

O que o código faz?

Este código:

- Ativa a câmera do computador (entrada)
 - Captura uma foto
 - Processa a imagem procurando rostos
 - Conta quantos rostos aparecem
 - Mostra a imagem com os rostos destacados e informa o número detectado
-

Relação com os Componentes de Hardware

1. Entrada



Função: Envia dados para o computador



No código: A câmera do dispositivo é ativada para capturar uma imagem.



Trecho relevante:

```
from IPython.display import display, Javascript
```

```
# Usa a webcam via navegador (navigator.mediaDevices)
```

|  **Hardware envolvido:**

- **Câmera (webcam)** → envia imagem ao computador (como um scanner ou microfone envia sinais)
-

2. Processamento



Função: Executa cálculos e lógica do programa



No código: A imagem capturada é transformada em dados (matriz de pixels), convertida para tons de cinza e passada por um algoritmo de detecção de rostos.



Trechos importantes:

```
gray = cv2.cvtColor(image_np, cv2.COLOR_RGB2GRAY) # converte para cinza
```

```
faces = face_cascade.detectMultiScale(gray, 1.3, 5) # detecta rostos
```

| 🎯 **Hardware envolvido:**

- **CPU** → executa o código Python, converte a imagem, roda o algoritmo
- **GPU (se disponível)** → pode acelerar a detecção e manipulação de imagem (OpenCV)

3. Armazenamento



Função: Guarda dados temporários ou permanentes



No código: A imagem capturada é convertida em bytes e armazenada na memória para processamento posterior.



Trechos relevantes:

```
image_bytes = take_photo()
```

```
image = PILImage.open(io.BytesIO(image_bytes))
```

| 🎯 **Hardware envolvido:**

- **RAM** → armazena temporariamente a imagem e os dados do rosto
- (se salvar a imagem, **HD/SSD** será envolvido)

4. Saída



Função: Exibe os resultados ao usuário



No código: Mostra na tela a quantidade de rostos detectados e a imagem com retângulos ao redor dos rostos.



Trechos importantes:

```
print(f"✅ Rostos detectados: {len(faces)}")
```

```
display(PILImage.fromarray(image_np))
```

| 🎯 **Hardware envolvido:**

- **Monitor** → exibe texto e imagem com as marcações
- (opcionalmente: **alto-falante**, se quisesse emitir um som)

ENTRADA (Câmara)



Captura da imagem



PROCESSAMENTO (CPU/GPU)

- Conversão de imagem
- Detecção de rostos



ARMAZENAMENTO (RAM)

- Guarda imagem e dados temporariamente



SAÍDA (Monitor)

- Mostra imagem com rostos marcados
- Imprime o número de rostos detectados

Conclusão

Mesmo um código simples como esse envolve **diversos componentes de hardware**, mostrando na prática a interação entre **entrada, processamento, armazenamento e saída**. Isso ajuda estudantes a **verem o computador como um sistema completo**, não apenas como uma tela e teclado.

O que o código faz?

Este programa:

1. Pede ao usuário que envie um arquivo de áudio .wav (entrada)
 2. Reproduz o áudio (saída)
 3. Analisa o conteúdo do áudio, detectando "eventos sonoros" (picos de som)
 4. Exibe quantos eventos foram encontrados
 5. Mostra um gráfico da energia sonora ao longo do tempo (saída visual)
-

Relação com os Componentes de Hardware

1. Entrada



Função: Envia dados para o computador



No código: O usuário **seleciona e envia um arquivo de áudio** usando um botão.



Trecho relevante:

```
uploaded = files.upload()
```

```
filename = next(iter(uploaded))
```



Hardware envolvido:

- **Dispositivos de entrada:** Mouse e teclado (para navegar e escolher o arquivo)
 - **HD/SSD (do usuário):** Origem do arquivo enviado
-

2. Processamento



Função: Executa cálculos e lógica do programa



No código: O áudio é lido, a energia de cada trecho ("frame") é calculada, e são identificados **eventos sonoros** (como batidas, aplausos, ruídos altos etc).



Trechos importantes:

```
y, sr = librosa.load(filename, sr=None)
```

```
energy = np.array([...])  
events = np.where(energy > threshold)[0]  
event_count = ...
```

| 🎯 **Hardware envolvido:**

- **CPU** → realiza os cálculos com a biblioteca librosa e numpy
 - **(opcional) GPU** → se estiver habilitada no Colab, pode acelerar o processamento de sinais
-

3. Armazenamento



Função: Guarda dados temporários para análise e visualização

🔧 **No código:** O arquivo de áudio é carregado na memória; as energias dos frames e eventos são armazenados temporariamente.

📌 **Trechos relevantes:**

```
y, sr = librosa.load(filename, sr=None)  
energy = np.array([...])
```

| 🎯 **Hardware envolvido:**

- **RAM** → onde os dados do áudio e os resultados dos cálculos ficam armazenados enquanto o código roda
-

4. Saída



Função: Exibe os resultados ao usuário

🔧 **No código:** O áudio é reproduzido em um player interativo, é impresso o número de eventos detectados e é exibido um gráfico da energia sonora.

