PADO

# Labs


# Microcontroladores Aplicados a IoT

Dilson Liukiti Ito


# 02 – ESP32: GPIO

**Microcontroladores Aplicados a IoT**

# Estabelecendo comunicação serial


- Checar a porta serial no Linux

  $ ls /dev/ttyUSB*

- Quando há um dispositivo conectado

```
dilson@dilson-VirtualBox:~$ ls /dev/ttyUSB*
ls: não foi possível acessar '/dev/ttyUSB*': Arquivo ou diretório inexistente
dilson@dilson-VirtualBox:~$
```

- Quando não há dispositivo serial conectado

```
dilson@dilson-VirtualBox:~$ ls /dev/ttyUSB*
/dev/ttyUSB0
dilson@dilson-VirtualBox:~$
```

# Erro de permissão na serial


```
0x8000 partition_table/partition-table.bin 0x1000 bootloader/bootloader.bin 0x10000 main.bin
esptool.py v3.3-dev
Serial port /dev/ttyUSB0
Traceback (most recent call last):
  File "/home/dilson/.espressif/python_env/idf4.3_py3.8_env/lib/python3.8/site-packages/serial/serialposix.py", line 322, in open
    self.fd = os.open(self.portstr, os.O_RDWR | os.O_NOCTTY | os.O_NONBLOCK)
PermissionError: [Errno 13] Permission denied: '/dev/ttyUSB0'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py", line 5347, in <module>
    _main()
  File "/home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py", line 5340, in _main
    main()
  File "/home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py", line 4647, in main
    esp = esp or get_default_connected_device(ser_list, port=args.port, connect_attempts=args.connect_attempts,
  File "/home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py", line 114, in get_default_connected_device
    _esp = chip_class(each_port, initial_baud, trace)
  File "/home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py", line 320, in __init__
    self._port = serial.serial_for_url(port)
  File "/home/dilson/.espressif/python_env/idf4.3_py3.8_env/lib/python3.8/site-packages/serial/__init__.py", line 90, in serial_for_url
    instance.open()
  File "/home/dilson/.espressif/python_env/idf4.3_py3.8_env/lib/python3.8/site-packages/serial/serialposix.py", line 325, in open
    raise SerialException(msg.errno, "could not open port {}: {}".format(self._port, msg))
serial.serialutil.SerialException: [Errno 13] could not open port /dev/ttyUSB0: [Errno 13] Permission denied: '/dev/ttyUSB0'
CMake Error at run_serial_tool.cmake:50 (message):
  /home/dilson/.espressif/python_env/idf4.3_py3.8_env/bin/python
  /home/dilson/Documentos/PadoLabs/ESP32/esp-idf/components/esptool_py/esptool/esptool.py
  --chip esp32 failed
```

# Acesso à comunicação serial


- Verifique se seu usuário está no grupo dialout

$ id

```
dilson@dilson-VirtualBox:~$ id
uid=1000(dilson) gid=1000(dilson) grupos=1000(dilson),4(adm),20(dialout),24(cdro
m),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare)
dilson@dilson-VirtualBox:~$
```

- Se não estiver, adicione seu usuário ao grupo dialout

$ sudo usermod -a -G dialout $USER

$ sudo adduser $USER dialout

- Faça o logout e o login para ter acessos de escrita na porta serial

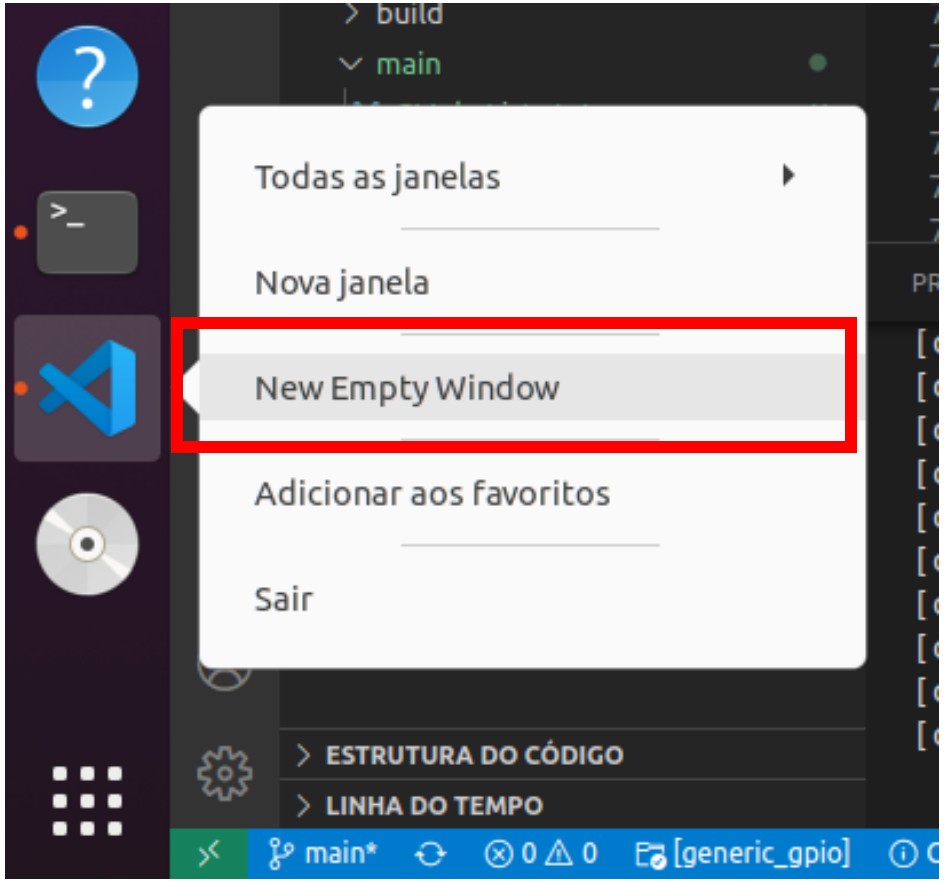
https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/establish-serial-connection.html

https://askubuntu.com/questions/112568/how-do-i-allow-a-non-default-user-to-use-serial-device-ttyusb0

# Resetar "view" do VSCode

- Clique com o botão direito do mouse sobre o ícone do VSCode na barra de tarefas.

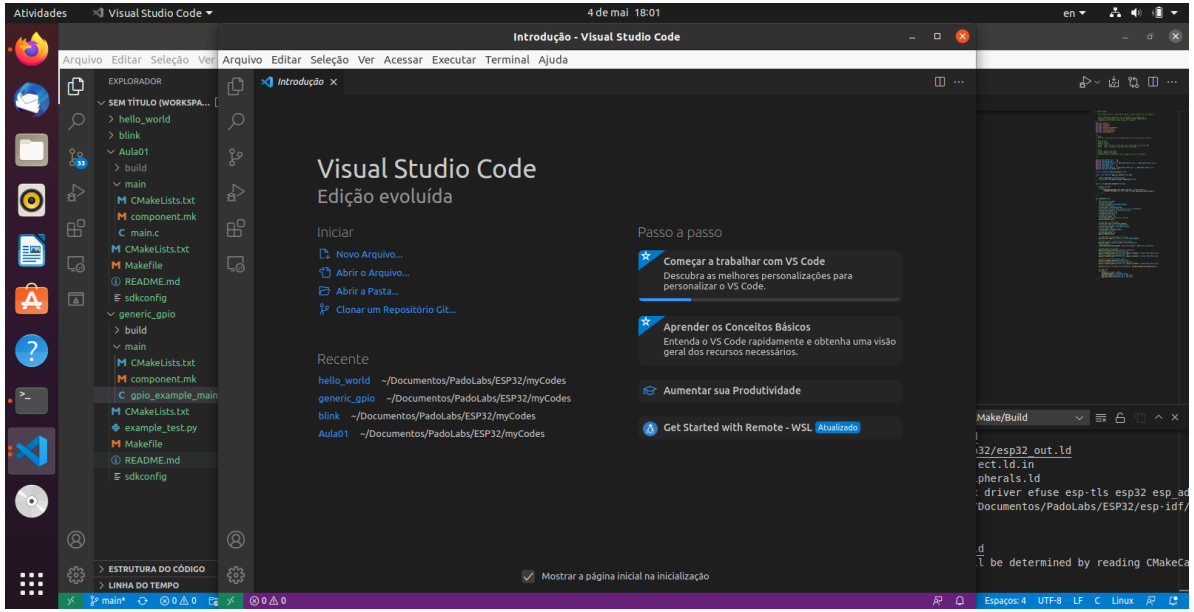
- Na janela que se abre, clique em "New Empty Window"



# Resetar "view" do VSCode

- Uma nova instância do VSCode irá se iniciar (com a página Introdução aberta)

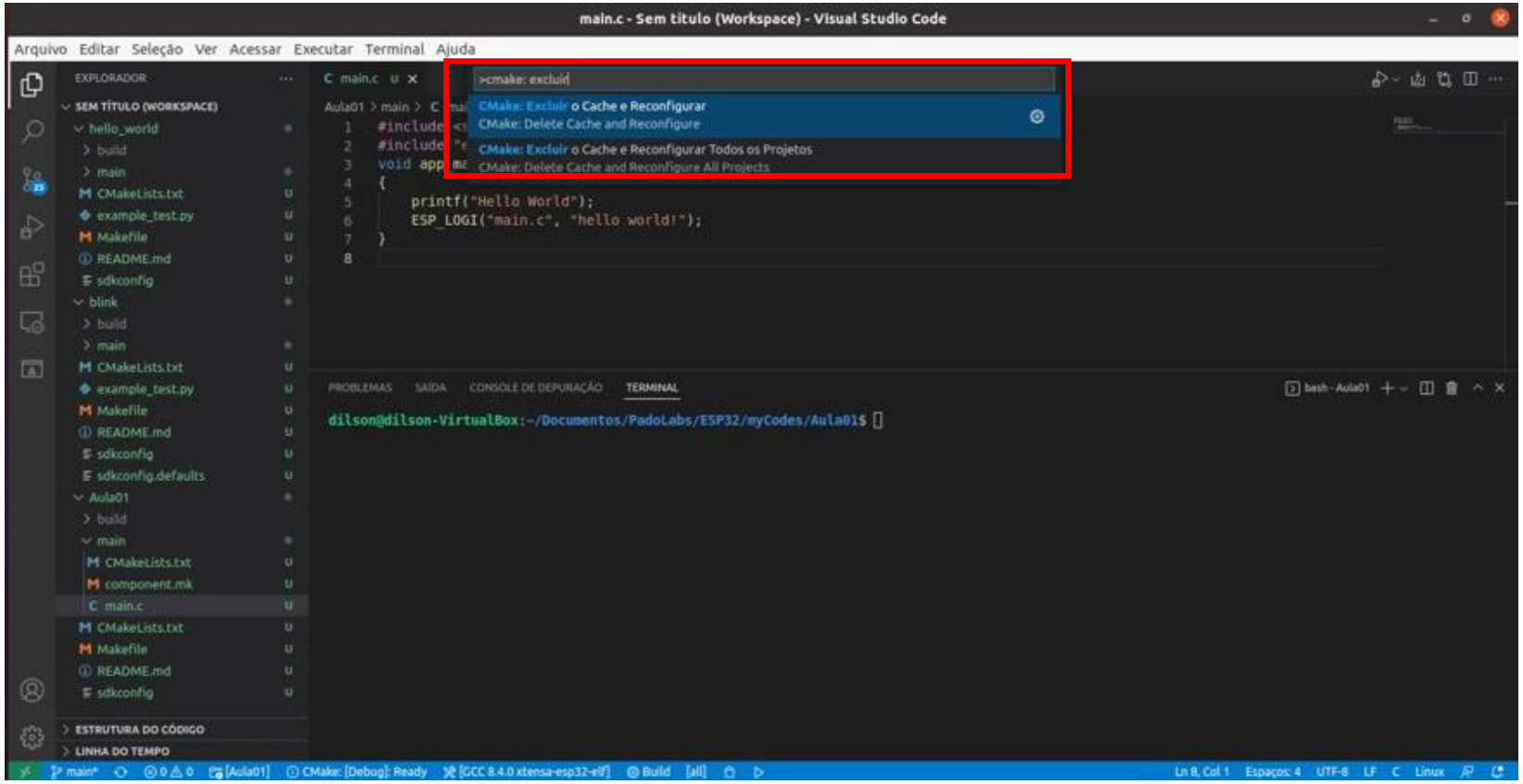
- Feche as outras janelas do VSCode, deixando a nova instância por último.




