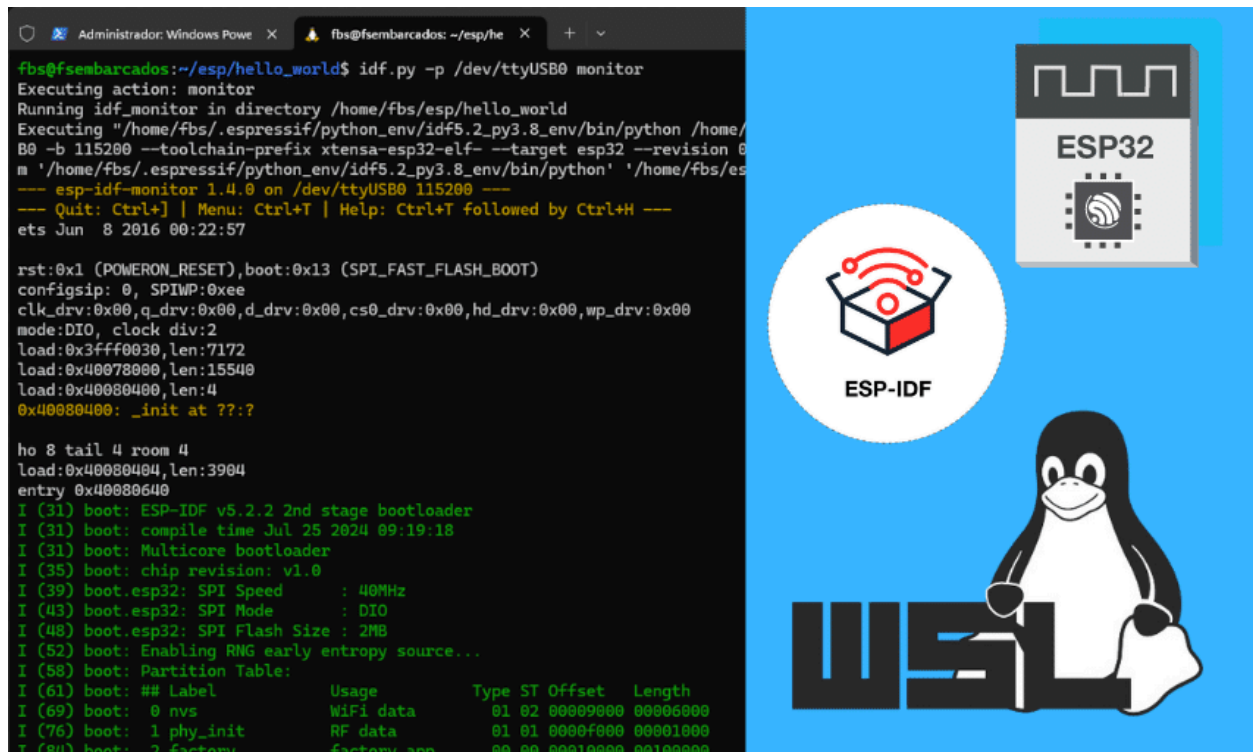


Instalação do ESP-IDF no Windows Subsystem for Linux (WSL)

Autor: [Fábio Souza](#)



O ESP-IDF (Espressif IoT Development Framework) é o principal framework para desenvolvimento com a linha completa de SoCs ESP32 da Espressif, incluindo o ESP32, ESP32-S, ESP32-C, ESP32-H e ESP32-P. Ele proporciona um SDK completo para criar aplicações nessas plataformas, utilizando linguagens de programação como C e C++.

O ESP-IDF é compatível com Windows, Linux e Mac OS. No entanto, usuários do Windows 11 podem utilizar o ESP-IDF através do WSL (Windows Subsystem for Linux), abrindo diversas possibilidades para desenvolvimento. Essa alternativa simplifica a configuração do ambiente de desenvolvimento para ESP32, seja para testes, manutenção ou criação de novos projetos.

Este tutorial irá guiar você na instalação do ESP-IDF no WSL.

O que é WSL?

WSL (Windows Subsystem for Linux) é uma camada de compatibilidade que permite a execução de binários Linux nativamente no Windows 10 e Windows 11. Ele permite que os desenvolvedores usem um ambiente Linux completo sem a necessidade de uma máquina virtual, proporcionando acesso a ferramentas de desenvolvimento poderosas diretamente no Windows. Isso é especialmente útil para desenvolvedores que trabalham com sistemas embarcados no Linux, como o ESP-IDF.

