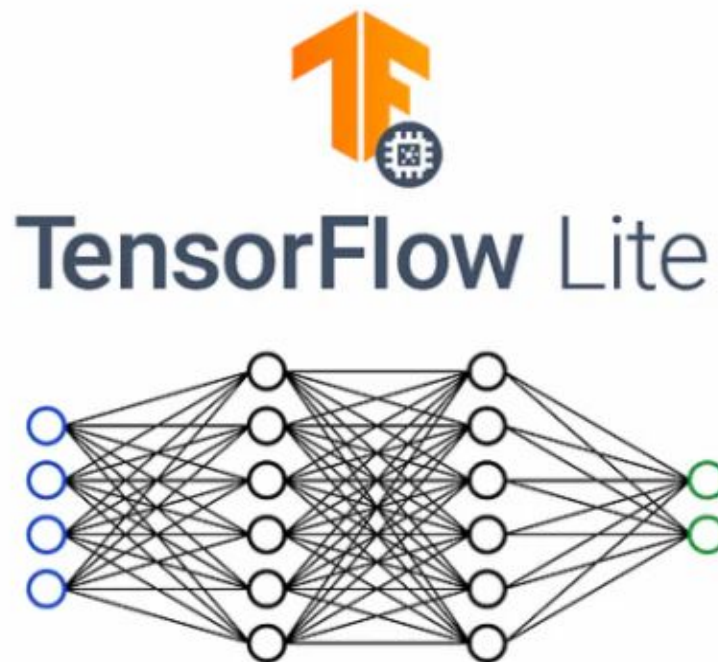


Universidade Católica de Petrópolis

Semana Científica do CEC - 2025

Minicurso: Sua Primeira Inteligência Artificial Embarcada: Criando um Dispositivo Ativado por Voz com TinyML.

Robson C. Augusto
Felipe Baldner
Ana Carolina Carius



- [1] Training
- [2] Distillation
- [3] Quantization
- [4] Encoding
- [5] Compilation



TinyML

Minicurso: Sua Primeira Inteligência Artificial Embarcada: Criando um Dispositivo Ativado por Voz com TinyML.

Link do repositório com material do mini curso

[r4h1/Mini-Curso-TinyML-com-ESP32: Mini curso voltado para treinamento e implementação de TinyML \(modelos de machine learning pequenos\) junto com microcontroladores. O objetivo deste mini curso é treinar um modelo e rodar diretamente em ESP32 WROVER.](#)

Outros links úteis.



[Arduino - Home](#)



[colab.google](#)



[Visual Studio Code - Code Editing. Redefined](#)



[TensorFlow Lite](#)

Minicurso: Sua Primeira Inteligência Artificial Embarcada: Criando um Dispositivo Ativado por Voz com TinyML.



[ESP32 Wi-Fi & Bluetooth SoC | Espressif Systems](https://www.espressif.com/en/products/systems/esp32)



[Welcome to Python.org](https://www.python.org)

[PyPI · O Python Package Index](https://pypi.org)



Minicurso: IA e Visão Computacional com Arduíno: Controle de Acesso Inteligente

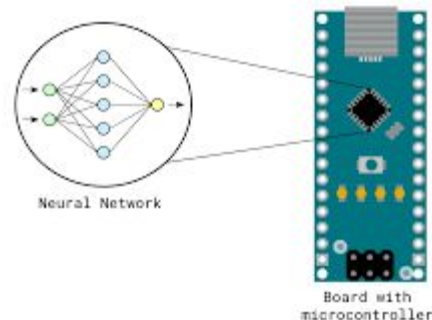
- 1 - Introdução e O Mundo do TinyML.**
- 2 - Preparando o Ambiente e o Hardware.**
- 3 - Implementação com o Modelo Pré-Treinado.**
- 4 - Testes, Resultados, Perguntas e Agradecimento.**

1 - Introdução e O Mundo do TinyML.

O que é IA e Machine Learning? (Analogia: Ensinar um computador a reconhecer padrões, como uma criança aprende a diferença entre um gato e um cachorro).

