**Simulação de Monitoramento de Vulcão com ESP32 e Python – GS FIAP 2025**

Global Solution 2025

Curso: Inteligência Artificial – FIAP

Tema: Desastres Naturais – Monitoramento de Vulcões

Grupo: Felipe Menezes (RM557891), Flavia Bocchino (RM564213), Pedro Zani (RM564956)

Data: Junho de 2025 – São Paulo, Brasil

Github: https://github.com/flahbocchino/global-solution-vulcao-2025/tree/main

**Introdução**

Erupções vulcânicas são fenômenos naturais com grande potencial destrutivo. Um exemplo marcante foi a erupção do vulcão **Eyjafjallajökull**, na Islândia, em 2010, que causou o fechamento de diversos aeroportos na Europa por conta da fumaça e das cinzas expelidas no ar. Esse tipo de evento pode representar riscos à saúde humana, impactos ambientais e prejuízos econômicos.

A grande pergunta é: **como saber se um vulcão vai entrar em erupção?**

Para responder a isso, cientistas usam instrumentos que monitoram o solo e o ar. Eles medem parâmetros como **temperatura**, **pressão atmosférica**, **emissão de gases**, entre outros. Quando algum desses dados se afasta do normal, é sinal de que o vulcão pode estar prestes a entrar em atividade.

Pensando nisso, nosso grupo propôs o desenvolvimento de um **sistema inteligente de monitoramento vulcânico**, que simula, de forma simplificada, como esse processo funciona.

**Sumário das Etapas**

1. **Etapa 1: Definir a Lógica da Solução (baseada em sensores)**
2. **Etapa 2: ESP32 + Sensor (simulado no Wokwi)**
3. **Etapa 3: Análise em Python (Google Colab)**
4. **Resultados Esperados**
5. **Etapa 4: Organizar o Repositório no GitHub**
6. **Etapa 5: Montar o PDF final**
7. **Etapa 6: Referências (links de apoio e repositórios)**
8. **Etapa Extra: Evidências Visuais (Prints)**
9. **Etapa Complementar: Arquitetura da Solução (**
10. **Etapa Complementar: Circuito e Código ESP32**
11. **Etapa Complementar: Lógica de Análise em Python**
12. **Conclusão**
13. **Link do Youtube**

**Etapa 1 – Lógica da Solução**

Nosso sistema usa um **sensor DHT22** para medir a **temperatura** e a **umidade** do ar próximo ao vulcão. Esses dados são analisados por um microcontrolador ESP32.

Se a **temperatura ultrapassar 45 °C** e a **umidade for menor que 20%**, o sistema interpreta isso como **risco de erupção** e dispara um alerta simulado no monitor serial.

Essa lógica é baseada em uma estrutura condicional simples:

“Se acontecer isso, então faça aquilo.”

**Etapa 2 – Simulação com ESP32 + DHT22**

Utilizamos a plataforma **Wokwi** para simular o funcionamento do circuito, incluindo:

* Conexão entre o sensor DHT22 e a placa ESP32
* Código que coleta dados do ambiente a cada 2 segundos
* Emissão automática de alerta em caso de risco

Essa simulação permite testar o sistema sem precisar de componentes físicos, facilitando o desenvolvimento e a validação da lógica.

**Etapa 3 – Análise de Dados com Python**

Os dados do sensor foram simulados em uma planilha com a linguagem Python, dentro do ambiente **Google Colab**, permitindo:

* Simular diferentes condições de temperatura e umidade
* Aplicar a mesma lógica de risco usada no ESP32
* Gerar **gráficos** que ajudam a visualizar os momentos de alerta
* Exibir mensagens automáticas quando o risco é identificado

Isso facilita o entendimento dos dados e simula o papel de cientistas que analisam esse tipo de informação em tempo real.

**Etapa 4 - Resultados Esperados**

A solução proposta neste projeto tem como principal objetivo simular um sistema de alerta precoce para risco de erupção vulcânica, baseado na coleta e análise de dados ambientais, como temperatura e umidade do ar. A lógica do sistema foi projetada para identificar condições críticas e, a partir disso, emitir alertas automáticos. Espera-se que, ao atingir determinados valores previamente definidos, o sistema reaja de forma autônoma e previsível, reproduzindo o comportamento de um sistema real de monitoramento utilizado por centros de pesquisa e defesa civil.

No ambiente simulado com o microcontrolador ESP32, o sensor DHT22 coleta dados a cada 2 segundos. Esses dados são analisados pelo código embutido na placa, que verifica se a temperatura está acima de 45 °C e a umidade está abaixo de 20%. Quando essas duas condições ocorrem simultaneamente, o sistema imprime um alerta diretamente no monitor serial, simulando uma resposta local diante de uma situação crítica. Isso representa o comportamento esperado de um sistema de segurança em campo, que poderia, por exemplo, acionar um alarme sonoro ou luz de advertência em regiões próximas ao vulcão.

Paralelamente, no ambiente de simulação e análise de dados em Python, usando o Google Colab, foram geradas séries de dados que imitam leituras reais ao longo do tempo. Esses dados foram analisados com a mesma lógica do ESP32, permitindo identificar momentos de risco. Espera-se que o gráfico gerado, com curvas de temperatura e umidade, evidencie visualmente os pontos críticos, sinalizados por faixas de alerta. Além disso, o próprio código é capaz de percorrer os dados simulados e imprimir mensagens automáticas indicando os momentos em que os critérios de risco são atendidos, como por exemplo: “⚠️ ALERTA no tempo 90s – Temp: 46.5°C / Umidade: 17.8%”.

O resultado final esperado é a construção de uma prova de conceito funcional (MVP), que consiga representar o comportamento de um sistema inteligente de alerta baseado em sensores. A simulação combina o uso de hardware (ESP32 e DHT22) com lógica condicional e análise gráfica em Python, mostrando como tecnologias simples e acessíveis podem ser integradas para fins educacionais e, futuramente, adaptadas para aplicações reais de prevenção a desastres naturais.

**Etapa 5– Organização no GitHub**

Criamos um repositório público no **GitHub** para centralizar todos os arquivos e etapas do projeto, incluindo:

* Código do ESP32 (.ino)
* Notebook em Python (.ipynb)
* Gráficos gerados
* Prints das simulações
* Estrutura de pastas bem definida

O repositório permite que qualquer pessoa acesse e entenda a solução proposta.