- A próxima vez que executar o arquivo edit.desktop o VSCode abrirá em uma "instância limpa"

# CMake: excluir e reconfigurar

- Ctrl + Shift + P
- Digite "cmake: Excluir"
- Na lista que aparece, clique em "CMake: Excluir o Cache e Reconfigurar"



# idf.py


- Deleta todo o diretório build

  $ idf.py fullclean

# idf.py


- Deve ser executado dentro de um diretório de "projeto", que contenha um arquivo "CMakeLists.txt"
- Para fazer o build do projeto $ idf.py build $ idf.py all
- Para gravar no ESP32 $ idf.py flash –p /dev/ttyUSB0
- Para estabelecer uma comunicação serial

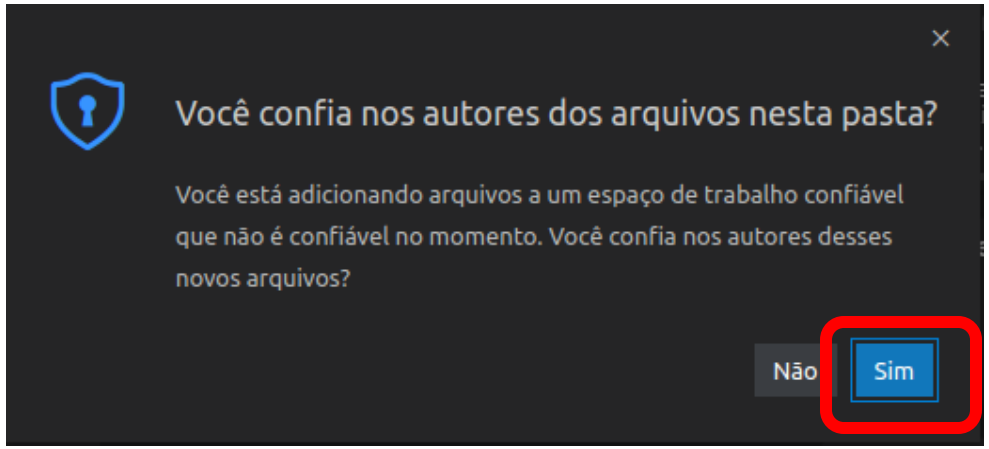
  $ idf.py monitor –p /dev/ttyUSB0
- Para finalizar a comunicação serial Ctrl + ]

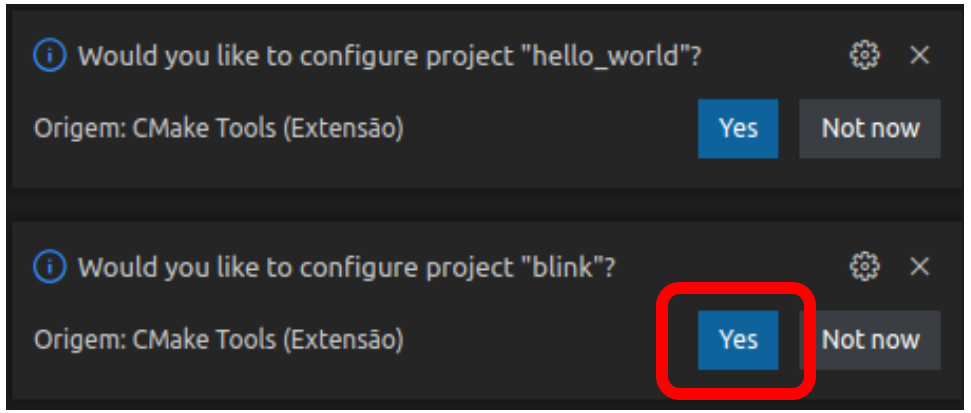
# Exemplo

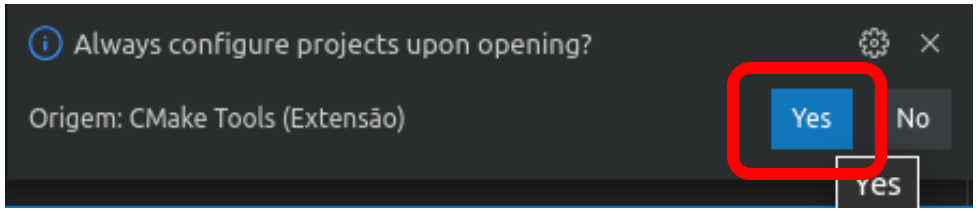
- Navegue até /ESP32/esp-idf/examples/get-started/
- Copie a pasta blink e sample_project para o ESP32/myCodes/
- Renomeie a pasta sample_project em ESP32/myCodes/ para Aula02
- Navegue até /ESP32/
- Execute no terminal

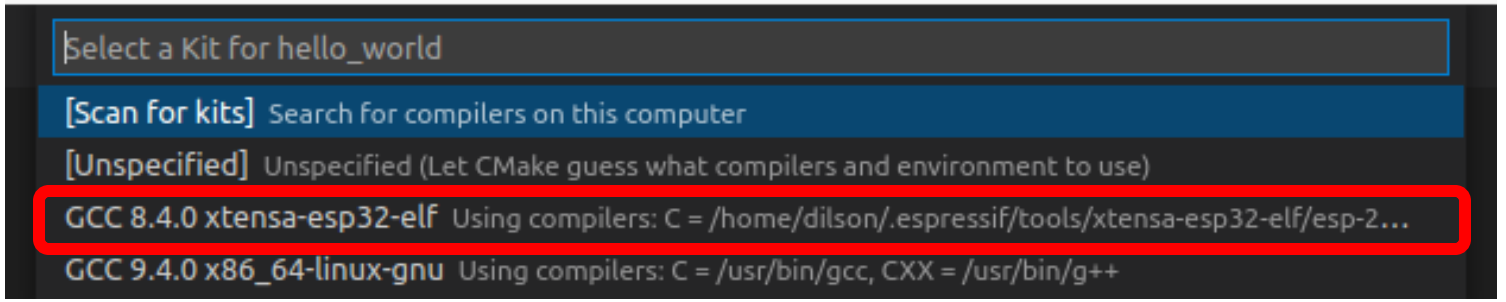
```
$ dex edit.desktop
```

# Avisos ao abrir o VSCode

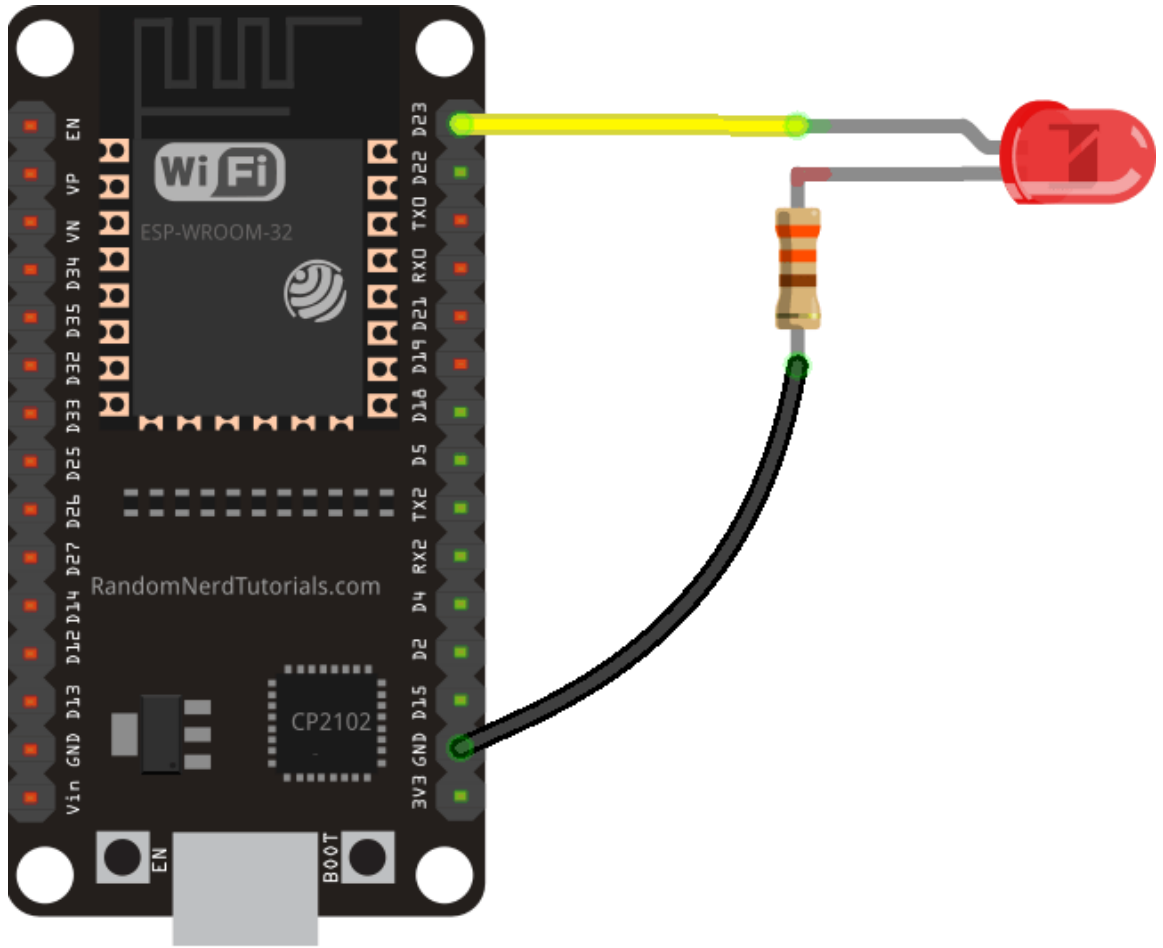


# Monte o circuito abaixo




https://www.dev.faranux.com/wp-content/uploads/2018/06/83211781-blinkin_led_schematic_bb.png

# No VSCode


- Abra um terminal para o projeto blink
- Digite

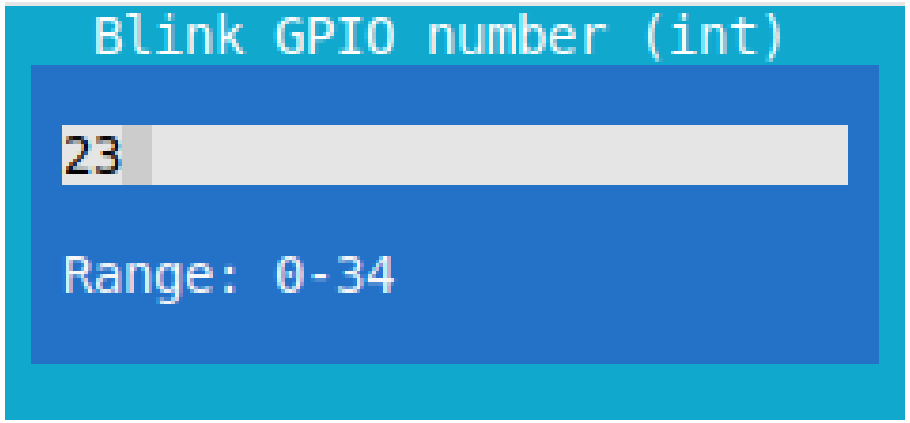
```
$ idf.py menuconfig
```

# idf.py menuconfig


```
(Top)
                Espressif IoT Development Framework Configuration
    SDK tool configuration  --->
    Build type  --->
    Application manager  --->
    Bootloader config  --->
    Security features  --->
    Serial flasher config  --->
    Partition Table  --->
    Example Configuration  --->
    Compiler options  --->
    Component config  --->
    Compatibility options  --->

[Space/Enter] Toggle/enter  [ESC] Leave menu           [S] Save
[O] Load                    [?] Symbol info           [/] Jump to symbol
[F] Toggle show-help mode   [C] Toggle show-name mode [A] Toggle show-all mode
[Q] Quit (prompts for save) [D] Save minimal config (advanced)
```

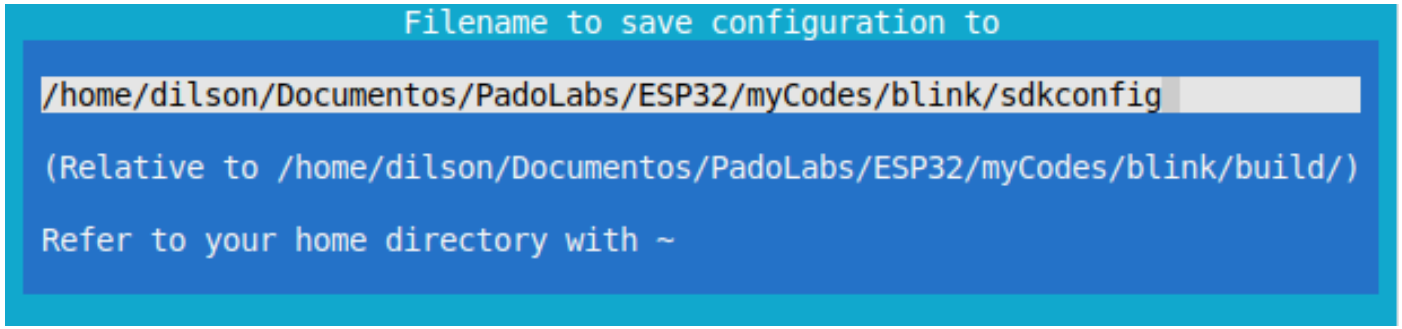
# idf.py


- Navegue até Example Configuration / Blink GPIO number;
- Altere o número para 23;
- Tecle Enter;



