Arduino com Shield Ethernet

Sumário

Exercício 1: Explorando as Especificações do Arduino para Uso como Webserver

Objetivo:

- Analisar as especificações do Arduino UNO e Mega em relação à memória, velocidade de processamento, frequência do clock e o uso de SD para armazenamento de páginas, considerando suas influências na capacidade de atuar como um Webserver.

Parte 1: Memória, Processamento e Clock

Memória RAM:

- Pesquise e registre a quantidade de memória RAM disponível nos Arduinos UNO e Mega. • Discuta os possíveis impactos no desempenho do Webserver devido à disponibilidade de RAM.
- Velocidade de Processamento e Relação com o Clock:
- Identifique a taxa de instruções por segundo (IPS) ou outra medida equivalente para os Arduinos.
- Explique a importância da velocidade de processamento na capacidade do Arduino em lidar com solicitações de um Webserver.
- Pesquise e relate a relação entre a velocidade de processamento e a frequência do clock nos Arduinos, explicando como a variação na frequência do clock pode influenciar a velocidade de processamento e, consequentemente, a capacidade de resposta do Webserver.

Parte 2: Uso do SD para Armazenamento de Páginas

Capacidade de Armazenamento no SD:

- Verifique e anote a capacidade máxima de armazenamento de um cartão SD compatível com o Arduino.
- Discuta como o uso de um cartão SD pode impactar a disponibilidade de memória para operações do Webserver.

Eficiência e Limitações do Armazenamento em SD:

- Analise os benefícios e limitações de armazenar páginas web no cartão SD em comparação com a memória flash do Arduino.
- Explore como o acesso aos arquivos no cartão SD pode afetar o desempenho do Webserver.

Relatório e Discussão:

- Compile as informações coletadas sobre memória, processamento, clock e armazenamento em SD, elaborando um relatório abordando as limitações e considerações importantes ao utilizar o Arduino como Webserver.
- Promova uma discussão em sala de aula para explorar as implicações dessas descobertas na prática, considerando estratégias para lidar com as limitações de hardware ao desenvolver aplicações web utilizando Arduinos.

Este exercício visa oferecer uma compreensão ampla das especificações do Arduino UNO e Mega e suas influências ao atuar como Webserver, integrando aspectos de memória, processamento, clock e armazenamento em SD, e explorando estratégias para otimizar o desempenho dentro das limitações do hardware.

Exercício 2-1: Conexão Básica da Shield Ethernet

Objetivo:

- Conectar a Shield Ethernet ao Arduino e configurar uma conexão básica.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; // Endereço MAC do Shield
EthernetServer server(80); // Porta para o servidor

void setup() {
 Ethernet.begin(mac); // Inicialização da conexão Ethernet

 // Inicialização do servidor na porta 80
 server.begin();

 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available(); // Verifica a chegada de clientes

 if (client) {
 Serial.println("Novo cliente!"); // Mensagem ao detectar um cliente
client.stop(); // Encerra a conexão
 }
}
```

Exercício 2-2: Configuração do Webserver Simples sem HTML Objetivo:

- Configurar um Webserver que responde a solicitações básicas.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 char c = client.read(); // Lê o caracter da requisição
 Serial.write(c); // Imprime o caracter no monitor serial
 }
 }
 client.stop(); // Encerra a conexão
 }
}
```

Estes exercícios progridem de uma conexão básica da Shield Ethernet até um Webserver que responde a solicitações e controla saídas do Arduino. Cada exercício introduz uma camada adicional de complexidade para compreender a configuração e funcionamento de um Webserver com a utilização da Shield Ethernet no Arduino.

Exercício 3-1: Configuração Básica do Webserver com HTML Objetivo:

- Configurar um Webserver no Arduino para responder a requisições web básicas.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/html");
 client.println();
 client.println("<html><body><h1>Hello, World!</h1></body></html>");
client.stop();
 }
}
```

Exercício 3-2: Configuração Básica do Webserver com HTML e uma saída

Objetivo:

- Controlar uma saída (por exemplo, um LED) através de uma requisição web.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 pinMode(9, OUTPUT); // Configura o pino 9 como saída
 digitalWrite(9, LOW); // Inicialmente desliga o pino 9
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 char c = client.read();
 Serial.write(c);

 if (c == '1') {
 digitalWrite(9, HIGH); // Liga o pino 9
 } else if (c == '0') {
 digitalWrite(9, LOW); // Desliga o pino 9
 }
 }
 }
 client.stop();
 }
}
```

Exercício 3-3: Controle Avançado de Saídas

Objetivo:

