

Comunicação em IoT

RESIDÊNCIA – Aula Síncrona (06/05/2025)

Prof. Dr Ricardo Menezes Prates

Executores:



INSTITUTO FEDERAL
São Paulo



INSTITUTO FEDERAL
Rio Grande do Norte



INSTITUTO FEDERAL
Maranhão



INSTITUTO FEDERAL
Ceará



INSTITUTO
HARDWARE BR

Coordenação:



Iniciativa:

MINISTÉRIO DA
CIÊNCIA, TECNOLOGIA
E INOVAÇÃO

GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO

- *Introdução*

- Planejamento da aula

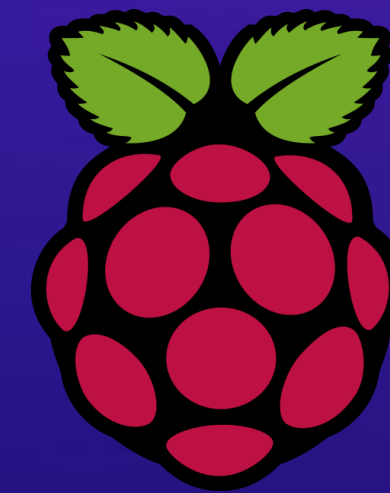
- *Comunicação em IoT*

- *Wi-Fi*
 - *Bluetooth*
 - *Lora*
 - *Zigbee*

- *Práticas com o módulo CYW43439 – interfaces sem fio*

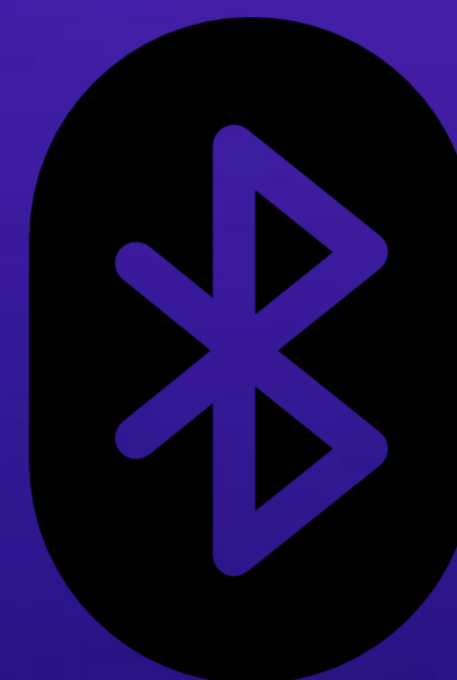
- *Inicializar a arquitetura do módulo wireless*
 - *Varredura WI-FI: SSID, RSSI, endereço MAC*
 - *Conexão com rede local WI-FI (WLAN)*
 - *Pico Pi W Webserver – vinculada a atividade da semana!*
 - *Bluetooth Low Energy (BLE)*

- *Conclusões*



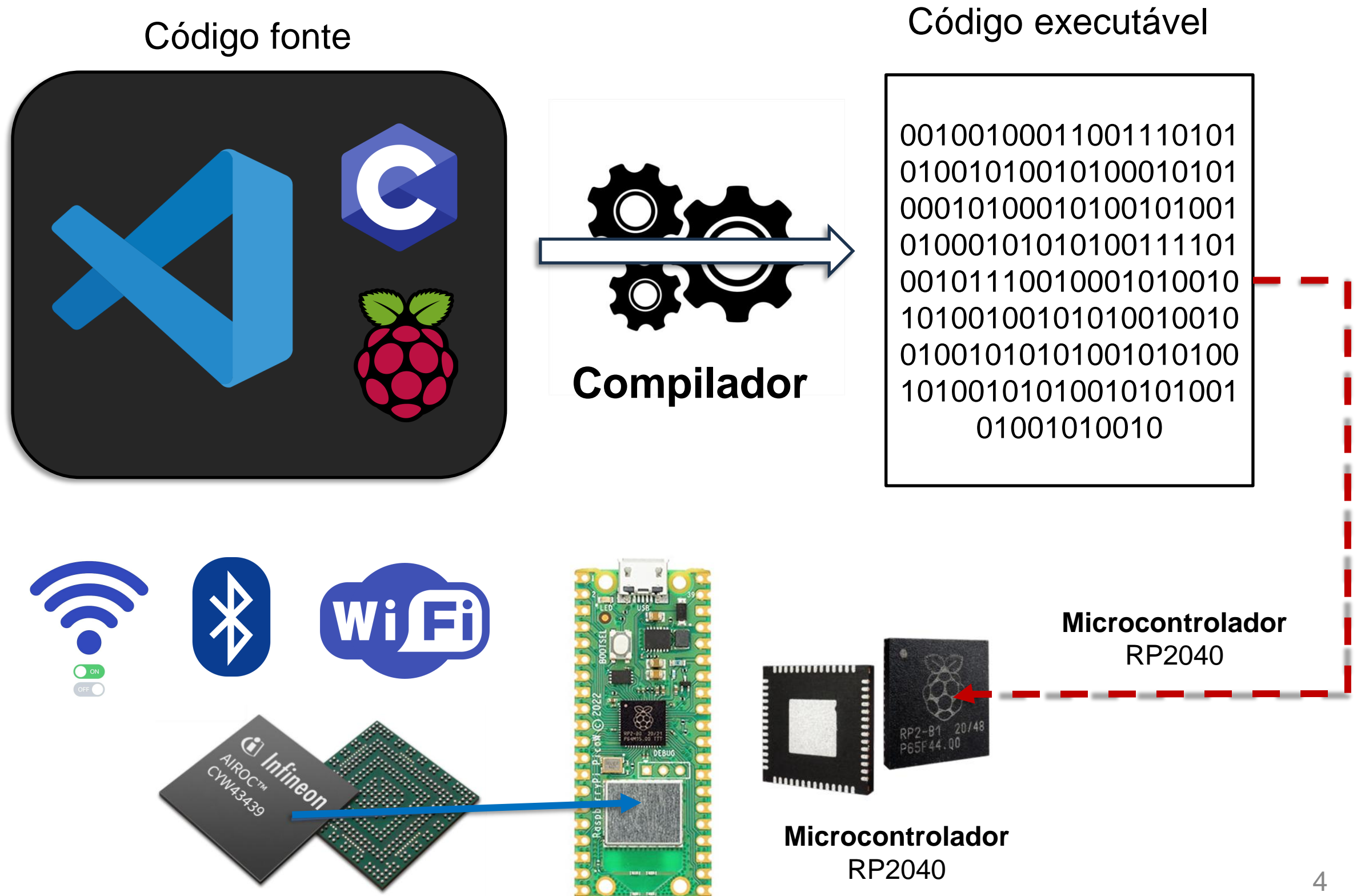


Introdução



Objetivos

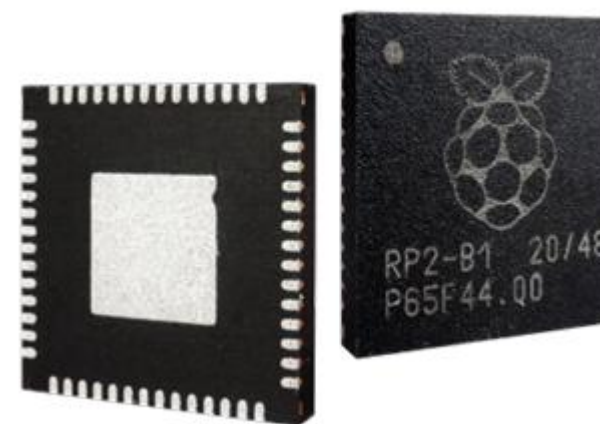
- Compreender as características das **interfaces de comunicação sem fio** e desenvolver rotinas de configuração do microcontrolador RP2040 e do modulo CYW43439.
- Implementar e configurar a comunicação WI-FI e Bluetooth no Raspberry Pi Pico W.



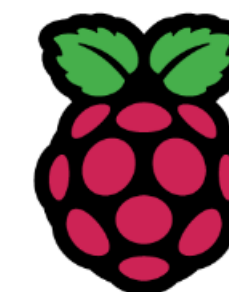
Planejamento da Aula

- Pré-requisitos para esta aula:

- ✓ Linguagem C e Pico SDK- RP2040;
- ✓ Portas de Entrada e Saída (GPIO);
- ✓ Rotinas de interrupção;
- ✓ *Clock* e Temporizadores;
- ✓ Conversor Analógico/Digital (ADC).



raspberrypi/**pico-
sdk**



Revisão da aula Assíncrona

- Internet das Coisas (IoT, do inglês de *Internet of Things*):
 - ✓ É um conceito que descreve a conexão de objetos físicos à internet para que possam coletar, trocar e agir com base em dados.
 - ✓ Esses dispositivos podem variar de objetos cotidianos, como termostatos e lâmpadas inteligentes, até sistemas complexos, como veículos autônomos e cidades inteligentes.
 - ✓ O objetivo da IoT é criar um ambiente mais eficiente, automatizado e integrado.





Comunicação em IoT



IoT

- Redes sem fio (*wireless*):
 - ✓ Termo genérico utilizado para redes de dados que usam ondas eletromagnéticas para comunicação entre os dispositivos.



IoT

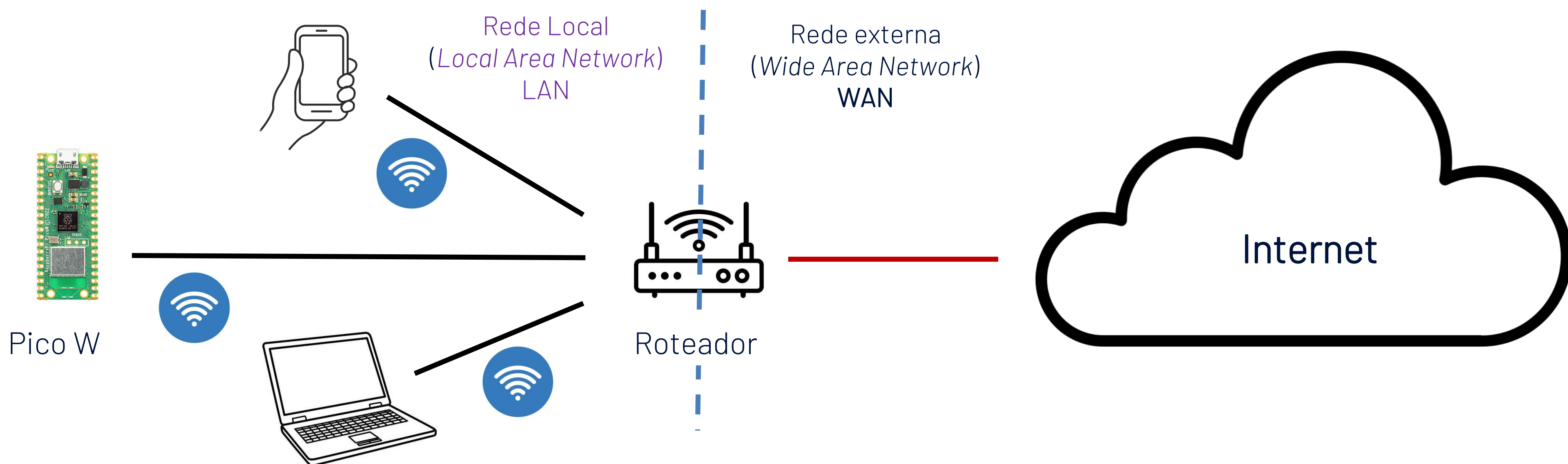
- Tipos de redes *wireless*:
 - ✓ WI-FI;
 - ✓ Bluetooth;
 - ✓ Rádio (AM / FM);
 - ✓ LoRa;
 - ✓ ZigBee;
 - ✓ Bandas de celular (2G, 3G, 4G e 5G);
 - ✓ Z-Wave, WiMAX, NFC etc.



IoT

- Rede WLAN ("Wireless LAN):

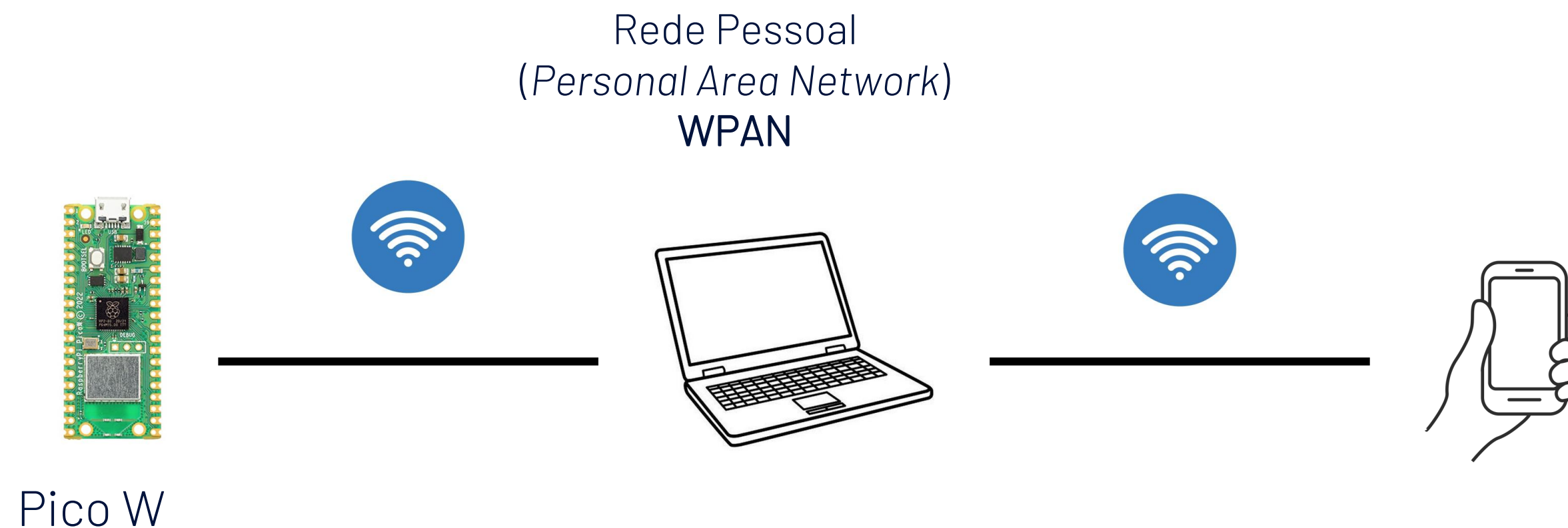
- ✓ Se refere a uma rede local (LAN), a exemplo de uma residência ou escritório, onde seja necessária uma rede com pequena área de cobertura sem o emprego de cabeamento.



IoT

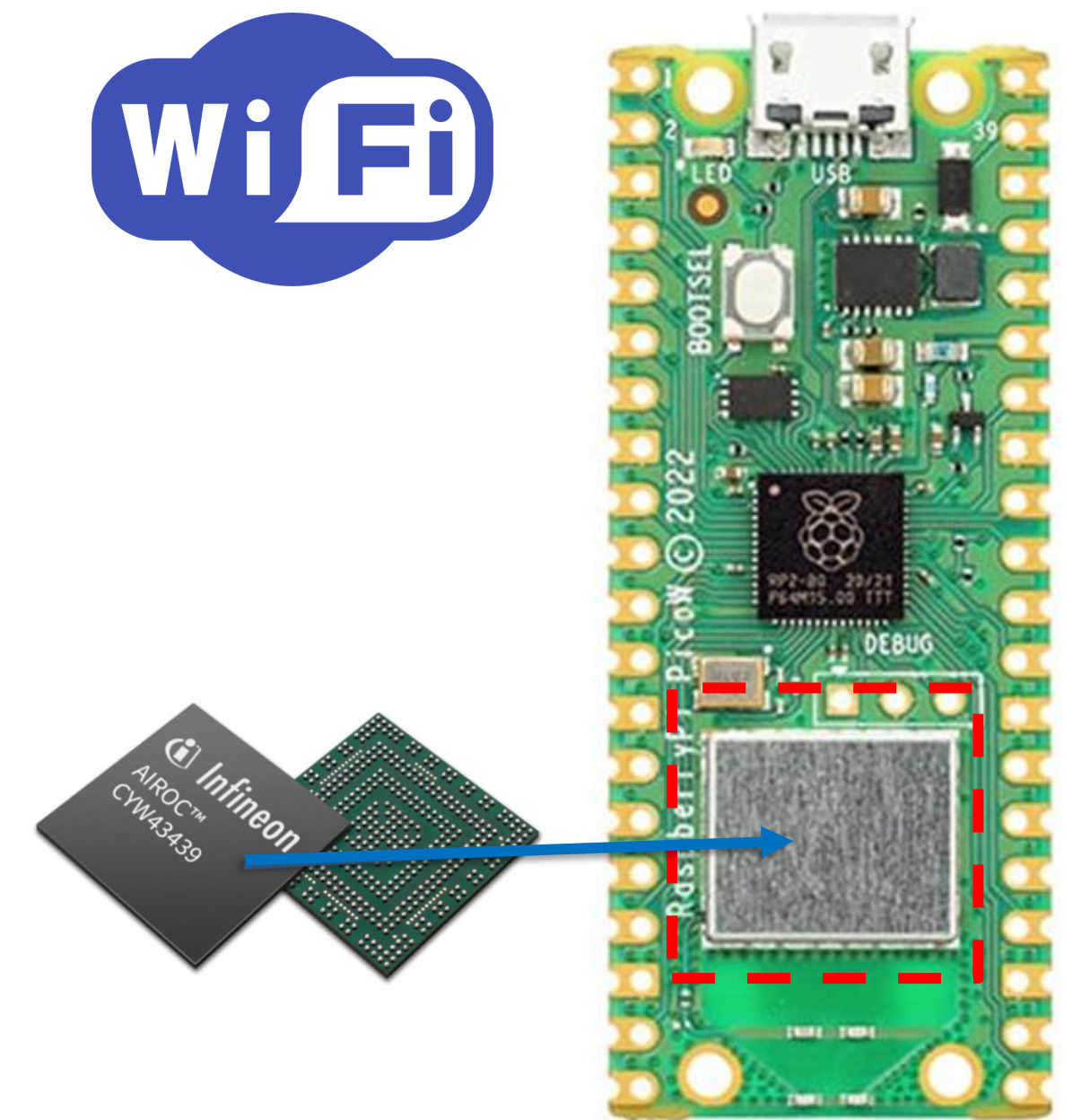
- Rede WPAN ("Wireless PAN):

- ✓ Se refere a uma rede pessoal (PAN), desenvolvida para atender demandas de comunicação sem fio entre dispositivos de curta distância e com baixo consumo de energia.



