Desenvolvimento de Experimento de Termodinâmica com Arduino

Este projeto visa medir a relação entre pressão e temperatura em um recipiente isolado (compressão adiabática) utilizando sensores controlados por Arduino.

Prof: Felipe Mondaini

Integrantes: Adson Torino

Emanuel Cabral

Gabriel Santos

Gustavo Kanji

Nícolas Henriques

Vicente Zanatta

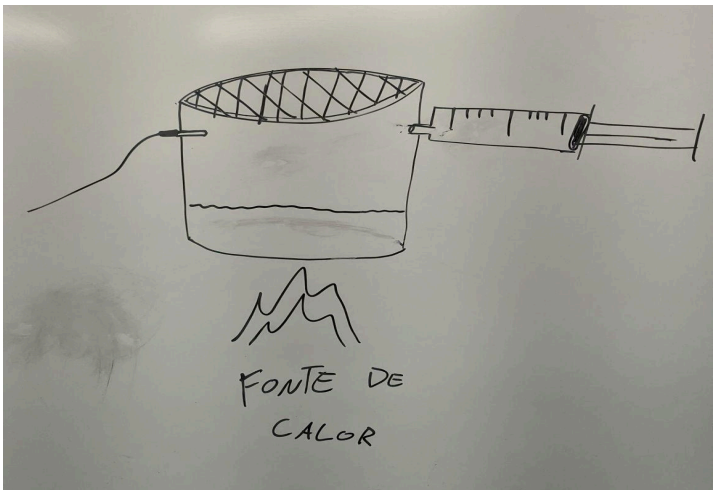
Diário de desenvolvimento

09/08

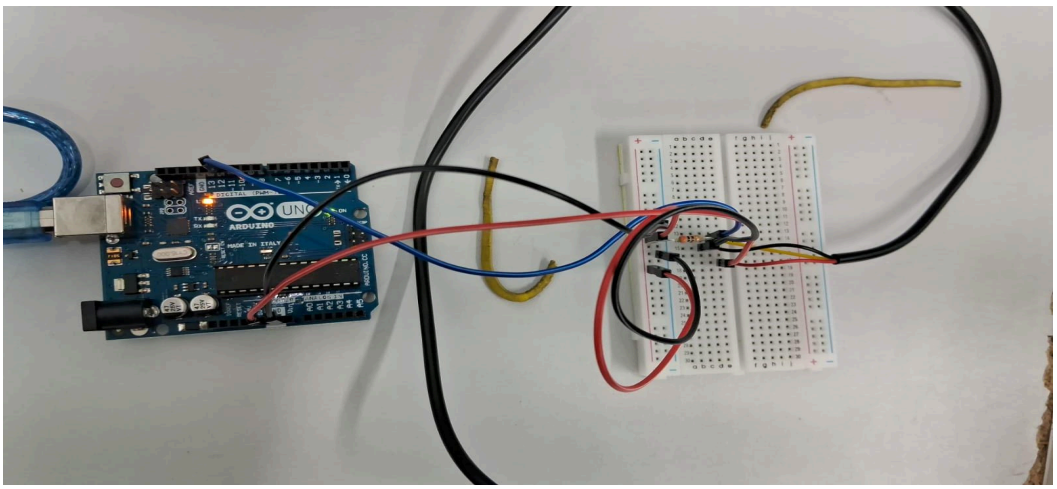
- Grupo criado;

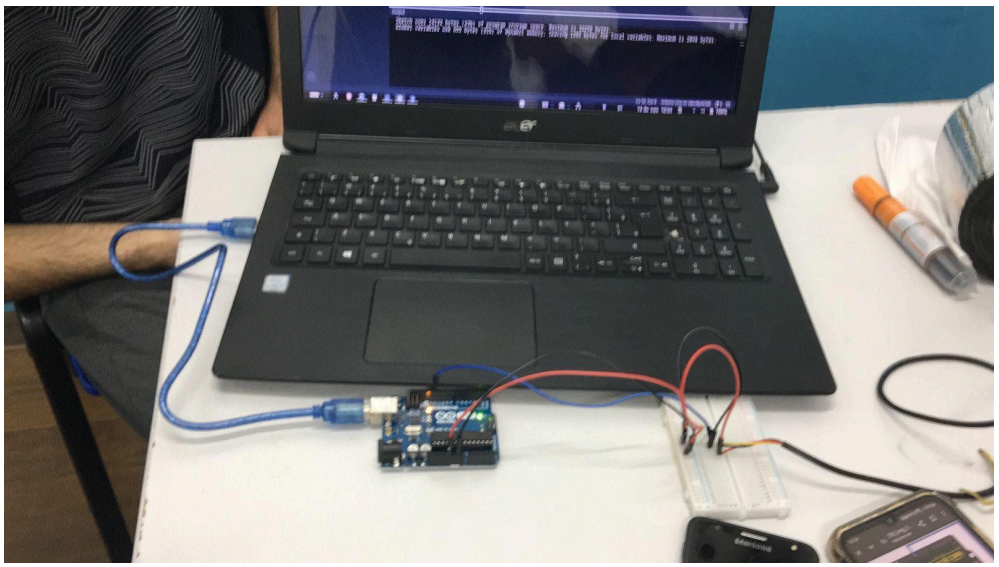
14/08

- Idealização do experimento;



