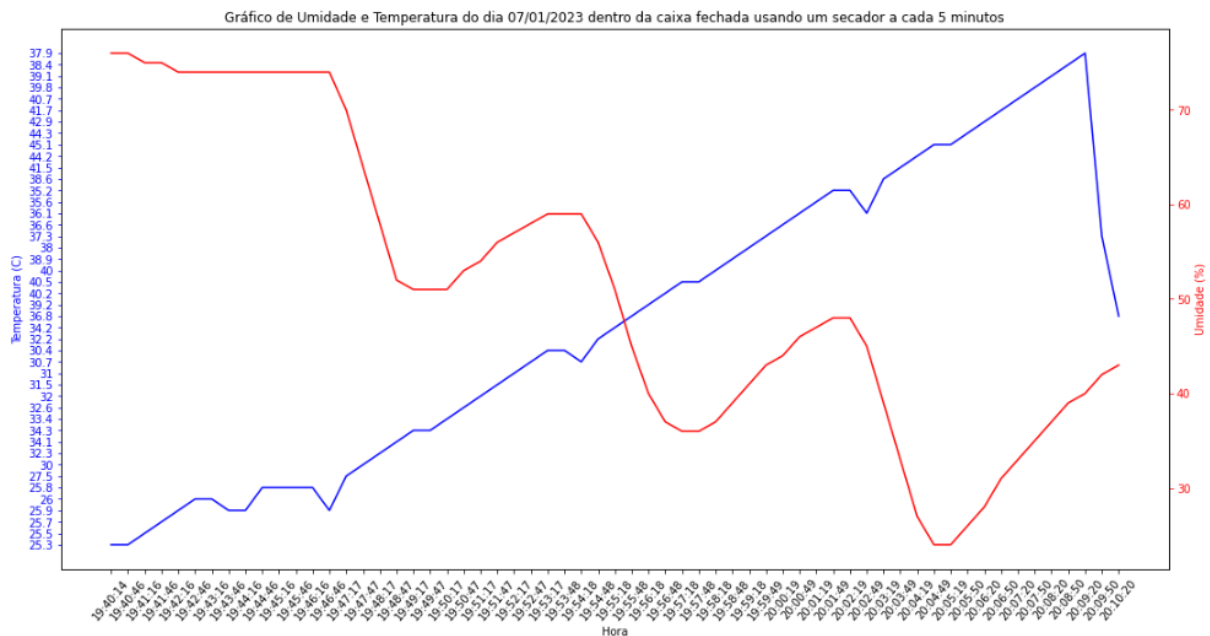
Na coleta 3, a umidade e a temperatura se mantiveram constante, o que mostrou que apenas o desumidificador não foi suficiente para diminuir a umidade.

Figura 15 – Gráfico da coleta 4:



Na coleta 4, a temperatura aumentou gradativamente de 25 C° para 25,5 C°, o que não resultou na diminuição da umidade, o que indica que as 3 lanternas não foram eficientes para aumentar a temperatura.

Figura 16 – Gráfico da coleta 5:



Na coleta 5, a temperatura aumentou continuamente, com maior visibilidade nos 3 momentos em que o secador foi ligado. O aumento da temperatura causou uma diminuição imediata da umidade. No minuto que o secador foi ligado, a umidade desceu imediatamente, mas assim que o secador foi desligado, a umidade voltou a subir.

Os dados completos de cada coleta em formato CSV estão disponíveis na pasta “sprint 3” no GitHub, que pode ser acessado através desse link: <https://github.com/infocbra/pratica-integrada-cd-e-ic-2022-2-g2-cmrj>

Ao analisar os resultados, concluímos que a caixa projetada funciona melhor quando há fontes de calor para aumentar a temperatura. Nesse sentido, com o objetivo de diminuir ao máximo a umidade dentro da caixa, os ambientes feitos para as coletas 2 (caixa fechada com um umidificador e um notebook - fonte de calor) e 5 (caixa fechada com um secador ligado a cada 5 minutos) foram os mais efetivos.

