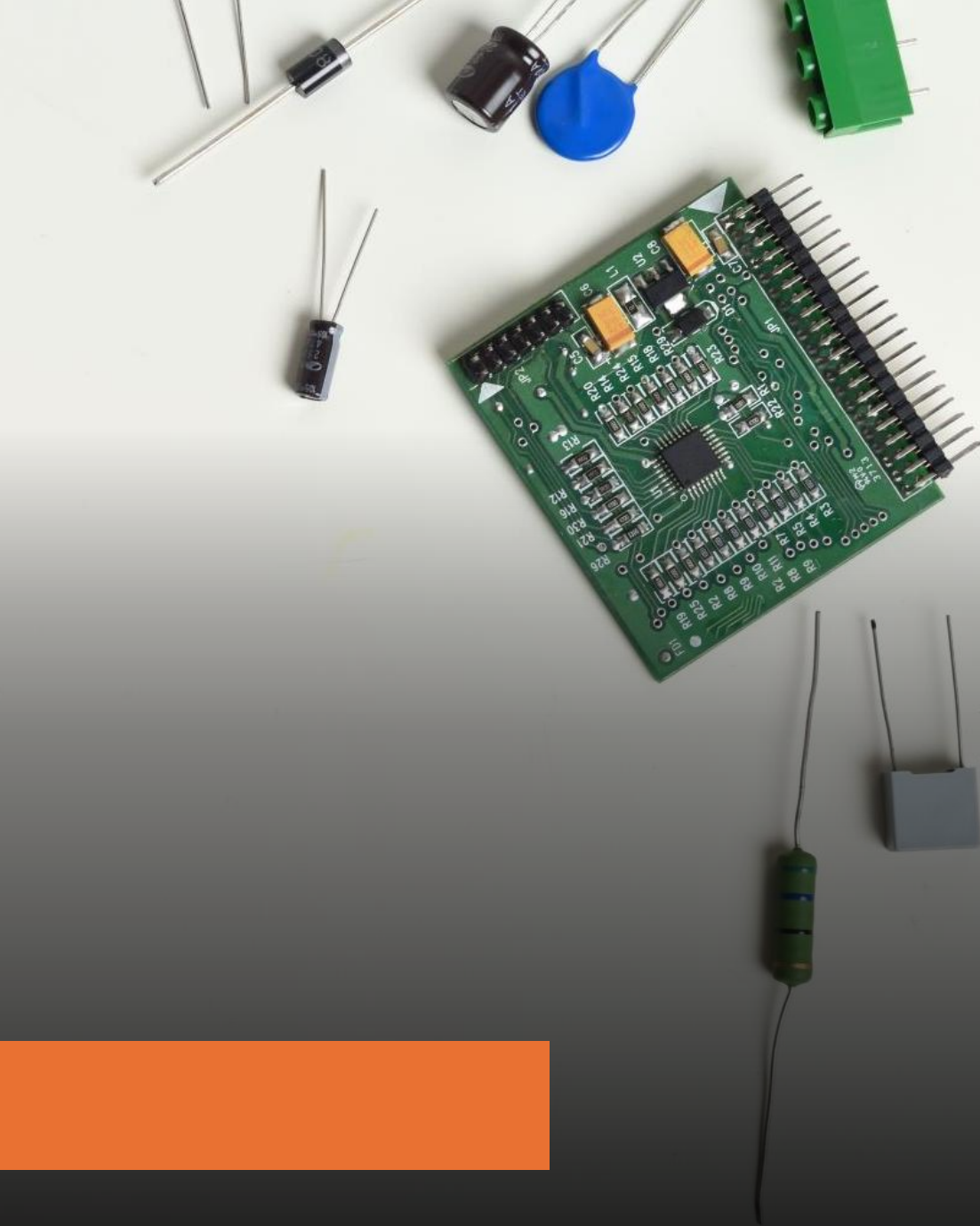


# Sistemas Embebidos

Prof. Flávia Pires

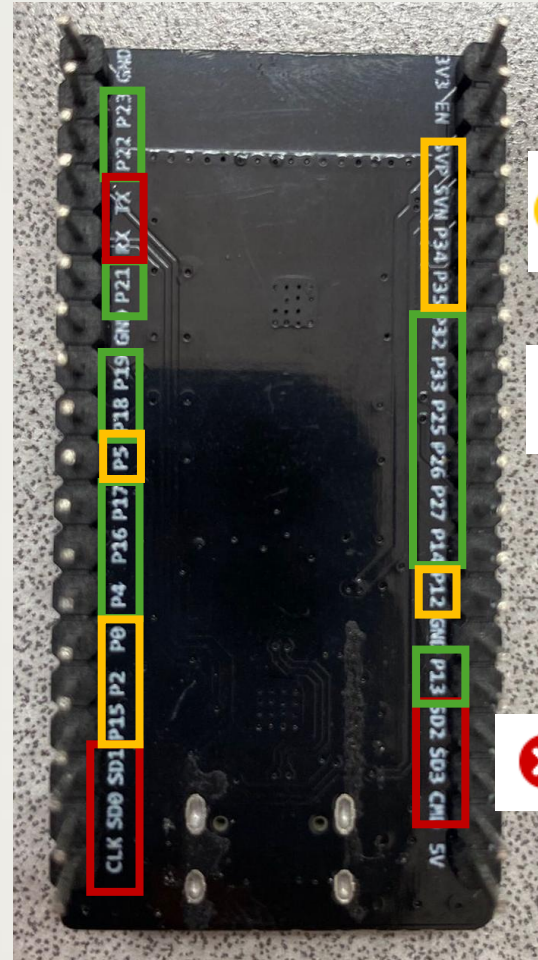
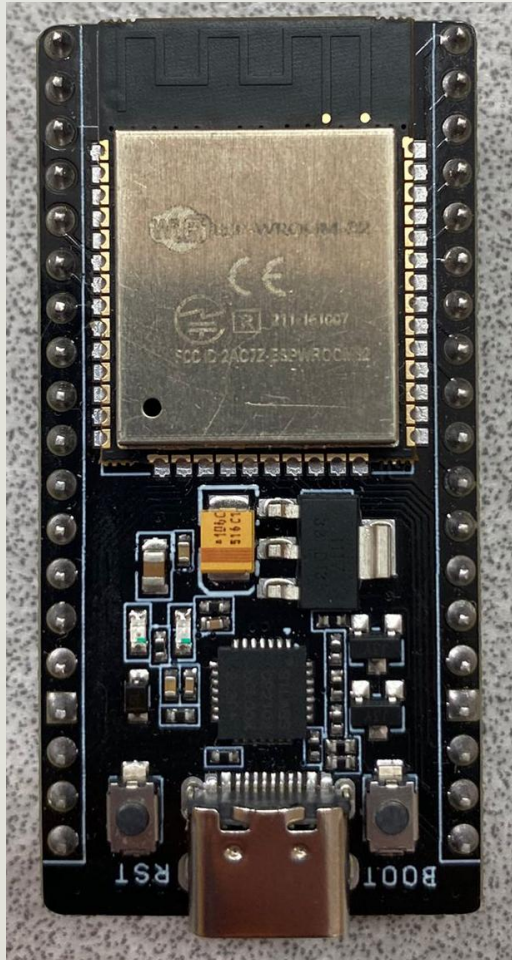
**email:** [fpres@ipb.pt](mailto:fpres@ipb.pt)