- Reunião para produção do recipiente onde o experimento será realizado;
- Código começado;
- Bibliotecas para o Arduino instaladas;
- Arduino montado com o sensor de temperatura (sensor: DS18B20);

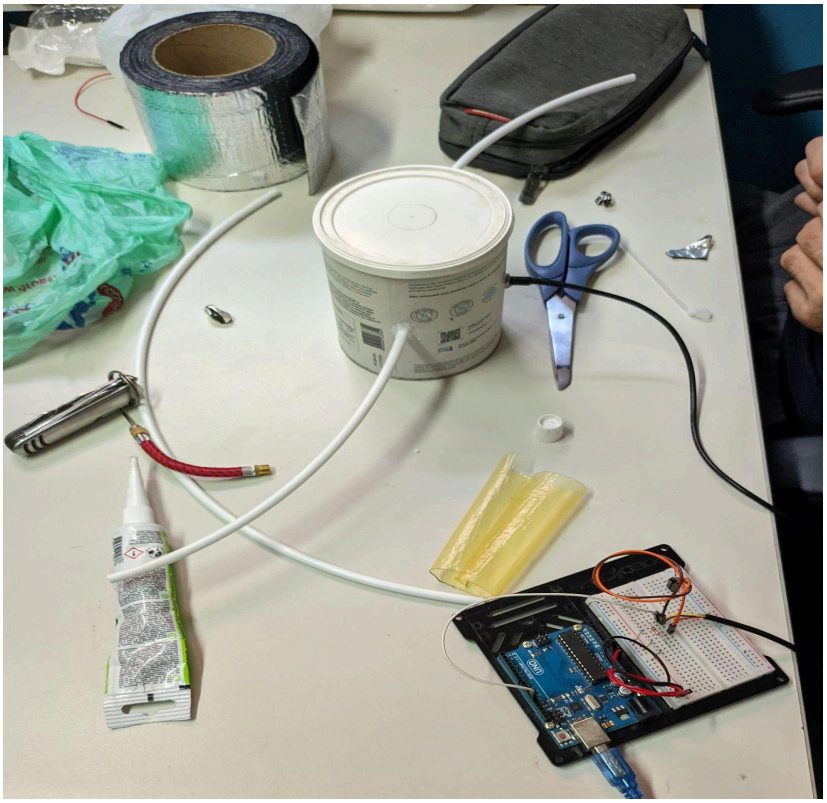




- Recipiente 1 furado para o sensor de temperatura.

15/08

- Criação do recipiente 1;

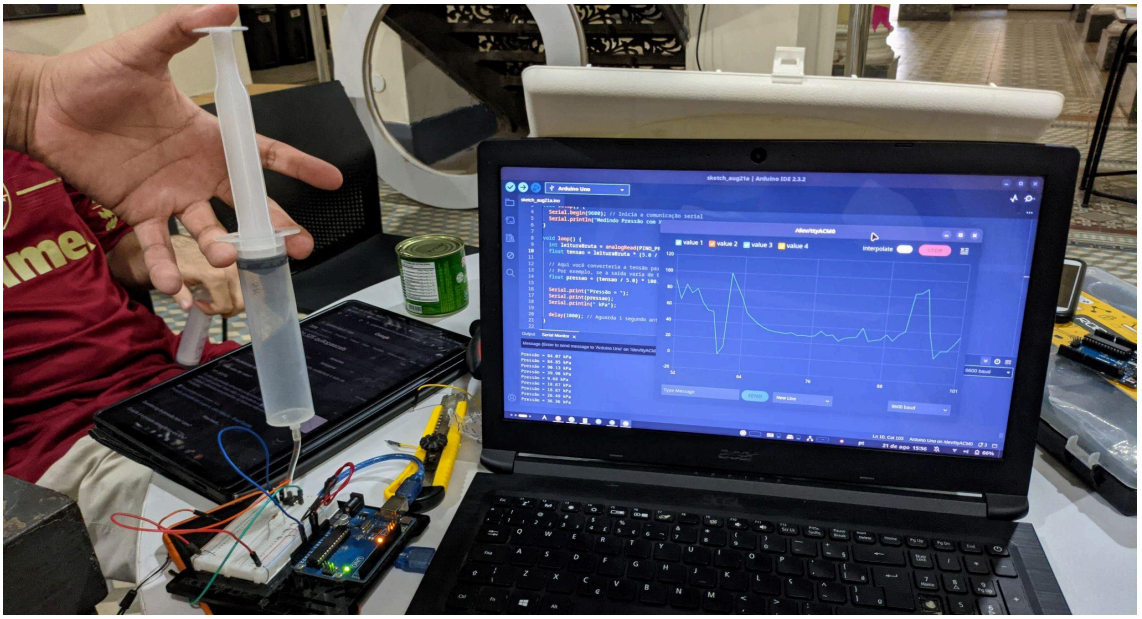


- Cantos do recipiente 1 furados para a inserção das seringas;
- Recipiente 1 vedado;
- Primeiro teste: Falha (as seringas se soltaram);

- Segundo teste: Falha (a cola derreteu devido a alta temperatura).