# idf.py

- Tecle "S" para salvar. Uma nova janela irá aparecer




- Confirme com Enter;


- Tecle ESC até sair e retornar ao terminal;

# No VSCode abra o projeto blink


- Execute no terminal do VSCode no projeto blink:

```
$ idf.py build
```

- Conecte o ESP32 ao USB do PC;
- Faça o upload e inicie a comunicação com o kit ESP32

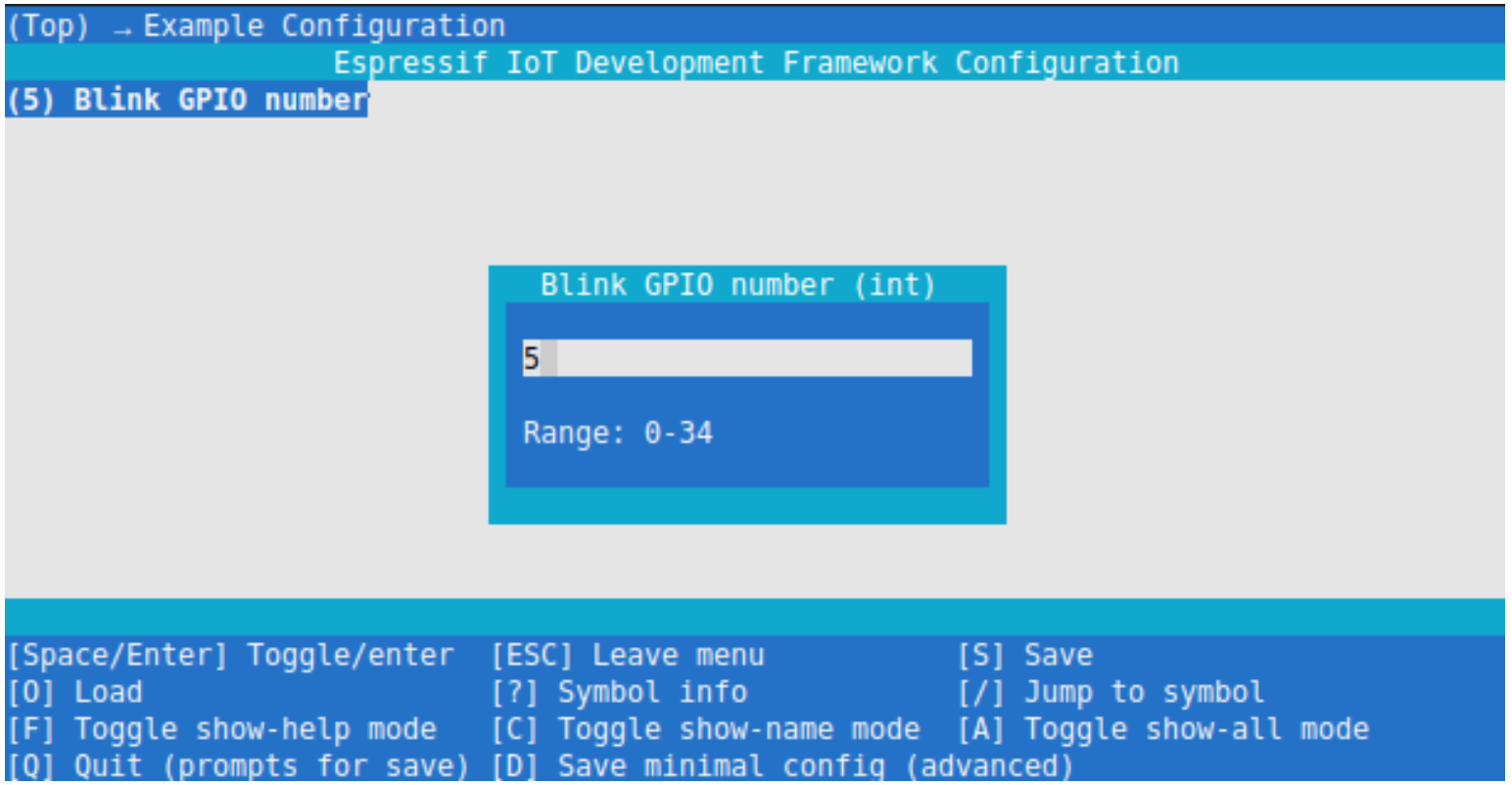
```
$ idf.py flash monitor
```

# Kconfig

- Setar configurações do projeto

- https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/kconfig.html

```
menu "Example Configuration"
  config BLINK_GPIO
    int "Blink GPIO number"
    range 0 34
    default 5
    help
     GPIO number (IOxx) to blink on and off.

     Some GPIOs are used for other purposes
    (flash connections, etc.) and cannot be
     used to blink.

     GPIOs 35-39 are input-only so cannot
     be used as outputs.

endmenu
```



# Kconfig (cont.)


```
menu "Example Configuration"
  config BLINK_GPIO
    int "Blink GPIO number"
    range 0 34
    default 5
    help
     GPIO number (IOxx) to blink on and off.

      Some GPIOs are used for other purposes
     (flash connections, etc.) and cannot be
      used to blink.

      GPIOs 35-39 are input-only so cannot
      be used as outputs.

endmenu
```

```
                                Symbol information
Name: BLINK_GPIO
Prompt: Blink GPIO number
Type: int
Value: 5

Help:

  GPIO number (IOxx) to blink on and off.

  Some GPIOs are used for other purposes (flash connections, etc.) and cannot be used t

  GPIOs 35-39 are input-only so cannot be used as outputs.

Default:
  - 5

Kconfig definition, with parent deps. propagated to 'depends on'
=================================================================
    ↓↓↓↓↓↓↓↓↓↓↓↓↓↓↓
[ESC/q] Return to menu      [/] Jump to symbol
```

# .clang-format


- Aplica uma formatação em todo o código
- Ctrl + Shift + I

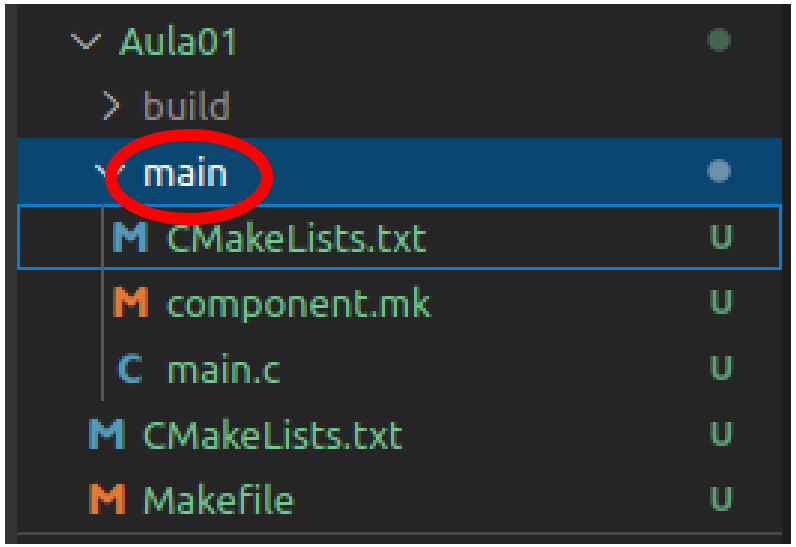
```
{
  BasedOnStyle: LLVM,
  IndentWidth: 4,
  TabWidth: 4,
  UseTab: Never,
  BreakBeforeBraces: Allman,
  AllowShortIfStatementsOnASingleLine: false,
  AllowShortFunctionsOnASingleLine: false,
  AllowShortBlocksOnASingleLine: true,
  IndentCaseLabels: false,
  ColumnLimit: 80,
  AccessModifierOffset: -4,
  SortIncludes: false,
}
```

# Estrutura de um projeto

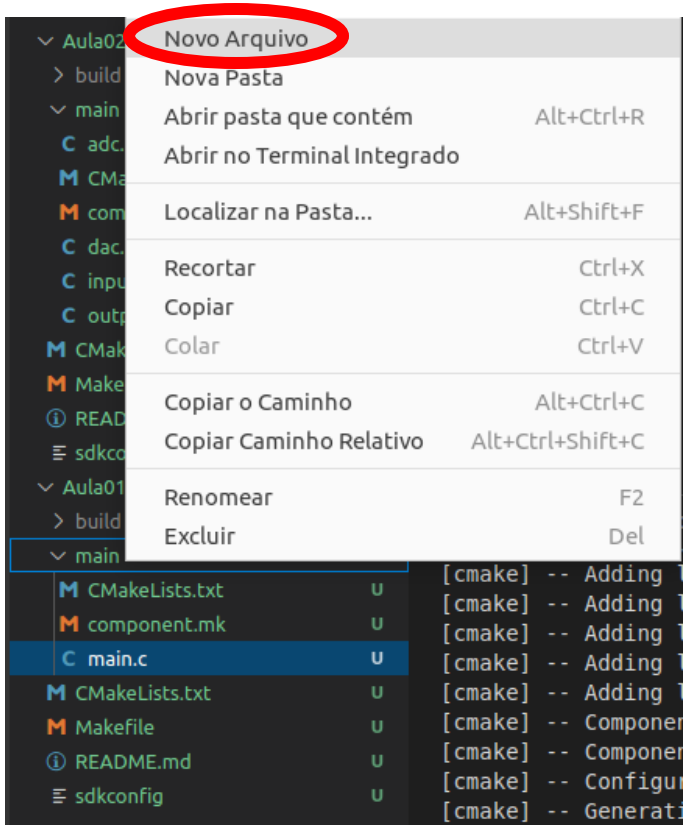

- Dois arquivos CMakeLists.txt
  - Um dentro do diretório raiz do projeto
  - Outro dentro dos arquivos de código ("main")
- Pasta build
  - Nomedoprojeto.bin
- Pasta dos arquivos de código ("main")
- Arquivo sdkconfig
- Arquivo KConfig
- Arquivos mk / Makefile

# Hello World

- Clique com o botão direito sobre "main"

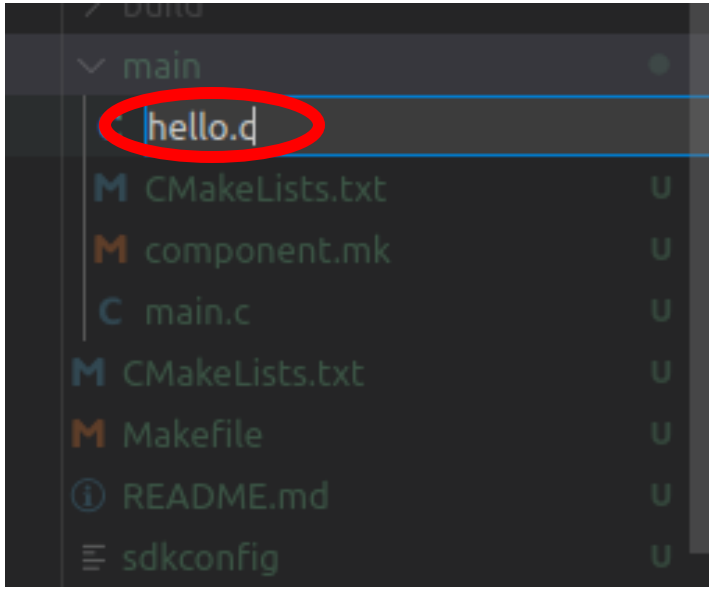


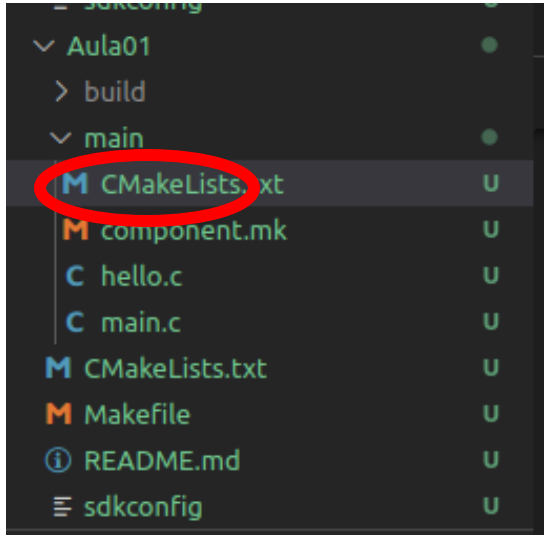

- No menu que surge, clique em Novo Arquivo



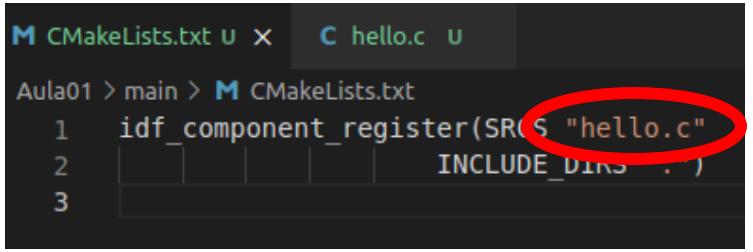
# Hello World (cont.)