O que é TinyML? (Levar essa IA para "cérebros" minúsculos e de baixo consumo, sem precisar da internet. Analogia: A diferença entre perguntar para a Siri/nuvem e ter um dicionário no bolso).

Por que isso é revolucionário? (Privacidade, velocidade, baixo custo, novas aplicações em saúde, agricultura, indústria).

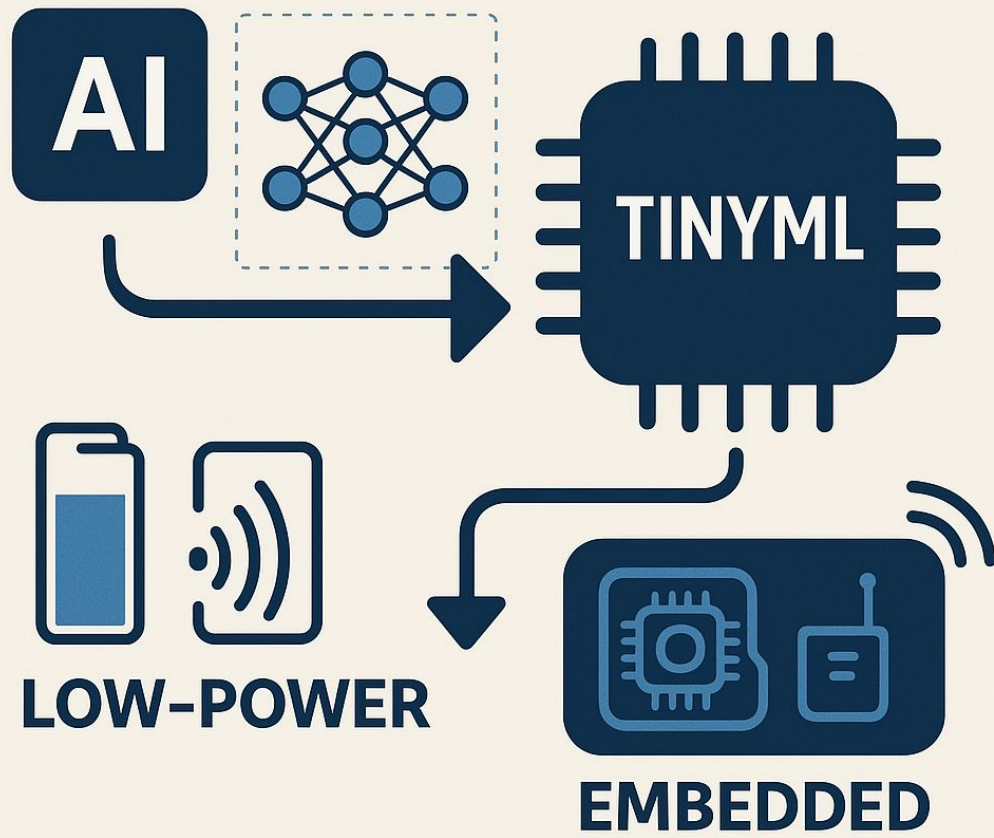


1 - Introdução e O Mundo do TinyML.

O Nosso Projeto:

- **O Desafio:** Fazer um ESP32 WROVER, um microcontrolador de baixo consumo e custo, entender a voz humana.

Deploying TinyML Models on Microcontrollers – Running AI on Low-Power Embedded Devices



Tiny AI runs on diverse environment!