Gab. 106

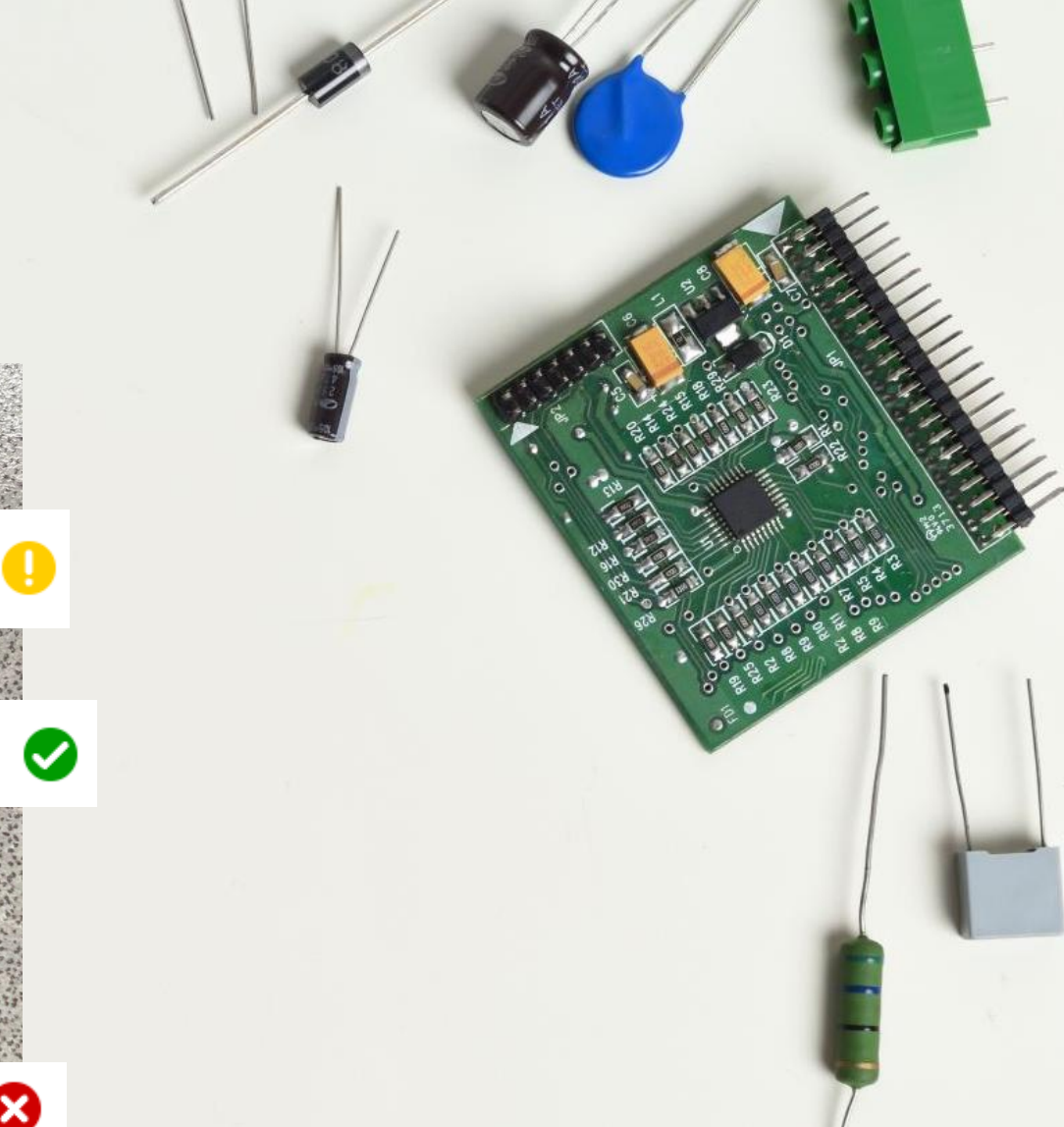


# Aula 7

- **Software:** Thonny
- **Hardware:** ESP32, Resistências, LEDs, Botões



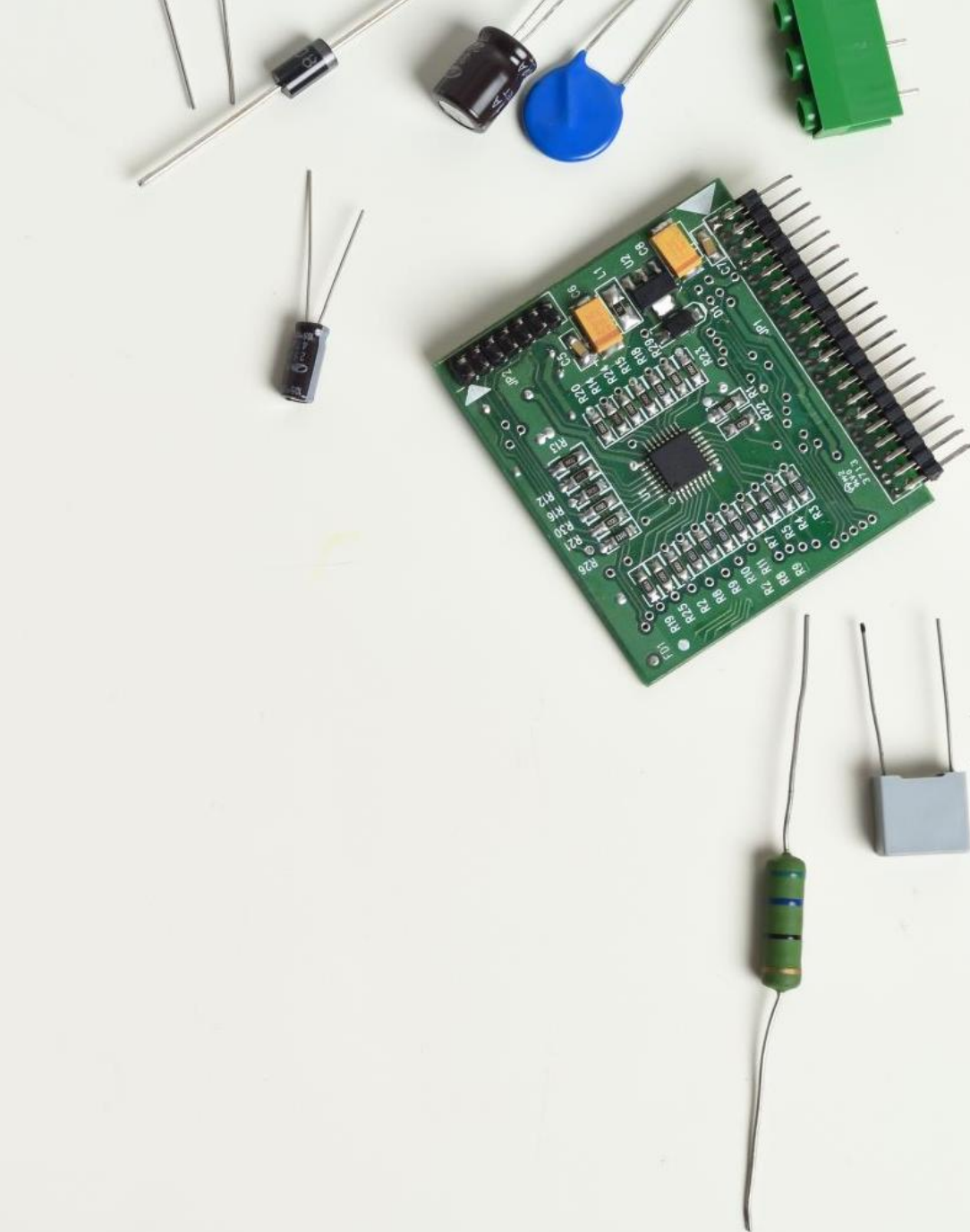
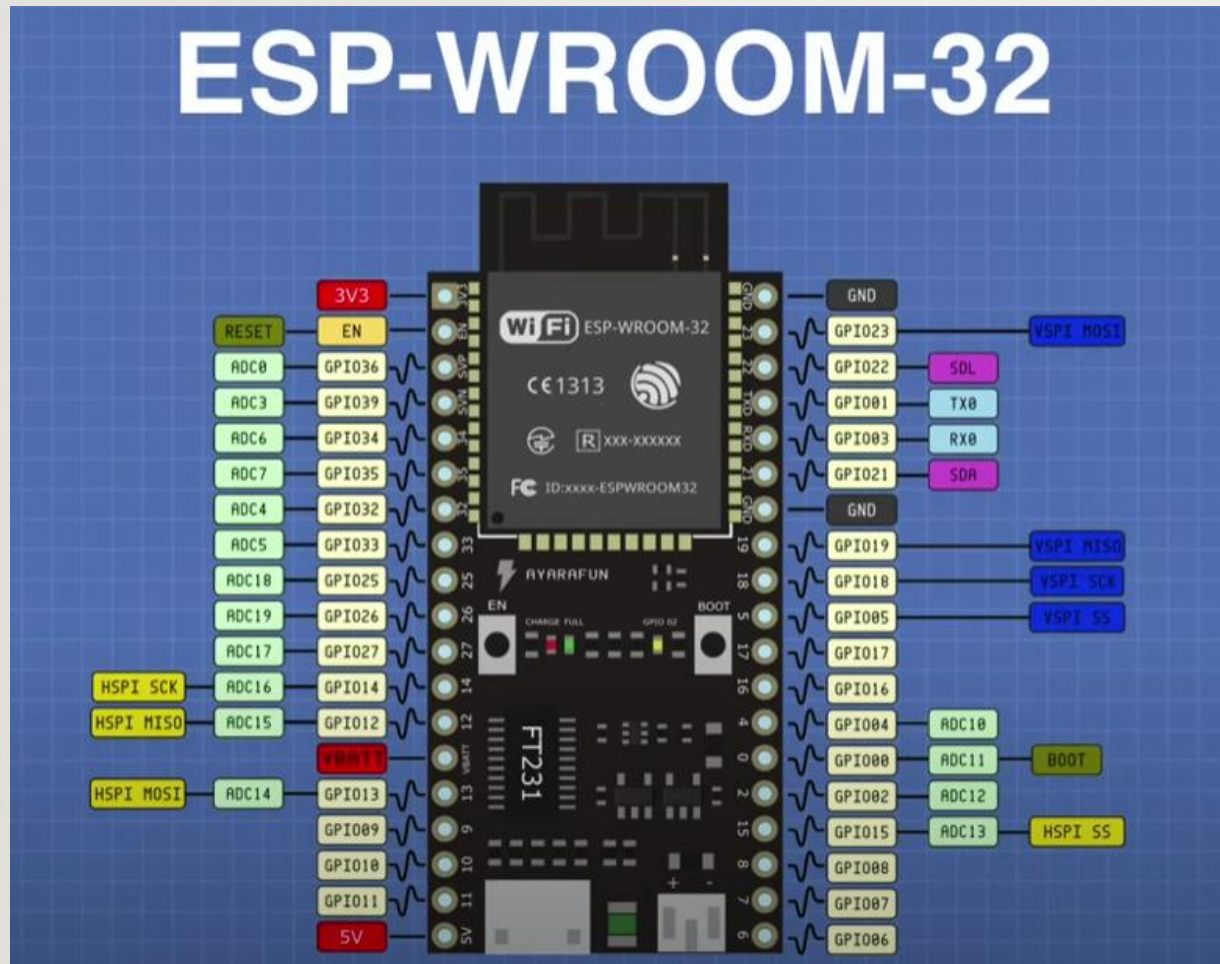
- ✓ – Your top priority pins. They are perfectly safe to use.
- ! – Pay close attention because their behavior, particularly during boot, can be unpredictable. Use them only when absolutely necessary.
- ✗ – It is recommended that you avoid using these pins.