**Etapa 6 – Montagem do PDF**

Este documento (PDF) foi elaborado reunindo:

* A descrição completa da solução
* Etapas do desenvolvimento
* Explicações técnicas adaptadas a leigos
* Links de apoio e prints ilustrativos

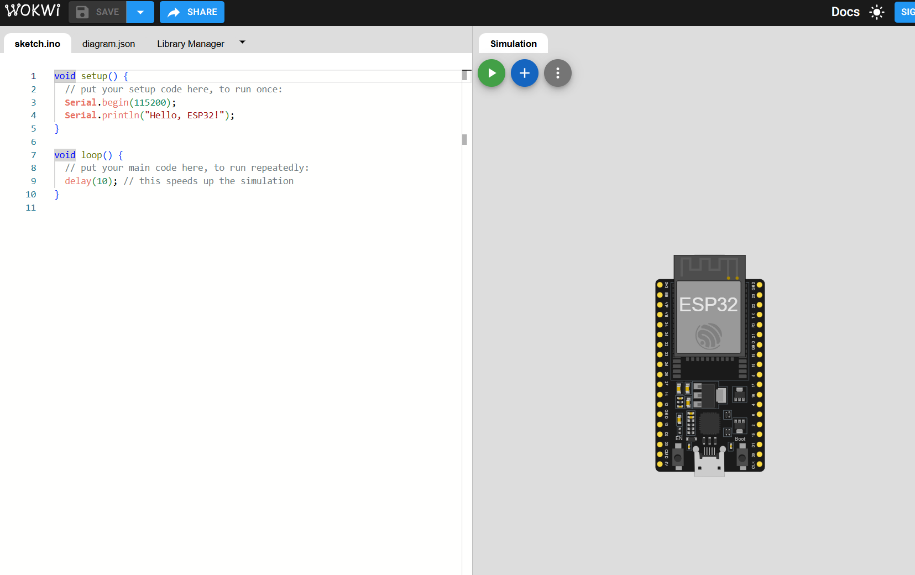
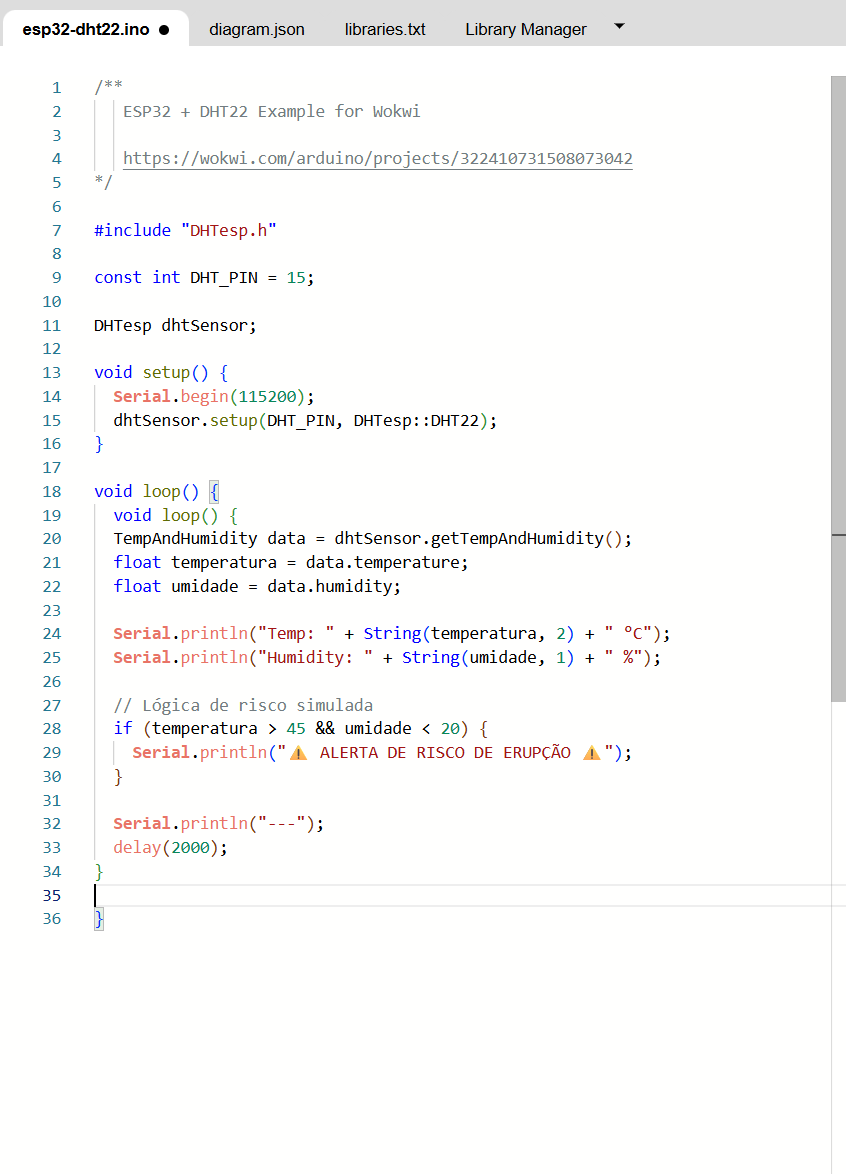
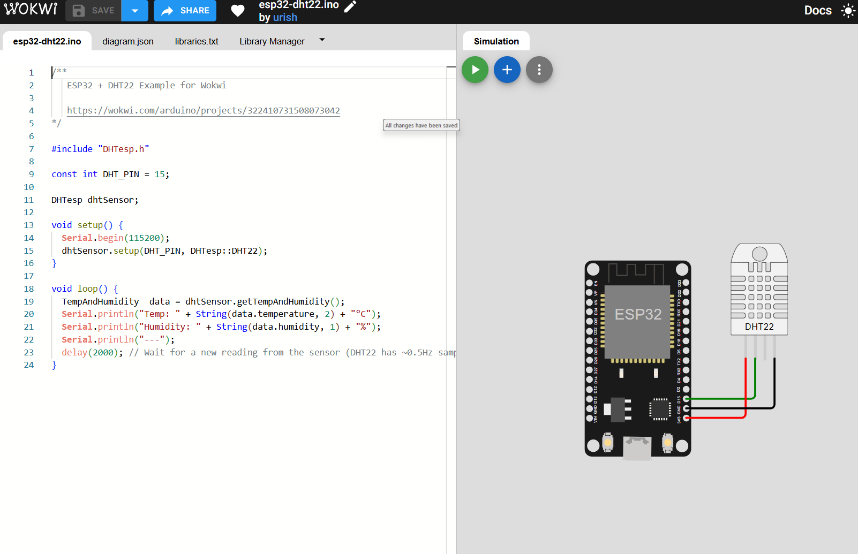
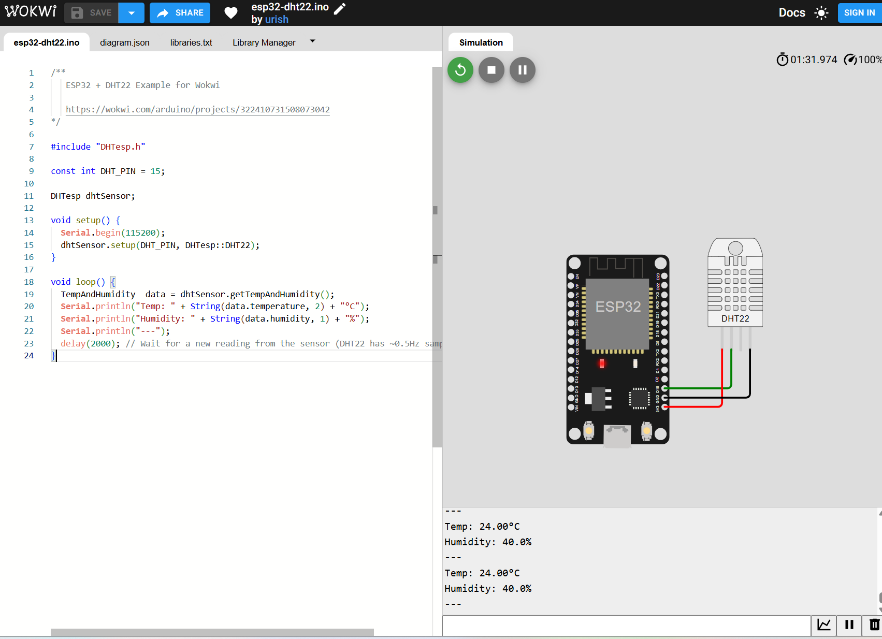
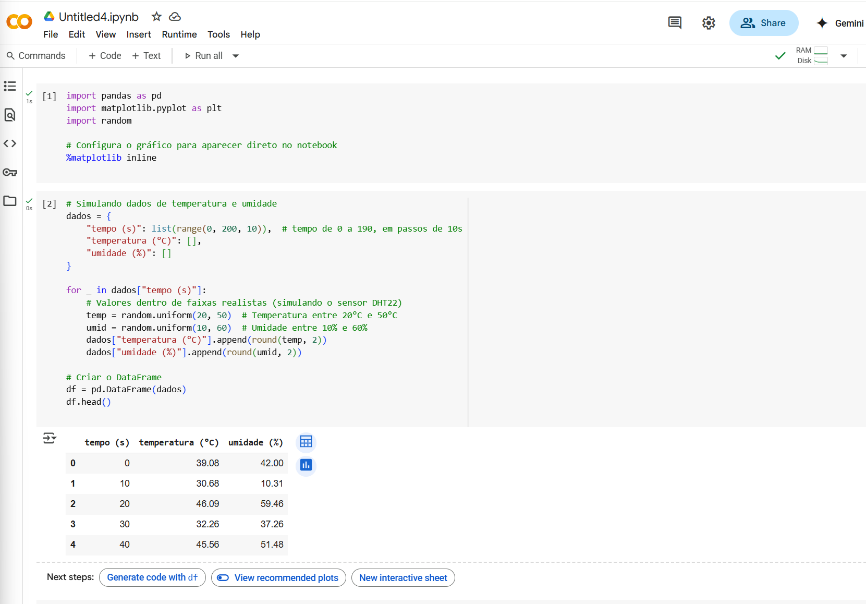
