Projeto Arduino #webserver

1 – Criando o servidor web no Arduino. Os dados gerados são aleatórios.

Conectar o cartão ethernet no Arduino. Identificar o endereço IP que deverá ser utilizado e configurar o IP e a porta no programa do servidor. Conectar o Arduino na rede via cabo.
Obs.: Instalar a biblioteca: **ArduinoJson**, desenvolvida por **Benoit Blanchon**, versão 6.21.3.

Programa 1: Script do webserver

```
#include <Ethernet.h>
#include <ArduinoJson.h>

byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 0, 100);

// Cria um servidor web na porta 80
EthernetServer server(80);

void setup() {
 // Inicializa a comunicação serial
 Serial.begin(9600);

 // Inicializa a conexão Ethernet e o servidor
 Ethernet.begin(mac, ip);
 server.begin();

 Serial.println("Servidor Web Iniciado. Aguarde as requisições...");
}

void loop() {
 // Escutando novas conexões
 EthernetClient client = server.available();

 if (client) {
  Serial.println("Novo Cliente Conectado");

  // Aguardando a solicitação HTTP
  while (client.connected()) {
   if (client.available()) {
    char c = client.read();

    // Verifica se a solicitação HTTP é concluída
    if (c == '\n') {
     // Leitura dos dados das portas analógicas A0, A1 e A2
     int analogValueA0 = analogRead(A0);
     int analogValueA1 = analogRead(A1);
     int analogValueA2 = analogRead(A2);

     // Cria um objeto JSON
     StaticJsonDocument<200> jsonDocument;
     jsonDocument["A0"] = analogValueA0;
     jsonDocument["A1"] = analogValueA1;
     jsonDocument["A2"] = analogValueA2;
```

```
        // Converte o objeto JSON em uma string
        String jsonString;
        serializeJson(jsonDocument, jsonString);

        // Envia os dados como resposta POST no formato JSON com cabeçalhos CORS
        client.println("HTTP/1.1 200 OK");
        client.println("Access-Control-Allow-Origin: *");  // Permitir qualquer origem (em produção,
especifique origin(s) específico(s))
        client.println("Content-Type: application/json");
        client.println("Connection: close");
        client.println();
        client.println(jsonString);
        break;
      }
     }
    }

    // Fecha a conexão
    delay(1);
    client.stop();
    Serial.println("Cliente Desconectado");
  }
}
```

2 - Criando uma página HTML para consumir os dados do webserver Arduino. Salvar a página no Desktop da sua máquina, com o nome a sua escolha e a extensão deverá ser .html

Programa 2: Página HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Arduino Web Server</title>
</head>
<body>

<h1>Dados do Arduino</h1>

<div id="dadosArduino"></div>

<script>
// Função para fazer a requisição AJAX
function getDataFromArduino() {
   var xhttp = new XMLHttpRequest();
   xhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
         // Converte a resposta JSON em um objeto JavaScript
         var data = JSON.parse(this.responseText);

         // Exibe os dados na página
         document.getElementById("dadosArduino").innerHTML =
            "<p>Temperatura: " + data.A0 + "</p>" +
            "<p>Umidade: " + data.A1 + "</p>" +
            "<p>Luminosidade: " + data.A2 + "</p>";
      }
   };

   // Realiza uma requisição POST para o servidor Arduino
   xhttp.open("POST", "http://192.168.0.100/", true);
   xhttp.send();
}

// Atualiza os dados automaticamente a cada 5 segundos
setInterval(getDataFromArduino, 5000);

// Chama a função para obter os dados pela primeira vez
getDataFromArduino();
</script>

</body>
</html>
```

Desafio

Implementar o serviço WEB, disponibilizando os dados dos sensores de temperatura, umidade e luminosidade.