21/08

- Discussão de um novo recipiente;
- Troca de recipiente: o professor Saraiva sugeriu uma lata de azeite;
- Lata de ervilha escolhida como o novo recipiente;
- Lata furada para as seringas e os sensores;
- Calibragem do sensor de pressão;



- Ajuste do código;
- Recipiente 1 descartado.

22/08

- Parte de cima da lata soldada para vedação;

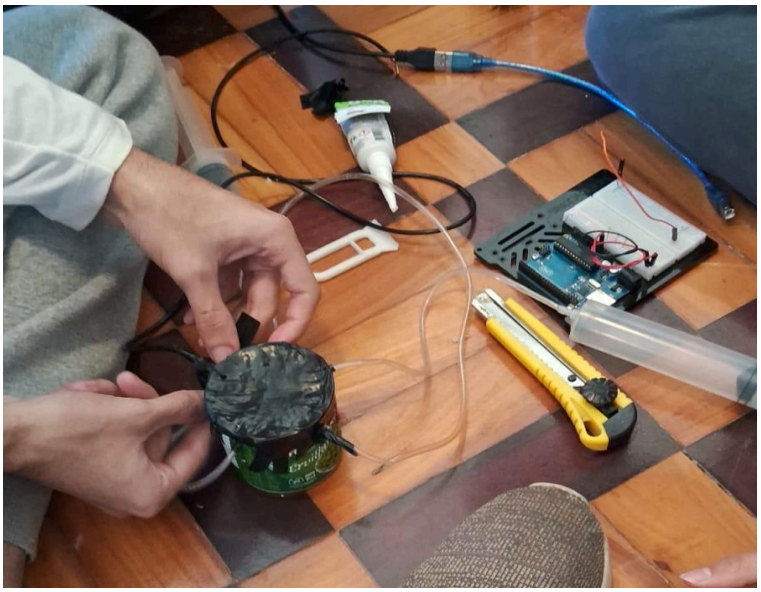
- Furos para as seringas e sensores vedados com silicone e fita isolante;

26/08

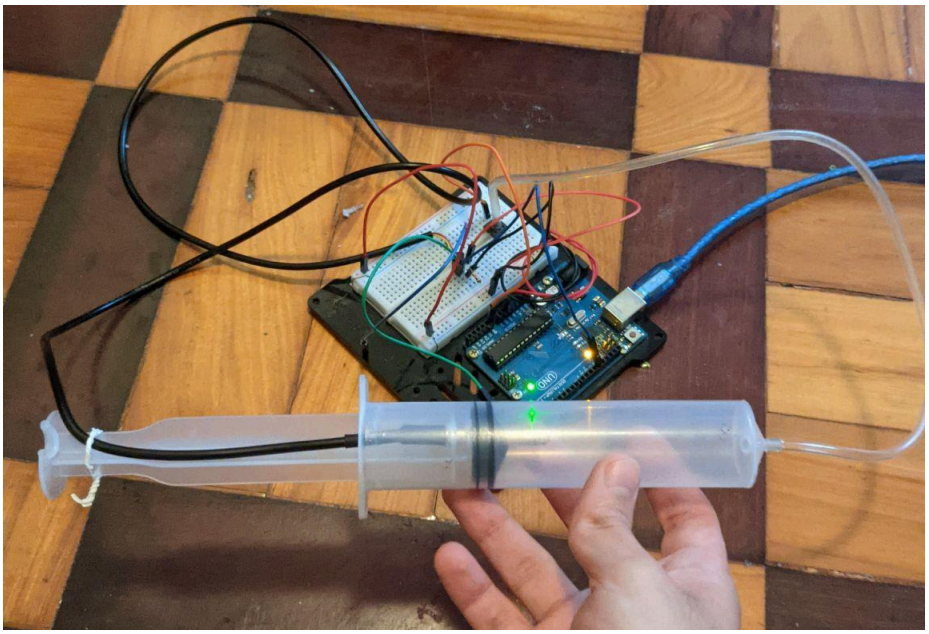
- Teste de pressão no recipiente 2: Falha (a solda não aguentou e vazou);
- Código para o sensor de pressão e de temperatura mesclados;