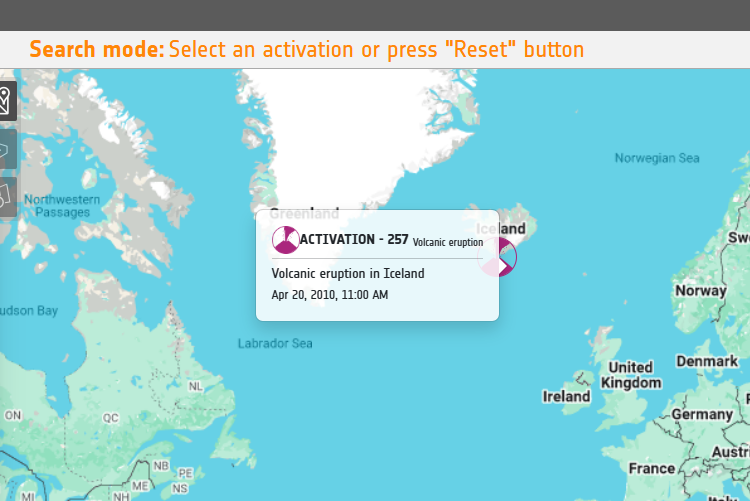
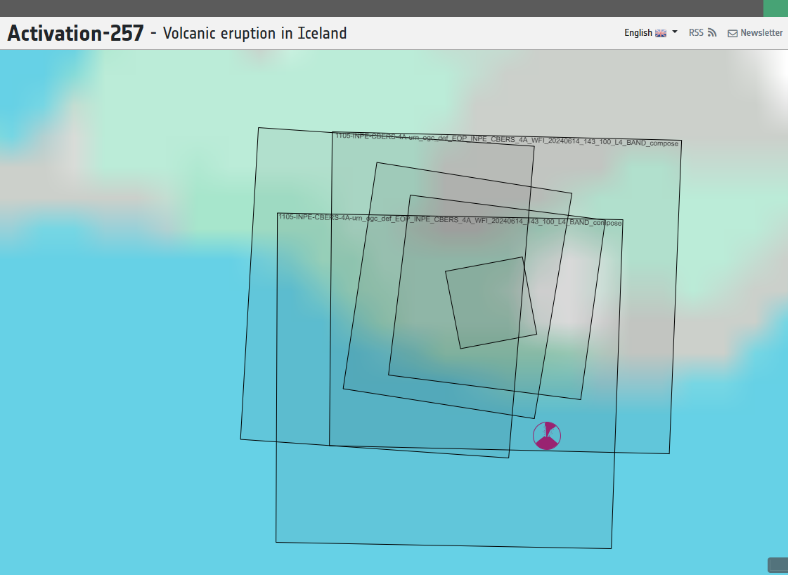
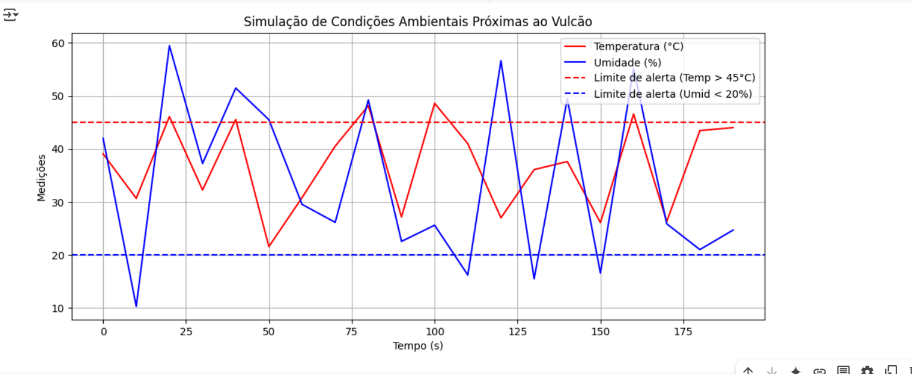
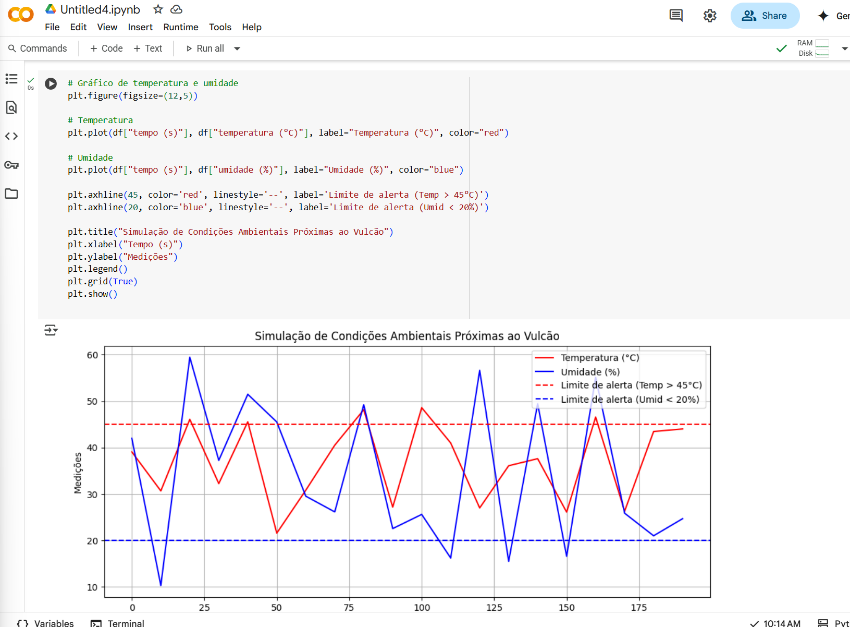
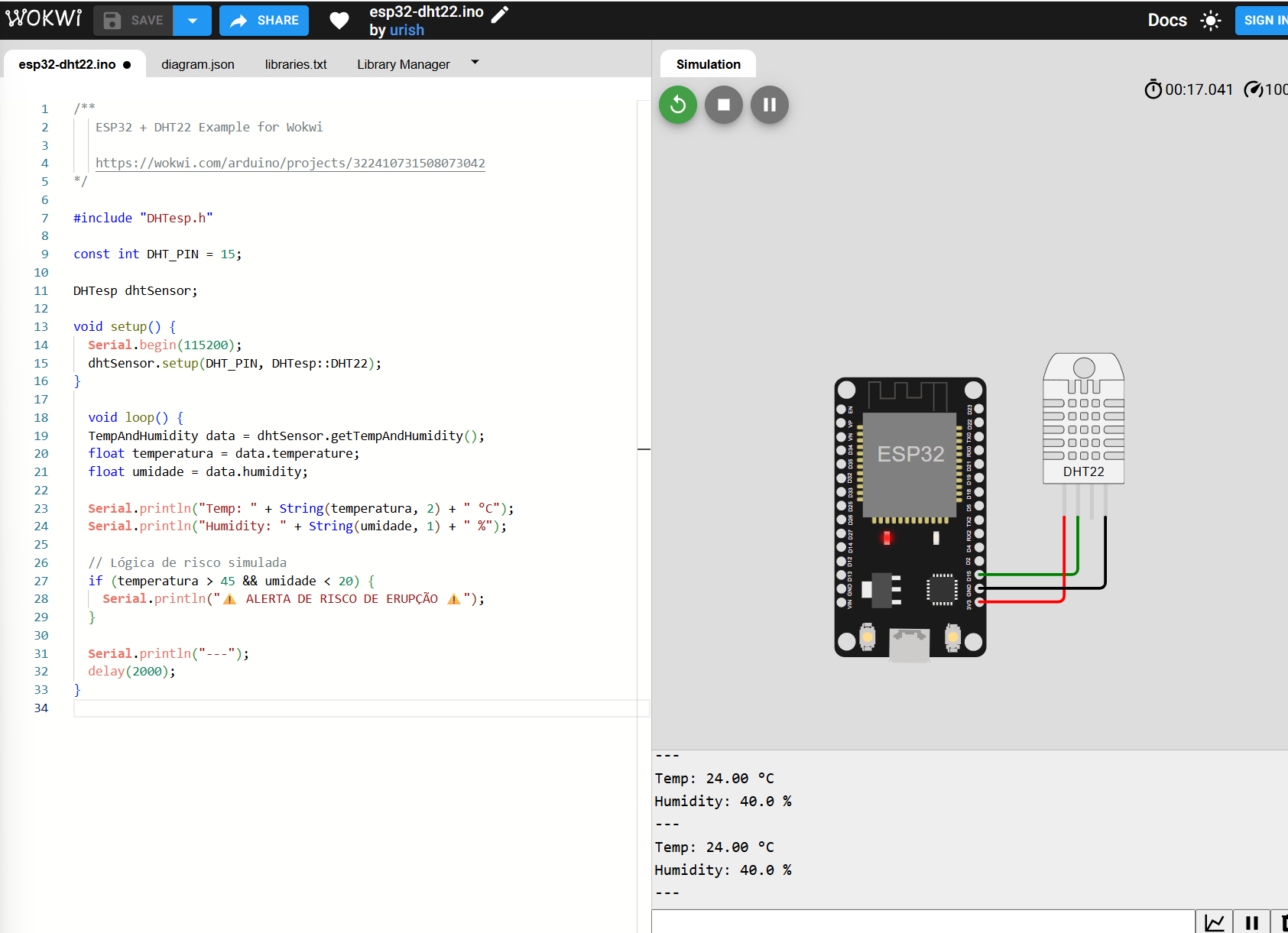
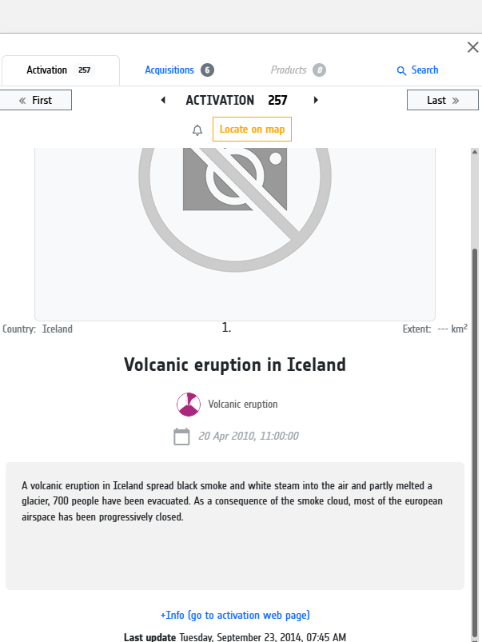
