

|                         |   |   |
|-------------------------|---|---|
| Roteiro Nº<br><b>01</b> | <b>Introdução ao Thonny e ESP32</b>             |   |
| Curso<br>Engenharia     | Disciplina<br>ELT 1119 – Redes e Aplicações IoT | Professor<br>Carlos Alberto Vasconcelos Bezerra |
| Nome do Estudante       |   | Data  |

## 1 Objetivos de Aprendizagem

- Familiarizar-se com o ambiente de desenvolvimento Thonny.
- Conectar e configurar o ESP32 para programação em MicroPython.
- Escrever e executar um programa simples para controlar um LED.
- Compreender conceitos básicos de GPIO e loops em MicroPython.

## 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;

## 3 Fundamentação Teórica

## 4 Procedimento Experimental

### 1. Introdução ao Thonny

- **Acesse o Thonny IDE:** Abra o Thonny clicando no ícone do programa.
- **Explorando a Interface:**
  - **Editor:** Área onde o código é escrito.
  - **Shell:** Área onde você pode ver a saída do código e executar comandos diretos.
  - **Barra de Ferramentas:** Contém botões para rodar, parar, e depurar o código.
- **Configuração do Thonny para ESP32:**
  - Vá para Tools > Options > Interpreter.
  - Selecione MicroPython (ESP32) na lista de intérpretes.
  - Certifique-se de que o dispositivo está conectado via USB e se o Driver CP2102 está instalado.

### 2. Conectando o ESP32

- **Passo 1:** Conecte o ESP32 ao computador usando um cabo USB.
- **Passo 2:** Verifique se o Thonny reconheceu o Raspberry Pi Pico (isso será indicado na barra inferior do Thonny).
- **Passo 3:** No Shell, digite o comando `print("Hello, World!")` e pressione Enter para garantir que a comunicação está funcionando.

### 3. Primeiro Programa: Piscar um LED

- **Passo 1:** Crie um novo arquivo no Thonny.
- **Passo 2:** Digite o seguinte código:

```
from machine import Pin
import time
# Configura o pino GPIO2 como saída
led = Pin(2, Pin.OUT)

while True:
    led.value(not led.value()) # Inverte o estado do LED
    time.sleep(1)             # Aguarda 1 segundo
```

- **Passo 3:** Salve o arquivo com o nome blink.py.
- **Passo 4:** Execute o programa clicando no botão Run (ícone de "play").
- **Passo 5:** Observe o LED no ESP32 piscando.
- **Passo 6:** Modifique o tempo de sleep(1) para sleep(0.5) e execute novamente para observar a mudança na frequência do piscar.

#### 4. Exploração

- **Modificação do Código:**
  - **Modificar o código para controlar o LED ao pulsar um botão colocado no pino 0 do ESP32, ou seja pulsa o botão o LED irá piscar, pulsando novamente o LED apagará.**

```
from machine import Pin
import time
```

```
# Configura o pino GPIO2 como saída para o LED
led = Pin(2, Pin.OUT)
```

```
# Configura o pino GPIO0 como entrada com pull-up interno para o botão
botao = Pin(0, Pin.IN, Pin.PULL_UP)
```

```
# Variáveis para armazenar o estado atual
led_piscando = False
botao_pressionado = False
```

```
# Função para alternar o estado do LED
def alternar_led():
    led.value(not led.value())
```

```
# Loop principal
while True:
    if not botao.value(): # Verifica se o botão está pressionado (nível lógico baixo)
        if not botao_pressionado: # Verifica se o botão não estava previamente pressionado
            botao_pressionado = True
            led_piscando = not led_piscando # Alterna o estado de piscagem do LED
            if not led_piscando:
                led.off() # Se parar de piscar, garante que o LED esteja desligado
        else:
            botao_pressionado = False # Reseta o estado do botão quando não está pressionado

    if led_piscando:
        alternar_led() # Alterna o estado do LED
        time.sleep(0.5) # Aguarda 500 ms
    else:
        time.sleep(0.1) # Aguarda 100 ms para evitar uso excessivo da CPU
```

## 5 Questões

### Exercício 1: Piscar LEDs

Conectar um LED ao ESP32 e fazer com que ele pisque, quando uma chave estiver fechada e pare de piscar ao abrir a chave.