O ambiente 2 foi o mais próximo do ideal, com a temperatura subindo gradativamente e a umidade descendo proporcionalmente. O ambiente 5 foi o que mais reduziu a umidade, porém é o mais difícil de manter por longos períodos, considerando que o secador precisaria ser ligado após determinados períodos de tempo.

## 3.1 Código implementado

Para a sprint 1, o código implementado para o recebimento dos dados do sensor contendo os comentários explicando o funcionamento de cada parte foi o seguinte:

Figura 17 – Código do arduino sprint 1 (parte 1)



```
sketch_dec08a | Arduino 1.8.19
Arquivo Editar Sketch Ferramentas Ajuda

sketch_dec08a $
#include <Adafruit_Sensor.h> // Biblioteca DHT Sensor Adafruit
#include <DHT.h>
#include <DHT_U.h>
#define DHTTYPE DHT11 // Sensor DHT11
#define DHTPIN 2 // Pino do Arduino conectado no Sensor(Data)
DHT_Unified dht(DHTPIN, DHTTYPE); // configurando o Sensor DHT - pino e tipo
uint32_t delayMS; // variável para atraso no tempo
void setup()
{
  Serial.begin(9600); // monitor serial 9600 bps
  dht.begin(); // inicializa a função
  Serial.println("Usando o Sensor DHT");
  sensor_t sensor;
```

Fonte: Elaborado pelos autores

Figura 18 – Código do arduino sprint 1 (parte 2)

```
dht.begin(); // inicializa a função
Serial.println("Usando o Sensor DHT");
sensor_t sensor;
dht.temperature().getSensor(&sensor); // imprime os detalhes do Sensor de Temperatura
Serial.println("-----");
Serial.println("Temperatura");
Serial.print ("Sensor: "); Serial.println(sensor.name);
Serial.print ("Valor max: "); Serial.print(sensor.max_value); Serial.println(" *C");
Serial.print ("Valor min: "); Serial.print(sensor.min_value); Serial.println(" *C");
Serial.print ("Resolucao: "); Serial.print(sensor.resolution); Serial.println(" *C");
Serial.println("-----");
dht.humidity().getSensor(&sensor); // imprime os detalhes do Sensor de Umidade
Serial.println("-----");
Serial.println("Umidade");
Serial.print ("Sensor: "); Serial.println(sensor.name);
Serial.print ("Valor max: "); Serial.print(sensor.max_value); Serial.println("%");
Serial.print ("Valor min: "); Serial.print(sensor.min_value); Serial.println("%");
Serial.print ("Resolucao: "); Serial.print(sensor.resolution); Serial.println("%");
Serial.println("-----");
```

Fonte: Elaborado pelos autores

Figura 19 – Código do arduino sprint 1 (parte 3)

```

    delayMS = sensor.min_delay / 1000;           // define o atraso entre as leituras
}
void loop()
{
    delay(delayMS);                             // atraso entre as medições
    sensors_event_t event;                      // inicializa o evento da Temperatura
    dht.temperature().getEvent(&event);          // faz a leitura da Temperatura
    if (isnan(event.temperature))                // se algum erro na leitura
    {
        Serial.println("Erro na leitura da Temperatura!");
    }
    else                                         // senão
    {
        Serial.print("Temperatura: ");          // imprime a Temperatura
        Serial.print(event.temperature);
        Serial.println(" *C");
    }
    dht.humidity().getEvent(&event);             // faz a leitura de umidade
    if (isnan(event.relative_humidity))          // se algum erro na leitura
    {

```

Fonte: Elaborado pelos autores

Figura 20 – Código do arduino sprint 1 (parte 4)

```

        Serial.println("Erro na leitura da Umidade!");
    }
    else                                         // senão
    {
        Serial.print("Umidade: ");              // imprime a Umidade
        Serial.print(event.relative_humidity);
        Serial.println("%");
    }
}

```

Fonte: Elaborado pelos autores

A imagem a seguir mostra a saída desse código e os dados que foram enviados pelo sensor. Na imagem é possível ver informações sobre o sensor, a hora da coleta e os dados de temperatura e a umidade.