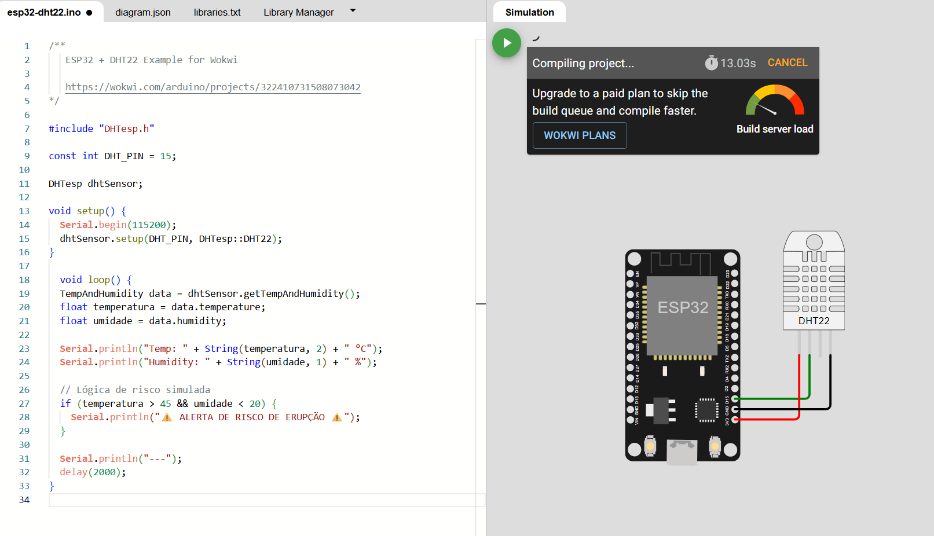
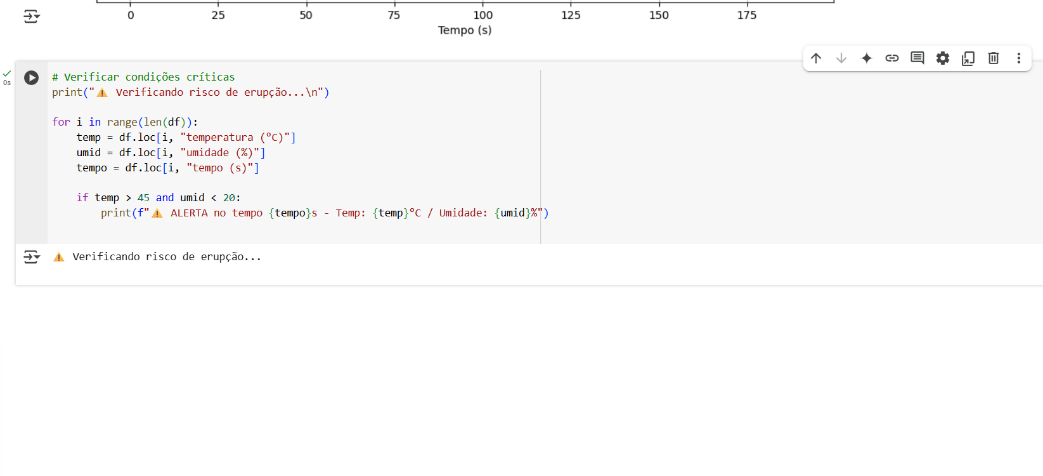
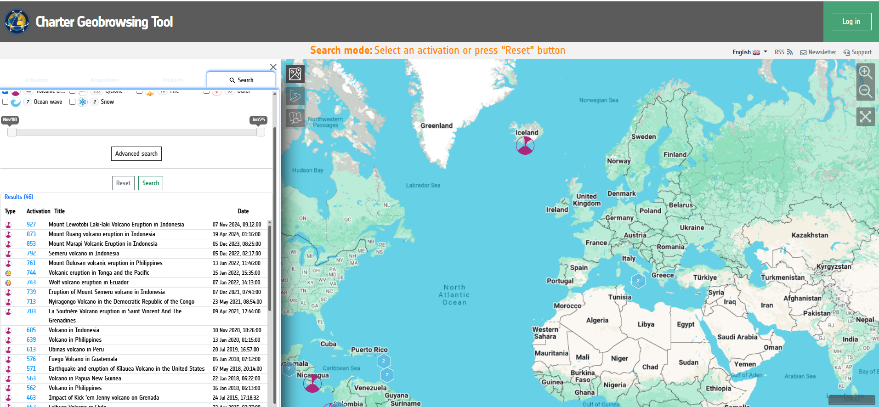
Assim, ele pode ser usado como referência para outras iniciativas educacionais.