28/08

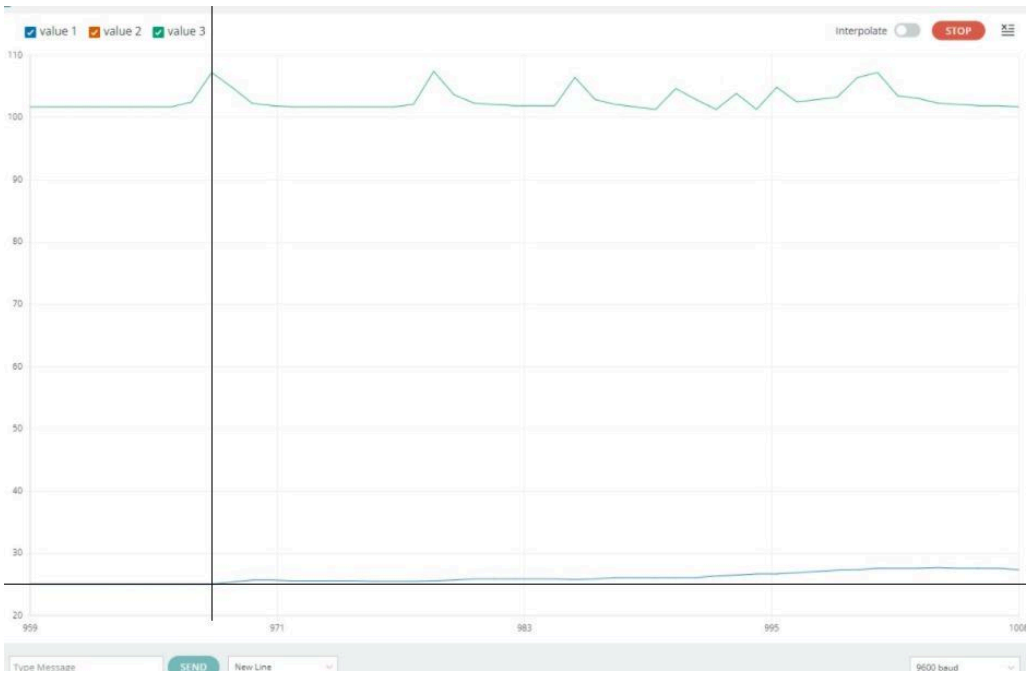
- Reforço do recipiente 2: Topo e todas os furos vedados com silicone e fita isolante;



- Teste de pressão no recipiente 2: Falha (a solda se descolou completamente);
- Código ajustado para representar as curvas de pressão e temperatura no gráfico;
- Recipiente 2 descartado;
- Novo recipiente escolhido: A própria seringa;
- Recipiente 3 furado para o sensor de temperatura;



- Teste no recipiente 3: Sucesso (variação de temperatura detectada em função do aumento da pressão);



Código (C++)

```
// Inclusão das bibliotecas
#include <OneWire.h>
#include <DallasTemperature.h>

// Configuração do sensor de temperatura DS18B20
const int PINO_ONEWIRE = 12; // Define o pino do sensor de temperatura
OneWire oneWire(PINO_ONEWIRE); // Cria um objeto OneWire
DallasTemperature sensor(&oneWire); // Informa a referência da biblioteca DallasTemperature para a biblioteca OneWire
DeviceAddress endereco_temp; // Cria um endereço temporário para a leitura do sensor

// Configuração do sensor de pressão XGZP101DB1R
const int PINO_PRESSAO = A0; // Pino analógico conectado ao VO+
float ant = 0; // Variável para armazenar a pressão anterior

void setup() {
  // Inicia a comunicação serial
   Serial.begin(9600); //Inicia a comunicação serial com uma taxa de transmissão de 9600 bps (bits por segundo), permitindo enviar e receber dados entre o Arduino e o computador
  Serial.println("Medindo Temperatura e Pressão"); // Mensagem inicial

  // Inicia o sensor de temperatura
  sensor.begin();
}

void loop() {
  // Leitura da temperatura
  sensor.requestTemperatures(); // Envia comando para realizar a conversão de temperatura
  float temperatura = 0.0;
   if (!sensor.getAddress(endereco_temp, 0)) { // Tenta obter o endereço no barramento do sensor de temperatura
        Serial.println("SENSOR DE TEMPERATURA NÃO CONECTADO"); // Sensor não conectado, imprime mensagem de erro
  } else {
```

```
        temperatura = sensor.getTempC(endereco_temp); // Busca temperatura para o dispositivo
  }

  // Leitura da pressão
   int leituraBruta = analogRead(PINO_PRESSAO); // Lê o valor analógico do sensor de pressão, o valor retornado será entre 0 e 1023, correspondente a uma tensão entre 0 e 5V
   float tensao = leituraBruta * (5.0 / 1023.0); // Converte o valor lido do ADC (Conversor Analógico-Digital) em uma tensão real (em volts)

  // Conversão da tensão para pressão
  float pressao = (tensao / 5.0) * 201.62;

  // Envia os valores para o Serial Plotter
  Serial.print(temperatura, 1); // Imprime a temperatura com 1 casa decimal
  Serial.print(" - "); // Separador entre temperatura e pressão
  Serial.println(pressao); // Imprime a pressão

  // Atualiza a pressão anterior
  ant = pressao;

  // Aguarda 1 segundo antes de ler novamente
  delay(1000);
}
```

- Explicação do código:

Este código faz a leitura de temperatura e pressão utilizando um Arduino, o sensor de temperatura DS18B20 e o sensor de pressão XGZP101DB1R. Ele é dividido em duas partes principais: configuração e loop principal.

1. Inclusão das Bibliotecas
- A biblioteca `OneWire.h` permite a comunicação com dispositivos que utilizam o protocolo OneWire, como o sensor DS18B20.
- A biblioteca `DallasTemperature.h` facilita o uso de sensores de temperatura DS18B20, que usam o protocolo OneWire.

2. Configuração do Sensor de Temperatura DS18B20
- Define o pino onde o sensor DS18B20 está conectado (pino 12).
- Um objeto `OneWire` é criado para comunicação com o sensor no pino 12.
- Um objeto `DallasTemperature` é criado e associado ao objeto `OneWire`.
- `DeviceAddress endereco_temp` cria uma variável que armazenará o endereço do sensor de temperatura no barramento OneWire.