Ferramentas necessárias para configurar o ESP-IDF no WSL

Para a instalação, configuração e uso do ESP-IDF no WSL, vamos precisar das seguintes ferramentas:

1. Computador com Windows 11
2. WSL 2
3. Ubuntu on Windows using WSL
4. [usbipd-win](#)
5. ESP-IDF

Instalar o WSL

Primeiro, precisamos habilitar o WSL no Windows. Abra o PowerShell como Administrador e execute o seguinte comando:

```
wsl --install
```

Isso instalará o WSL e a distribuição padrão do Ubuntu. Se já tiver o WSL instalado, você pode atualizar para a versão mais recente com:

```
wsl --update
```

Se quiser instalar uma nova versão do Linux, você pode verificar a lista de distribuições disponíveis no WSL usando o PowerShell:

```
wsl -l -o
```

```
Administrador: Windows Pow
PS C:\Users\fabio> wsl -l -o
Veja a seguir uma lista de distribuições válidas que podem ser instaladas.
Instale usando 'wsl.exe --install <Distro>'.

NAME                                FRIENDLY NAME
Ubuntu                              Ubuntu
Debian                              Debian GNU/Linux
kali-linux                          Kali Linux Rolling
Ubuntu-18.04                        Ubuntu 18.04 LTS
Ubuntu-20.04                        Ubuntu 20.04 LTS
Ubuntu-22.04                        Ubuntu 22.04 LTS
Ubuntu-24.04                        Ubuntu 24.04 LTS
OracleLinux_7_9                    Oracle Linux 7.9
OracleLinux_8_7                    Oracle Linux 8.7
OracleLinux_9_1                    Oracle Linux 9.1
openSUSE-Leap-15.5                 openSUSE Leap 15.5
SUSE-Linux-Enterprise-Server-15-SP4 SUSE Linux Enterprise Server 15 SP4
SUSE-Linux-Enterprise-15-SP5       SUSE Linux Enterprise 15 SP5
openSUSE-Tumbleweed                openSUSE Tumbleweed
PS C:\Users\fabio>
```

Para instalar uma distribuição específica, adicione o nome da distribuição ao comando:

```
wsl --install --distribution <NomeDaDistribuição>
```

Substitua **<NomeDaDistribuição>** pelo nome da distribuição Linux que você deseja instalar, como **Ubuntu**, **Debian**, **Kali-Linux**, etc. Exemplo:

```
wsl --install --distribution Ubuntu-22.04
```

Instalar e Configurar o **usbipd-win**

Para usar dispositivos USB, Serial ou JTAG no WSL, você precisa instalar o **usbipd-win**, uma ferramenta que permite que dispositivos USB sejam usados dentro do WSL.