WI-FI

- Especificação IEEE 802. 11.
- ✓ Atualmente, a rede WLAN padrão é a WI-FI (“*Wireless Fidelity*”).
- ✓ Trata-se de um tipo de rede wireless para construção de LANs para comunicação de computadores e dispositivos portáteis.
- ✓ O Raspberry Pi Pico W conta com a Interface *wireless* de banca única de 2,4 GHz integrada - 802.11n. Geração Wi-Fi 4 - suporta velocidades de até 600 Mbps.



WI-FI

- Como funciona?

1. **Transmissão de sinal:** Um dispositivo Wi-Fi, a exemplo de um roteador, transmite um sinal de rádio que pode ser percebido por dispositivos próximos.

2. **Conexão:** Um dispositivo Wi-Fi, como um smartphone, laptop ou Raspberry Pi Pico W, detecta o sinal e se conecta à rede Wi-Fi.

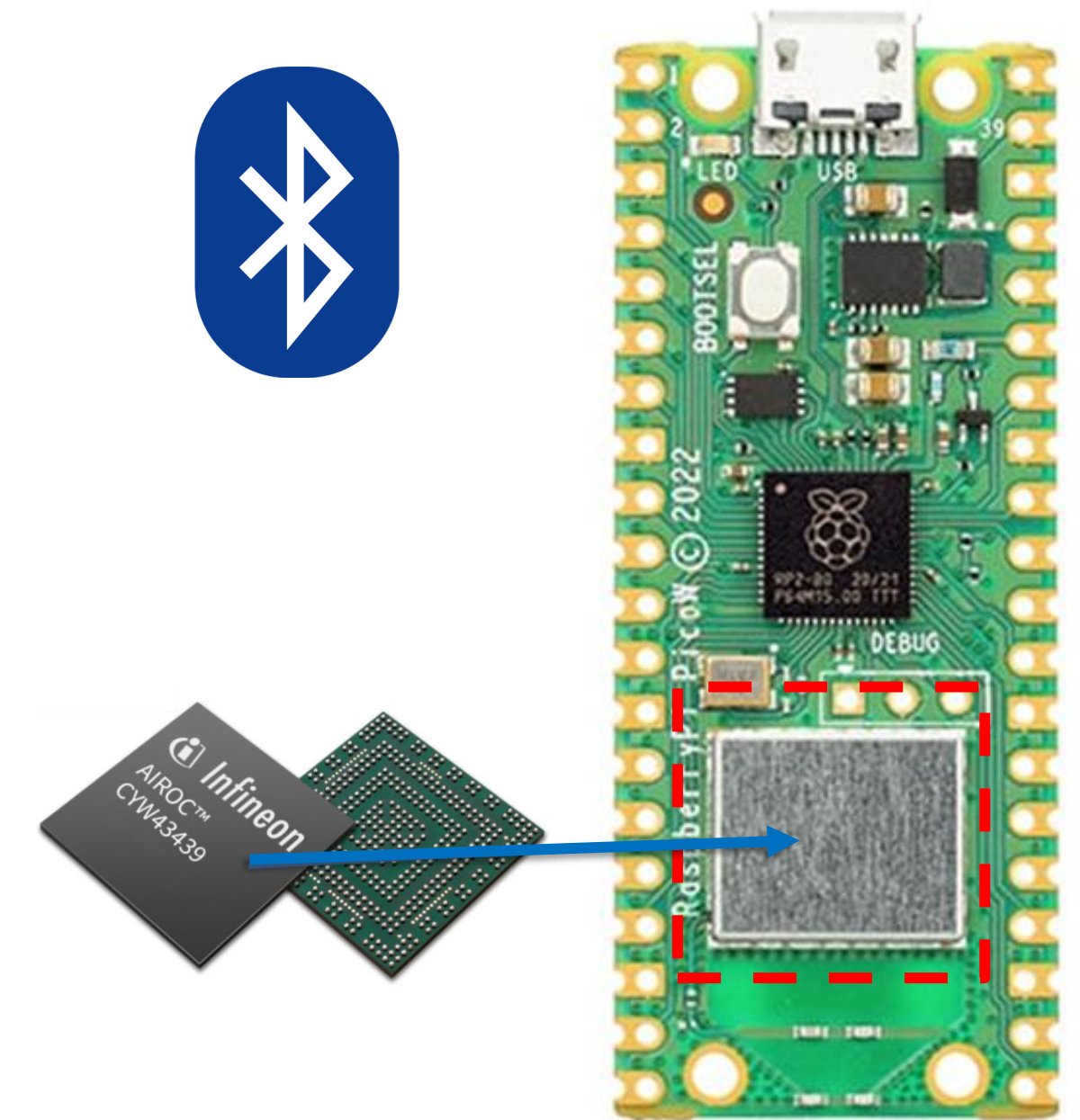
3. **Autenticação:** O dispositivo precisa se autenticar na rede Wi-Fi, geralmente fornecendo uma senha.

4. **Comunicação:** Após a autenticação, o dispositivo pode se comunicar com a rede Wi-Fi e acessar a internet ou recursos da rede local (LAN).



Bluetooth

- Especificação IEEE 802.15.1
- ✓ O protocolo Bluetooth é uma tecnologia de comunicação sem fio que permite que dispositivos se conectem e troquem dados entre si.
- ✓ Desenvolvido para atender demandas de conexão WPAN.
- ✓ **Bluetooth Clássico:** Usado para aplicações que exigem uma conexão contínua, como fones de ouvido e alto-falantes.
- ✓ **Bluetooth Low Energy (BLE):** Usado para aplicações que exigem baixa potência e transferência de dados intermitente, como dispositivos de saúde e fitness.



Bluetooth

- Como funciona?

1. **Descoberta:** Um dispositivo Bluetooth procura por outros dispositivos próximos que estejam habilitados para Bluetooth.

2. **Emparelhamento:** Os dispositivos se conectam e se autenticam mutuamente, geralmente através de uma senha ou código de acesso.

3. **Conexão:** Após o emparelhamento, os dispositivos estabelecem uma conexão e podem trocar dados.

4. **Comunicação:** Os dispositivos podem trocar dados, como áudio, arquivos e informações de dispositivo.



Power Class	Maximum Output Power	Operating Range
Class 1	100 mW (20 dBm)	100 meters
Class 2	2.5 mW (4 dBm)	10 meters
Class 3	1 mW (0 dBm)	1 meter

LoRa

- Características:

- ✓ LoRa (acrônimo para Long Range) corresponde à tecnologia de comunicação para longas distâncias, com baixo consumo de energia elétrica;
- ✓ Projetada para conectar dispositivos IoT (Internet das Coisas) em uma rede de área ampla – No Brasil, a frequência padrão é de 915 MHz (Rede não licenciada).

- ✓ Características:

1 – **Longo alcance:** LoRa pode alcançar distâncias de até 15 km em áreas rurais e 5 km em áreas urbanas.

2 – **Baixa taxa de transmissão:** Taxa de transmissão de dados relativamente baixa, geralmente entre 0,3 e 50 kbps.

VANTAGENS DA REDE LORA



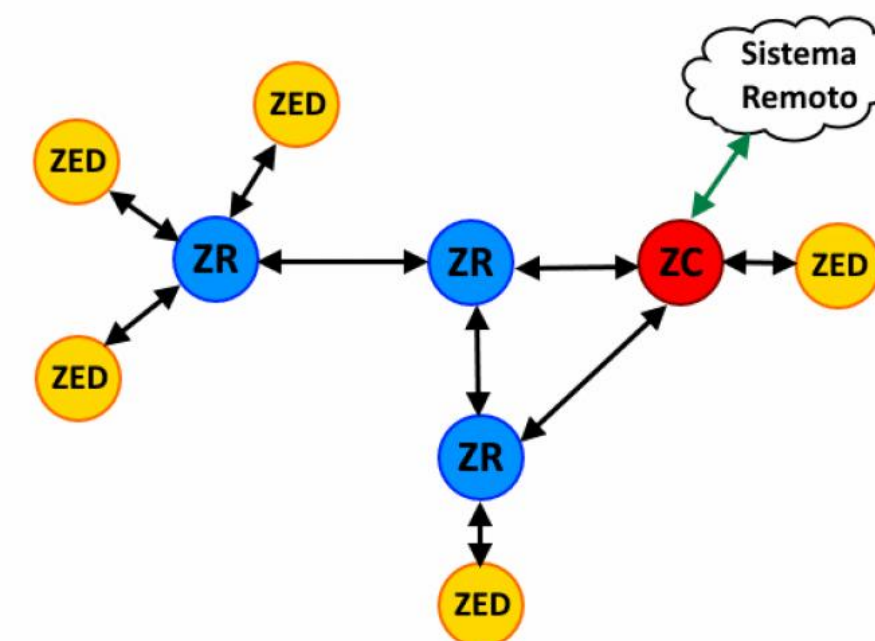
aiko



ZigBee

• Características:

- ✓ Protocolo aberto, voltado para a criação de redes sem-fio de curto alcance (Personal Area Network - PAN) seguras, confiáveis e com baixo consumo de energia.
- ✓ É mantido pela *Connectivity Standard Alliance* (anteriormente chamada de Zigbee Alliance), uma entidade que tem como membros empresas como a Amazon, Apple, Google, Espressif, Infineon, NXP, Nordic e a ST.
- ✓ Este protocolo forma uma rede mesh, onde cada dispositivo pode se comunicar com outros dispositivos e atuar como um roteador para estender a cobertura da rede.
- ✓ Visa comunicações de curto alcance (10 metros).

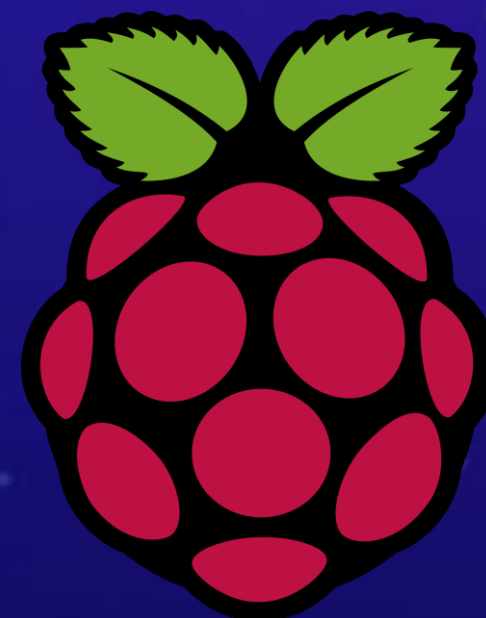


<https://www.makerhero.com/blog/o-que-e-zigbee/>

Comparativo

Resumo comparativo entre as tecnologias			
Tecnologia	Alcance	Velocidade Máxima	Uso Principal
Bluetooth	1-10 metros	Até 50 Mbps	Dispositivos pessoais
Wi-Fi	Até 100 metros	Até 9,6 Gbps (Wi-Fi 6)	Redes domésticas e IoT
Zigbee	10-100 metros	Até 250 kbps	Automação e sensores IoT
LoRa	Até 15 km	Baixa (0,3-50 kbps)	IoT de longo alcance
NFC	Até 4 cm	Até 424 kbps	Pagamentos e acessos
LTE/5G	Global	1-10+ Gbps	Comunicação móvel

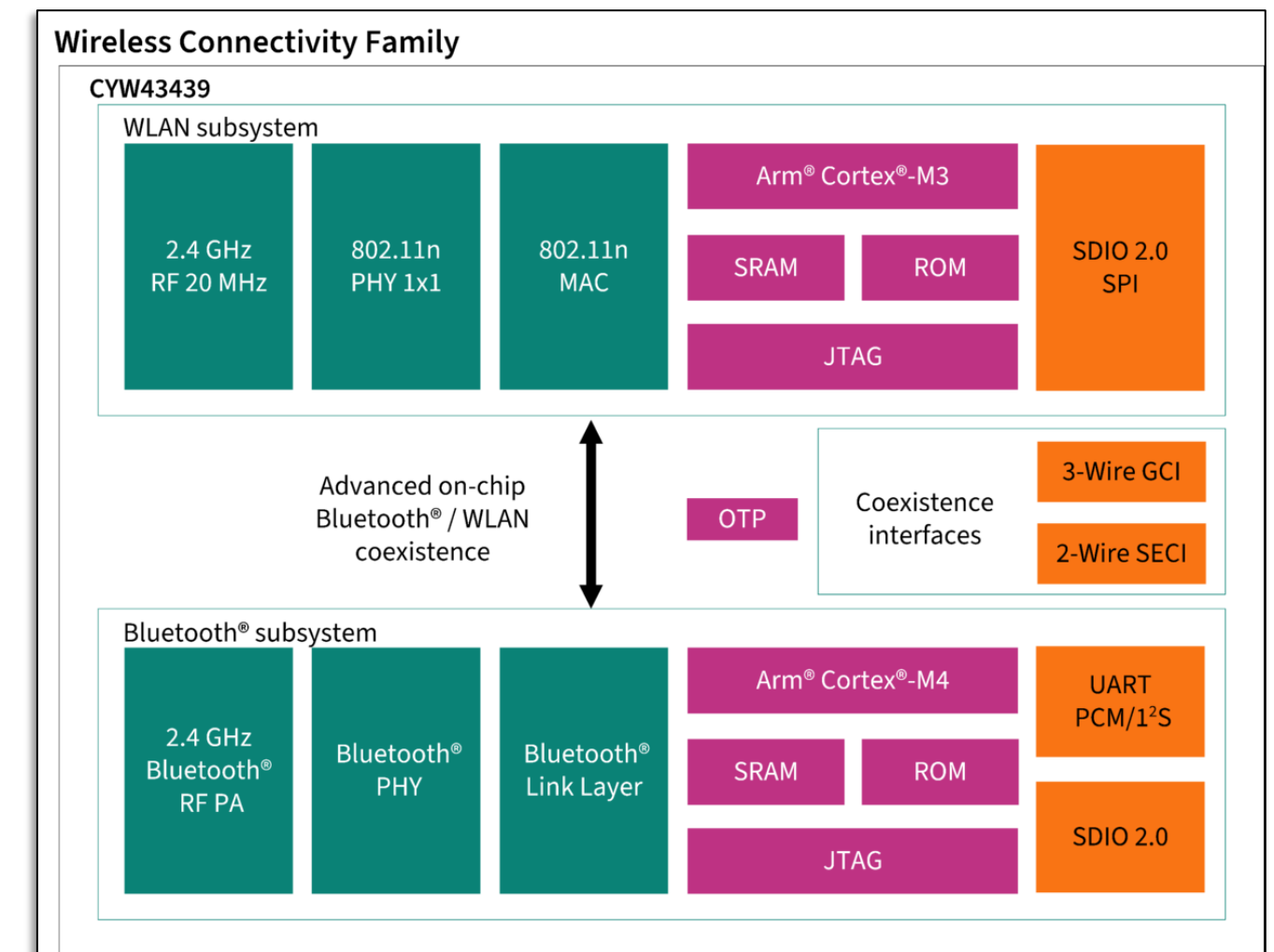
Práticas com o módulo CYW43439



Práticas com o módulo CYW43439


- Características:

- ✓ O Raspberry Pi Pico W adiciona interfaces sem fio de 2,4 GHz integradas ao hardware usando o **Infineon CYW43439**.
- ✓ WI-FI (IEEE 802.11 b/g/n), banda única - 2,4 GHz;
- ✓ Bluetooth 5.2 (IEEE 802.15.1):
 - 1 - Bluetooth Classic;
 - 2 - Bluetooth Low Energy (LE) Central e Periférica.
- ✓ A interface sem fio é conectada via SPI ao microcontrolador.



Práticas com o módulo CYW43439

- Referências importantes para consulta:

 **Raspberry Pi**

For industry

For home

HardwareSoftwareDocumentationNewsForumsFoundation

Documentation

Q Search

CTRLK

▶ Computers

▶ Accessories

▶ Microcontrollers

▶ Services

▼ Pico C SDK

- ▶ Introduction
- ▶ Hardware APIs
- ▶ High Level APIs
- ▶ Third-party Libraries
- ▼ Networking Libraries
 - pico_btstack

Networking Libraries

Functions for implementing networking


pico_btstack	Integration/wrapper libraries for BTstack the documentation for which is here .
pico_lwip	Integration/wrapper libraries for lwIP the documentation for which is here .
pico_lwip_arch	lwIP compiler adapters. This is not included by default in pico_lwip in case you wish to implement your own.
pico_lwip_freertos	Glue library for integration lwIP in NO_SYS=0 mode with the SDK.
pico_lwip_nosys	Glue library for integration lwIP in NO_SYS=1 mode with the SDK.
pico_cyw43_driver	A wrapper around the lower level cyw43_driver, that integrates it with pico_async_context for handling background work.
pico_btstack_cyw43	Low-level Bluetooth HCI support.
pico_cyw43_arch	Architecture for integrating the CYW43 driver (for the wireless on Pico W) and lwIP (for TCP/IP stack) into the SDK. It is also necessary for accessing the on-board LED on Pico W.
cyw43_driver	Driver used for Pico W wireless.
cyw43_ll	Low Level CYW43 driver interface.

<https://www.raspberrypi.com/documentation/pico-sdk/networking.html>

RP2040 A microcontroller by Raspberry Pi

Connecting to the Internet with Raspberry Pi Pico W

Getting Raspberry Pi Pico W online with C/C++ or MicroPython



Raspberry Pi Ltd

Práticas com o módulo CYW43439

- Referências importantes para consulta:

Exemplo:

cyw43_arch_init

int **cyw43_arch_init** (void)

Initialize the CYW43 architecture.