3. Configuração do Sensor de Pressão XGZP101DB1R
- Define o pino analógico `A0` para leitura da tensão de saída do sensor de pressão.
- A variável `ant` é inicializada para armazenar o valor da pressão lida anteriormente, embora não seja usada para cálculos subsequentes.

4. Função `setup()`
- Inicia a comunicação serial a 9600 bps para enviar dados ao monitor serial do Arduino.
- Exibe a mensagem "Medindo Temperatura e Pressão" no monitor serial.
- Inicializa o sensor de temperatura DS18B20 com `sensor.begin()`.

5. Função `loop()`
- Leitura da Temperatura: `sensor.requestTemperatures()` solicita ao sensor que faça a leitura da temperatura. Verifica se o endereço do sensor de temperatura foi encontrado no barramento OneWire. Se o sensor estiver conectado, a temperatura é lida em graus Celsius e armazenada na variável `temperatura`. Se o sensor não estiver conectado, uma mensagem de erro é exibida.

- Leitura da Pressão: `analogRead(PINO_PRESSAO)` lê o valor analógico do sensor de pressão, que é uma leitura bruta (entre 0 e 1023). O valor bruto é convertido em tensão (volts), assumindo uma referência de 5V. A tensão é então convertida em pressão usando uma fórmula específica que presume uma faixa de pressão de 0 a 100 kPa, com uma saída de tensão proporcional de 0 a 5V.

- Envio dos Valores para o Monitor Serial: A temperatura (com uma casa decimal) e a pressão são enviadas para o monitor serial, separadas por uma vírgula, facilitando a visualização em forma de gráfico.

- Delay: Aguarda 1 segundo antes de fazer uma nova leitura, repetindo o ciclo.

Resumindo, este código faz a leitura contínua da temperatura e pressão e envia os valores para o monitor serial do Arduino, onde podem ser visualizados ou plotados como gráficos. Ele utiliza bibliotecas especializadas para facilitar a comunicação com os sensores conectados ao Arduino.

- Nota

No Arduino, o código é organizado de maneira diferente em comparação com programas escritos em linguagens como C ou C++. Em vez de uma função main(), o Arduino usa duas funções principais: setup() e loop().

setup(): Esta função é chamada uma vez quando o Arduino é ligado ou reiniciado. É usada para configurar as definições iniciais, como configurar pinos, iniciar a comunicação serial, inicializar sensores, etc.

loop(): Após a conclusão da função setup(), o Arduino entra na função loop(), que é executada repetidamente até que o dispositivo seja desligado. Essa função é onde o código principal do seu programa fica, ou seja, onde você coloca o código que deve ser executado continuamente.

O Arduino abstrai a função main() tradicional, escondendo-a do usuário para simplificar o desenvolvimento. No fundo, existe uma função main() no código gerado pelo compilador do Arduino, que cuida da inicialização do hardware e, em seguida, chama setup() e loop().

Teoria

- Lei dos Gases Ideais

A Lei dos Gases Ideais é uma das relações fundamentais na termodinâmica e pode ser expressa pela equação PV = nRT, onde P é a pressão do gás, V é o volume do gás, n é o número de moles do gás, R é a constante universal dos gases ideais (aproximadamente 8,314 J/mol·K), e T é a temperatura absoluta do gás (em Kelvin). Esta equação descreve o comportamento de gases ideais, que são teóricos e seguem essa relação em todas as condições de temperatura e pressão. No contexto do experimento, essa lei ajuda a entender como a pressão e a temperatura de um gás estão inter-relacionadas em um sistema fechado, como o recipiente utilizado.

- Processo Adiabático

Um processo adiabático é aquele no qual não há troca de calor entre o sistema e o ambiente. Isso significa que todo o trabalho feito sobre o gás (ou por ele) resulta em uma mudança em sua energia interna, o que afeta sua temperatura e pressão. Para um gás ideal, a relação entre a pressão e a temperatura durante um processo adiabático pode ser expressa como P1 V1^γ = P2 V2^γ e T1 V1^(γ-1) = T2 V2^(γ-1), onde γ (gama) é o índice adiabático ou razão de capacidades caloríficas (γ = Cp/Cv), e P1, V1, T1 e P2, V2, T2 são as pressões, volumes e temperaturas iniciais e finais, respectivamente. No experimento, ao comprimir ou expandir o gás dentro do recipiente sem permitir que o calor escape, observa-se o aumento ou diminuição da temperatura devido ao processo adiabático.

- Primeira Lei da Termodinâmica

A Primeira Lei da Termodinâmica afirma que a energia não pode ser criada nem destruída, apenas transformada. Em termos de um sistema fechado, isso pode ser expresso como ΔU = Q - W, onde ΔU é a variação da energia interna do sistema, Q é o calor adicionado ao sistema, e W é o trabalho realizado pelo sistema. Para um processo adiabático, Q = 0, então ΔU = -W. Isso significa que o trabalho realizado sobre o gás (por exemplo, comprimindo-o) aumenta sua energia interna, o que eleva sua temperatura.

