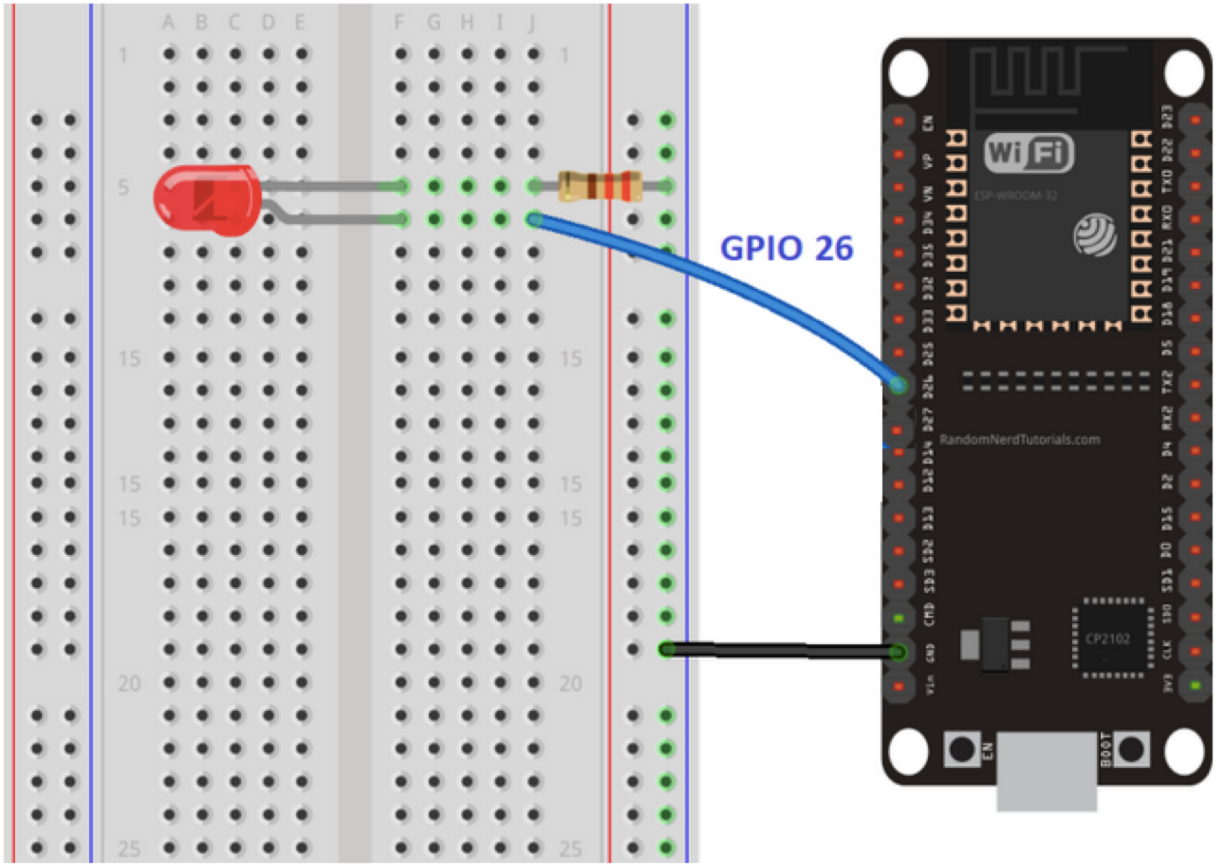
DevTitans

Servidor Web para acesso a um LED

Introdução

- Nesta aula criaremos um servidor web autônomo com um ESP32 que controla a saída de um LED usando o ambiente de programação Arduino IDE
- O servidor web poderá ser acessado com qualquer dispositivo que esteja com um navegador na rede local
- A ideia é mostrar como criar o servidor web e como o código funciona passo a passo

Circuito



Você pode ligar o LED em qualquer pino de saída que você queira

Codificação
:: Variávis globais

```
#include <WiFi.h>

const char* ssid = ”XXX";
const char* password = "XXX";
WiFiServer server(80);
String header;

bool led1Status = false;
const int LED1_Pin = 26;
```

