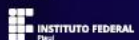


Aplicações com Microcontroladores

Unidade 2 | Capítulo 3

Executores:



Coordenação:



Iniciativa:



Sumário



Objetivos



Revisão



Exemplo Prático



Exemplo de Código



Conclusão





Objetivo

- Desenvolver aplicações IoT utilizando microcontroladores e protocolos de comunicação.
- Realizar práticas de aplicações IoT

Revisão

- Nas últimas aulas você aprendeu:
 - Compreender as características e funcionamento da interface de comunicação sem fio
 - Desenvolver rotinas de configuração de interfaces sem fio, codificando-as para o microcontrolador
 - Avaliar e implementar protocolos para comunicação em IoT

Exemplo Prático

-Servidor Web

- Hospeda a aplicação.
- Gerencia as requisições HTTP feitas pelos usuários.

-Sensores e Atuadores

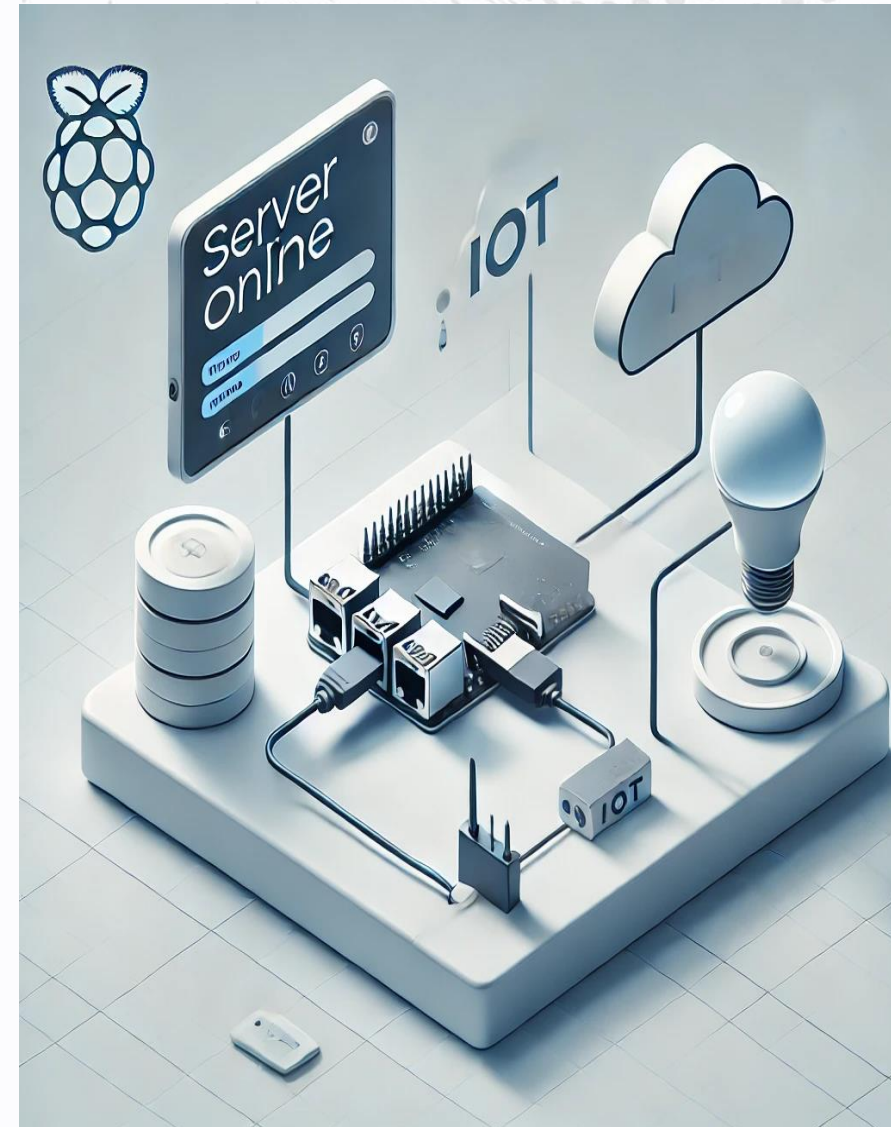
- Conexão via GPIOs ou interfaces como I2C, SPI ou UART.
- Sensores: Capturam informações do ambiente.
- Atuadores: Executam ações físicas baseadas em comandos recebidos.

-Interface Web

- Ponto de interação do usuário com o sistema.
- Permite a visualização de dados coletados.
- Facilita o envio de comandos para controlar os atuadores.