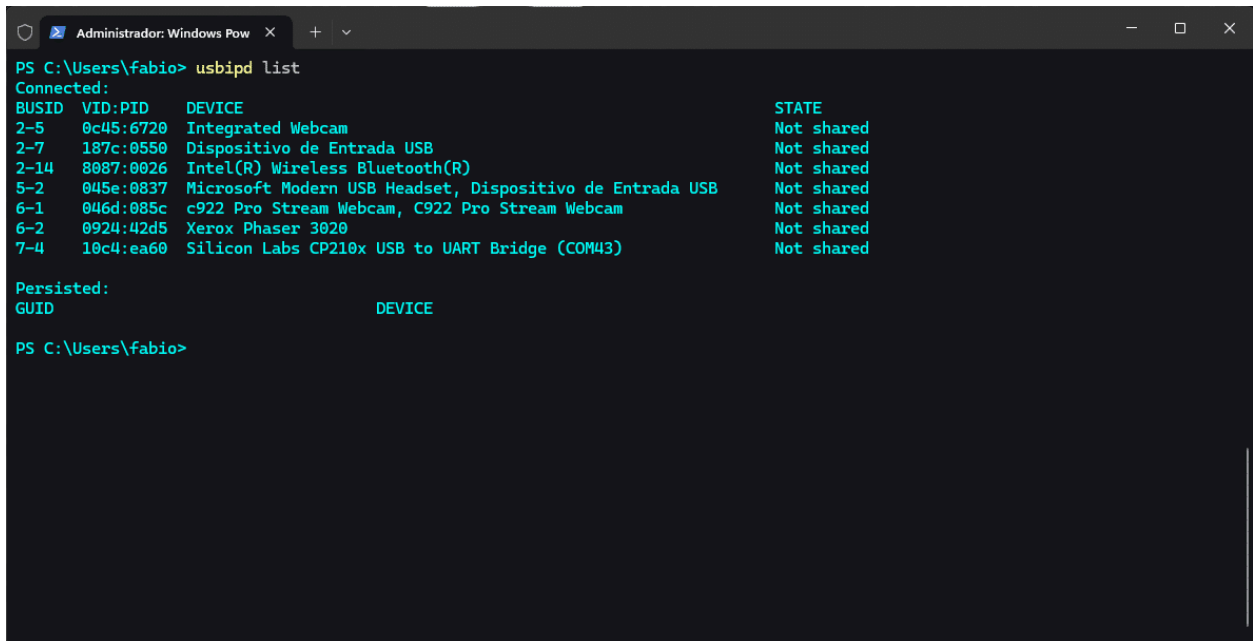
Instalação

Abra o PowerShell como Administrador e execute o seguinte comando para instalar o `usbipd-win`:

```
winget install usbipd
```

Para verificar os dispositivos USB disponíveis, use o comando:

```
usbipd list
```



```
PS C:\Users\fabio> usbipd list
Connected:
BUSID  VID:PID  DEVICE                                STATE
-----
2-5    0c45:6720 Integrated Webcam                     Not shared
2-7    187c:0550 Dispositivo de Entrada USB           Not shared
2-14   8087:0026 Intel(R) Wireless Bluetooth(R)        Not shared
5-2    045e:0837 Microsoft Modern USB Headset, Dispositivo de Entrada USB Not shared
6-1    046d:085c c922 Pro Stream Webcam, C922 Pro Stream Webcam Not shared
6-2    0924:42d5 Xerox Phaser 3020                     Not shared
7-4    10c4:ea60 Silicon Labs CP210x USB to UART Bridge (COM43) Not shared

Persisted:
GUID                                DEVICE
-----
PS C:\Users\fabio>
```

Para acessar o dispositivo desejado do Windows no WSL, o dispositivo deve estar vinculado com o `usbipd`. No PowerShell com direitos de administrador, digite o comando:

```
usbipd bind --busid <BUSID>
```

Academia ESP32 Profissional - Professor Fábio Souza

<https://cursos.embarcados.com.br/cursos/academia-esp32-profissional/>

Substitua <BUSID> pelo identificador do dispositivo listado no comando anterior (por exemplo, 7-4, no meu caso):

```
usbipd bind --busid 7-4
```

Nota: este comando precisa ser usado apenas uma vez, a menos que o computador tenha sido reiniciado.

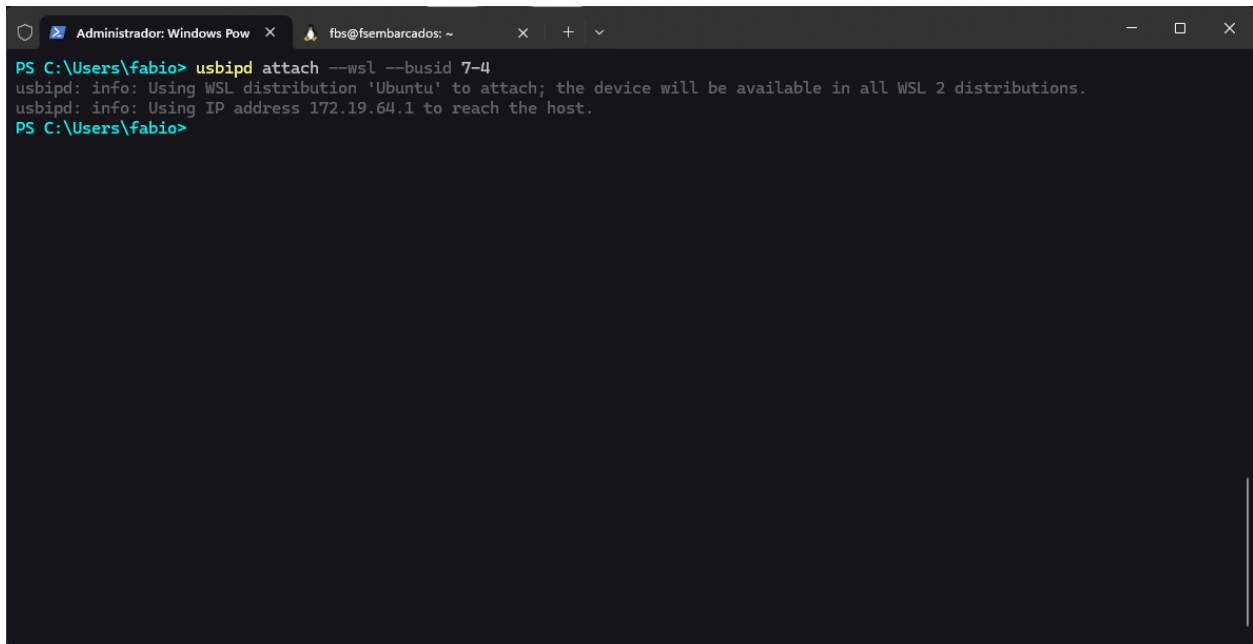
Após a vinculação, anexe o dispositivo especificado ao WSL com o comando abaixo, ainda no prompt de comando PowerShell:

```
usbipd attach --wsl --busid <BUSID>
```