HealthCare Devices



Medical Devices



Satellites



Robotics



Smart Factory



Autonomous vehicle



Smart Logistics



Smart Cities



IoT sensors



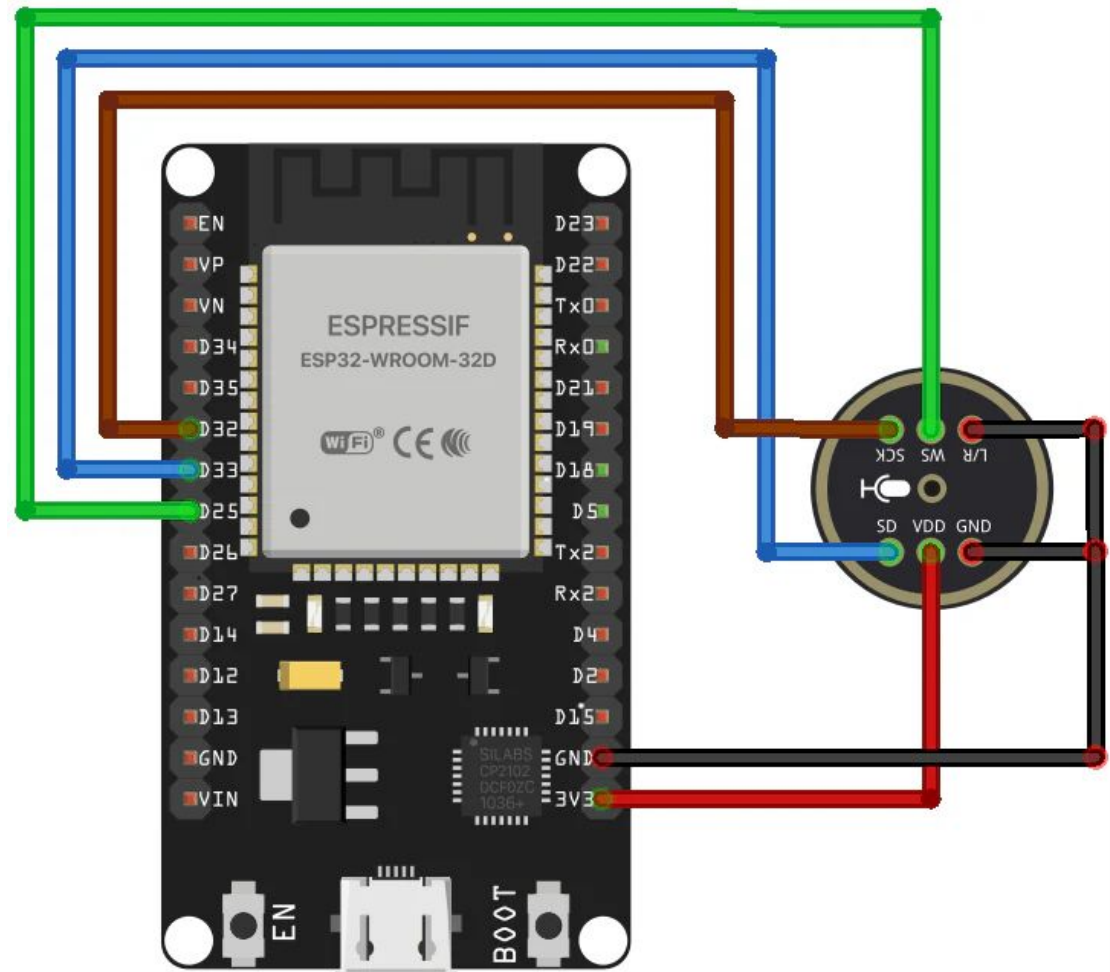
Agriculture



Smart Farming

2 - Preparando o Ambiente e o Hardware.

Hardware: Usaremos um ESP32 WROVER e um microfone digital INMP441.



2 - Preparando o Ambiente e o Hardware.