- Controlar múltiplas saídas através de diferentes requisições web.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);
 digitalWrite(9, LOW);
 digitalWrite(10, LOW);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 String request = client.readStringUntil('\r');
 Serial.println(request);

 if (request.indexOf("/ligar1") != -1) {
 digitalWrite(9, HIGH);
 } else if (request.indexOf("/desligar1") != -1) {
 digitalWrite(9, LOW);
 } else if (request.indexOf("/ligar2") != -1) {
 digitalWrite(10, HIGH);
 } else if (request.indexOf("/desligar2") != -1) {
 digitalWrite(10, LOW);
 }
 }
 }
 client.stop();
 }
}
```

Estes exercícios fornecem desde a configuração básica de um Webserver até o controle avançado de múltiplas saídas por meio de requisições web, permitindo o desenvolvimento progressivo das habilidades de programação e interação do Arduino com requisições web.

Exercício 4-1: Configuração Básica de Endpoints

Objetivo:

- Criar endpoints simples para estabelecer a comunicação inicial com o Node-RED.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 char c = client.read();
 Serial.write(c);

 // Verifica se a requisição é para um endpoint específico
 if (c == 'A') {
 // Responde com uma mensagem específica do endpoint A
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/plain");
 client.println();
 client.println("Resposta do Endpoint A");
 } else if (c == 'B') {
 // Responde com uma mensagem específica do endpoint B
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/plain");
 client.println();
 client.println("Resposta do Endpoint B");
 }
 }
 }
 client.stop();
 }
}
```

Exercício 4-2: Aprimoramento de Endpoints e Parâmetros Objetivo:

- Criar endpoints que aceitam parâmetros para interagir com o Node-RED.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 String request = client.readStringUntil('\r');
 Serial.println(request);

 if (request.indexOf("/endpoint") != -1) {
 // Encontrou uma requisição para o endpoint
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/plain");
 client.println();

 // Analisa os parâmetros passados no endpoint
 int paramIndex = request.indexOf("param=");
 if (paramIndex != -1) {
 String parametro = request.substring(paramIndex + 6);
 client.println("Parâmetro recebido: " + parametro);
 } else {
 client.println("Nenhum parâmetro fornecido");
 }
 }
 }
 }
 client.stop();
 }
}
```

Exercício 4-3: Implementação de Endpoints Restful

Objetivo:

- Criar endpoints Restful para uma interação mais estruturada com o Node-RED.

Código de Exemplo:

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

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
EthernetServer server(80);

void setup() {
 Ethernet.begin(mac);
 server.begin();
 Serial.begin(9600);
 Serial.println("Servidor inicializado");
}

void loop() {
 EthernetClient client = server.available();

 if (client) {
 Serial.println("Novo cliente!");
 while (client.connected()) {
 if (client.available()) {
 String request = client.readStringUntil('\r');
 Serial.println(request);

 if (request.indexOf("GET /endpoint1") != -1) {
 // Responde ao endpoint1 com uma mensagem
 client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/plain");
 client.println();
 client.println("Resposta do Endpoint1");
 } else if (request.indexOf("POST /endpoint2") != -1) {
 // Processa dados enviados para o endpoint2
 // Aqui você poderia analisar dados enviados por POST e realizar ações
client.println("HTTP/1.1 200 OK");
 client.println("Content-Type: text/plain");
 client.println();
 client.println("Ação realizada no Endpoint2");
 }
 }
 }
 client.stop();
 }
}
```

Estes exercícios abordam a criação de endpoints simples até a implementação de endpoints mais estruturados para interação com o Node-RED, permitindo o desenvolvimento de habilidades progressivas na configuração e compreensão do funcionamento desses endpoints.

Exercício 5-1: Configuração Básica de Conexão no Node-RED Objetivo:

- Configurar um fluxo no Node-RED para se conectar ao Webserver do Arduino.

Fluxo no Node-RED:

- Use um nó HTTP Request no Node-RED.
- Configure o nó para fazer uma solicitação ao IP do Arduino e a porta onde o Webserver está escutando (normalmente 80).
- Configure o método como GET.
- Execute o fluxo e verifique se está recebendo as respostas do Arduino no console do Node RED.

```
<flows>
 <flow name="Configuração Básica de Conexão">
 <httpRequestNode url="http://IP_do_Arduino:80" method="GET"></httpRequestNode>
</flow>
</flows>
```

Exercício 5-2: Integração com Parâmetros no Node-RED

Objetivo:

- Configurar um fluxo no Node-RED que envie parâmetros para o Webserver do Arduino.