This method initializes the `cyw43_driver` code and initializes the lwIP stack (if it was enabled at build time). This method must be called prior to using any other `pico_cyw43_arch`, `cyw43_driver` or lwIP functions.

NOTE

this method initializes wireless with a country code of `PICO_CYW43_ARCH_DEFAULT_COUNTRY_CODE` which defaults to `CYW43_COUNTRY_WORLDWIDE`. Worldwide settings may not give the best performance; consider setting `PICO_CYW43_ARCH_DEFAULT_COUNTRY_CODE` to a different value or calling **cyw43_arch_init_with_country**

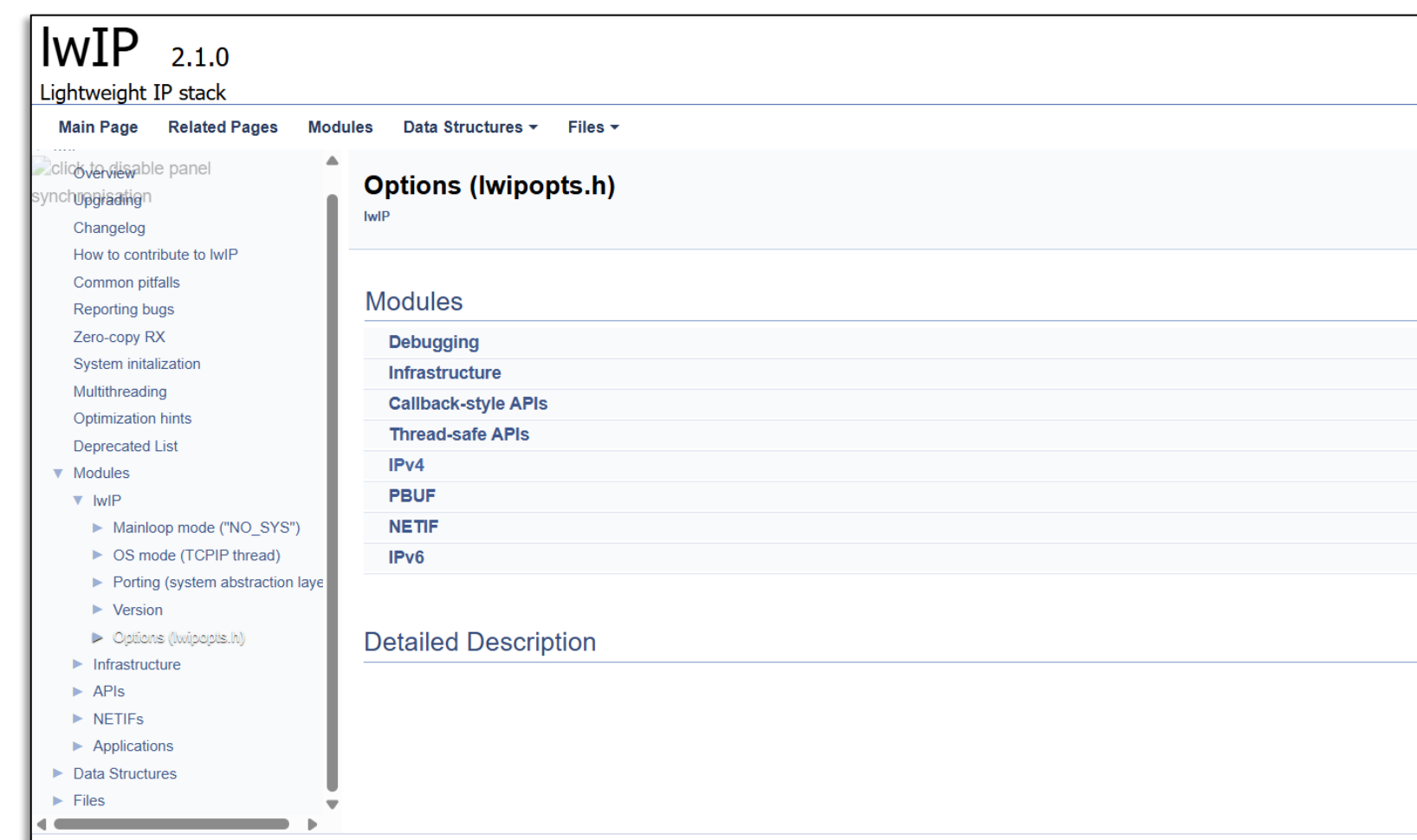
By default this method initializes the `cyw43_arch` code's own `async_context` by calling **cyw43_arch_init_default_async_context**, however the user can specify use of their own `async_context` by calling `cyw43_arch_set_async_context()` before calling this method

Returns

0 if the initialization is successful, an error code otherwise

Práticas com o módulo CYW43439

- Referências importantes para consulta:
 - ✓ Biblioteca LwIP (Lightweight IP), que é uma implementação leve do protocolo IP para sistemas embarcados e dispositivos com recursos limitados;
 - ✓ A biblioteca suporta os protocolos IPV4 e IPV6, além de TCP, UDP e ICMP;
 - ✓ O SDK inclui a implementação de LwIP.



https://www.nongnu.org/lwip/2_1_x/group_lwip_opts.html

Práticas com o módulo CYW43439

- Referências importantes para consulta:

Exemplos:

◆ tcp_new()

```
struct tcp_pcb* tcp_new ( void )
```

Creates a new TCP protocol control block but doesn't place it on any of the TCP PCB lists. The pcb is not put on any list until binding using **tcp_bind()**. If memory is not available for creating the new pcb, NULL is returned.

◆ tcp_bind()

```
err_t tcp_bind ( struct tcp_pcb * pcb,  
                const ip_addr_t * ipaddr,  
                u16_t port  
                )
```

Binds the connection to a local port number and IP address. If the IP address is not given (i.e., ipaddr == IP_ANY_TYPE), the connection is bound to all local IP addresses. If another connection is bound to the same port, the function will return ERR_USE, otherwise ERR_OK is returned.

Parameters

pcb the **tcp_pcb** to bind (no check is done whether this pcb is already bound!)

ipaddr the local ip address to bind to (use IPx_ADDR_ANY to bind to any local address)

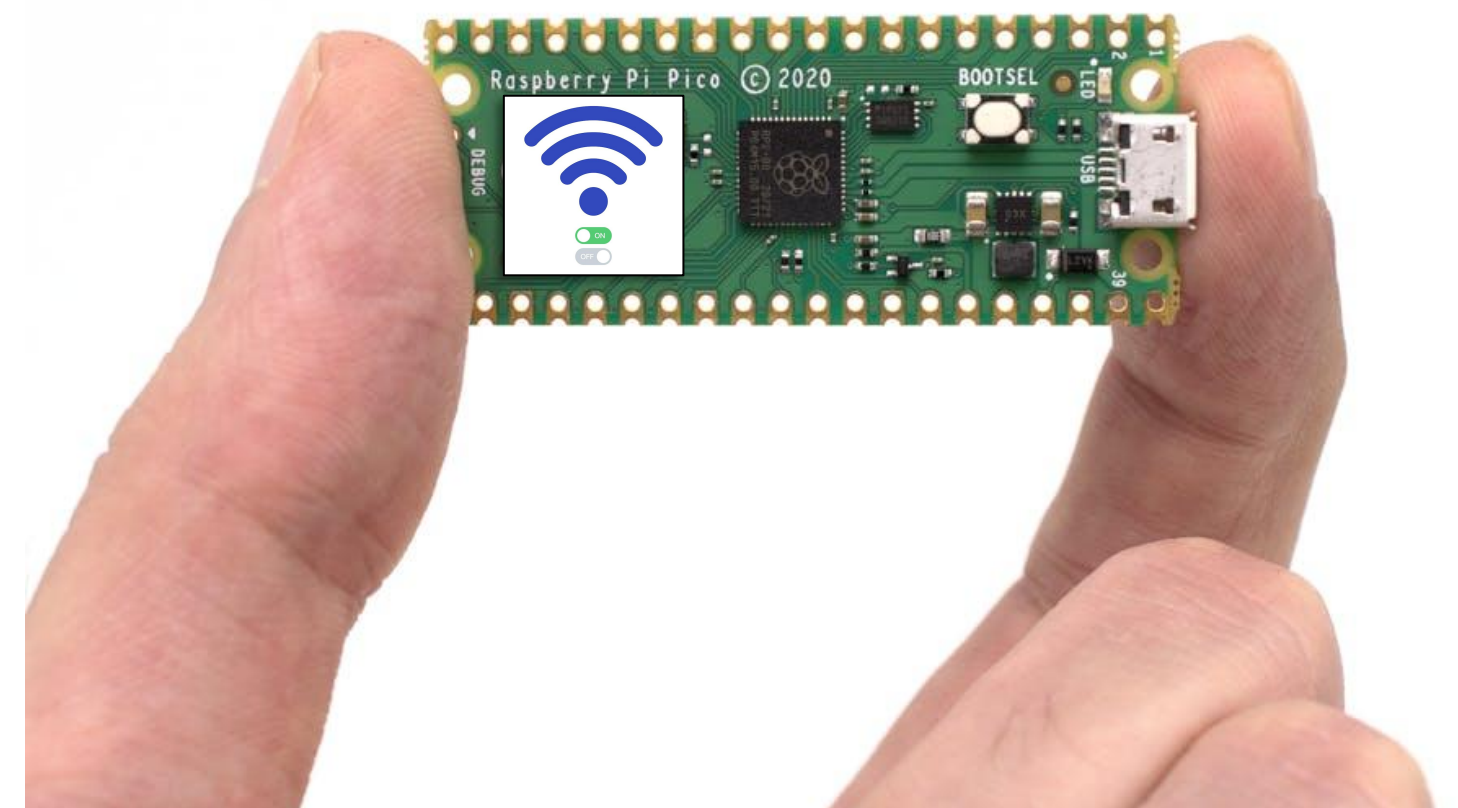
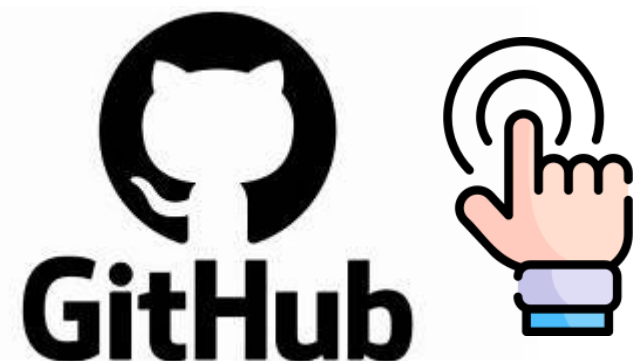
port the local port to bind to

Returns

ERR_USE if the port is already in use ERR_VAL if bind failed because the PCB is not in a valid state ERR_OK if bound

Vamos para o microcontrolador!

Inicializar a arquitetura do módulo wireless



Inicializa o Módulo Wireless

Código C:

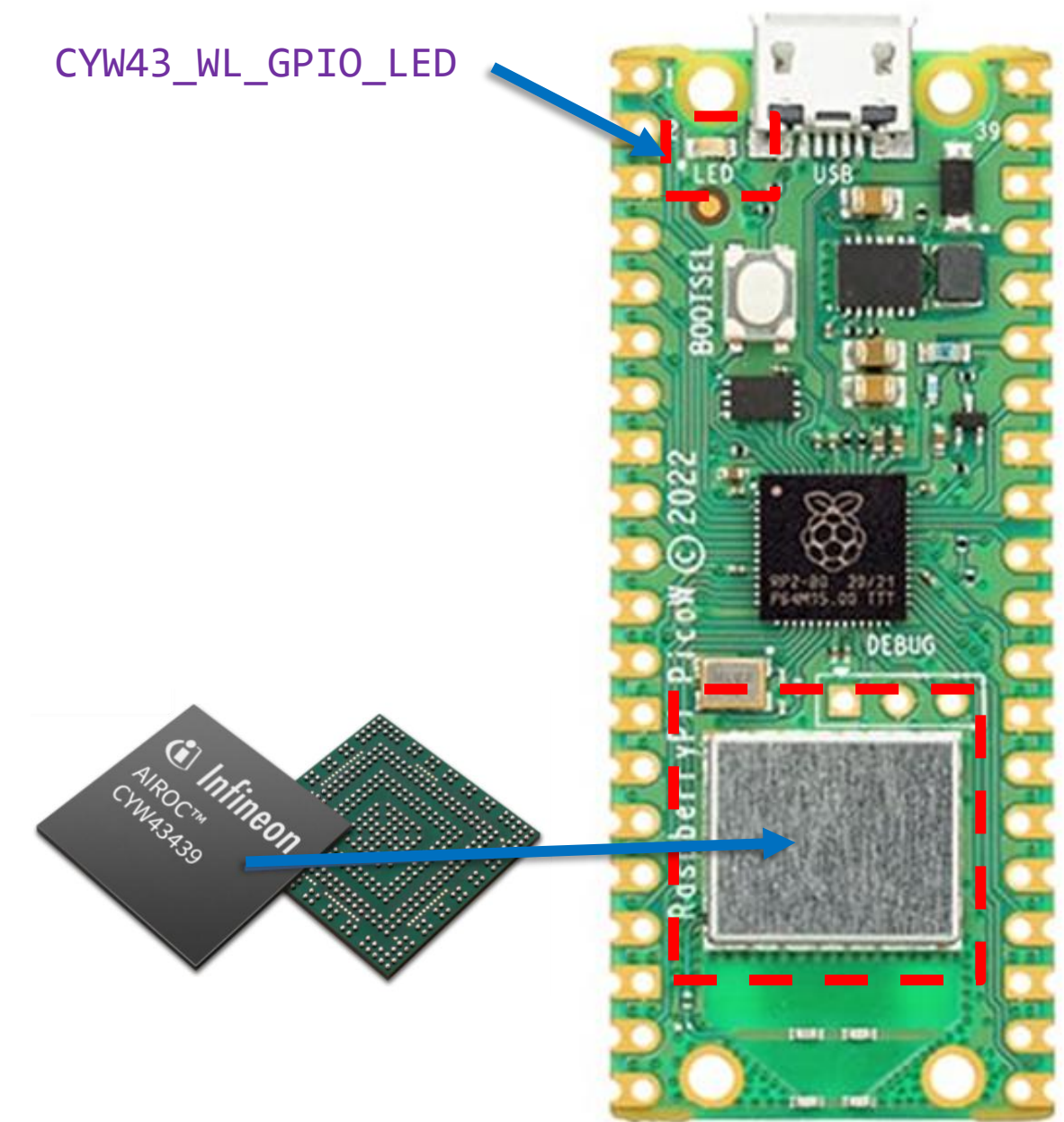
```
#include "pico/stdlib.h"           // Biblioteca padrão para entrada e saída
#include "pico/cyw43_arch.h"       // Biblioteca de arquitetura CYW43 para Raspberry Pi Pico

int main() {

    // Inicializa o sistema de entrada e saída padrão (printf, etc.)
    stdio_init_all();

    // Inicializa a arquitetura CYW43 (Wi-Fi e Bluetooth)
    if (cyw43_arch_init()) { // Verifica se houve erro na inicialização
        // 0 se a inicialização for bem sucedida, um código de erro caso contrário
        printf("Falha ao iniciar o wi-fi"); // Mensagem de erro
        return -1; // Sai do programa com código de erro
    }

    while (true) { // Loop principal
        // Definir o estado lógico do pino GPIO no chip CYW43439
        cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 1); // Pino em nível alto
        sleep_ms(1000); // Delay de 1000 milissegundos
        cyw43_arch_gpio_put(CYW43_WL_GPIO_LED_PIN, 0); // Pino em nível baixo
        sleep_ms(1000); // Delay de 1000 milissegundos
    }
}
```