Inclusão da biblioteca, detalhes da rede, geração do servidor e detalhes dos pinos

Codificação
:: setup (parte 1)

```
void setup() {
  Serial.begin(115200);
  pinMode(LED1_Pin, OUTPUT);
  digitalWrite(LED1_Pin, LOW);
```

Início do setup: definição da velocidade da comunicação serial e detalhes do pino de acesso ao LED

Codificação
:: setup (parte 2)

```
// Conexao com a rede Wi-Fi
Serial.print("Conectando a ");
Serial.println(ssid);
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
```

Conexão com a rede Wifi

Codificação
:: setup (parte 3)

```
// Imprime o endereco IP local
// e inicia o Servidor Web
  Serial.println("");
  Serial.println("WiFi conectado.");
  Serial.println("Endereco IP: ");
  Serial.println(WiFi.localIP());
  server.begin();
}
```

Após o estabelecimento da conexão com a rede Wifi, imprime o endereço IP do servidor (DHCP) e inicia o servidor

Codificação
:: loop (parte 1)

```
void loop(){
  WiFiClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
            . . .
```

A função **`server.available()`** é usada para verificar se há alguma nova conexão de cliente pendente no servidor. Ela retorna um objeto **`WiFiClient`** que representa a conexão com o cliente

Codificação
:: loop (parte 1)

```
void loop(){
  WiFiClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
            . . .
```

A função **`client.connected()`** é usada para verificar se a conexão com o cliente ainda está ativa. Ela retorna **`true`** se a conexão estiver aberta e **`false`** se a conexão estiver fechada

Codificação
:: loop (parte 1)

```
void loop(){
  WiFiClient client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
            . . .
```

A função **`client.available()`** é usada para verificar se há dados disponíveis para leitura do cliente. Ela retorna o número de bytes disponíveis para leitura a partir do buffer de recebimento do cliente

Codificação
:: loop (parte 2)

```
if (client.available()) {
  char c = client.read();
  header += c;
  if (c == '\n') {
    client.println("HTTP/1.1 200 OK");
    client.println("Content-type:text/html");
    client.println();
```

A função **`client.println()`** é usada para enviar dados para o cliente conectado. As primeiras linhas são cabeçalhos **HTTP**, seguidas por uma linha em branco para indicar o final dos cabeçalhos

Codificação
:: loop (parte 3)

```
// ajusta o GPIO on e off
if (header.indexOf("GET /LED1/on") >= 0) {
    Serial.println(”LED1 on");
    led1Status = true;
    digitalWrite(LED1_Pin, HIGH);
} else if (header.indexOf("GET /LED1/off") >= 0) {
    Serial.println(”LED1 off");
    led1Status = false;
    digitalWrite(LED1_Pin, LOW);
}
```

Foi usada a função String **`header.indexOf("GET /LED1/on")`**, mas poderia ter sido utilizada também a função **`header.endsWith("GET /LED1/on")`**

Codificação
:: loop (parte 4)

```
client.println("<html>");
client.println("<body><h1>ESP32 Web Server</h1>");
if (led1Status) {
  client.println("<p>LED1 - State ON</p>");
  client.println("<p><a href=\"/LED1/off\"><button
    class=\"button\">OFF</button></a></p>");
} else {
  client.println("<p>LED1 - State OFF</p>");
  client.println("<p><a href=\"/LED1/on\"><button
  class=\"button\">ON</button></a></p>");
}
client.println("</body></html>");
```

Foi gerado um botão para ligar/desligar o LED1. O rótulo do botão é alterado a cada caso. No final, há a finalização da página web enviada ao cliente

Codificação
:: loop (parte 5)

```
            header = "";
            // Fecha a conexao
            client.stop();
          } // if nova linha
        } // if cliente disponivel
      } // if cliente conectado
    } // if cliente se conectou
} // loop
```

Fechar a conexão com o cliente

Execução

:: Endereço IP do servidor

```
Conectando a ICOMP_1A
.....
WiFi conectado.
Endereco IP:
10.208.2.114
LED1 on
LED1 off
LED1 on
LED1 off
```

Execução

:: Acesso via Navegador