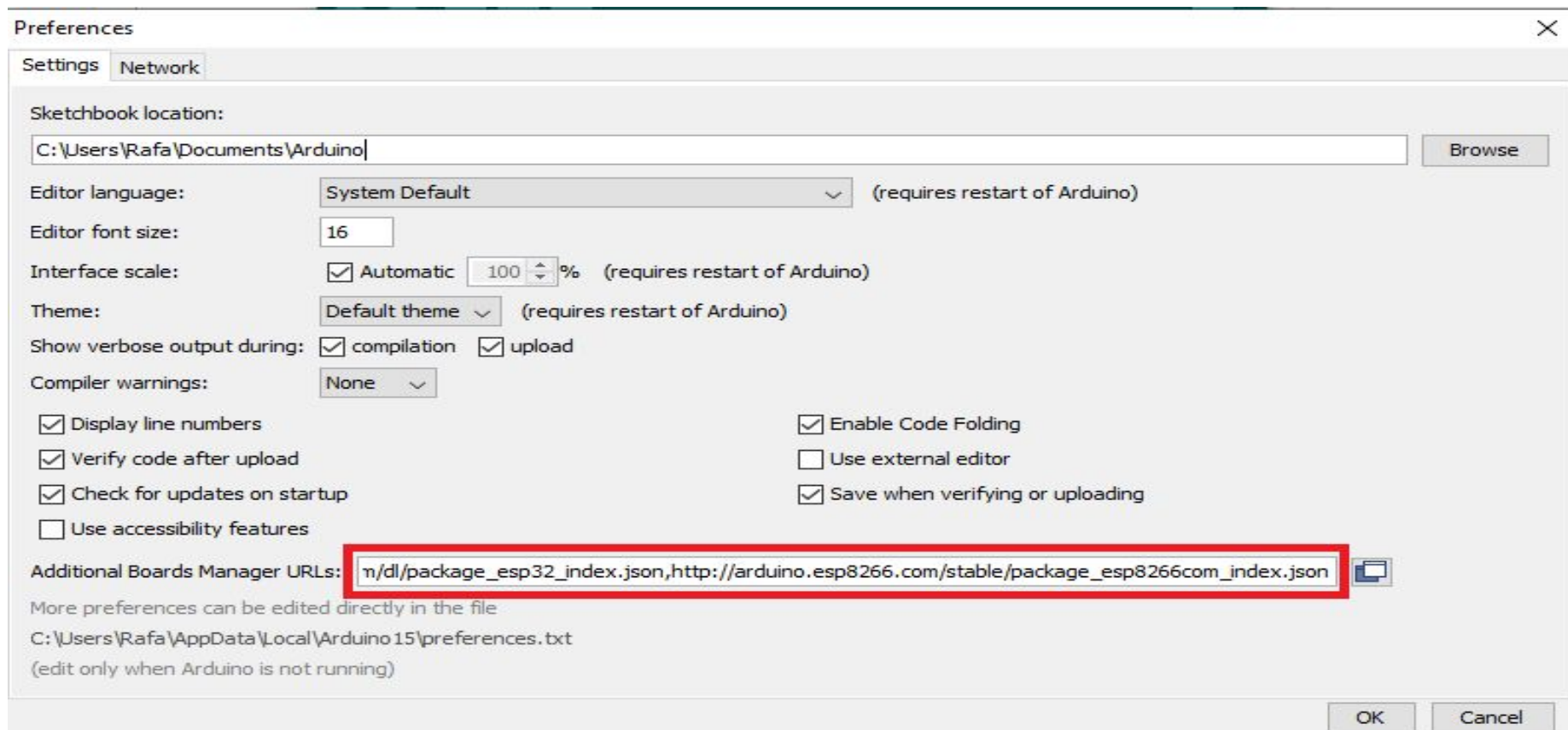
Software e Ambiente:

1 - Abrir a IDE do Arduino.

2 - Adicionar a URL do ESP32 nas preferências.

```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\_esp32\_index.json
```

2 - Preparando o Ambiente e o Hardware.

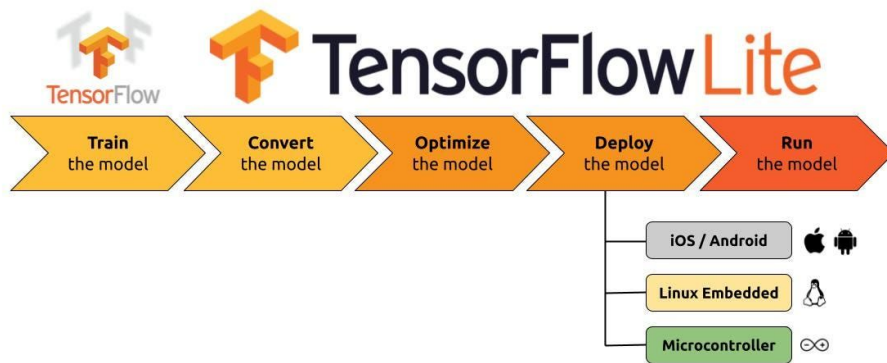


2 - Preparando o Ambiente e o Hardware.