Inicializa o Módulo Wireless



Trecho de código do CMakeLists.txt:

```
# Initialise the Raspberry Pi Pico SDK
pico_sdk_init()

# Add executable. Default name is the project name, version 0.1

add_executable(iot_01 iot_01.c )

pico_set_program_name(iot_01 "iot_01")
pico_set_program_version(iot_01 "0.1")

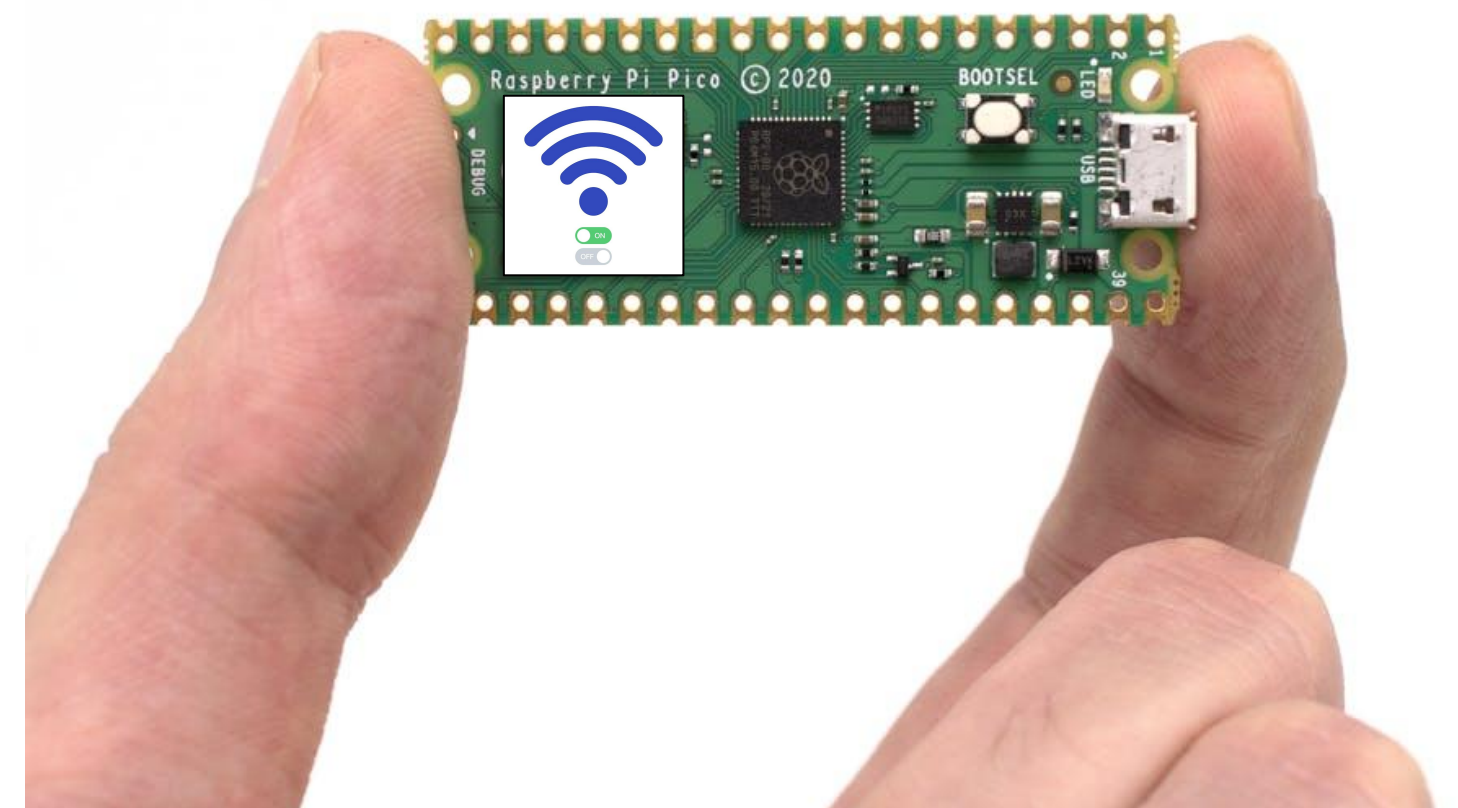
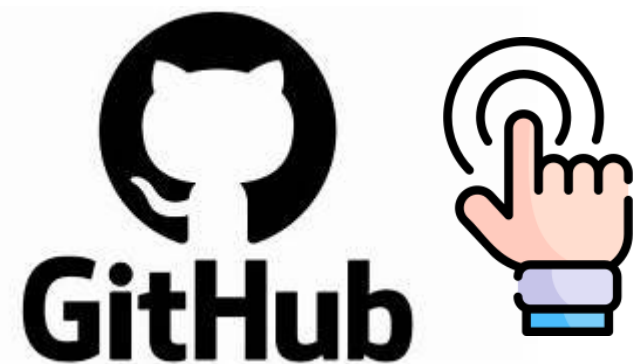
# Modify the below lines to enable/disable output over UART/USB
pico_enable_stdio_uart(iot_01 0)
pico_enable_stdio_usb(iot_01 1)

# Add the standard library to the build
target_link_libraries(iot_01
    pico_stdlib
    pico_cyw43_arch_none)
```



Vamos para o microcontrolador!

Varredura WI-FI



Varredura WI-FI

Pretende-se obter:

1. **SSID (Service Set Identifier)**: nome que identifica uma rede Wi-Fi;
2. **RSSI (Received Signal Strength Indicator)**: medida da intensidade do sinal de uma rede Wi-Fi recebida por um dispositivo.

Níveis de RSSI e Qualidade do Sinal:

1. **Excelente**: -30 dBm a -50 dBm (sinal muito forte)
2. **Bom**: -50 dBm a -67 dBm (sinal forte)
3. **Regular**: -67 dBm a -80 dBm (sinal médio)
4. **Fraco**: -80 dBm a -100 dBm (sinal fraco)

Obs.: É um indicador importante da qualidade da conexão Wi-Fi.



Varredura WI-FI

3. **CHAN (Canal) Wi-Fi**: refere-se ao canal de frequência específico utilizado por uma rede Wi-Fi para transmitir e receber dados.
- ✓ Os canais Wi-Fi são uma parte fundamental da comunicação sem fio e são utilizados para minimizar a interferência entre diferentes redes Wi-Fi próximas.
 - ✓ Na faixa de 2,4 GHz, existem 11 a 14 canais disponíveis, dependendo da região. No entanto, devido à sobreposição de canais, apenas 3 canais (**1, 6 e 11**) são considerados não sobrepostos e são recomendados para uso.

Varredura WI-FI

- 4. **MAC (Media Access Control)**: é um endereço único atribuído a dispositivos de rede, incluindo dispositivos Wi-Fi. Ele é usado para identificar dispositivos em uma rede local.
- ✓ Diferença entre Endereço MAC e Endereço IP:
 - i. **Endereço MAC**: É um endereço físico único atribuído a um dispositivo de rede.
 - ii. **Endereço IP**: É um endereço lógico atribuído a um dispositivo em uma rede para comunicação em diferentes redes.

Obs.: No Windows, para obter essas informações, use o comando no prompt: `ipconfig /all`

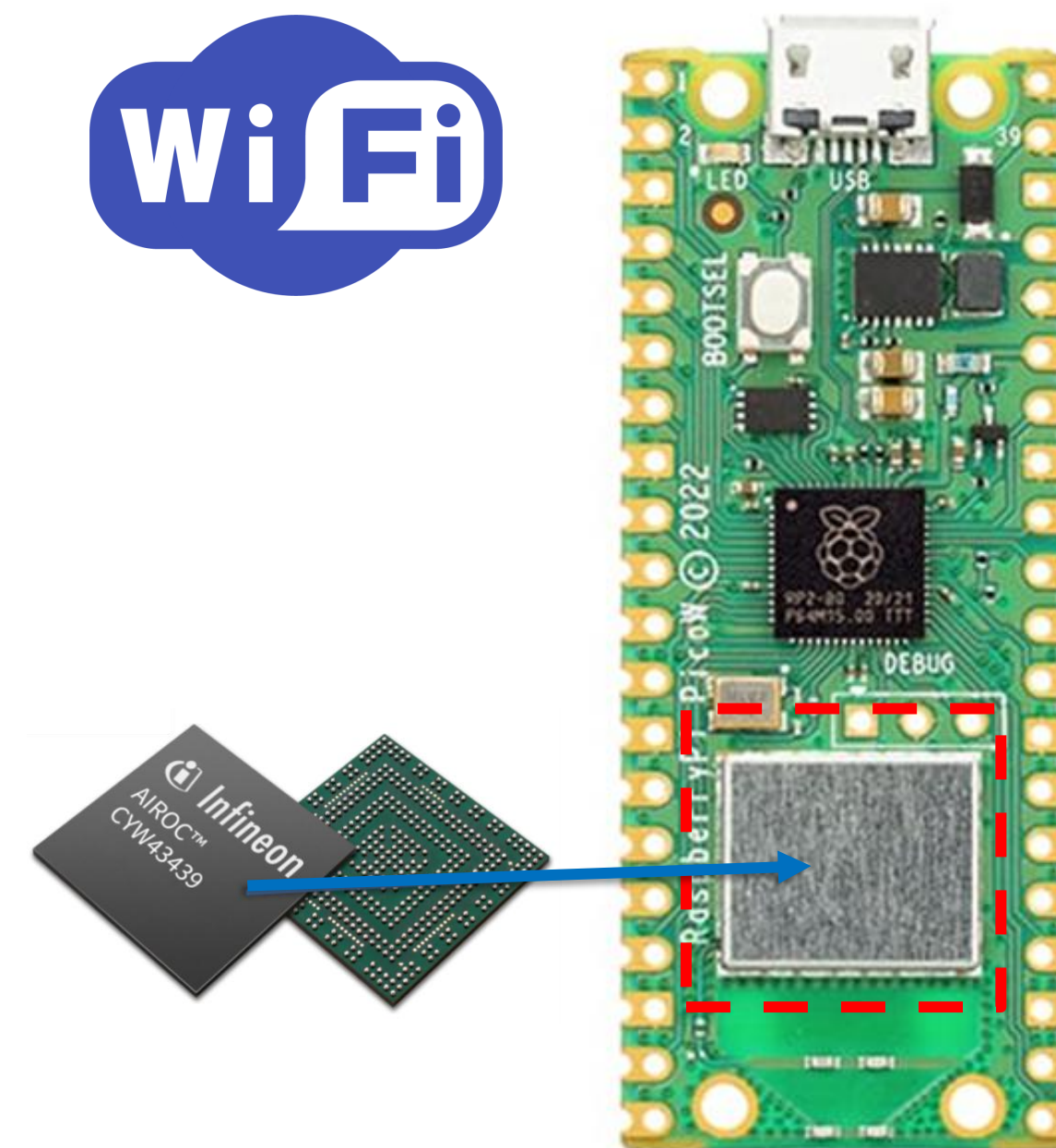
Varredura WI-FI

Trecho do Código C:

```
// Inicializa a arquitetura CYW43 (Wi-Fi e Bluetooth)
if (cyw43_arch_init()) { // Verifica se houve erro na inicialização
    printf("Falha ao inicializar Wi-Fi\n"); // Mensagem de erro
    return 1; // Sai do programa com código de erro
}
printf("Wi-Fi inicializado com sucesso\n"); // Mensagem de sucesso na inicialização
```

```
cyw43_arch_enable_sta_mode(); // Habilita o modo estação (client) para a Pico
```

https://github.com/rmprates84/pico_wifi_cyw43_wifi_scan_02.git



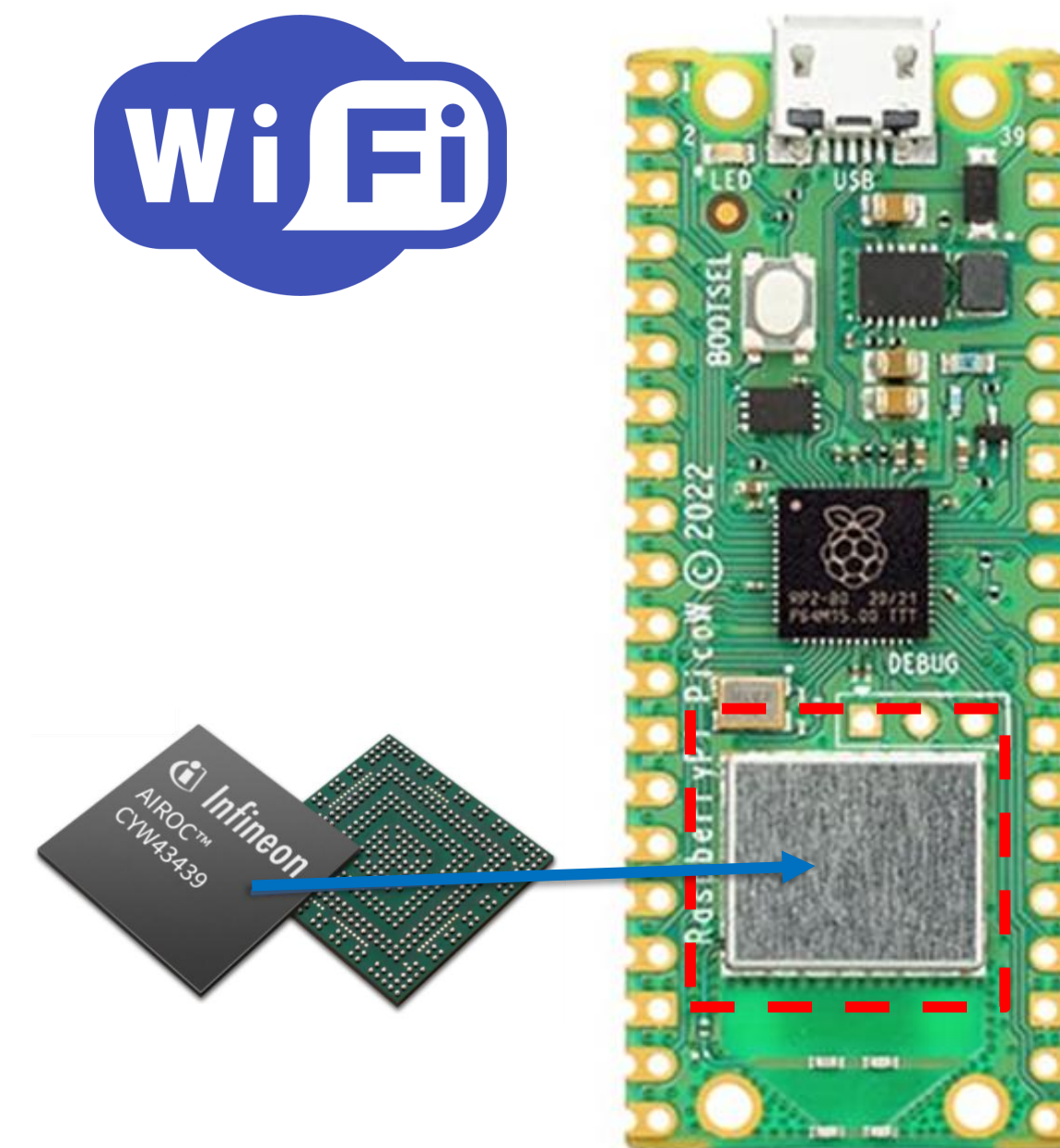
Varredura Wi-Fi

Trecho do Código C:

```
// Callback para tratar os resultados da varredura Wi-Fi
static int scan_result(void *env, const cyw43_ev_scan_result_t *result) {
    if (result) { // Verifica se há um resultado válido

        // Imprime informações detalhadas da rede encontrada
        //(SSID, RSSI, canal, endereço MAC, etc.)
        printf("SSID: %-32s RSSI: %4d CHAN: %3d MAC: %02x:%02x:%02x:%02x:%02x:%02x SEC: %u\n",
            result->ssid, result->rssi, result->channel,
            result->bssid[0], result->bssid[1], result->bssid[2],
            result->bssid[3], result->bssid[4], result->bssid[5],
            result->auth_mode);
    }
    return 0; // Retorna 0 para continuar a varredura
}
```

https://github.com/rmprates84/pico_wifi_cyw43_wifi_scan_02.git



Varredura Wi-Fi

Exemplo de resposta no monitor serial:

---- A porta serial COM21 abriu ----

Iniciando varredura Wi-Fi....

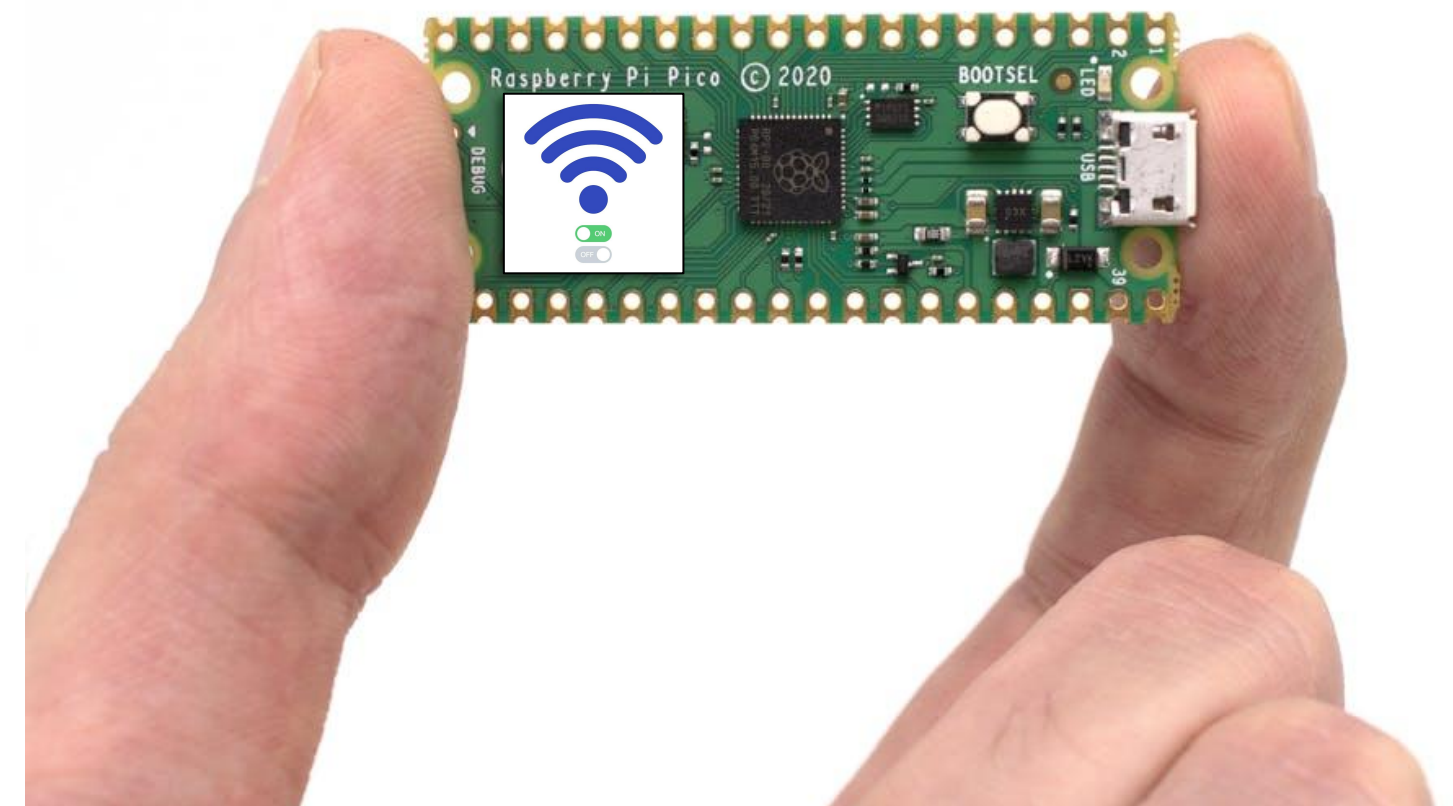
SSID: JR TELECOMxxx	RSSI: -89	CHAN: 1	MAC: 74:6f:XX:a5:7c:XX	SEC: 5
SSID: Jrtelecom-Dxxxx	RSSI: -94	CHAN: 9	MAC: e0:00:XX:a4:4e:XX	SEC: 7
SSID: DEVELOPER JRxxxxx	RSSI: -86	CHAN: 6	MAC: 30:XX:48:19:4b:XX	SEC: 5
SSID: -RAPIDUS- Rafaelxxx	RSSI: -92	CHAN: 8	MAC: 00:eb:XX:85:c9:XX	SEC: 5
SSID: Jrtelecomxx	RSSI: -89	CHAN: 9	MAC: e0:00:XX:a4:4e:XX	SEC: 7
SSID: Ricaxxx	RSSI: -69	CHAN: 11	MAC: 38:6b:XX:8c:37:XX	SEC: 7
SSID: Duplexxxxx	RSSI: -60	CHAN: 11	MAC: f4:XX:60:XX:f9:e0	SEC: 7
SSID: Duplexxx	RSSI: -60	CHAN: 11	MAC: f4:79:60:XX:f9:XX	SEC: 7
SSID: Ricaxxx	RSSI: -65	CHAN: 11	MAC: 38:6b:XX:8c:37:XX	SEC: 7

Varredura concluída



Vamos para o microcontrolador!