**Etapa 7 – Referências**

* 🔗 GitHub do projeto:  
  <https://github.com/flahbocchino/global-solution-vulcao-2025>
* 🔗 Notebook Python (Google Colab): <https://colab.research.google.com/drive/1iYSYdviyoVSrE-5FApap5HjHP_1VGkEz?usp=sharing>
* Wokwi: <https://wokwi.com/projects/322410731508073042?utm_source=chatgpt.com>

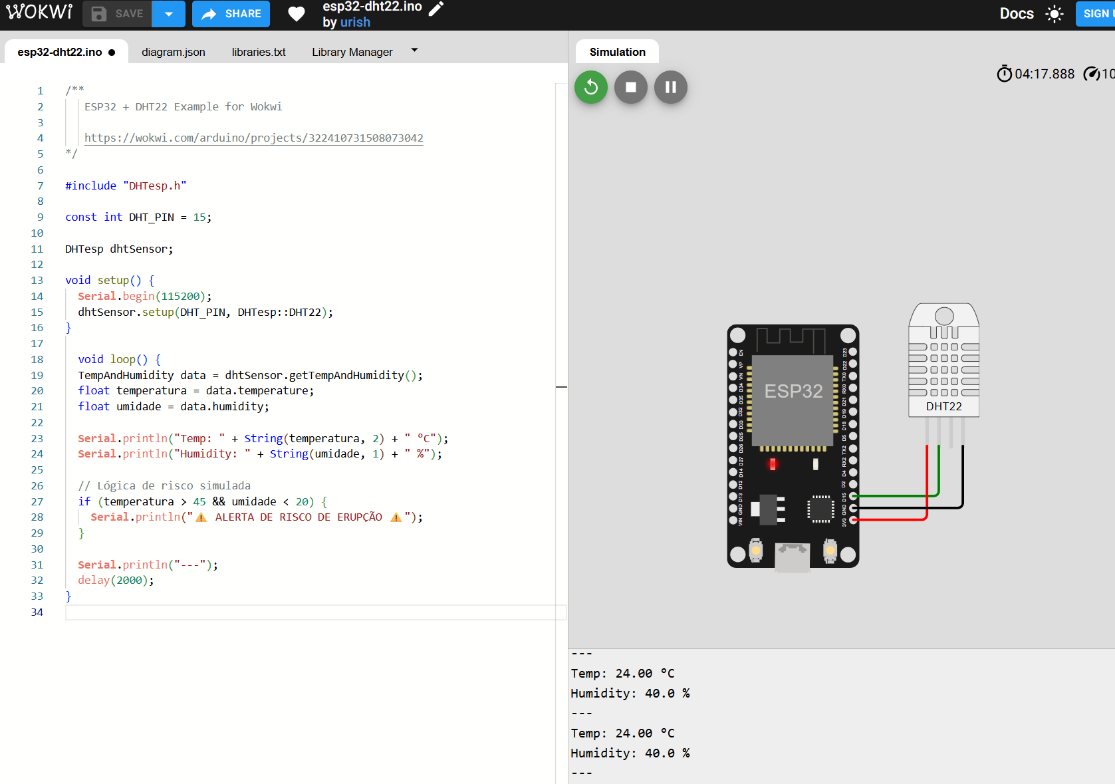
**Etapa 8 – Evidências Visuais**

|  |  |  |
| --- | --- | --- |
| Apêndice – Tabela de Figuras Utilizadas no Projeto | | |
| 1 | print\_evento\_erupcao\_islandia.png | |  | | --- | |  |  |  | | --- | | Registro do evento de erupção do vulcão Eyjafjallajökull (2010) | |
| 2 | descricao\_atividade\_257\_islandia.png | |  | | --- | |  |  |  | | --- | | Detalhes oficiais da ativação nº 257 com data, tipo e impacto | |
| 3 | charter\_geobrowser\_ativacao257\_localizacao.png | Mapa da ativação nº 257 destacada no Charter Geobrowsing Tool |
| 4 | mapa\_ativacao257\_cobertura\_satellite.png | Áreas de observação por satélite durante a erupção na Islândia |
| 5 | |  | | --- | |  |  |  | | --- | | wokwi\_inicio\_simulacao\_esp32.png | | |  | | --- | |  |  |  | | --- | | Interface inicial do Wokwi com ESP32 sem lógica adicionada | |
| 6 | |  | | --- | |  |  |  | | --- | | codigo\_esp32\_alerta\_erupcao.png | | |  | | --- | |  |  |  | | --- | | Código que avalia temperatura e umidade com estrutura condicional | |
| 7 | simulacao\_esp32\_dht22\_funcionando.png | Simulação em execução mostrando dados no monitor serial |
| 8 | compilando\_sistema\_esp32\_alerta\_wokwi.png | Código sendo compilado no Wokwi para teste funcional |
| 9 | sistema\_pronto\_simulacao\_wokwi\_esp32\_dht22.png | Sistema final montado e pronto para simulação no Wokwi |
| 10 | |  | | --- | |  |  |  | | --- | | grafico\_colab\_analise\_alerta.png | | Gráfico com limites de risco sobrepostos (Python/Colab) |
| 11 | grafico\_final\_analise\_colab.png | Gráfico final com comportamento ambiental simulado |
| 12 | simulacao\_dados\_colab\_ambiente\_vulcao.png | Geração de dados de temperatura e umidade em Python |
| 13 | |  | | --- | |  |  |  | | --- | | colab\_analise\_risco\_erupcao\_python.png | | Bloco de código em Python que detecta condições críticas e exibe alertas |

****

**Etapa 9 Complementar: Arquitetura da Solução (visual)**

Nesta etapa incluímos uma **imagem do circuito** montado virtualmente no Wokwi, mostrando como o sensor está conectado ao ESP32, simulando a leitura dos dados ambientais. Essa arquitetura representa como o sistema funcionaria na prática em uma instalação real.



**Etapa 10 Complementar: Circuito e Código ESP32 (.ino)**

Aqui mostramos o código utilizado no ESP32, escrito em linguagem C++ (Arduino). Ele realiza a leitura da temperatura e umidade, compara os valores com os critérios de risco definidos (ex: temperatura > 45 °C e umidade < 20%), e exibe um alerta no monitor serial se o risco for detectado.