No meu caso, ficou assim o comando:

Academia ESP32 Profissional - Professor Fábio Souza

<https://cursos.embarcados.com.br/cursos/academia-esp32-profissional/>



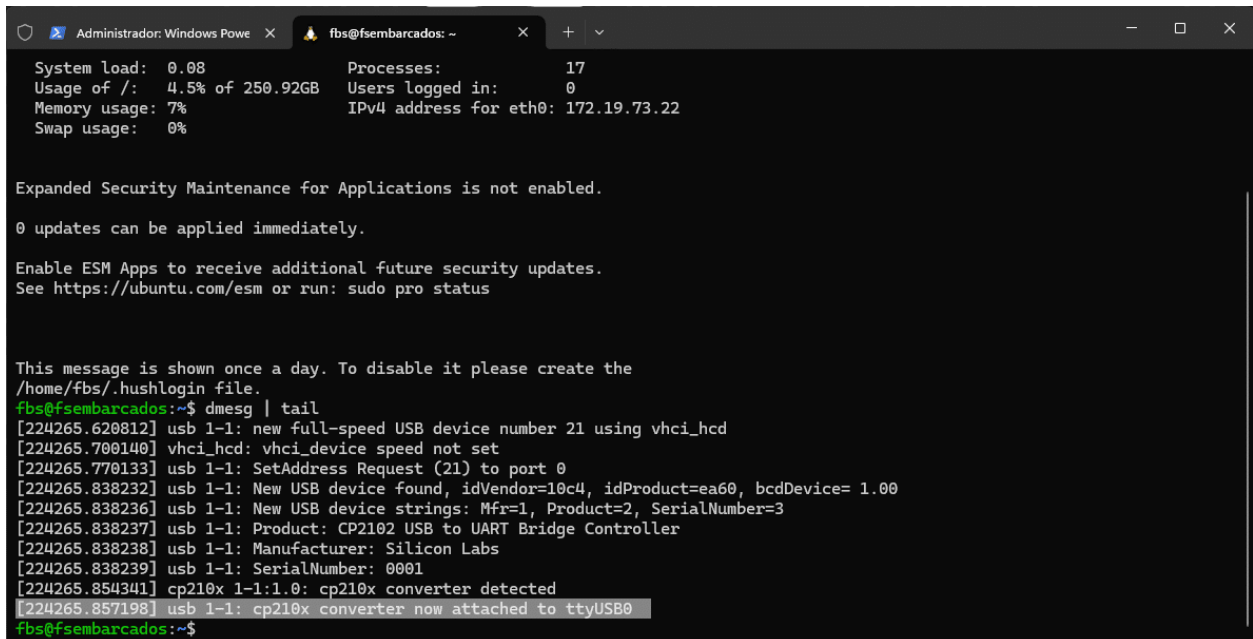
```
Administrador: Windows Pow  x  fbs@fsembarcados: ~  x  +  v
PS C:\Users\fabio> usbipd attach --wsl --busid 7-4
usbipd: info: Using WSL distribution 'Ubuntu' to attach; the device will be available in all WSL 2 distributions.
usbipd: info: Using IP address 172.19.64.1 to reach the host.
PS C:\Users\fabio>
```

Material de apoio ao aluno. Todos os direitos reservados. Proibida a reprodução total ou parcial, por qualquer meio, sem a devida autorização.

www.embarcados.com.br

Agora vamos verificar o dispositivo USB no WSL. No terminal do Ubuntu (WSL), verifique se o dispositivo USB está disponível:

```
dmesg | tail
```



```
System load: 0.08          Processes:           17
Usage of /:  4.5% of 250.92GB Users logged in:      0
Memory usage: 7%          IPv4 address for eth0: 172.19.73.22
Swap usage:  0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

This message is shown once a day. To disable it please create the
/home/fbs/.hushlogin file.
fbs@fsembarcados:~$ dmesg | tail
[224265.620812] usb 1-1: new full-speed USB device number 21 using vhci_hcd
[224265.700140] vhci_hcd: vhci_device speed not set
[224265.770133] usb 1-1: SetAddress Request (21) to port 0
[224265.838232] usb 1-1: New USB device found, idVendor=10c4, idProduct=ea60, bcdDevice= 1.00
[224265.838236] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[224265.838237] usb 1-1: Product: CP2102 USB to UART Bridge Controller
[224265.838238] usb 1-1: Manufacturer: Silicon Labs
[224265.838239] usb 1-1: SerialNumber: 0001
[224265.854341] cp210x 1-1:1.0: cp210x converter detected
[224265.857198] usb 1-1: cp210x converter now attached to ttyUSB0
fbs@fsembarcados:~$
```

Pronto, o dispositivo USB está pronto para ser acessado pelo Ubuntu.