-Rede de Comunicação

- Conecta todos os elementos do sistema.
- Permite acesso ao servidor por dispositivos na mesma rede local.
- Suporte a acesso remoto com configurações de segurança adequadas.



Exemplos de Códigos

- Configuração básica de um novo programa

Basic Settings

Name:

Board type:

Location:

Select Pico SDK version:

- Digite o nome do seu programa
- Selecione a Pico W
- Selecione a versão do SDK

Stdio support

☐ Console over UART ☒ Console over USB (disables other USB use)

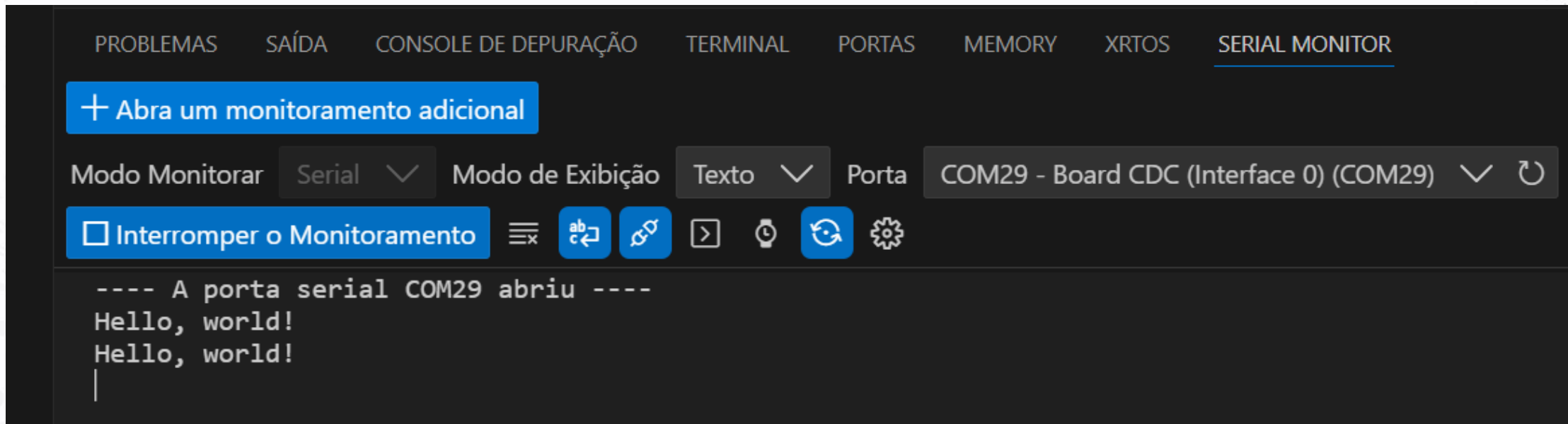
Pico wireless options

☐ None ☒ Pico W onboard LED ☐ Polled lwIP ☐ Background lwIP

- Selecione o Console over USB
- Selecione Pico W onboard LED

Exemplos de Códigos

- Sugestão: Compile o código gerado e verifique se o texto “Hello world” foi impresso no terminal.



PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO TERMINAL PORTAS MEMORY XRTOS SERIAL MONITOR

+ Abra um monitoramento adicional

Modo Monitorar Serial ▼ Modo de Exibição Texto ▼ Porta COM29 - Board CDC (Interface 0) (COM29) ▼ ↺

☐ Interromper o Monitoramento ☰ 🔍 🔗 > ⌚ ↺ ⚙️

```
---- A porta serial COM29 abriu ----
Hello, world!
Hello, world!
|
```

Importação de Bibliotecas

```
#include "pico/stdlib.h"  
#include "hardware/adc.h"  
#include "pico/cyw43_arch.h"  
#include <stdio.h>  
#include <string.h>  
#include <stdlib.h>  
#include "lwip/pbuf.h"  
#include "lwip/tcp.h"  
#include "lwip/netif.h"
```


Configurações Iniciais

```
#define WIFI_SSID "Multilaser2"  
#define WIFI_PASSWORD "123456789"  
  
#define LED_PIN CYW43_WL_GPIO_LED_PIN  
#define LED_BLUE_PIN 12  
#define LED_GREEN_PIN 11  
#define LED_RED_PIN 13
```