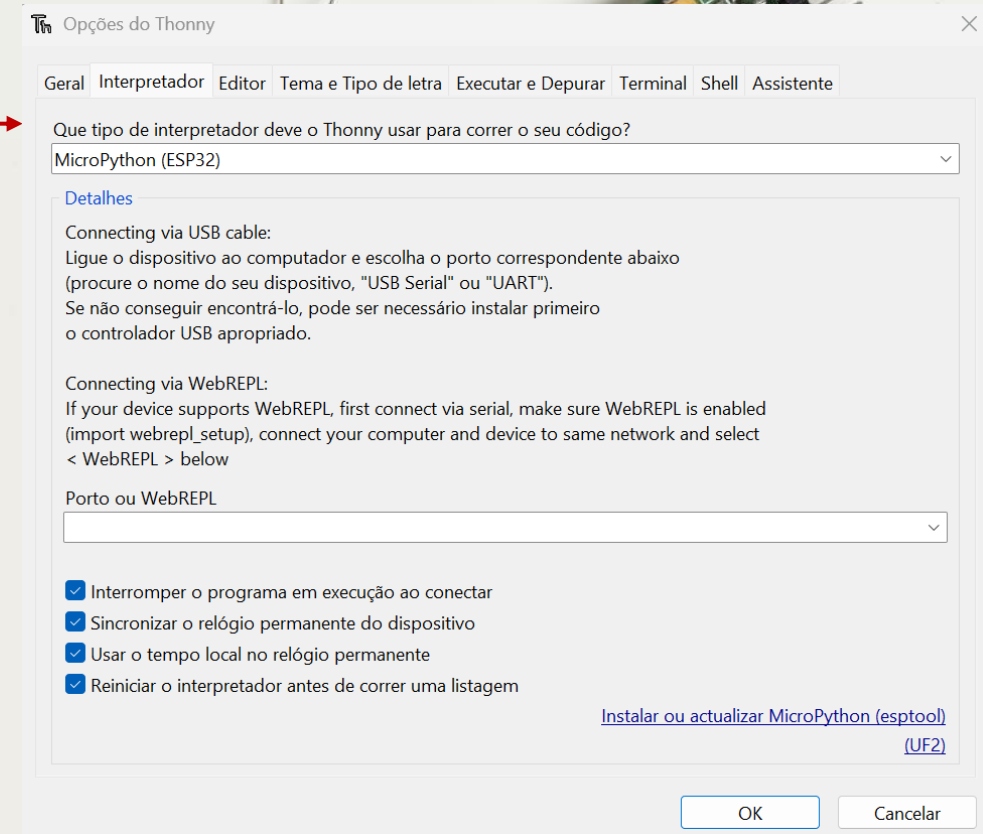
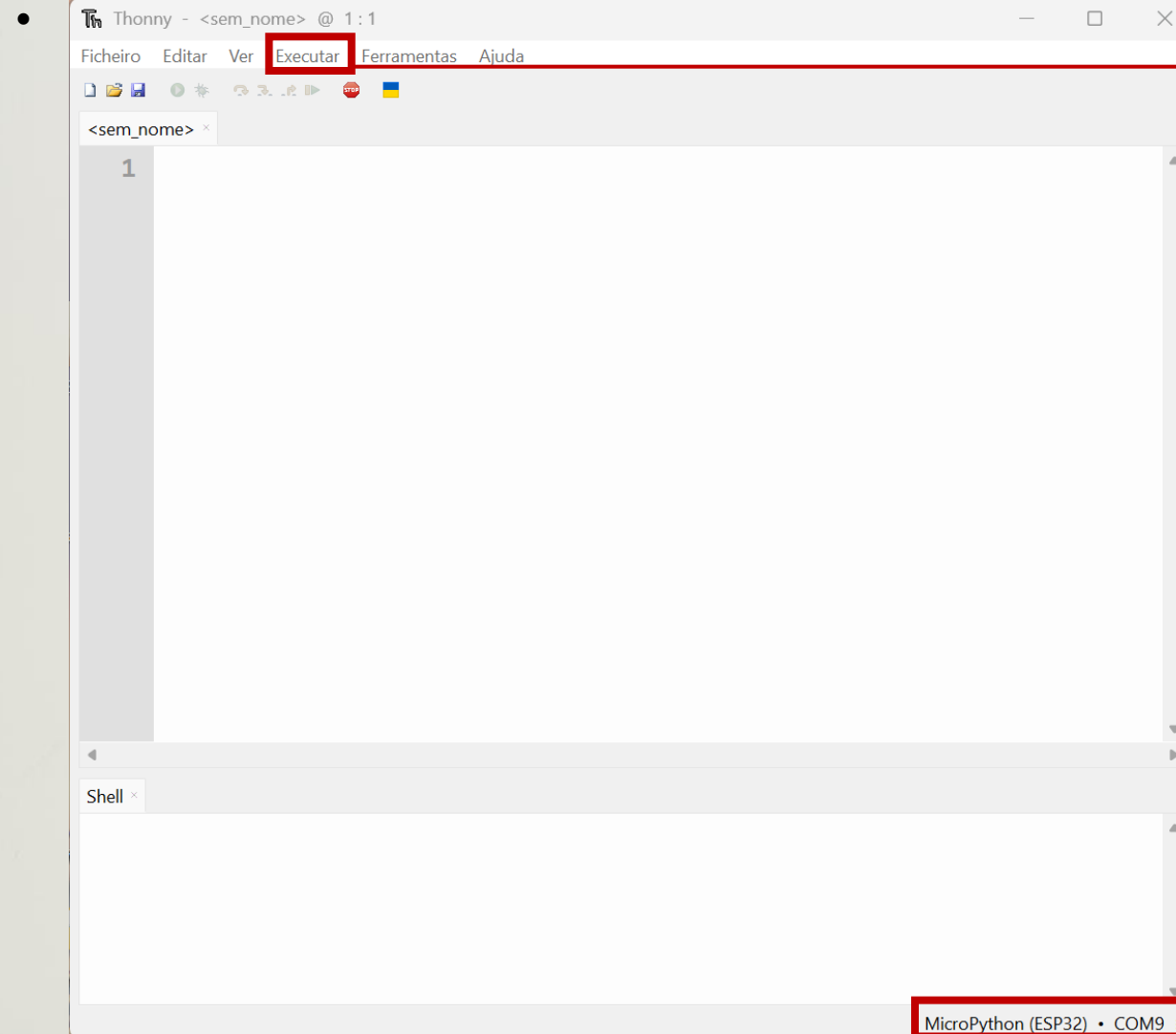
# Aula 7

