Roteiro Nº 09	**Node-RED e ESP32 via Requisições HTTP**	
Curso Engenharia Elétrica	**Disciplina** ELT 1119 – Redes e Aplicações IoT	**Professor** Carlos Alberto Vasconcelos Bezerra
Nome do Estudante		**Data**

1. Objetivos da aula

- Compreender como o **Node-RED** pode ser usado como ferramenta de automação visual para controle de dispositivos embarcados.
- Integrar o ESP32 a uma interface de controle no Node-RED via **requisições HTTP**.
- Controlar fisicamente um **LED conectado ao ESP32** com comandos enviados por fluxo no Node-RED.
- Interpretar o funcionamento de um servidor HTTP embarcado.

2. Materiais utilizados

- 1 placa ESP32
- 1 LED externo
- 1 resistor de 220 Ω
- Jumpers
- 1 protoboard
- Cabo USB para comunicação e alimentação
- Computador com:
  - Python instalado
  - Thonny IDE
  - Node.js
  - Node-Red
  - Acesso à internet

3. Resumo teórico

O Node-RED é uma ferramenta de programação visual desenvolvida pela IBM, utilizada para conectar dispositivos físicos, APIs e serviços online de maneira modular e intuitiva. Ele permite que sistemas embarcados, como o ESP32, sejam facilmente integrados a sistemas maiores de automação, utilizando protocolos como HTTP e MQTT.

Nesta aula, será utilizado o ESP32 com MicroPython atuando como um servidor HTTP local, capaz de interpretar requisições feitas a URLs específicas. O Node-RED, por sua vez, será utilizado para emitir essas requisições HTTP através de nós de injeção e requisição, permitindo o controle de um LED conectado ao pino GPIO 5 da placa.

Para isto será necessário:

- Estruturar fluxos no Node-RED para realizar automação com ESP32.
- Enviar comandos HTTP para um servidor embarcado.
- Visualizar e interpretar o retorno dessas ações em tempo real.

4. Desenvolvimento da aula

4.1 Montagem do Circuito

- Monte o circuito conectando o **LED ao GPIO 5 do ESP32** com resistor limitador de corrente (330 Ω).
- Cátodo do LED no GND.

4.2 Programação do ESP32

- Programe o ESP32 com o código MicroPython (anexo1) que:
  - Conecta à rede Wi-Fi.
  - Cria um servidor HTTP.
  - Responde às URLs /ligar e /desligar acionando o LED.

4.3 Descoberta do IP

- Verifique o IP local do ESP32 exibido no terminal após a conexão.

4.4 Configuração do Node-RED

- Abra o ambiente Node-RED no navegador (http://localhost:1880 ou IP correspondente).
- Crie um novo fluxo (anexo 2) contendo os seguintes nós:
  - Dois nós inject: um com payload "Liga LED" e outro com "Desliga LED".
  - Dois nós http request configurados para enviar GET para:
    - http://<IP_DO_ESP32>/ligar
    - http://<IP_DO_ESP32>/desligar

4.5 Teste no Node-RED

- Clique nos botões dos nós inject e observe o acionamento do LED físico.
- Monitore no painel debug se há resposta como HTML ou texto simples.

5. Códigos utilizados (em anexo)

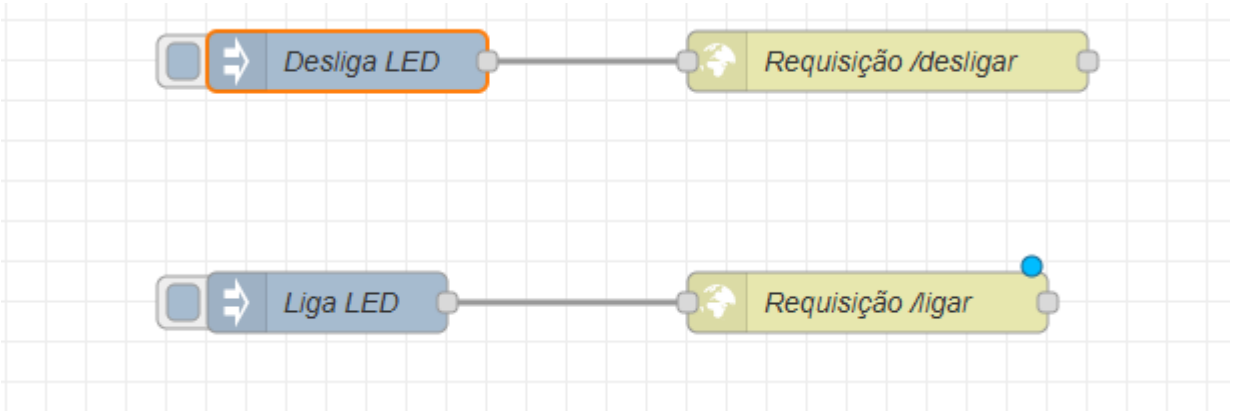
Anexo 1

```
import network
import socket
from machine import Pin

# ============================
# CONFIGURAÇÕES DE REDE
# ============================
SSID = 'Alencar Oi 2.4G'        # Substitua pelo nome da sua rede Wi-Fi
SENHA = 'Carloshenrique'       # Substitua pela senha do seu Wi-Fi

# Conecta ao Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(SSID, SENHA)

print("Conectando ao Wi-Fi...")
while not wifi.isconnected():
    pass

print("Conectado com sucesso!")
ip_local = wifi.ifconfig()[0]
print("Endereço IP:", ip_local)

# ============================
# CONFIGURAÇÃO DO LED
```

