**Troca de dados por WIFI entre ESP32 e um software em C#**

Neste projeto realizo a comunicação e troca de informações via WIFI entre ESP32 e um software em um computador em uma rede local.

Clique [AQUI](https://drive.google.com/file/d/1ckEA9WkTQv2Epf-S-UkcWRzVbW1uzwTh/view?usp=sharing) para fazer download dos arquivos.

Índice da página

1. [Troca de dados por WIFI entre ESP32 e um software em C#](https://iurkiv.com.br/troca-de-dados-por-wifi-entre-esp32-e-um-software-em-c/#troca-de-dados-por-wifi-entre-esp32-e-um-software-em-c)
   * [ESP32](https://iurkiv.com.br/troca-de-dados-por-wifi-entre-esp32-e-um-software-em-c/#esp32)
   * [Software em C#](https://iurkiv.com.br/troca-de-dados-por-wifi-entre-esp32-e-um-software-em-c/#software-em-c)

**ESP32**

Configuro o ESP32 como o servidor com IP local fixo. O endereço escolho um qualquer que seja válido no range do roteador.  
O ESP32 aguarda um cliente (software) se conectar. Após conectado ele apenas retorna a string que recebe do cliente com uma mensagem de confirmação.

/\*

Comunicação e controle WIFI entre software e ESP32-WROOM

Feito como exemplo para treinar e ter como base para comunicação e controle WIFI.

ESP32 está no modo servidor e programa se conecta como cliente.

ESP32 tem IP fixo

Compilador: Arduino IDE 2.2.1

Autor: Anderson Iurkiv

Janeiro 2024

\*/

// Bibliotecas

#include <WiFi.h>

// Definições para a configuração do IP

IPAddress ip(192,xxx,xxx,xxx); // IP do ESP. Tem de colocar um IP válido na faixa do roteador.

IPAddress gateway(xxx,xxx,xxx,xxx); // IP do servidor.

IPAddress subnet(xxx,xxx,xxx,xxx); // Mascara de rede. Colocar a mesma do roteador.

// Variáveis

const char \*ssid = "NomeDaRede"; // Inserir aqui o nome da rede WIFI.

const char \*password = "SenhaDaRede"; // Inserir aqui a senha da rede WIFI.

int baudRate = 115200; // Velocidade da comunicação.

// Objetos e variáveis

WiFiServer server(80); // Declaração do servidor. Parâmetro é o número da porta.

WiFiClient client; // Declaração do cliente para aceitar a conexão.

String strRead = ""; // String de leitura

String strWrite = ""; // String de escrita

// Declaração de funções

void servidor\_escuta\_responde(); // Função que controla o recebimento e envio de mensagens.

void setup()

{

Serial.begin(baudRate); // Inicia a serial

}

// Loop infinito

void loop()

{

if(WiFi.status() != WL\_CONNECTED) // Aguarda conexão

{

Serial.println("\nWiFi desconectado");

Serial.print("Conectando-se em ");

Serial.println(ssid);

WiFi.config(ip, gateway, subnet); // Configurando o WIFI.

WiFi.begin(ssid, password); // Conectando a rede

while(WiFi.status() != WL\_CONNECTED) // Aguarda conexão.

{

delay(1000); // Verifica conexão a cada 1s

Serial.print(".");

}

Serial.println("\nWiFi conectado");

Serial.print("IP: ");

Serial.println(WiFi.localIP()); // Mostra o endereço IP

server.begin(); // Inicializa modo servidor

}

servidor\_escuta\_responde(); // Chama a função de controle das mensagens.

}

/\* Função servidor\_escuta\_responde()

Aceita a conexão somente de 1 cliente.

Recebe uma mensagem, trata e retorna.

\*/

void servidor\_escuta\_responde()

{

int posRead = 0;

char aux;

client = server.available(); // Aguarda cliente conectar.

if(client) // Se cliente conectado, faça

{

Serial.println("Cliente conectado.");

while(client.connected()) // Permanece em loop enquanto conectado.

{

if(client.available()) // Se existir dados para ler, faça

{

strRead = client.readString(); // Lê uma string de bytes

strWrite = "Mensagem recebida: " + strRead; // Monta a string de retorno.

client.print(strWrite); // Retorna a string por WIFI

Serial.println(strWrite); // Retorna a string por Serial

strRead = ""; // Limpa buffer

strWrite = "";

}

}

client.stop(); // Finaliza conexão.

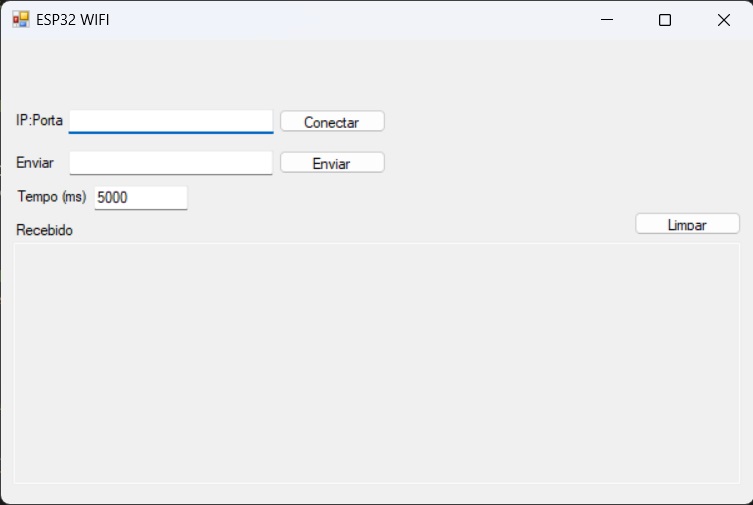
Serial.println("Cliente desconectado.");

}

}

**Software em C#**

Configuro o software como cliente.  
Criei uma interface simples para digitar as informações de conexão (IP e Porta), um campo para digitar a string de envio e um campo que mostra a string recebida.