3 - Abrir o gerenciador de placas e instalar o pacote **ESP32 v.2.011**

4 - Abrir o gerenciador de bibliotecas e instalar a biblioteca **arduinoFFT**

5 - Instalar a biblioteca **TensorFlowLite_ESP32 (0.3.0)**



3 - Implementação com o Modelo Pré-Treinado.

1 - Criar uma pasta no desktop com nome 'TinyML_ESP' e salve os Sketches dentro.

2 - Criar um novo sketch na IDE do Arduino com nome 'teste_microfone'

2.1 - Copiar e colar o código no novo sketch, link da descrição do código github abaixo: [Mini-Curso-TinyML-com-ESP32/teste microfone digital at main · r4h1/Mini-Curso-TinyML-com-ESP32](https://github.com/r4h1/Mini-Curso-TinyML-com-ESP32/tree/main)

2.2 - Compilar e Carregar o código no ESP32. Verificar a saída:

Exemplo: *'RMS (Volume): 110.47'*

Conforme falamos no microfone o valor de 'RMS (volume)' deve mudar.

3 - Implementação com o Modelo Pré-Treinado.

3 - Coletando dados para treinar o modelo:

3.1 - Criar um novo sketch com nome de 'ESP32_gravador'.

3.2 - Acessar o link do github e copiar o código do 'ESP32_gravador' e colar nesse novo sketch. Link: [Mini-Curso-TinyML-com-ESP32/ESP32_gravador at main · r4h1/Mini-Curso-TinyML-com-ESP32](https://github.com/r4h1/Mini-Curso-TinyML-com-ESP32/tree/main)

3.3 Compilar e Carregar o código no ESP32.

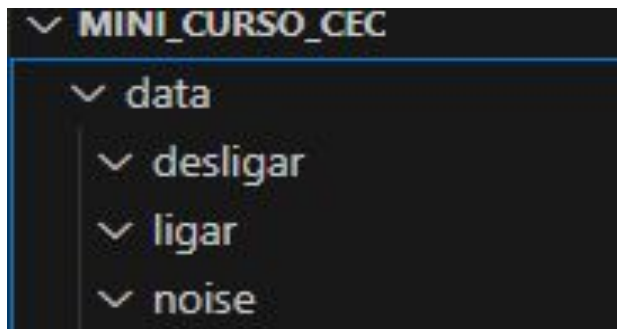
3.3.1 Após carregar, verificar mensagem no monitor serial: *'ESP32 pronto (16 bits mono) Envie 'R' para gravar'* Se a mensagem apareceu, indica que está tudo funcionando. Agora feche o monitor serial na IDE do Arduino. E vamos para o Python.

3 - Implementação com o Modelo Pré-Treinado.

4 - Vamos criar uma pasta no desktop com nome 'Mini_Curso'.

4.1 - Dentro da pasta 'Mini_Curso', vamos criar mais uma pasta, 'data'. E dentro da pasta 'data', vamos criar mais 3 pastas 'ligar', 'desligar' e 'noise'.

4.2 - Vamos abrir o VScode. E depois vamos clicar em 'file' e depois em 'open folder' e vamos achar nossa pasta 'Mini_Curso'. Deve parecer assim:



3 - Implementação com o Modelo Pré-Treinado.

4.3 - Vamos criar um novo arquivo Python, em: 'File > New File > Python File'

4.4 - Vamos nomear esse arquivo Python como 'record_audio'.

4.5 - Vamos copiar e colar o arquivo Python no link do github. Link:

[Mini-Curso-TinyML-com-ESP32/record_audio.py at main · r4h1/Mini-Curso-TinyML-com-ESP32](https://github.com/r4h1/Mini-Curso-TinyML-com-ESP32/blob/main/r4h1/Mini-Curso-TinyML-com-ESP32/record_audio.py)

4.5.1 - Vamos verificar se as bibliotecas Python estão instaladas: Usamos o comando **pip install** + nome da biblioteca.

Exemplo: **pip install numpy pyserial wave matplotlib.pyplot**

Verificamos no terminal do VScode em: ... > **Terminal**

3 - Implementação com o Modelo Pré-Treinado.