Função de Callback HTTP

```
// Função de callback para processar requisições HTTP recebidas.
static err_t tcp_server_recv(void *arg, struct tcp_pcb *tpcb, struct pbuf *p, err_t err) {
    if (!p) { // Verifica se o pacote recebido é nulo (conexão encerrada).
        tcp_close(tpcb); // Fecha a conexão TCP.
        tcp_recv(tpcb, NULL); // Desabilita o recebimento de dados.
        return ERR_OK;
    }
    // Copia a requisição HTTP para uma string alocada dinamicamente.
    char *request = (char *)malloc(p->len + 1);
    memcpy(request, p->payload, p->len);
    request[p->len] = '\0';
    printf("Request: %s\n", request); // Exibe a requisição recebida.
    // Controle dos LEDs com base na URL requisitada.
    if (strstr(request, "GET /blue_on") != NULL) // Liga o LED azul.
        gpio_put(LED_BLUE_PIN, 1);
    else if (strstr(request, "GET /blue_off") != NULL) // Desliga o LED azul.
        gpio_put(LED_BLUE_PIN, 0);
    else if (strstr(request, "GET /green_on") != NULL) // Liga o LED verde.
        gpio_put(LED_GREEN_PIN, 1);
    else if (strstr(request, "GET /green_off") != NULL) // Desliga o LED verde.
        gpio_put(LED_GREEN_PIN, 0);
    else if (strstr(request, "GET /red_on") != NULL) // Liga o LED vermelho.
        gpio_put(LED_RED_PIN, 1);
    else if (strstr(request, "GET /red_off") != NULL) // Desliga o LED vermelho.
        gpio_put(LED_RED_PIN, 0);
    else if (strstr(request, "GET /on") != NULL) // Liga o LED embutido.
        cyw43_arch_gpio_put(LED_PIN, 1);
    else if (strstr(request, "GET /off") != NULL) // Desliga o LED embutido.
        cyw43_arch_gpio_put(LED_PIN, 0);
}
```

```
// Leitura da temperatura interna usando o ADC.
adc_select_input(4); // Seleciona o canal 4 do ADC para o sensor interno.
uint16_t raw_value = adc_read(); // Lê o valor do ADC.
const float conversion_factor = 3.3f / (1 << 12); // Fator de conversão do ADC.
float temperature = 27.0f - ((raw_value * conversion_factor) - 0.706f) / 0.001721f;
// Criação da resposta HTML com os controles e a temperatura interna.
char html[1024];
snprintf(html, sizeof(html),
    "HTTP/1.1 200 OK\r\n" "Content-Type: text/html\r\n" "\r\n" "<!DOCTYPE html>\n"
    "<html>\n" "<head>\n" "<title>LED Control</title>\n" "<style>\n"
    "body { font-family: Arial, sans-serif; text-align: center; margin-top: 50px; }\n"
    "h1 { font-size: 64px; margin-bottom: 30px; }\n"
    "button { font-size: 36px; margin: 10px; padding: 20px 40px; border-radius: 10px; }\n"
    ".temperature { font-size: 48px; margin-top: 30px; color: #333; }\n"
    "</style>\n" "</head>\n" "<body>\n" "<h1>LED Control</h1>\n"
    "<form action=\"./blue_on\"><button>Ligar Azul</button></form>\n"
    "<form action=\"./blue_off\"><button>Desligar Azul</button></form>\n"
    "<form action=\"./green_on\"><button>Ligar Verde</button></form>\n"
    "<form action=\"./green_off\"><button>Desligar Verde</button></form>\n"
    "<form action=\"./red_on\"><button>Ligar Vermelho</button></form>\n"
    "<form action=\"./red_off\"><button>Desligar Vermelho</button></form>\n"
    "<p class=\"temperature\">Temperatura Interna: %.2f &deg;C</p>\n"
    "</body>\n" "</html>\n", temperature);
tcp_write(tpcb, html, strlen(html), TCP_WRITE_FLAG_COPY); // Envia a resposta ao cliente.
tcp_output(tpcb); // Força o envio imediato.
free(request); // Libera a memória alocada para a requisição.
pbuf_free(p); // Libera o buffer de pacotes.
return ERR_OK;
```

Novas Conexões

```
static err_t tcp_server_accept(void *arg, struct tcp_pcb *newpcb, err_t err)
{
    tcp_recv(newpcb, tcp_server_recv);
    return ERR_OK;
}
```