Abaixo o código do software que criei em C#.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

/\*

\* Comunicação e controle WIFI entre software e ESP32-WROOM

\* Feito como exemplo para treinar e ter como base para comunicação e controle WIFI.

\* Software é configurado como cliente.

\*

\* Compilador: Visual Studio Community 2022 - 17.6.5

\* Autor: Anderson Iurkiv

\* Janeiro 2024

\*/

namespace WifiClient

{

public partial class Form1 : Form

{

// Objetos e variáveis

private TcpClient client;

private NetworkStream stream;

private bool conectado = false;

public Form1()

{

InitializeComponent();

// Iniciando variáveis globais.

textBoxIP.Text = "192.168.15.101:80";

textBoxTempo.Text = "5000";

textBoxRecebido.Enabled = false;

client = null;

stream = null;

}

/\* Evento buttonConectar\_Click()

\* Controla a ação de clicar no botão conectar.

\*/

private void buttonConectar\_Click(object sender, EventArgs e)

{

// Se não estiver conectado, conecta.

if (!conectado)

{

// Só tenta conectar se o campo com o IP de conexão não for nulo.

if (!string.IsNullOrEmpty(textBoxIP.Text))

{

// Configura botões, campos ...

client = new TcpClient();

textBoxIP.Enabled = false;

textBoxTempo.Enabled = false;

buttonConectar.Enabled = false;

buttonConectar.Text = "Conectando";

string[] strCon = textBoxIP.Text.Split(':'); // Separa a string em IP e porta

client.Connect(strCon[0], Convert.ToInt32(strCon[1])); // Conecta

// Se conectar

if (client.Connected)

{

stream = client.GetStream();

buttonConectar.Enabled = true;

buttonConectar.Text = "Desconectar";

conectado = true;

}

// Se não conectar, avisa e reconfigura botões e campos.

else

{

buttonConectar.Enabled = true;

buttonConectar.Text = "Conectar";

textBoxTempo.Enabled = true;

textBoxRecebido.Text = "Erro ao conectar.";

client.Close();

client = null;

}

}

}

// Se não, desconecta

else

{

buttonConectar.Enabled = false;

buttonConectar.Text = "Desconectando";

client.Close();

stream.Close();

client = null;

stream = null;

textBoxIP.Enabled = true;

buttonConectar.Enabled = true;

textBoxTempo.Enabled = true;

buttonConectar.Text = "Conectar";

conectado = false;

}

}

/\* Evento buttonEnviar\_Click()

\* Controla a ação de enviar dados.

\*/

private void buttonEnviar\_Click(object sender, EventArgs e)

{

// Só envia se estiver conectado

if (conectado)

{

byte[] dados = Encoding.ASCII.GetBytes(textBoxEnviar.Text);

stream.Write(dados, 0, dados.Length);

textBoxEnviar.Clear();

textBoxRecebido.Clear();

// Abre a função que aguarda o recebimento da mensagem em outra thread.

Task.Run(() => aguarda\_receber\_dados(Convert.ToInt64(textBoxTempo.Text)));

}

}

/\* Função aguarda\_receber\_dados()

\* Aguarda é retorno da mensagem até o tempo configurado no textBoxTempo

\*/

private void aguarda\_receber\_dados(long tempo)

{

bool recebido = false;

Stopwatch stopWatch = new Stopwatch();

stopWatch.Start();

while(stopWatch.ElapsedMilliseconds < tempo)

{

if (stream.DataAvailable)

{

stopWatch.Stop();

// Como abro a função em uma thread, tenho de usar delegate para alterar o conteúdo do textBox

Invoke(new Action (() => atualiza\_textBoxRecebido(stopWatch.ElapsedMilliseconds)));

recebido = true;

break;

}

Thread.Sleep(100);

}

if(!recebido)

{

stopWatch.Stop();

Invoke(new Action(() => textBoxRecebido.Text = "Tempo expirou"));

}

}

/\* Função atualiza\_textBoxRecebido()

\* Altera o conteúdo do textBoxRecebido.

\* Recebe como parâmentro o tempo de retorno em ms

\*/

private void atualiza\_textBoxRecebido(long tempo)

{

byte[] dados = new byte[100];

int tam = stream.Read(dados, 0, 100);

string str = "Tempo: " + tempo.ToString() + " ms - " + Encoding.ASCII.GetString(dados, 0, tam);

textBoxRecebido.Text = str;

}

/\* Função Form1\_FormClosing()

\* Garante que desconecta antes de fechar o programa.

\*/

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (client != null)

if (client.Connected)

client.Close();

}

/\* Função buttonLimpar\_Click()

\* Limpa o textBoxRecebido

\*/

private void buttonLimpar\_Click(object sender, EventArgs e)

{

textBoxRecebido.Clear();

}

}

}