Figura 21 – Saída do código do arduino sprint 1

```
COM3
22:53:00.769 -> Usando o Sensor DHT
22:53:02.399 -> -----
22:53:02.432 -> Temperatura
22:53:02.466 -> Sensor: DHT11
22:53:02.466 -> Valor max: 50.00 *C
22:53:02.500 -> Valor min: 0.00 *C
22:53:02.534 -> Resolucao: 2.00 *C
22:53:02.534 -> -----
22:53:02.568 -> -----
22:53:02.636 -> Umidade
22:53:02.636 -> Sensor: DHT11
22:53:02.670 -> Valor max: 80.00%
22:53:02.670 -> Valor min: 20.00%
22:53:02.704 -> Resolucao: 5.00%
22:53:02.704 -> -----
22:53:03.720 -> Temperatura: 27.50 *C
22:53:03.754 -> Umidade: 69.00%
22:53:04.704 -> Temperatura: 27.50 *C
22:53:04.738 -> Umidade: 69.00%
22:53:05.755 -> Temperatura: 27.50 *C
22:53:05.755 -> Umidade: 69.00%
22:53:06.740 -> Temperatura: 27.50 *C
22:53:06.774 -> Umidade: 69.00%
22:53:07.793 -> Temperatura: 27.50 *C
22:53:07.793 -> Umidade: 69.00%
22:53:08.778 -> Temperatura: 27.50 *C
22:53:08.812 -> Umidade: 69.00%
22:53:09.797 -> Temperatura: 27.50 *C
22:53:09.831 -> Umidade: 69.00%
22:53:10.821 -> Temperatura: 27.50 *C
22:53:10.821 -> Umidade: 69.00%
22:53:11.840 -> Temperatura: 27.50 *C
22:53:11.874 -> Umidade: 69.00%
22:53:12.826 -> Temperatura: 27.50 *C
22:53:12.860 -> Umidade: 69.00%
22:53:13.878 -> Temperatura: 27.50 *C
22:53:13.878 -> Umidade: 69.00%
22:53:14.862 -> Temperatura: 27.50 *C
☐ Auto-rolagem ☐ Show timestamp
```

Fonte: Elaborado pelos autores.

Para a sprint 2, o código usado foi praticamente o mesmo, contendo apenas uma modificação no tempo de coleta dos dados. Antes, os dados do sensor eram lidos a cada 1 segundo. Com essa modificação, os dados são lidos a cada 30 segundos.