Função Principal

```
// Função principal.
int main()
{
    stdio_init_all(); // Inicializa a entrada/saída padrão.
    // Configuração dos pinos dos LEDs como saída.
    gpio_init(LED_BLUE_PIN);
    gpio_set_dir(LED_BLUE_PIN, GPIO_OUT);
    gpio_put(LED_BLUE_PIN, false);
    gpio_init(LED_GREEN_PIN);
    gpio_set_dir(LED_GREEN_PIN, GPIO_OUT);
    gpio_put(LED_GREEN_PIN, false);
    gpio_init(LED_RED_PIN);
    gpio_set_dir(LED_RED_PIN, GPIO_OUT);
    gpio_put(LED_RED_PIN, false);
    while (cyw43_arch_init()) { // Inicializa o Wi-Fi.
        printf("Falha ao inicializar Wi-Fi\n");
        sleep_ms(100);
        return -1;    }
    cyw43_arch_gpio_put(LED_PIN, 0);
    // Configura o Wi-Fi no modo estação.
    cyw43_arch_enable_sta_mode();
    printf("Conectando ao Wi-Fi...\n");
```

```
while (cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD,
CYW43_AUTH_WPA2_AES_PSK, 20000)) { // Tenta conectar ao Wi-Fi.
    printf("Falha ao conectar ao Wi-Fi\n");
    sleep_ms(100);
    return -1;    }
printf("Conectado ao Wi-Fi\n");
if (netif_default) // Exibe o IP atribuído ao dispositivo.
    printf("IP do dispositivo: %s\n", ipaddr_ntoa(&netif_default->ip_addr));
// Configura o servidor TCP na porta 80.
struct tcp_pcb *server = tcp_new();
if (!server) {
    printf("Falha ao criar servidor TCP\n");
    return -1;    }
if (tcp_bind(server, IP_ADDR_ANY, 80) != ERR_OK) {
    printf("Falha ao associar servidor TCP à porta 80\n");
    return -1;    }
server = tcp_listen(server);
tcp_accept(server, tcp_server_accept);
printf("Servidor ouvindo na porta 80\n");
adc_init(); // Inicializa o ADC.
adc_set_temp_sensor_enabled(true); // Habilita o sensor de temperatura interno.
while (true) { // Loop principal para manter o servidor ativo.
    cyw43_arch_poll(); } // Realiza operações do Wi-Fi.
cyw43_arch_deinit(); // Desativa o Wi-Fi antes de finalizar o programa.
return 0; }
```


Configuração do CMakeLists

```
# Modify the below lines to enable/disable output over UART/USB
pico_enable_stdio_uart(led_control_webserver 0)
pico_enable_stdio_usb(led_control_webserver 1)
# Add the standard library to the build
target_link_libraries(led_control_webserver
    pico_stdlib
    hardware_gpio
    hardware_adc
    pico_cyw43_arch_lwip_threadsafe_background )
# Add the standard include files to the build
target_include_directories(led_control_webserver PRIVATE
    ${CMAKE_CURRENT_LIST_DIR}
    ${PICO_SDK_PATH}/lib/lwip/src/include
    ${PICO_SDK_PATH}/lib/lwip/src/include/arch
    ${PICO_SDK_PATH}/lib/lwip/src/include/lwip )
target_sources(led_control_webserver PRIVATE
    ${PICO_SDK_PATH}/lib/lwip/src/apps/http/httpd.c
    ${PICO_SDK_PATH}/lib/lwip/src/apps/http/fs.c )
# Add any user requested libraries
pico_add_extra_outputs(led_control_webserver)
```

Test do código

☐ Interromper o Monitoramento



Falha ao conectar ao Wi-Fi

---- Porta serial fechada COM29 devido à desconexão da máquina ----

---- Porta serial reaberta COM29 ----

Conectando ao Wi-Fi...