Instalação do ESP-IDF no WSL

O processo de instalação do ESP-IDF no WSL segue o mesmo procedimento apresentado na documentação oficial da Espressif para instalação no Linux: [Standard Toolchain Setup for Linux and macOS – ESP32 – — ESP-IDF Programming Guide v5.3 documentation](#)

Academia ESP32 Profissional - Professor Fábio Souza

<https://cursos.embarcados.com.br/cursos/academia-esp32-profissional/>

Abra o Ubuntu. É importante que antes de iniciar a instalação, você atualize os pacotes do sistema:

```
sudo apt update  
sudo apt upgrade
```

Agora vamos instalar as dependências necessárias para o ESP-IDF:

```
sudo apt-get install git wget flex bison gperf python3  
python3-pip python3-venv cmake ninja-build ccache  
libffi-dev libssl-dev dfu-util libusb-1.0-0
```

Em seguida, vamos clonar o repositório do ESP-IDF:

```
mkdir -p ~/esp  
cd ~/esp  
git clone -b v5.2.2 --recursive  
https://github.com/espressif/esp-idf.git
```

Após o download, execute o script de instalação das ferramentas usadas pelo ESP-IDF, como o compilador, depurador, pacotes Python, etc:

```
cd ~/esp/esp-idf  
./install.sh
```

As ferramentas instaladas ainda não foram adicionadas à variável de ambiente PATH. Para tornar as ferramentas utilizáveis a partir da linha de

comando, algumas variáveis de ambiente devem ser configuradas. O ESP-IDF fornece outro script que faz isso. No terminal onde você vai usar o ESP-IDF, execute:

```
source $HOME/esp/esp-idf/export.sh
```

Este comando irá configurar as variáveis de ambiente necessárias para que as ferramentas do ESP-IDF sejam utilizáveis a partir da linha de comando.

Pronto, seu ambiente está preparado para desenvolver com o ESP-IDF. Vamos desenvolver nosso “Hello, World!” por linha de comandos no IDF para validar nossa instalação e configurações.

Exemplo de aplicação com ESP-IDF no WSL

Vamos aproveitar o exemplo “Hello World” já disponível na pasta exemplos dentro do ESP-IDF. Vamos fazer uma cópia do exemplo:

```
cd ~/esp  
cp -r $IDF_PATH/examples/get-started/hello_world .
```

Agora, entre no diretório `hello_world`, defina o ESP32 como alvo (ou outro SoC que esteja usando):

```
cd ~/esp/hello_world  
idf.py set-target esp32
```

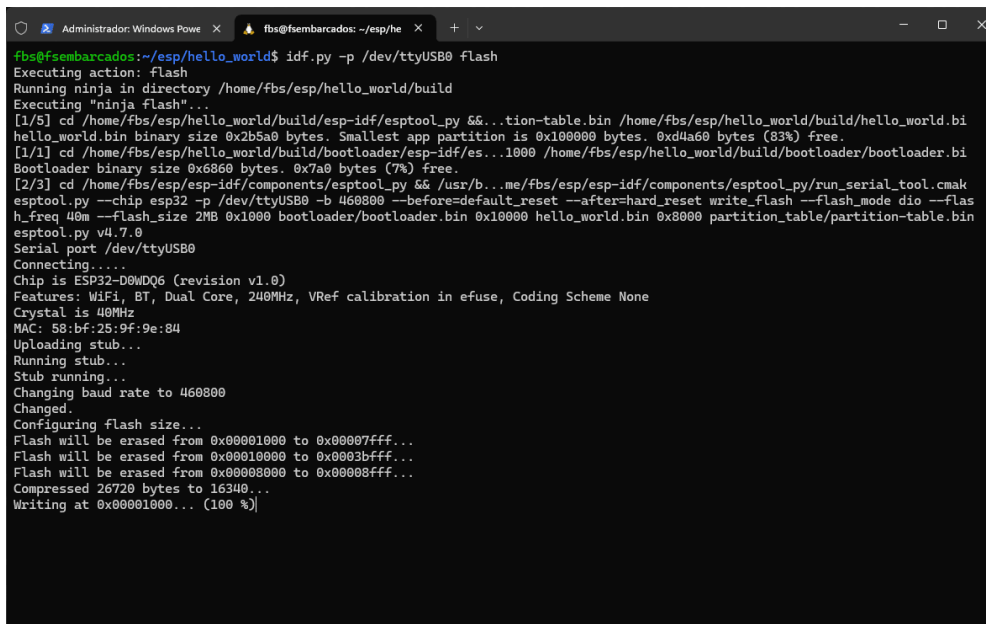
Agora é só fazer o build da aplicação:

```
idf.py build
```