- Digite hello.c como nome do arquivo




- Edite o arquivo CMakeLists.txt de dentro do diretório "main"

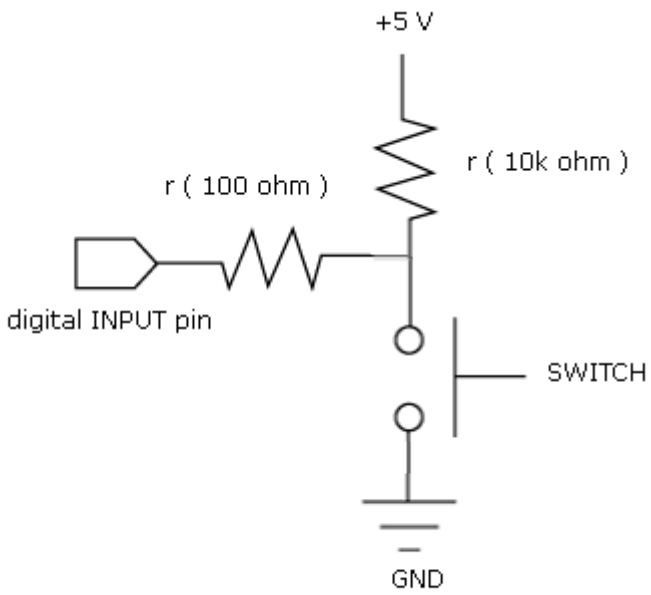


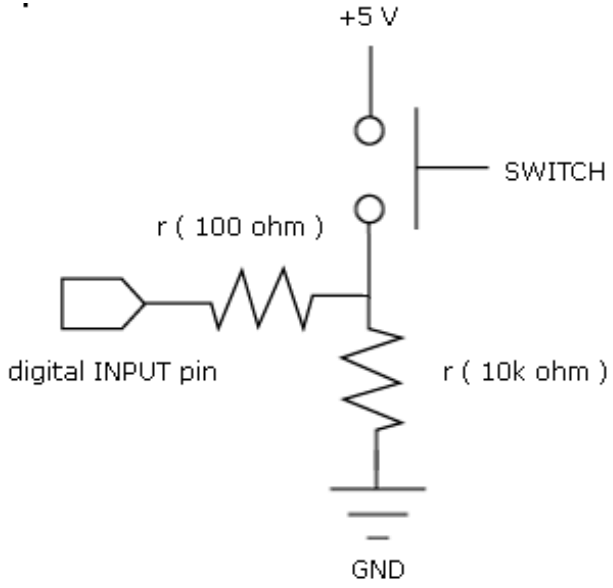
# hello.c


```c
#include <stdio.h>
#include "esp_log.h"
void app_main(void)
{
    printf("Hello World\n");
    ESP_LOGI("hello.c",  "hello world!");
}
```

# GPIO

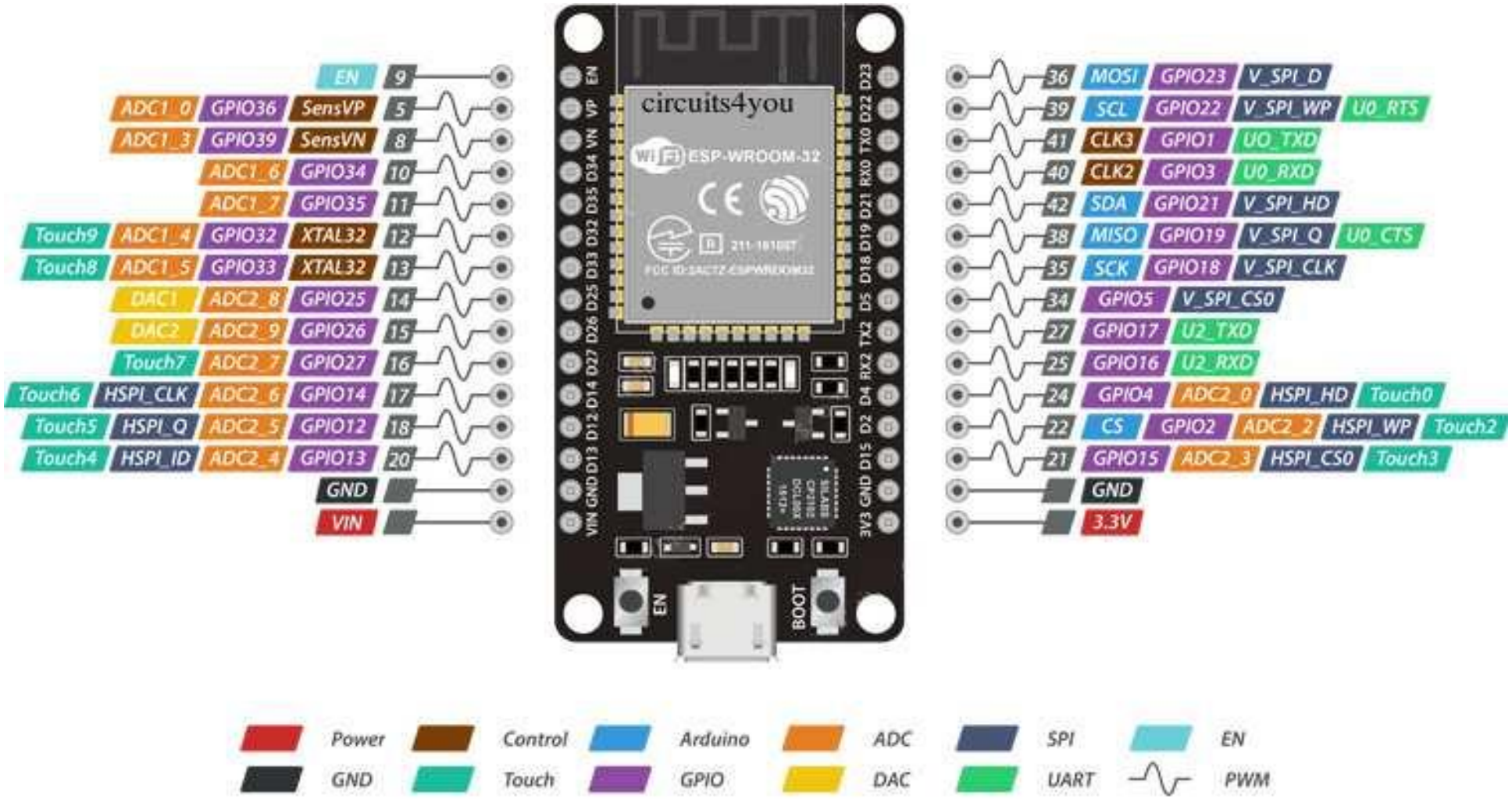
- General Purpose Input Output
- Entrada e saída
- Meio por onde o chip se comunica com o mundo externo
- Pull-up, pull-down;



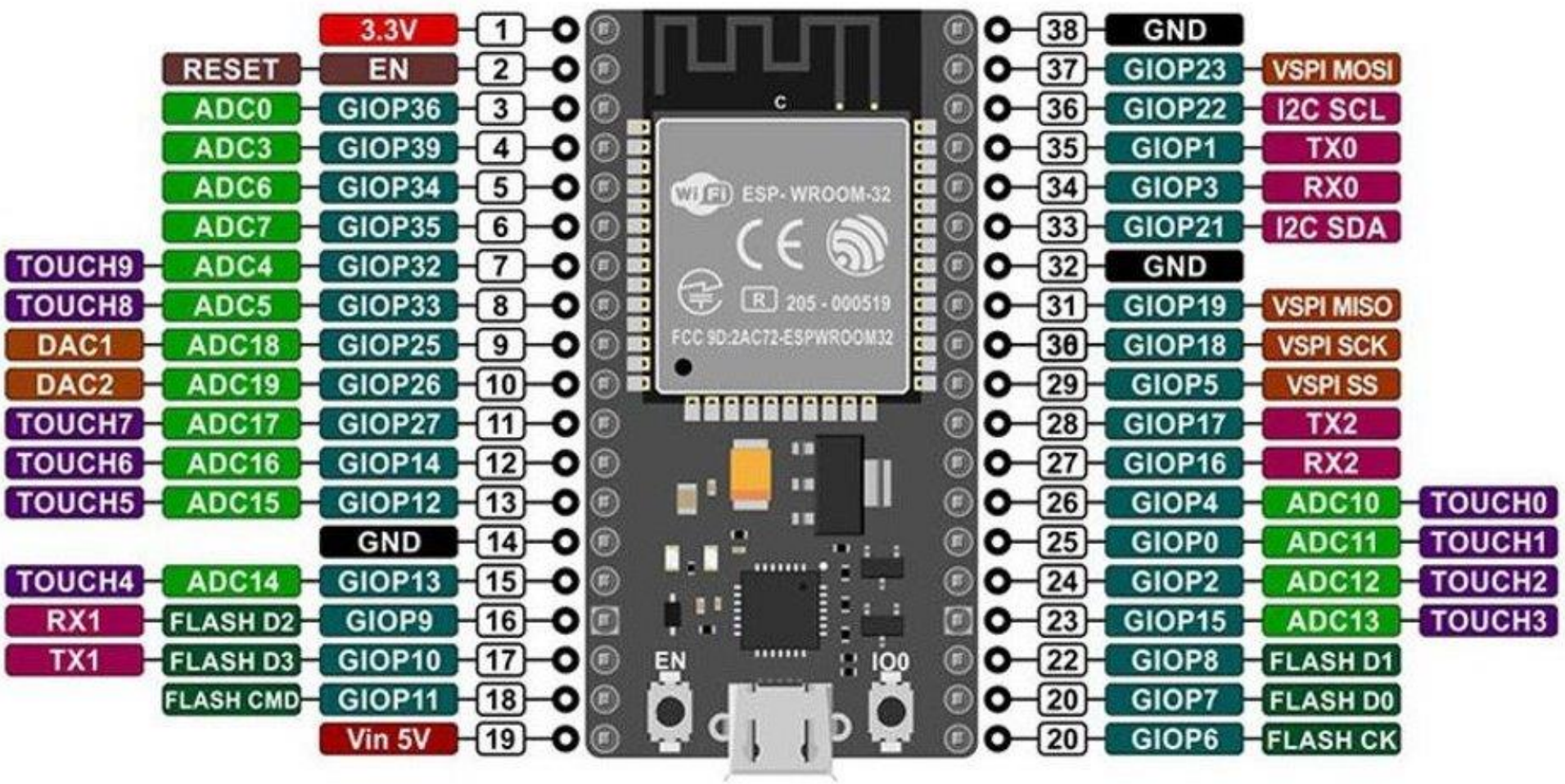

https://i.stack.imgur.com/gQcoz.png

# Kit 30 pinos




https://circuits4you.com/wp-content/uploads/2018/12/ESP32-Pinout.jpg

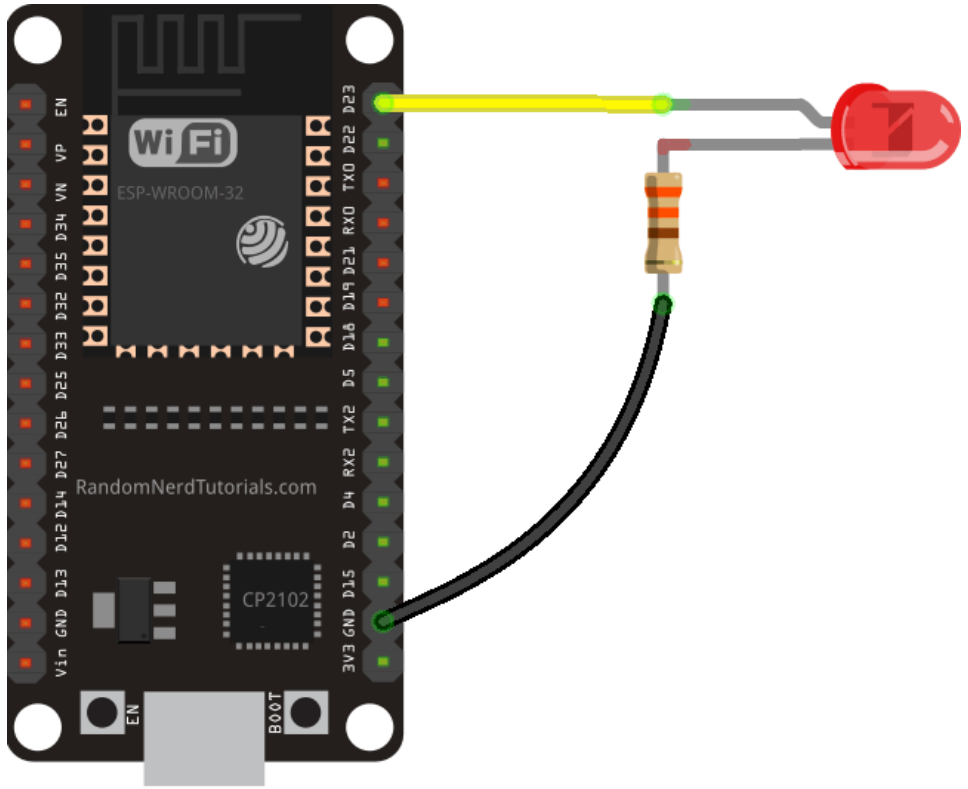
# Kit 38 pinos




https://miro.medium.com/max/1400/1*OpXj1n3p2z0adtvKLJF-Mg.jpeg

# Pino como saída

- Não exceder 20mA por pino!