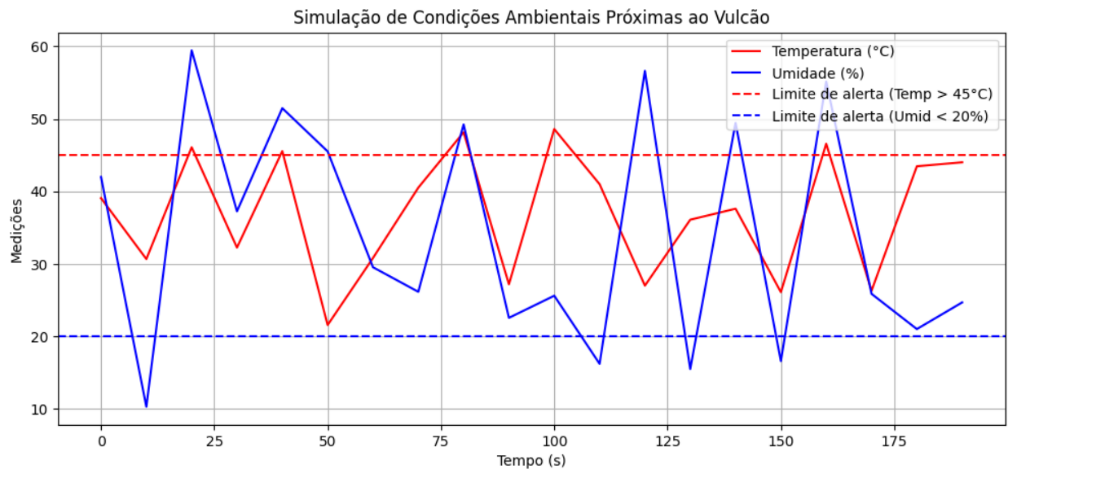


**Etapa 11 Complementar: Lógica de Análise em Python**

Esta etapa detalha o trecho de código Python usado no Google Colab. O script percorre os dados simulados e aplica a mesma lógica do ESP32 para verificar se há risco de erupção, além de gerar um gráfico que mostra visualmente os momentos críticos. O objetivo é complementar o monitoramento com análise gráfica e alertas automatizados.

Nesta etapa, o código em Python foi utilizado para simular dados ambientais e verificar condições críticas associadas à erupção de um vulcão. Com os dados gerados, foi possível aplicar a mesma lógica usada no ESP32 para identificar situações de risco com base na temperatura e na umidade.

Quando a **temperatura ultrapassa 45 °C** e a **umidade está abaixo de 20%**, o sistema reconhece esse cenário como um possível sinal de erupção e exibe um alerta no console. Além disso, um gráfico é gerado para que os dados possam ser analisados visualmente.



**Etapa 12 - Conclusão**

Este projeto mostrou como é possível usar tecnologias acessíveis para simular um sistema de alerta de risco vulcânico. A solução integra o uso de um sensor (DHT22), uma placa de controle (ESP32), lógica computacional e análise gráfica em Python — tudo organizado de forma funcional e pedagógica.

Mesmo sendo uma simulação, o projeto representa o funcionamento de sistemas reais utilizados em regiões de risco. Ele reforça o papel da tecnologia como aliada na prevenção de desastres naturais, podendo salvar vidas.

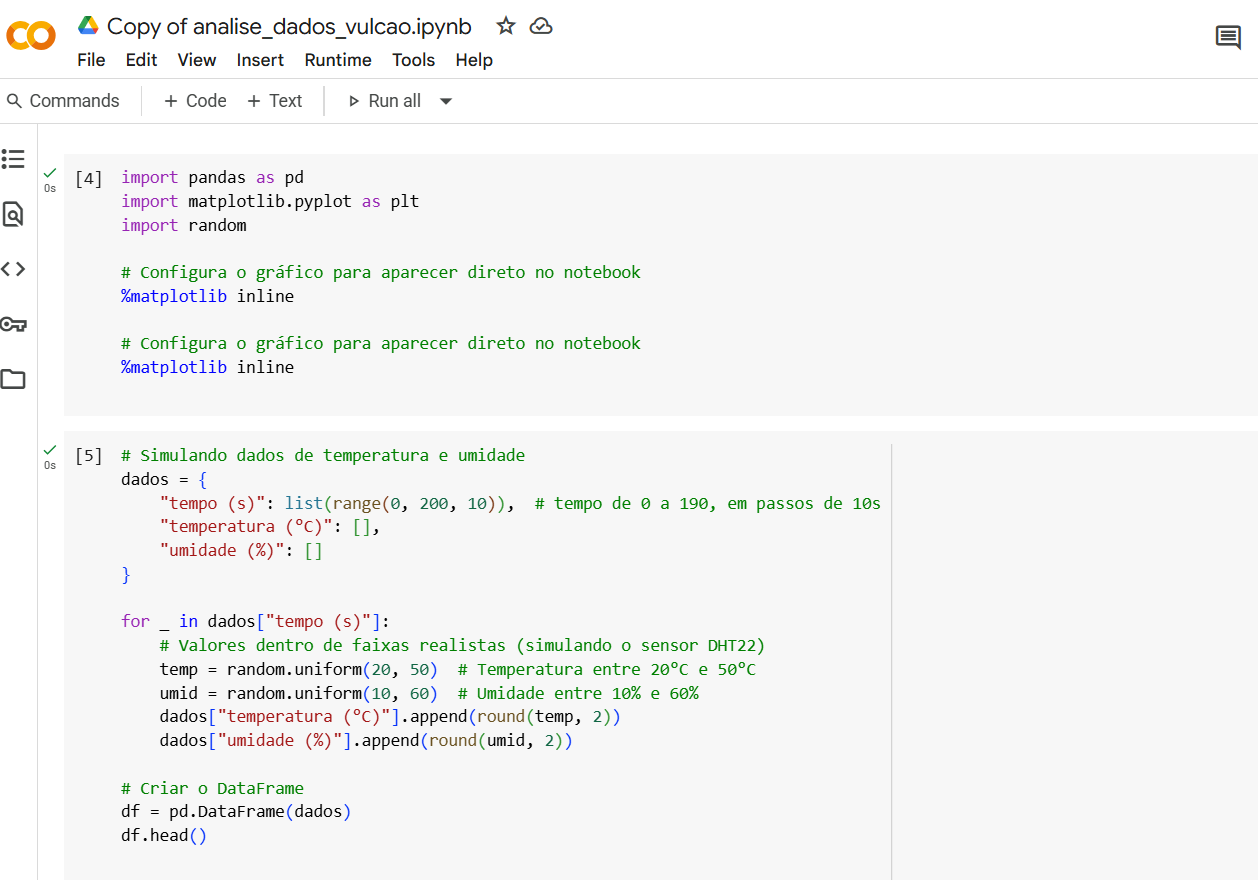
Além disso, a experiência permitiu aplicar, de forma prática, conhecimentos sobre lógica de programação, estruturação de dados, simulação embarcada e análise visual. O aprendizado envolveu tanto o desenvolvimento técnico quanto a visão crítica sobre aplicações tecnológicas em contextos sociais e ambientais.