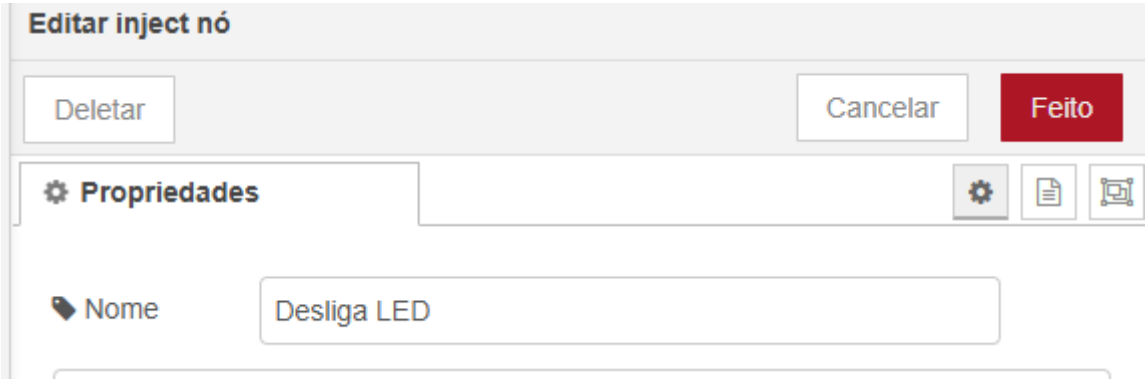
```
# ============================
led = Pin(5, Pin.OUT)

# ============================
# FUNÇÃO HTML DA PÁGINA
# ============================
def pagina_html():
    estado = "Ligado" if led.value() else "Desligado"
    html = f"""<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Controle de LED</title>
</head>
<body style="font-family:sans-serif; text-align:center; padding-top:50px;">
    <h1>Controle do LED (GPIO 5)</h1>
    <h2>Estado atual: <strong>{estado}</strong></h2>
    <a href="/ligar"><button style="font-size:24px; padding:10px;">LIGAR</button></a>
    <a href="/desligar"><button style="font-size:24px; padding:10px;">DESLIGAR</button></a>
</body>
</html>"""
    return html

# ============================
# SERVIDOR WEB
# ============================
servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servidor.bind(('', 80))
servidor.listen(1)

print("\nServidor web iniciado!")
print("Acesse no navegador ou pelo Node-RED:")
print(f"http://{ip_local}/ligar")
print(f"http://{ip_local}/desligar")

while True:
    conexao, endereco = servidor.accept()
    print("Cliente conectado:", endereco)
    requisicao = conexao.recv(1024).decode()
    print("Requisição recebida:", requisicao)

    if 'GET /ligar' in requisicao:
        led.on()
    elif 'GET /desligar' in requisicao:
        led.off()

    resposta = pagina_html()

    # Cabeçalhos HTTP com UTF-8 corretamente especificado
    conexao.send('HTTP/1.1 200 OK\r\n')
```

```
conexao.send('Content-Type: text/html; charset=utf-8\r\n')
conexao.send('Connection: close\r\n\r\n')
conexao.sendall(resposta)
conexao.close()
```