Falha ao conectar ao Wi-Fi

---- Porta serial fechada COM29 devido à desconexão da máquina ----

---- Porta serial reaberta COM29 ----

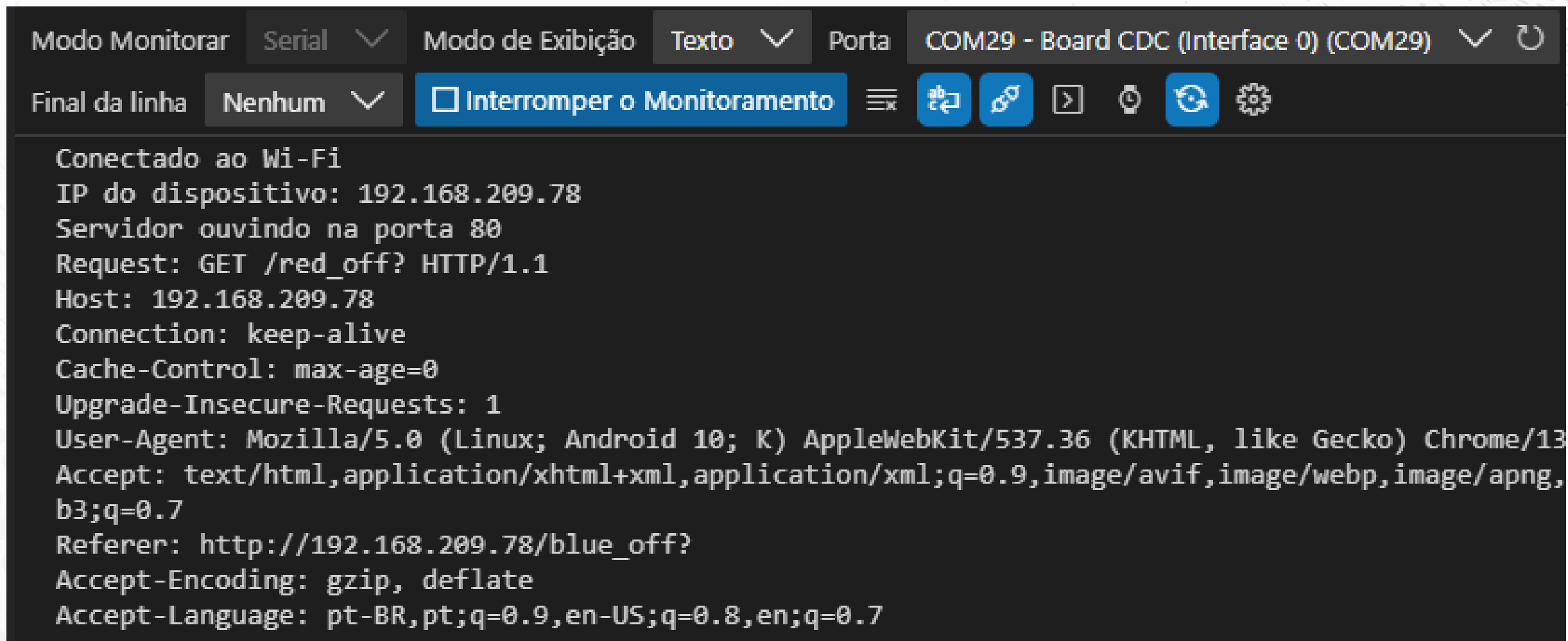
Conectando ao Wi-Fi...

