Conexão Node Js com Scanner

Para conectar um scanner a Node.js, você precisa de um **driver de dispositivo** e de uma **biblioteca** que possibilite a comunicação com ele. O processo pode variar dependendo do tipo de scanner, do sistema operacional e da biblioteca que você escolhe.

Aqui está um guia geral, com exemplos para diferentes cenários:

1. Identificando o Tipo de Scanner:

- **Scanner USB:** A maioria dos scanners USB funciona com o protocolo HID (Human Interface Device).
- **Scanner de Rede:** Se o scanner possui rede (Ethernet, Wi-Fi), você pode usar protocolos como TCP/IP ou HTTP para comunicação.
- **Scanner Bluetooth:** Para scanners Bluetooth, você precisa usar a tecnologia Bluetooth.

2. Escolhendo a Biblioteca Node.js Adequada:

- **HID:**
  - **node-hid:** Esta biblioteca é a mais comum para lidar com dispositivos HID em Node.js. Veja a documentação em: https://www.npmjs.com/package/node-hid
- **TCP/IP e HTTP:**
  - **net:** A biblioteca net do Node.js fornece ferramentas para comunicação TCP/IP.
  - **http:** Para comunicação HTTP, use a biblioteca http do Node.js.
- **Bluetooth:**
  - **noble:** Esta biblioteca é um módulo Bluetooth Low Energy (BLE) para Node.js. Você pode encontrá-la em: https://www.npmjs.com/package/noble
  - **serialport:** Esta biblioteca permite comunicação serial, que pode ser utilizada para alguns scanners Bluetooth. Encontre-a em: https://www.npmjs.com/package/serialport

3. Escrevendo o Código Node.js:

Exemplo com Scanner USB (HID):

```
const hid = require('node-hid');

// Encontra o dispositivo HID do scanner (substitua por seu ID de
produto/fabricante)


const devices = hid.devices();
const scannerDevice = devices.find(device =>
  device.productId === 0xXXXX && device.vendorId === 0xYYYY
);

if (!scannerDevice) {
  console.error('Dispositivo de scanner não encontrado.');
  process.exit(1);
}

// Abre uma conexão com o scanner
const scanner = new hid.HID(scannerDevice.path);

// Define um listener para dados do scanner
```

```
scanner.on('data', (data) => {
  // Processa os dados recebidos do scanner
  console.log('Dados do Scanner:', data.toString());
});

// Fecha a conexão com o scanner quando o processo termina
process.on('exit', () => {
  scanner.close();
});
```

Exemplo com Scanner de Rede (TCP/IP):

```
const net = require('net');

// Define o endereço IP e a porta do scanner
const scannerAddress = '192.168.1.100';
const scannerPort = 5000;

// Cria um socket TCP
const socket = new net.Socket();

// Conecta ao scanner
socket.connect(scannerPort, scannerAddress, () => {
  console.log('Conectado ao scanner.');
});

// Define um listener para dados do scanner
socket.on('data', (data) => {
  // Processa os dados recebidos do scanner
  console.log('Dados do Scanner:', data.toString());
});

// Fecha a conexão com o scanner quando o processo termina
socket.on('close', () => {
  console.log('Conexão com o scanner fechada.');
});

socket.on('error', (err) => {
  console.error('Erro na conexão com o scanner:', err);
});
```

Observações:

- **Identificação do Dispositivo:** Você precisa encontrar o ID de produto/fabricante do seu scanner para identificá-lo corretamente. Consulte a documentação do scanner ou use ferramentas de reconhecimento de dispositivo.
- **Formatação de Dados:** O formato dos dados recebidos do scanner varia. Consulte a documentação do fabricante para entender a estrutura dos dados.
- **Configuração do Scanner:** Certifique-se de que o scanner está configurado para enviar dados em um formato compatível com o código Node.js.
- **Documentação:** Consulte a documentação do fabricante do scanner para obter informações detalhadas sobre a comunicação com o dispositivo.