Figura 22 – Modificação no atraso entre as leituras

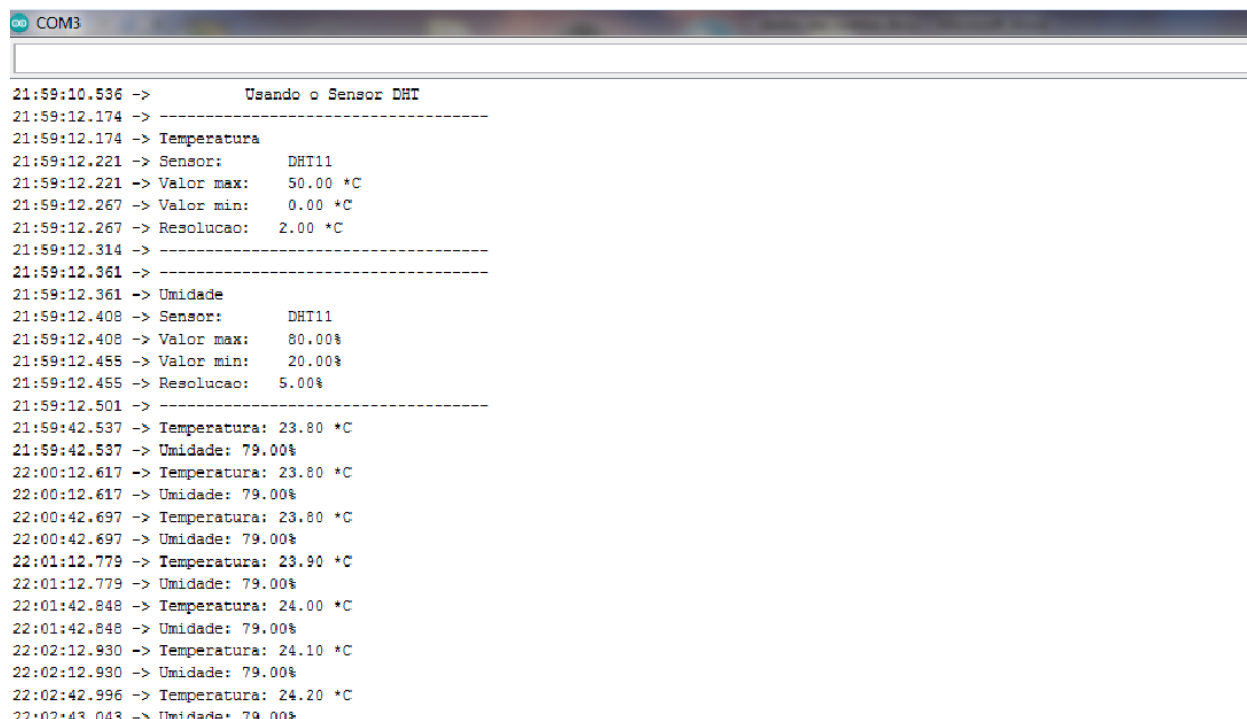
```
delayMS = sensor.min_delay / 33.3; // define o atraso entre as leituras
}
void loop()
{
    delay(delayMS); // atraso entre as medições
    sensors_event_t event; // inicializa o evento da Temperatura
    dht.temperature().getEvent(&event); // faz a leitura da Temperatura
    if (isnan(event.temperature)) // se algum erro na leitura
    {
        Serial.println("Erro na leitura da Temperatura!");
    }
    else // senão
    {
        Serial.print("Temperatura: "); // imprime a Temperatura
        Serial.print(event.temperature);
        Serial.println(" *C");
    }
    dht.humidity().getEvent(&event); // faz a leitura de umidade
    if (isnan(event.relative_humidity)) // se algum erro na leitura
    {
```

Fonte: Elaborado pelos autores.



A figura a seguir mostra um exemplo de saída desse código utilizado na sprint 2. Essa foi a saída registrada na coleta 3 que foi explicada anteriormente. Na imagem é possível ver informações sobre o sensor, a hora da coleta e os dados de temperatura e a umidade.

Figura 23 – Saída do código do arduino da sprint 2



```
21:59:10.536 -> Usando o Sensor DHT
21:59:12.174 -> -----
21:59:12.174 -> Temperatura
21:59:12.221 -> Sensor: DHT11
21:59:12.221 -> Valor max: 50.00 *C
21:59:12.267 -> Valor min: 0.00 *C
21:59:12.267 -> Resolucao: 2.00 *C
21:59:12.314 -> -----
21:59:12.361 -> -----
21:59:12.361 -> Umidade
21:59:12.408 -> Sensor: DHT11
21:59:12.408 -> Valor max: 80.00%
21:59:12.455 -> Valor min: 20.00%
21:59:12.455 -> Resolucao: 5.00%
21:59:12.501 -> -----
21:59:42.537 -> Temperatura: 23.80 *C
21:59:42.537 -> Umidade: 79.00%
22:00:12.617 -> Temperatura: 23.80 *C
22:00:12.617 -> Umidade: 79.00%
22:00:42.697 -> Temperatura: 23.80 *C
22:00:42.697 -> Umidade: 79.00%
22:01:12.779 -> Temperatura: 23.90 *C
22:01:12.779 -> Umidade: 79.00%
22:01:42.848 -> Temperatura: 24.00 *C
22:01:42.848 -> Umidade: 79.00%
22:02:12.930 -> Temperatura: 24.10 *C
22:02:12.930 -> Umidade: 79.00%
22:02:42.996 -> Temperatura: 24.20 *C
22:02:43.043 -> Umidade: 79.00%
```