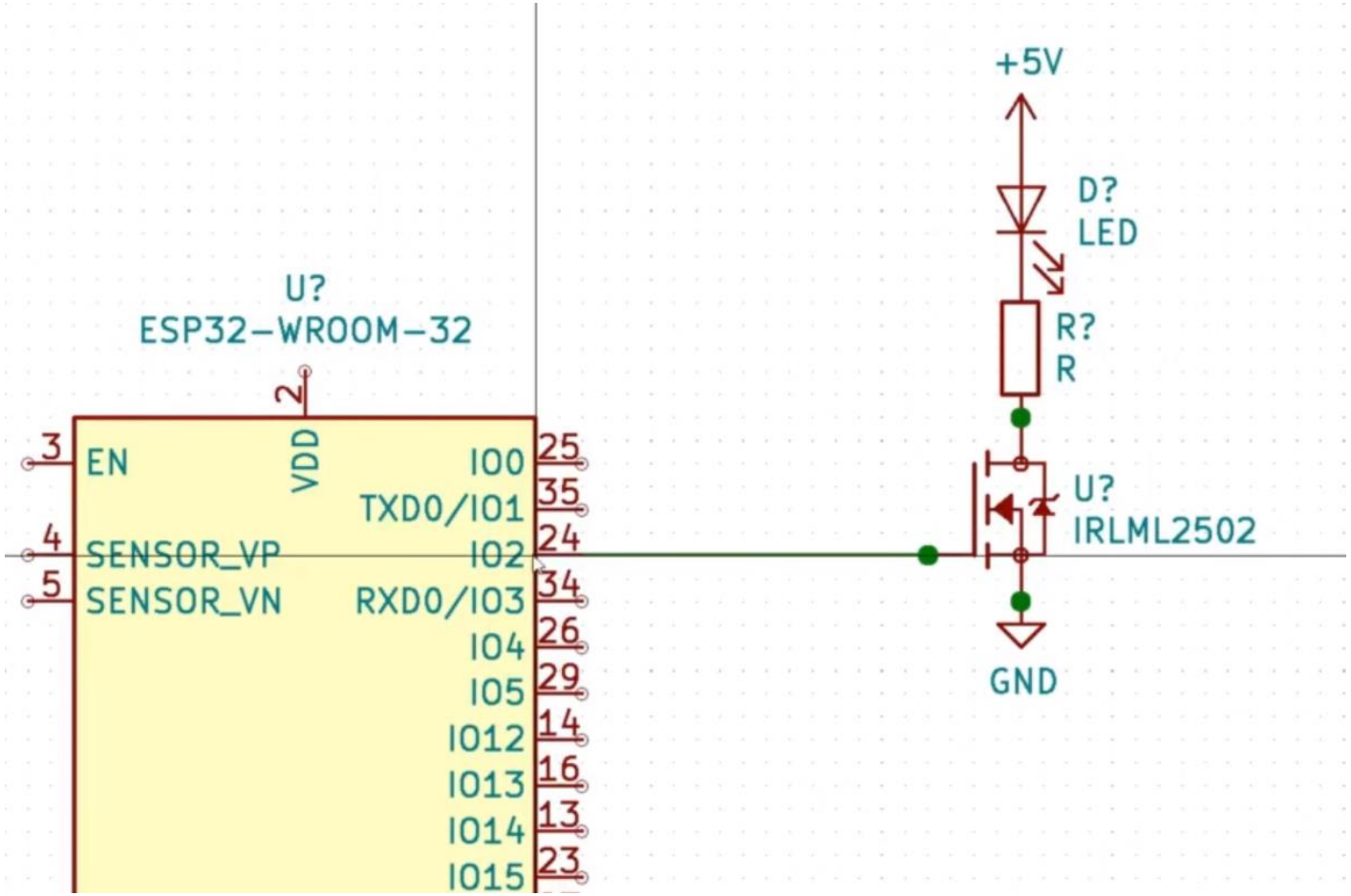

GPIO23

GND

https://www.dev.faranux.com/wp-content/uploads/2018/06/83211781-blinkin_led_schematic_bb.png

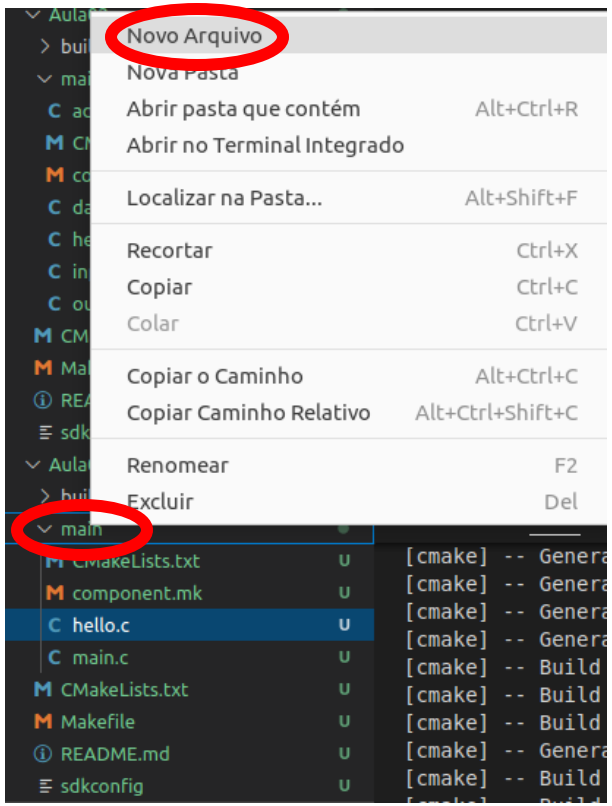
# Pino como saída

- Como fazer para lidar com correntes maiores?



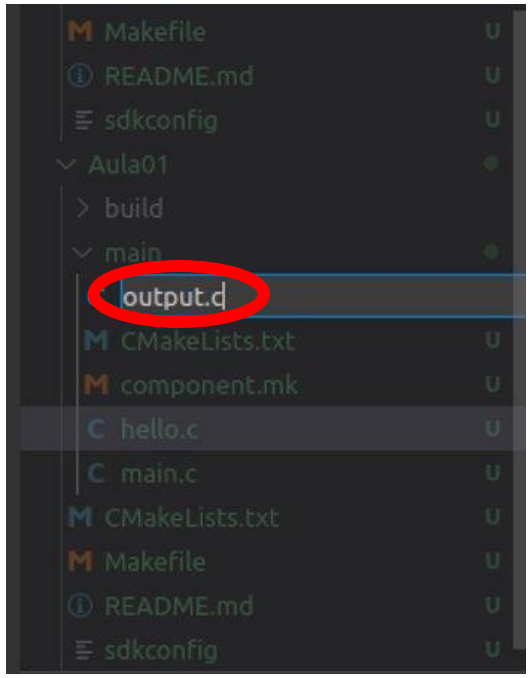
# Criando arquivo output.c

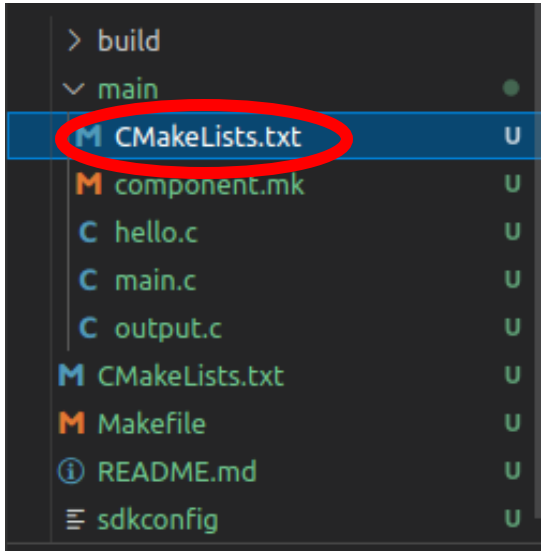
- Clique com o botão direito sobre "main"
- No menu que surge, clique em Novo Arquivo



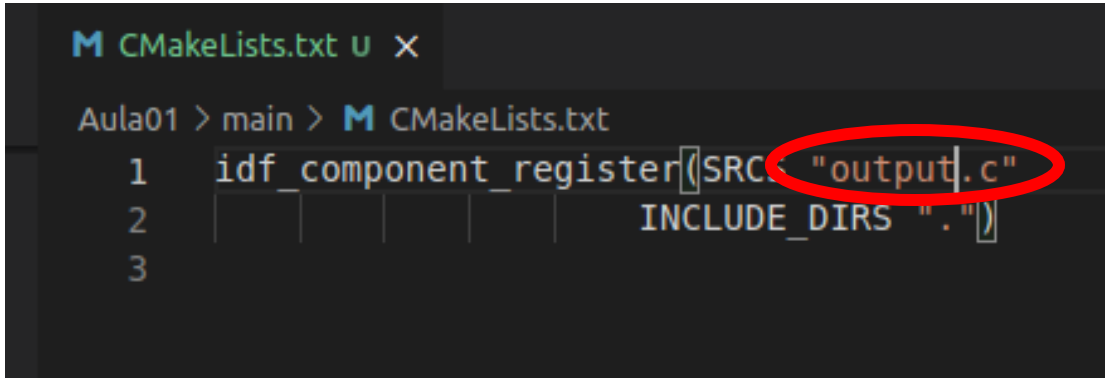
# Criando arquivo output.c (cont.)

- Digite output.c como nome do arquivo




- Edite o arquivo CMakeLists.txt de dentro do diretório "main"



# app_main template

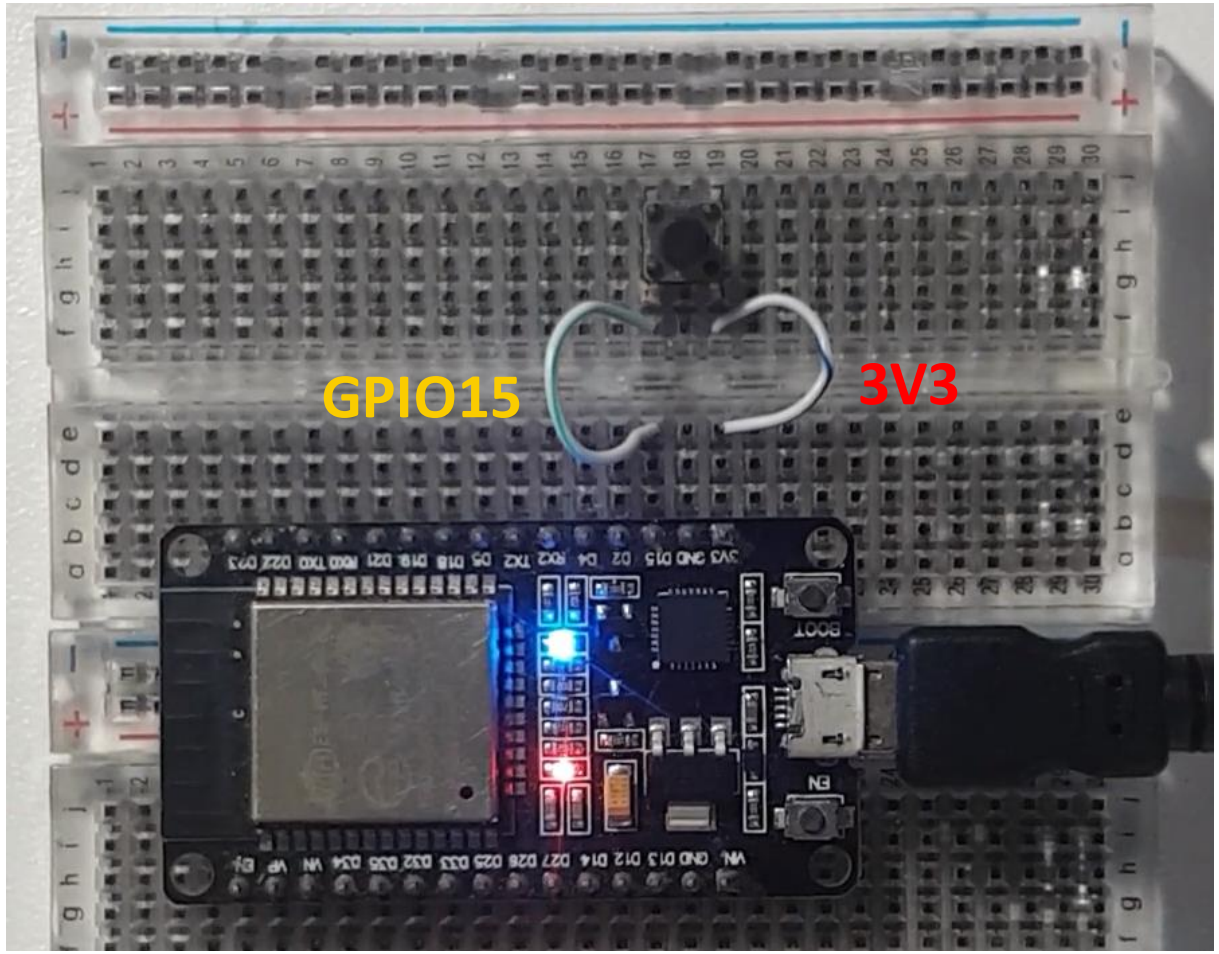

```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_log.h"

const char *TAG = "arquivo.c";

void app_main(void)
{
   ESP_LOGI(TAG, "Starting!");
}
```