Conexão com rede local WI-FI (WLAN)



Conexão com rede local WI-FI

Pretende-se obter:

1. Conexão com uma rede WLAN (*Wireless Local Area Network* – Rede local sem fio);
2. Permite que dispositivos se conectem à internet ou a uma rede local sem fio.
3. Para este caso, será obtido um **endereço IP** (Internet Protocol): rótulo numérico único atribuído a cada dispositivo conectado à internet ou a uma rede local



Conexão com rede local WI-FI

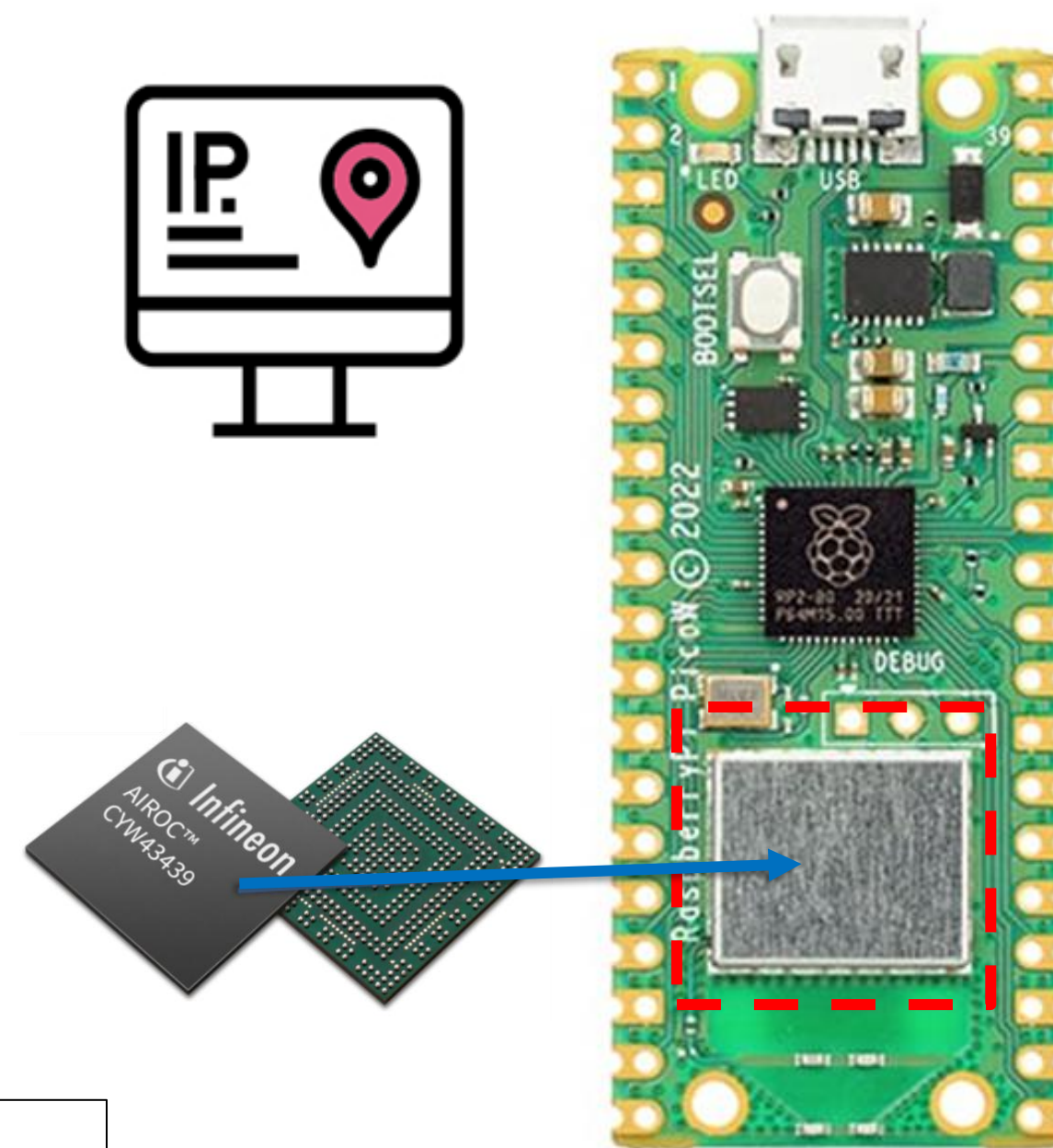
Trecho do Código C:

```
// Conectar à rede WiFi - fazer um loop até que esteja conectado
while(cyw43_arch_wifi_connect_timeout_ms(WIFI_SSID, WIFI_PASSWORD, CYW43_AUTH_WPA2_AES_PSK, 30000) != 0){
    printf("Tentando conexão...\n");
}
printf("Conectado com sucesso! \n");

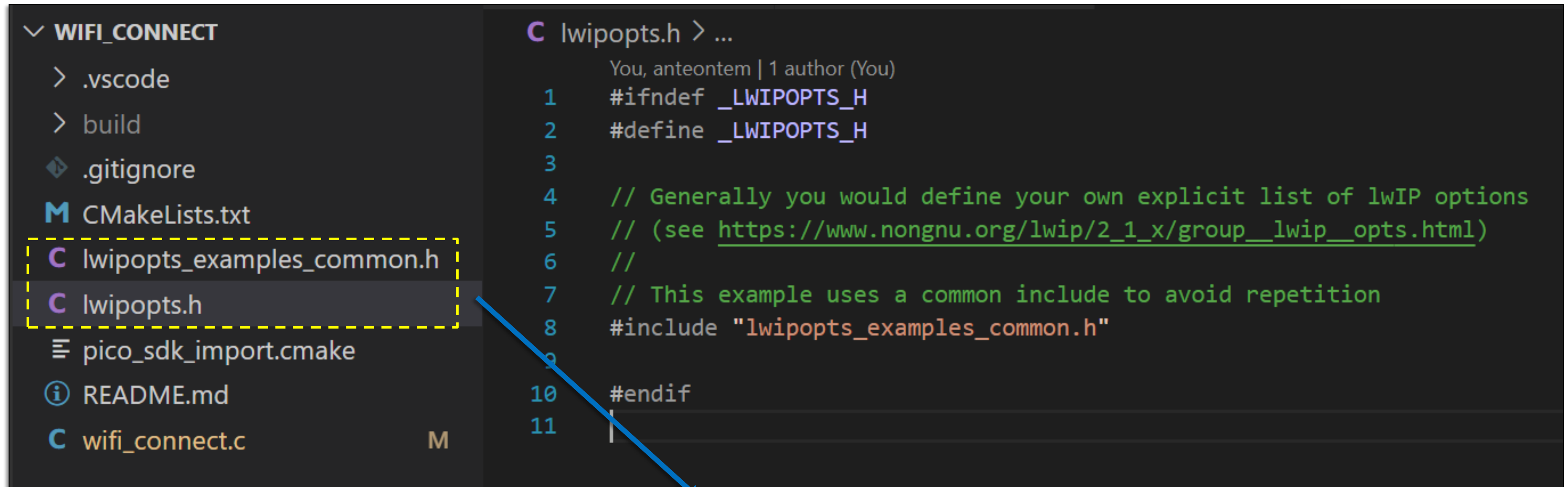
// Caso seja a interface de rede padrão - imprimir o IP do dispositivo.
if (netif_default)
{
    printf("IP do dispositivo: %s\n", ipaddr_ntoa(&netif_default->ip_addr));
}
```

https://github.com/rmprates84/pico_wifi_cyw43_connect_03.git

lwIP 2.1.0
Lightweight IP stack



Conexão com rede local WI-FI



WIFI_CONNECT

- > .vscode
- > build
- 🔍 .gitignore
- M CMakeLists.txt
- lwipopts_examples_common.h
- lwipopts.h
- ☰ pico_sdk_import.cmake
- 📖 README.md
- C wifi_connect.c M

lwipopts.h > ...

```
1  You, antontem | 1 author (You)
2  #ifndef _LWIPOPTS_H
3  #define _LWIPOPTS_H
4  // Generally you would define your own explicit list of lwIP options
5  // (see https://www.nongnu.org/lwip/2\_1\_x/group\_\_lwip\_\_opts.html)
6  //
7  // This example uses a common include to avoid repetition
8  #include "lwipopts_examples_common.h"
9
10 #endif
11
```

lwIP 2.1.0
Lightweight IP stack

Conexão com rede local WI-FI

Trecho de código do CMakeLists.txt:

```
# Initialise the Raspberry Pi Pico SDK
pico_sdk_init()

# Add executable. Default name is the project name, version 0.1

add_executable(wifi_connect wifi_connect.c )

pico_set_program_name(wifi_connect "wifi_connect")
pico_set_program_version(wifi_connect "0.1")

# Modify the below lines to enable/disable output over UART/USB
pico_enable_stdio_uart(wifi_connect 0)
pico_enable_stdio_usb(wifi_connect 1)

# Add the standard library to the build
target_link_libraries(wifi_connect
    pico_cyw43_arch_lwip_threadsafe_background
    pico_stdlib)
```

Exemplo de resposta no monitor serial:

Conectado com sucesso!
IP do dispositivo: 192.168.188.102

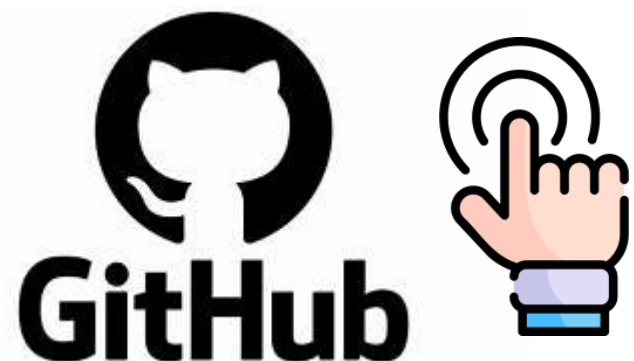
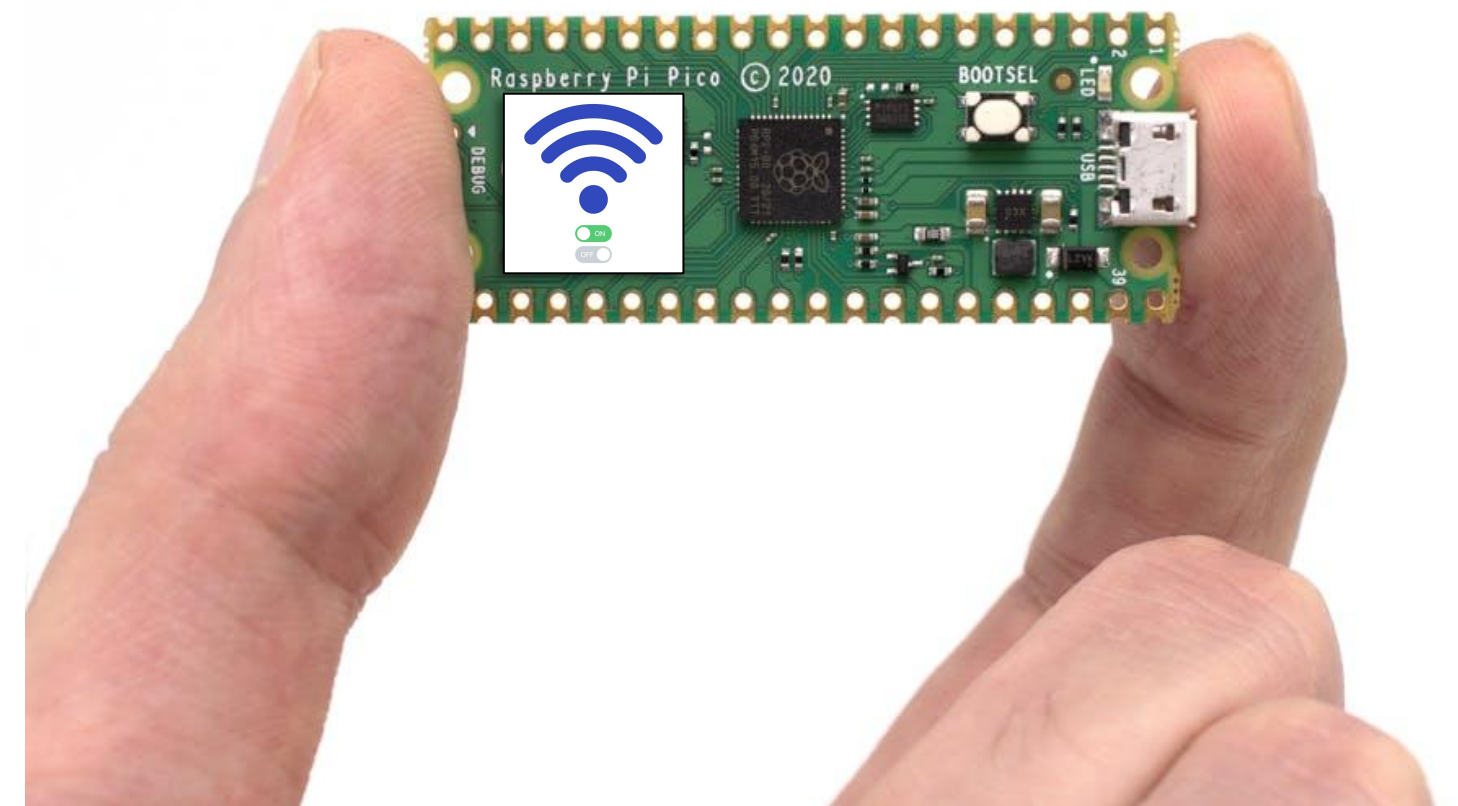
A partir desta conexão, será possível:

- Fazer requisições http ou https;
- **Desenvolver Webservers!**



Vamos para o microcontrolador!

Pico Pi W Webserver



Pico Pi W Webserver

Aplicação desejada:

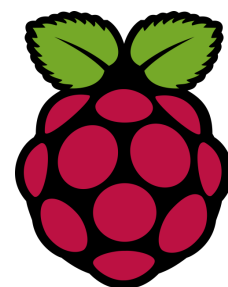
- ✓ Conectar o Raspberry Pi Pico W a uma rede Wi-Fi e configurar um servidor **HTTP básico** para interação com um celular ou computador.
- ✓ O propósito de monitorar a temperatura do RP2040 (sensor integrado) e acender/apagar LEDs na placa de desenvolvimento BitDogLab.



Pico Pi W Webserver

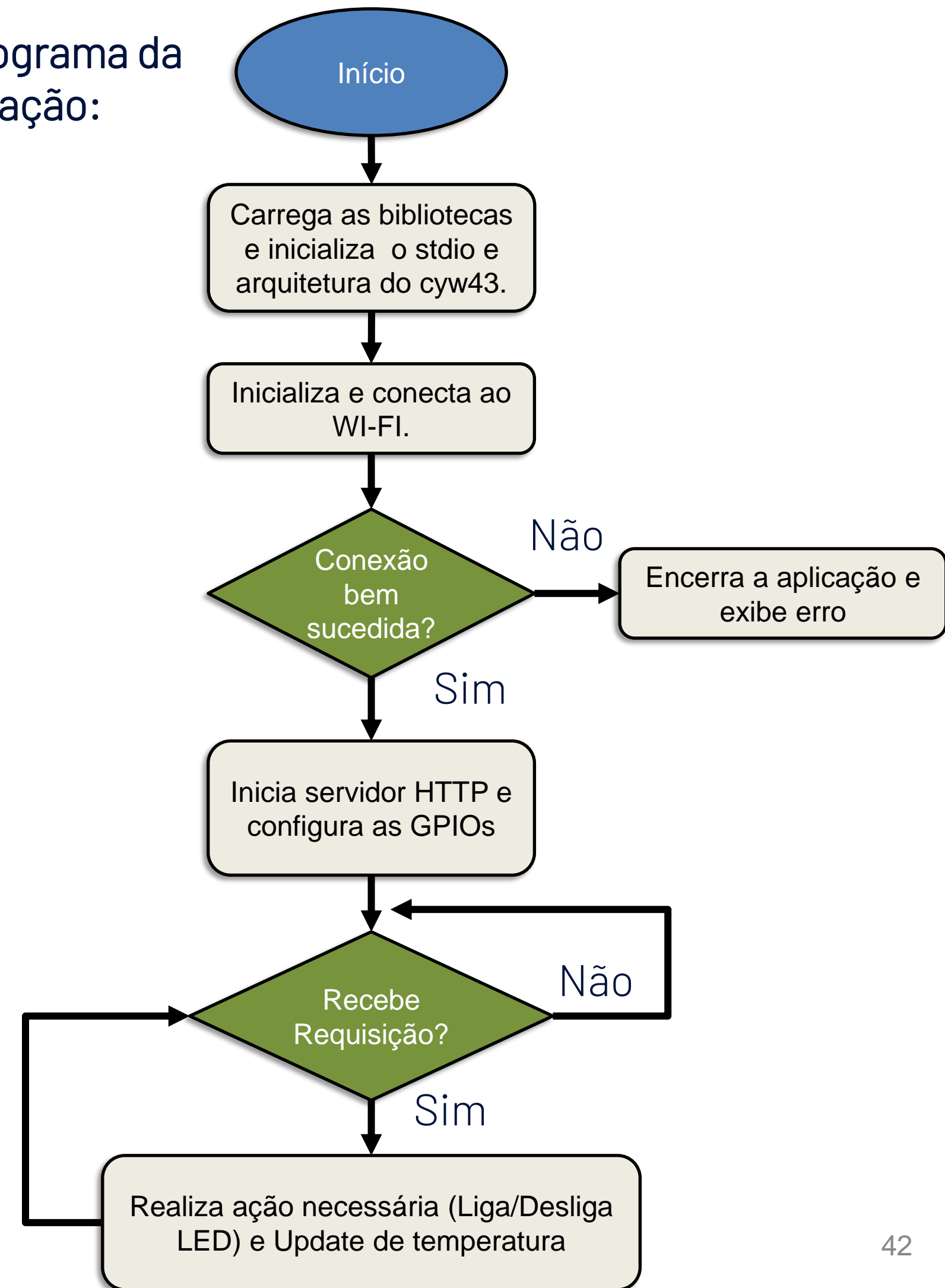
Aplicação desejada:

- ✓ Será utilizada a linguagem C, o SDK e a implementação LwIP (*Lightweight IP Stack*).



lwIP 2.1.0
Lightweight IP stack

Fluxograma da aplicação:



Pico Pi W Webserver

Como irá funcionar:



Requisição HTML

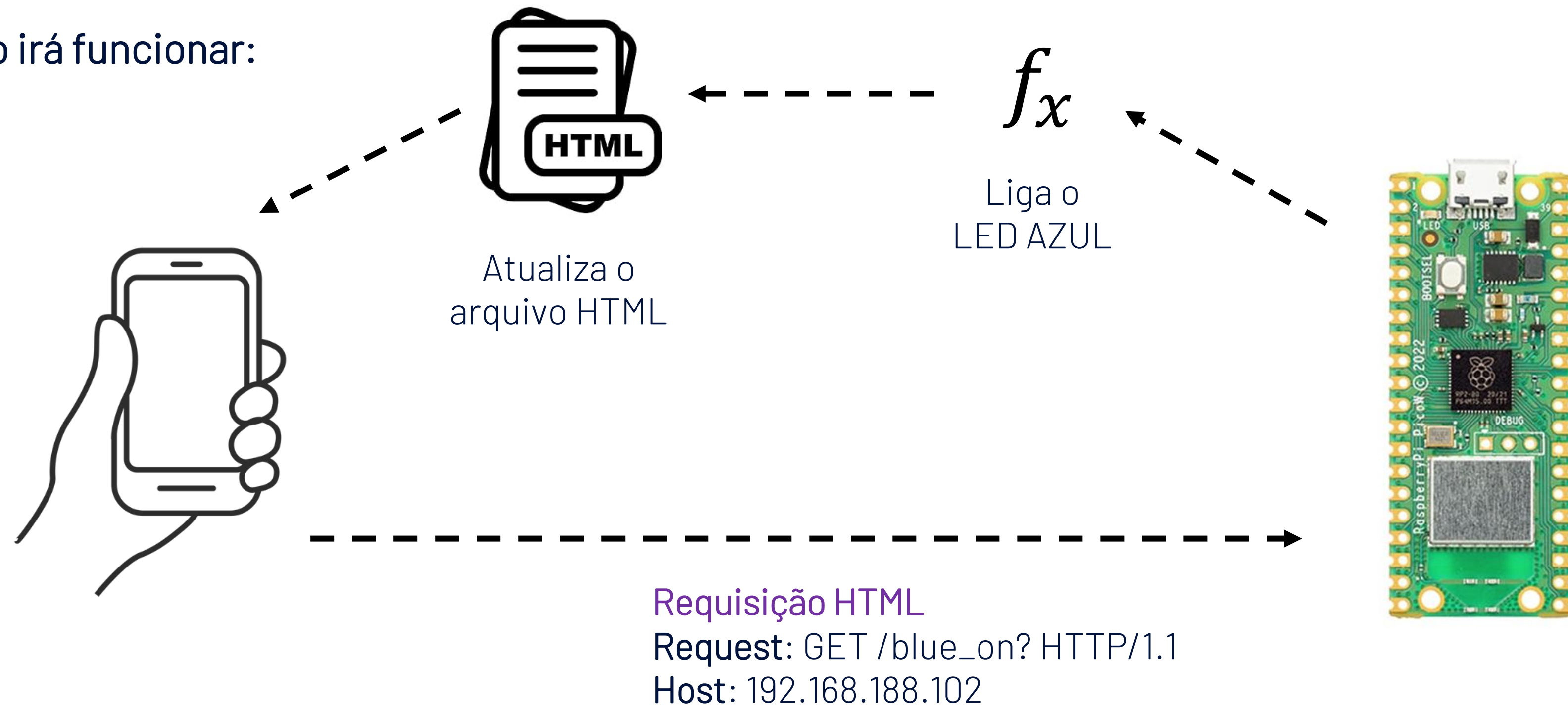
Request: GET /blue_on? HTTP/1.1

Host: 192.168.188.102



Pico Pi W Webserver

Como irá funcionar:



Pico Pi W Webserver

Vamos para o Código! Trecho:

```
// Configura o servidor TCP - cria novos PCBs TCP. É o primeiro passo para estabelecer uma conexão TCP.
struct tcp_pcb *server = tcp_new();
if (!server)
{
    printf("Falha ao criar servidor TCP\n");
    return -1;
}

//vincula um PCB (Protocol Control Block) TCP a um endereço IP e porta específicos.
if (tcp_bind(server, IP_ADDR_ANY, 80) != ERR_OK)
{
    printf("Falha ao associar servidor TCP à porta 80\n");
    return -1;
}

// Coloca um PCB (Protocol Control Block) TCP em modo de escuta, permitindo que ele aceite conexões de entrada.
server = tcp_listen(server);

// Define uma função de callback para aceitar conexões TCP de entrada. É um passo importante na configuração de servidores TCP.
tcp_accept(server, tcp_server_accept);
printf("Servidor ouvindo na porta 80\n");
```


Pico Pi W Webserver

Instruções HTML:



```
// Instruções html do webserver
    snprintf(html, sizeof(html), // Formatar uma string e armazená-la em um buffer de
caracteres
    "HTTP/1.1 200 OK\r\n"
    "Content-Type: text/html\r\n"
    "\r\n"
    "<!DOCTYPE html>\r\n"
    "<html>\r\n"
    "<head>\r\n"
    "<title> Embarcotech - LED Control </title>\r\n"
    "<style>\r\n"
    "body { background-color: #b5e5fb; font-family: Arial, sans-serif; text-
align: center; margin-top: 50px; }\r\n"
    "h1 { font-size: 64px; margin-bottom: 30px; }\r\n"
    "button { background-color: LightGray; font-size: 36px; margin: 10px;
padding: 20px 40px; border-radius: 10px; }\r\n"
    ".temperature { font-size: 48px; margin-top: 30px; color: #333; }\r\n"
    "</style>\r\n"
    "</head>\r\n"
    "<body>\r\n"
    "<h1>Embarcotech: LED Control</h1>\r\n"
    "<form action=\"./blue_on\"><button>Ligar Azul</button></form>\r\n"
    "<form action=\"./blue_off\"><button>Desligar Azul</button></form>\r\n"
    "<form action=\"./green_on\"><button>Ligar Verde</button></form>\r\n"
    "<form action=\"./green_off\"><button>Desligar Verde</button></form>\r\n"
    "<form action=\"./red_on\"><button>Ligar Vermelho</button></form>\r\n"
    "<form action=\"./red_off\"><button>Desligar Vermelho</button></form>\r\n"
    "<p class=\"temperature\">Temperatura Interna: %.2f &deg;C</p>\r\n"
    "</body>\r\n"
    "</html>\r\n",
    temperature);
```

Pico Pi W Webserver

Trecho do código:

```
// Inicializa o conversor ADC
adc_init();
adc_set_temp_sensor_enabled(true);

while (true)
{
    cyw43_arch_poll(); // Necessário para manter o Wi-Fi ativo
    sleep_ms(100);     // Reduz o uso da CPU
}

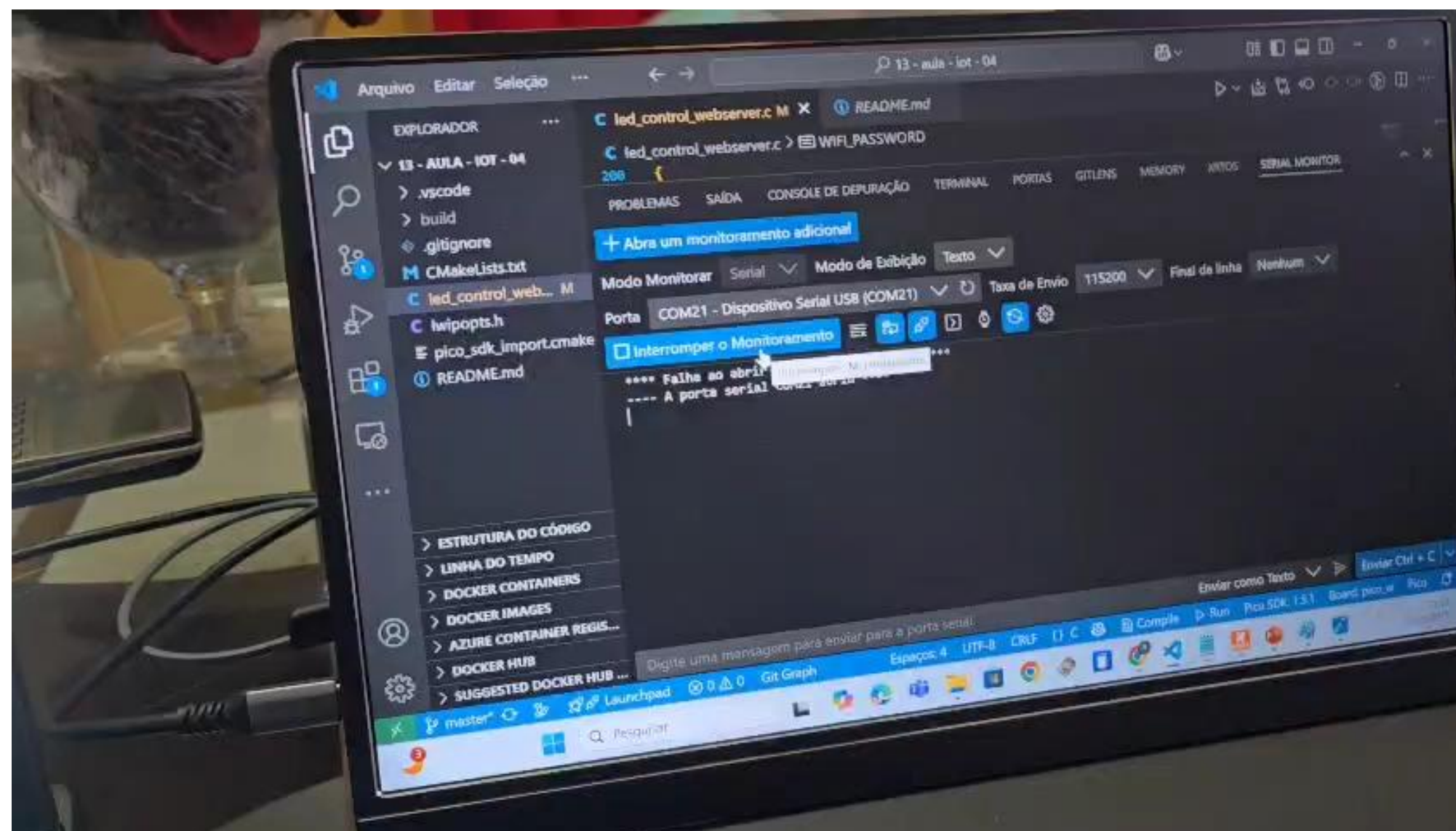
//Desligar a arquitetura CYW43.
cyw43_arch_deinit();
return 0;
}
```

Detalhamento da requisição:

- ✓ **Método:** GET (o cliente está solicitando um recurso específico do servidor)
- ✓ **URL:** /blue_off? (o cliente está solicitando o recurso /blue_off – desligar LED azul)
- ✓ **Host:** 192.168.188.102 (o endereço IP do servidor)
- ✓ **User-Agent:** Mozilla/5.0... (informações sobre o navegador e o sistema operacional do cliente)
- ✓ **Referer:** http://192.168.188.102/blue_on? (a página que o cliente estava anteriormente)

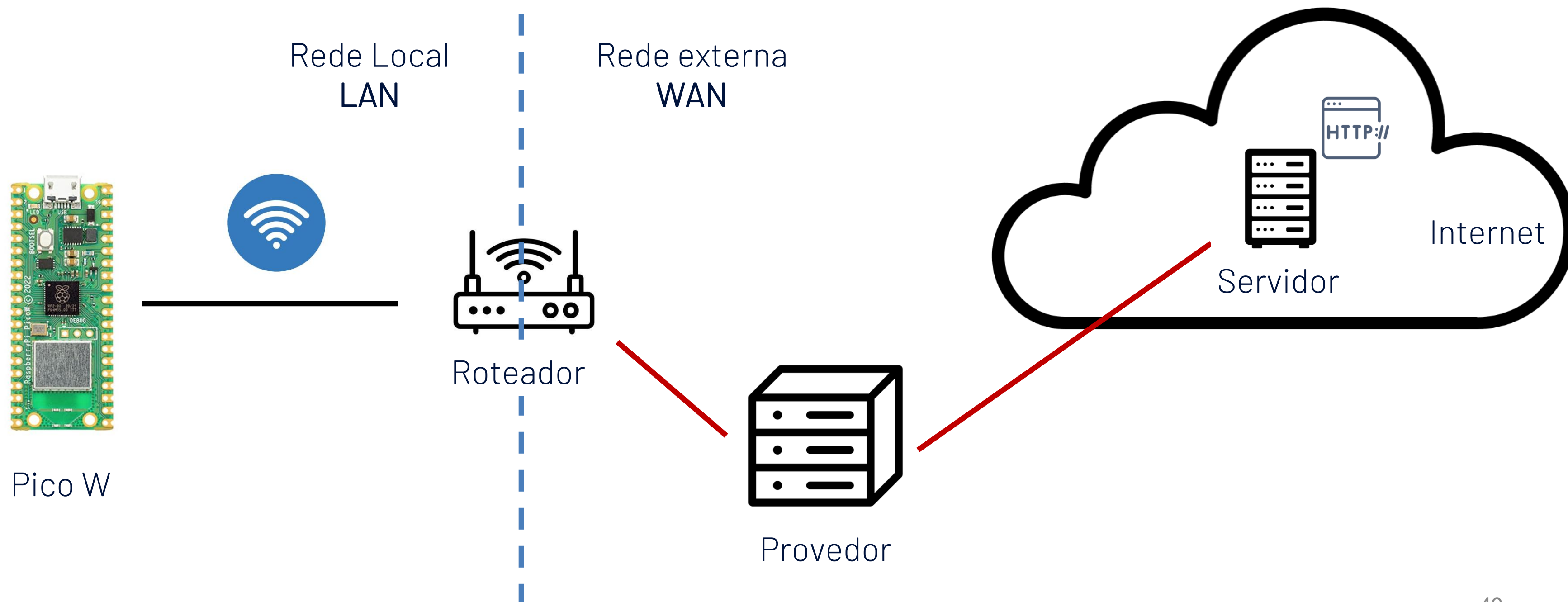
Pico Pi W Webserver

Demonstração do funcionamento do Webserver:



Conteúdo extra – estudo dirigido:

Pergunta: É possível fazer uma requisição http ou https em uma WAN com o Pico W?



Conteúdo extra – estudo dirigido:

Pergunta: É possível fazer uma requisição http ou https em uma WAN com o Pico W?

Veja este exemplo!



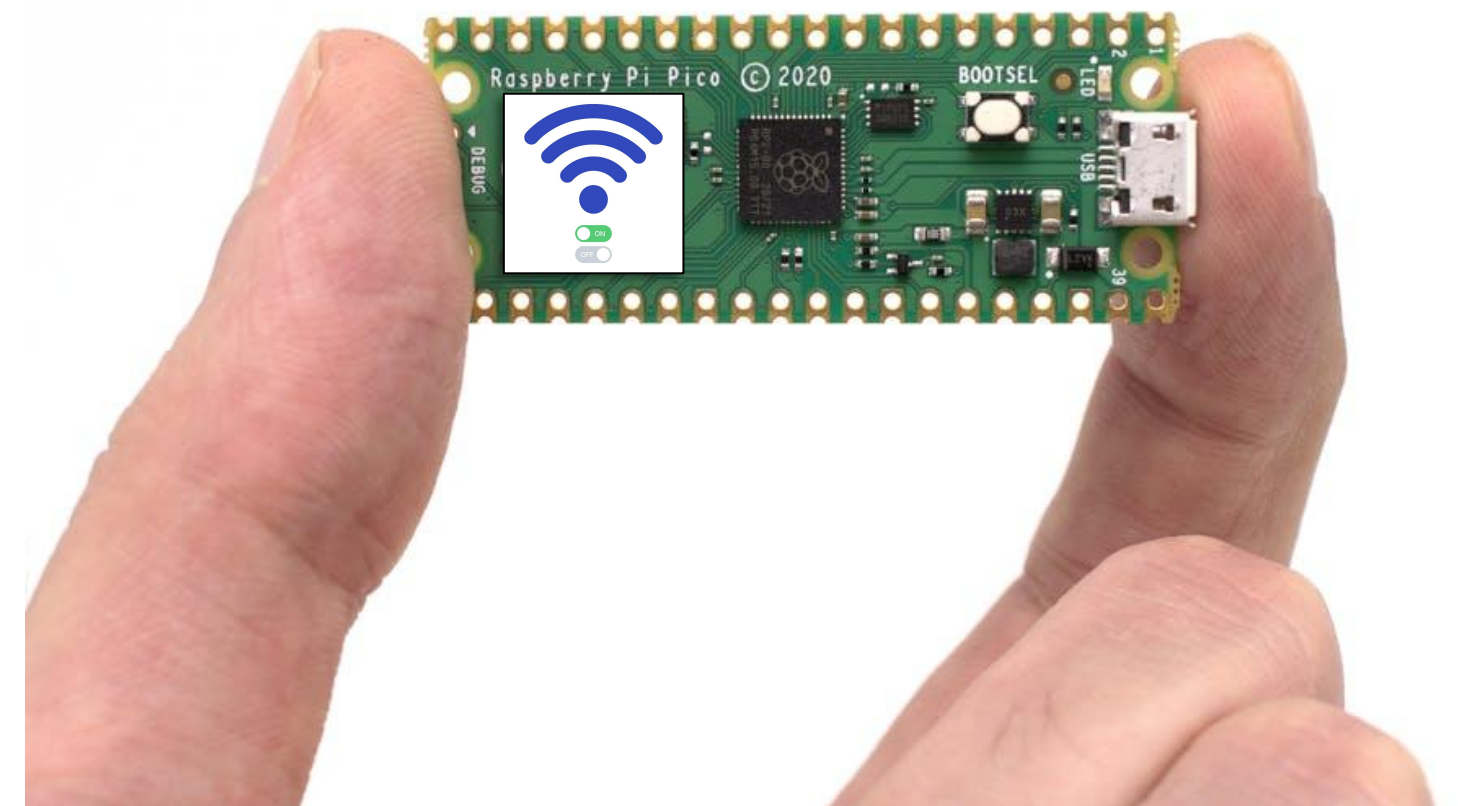
Vamos para o microcontrolador!