- **Software:** Thonny
- **Hardware:** ESP32, Resistências, LEDs, Botões



# Aula 7

- Instalação Thonny (<https://thonny.org/>)

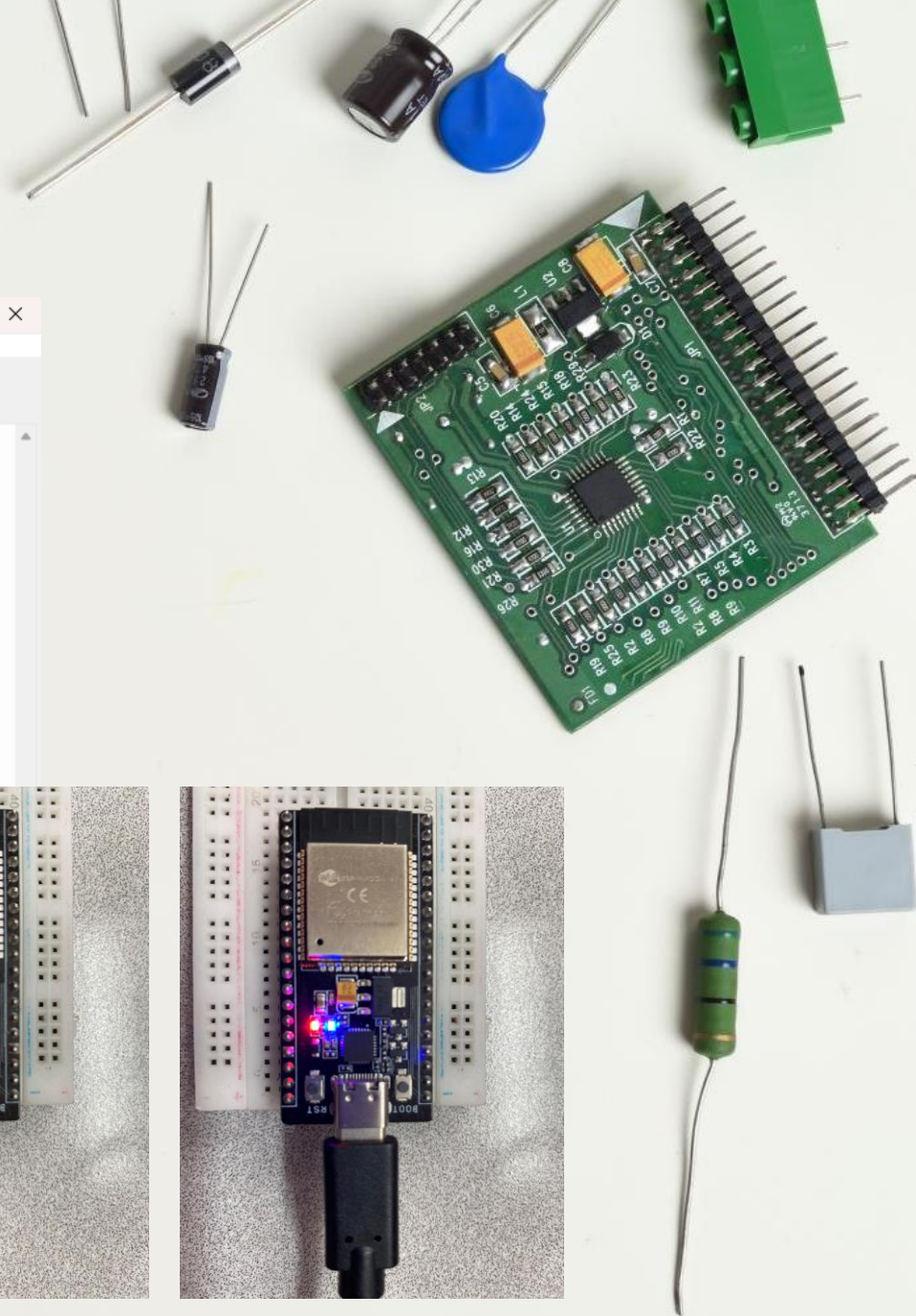
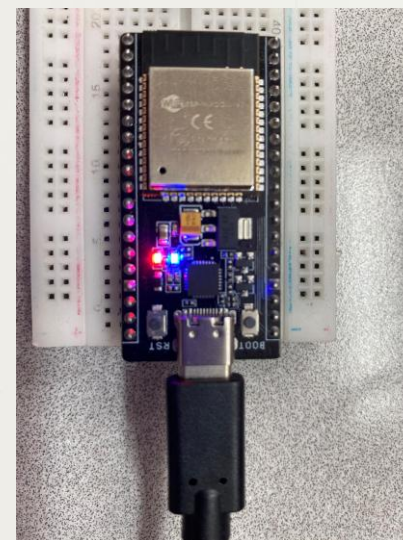
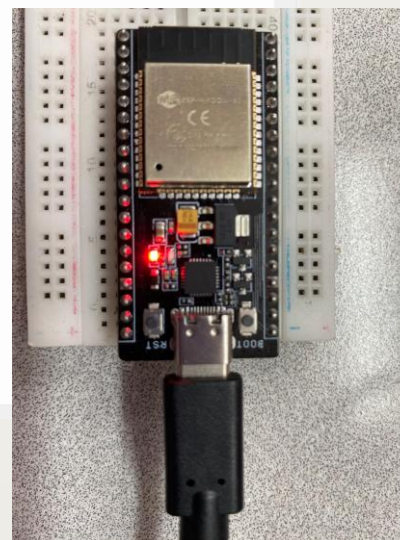


Interpretador e Porta

# Aula 7

- Controlar o LED Interno com Sleep

```
Thonny - C:\Users\flavi\BLINK_LED\main.py @ 13:53
Ficheiro  Editar  Ver  Executar  Ferramentas  Ajuda
main.py *
1 #Exemplo LED BLINK COM SLEEP
2 from machine import Pin # Importa a classe Pin da biblioteca 'machine' para controlar GPIOs
3 from time import sleep # Importa a função sleep para adicionar pausas
4
5 # Define o pino 2 (GPIO2) como saída digital
6 led = Pin(2, Pin.OUT) # GPIO2 geralmente é o LED onboard
7
8 # Loop infinito que pisca o LED
9 while True:
10     led.on() # Liga o LED (coloca o pino em nível alto - 3.3V)
11     sleep(1) # Aguarda 1 segundo com o LED ligado
12     led.off() # Desliga o LED (coloca o pino em nível baixo - 0V)
13     sleep(1) # Aguarda 1 segundo com o LED desligado
```