Lembre-se de que este é apenas um guia geral. As etapas exatas para conectar seu scanner Node.js variam dependendo do seu scanner específico e da biblioteca que você escolheu.

- **Scanner Bluetooth ao Node.js**

Para conectar um scanner Bluetooth ao Node.js, você precisará usar uma biblioteca como a noble (para scanners Bluetooth Low Energy - BLE) ou a serialport (para scanners Bluetooth que usam comunicação serial).

Aqui estão exemplos de como fazer isso usando ambas as bibliotecas:

1. Usando a Biblioteca noble (para scanners BLE):

```
      const noble = require('noble');

// Define o UUID do serviço do scanner
const scannerServiceUUID = 'XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX'; // Substitua
pelo UUID do seu scanner

noble.on('stateChange', (state) => {
  if (state === 'poweredOn') {
    console.log('Bluetooth ativado.');

    // Começa a procurar dispositivos BLE
    noble.startScanning();
  } else {
    console.log('Bluetooth desativado.');
    noble.stopScanning();
  }
});

noble.on('discover', (peripheral) => {
  // Verifica se o dispositivo encontrado é o scanner
  if (peripheral.advertisement.serviceUuids.includes(scannerServiceUUID)) {
    console.log('Scanner Bluetooth encontrado:',
peripheral.advertisement.localName);

    // Conecta ao scanner
    peripheral.connect((error) => {
      if (error) {
        console.error('Erro ao conectar ao scanner:', error);
        return;
      }

      console.log('Conectado ao scanner.');

      // Busca o serviço do scanner
      peripheral.discoverServices([scannerServiceUUID], (error, services) => {
        if (error) {
          console.error('Erro ao descobrir serviços:', error);
          return;
        }

        const scannerService = services[0];

        // Busca as características do serviço
        scannerService.discoverCharacteristics([], (error, characteristics) => {
          if (error) {
```

```
            console.error('Erro ao descobrir características:', error);
            return;
          }

          // Define um listener para dados da característica relevante
          characteristics.forEach((characteristic) => {
            characteristic.on('data', (data) => {
              console.log('Dados do Scanner:', data.toString());
            });
          });

          // Subscreva para dados da característica
          characteristics.forEach((characteristic) => {
            characteristic.subscribe((error) => {
              if (error) {
                console.error('Erro ao se inscrever na característica:', error);
                return;
              }

              console.log('Inscrito na característica.');
            });
          });
        });
      });
    });
  }
});
```

content_copyUse code with caution.JavaScript

2. Usando a Biblioteca serialport (para scanners que usam comunicação serial):

```
      const SerialPort = require('serialport');

// Define a porta serial do scanner
const scannerPort = '/dev/ttyACM0'; // Substitua pela porta do seu scanner

// Cria um objeto SerialPort
const port = new SerialPort(scannerPort, { baudRate: 115200 });

// Define um listener para dados do scanner
port.on('data', (data) => {
  console.log('Dados do Scanner:', data.toString());
});

// Fecha a conexão com o scanner quando o processo termina
process.on('exit', () => {
  port.close();
});
```

content_copyUse code with caution.JavaScript

Observações importantes:

- **UUID do Serviço:** Você precisará descobrir o UUID do serviço do seu scanner BLE. Consulte a documentação do fabricante do scanner ou use ferramentas de varredura Bluetooth para descobrir o UUID.

- **Porta Serial:** Se o seu scanner Bluetooth usa comunicação serial, você precisará encontrar a porta serial correspondente no seu sistema. Use ferramentas como ls /dev/tty* no Linux para listar as portas seriais disponíveis.
- **Configuração do Scanner:** Certifique-se de que o seu scanner Bluetooth está configurado para enviar dados em um formato compatível com o código Node.js.
- **Documentação:** Consulte a documentação do fabricante do scanner para obter informações detalhadas sobre a comunicação Bluetooth e os formatos de dados.

Lembre-se de que esses exemplos são apenas para demonstração. Você pode precisar fazer ajustes no código de acordo com as características específicas do seu scanner e a biblioteca que você escolheu.