Fonte: Elaborado pelos autores

Para a sprint 3, conseguimos escrever os dados do arduino diretamente em um arquivo .csv utilizando o Python. Dessa forma, houveram modificações no código do arduino para facilitar a escrita do arquivo csv. A figura a seguir mostra o código do arduino:

Figura 24 - Código do arduino sprint 3 (parte 1)



```
sketch_jan05a | Arduino 1.8.19
Arquivo  Editar  Sketch  Ferramentas  Ajuda

sketch_jan05a

#include <Adafruit_Sensor.h>                // Biblioteca DHT Sensor Adafruit
#include <DHT.h>
#include <DHT_U.h>
#define DHTTYPE DHT11                        // Sensor DHT11
#define DHTPIN 2                            // Pino do Arduino conectado no Sensor(Data)
DHT_Unified dht(DHTPIN, DHTTYPE);           // configurando o Sensor DHT - pino e tipo
uint32_t delayMS;                          // variável para atraso no tempo
void setup()
{
  Serial.begin(9600);                        // monitor serial 9600 bps
  dht.begin();                              // inicializa a função
  sensor_t sensor;
  delayMS = sensor.min_delay / 33.3;         // define o atraso entre as leituras
}
void loop()
{
  delay(delayMS);                          // atraso entre as medições
  sensors_event_t event;                   // inicializa o evento da Temperatura
  dht.temperature().getEvent(&event);       // faz a leitura da Temperatura
  if (isnan(event.temperature))            // se algum erro na leitura
  {
    Serial.println("Erro na leitura da Temperatura!");
  }
  else                                     // senão
  {
    Serial.print(event.temperature);
    Serial.print(";");
  }
  dht.humidity().getEvent(&event);          // faz a leitura de umidade
```

Fonte: Elaborado pelos autores.

Figura 25 - Código do arduino sprint 3 (parte 2)

```
if (isnan(event.relative_humidity))        // se algum erro na leitura
{
  Serial.println("Erro na leitura da Umidade!");
}
else                                       // senão
{
  Serial.print(event.relative_humidity);
  Serial.println(";");
}
}
```

Fonte: Elaborado pelos autores.

O código em Python utilizado para escrever o arquivo .csv foi o seguinte:

Figura 26 – Código em Python para escrever o arquivo .csv

```
codigoaut.py X logger.csv
codigoaut.py > ...
1 import serial      #importa a biblioteca do arduino
2 import datetime    #importa a biblioteca de data e hora
3
4 porta = "COM3"     #porta do arduino conectada ao computador
5 baud = 9600        #baud rate
6 arquivo = "logger.csv" #arquivo .csv em que serão colocados os dados
7
8 ser = serial.Serial(porta,baud) #variável correspondente ao sensor
9 ser.flushInput()    #limpa a entrada caso tenha um dado fora do padrão
10 print("Abrindo Serial")
11
12 tabelas = "Data;Hora;Temperatura(C°);Umidade(%)\\n" #primeira linha do arquivo .csv
13 print(tabelas)
14 file = open(arquivo,"a") #abre o arquivo .csv
15 file.write(tabelas) #escreve a primeira linha do arquivo .csv
16
17 amostra = 70       #quantidade de dados que serão armazenados
18 linha = 0
19 while linha <= amostra: #loop para armazenar os dados
20     agora = datetime.datetime.now() #armazena o dia e a hora
21     agora_string = agora.strftime("%d/%m/%Y;%H:%M:%S;") #tratamento da string do dia e hora
22     sensor = str(ser.readline().decode("utf-8")) #leitura dos dados do sensor
23     dados = agora_string + sensor #concatenação dos dados que serão escritos
24     dados1 = dados[0:33] #tratamento para os dados serem escritos linha por linha
25     print(dados1)
26     file.write(dados1) #escreve o dado no arquivo .csv
27     linha = linha+1 #incremento para o loop
28
29 print("Final de leituras")
30 file.close()
31 ser.close()
32
```