Bluetooth Low Energy (BLE)

Servidor
(Periférico)



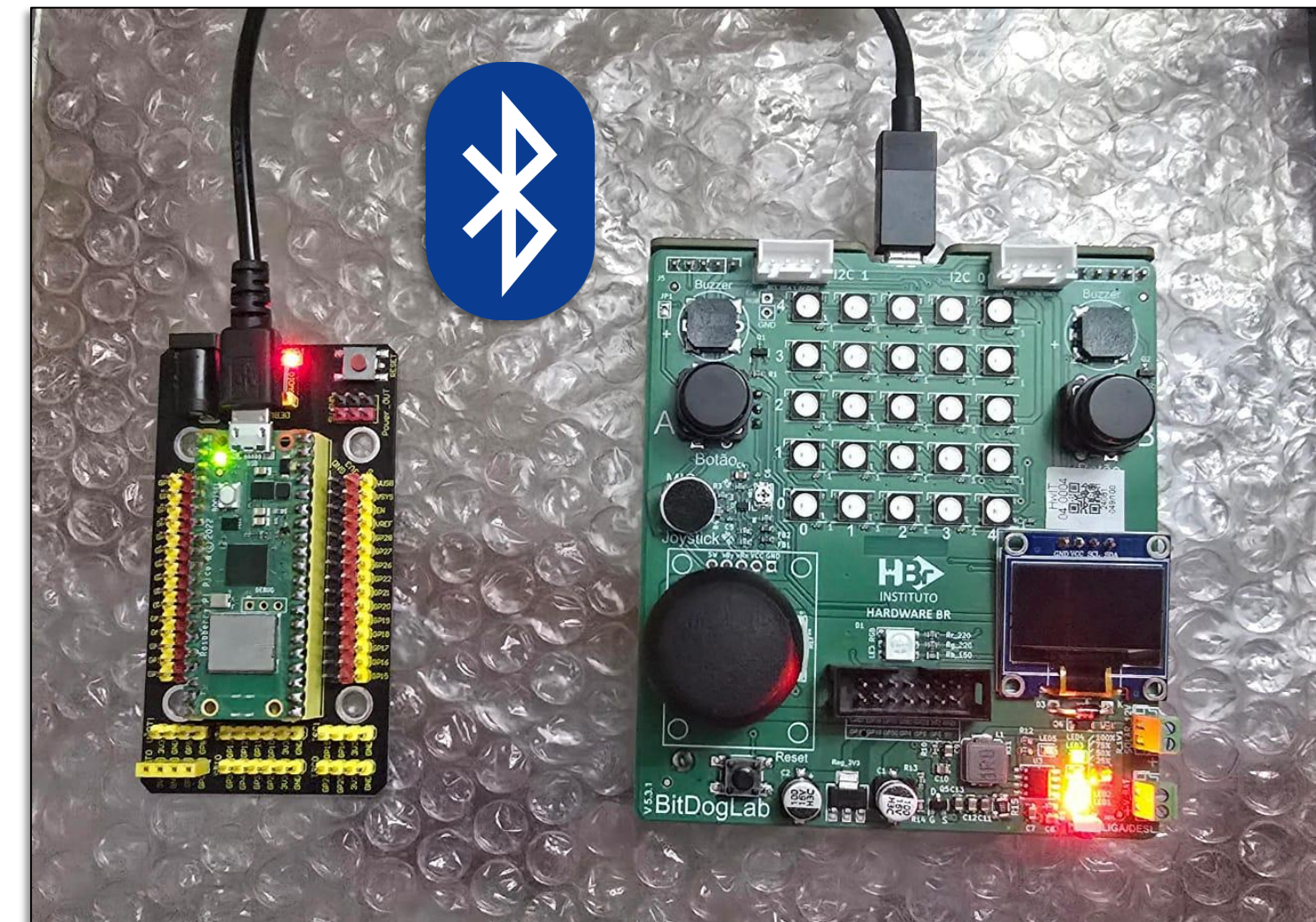
Cliente
(Central)



Bluetooth Low Energy (BLE)

Aplicação desejada:

- ✓ Um dispositivo BLE, tido como “periférico”, anuncia sua presença e os serviços que oferece;
- ✓ Outro dispositivo BLE, tido como “central”, procura por dispositivos periféricos próximos e detecta o anúncio;
- ✓ O dispositivo central se conecta ao periférico;
- ✓ Os dispositivos trocam dados por meio de características (atributos) definidas no perfil do dispositivo.



Bluetooth Low Energy (BLE)

Trecho do código:

```
//Protocolos bluetooth
l2cap_init(); // Inicializa o protocolo L2CAP (Logical Link Control and Adaptation Protocol)
sm_init(); // Inicializa o Security Manager (gerenciamento de segurança Bluetooth)

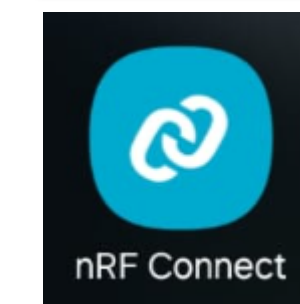
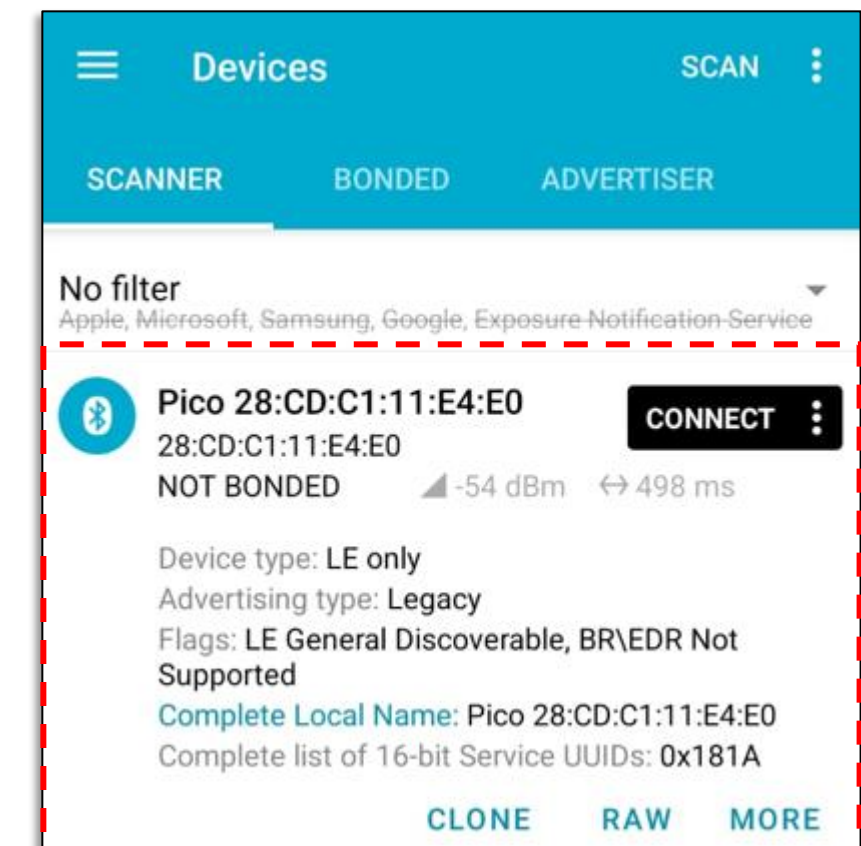
// Configura o servidor ATT (Attribute Protocol), com callbacks
att_server_init(profile_data, att_read_callback, att_write_callback);

// Registra o callback para eventos HCI
// Eventos HCI são notificações do controlador ao host sobre a ocorrência de eventos.
hci_event_callback_registration.callback = &packet_handler;
hci_add_event_handler(&hci_event_callback_registration);

// Registra o callback para pacotes do servidor ATT
att_server_register_packet_handler(packet_handler);

// Configura e adiciona o temporizador de batimento cardíaco
heartbeat.process = &heartbeat_handler; // Define a função de processamento do batimento
btstack_run_loop_set_timer(&heartbeat, HEARTBEAT_PERIOD_MS); // Configura o temporizador
btstack_run_loop_add_timer(&heartbeat); // Adiciona o temporizador à fila

// Liga o controlador HCI para ativar o Bluetooth
hci_power_control(HCI_POWER_ON);
```



Bluetooth Low Energy (BLE)

Trecho do código:

```
//Protocolos bluetooth
l2cap_init(); // Inicializa o protocolo L2CAP (Logical Link Control and Adaptation Protocol)
sm_init(); // Inicializa o Security Manager (gerenciamento de segurança Bluetooth)

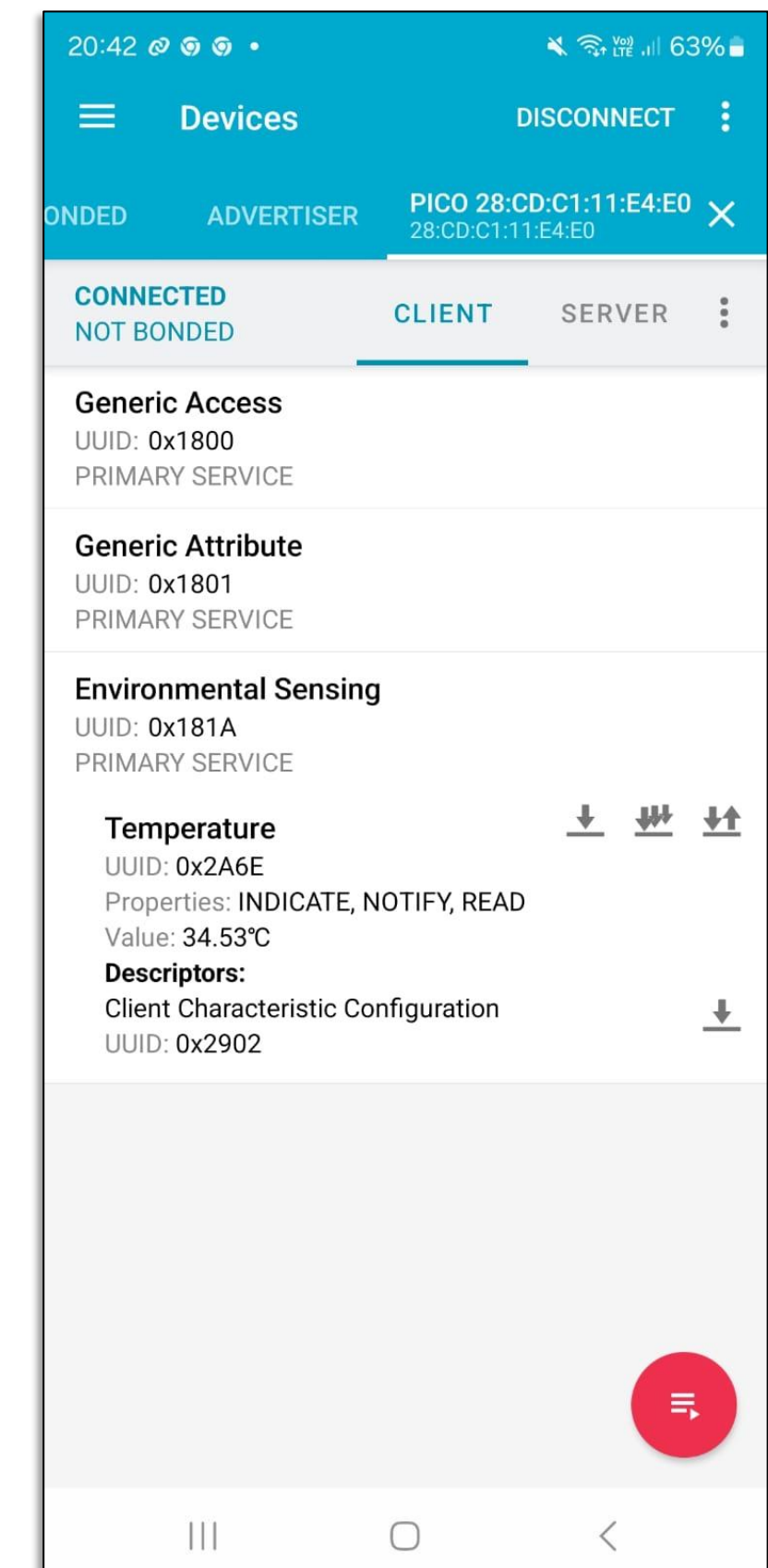
// Configura o servidor ATT (Attribute Protocol), com callbacks
att_server_init(profile_data, att_read_callback, att_write_callback);

// Registra o callback para eventos HCI
// Eventos HCI são notificações do controlador ao host sobre a ocorrência de eventos.
hci_event_callback_registration.callback = &packet_handler;
hci_add_event_handler(&hci_event_callback_registration);

// Registra o callback para pacotes do servidor ATT
att_server_register_packet_handler(packet_handler);

// Configura e adiciona o temporizador de batimento cardíaco
heartbeat.process = &heartbeat_handler; // Define a função de processamento do batimento
btstack_run_loop_set_timer(&heartbeat, HEARTBEAT_PERIOD_MS); // Configura o temporizador
btstack_run_loop_add_timer(&heartbeat); // Adiciona o temporizador à fila

// Liga o controlador HCI para ativar o Bluetooth
hci_power_control(HCI_POWER_ON);
```



Bluetooth Low Energy (BLE)

Servidor:

```
---- Porta serial reaberta COM5 ----  
BTstack up and running on  
28:CD:C1:11:E4:E0.  
Write temp 34.53 degc  
Write temp 34.53 degc  
Write temp 34.53 degc  
Write temp 34.53 degc  
Write temp 34.53 degc  
Write temp 34.53 degc
```



Periférico

Cliente:

```
---- Porta serial reaberta COM21 ----  
BTstack up and running on 28:CD:C1:0D:51:9F.  
Connecting to device with addr 28:CD:C1:11:E4:E0.  
read temp 35.00 degc  
read temp 35.00 degc  
read temp 35.47 degc  
read temp 35.00 degc  
read temp 35.00 degc  
read temp 35.00 degc  
read temp 35.00 degc
```