- Nota

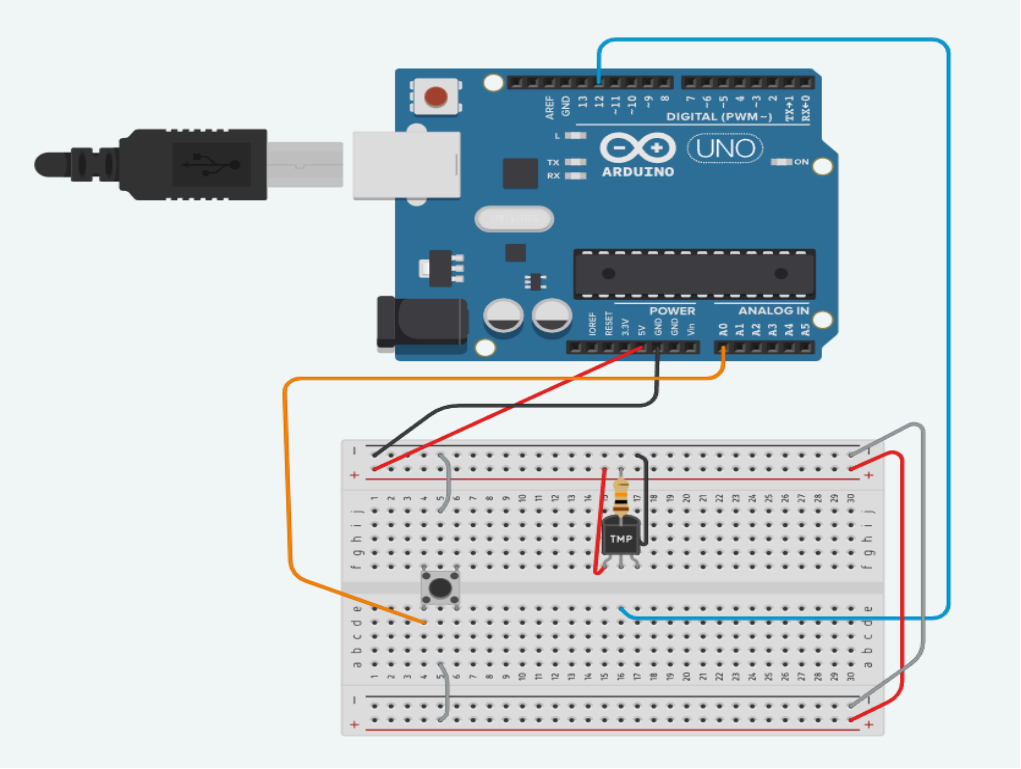
Apesar do nosso experimento não ser ideal, ainda assim é possível ver uma alteração de temperatura em função da pressão.

Arduino

- O Arduino foi utilizado como uma interface para controlar os sensores de pressão e temperatura e capturar os dados necessários para análise. O código desenvolvido para o Arduino faz a leitura dos sinais elétricos gerados pelos sensores, converte esses sinais em valores de pressão e temperatura, e exibe os resultados em um gráfico, permitindo a visualização da relação entre essas duas grandezas durante o experimento.
- Funcionamento do sensor de pressão: O sensor XGZP101DB1R funciona utilizando um princípio chamado piezoresistivo, onde um material de silício no sensor muda sua resistência elétrica em resposta à pressão aplicada. Esse material é montado sobre um substrato e encapsulado para proteger e facilitar sua montagem em

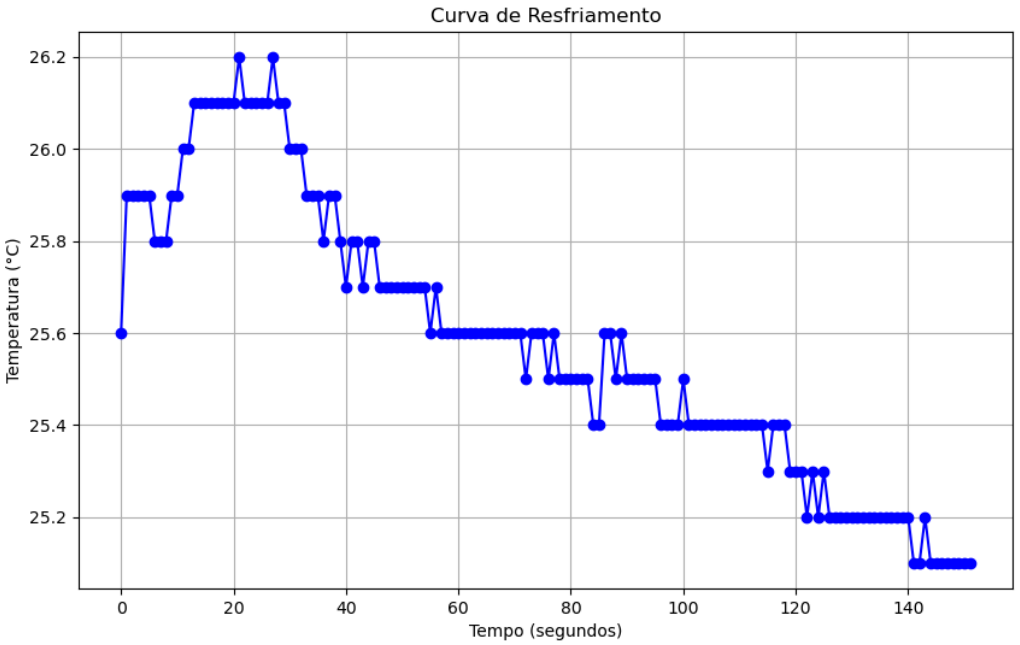
circuitos. Quando a pressão é aplicada ao sensor, ela causa uma deformação no material de silício, alterando sua resistência. Essa mudança é então convertida em um sinal elétrico que pode ser lido por um dispositivo externo.