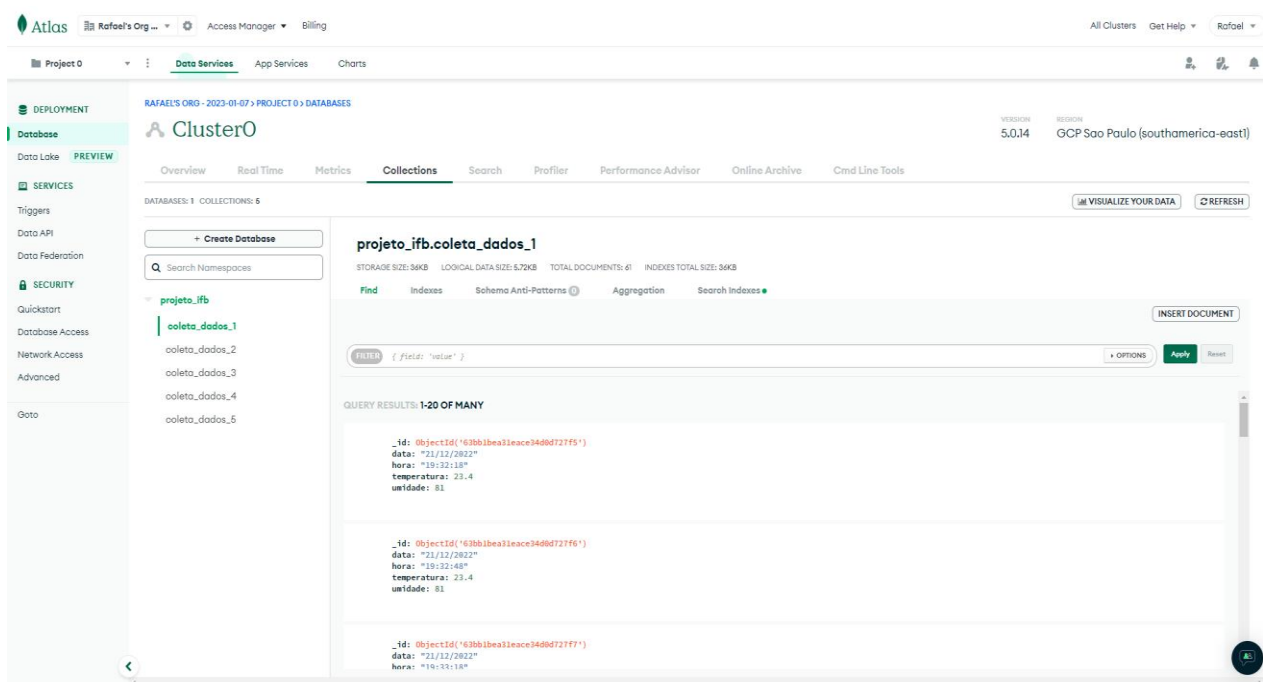
Figura 27 – Exemplo de arquivo .csv gerado

```
logger.csv - arquivos - Visual Studio Code
logger.csv
1 Data;Hora;Temperatura(C°);Umidade(%)
2 07/01/2023;16:02:38;24.60;71.00;
3 07/01/2023;16:03:10;24.60;71.00;
4 07/01/2023;16:03:40;24.60;72.00;
5 07/01/2023;16:04:10;24.60;72.00;
6 07/01/2023;16:04:40;24.60;72.00;
7 07/01/2023;16:05:10;24.60;72.00;
8 07/01/2023;16:05:40;24.60;72.00;
9 07/01/2023;16:06:10;24.60;72.00;
10 07/01/2023;16:06:40;24.60;72.00;
11 07/01/2023;16:07:10;24.60;72.00;
12 07/01/2023;16:07:41;24.60;72.00;
13 07/01/2023;16:08:11;24.60;72.00;
14 07/01/2023;16:08:41;24.60;72.00;
15 07/01/2023;16:09:11;24.60;72.00;
16 07/01/2023;16:09:41;24.60;72.00;
17 07/01/2023;16:10:11;24.60;72.00;
18 07/01/2023;16:10:41;24.60;73.00;
19 07/01/2023;16:11:11;24.60;73.00;
20 07/01/2023;16:11:41;24.60;73.00;
```