Central



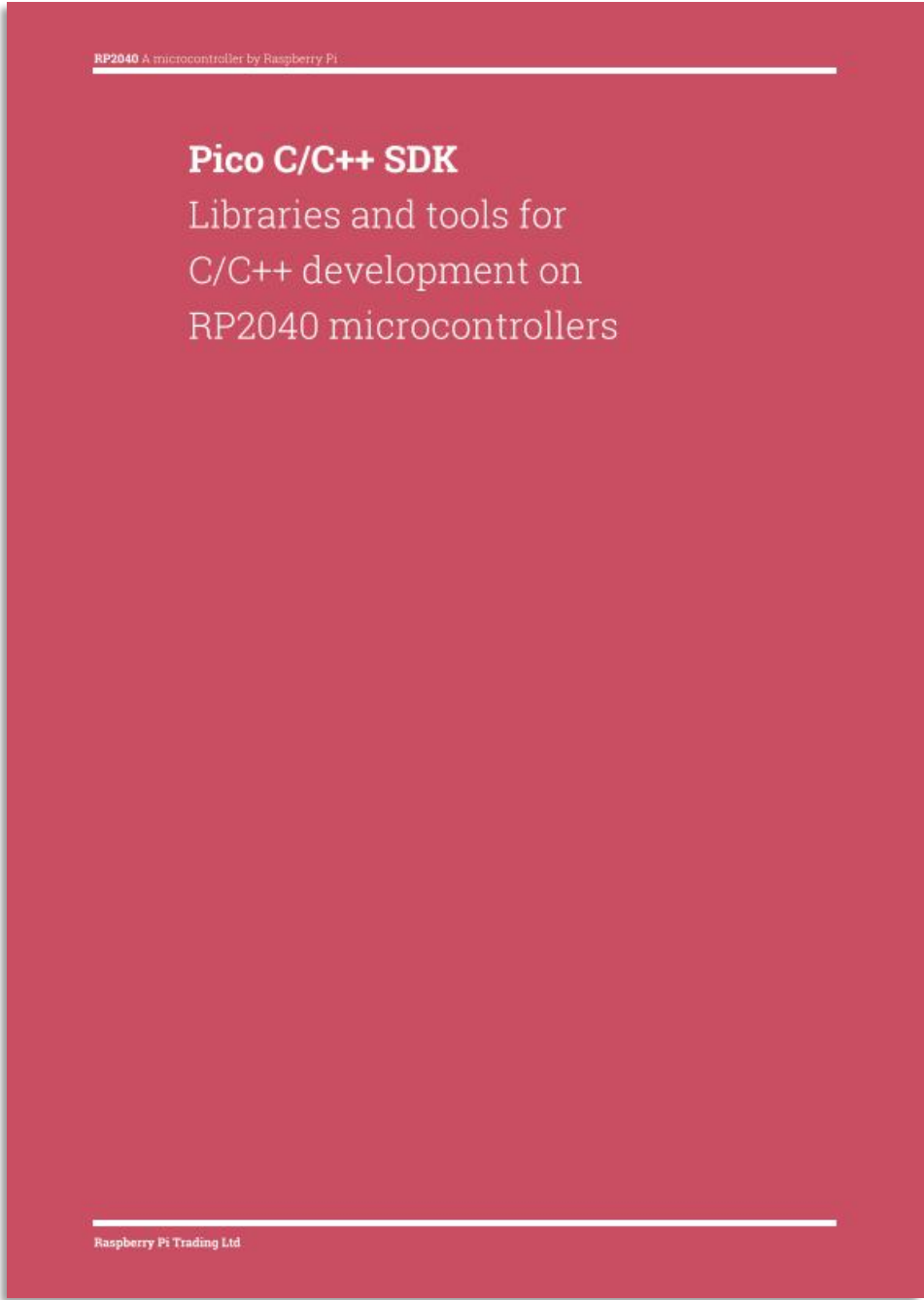
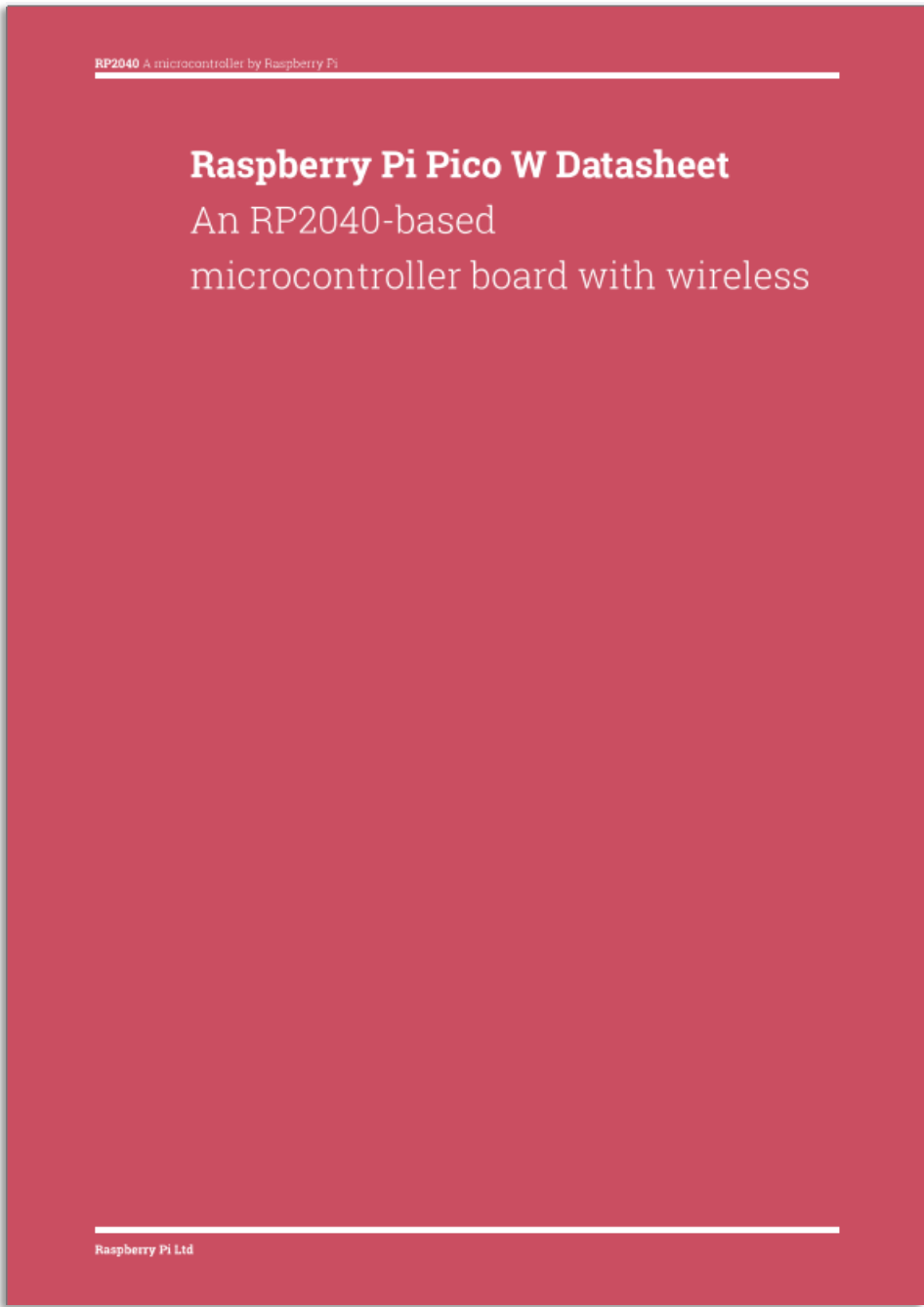
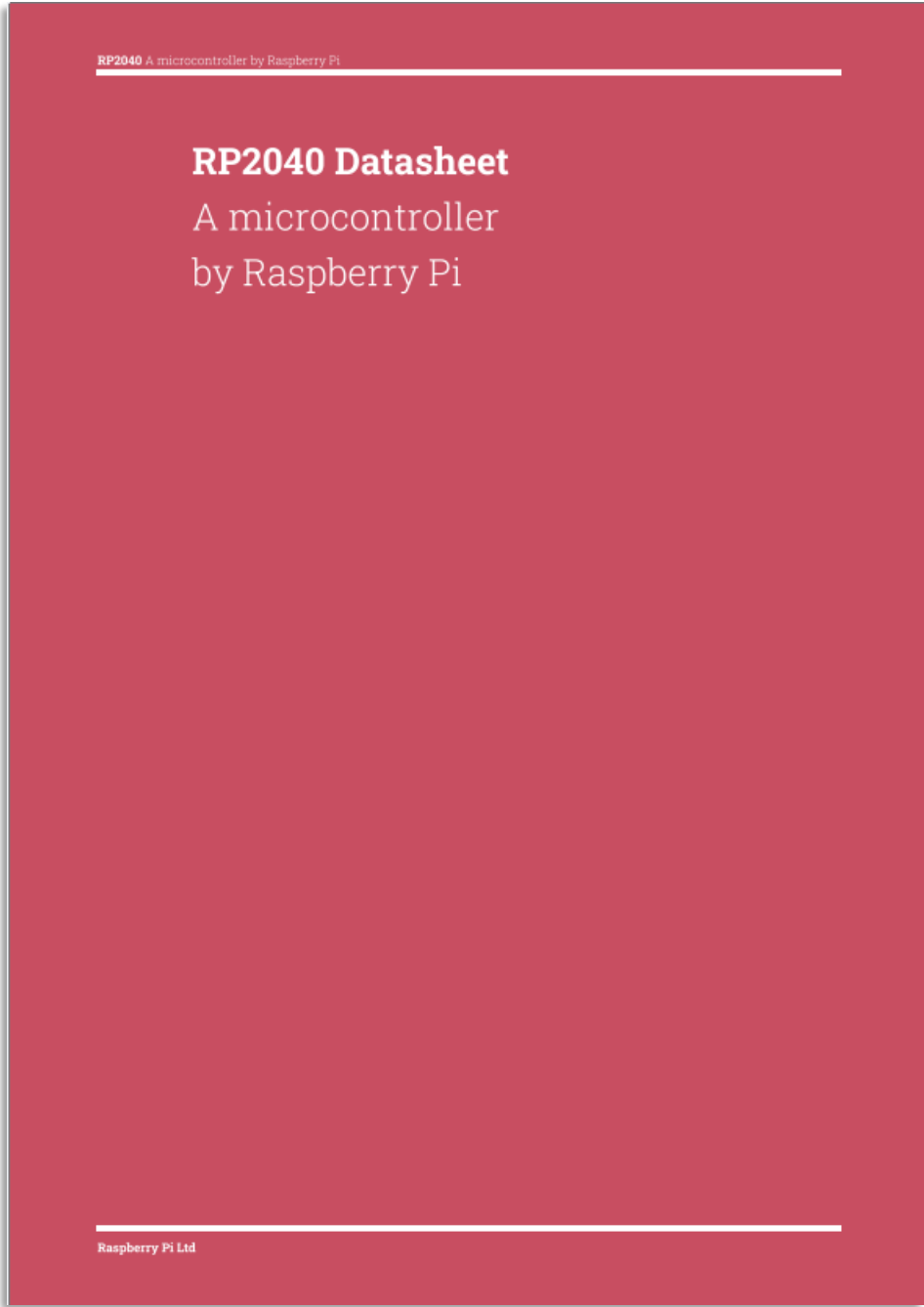
Conclusões



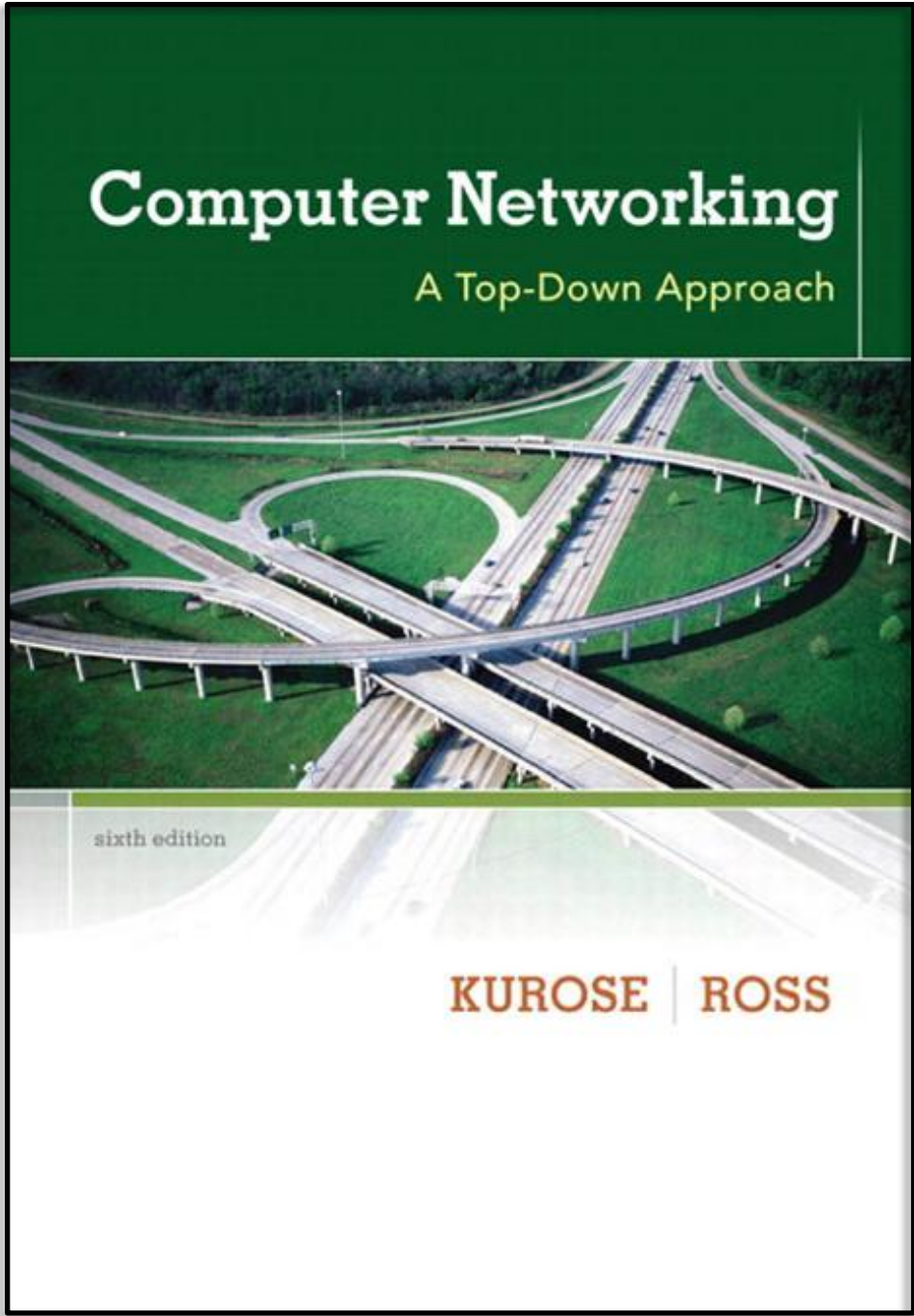
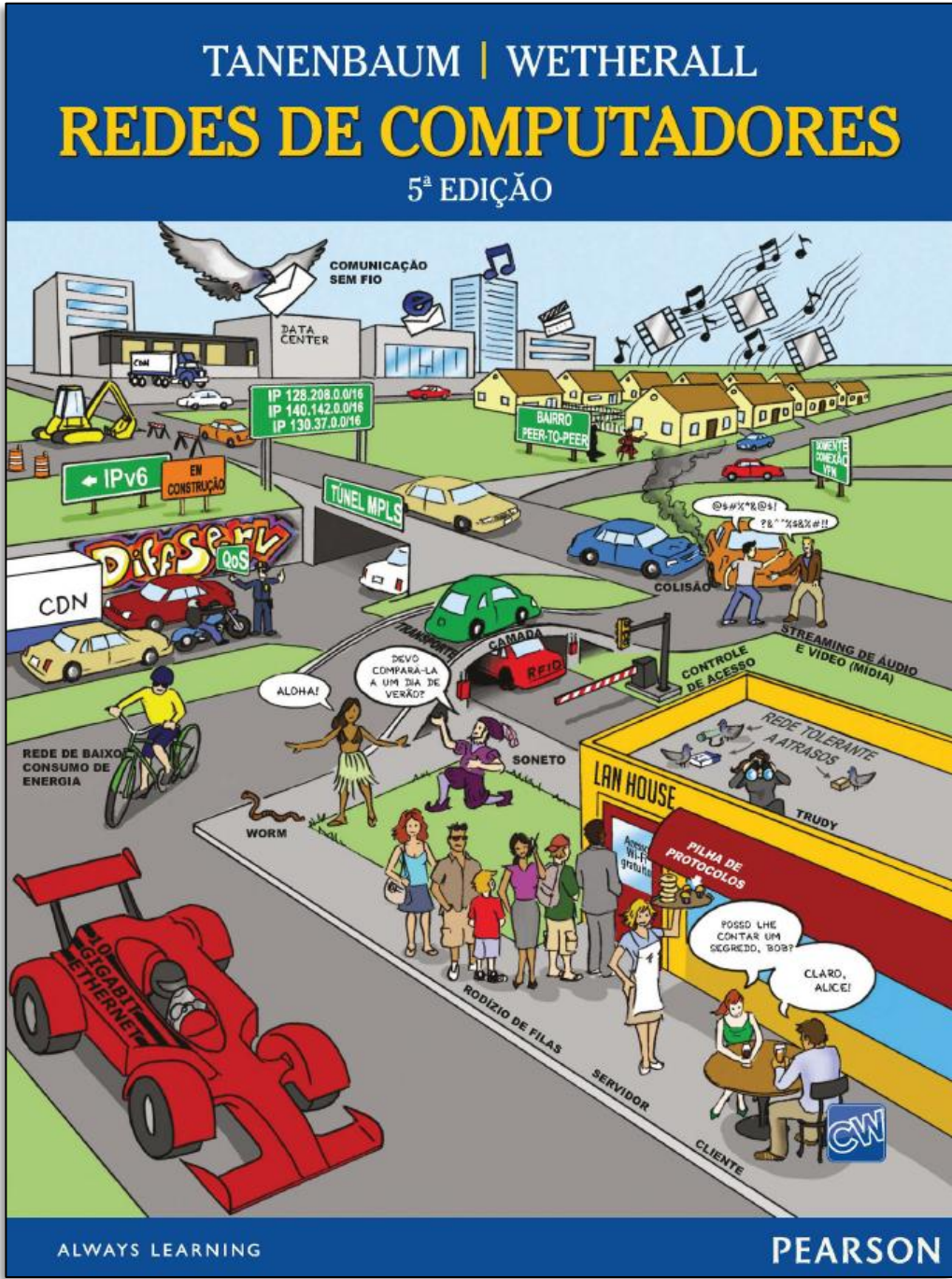
Conclusões

- As interfaces de comunicação sem fio possuem características particulares que podem viabilizar diversos projetos de IoT.
- O Raspberry Pi Pico W conta com as tecnologias de Wi-Fi e Bluetooth integradas.
- Explorando a linguagem C e o SDK, é possível desenvolver rotinas de configuração do microcontrolador RP2040 e do módulo CYW43439
- Nesta aula, foi possível configurar redes locais (LANs) e redes pessoais (PANs) para o desenvolvimento de diferentes aplicações sem fio.

Bibliografia Recomendada



Bibliografia Recomendada - Complementar



Please note that Cypress is an Infineon Technologies Company. The document following this cover page is marked as “Cypress” document as this is the company that originally developed the product. Please note that Infineon will continue to offer the product to new and existing customers as part of the Infineon product portfolio.

Continuity of document content
The fact that Infineon offers the following product as part of the Infineon product portfolio does not lead to any changes to this document. Future revisions will occur when appropriate, and any changes will be set out on the document history page.

Continuity of ordering part numbers
Infineon continues to support existing part numbers. Please continue to use the ordering part numbers listed in the datasheet for ordering.

www.infineon.com

Comunicação em IoT

RESIDÊNCIA – Aula Síncrona (06/05/2025)

Prof. Dr Ricardo Menezes Prates

Executores:



INSTITUTO FEDERAL
São Paulo



INSTITUTO FEDERAL
Rio Grande do Norte



INSTITUTO FEDERAL
Maranhão



INSTITUTO FEDERAL
Ceará



INSTITUTO
HARDWARE BR

Coordenação:



Iniciativa:



GOVERNO FEDERAL
BRASIL
UNIÃO E RECONSTRUÇÃO