📌 **Trechos importantes:**

```
display(Audio(filename)) # player de áudio  
print(f"🔊 Eventos sonoros detectados: {event_count}") # texto  
plt.plot(...) # gráfico
```

| 🎧 Hardware envolvido:

- **Caixa de som / Fone de ouvido** → permite ouvir o áudio
 - **Monitor** → exibe texto e gráfico
-

📄 Fluxo Ilustrado

ENTRADA (HD/Mouse/Teclado)

↓

Upload de arquivo .wav

↓

ARMAZENAMENTO (RAM)

- Guarda o áudio carregado e energia por frame

↓

PROCESSAMENTO (CPU)

- Converte som em números
- Calcula energia sonora
- Detecta eventos

↓

SAÍDA (Monitor/Alto-falante)

- Reproduz som
 - Mostra total de eventos
 - Exibe gráfico
-

🧩 Conclusão

Este exemplo mostra como um simples código de análise de áudio envolve **diversos tipos de hardware**: da **entrada do arquivo** à **reprodução do som** e **análise visual e estatística**. Ele é ótimo para demonstrar:

- Como o computador **entende o som como números**
 - Como **energia sonora** pode indicar "eventos"
 - E como a **visualização** facilita a interpretação de dados complexos.
-

O que o código faz?

Esse código **simula** um sistema físico com sensores e atuadores, comum em projetos de **IoT ou automação residencial**.

Ele:


1. Gera dados de sensores (temperatura e luminosidade)
 2. Aplica regras para ativar ou desativar dispositivos (ar-condicionado e lâmpada)
 3. Registra tudo em uma tabela
 4. Mostra gráficos com os dados e os estados dos atuadores
-

Relação com os Componentes de Hardware

1. Entrada



Função: Envia dados para o computador

 **No código:** Os dados dos sensores são **simulados** via geração aleatória, mas **representam sensores reais** de:

- **Temperatura** (como DHT11, LM35)
- **Luminosidade** (como LDR, fotodiodo)

|  **Trechos relevantes:**

```
temperaturas = np.random.normal(loc=24, scale=3, size=n)
```

```
luminosidades = np.random.normal(loc=400, scale=150, size=n)
```

|  **Hardware que seria usado em um cenário real:**

- **Sensor de temperatura**
- **Sensor de luz**

No caso da simulação, esses sensores são substituídos por **funções do processador** que geram os valores.

2. Processamento



Função: Executa cálculos e lógica do programa

🔧 **No código:** A lógica de controle decide o estado dos atuadores com base nos dados simulados:

- Se a temperatura passar de 25°C → liga o ar-condicionado
- Se a luminosidade estiver abaixo de 300 lux → liga a lâmpada

| 📌 **Trechos importantes:**

```
def controle_atuadores(temp, luz):
```

```
    ar_condicionado = "ON" if temp > 25 else "OFF"
```

```
    lampada = "ON" if luz < 300 else "OFF"
```

| 🎯 **Hardware envolvido:**

- **CPU** → executa a lógica do programa e decide as ações
- Em sistemas reais: **Microcontroladores** (como Arduino ou ESP32) fariam esse papel

3. Armazenamento



Função: Guarda dados temporários ou para análise posterior

🔧 **No código:** As leituras dos sensores e os estados dos atuadores são **registrados em uma tabela (DataFrame)**.

| 📌 **Trecho relevante:**

```
df = pd.DataFrame(dados)
```

| 🎯 **Hardware envolvido:**

- **RAM** → armazena os dados simulados e processados temporariamente
- Em sistemas reais: **Memória flash ou banco de dados local/remoto**

4. Saída



Função: Exibe os resultados ao usuário

🔧 **No código:** Exibe:

- Uma **tabela** com as leituras e ações

- Dois **gráficos interativos**: temperatura x estado do ar, luminosidade x estado da lâmpada

| 📌 Trechos importantes:

`display(df.tail(8))` # mostra tabela

`plt.plot(...)` # mostra gráficos

| 🎯 Hardware envolvido:

- **Monitor** → exibe os gráficos e a tabela
- Em sistemas físicos: os atuadores são **dispositivos físicos** como:
 - **Relés para ligar/desligar o ar-condicionado**
 - **LEDs ou lâmpadas reais**

📄 Fluxo Ilustrado

ENTRADA (Simulada: sensores de temperatura e luz)

↓

Geração de dados de sensores

↓

PROCESSAMENTO (CPU)

- Aplica lógica de controle
- Decide quando ligar ou desligar atuadores

↓

ARMAZENAMENTO (RAM)

- Guarda os dados e decisões para exibição

↓

SAÍDA (Monitor)

- Exibe tabela com leituras e estados
- Mostra gráficos com o comportamento ao longo do tempo

🌿 Conclusão

Esse código é um **excelente exemplo didático** para mostrar como um **sistema embarcado ou de IoT** funciona:

- **Sensores** geram dados (entrada)

- **Unidade de controle** aplica lógica (processamento)
 - **Decisões** são armazenadas (armazenamento)
 - **Estados** dos atuadores são atualizados e mostrados (saída)
-