Fluxo no Node-RED:

- Use um nó HTTP Request no Node-RED.
- Configure o nó para fazer uma solicitação GET ou POST ao IP do Arduino e a porta do Webserver.
- Acrescente parâmetros na solicitação, se o endpoint do Arduino suportar parâmetros.
- Analise a resposta recebida no Node-RED e manipule-a conforme necessário.

```
<flows>
 <flow name="Integração com Parâmetros">
 <httpRequestNode url="http://IP_do_Arduino:80" method="POST">
 <parameters>
 <parameter name="parametro1" value="valor1"></parameter>
 <parameter name="parametro2" value="valor2"></parameter>
 </parameters>
 </httpRequestNode>
 </flow>
</flows>
```

Exercício 5-3: Uso de Endpoints Restful no Node-RED

Objetivo:

Configurar um fluxo no Node-RED que interaja com endpoints Restful do Arduino.

Fluxo no Node-RED:

- Utilize o nó HTTP Request no Node-RED.
- Configure a solicitação para diferentes métodos (GET, POST, PUT, DELETE) e endpoints específicos do Arduino.
- Implemente a manipulação de dados, como envio de dados no corpo da requisição para o Arduino.
- Realize ações com base nas respostas recebidas do Arduino.

```
<flows>
 <flow name="Uso de Endpoints Restful">
 <httpRequestNode url="http://IP_do_Arduino/endpoint1"
method="GET"></httpRequestNode>
 <httpRequestNode url="http://IP_do_Arduino/endpoint2" method="POST">
<parameters>
 <parameter name="data" value="example_data"></parameter>
 </parameters>
 </httpRequestNode>
 <httpRequestNode url="http://IP_do_Arduino/endpoint3"
method="PUT"></httpRequestNode>
 <httpRequestNode url="http://IP_do_Arduino/endpoint4"
method="DELETE"></httpRequestNode>
 </flow>
</flows>
```

Estes exercícios permitem que os alunos configurem e compreendam a integração entre os endpoints do Arduino e o Node-RED, avançando de uma configuração básica de conexão até a interação com endpoints Restful, promovendo um entendimento gradativo e prático da integração entre os dois sistemas.

Questionário sobre Exercícios com Arduino e Node-RED

Questão 1:

Qual biblioteca é comumente usada para estabelecer a conexão Ethernet no Arduino?

- a) Ethernet2.h
- b) WiFi.h
- c) Ethernet.h
- d) Internet.h

Questão 2:

Qual é a porta padrão em que um Webserver no Arduino geralmente escuta por requisições HTTP?

- a) 8080
- b) 80
- c) 8000
- d) 8888

Questão 3:

No Arduino, qual função é usada para definir um pino como saída?

- a) `pinMode()`

- b) `setOutput()`
- c) `outputMode()`
- d) `digitalWrite()`

Questão 4:

Em um endpoint Restful, qual método HTTP é geralmente usado para obter dados do servidor?

- a) POST
- b) DELETE
- c) PUT
- d) GET

Questão 5:

Qual é a diferença entre o método GET e o método POST em requisições HTTP? - a) GET é para enviar dados e POST é para receber dados.

- b) GET é para receber dados e POST é para enviar dados.
- c) Ambos são usados para a mesma finalidade.
- d) GET e POST são métodos obsoletos em requisições HTTP.

Questão 6:

Qual comando no Arduino é usado para ligar uma saída?

- a) `output(HIGH)`
- b) `set(HIGH)`
- c) `digitalWrite(LOW)`
- d) `digitalWrite(HIGH)`

Questão 7:

Para controlar uma saída do Arduino via Webserver, qual caracter é tipicamente enviado na requisição para ligar a saída?

- a) 'ON'
- b) '1'
- c) 'HIGH'
- d) 'TRUE'

Questão 8:

Para configurar um fluxo no Node-RED para se conectar ao Webserver do Arduino, qual nó é comumente usado?

- a) TCP Request
- b) HTTP Request
- c) Ethernet Connection
- d) Web Connect

Questão 9:

Qual é o objetivo de criar endpoints Restful?

- a) Simplificar o acesso a servidores web.
- b) Permitir a comunicação de dispositivos sem fio.
- c) Organizar e padronizar a interação com recursos de um servidor.
- d) Conectar-se a APIs externas.

Questão 10:

Em um fluxo no Node-RED, qual nó seria utilizado para processar dados recebidos de um Webserver do Arduino?

- a) HTTP Response
- b) Function
- c) Debug
- d) Template