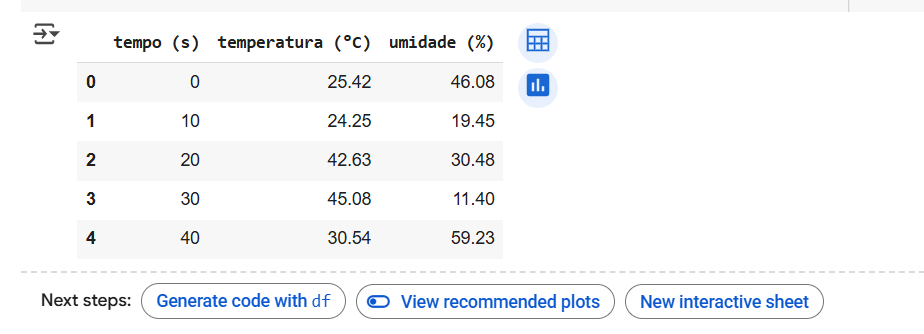
**Testes finais:**

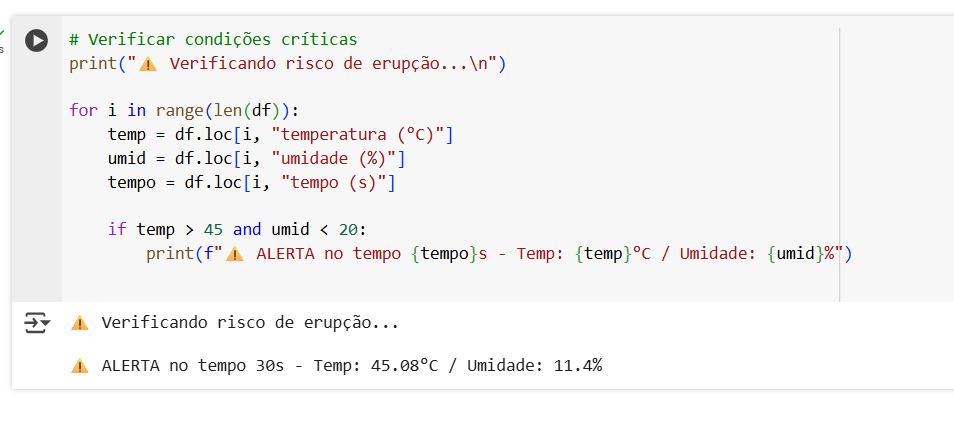
No Google Colab, os dados simulados foram processados com a mesma lógica do ESP32, permitindo detectar cenários de risco como:

* Temperatura: 45.08 °C
* Umidade: 11.4%

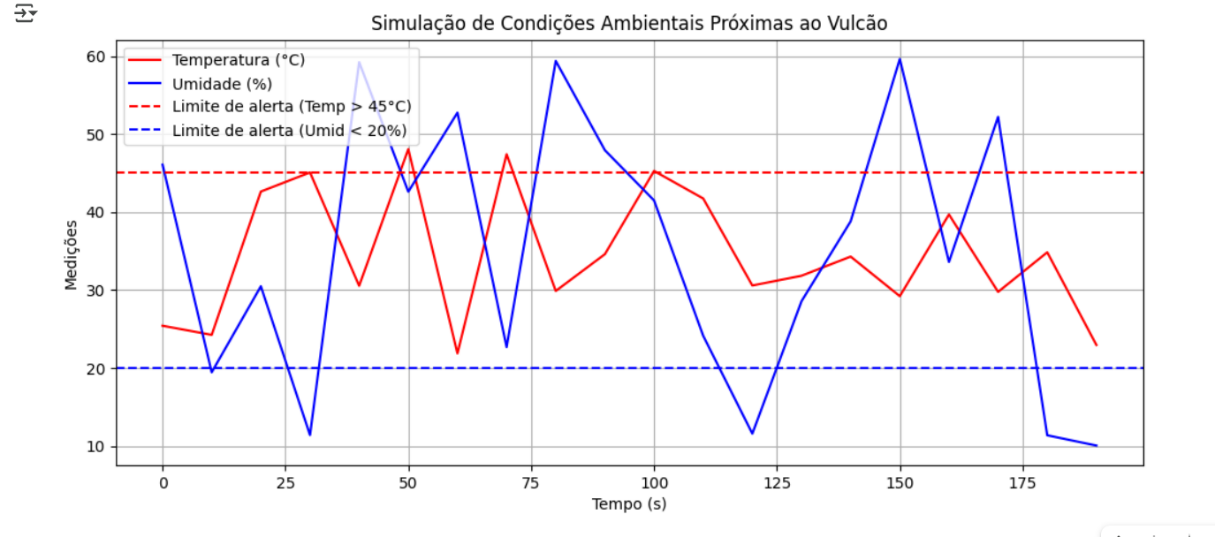
Essa combinação ativou a mensagem de alerta, confirmando a precisão do sistema. O gráfico gerado também demonstrou claramente os momentos críticos com base nos limites pré-definidos (Temp > 45 °C, Umidade < 20%).











Text, letter

AI-generated content may be incorrect.

Graphical user interface, text

AI-generated content may be incorrect.

Graphical user interface

AI-generated content may be incorrect.

Graphical user interface

AI-generated content may be incorrect.

**Etapa 13 – Vídeo no Youtube**

<https://youtu.be/z1PHMGug9sA>