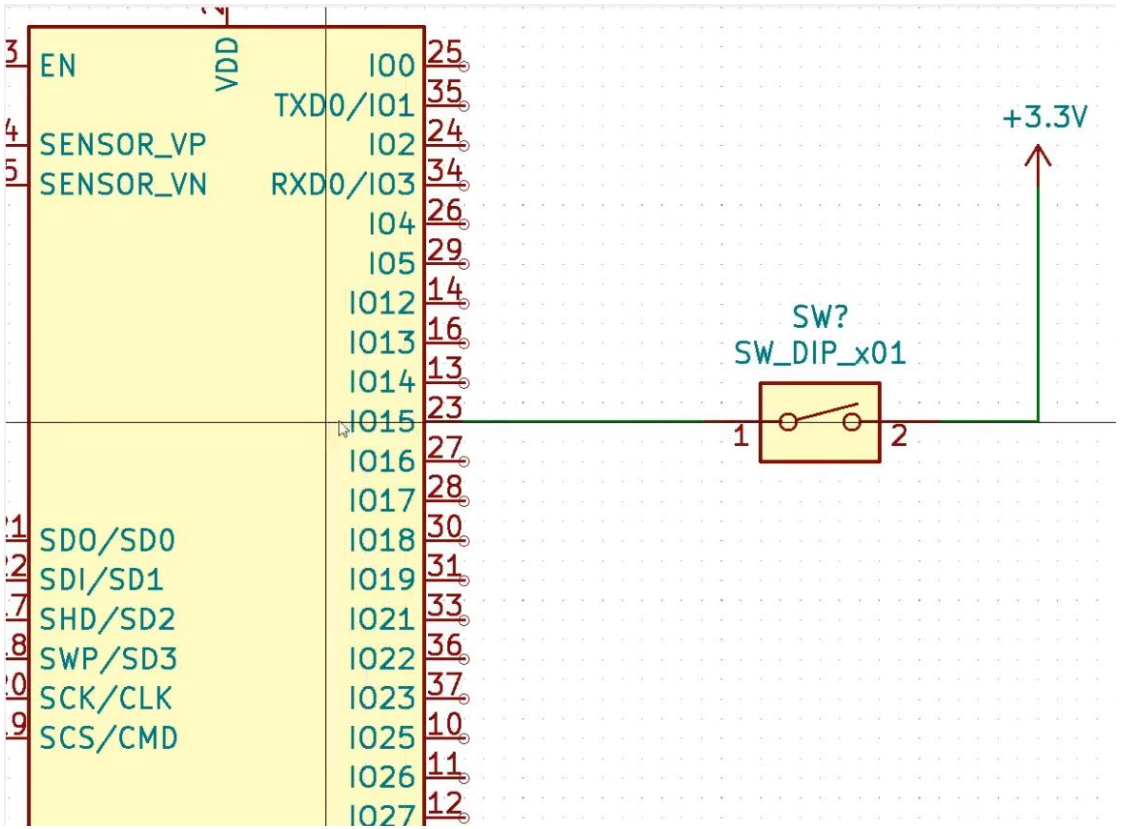
# output.c (cont.)


```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"

#define PIN_LED 2
const char *TAG = "output.c";
```

```c
void app_main(void)
{
    printf("Blink LED\n");
    ESP_LOGI(TAG, "Starting!");

    gpio_pad_select_gpio(PIN_LED);
    gpio_set_direction(PIN_LED, GPIO_MODE_OUTPUT);

    bool isOn = 0;
    while (true)
    {
        isOn = !isOn;
        gpio_set_level(PIN_LED, isOn);
        vTaskDelay(pdMS_TO_TICKS(1000));
     }
}
```

# Pino como entrada

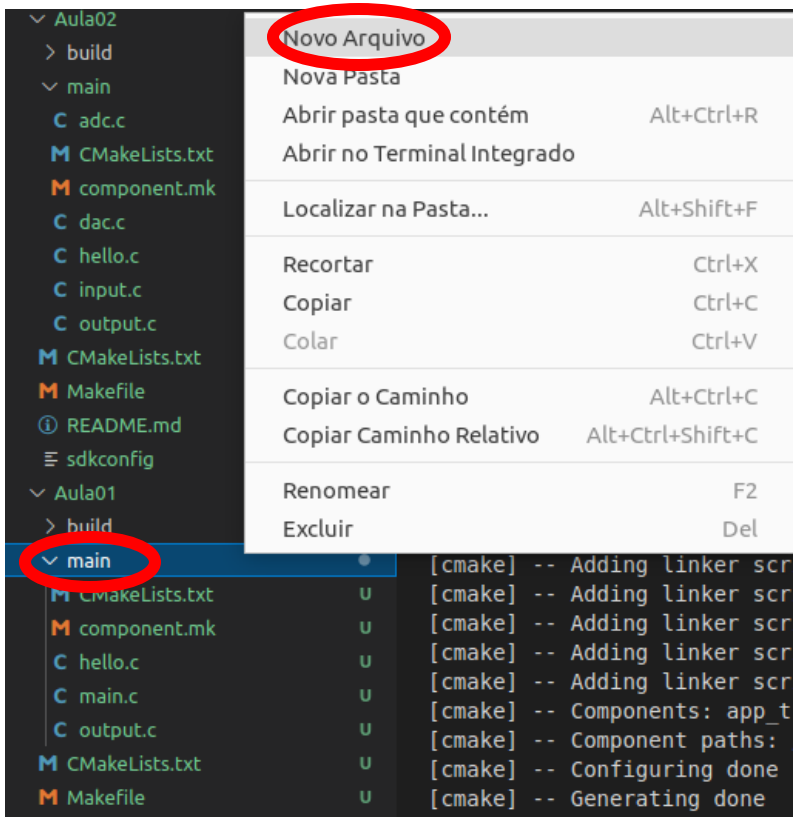



https://learnesp32.com/

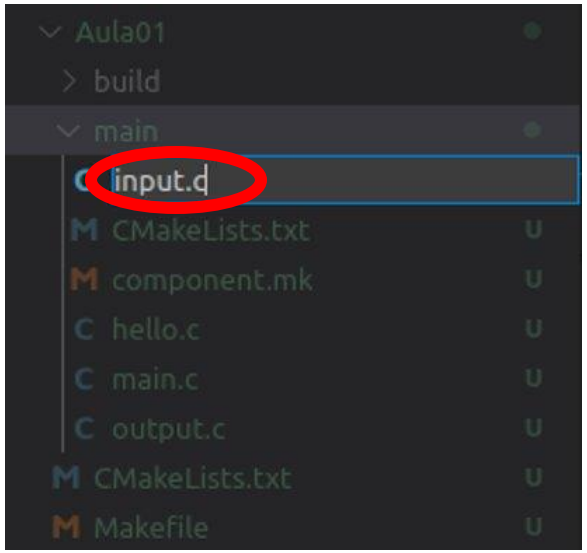
# Criando arquivo input.c

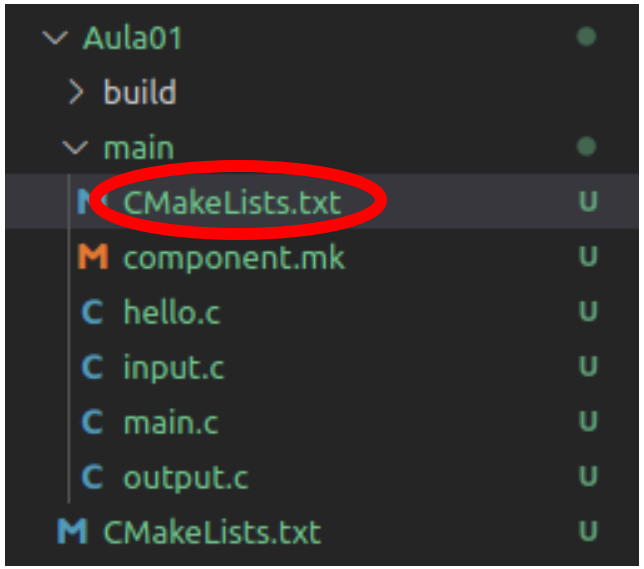
- Clique com o botão direito sobre "main"
- No menu que surge, clique em Novo Arquivo



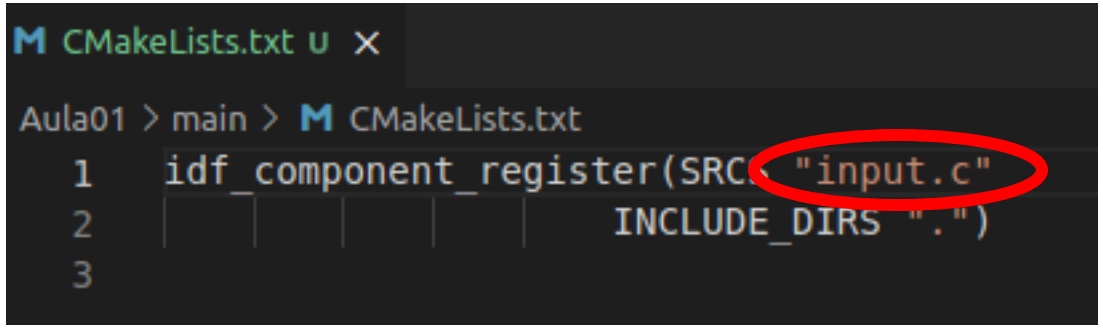
# Criando arquivo input.c (cont.)

- Digite output.c como nome do arquivo




- Edite o arquivo CMakeLists.txt de dentro do diretório "main"

