Roteiro Nº 02	Estrutura de controle de fluxo de programas – if, else, for e while	
Curso Engenharia	**Disciplina** ELT 1119 – Redes e Aplicações IoT	**Professor** Carlos Alberto Vasconcelos Bezerra
Nome do Estudante		**Data**

1. Objetivos da Aprendizagem

Explorar o uso de estruturas de controle de fluxo (if, else, for, while) em MicroPython utilizando um botão de entrada no pino GP5 e um LED no pino GP2.

2. Material Necessário

2.1 Equipamentos: 01 Matriz de contatos; 01 Multímetro; Computador; Kit de desenvolvimento ESP32-WROOM 32U.

.

2.2 Insumos: 01 Resistor: 220Ω /1W; 01 LED 5 mm; 01 Botão táctil.

3. Fundamentação Teórica

4. Procedimento Experimental

Preparação:

- Conecte o LED ao pino GP2.
- Conecte o botão ao pino GP5 e ao GND, configurando o pino GP5 com PULL_UP internamente.
- Verifique se o ambiente de desenvolvimento Thonny está pronto para a programação em MicroPython.

Digite os programas abaixo na IDE Thonny e verifique a operação deles no ESP32.

Exemplo 1: Controle de LED com if e else
Neste exemplo, o LED acenderá quando o botão for pressionado e apagará quando solto.

```
from machine import Pin
import time

# Configuração do LED e do botão
led = Pin(2, Pin.OUT)
botao = Pin(5, Pin.IN, Pin.PULL_UP)

while True:
    if botao.value() == 0:  # Botão pressionado
        led.on()
    else:  # Botão solto
        led.off()
    time.sleep(0.1)
```

Exemplo 2: Controle de LED com for
Aqui, o LED piscará um número determinado de vezes ao pressionar o botão.

```
from machine import Pin
import time

led = Pin(2, Pin.OUT)
botao = Pin(5, Pin.IN, Pin.PULL_UP)

while True:
    if botao.value() == 0:  # Botão pressionado
        for i in range(5):  # Pisca 5 vezes
            led.on()
            time.sleep(0.2)
            led.off()
            time.sleep(0.2)
    time.sleep(0.1)
```

Exemplo 3: Controle de LED com while
Este exemplo faz o LED piscar continuamente enquanto o botão estiver pressionado.

```
from machine import Pin
import time

led = Pin(2, Pin.OUT)
botao = Pin(5, Pin.IN, Pin.PULL_UP)

while True:
    while botao.value() == 0:  # Enquanto o botão estiver pressionado
        led.on()
        time.sleep(0.2)
        led.off()
        time.sleep(0.2)
    time.sleep(0.1)
```

Exemplo 4: Controle de LED com if, for e while
Neste exemplo, o LED acenderá de forma contínua por um número de vezes baseado em quantas vezes o botão foi pressionado.

```
from machine import Pin
import time

led = Pin(2, Pin.OUT)
botao = Pin(5, Pin.IN, Pin.PULL_UP)
contador = 0
ultimo_clique = time.ticks_ms()  # Marca o tempo do último clique

while True:
    # Se o botão for pressionado e solto, incrementa o contador
    if botao.value() == 0:  # Botão pressionado
        while botao.value() == 0:  # Aguarda o botão ser solto
            pass
        contador += 1  # Incrementa a contagem
        ultimo_clique = time.ticks_ms()  # Atualiza o tempo do último clique
        print(f"Contagem atual: {contador}")  # Exibe o valor contado
        time.sleep(0.1)  # Pequeno debounce para evitar contagem dupla

    # Se já passou 2 segundos sem pressionar o botão, inicia a sequência de piscadas
    if contador > 0 and time.ticks_diff(time.ticks_ms(), ultimo_clique) > 2000:
        print(f"Iniciando piscadas: {contador} vezes")  # Exibe antes de piscar
        for i in range(contador):  # LED pisca conforme a contagem acumulada
            led.on()
            time.sleep(0.3)
            led.off()
            time.sleep(0.3)

        print("Fim da sequência de piscadas\n")  # Exibe mensagem ao final
        contador = 0  # Reseta o contador após as piscadas
```

Exercícios Propostos:
Questão 1:
Modifique o Exemplo 2 para que, após o botão ser pressionado e o LED piscar 5 vezes, o LED permaneça aceso por 2 segundos e depois apague.

- Dica: Utilize uma combinação da estrutura for para controlar as piscadas e uma simples função time.sleep() para o tempo adicional.

Questão 2:
No Exemplo 3, o LED pisca continuamente enquanto o botão estiver pressionado. Adapte o código para que, após o botão ser pressionado por 3 segundos seguidos, o LED fique aceso continuamente, mesmo após soltar o botão. Um novo toque no botão deve apagar o LED.

- Dica: Você pode usar uma variável para armazenar o estado atual do LED (ligado ou desligado) e usar o tempo de pressão para controlar a mudança.

Questão 3:

Adapte o Exemplo 4 para que o LED pisque o número de vezes correspondente ao número de pressões do botão, mas faça com que a duração das piscadas aumente progressivamente a cada ciclo. A primeira piscada dura 0.2 segundos, a segunda 0.4 segundos, e assim por diante.