Com os arquivos .csv prontos, começamos a etapa de armazenamento. Nessa etapa utilizamos o MongoDB para a persistências dos dados das coletas. Foi seguido

um tutorial disponibilizado pelos orientadores no canvas para criação do cluster através do mongoDB Atlas. A figura a seguir mostra o cluster criado juntamente com a database e as collections:

Figura 28 – Cluster



Após toda configuração ser feita de acordo com o tutorial disponibilizado, avançamos para o desenvolvimento do código para assim fazermos a persistência na base de dados. Por facilidade e flexibilidade, foi utilizado o Google Colab para o desenvolvimento do código. Utilizamos bibliotecas tais como pandas, json e pymongo. Para leitura do csv foi utilizado o pandas, em seguida foi feita uma conversão dos dados para o formato json. Essa conversão foi feita através da biblioteca json, o que foi necessário pois é um formato válido pelo mongodb. Logo em seguida foi feita a persistência na collection correspondente. A figura a seguir mostra parte do código utilizado para persistências dos dados:

Figura 29 – Código em python para conexão com cluster

```
!python -m pip install pymongo
!pip install dnspython

▼ Importando bibliotecas

[ ] import pymongo
    from pymongo import MongoClient
    import pandas as pd
    import json
    import matplotlib.pyplot as plt

▼ Definindo a conexão com banco de dados

[ ] cluster = pymongo.MongoClient("mongodb+srv://db_user:vf8xq0lyz06c8d@cluster0.spt7r2s.mongodb.net/db_user?retryWrites=true&w=majority")

▼ Setando a Database do projeto

[ ] db = cluster.get_database('projeto_ifb')

▼ Descrição dos cenários coletado

Coleta 1 - coleta da temperatura e umidade fora da caixa
Coleta 2 - coleta da temperatura e umidade dentro da caixa fechada contendo um desumidificador e um notebook emitindo calor
Coleta 3 - coleta da temperatura e umidade dentro da caixa contendo apenas o desumidificador
Coleta 4 - coleta da temperatura e umidade dentro da caixa fechada contendo o desumidificador e 3 lanternas
Coleta 5 - coleta da temperatura e umidade dentro da caixa fechada usando um secador a cada 5 minutos
```

Figura 30 – Código em python leitura dos arquivos csv

```
Projeto-IFB.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

+ Código + Texto

▼ Lendo os arquivos csv

[ ] coleta_1 = pd.read_csv('coleta1.csv', delimiter=';')
    coleta_2 = pd.read_csv('coleta2.csv', delimiter=';')
    coleta_3 = pd.read_csv('coleta3.csv', delimiter=';')
    coleta_4 = pd.read_csv('coleta4.csv', delimiter=';')
    coleta_5 = pd.read_csv('coleta5.csv', delimiter=';')

coleta_1

data hora temperatura umidade
0 21/12/2022 19:32:18 23.4 81
1 21/12/2022 19:32:48 23.4 81
2 21/12/2022 19:33:18 23.4 81
3 21/12/2022 19:33:48 23.4 81
4 21/12/2022 19:34:18 23.4 81
... ..
56 21/12/2022 20:00:23 23.5 81
57 21/12/2022 20:00:53 23.5 81
58 21/12/2022 20:01:23 23.5 81
59 21/12/2022 20:01:53 23.4 81
60 21/12/2022 20:02:23 23.4 81
61 rows x 4 columns

[ ] coleta_2
[ ] coleta_3
[ ] coleta_4
```