4.6 - Após verificar as bibliotecas, vamos salvar e executar o script do Python. Devemos ter uma saída esperada no terminal como mostrado abaixo:

--- Gravador de Áudio (Mono 16 bits + RMS + Gráfico) ---

Digite ('ligar', 'desligar', 'noise', ou 'exit'):

4.6.1 - Vamos gravar as palavras 'ligar', 'desligar' e 'noise (fazer barulho ou deixar o som ambiente)' por trinta (30) vezes cada.

Você deve ver esse menu no terminal do VScode:

```
Digite ('ligar', 'desligar', 'noise', ou 'exit'): ligar  
Pressione Enter para iniciar a gravação...
```

3 - Implementação com o Modelo Pré-Treinado.

Após pressionar 'Enter' deverá esperar aparecer as seguinte mensagens para poder falar e gravar as palavras ou 'noise'.

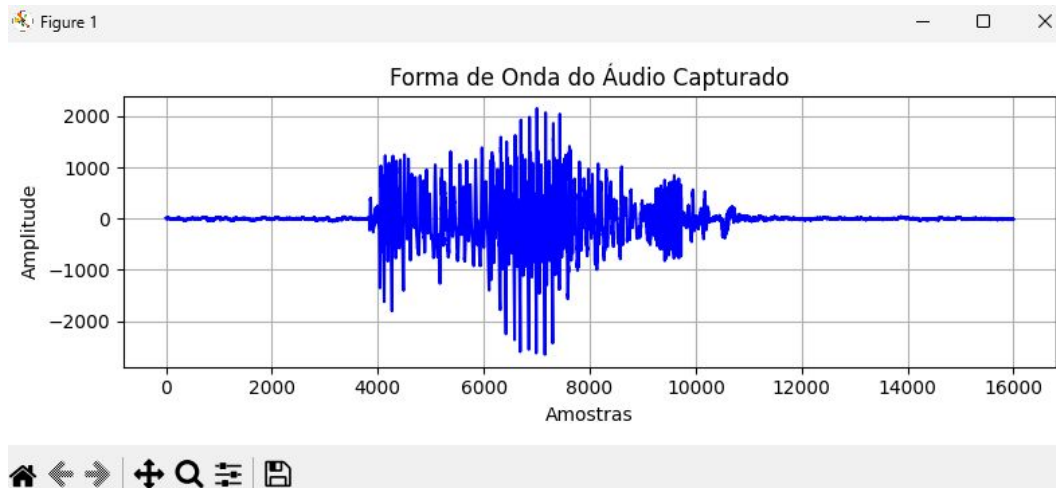
Atenção para escolher e gravar a palavra certa ou 'noise' no menu.

```
--- INICIANDO NOVA GRAVAÇÃO ---  
[DEBUG] Porta serial COM3 aberta com sucesso!  
[DEBUG] Enviando comando 'R'...  
[DEBUG] Aguardando resposta 'A' do ESP32...  
[DEBUG] Handshake OK. Gravando áudio...  
[DEBUG] Bytes recebidos: 32000
```

3 - Implementação com o Modelo Pré-Treinado.

Caso você grave corretamente você verá as seguinte saidas:

```
[DEBUG] Bytes recebidos: 32000  
[INFO] Nível RMS: 371.56  
  
>>> SUCESSO! Gravação salva em data/noise/noise_31.wav <<<
```



3 - Implementação com o Modelo Pré-Treinado.

5 - Hora de Treinar seu modelo com os dados obtidos das gravações. Tente gravar de 50-100 dados para cada.

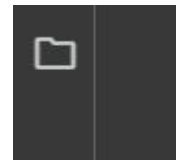
5.1 - Vamos compactar a pasta 'data' para um formato 'Zip'. Teremos então duas pastas, uma 'data' e a outra 'data.zip'.