Para gravar os binários que você acabou de construir para o ESP32, execute o seguinte comando:

```
idf.py -p PORT flash
```

Substitua **PORT** pelo nome da porta USB correspondente à sua placa ESP32 (por exemplo, **/dev/ttyUSB0**). Se a **PORT** não for definida, o **idf.py** tentará se conectar automaticamente usando as portas USB disponíveis. O meu ficou assim:



```
fbs@fsembarcados:~/esp/hello_world$ idf.py -p /dev/ttyUSB0 flash
Executing action: flash
Running ninja in directory /home/fbs/esp/hello_world/build
Executing "ninja flash"...
[1/5] cd /home/fbs/esp/hello_world/build/esp-idf/esptool.py && ...tion-table.bin /home/fbs/esp/hello_world/build/hello_world.bi
hello_world.bin binary size 0x2b5a0 bytes. Smallest app partition is 0x100000 bytes. 0xd4a60 bytes (83%) free.
[1/1] cd /home/fbs/esp/hello_world/build/bootloader/esp-idf/es...1000 /home/fbs/esp/hello_world/build/bootloader/bootloader.bi
Bootloader binary size 0x6860 bytes. 0x7a0 bytes (7%) free.
[2/3] cd /home/fbs/esp/esp-idf/components/esptool.py && /usr/b...me/fbs/esp/esp-idf/components/esptool.py/run_serial_tool.cmak
esptool.py --chip esp32 -p /dev/ttyUSB0 -b 460800 --before=default_reset --after=hard_reset write_flash --flash_mode dio --flas
h_freq 40m --flash_size 2MB 0x1000 bootloader/bootloader.bin 0x10000 hello_world.bin 0x8000 partition_table/partition-table.bin
esptool.py v4.7.0
Serial port /dev/ttyUSB0
Connecting....
Chip is ESP32-D0WD06 (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 58:bf:25:9f:9e:84
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00001000 to 0x00007fff...
Flash will be erased from 0x00010000 to 0x0003bfff...
Flash will be erased from 0x00008000 to 0x00008fff...
Compressed 26720 bytes to 16340...
Writing at 0x00001000... (100 %)
```

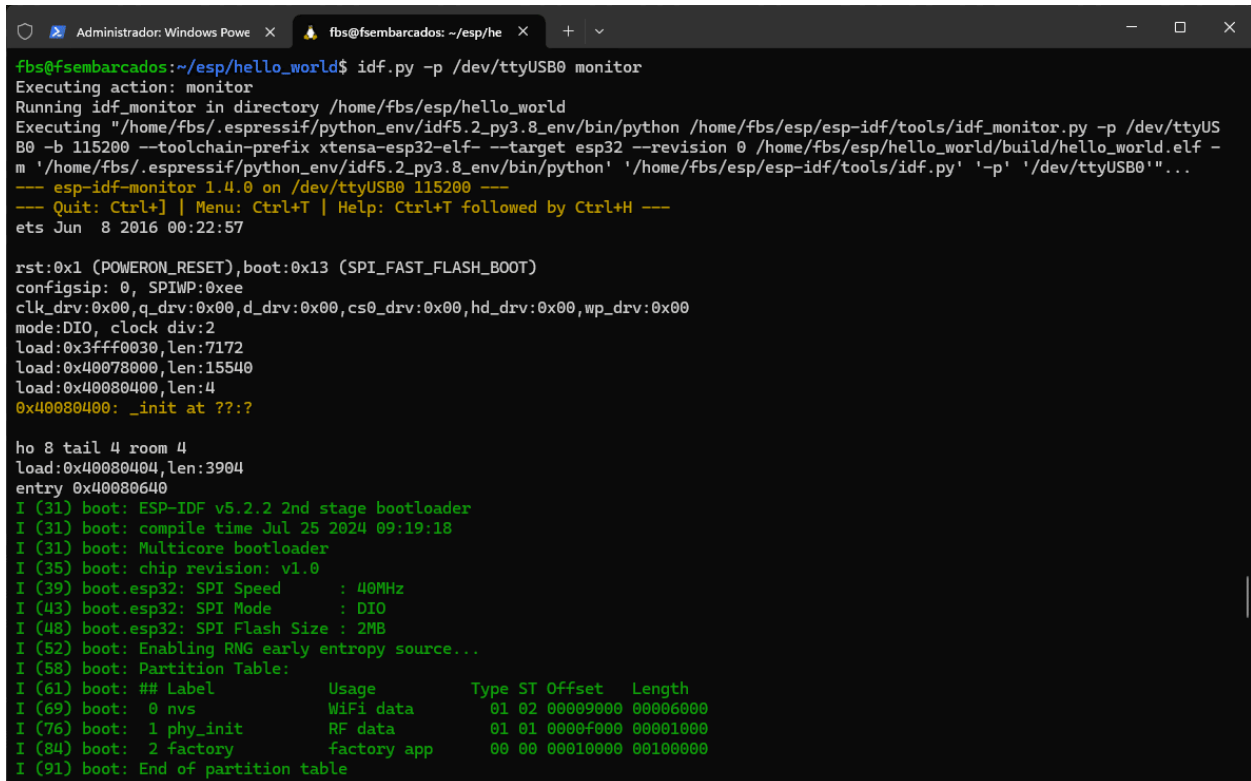
Academia ESP32 Profissional - Professor Fábio Souza