- Funcionamento do sensor de temperatura: O sensor de temperatura DS18B20 é um dispositivo digital que mede a temperatura e se comunica com microcontroladores via o protocolo 1-Wire. Ele opera em uma faixa de -55°C a +125°C, com precisão de ±0,5°C na faixa de -10°C a +85°C. Sua resolução é ajustável entre 9 e 12 bits, e pode ser alimentado tanto por uma fonte externa quanto pelo modo parasita, usando apenas a linha de dados para comunicação e alimentação. É fácil de usar, suporta múltiplos sensores no mesmo pino de dados e é resistente à umidade, sendo ideal para aplicações em monitoramento ambiental, automação residencial e processos industriais.
- Montagem do arduino:



Curva de Resfriamento:

- Fazendo-se uma leitura utilizando o arduino, extraímos a variação de temperatura ao longo de 2 minutos e com um script em python é possível ver o decaimento na temperatura:



Script utilizado:

import matplotlib.pyplot as plt

 # Dados de temperatura

temperaturas = [

25.6, 25.9, 25.9, 25.9, 25.9, 25.9, 25.8, 25.8, 25.8, 25.9, 25.9, 26.0, 26.0, 26.1, 26.1, 26.1, 26.1, 26.1, 26.1, 26.1, 26.2, 26.1, 26.1, 26.1, 26.1, 26.1, 26.2, 26.1, 26.1, 26.0, 26.0, 26.0, 25.9, 25.9, 25.9, 25.8, 25.9, 25.9, 25.8, 25.7, 25.8, 25.8, 25.7, 25.8, 25.8, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.6, 25.7, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.5, 25.6, 25.6, 25.6, 25.5, 25.6, 25.5, 25.5, 25.5, 25.5, 25.5, 25.5, 25.4, 25.4, 25.6, 25.6, 25.5, 25.6, 25.5, 25.5, 25.5, 25.5, 25.5, 25.5, 25.4, 25.4, 25.4, 25.4, 25.5, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.3, 25.4, 25.4, 25.4, 25.3, 25.3, 25.3, 25.2, 25.3, 25.2, 25.3, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.1, 25.1, 25.2, 25.1, 25.1, 25.1, 25.1, 25.1, 25.1, 25.1, 25.1

]

 # Tempo (em segundos) ajustado para corresponder aos 152 valores de temperatura

tempo_ajustado = list(range(152))

 # Criação da curva de resfriamento

plt.figure(figsize=(10, 6))

plt.plot(tempo_ajustado, temperaturas, marker='o', linestyle='-', color='blue')

plt.title('Curva de Resfriamento')

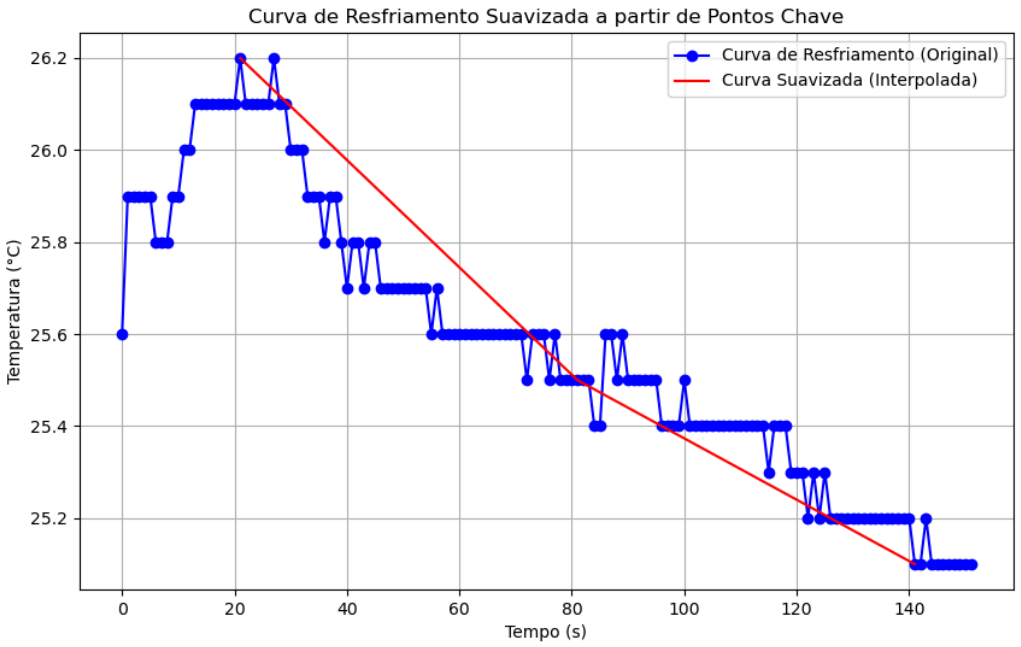
plt.xlabel('Tempo (segundos)')

plt.ylabel('Temperatura (°C)')

plt.grid(True)

plt.show()