5.2 - Agora vamos logar em sua conta do google e vamos abrir um novo 'Colab'. Acesse o link: colab.google

5.3 - Copie e cole o script Python, no seu novo 'colab', que está no link a seguir: [Mini-Curso-TinyML-com-ESP32/Treinamento_do_modelo_Mini_curso_CEC.ipynb at main · r4h1/Mini-Curso-TinyML-com-ESP32](https://colab.research.google.com/github/Mini-Curso-TinyML-com-ESP32/Treinamento_do_modelo_Mini_curso_CEC.ipynb)

3 - Implementação com o Modelo Pré-Treinado.

5.4 - Após copiar e colar cada célula do script em seu colab, vamos adicionar a pasta de dados 'data.zip' no colab. Clique no ícone da pasta, então pegamos a pasta 'data.zip' e arrastamos para área que abrirá.

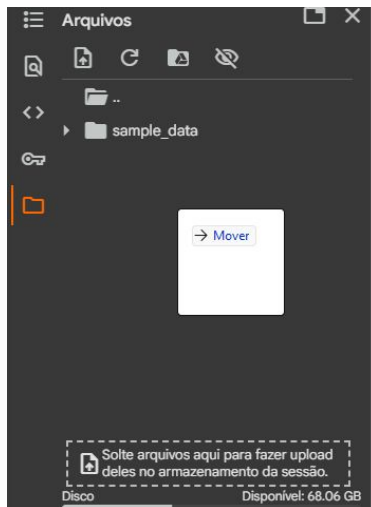
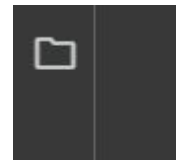


5.5 - Pronto! Vamos ao treinamento.

5.6 - Após o treinamento, vamos baixar o arquivo 'model.h' para usarmos no código final do Arduino.

3 - Implementação com o Modelo Pré-Treinado.

5.4 - Após copiar e colar cada célula do script em seu colab, vamos adicionar a pasta de dados 'data.zip' no colab. Clique no ícone da pasta, daí pegamos a pasta 'data.zip' e arrastamos para área que abrirá.

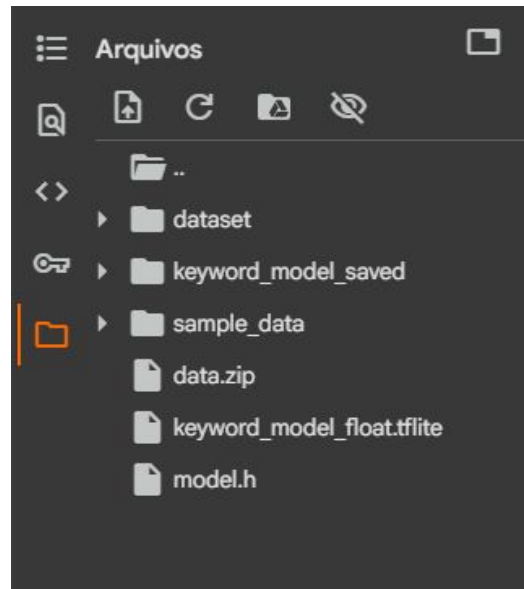


3 - Implementação com o Modelo Pré-Treinado.

5.5 - Pronto! Vamos ao treinamento.

5.5.1 - Importante! Não modifique o código, no Colab, de treinamento. Execute todas as células, desde a primeira até a ultima. Use a versão do Tensorflow recomendada, v2.16.1.

5.6 - Após o treinamento, vamos baixar o arquivo 'model.h' para usarmos no código final do Arduino.



3 - Implementação com o Modelo Pré-Treinado.

6 - Vamos para etapa final da implementação do modelo.



6.1 - Vamos para IDE do Arduino e vamos criar um novo Sketch, com o nome 'modelo_treinado_esp_wrover'.