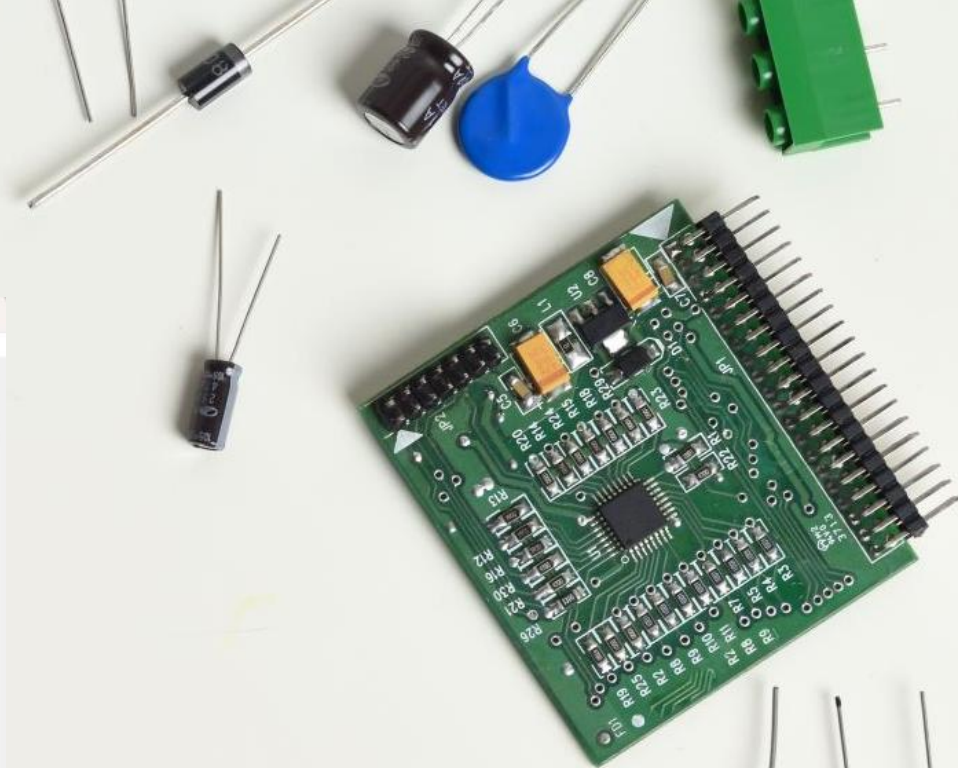
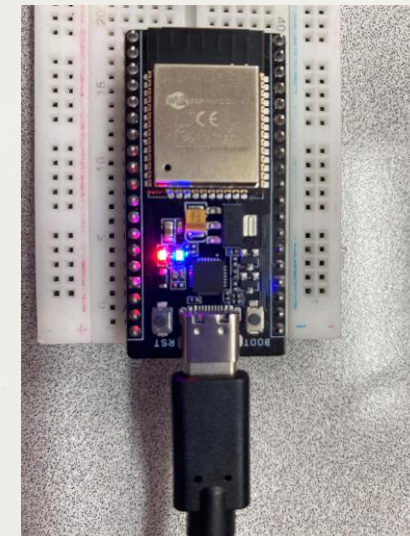
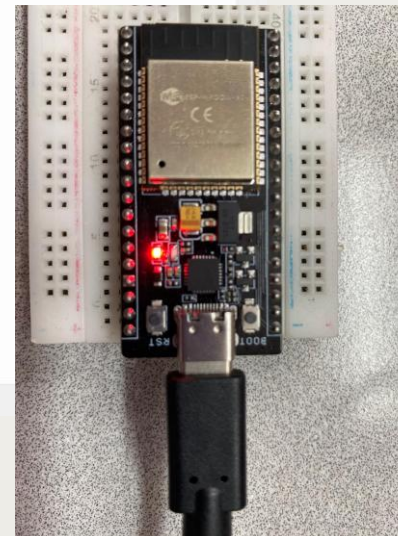
# Aula 7

- Controlar o LED interno sem Sleep

```
Thonny - C:\Users\flavi\BLINK_LED_S_SLEEP\main.py @ 15:74
Ficheiro  Editar  Ver  Executar  Ferramentas  Ajuda

main.py
1  from machine import Pin
2  import time
3
4  led = Pin(2, Pin.OUT)
5  estado_led = False
6  ultimo_tempo = time.ticks_ms()
7
8  while True:
9      tempo_atual = time.ticks_ms()
10
11     # Verifica se 1000 milissegundos (1 segundo) se passaram
12     if time.ticks_diff(tempo_atual, ultimo_tempo) >= 1000:
13         estado_led = not estado_led # Alterna o estado do LED
14         led.value(estado_led)      # Atualiza o pino com o novo estado
15         ultimo_tempo = tempo_atual # Atualiza o tempo da última mudança

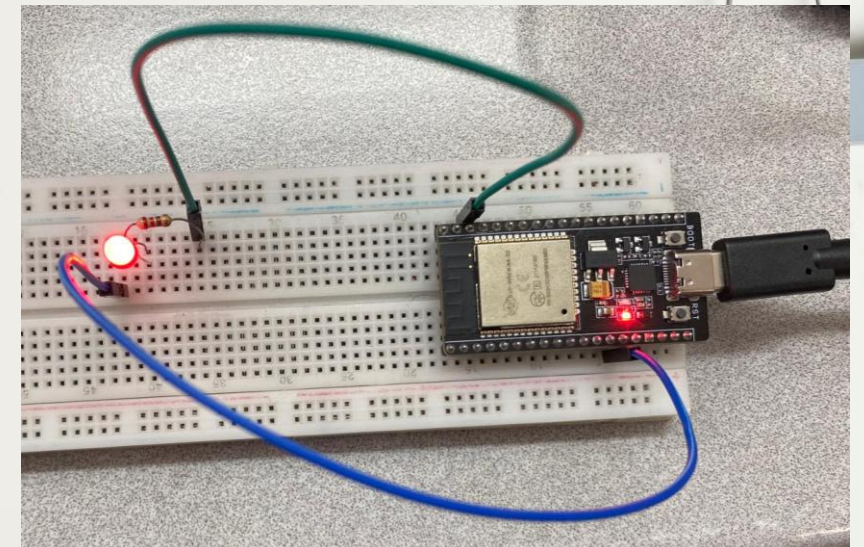
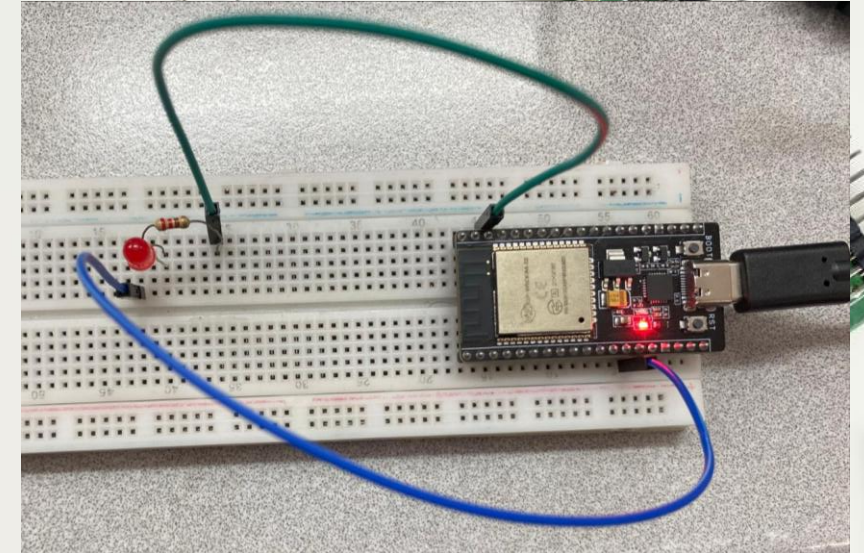
Shell
```