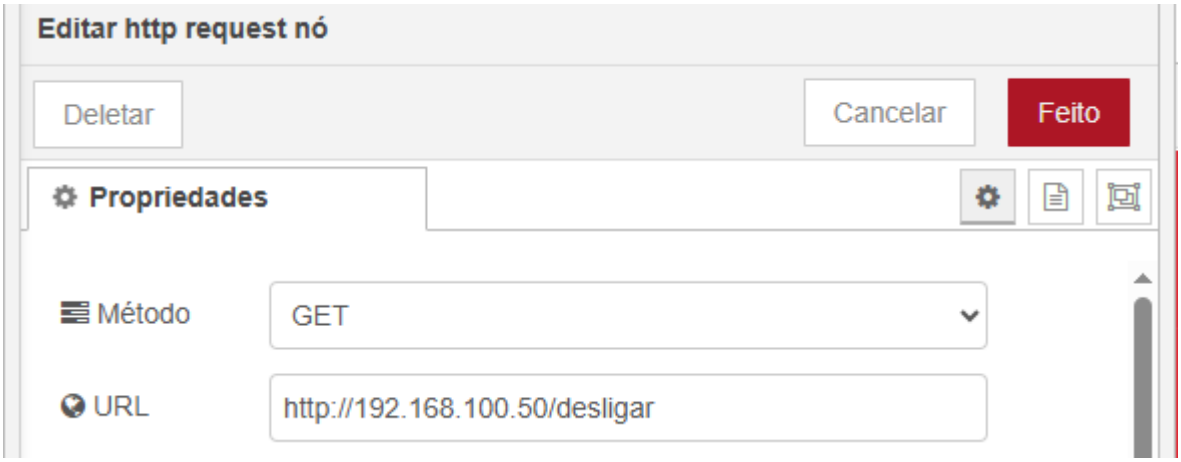
Anexo 2



Nó Inject



Nó http request



Anexo 3

Instalação do Node-RED

Antes de instalar o Node-RED, é necessário ter o **Node.js** instalado no sistema.
Node.js é o ambiente de execução JavaScript necessário para rodar o Node-RED.

Instalação no Windows, Linux ou macOS (via terminal)

1. Instale o Node.js

- Acesse: https://nodejs.org
- Baixe e instale a versão LTS (Long Term Support)

2. Instale o Node-RED com o npm
Após instalar o Node.js, abra o terminal (ou PowerShell no Windows) e digite:

npm install -g --unsafe-perm node-red

3. Como iniciar o Node-RED
Após a instalação, execute:

node-red
Isso iniciará o servidor e exibirá no terminal o endereço local, normalmente:

http://127.0.0.1:1880

4.Abra esse link no navegador para acessar o editor visual.

Bibliografia

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

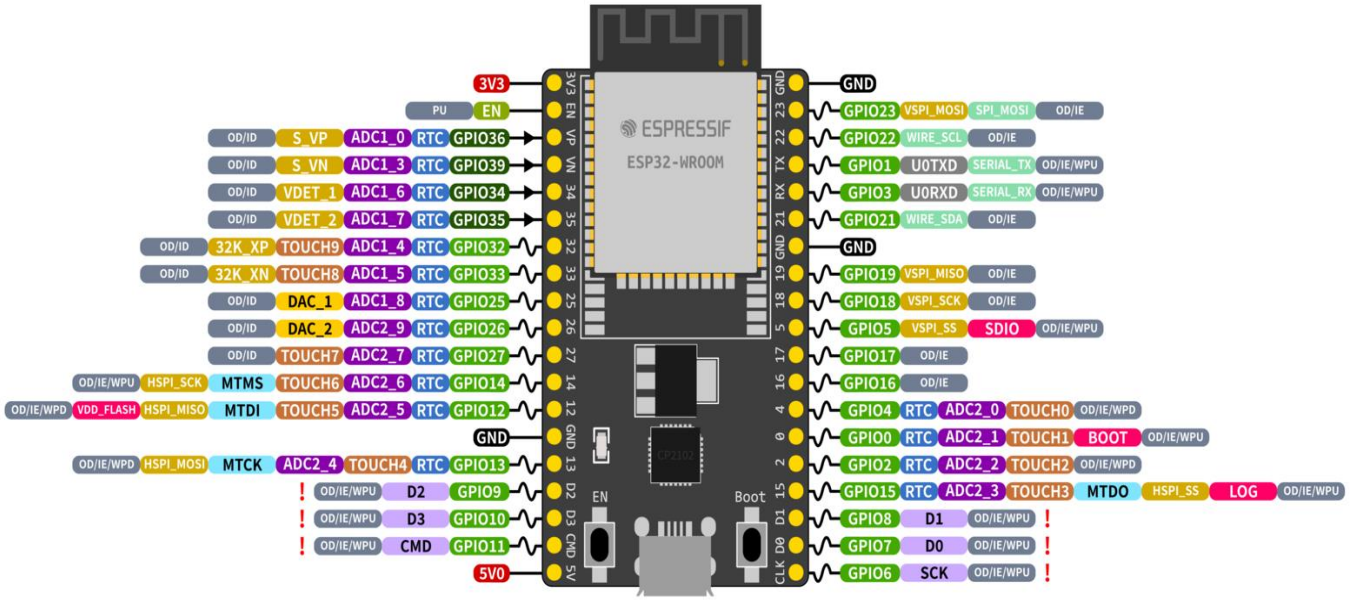
MENEZES, N. N. C. Introdução à programação com Python. São Paulo: Ed. Novatec, 2014.

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.

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)*