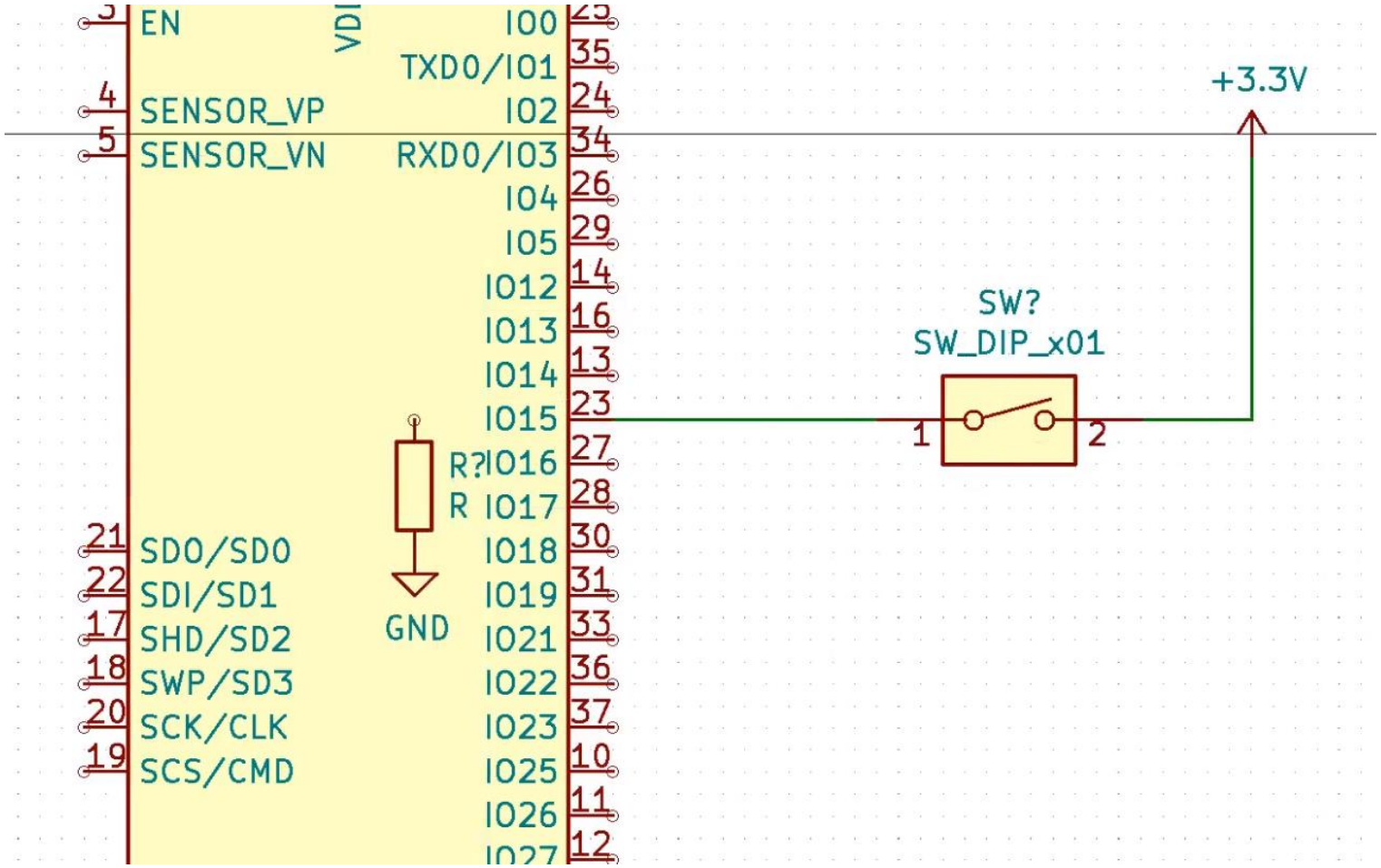


# input.c


```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"

#define PIN_SWITCH 15
#define PIN_LED 2

const char *TAG = "input.c";
```

```c
void app_main(void)
{
    printf("Switch and LED\n");
   ESP_LOGI(TAG, "Starting!");

   gpio_pad_select_gpio(PIN_LED);
   gpio_set_direction(PIN_LED, GPIO_MODE_OUTPUT);

   gpio_pad_select_gpio(PIN_SWITCH);
   gpio_set_direction(PIN_SWITCH, GPIO_MODE_INPUT);

   while (true)
   {
      int level = gpio_get_level(PIN_SWITCH);
      gpio_set_level(PIN_LED, level);
      vTaskDelay(1);
   }
}
```

# Pull Down interno



# Pull-down enable

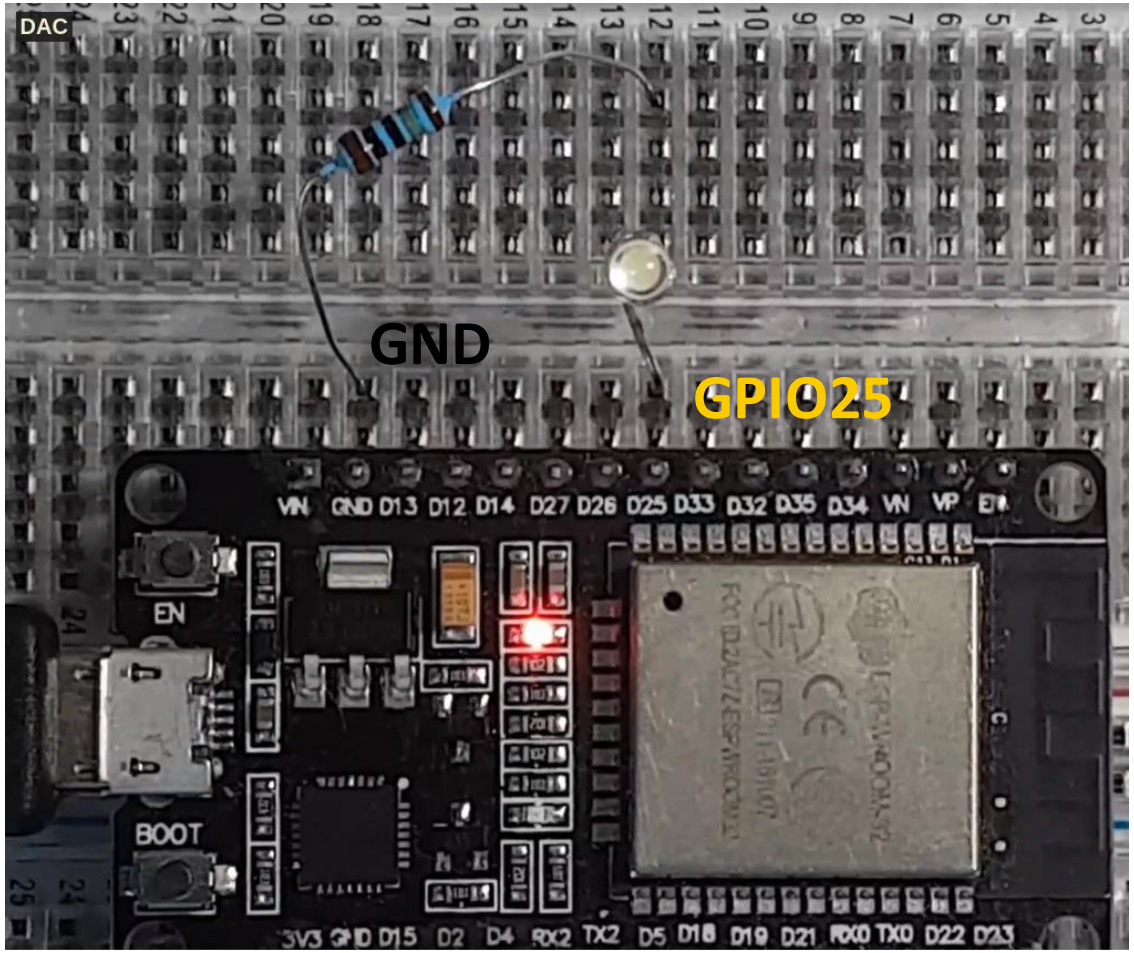

```
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "esp_log.h"

#define PIN_SWITCH 15
#define PIN_LED 2

const char *TAG = "input.c";
```

```
void app_main(void)
{
    printf("Switch and LED\n");
    ESP_LOGI(TAG, "Starting!");

    gpio_pad_select_gpio(PIN_LED);
    gpio_set_direction(PIN_LED, GPIO_MODE_OUTPUT);
    gpio_pad_select_gpio(PIN_SWITCH);
    gpio_set_direction(PIN_SWITCH, GPIO_MODE_INPUT);

    gpio_pulldown_en(PIN_SWITCH);
    gpio_pullup_dis(PIN_SWITCH);

    while (true)
    {
        int level = gpio_get_level(PIN_SWITCH);
        gpio_set_level(PIN_LED, level);
        vTaskDelay(1);
    }
}
```

# DAC

- Digital to Analog Converter;
- 2 canais: GPIO25 (canal 1) e GPIO26 (canal 2);
- Permite alterar os níveis de tensão em um pino;

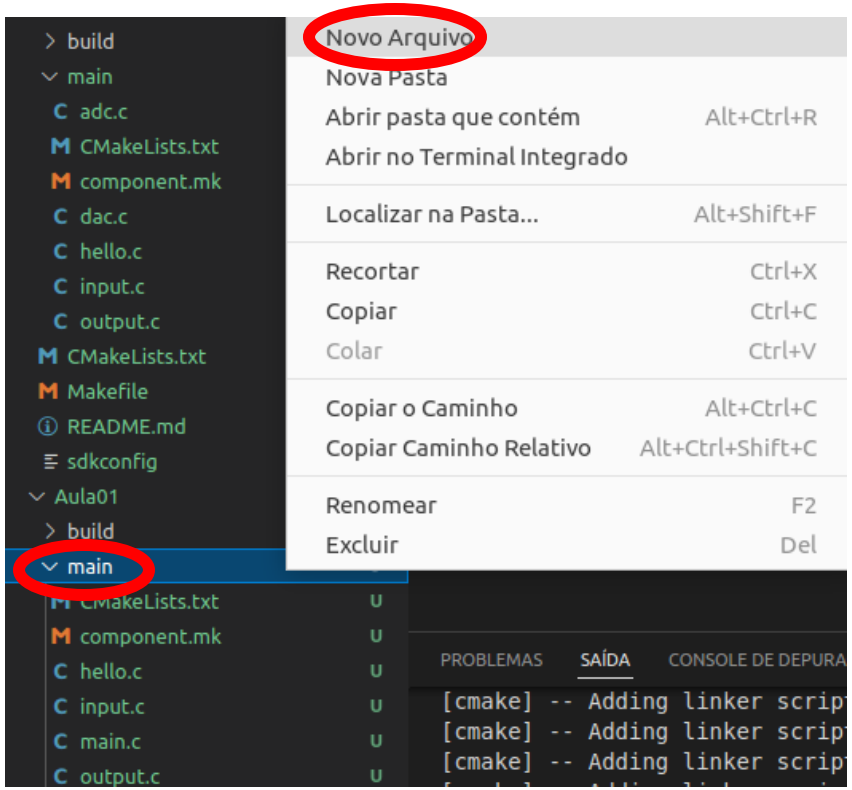
# Montagem DAC




https://learnesp32.com/

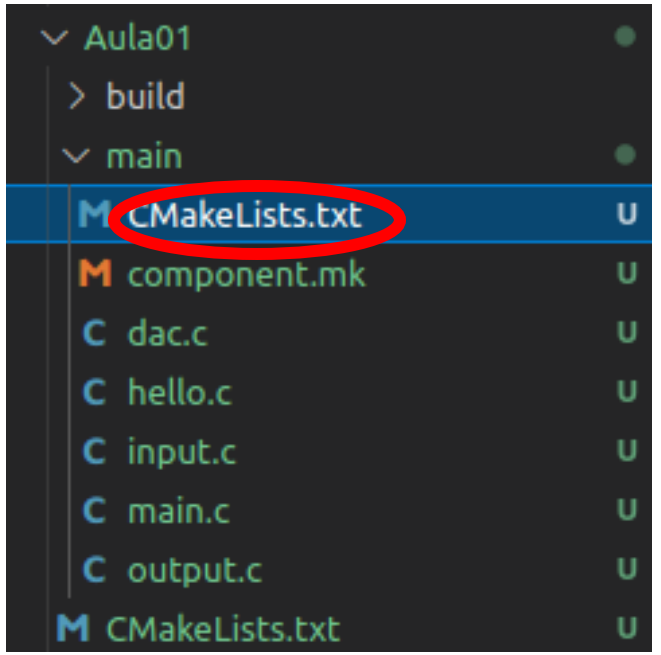
# Criando arquivo dac.c

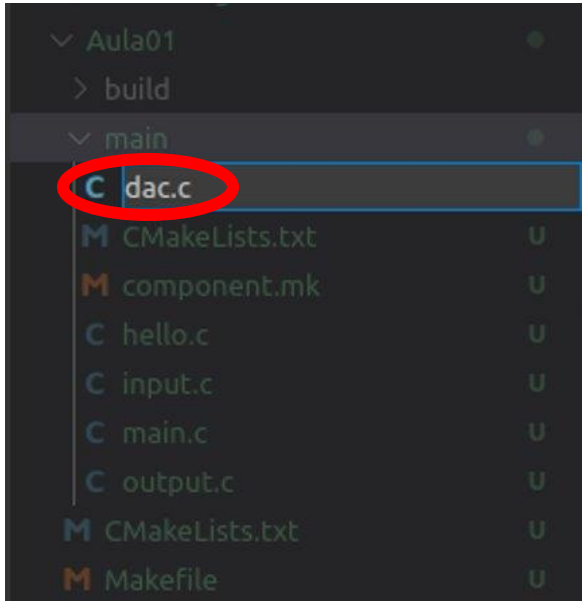
- Clique com o botão direito sobre "main"
- No menu que surge, clique em Novo Arquivo



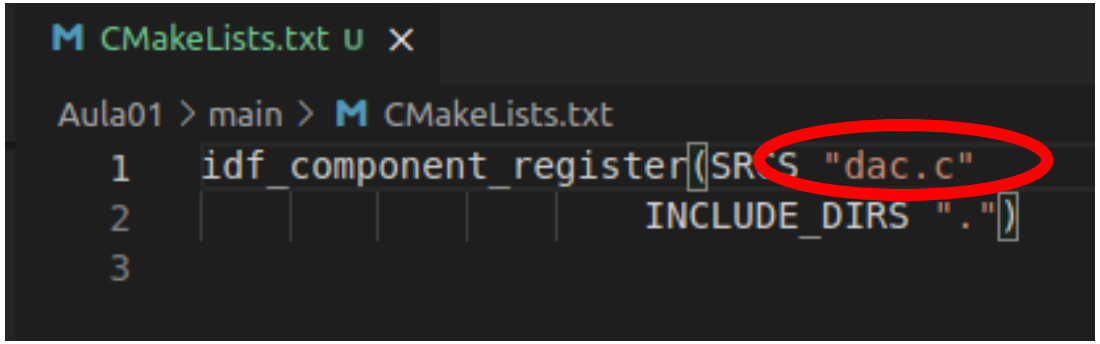
# Criando arquivo dac.c (cont.)