# Aula 7

- Controlar um LED externo PIN 22

```
main.py * ×
1 #Exemplo LED BLINK COM SLEEP
2 from machine import Pin # Importa a classe Pin da biblioteca 'machine' para controlar GPIOs
3 from time import sleep # Importa a função sleep para adicionar pausas
4
5 # Define o pino 22 (GPIO2) como saída digital
6 led = Pin(22, Pin.OUT) # GPIO2 geralmente é o LED onboard
7
8 # Loop infinito que pisca o LED
9 while True:
10     led.on() # Liga o LED (coloca o pino em nível alto - 3.3V)
11     sleep(1) # Aguarda 1 segundo com o LED ligado
12     led.off() # Desliga o LED (coloca o pino em nível baixo - 0V)
13     sleep(1) # Aguarda 1 segundo com o LED desligado
```





# Aula 7

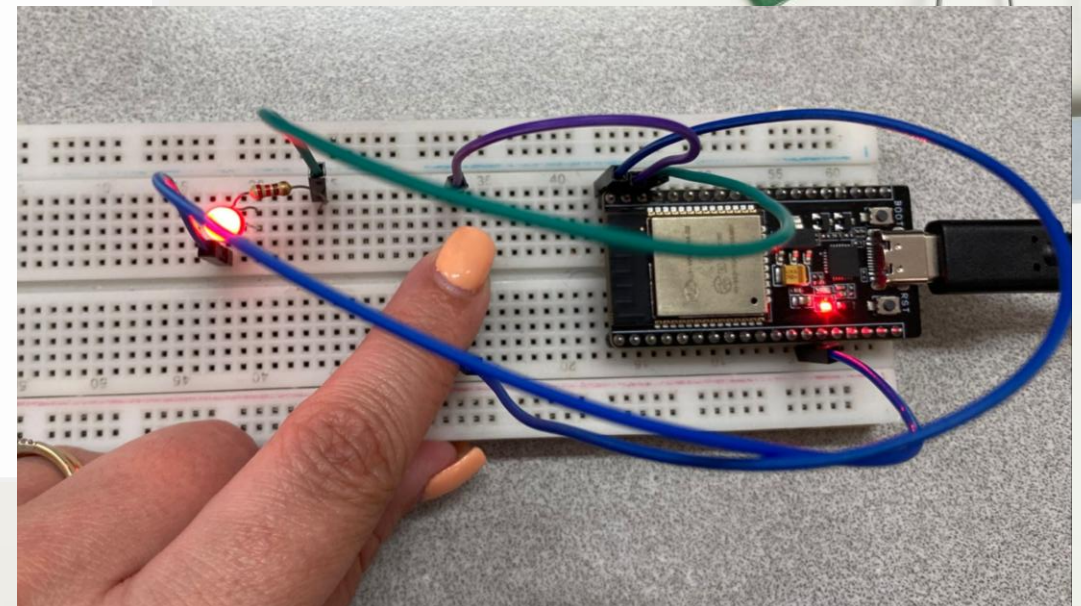
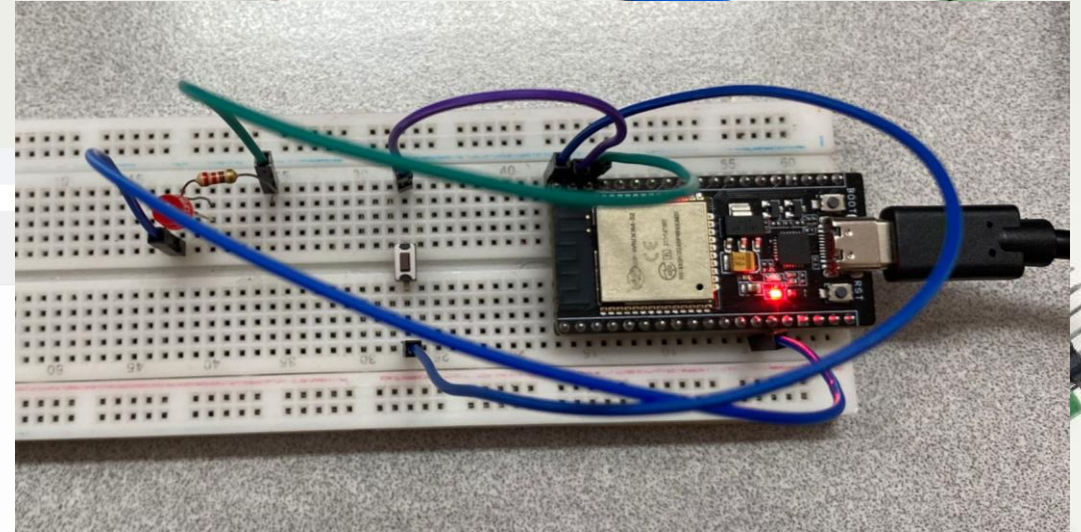
- Controlar um LED externo através do botão

Thonny - C:\Users\flavi\Botao\main.py @ 11:31

Ficheiro Editar Ver Executar Ferramentas Ajuda

main.py

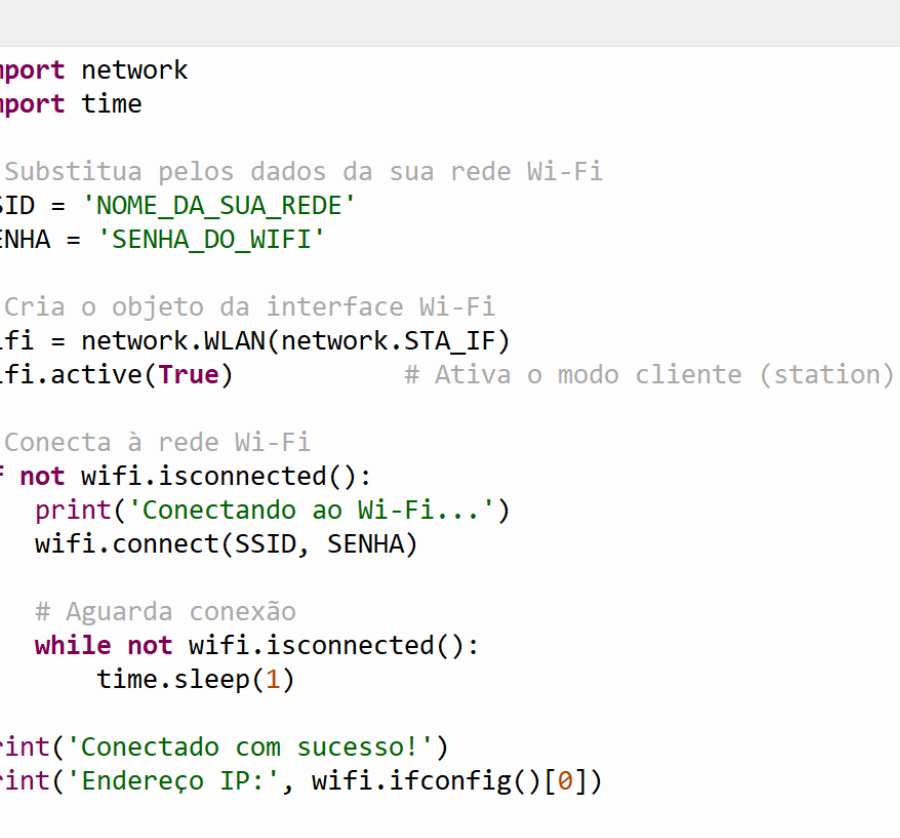
```
1 from machine import Pin
2 from time import sleep
3
4 # Configurações dos pinos
5 botao = Pin(23, Pin.IN, Pin.PULL_UP) # GPIO23 com resistor pull-up interno
6 led = Pin(22, Pin.OUT)                # GPIO22 com LED externo
7
8 while True:
9     print(botao.value()) # Mostrar o valor do botão
10    if botao.value() == 0: # Se o botão estiver pressionado
11        print(botao.value()) # Mostrar o valor do botão
12        led.on()
13    else:
14        led.off()
15    sleep(0.05) # Pequena pausa para evitar ruído (debounce simples)
```