- Interpolando-se os pontos máximo, médio e mínimo obtém-se o seguinte gráfico:



Script utilizado:

import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import interp1d

 # Dados de temperatura

temperaturas = [

25.6, 25.9, 25.9, 25.9, 25.9, 25.9, 25.8, 25.8, 25.8, 25.9, 25.9, 26.0, 26.0, 26.1, 26.1, 26.1, 26.1, 26.1, 26.1, 26.1, 26.1, 26.2, 26.1, 26.1, 26.1, 26.1, 26.1, 26.2, 26.1, 26.1, 26.0, 26.0, 26.0, 25.9, 25.9, 25.9, 25.8, 25.9, 25.9, 25.8, 25.7, 25.8, 25.8, 25.7, 25.8, 25.8, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.7, 25.6, 25.7, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6,

```
25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.6, 25.5, 25.6, 25.6, 25.6, 25.5, 25.6, 25.5, 25.5,
25.5, 25.5, 25.5, 25.5, 25.4, 25.4, 25.6, 25.6, 25.5, 25.6, 25.5, 25.5, 25.5, 25.5, 25.5, 25.5,
25.4, 25.4, 25.4, 25.4, 25.5, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4, 25.4,
25.4, 25.4, 25.4, 25.3, 25.4, 25.4, 25.4, 25.3, 25.3, 25.3, 25.2, 25.3, 25.2, 25.3, 25.2, 25.2,
25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.2, 25.1, 25.1, 25.2,
25.1, 25.1, 25.1, 25.1, 25.1, 25.1, 25.1, 25.1
]

# Tempo (em segundos)
tempo_ajustado = np.array(range(len(temperaturas)))

# Selecionar os pontos chave (máximo, mínimo, e alguns intermediários)
indice_max = np.argmax(temperaturas)
indice_min = np.argmin(temperaturas)

# Pontos intermediários escolhidos manualmente
indice_intermediario = int((indice_max + indice_min) / 2)

# Corrigir a concatenação do índice intermediário
indices_chave = np.sort(np.array([indice_max, indice_min, indice_intermediario]))

# Obter os tempos e temperaturas correspondentes aos pontos chave
tempos_chave = tempo_ajustado[indices_chave]
temperaturas_chave = np.array(temperaturas)[indices_chave]

# Interpolação linear entre os pontos chave
tempo_interpolado = np.linspace(tempos_chave[0], tempos_chave[-1], 500)
interp_func = interp1d(tempos_chave, temperaturas_chave, kind='linear')
temperaturas_interpoladas = interp_func(tempo_interpolado)

# Criação da curva de resfriamento interpolada
plt.figure(figsize=(10, 6))
plt.plot(tempo_ajustado, temperaturas, 'o', linestyle='-', color='blue', label='Curva de
Resfriamento (Original)')
plt.plot(tempo_interpolado, temperaturas_interpoladas, '-', color='red', label='Curva
Suavizada (Interpolada)')
plt.title('Curva de Resfriamento Suavizada a partir de Pontos Chave')
```

```
plt.xlabel('Tempo (s)')
plt.ylabel('Temperatura (°C)')
plt.legend()
plt.grid(True)
plt.show()
```

Conclusão

O desenvolvimento do experimento de termodinâmica utilizando o Arduino foi uma oportunidade valiosa para aplicar e testar conceitos em um ambiente prático. Através da construção e ajuste iterativo de diferentes recipientes, foi possível observar diretamente a relação entre pressão e temperatura em um sistema fechado.

As dificuldades encontradas, como a falha na vedação dos primeiros recipientes, foram cruciais para o aprendizado. Esses desafios ressaltaram a importância de aspectos práticos, como a escolha adequada de materiais e a importância de uma vedação eficiente para garantir a precisão dos resultados. A adaptação para o uso de uma seringa como recipiente se mostrou uma solução prática e eficaz, permitindo que o experimento prosseguisse com sucesso e que as medições fossem realizadas de forma confiável.

Além disso, o processo envolveu um aprendizado significativo em relação ao Arduino, especialmente nas conexões físicas dos sensores e na programação do código em C++. O uso do Arduino e a integração dos sensores de temperatura e pressão foram essenciais para a coleta precisa dos dados. A análise desses dados, através de scripts em Python, permitiu uma visualização clara das variações de temperatura ao longo do tempo, confirmando as previsões teóricas.

Este trabalho não apenas ilustra a aplicação prática em experimentos científicos, mas também evidencia a importância da experimentação e da iteração na resolução de problemas práticos. As falhas iniciais e as soluções encontradas enriquecem a experiência, mostrando que a experimentação científica é um processo dinâmico que envolve adaptação, aprendizado contínuo e o domínio de ferramentas tecnológicas como o Arduino.