- Digite output.c como nome do arquivo




- Edite o arquivo CMakeLists.txt de dentro do diretório "main"



# Código DAC


```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/gpio.h"
#include "driver/dac.h"

void app_main()
{
    dac_output_enable(DAC_CHANNEL_1);
    dac_output_voltage(DAC_CHANNEL_1, 255);
}
```

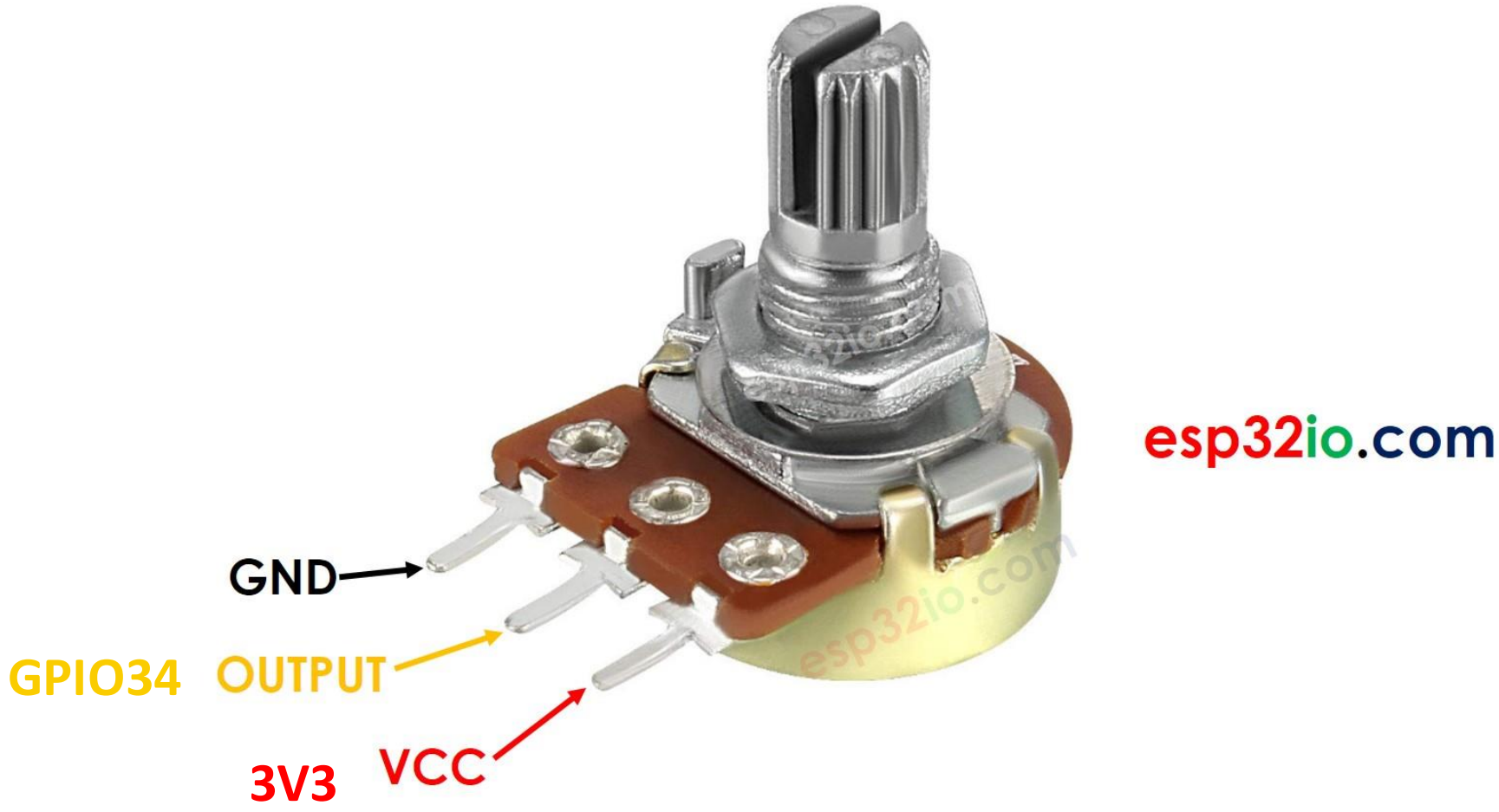
# ADC

- Analog to Digital Converter;
- **ADC1:** 8 canais: GPIO32 - GPIO39
- **ADC2:** 10 canais: GPIO0, GPIO2, GPIO4, GPIO12 - GPIO15, GOIO25 – GPIO27
- Faz leituras de tensão nos pinos;

- ESP32 DevKitC: GPIO 0 não pode ser utilizado devido a circuitos externos de auto programação;
- ADC2 é também utilizado pelo Wi-Fi,

https://docs.espressif.com/projects/esp-idf/en/v4.3.2/esp32/api-reference/peripherals/adc.html

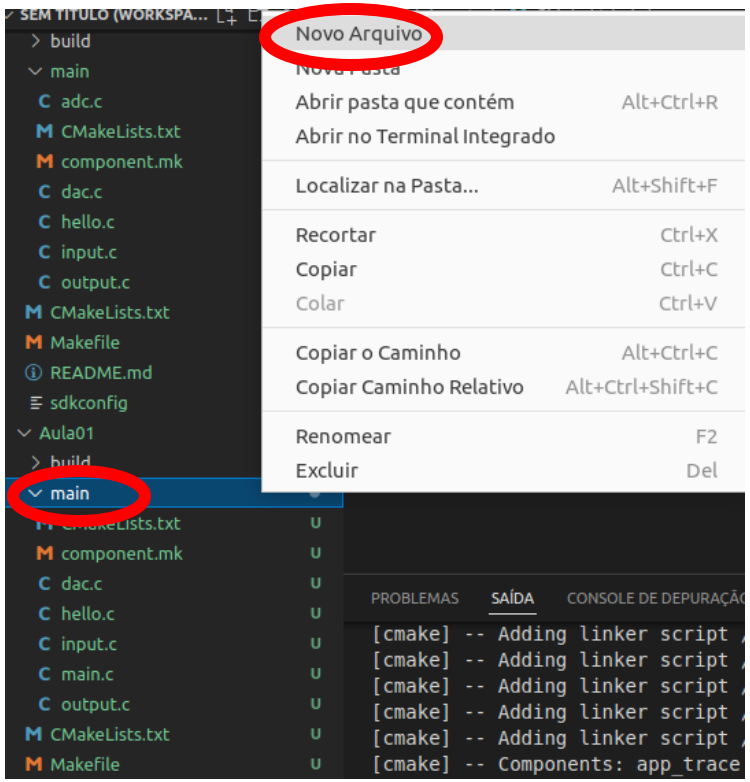
# ADC Montagem:




https://esp32io.com/images/tutorial/esp32-potentiometer-pinout.jpg

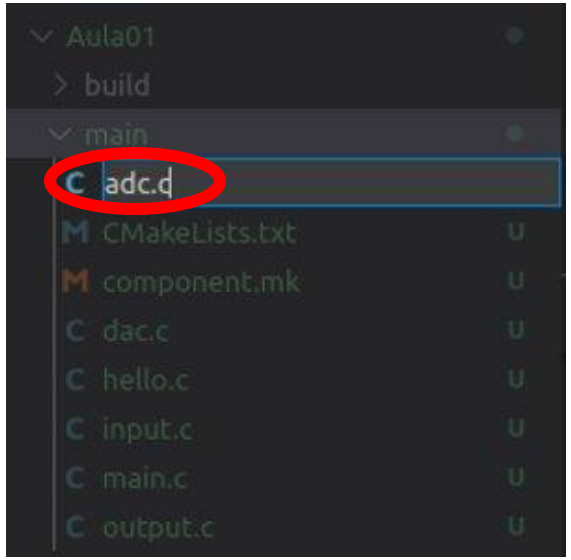
# Criando arquivo adc.c

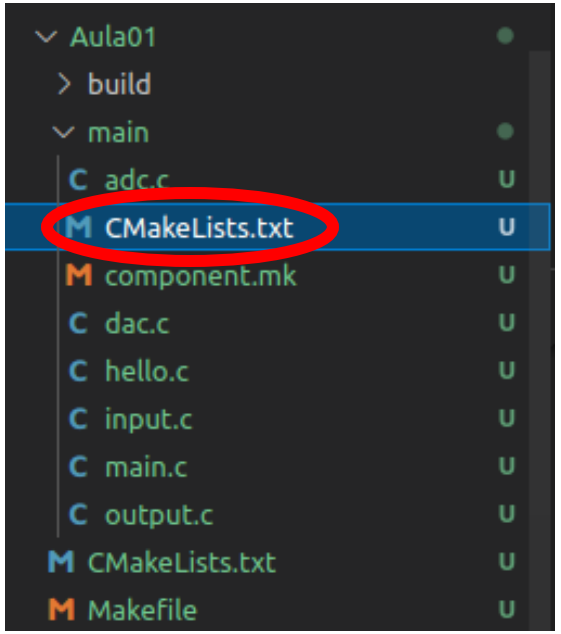
- Clique com o botão direito sobre "main"
- No menu que surge, clique em Novo Arquivo



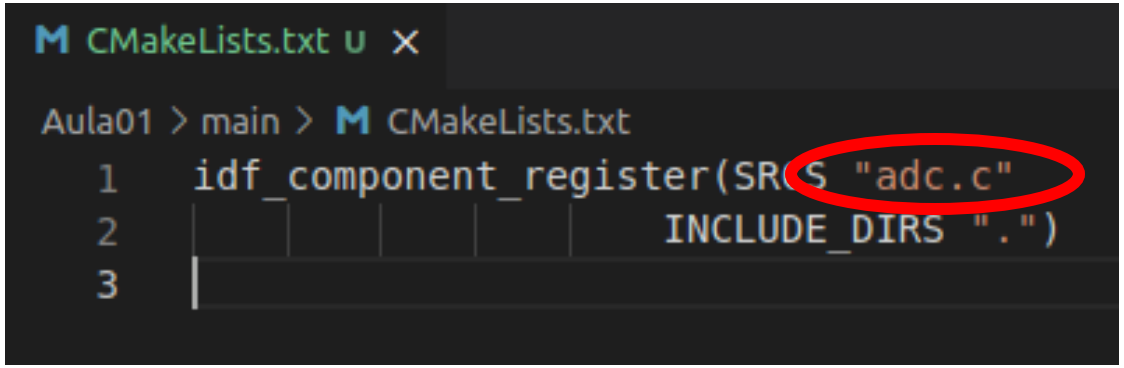
# Criando arquivo adc.c (cont.)

- Digite adc.c como nome do arquivo




- Edite o arquivo CMakeLists.txt de dentro do diretório "main"



# ADC: Código


```c
#include <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/adc.h"

void app_main()
{
    adc1_config_width(ADC_WIDTH_12Bit);
    adc1_config_channel_atten(ADC1_CHANNEL_6, ADC_ATTEN_11db);

    while(1)
    {
        int val = adc1_get_raw(ADC1_CHANNEL_6);
        printf("value is %d\n", val);
        vTaskDelay(pdMS_TO_TICKS(1000));
    }
}
```

# Exercícios

1. Faça um programa que ao manter o botão pressionado o LED pisca em uma frequência de 10 Hz e quando solto em 2 Hz

2. Faça um programa que ao pressionar o botão o led pisca e pressionar novamente o led fica ligado

# Exercícios (cont.)

3. Faça um programa que a medida que o botão é pressionado aumente a intensidade do LED ao atingir o valor máximo volte a zero. Adicione 5 passos.

4. O ESP32 possui um sensor hall integrado. Descubra como fazer leituras dele e apresente os dados a cada 500ms

# Exercícios (cont.)

5. Faça um monitor de bateria de Li Íon.

- Crie um circuito com um potenciômetro que simule os valores da bateria: 2,8V a 4,2V.
- Com o ESP32 faça a leitura desses valores e acenda os leds de acordo com a tabela

| LED Ligado | Faixa de Tensão |
|---|---|
| Verde | 4,2V à 3,75V |
| Amarelo | 3,75 à 3,4V |
| Vermelho | 3,4 à 2,8V |

# Recomendados


- Kolban's Book on ESP32
  - https://leanpub.com/kolban-ESP32
- Andreas Spiess
  - https://www.youtube.com/playlist?list=PL3XBzmAj53RnZPeWe799F-uoXERBldhn9
- Projetos com ESP32
  - https://hackaday.com/tag/ESP32/
  - https://www.hackster.io/search?i=projects&q=ESP32
  - https://www.instructables.com/howto/ESP32/

PADO

# Labs