Figura 31 – Código em python conversão dos dados para o formato json e persistências dos mesmos

```
Projeto-IFB.ipynb
Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda Todas as alterações foram salvas

+ Código + Texto

▼ Transformando os dados em formato JSON para persistência no MongoDB

[ ] dados_1 = coleta_1.to_dict(orient = "records")
    dados_2 = coleta_2.to_dict(orient = "records")
    dados_3 = coleta_3.to_dict(orient = "records")
    dados_4 = coleta_4.to_dict(orient = "records")
    dados_5 = coleta_5.to_dict(orient = "records")

[ ] dados_1
[ ] dados_2
[ ] dados_3
[ ] dados_4
[ ] dados_5

▼ Setando as collections e persistindo os dados

[ ] collection_1 = db.get_collection('coleta_dados_1')
    if(list(collection_1.find())):
        print('já existe dados na collection_1')
    else:
        collection_1.insert_many(dados_1)

[ ] collection_2 = db.get_collection('coleta_dados_2')
    if(list(collection_2.find())):
        print('já existe dados na collection_2')
    else:
        collection_2.insert_many(dados_2)

[ ] collection_3 = db.get_collection('coleta_dados_3')
    if(list(collection_3.find())):
```

Todos os códigos utilizados podem ser encontrados no Github através do link:  
<https://github.com/infocbra/pratica-integrada-cd-e-ic-2022-2-g2-cmr>

## 4. Considerações finais

Para a realização da sprint 1, a principal dificuldade encontrada foi a de conseguir os materiais necessários para a realização do projeto. Fizemos um pedido pela internet e a entrega chegou poucos dias antes do fechamento da sprint, o que nos deu pouco tempo para a montagem do hardware e desenvolvimento do código. Apesar das dificuldades encontradas, ficamos satisfeitos por atingir o objetivo dessa sprint de montar o hardware e configurá-lo para funcionar adequadamente.

Na sprint 2, a principal dificuldade foi em montar a caixa utilizada para o controle de umidade e coletar os dados dentro dela. Devido ao fato da caixa precisar ser fechada para coletar os dados, a solução encontrada foi a de colocar o sensor e a fonte de energia (nesse caso um notebook) dentro da caixa e fechá-la. Nesse caso, o calor emitido pelo notebook contribuiu para o aumento da temperatura e diminuição da umidade. Para a próxima sprint, os objetivos para a melhoria do projeto é melhorar o desempenho da caixa para ela reduzir ainda mais a umidade e conseguir colocar apenas o sensor dentro da caixa e a fonte de energia fora, para não interferir na temperatura.

Para a sprint 3, conseguimos resolver o problema da escrita do arquivo .csv e fizemos mais algumas coletas de dados. Salvamos os arquivos no banco de dados e escrevemos no relatório a análise dos gráficos gerados, concluindo assim as requisições dessa sprint.



## Referências

MURTA, José Gustavo Abreu. Sensores DHT11 e DHT22: Guia Básico dos Sensores de Umidade e Temperatura. **Blog Eletrogate**, 2019. Disponível em: <https://blog.eletrogate.com/sensores-dht11-dht22/>. Acesso em: 10 dez.2022.

Umidade no Filamento. **Tecnocubo**, c2022. Disponível em: <https://www.tecnocubo.com.br/pagina/umidade-no-filamento.html>. Acesso em: 10 dez.2022.

Como resolver umidade no filamento. **3DLab**, 2016. Disponível em: <https://3dlab.com.br/umidade-no-filamento/>. Acesso em: 22 dez.2022.