<https://cursos.embarcados.com.br/cursos/academia-esp32-profissional/>

Para verificar o funcionamento do “hello_world”, abra o monitor serial do IDF. Digite o seguinte comando (não se esqueça de substituir **PORT** pelo nome da sua porta serial):

```
idf.py -p PORT monitor
```

Este comando iniciará o monitor serial para que você possa ver a saída do seu programa “hello_world” na placa ESP32.



```
fbs@fsembarcados:~/esp/hello_world$ idf.py -p /dev/ttyUSB0 monitor
Executing action: monitor
Running idf_monitor in directory /home/fbs/esp/hello_world
Executing "/home/fbs/.espressif/python_env/idf5.2_py3.8_env/bin/python /home/fbs/esp/esp-idf/tools/idf_monitor.py -p /dev/ttyUSB0 -b 115200 --toolchain-prefix xtensa-esp32-elf --target esp32 --revision 0 /home/fbs/esp/hello_world/build/hello_world.elf -m '/home/fbs/.espressif/python_env/idf5.2_py3.8_env/bin/python' '/home/fbs/esp/esp-idf/tools/idf.py' '-p' '/dev/ttyUSB0'..."
--- esp-idf-monitor 1.4.0 on /dev/ttyUSB0 115200 ---
--- Quit: Ctrl+] | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:7172
load:0x40078000,len:15540
load:0x40080400,len:4
0x40080400: _init at ???:?

ho 8 tail 4 room 4
load:0x40080404,len:3904
entry 0x40080640
I (31) boot: ESP-IDF v5.2.2 2nd stage bootloader
I (31) boot: compile time Jul 25 2024 09:19:18
I (31) boot: Multicore bootloader
I (35) boot: chip revision: v1.0
I (39) boot.esp32: SPI Speed      : 40MHz
I (43) boot.esp32: SPI Mode      : DIO
I (48) boot.esp32: SPI Flash Size : 2MB
I (52) boot: Enabling RNG early entropy source...
I (58) boot: Partition Table:
I (61) boot: ## Label                Usage              Type            ST Offset   Length
I (69) boot:  0 nvs                   WiFi data          01 02 00009000 00006000
I (76) boot:  1 phy_init                 RF data            01 01 0000f000 00001000
I (84) boot:  2 factory                 factory app         00 00 00010000 00100000
I (91) boot: End of partition table
```

Será exibido o LOG da aplicação no console. Para sair do monitor IDF, use o atalho **Ctrl+]**.