- Dica: Use a variável do contador para aumentar o tempo de cada piscada dentro do loop for.

6 Simulação

Simular os exercícios propostos utilizando o simulador wokwi no link:

https://wokwi.com/projects/new/esp32

Bibliografia

ASCHER, D.; LUTZ, M. Aprendendo Python. Porto Alegre: Bookman, 2007.

MENEZES, N. N. C. Introdução à programação com Python. São Paulo: Ed. Novatec, 2014.

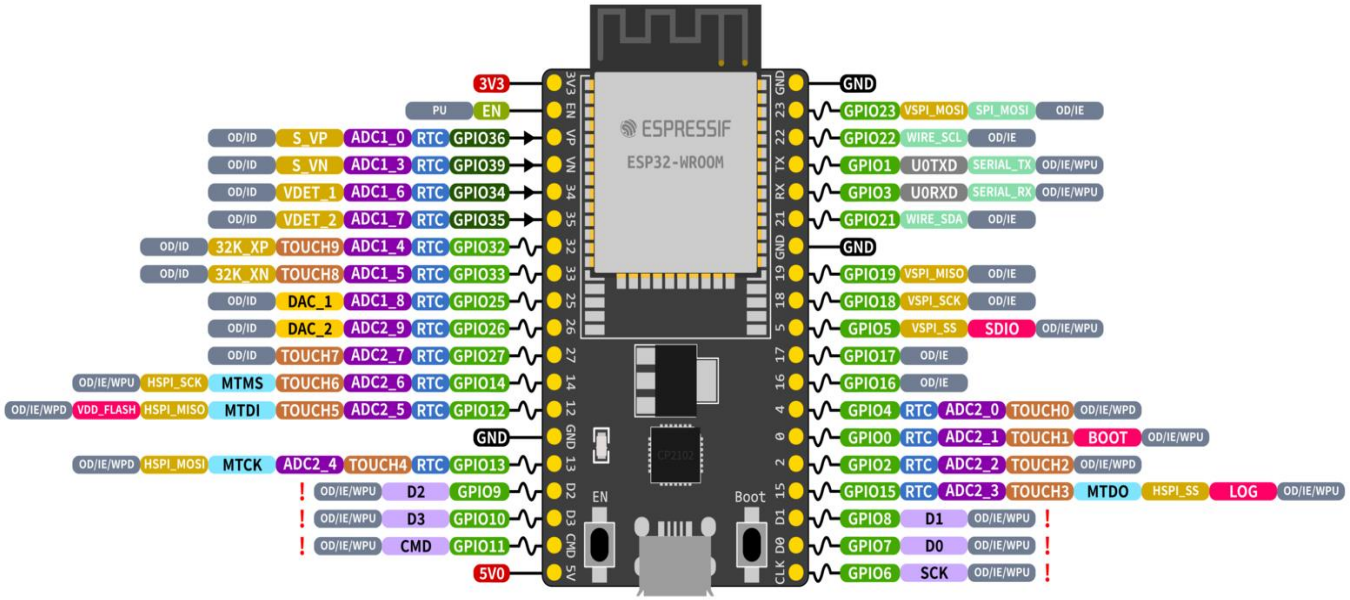
OLIVEIRA, S. Internet das coisas com ESP8266, Arduino e Raspberry Pi. São Paulo: Ed. Novatec, 2017.

OLIVEIRA, C. L. V. IoT com Micropython e NodeMCU. São Paulo: Ed. Novatec, 2022.

Pinos do ESP32 – WROOM 32U

ESP32-DevKitC

ESPRESSIF



ESP32 Specs

32-bit Xtensa® dual-core @240MHz
Wi-Fi IEEE 802.11 b/g/n 2.4GHz
BLuetooth 4.2 BR/EDR and BLE
520 KB SRAM (16 KB for cache)
448 KB ROM
34 GPIOs, 4x SPI, 3x UART, 2x I2C,
2x I2S, RMT, LED PWM, 1 host SD/eMMC/SDIO,
1 slave SDIO/SPI, TWAI®, 12-bit ADC, Ethernet

PWM Capable Pin
GPIOX GPIO Input Only
GPIOX GPIO Input and Output
DAC_X Digital-to-Analog Converter
DEBUG JTAG for Debugging
FLASH External Flash Memory (SPI)
ADCX_CH Analog-to-Digital Converter
TOUCHX Touch Sensor Input Channel
OTHER Other Related Functions
SERIAL Serial for Debug/Programming
ARDUINO Arduino Related Functions
STRAP Strapping Pin Functions

RTC RTC Power Domain (VDD3P3_RTC)
GND Ground
PWD Power Rails (3V3 and 5V)
! Pin Shared with the Flash Memory
Can't be used as regular GPIO

GPIO STATE
WPU: *Weak Pull-up (Internal)*
WPD: *Weak Pull-down (Internal)*
PU: *Pull-up (External)*
IE: *Input Enable (After Reset)*
ID: *Input Disabled (After Reset)*
OE: *Output Enable (After Reset)*
OD: *Output Disabled (After Reset)*