6.2 - Vamos copiar e colar o código que está no link a seguir, no novo sketch: [Mini-Curso-TinyML-com-ESP32/modelo_treinado_esp_wrover at main · r4h1/Mini-Curso-TinyML-com-ESP32](https://github.com/r4h1/Mini-Curso-TinyML-com-ESP32/blob/main/modelo_treinado_esp_wrover.ino)

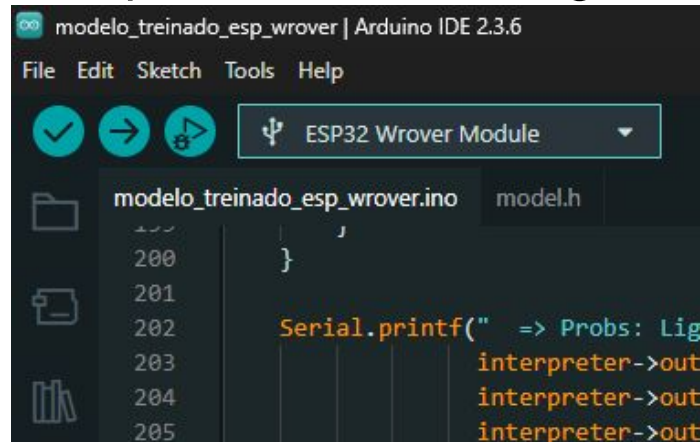
6.3 - Precisamos criar uma nova aba nesse sketch para incluir o nosso modelo. Para isso, basta colocarmos o modelo na mesma pasta do sketch que nomiamos 'modelo_treinado_esp_wrover'.

3 - Implementação com o Modelo Pré-Treinado.

Como a seguir:

 model	26/10/2025 21:14	Arquivo Fonte C ...	243 KB
 modelo_treinado_esp_wrover	26/10/2025 21:20	Arduino file	9 KB

Com isso seu sketch deve parecer como a seguir:



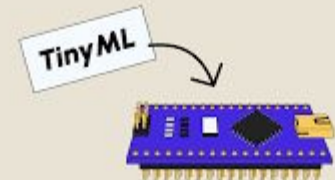
```
200     }  
201  
202     Serial.printf(" => Probs: Lig  
203  
204     interpreter->out  
205     interpreter->out  
206     interpreter->out
```

3 - Implementação com o Modelo Pré-Treinado.

6.4 - Vamos salvar o nosso sketch.

Lembre de selecionar a placa e a porta COM correta!

6.5 - Hora de compilar e carregar tudo para o ESP32 WROVER.



4 - Testes, Resultados, Perguntas e Agradecimento.

Se tudo correu bem, devemos observar os seguintes resultados no monitor serial da IDE do Arduino. E veremos o LED built-in do ESP32 (PIN2) ligar e desligar com os comandos de voz.

```
Ouvindo...
  Spectrograma (canto sup esq): 419.03 | (meio): 29.99 | (canto inf dir): 349.32
=> Probs: Ligar=0.34, Desligar=0.00, Noise=0.66
Ouvindo...
  Spectrograma (canto sup esq): 680.04 | (meio): 30.51 | (canto inf dir): 390.98
=> Probs: Ligar=0.00, Desligar=0.00, Noise=1.00
-----> Noise DETECTADO! (100.00%)
Ouvindo...
  Spectrograma (canto sup esq): 718.98 | (meio): 20.67 | (canto inf dir): 435.97
=> Probs: Ligar=0.11, Desligar=0.00, Noise=0.89
-----> Noise DETECTADO! (88.68%)
Ouvindo...
  Spectrograma (canto sup esq): 294.40 | (meio): 19.36 | (canto inf dir): 377.87
=> Probs: Ligar=0.00, Desligar=0.00, Noise=1.00
-----> Noise DETECTADO! (99.99%)
Ouvindo...
  Spectrograma (canto sup esq): 670.98 | (meio): 20.98 | (canto inf dir): 264.93
=> Probs: Ligar=0.00, Desligar=0.00, Noise=1.00
-----> Noise DETECTADO! (100.00%)
Ouvindo...
  Spectrograma (canto sup esq): 768.93 | (meio): 9.62 | (canto inf dir): 351.00
=> Probs: Ligar=0.92, Desligar=0.00, Noise=0.08
-----> Ligar DETECTADO! (91.90%)
```

4 - Testes, Resultados, Perguntas e Agradecimento.

Perguntas?



4 - Testes, Resultados, Perguntas e Agradecimento.

Obrigado!

GitHub: [r4h1](https://github.com/r4h1) (<https://github.com/r4h1>)