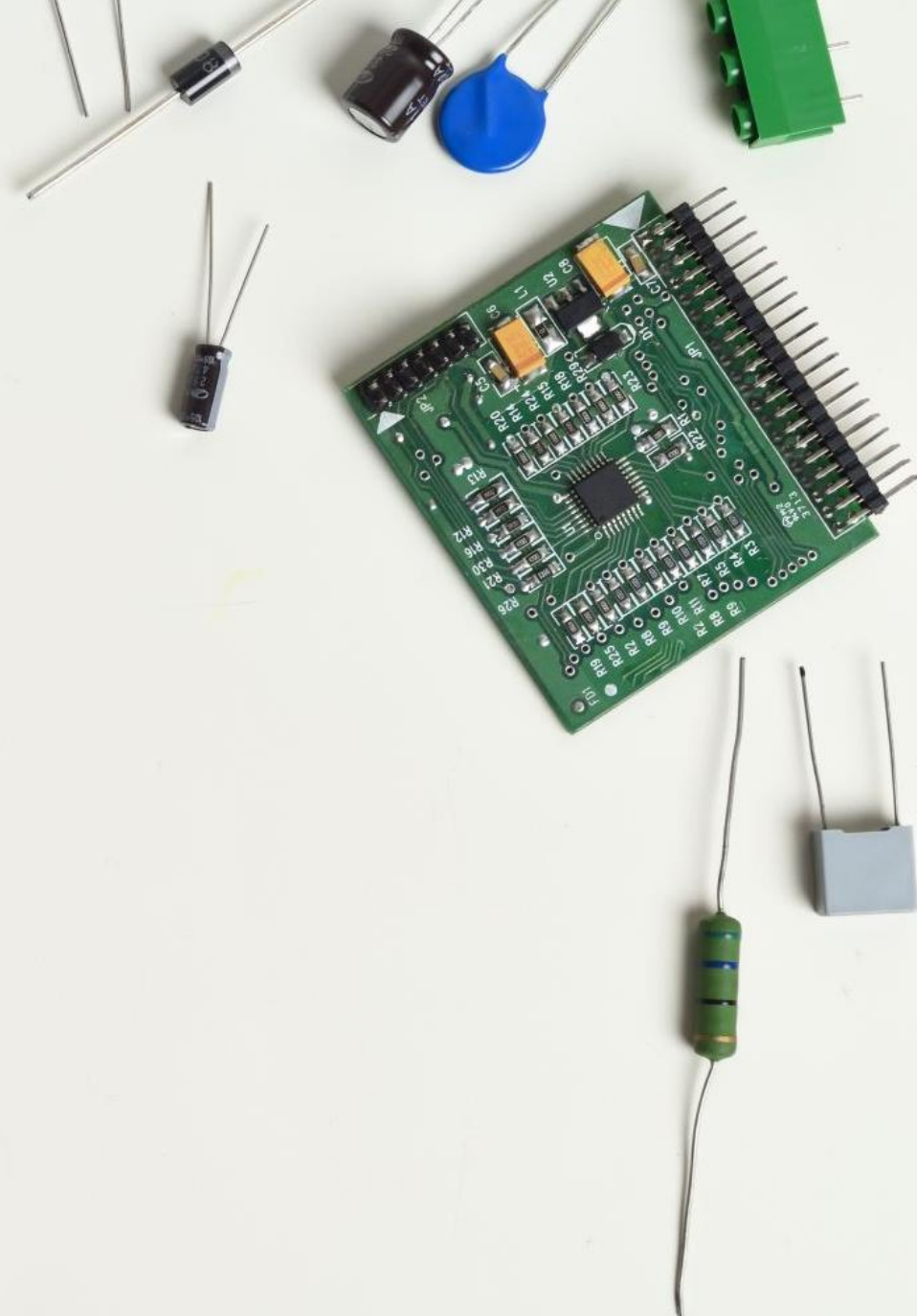


# Aula 7

- **Ligar o ESP ao Wifi – CRIAR HOTSPOT NO TELEMOVEL OU PC**



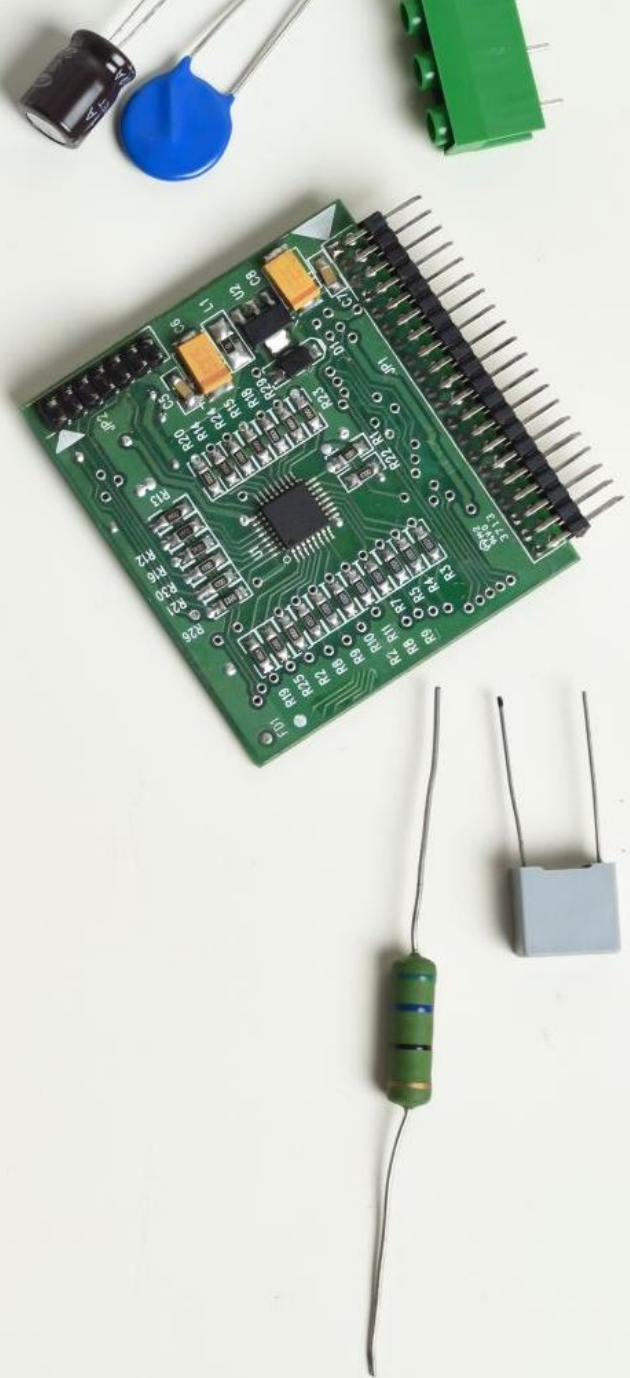
```
1 import network
2 import time
3
4 # Substitua pelos dados da sua rede Wi-Fi
5 SSID = 'NOME_DA_SUA_REDE'
6 SENHA = 'SENHA_DO_WIFI'
7
8 # Cria o objeto da interface Wi-Fi
9 wlan = network.WLAN(network.STA_IF)
10 wlan.active(True)          # Ativa o modo cliente (station)
11
12 # Conecta à rede Wi-Fi
13 if not wlan.isconnected():
14     print('Conectando ao Wi-Fi...')
15     wlan.connect(SSID, SENHA)
16
17     # Aguarda conexão
18     while not wlan.isconnected():
19         time.sleep(1)
20
21 print('Conectado com sucesso!')
22 print('Endereço IP:', wlan.ifconfig()[0])
```