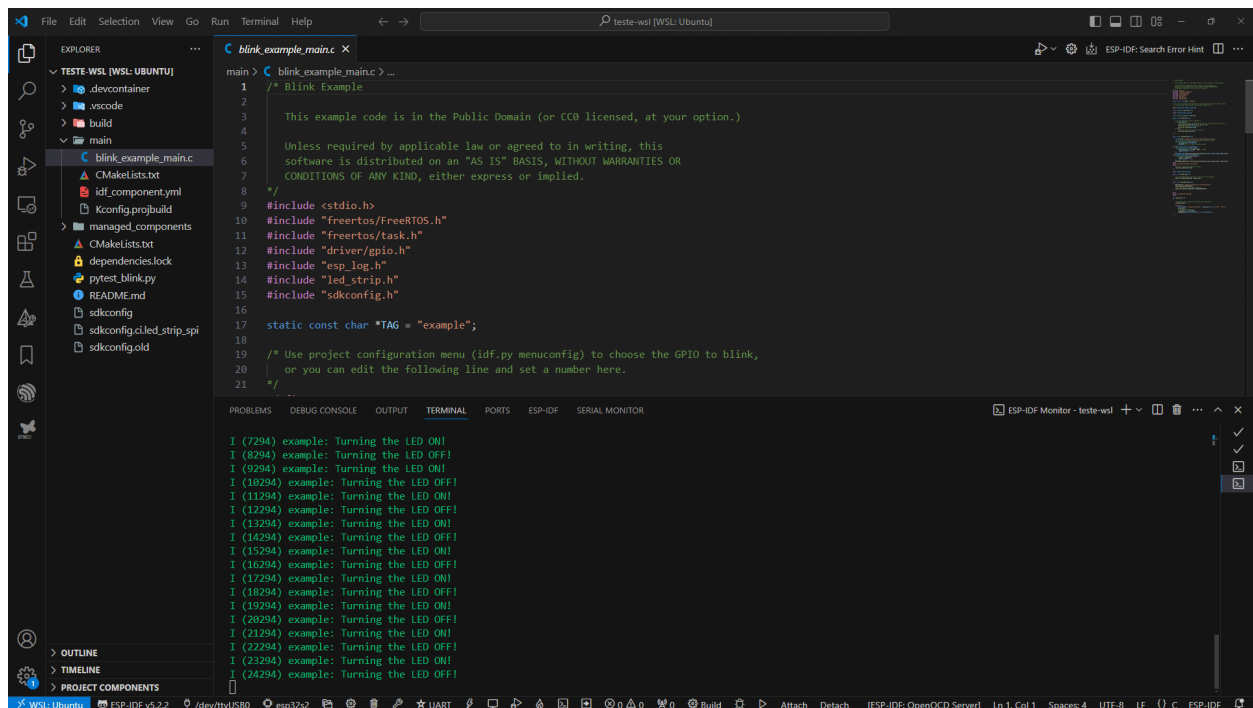
Academia ESP32 Profissional - Professor Fábio Souza

<https://cursos.embarcados.com.br/cursos/academia-esp32-profissional/>

Maravilha! Se você conseguiu executar todos os passos até aqui, o seu WSL está pronto para ser usado como ambiente de desenvolvimento para o ESP-IDF.

Agora é com você.

Dica: Você também pode usar o VSCode para acessar o WSL e fazer o desenvolvimento usando os recursos da Extensão do IDF para VSCode.



```
main > C:\blink_example_main.c > ...
1  /* Blink example
2
3  This example code is in the Public Domain (or CC0 licensed, at your option.)
4
5  Unless required by applicable law or agreed to in writing, this
6  software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
7  CONDITIONS OF ANY KIND, either express or implied.
8
9  */
10 #include <stdio.h>
11 #include "freertos/FreeRTOS.h"
12 #include "freertos/task.h"
13 #include "driver/gpio.h"
14 #include "esp_log.h"
15 #include "led_strip.h"
16 #include "sdkconfig.h"
17
18 static const char *TAG = "example";
19
20 /* Use project configuration menu (idf.py menuconfig) to choose the GPIO to blink,
21    or you can edit the following line and set a number here.
22 */
```

```
I (7294) example: Turning the LED ON!
I (8294) example: Turning the LED OFF!
I (9294) example: Turning the LED ON!
I (10294) example: Turning the LED OFF!
I (11294) example: Turning the LED ON!
I (12294) example: Turning the LED OFF!
I (13294) example: Turning the LED ON!
I (14294) example: Turning the LED OFF!
I (15294) example: Turning the LED ON!
I (16294) example: Turning the LED OFF!
I (17294) example: Turning the LED ON!
I (18294) example: Turning the LED OFF!
I (19294) example: Turning the LED ON!
I (20294) example: Turning the LED OFF!
I (21294) example: Turning the LED ON!
I (22294) example: Turning the LED OFF!
I (23294) example: Turning the LED ON!
I (24294) example: Turning the LED OFF!
```

Material de apoio ao aluno. Todos os direitos reservados. Proibida a reprodução total ou parcial, por qualquer meio, sem a devida autorização.

www.embarcados.com.br