Conectado ao Wi-Fi

IP do dispositivo: 192.168.209.78

Servidor ouvindo na porta 80

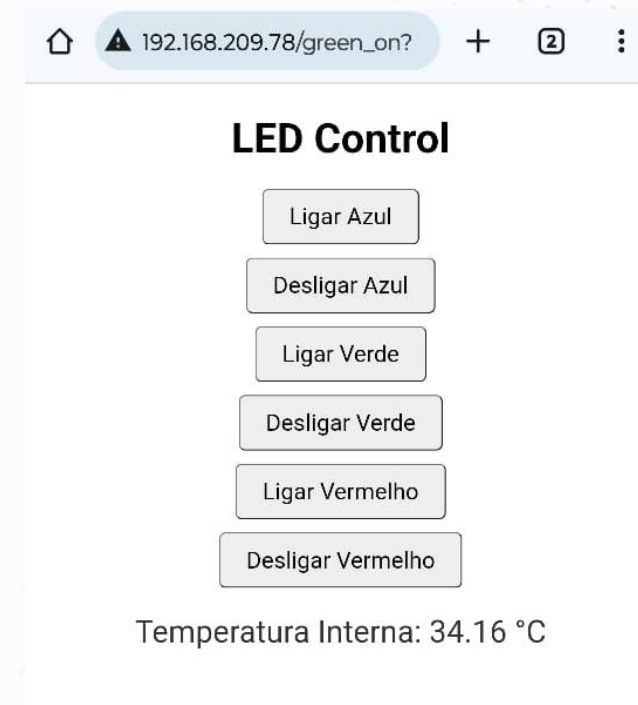
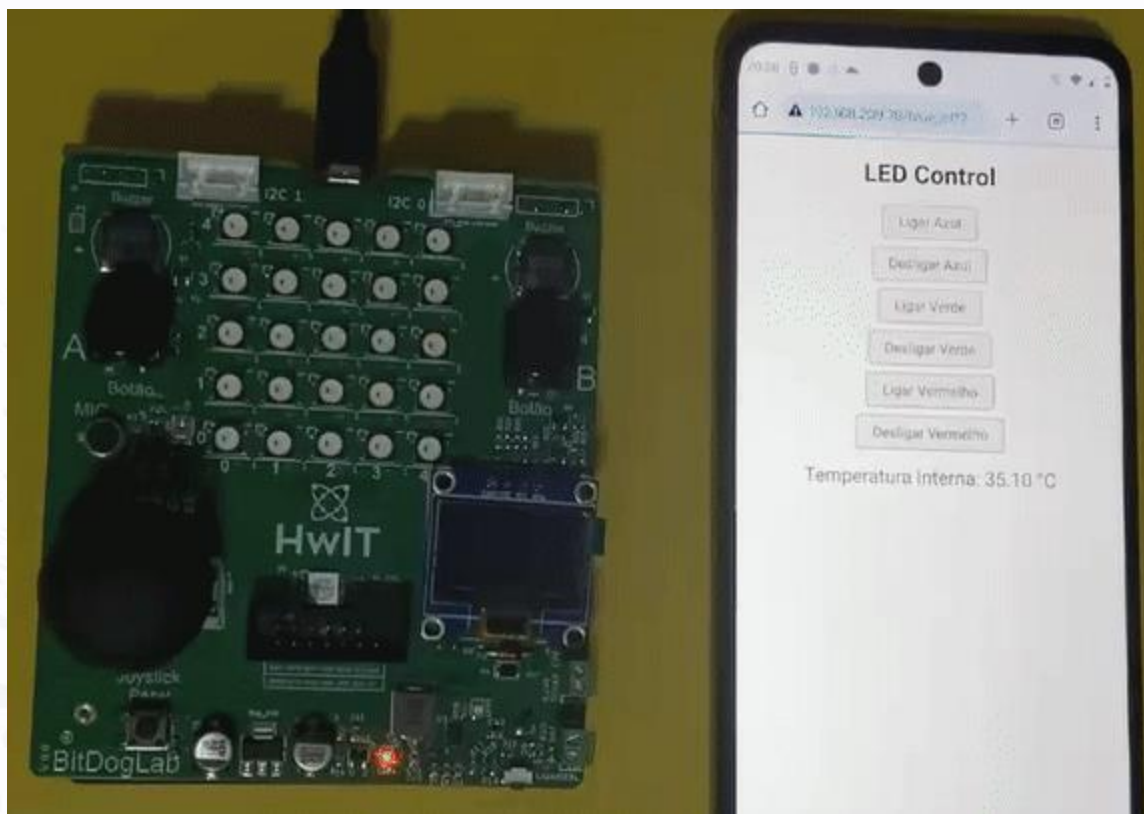
Test do código



The screenshot shows a serial terminal application interface. At the top, there is a header bar with several controls: 'Modo Monitorar' (Monitoring Mode), a dropdown menu set to 'Serial', 'Modo de Exibição' (Display Mode), a dropdown menu set to 'Texto' (Text), 'Porta' (Port), and a dropdown menu set to 'COM29 - Board CDC (Interface 0) (COM29)'. Below the header, there is a row of controls including 'Final da linha' (End of line), a dropdown menu set to 'Nenhum' (None), a blue button with a square icon and the text 'Interromper o Monitoramento' (Stop Monitoring), a hamburger menu icon, and several other icons for functionality like copy, paste, and settings. The main area of the terminal displays the following text:

```
Conectado ao Wi-Fi
IP do dispositivo: 192.168.209.78
Servidor ouvindo na porta 80
Request: GET /red_off? HTTP/1.1
Host: 192.168.209.78
Connection: keep-alive
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/13
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,
b3;q=0.7
Referer: http://192.168.209.78/blue_off?
Accept-Encoding: gzip, deflate
Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
```

Test do código



Conclusão

- Exploramos conceitos e aplicações de comunicação sem fio para IoT.
- Tecnologias destacadas: Wi-Fi.
- Aplicações práticas com a Raspberry Pi Pico W.
- Implementação de um servidor IoT para controle de LEDs e monitoramento de temperatura.

Link1: https://drive.google.com/file/d/1yBTNgRKwCv-3qnHReNlhDygt5mZzxe7D/view?usp=drive_link

Link2: [led_control_webserver_OK.zip](#)

Aplicações com Microcontroladores