# Aula 7

- Criar um webserver para controlar o LED interno do ESP

```
main.py * x
1 import network          # Biblioteca para rede Wi-Fi
2 import socket           # Biblioteca para comunicação de rede (servidor web)
3 from machine import Pin # Controlo de GPIOs (entradas/saídas digitais)
4 import time             # Biblioteca para pausas e contagem de tempo
5
6 # Função para LIGAR o ESP32 ao Wi-Fi
7 def conecta_wifi(ssid, senha):
8     wlan = network.WLAN(network.STA_IF) # Cria interface de rede no modo Station (cliente Wi-Fi)
9     wlan.active(True)                  # Ativa a interface Wi-Fi
10
11     # Se ainda não está a ligar, continua a tentar
12     if not wlan.isconnected():
13         print("Conectando ao Wi-Fi...")
14         wlan.connect(ssid, senha)
15
16         # Aguarda até obter ligação
17         while not wlan.isconnected():
18             time.sleep(1)
19
20     # Exibe o IP recebido via DHCP
21     print("Ligado! IP:", wlan.ifconfig()[0])
22     return wlan.ifconfig()[0] # Retorna o IP para uso posterior
23
24 # LED interno
25 led = Pin(2, Pin.OUT)
26 led.off() # Começa desligado
```

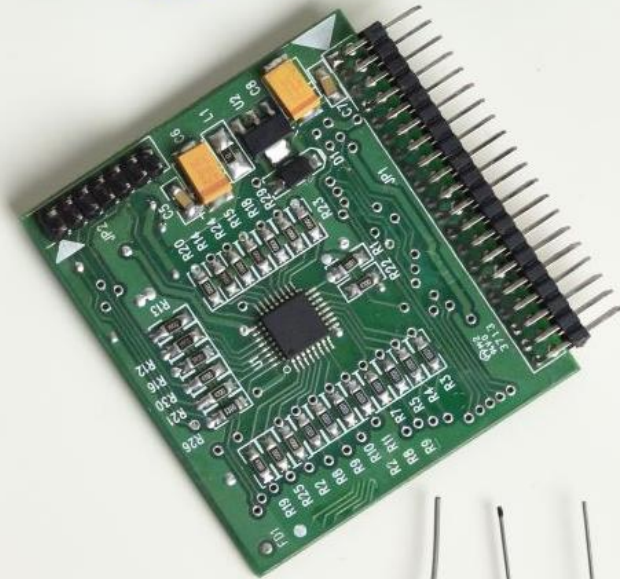




# Aula 7

- Criar um webserver para controlar o LED interno do ESP

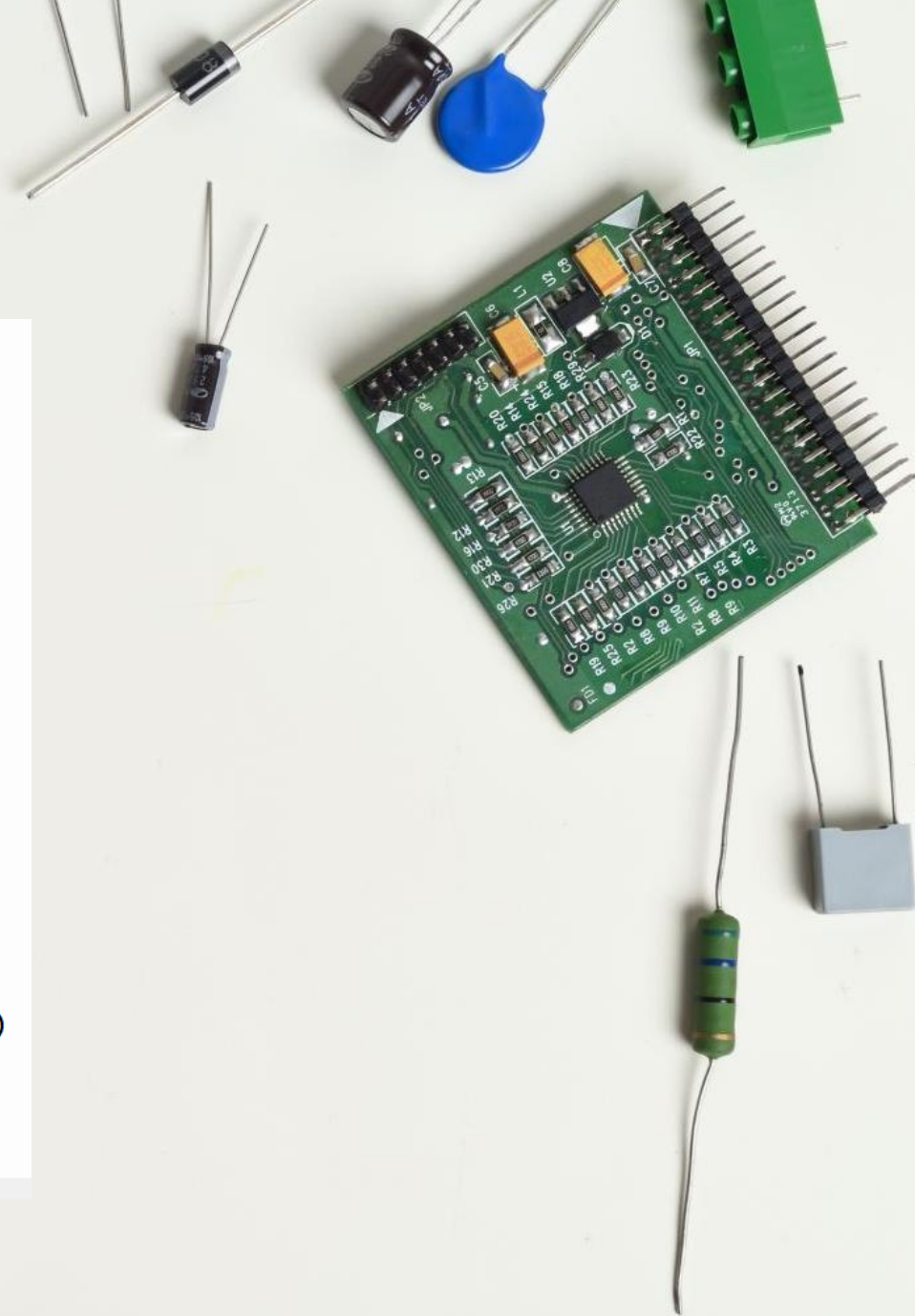
```
28 # Função que gera a página HTML conforme o estado atual do LED
29 def gerar_html(estado_led):
30     botao_texto = "Desligar" if estado_led else "Ligar" #texto botão
31     botao_valor = "off" if estado_led else "on" #valor enviado
32     return f"""<!DOCTYPE html>
33 <html>
34 <head><title>Controlo LED ESP32</title></head>
35 <body>
36     <h1>Controlo do LED</h1>
37     <p>Estado atual: {"Ligado" if estado_led else "Desligado"}</p>
38     <form method="get" action="/">
39         <button type="submit" name="led" value="{botao_valor}">{botao_texto}</button>
40     </form>
41 </body>
42 </html>"""
43
44 # Liga ao Wi-Fi
45 ip = conecta_wifi("NOME_REDE", "PASSWORD")
46
47 # Inicializa servidor
48 s = socket.socket()
49 s.bind((ip, 80))
50 s.listen(1)
51
52 print(f"Servidor a rodar em http://{ip}")
53
```



# Aula 7

- Criar um webserver para controlar o LED interno do ESP

```
54 # Loop principal
55 while True:
56     cl, addr = s.accept()
57     request = cl.recv(1024).decode()
58     print('Requisição de:', addr)
59
60     # Processa o parâmetro da URL
61     if 'led=on' in request:
62         led.on()
63     elif 'led=off' in request:
64         led.off()
65
66     # Atualiza o HTML com o estado atual do LED
67     html = gerar_html(led.value())
68
69     # Envia resposta HTTP
70     cl.send("HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n")
71     cl.send(html)
72     cl.close()
73
```





# Aula 7

- Criar um webserver para controlar o LED interno do ESP

