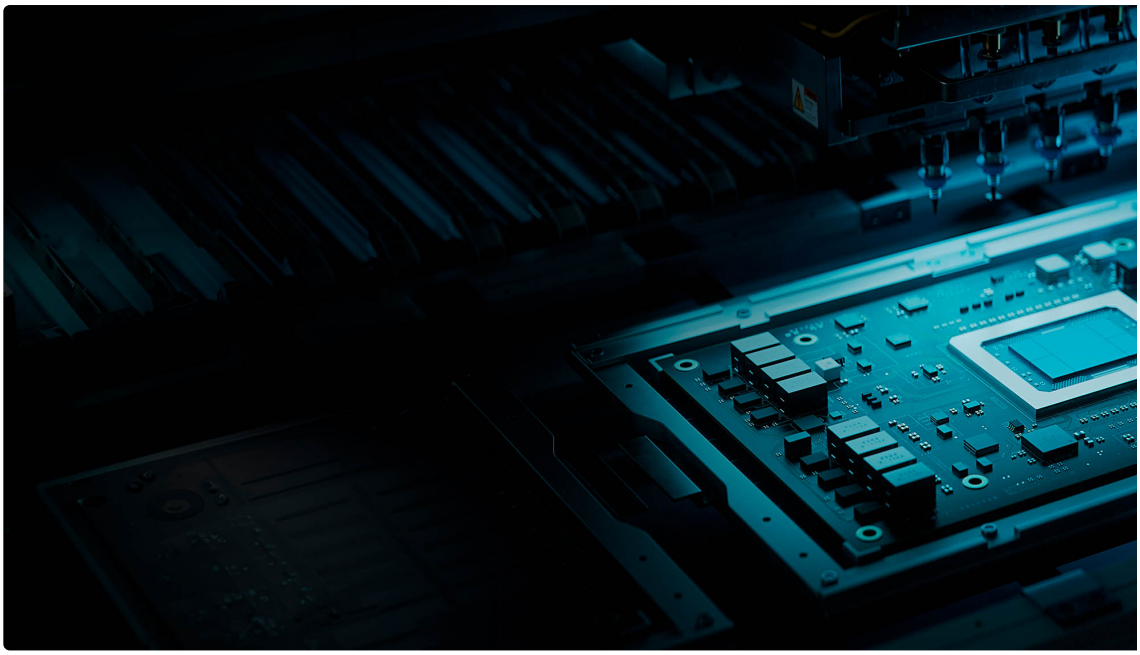


Portas de hardware e comunicação em C

Prof. Marcos Santana Farias

Apresentação

Este conteúdo engloba a programação de comunicação via porta serial RS232, o desenvolvimento de sistemas para controle de dispositivos e a aquisição de dados usando sensores e atuadores. Além disso, vamos analisar a criação de conexões de rede utilizando sockets TCP e UDP em C e aplicações cliente/servidor usando os protocolos HTTP e MQTT, preparando você para integrar e controlar tecnologias em indústrias e projetos de IoT.

Propósito

Preparação

Instale o DevC++ na versão 5.11 e a IDE Arduino. Ter uma placa ESP32 é uma vantagem para testar os programas que acessam redes wi-fi. Sugerimos a versão mais comum: DOIT ESP32 DevKit V1. Essa preparação permite a programação efetiva do ESP32 para o desenvolvimento de projetos envolvendo controle de dispositivos e comunicação.

Objetivos

Módulo 1

Porta serial e portas de microcontroladores

Desenvolver programas para comunicação via porta serial RS232 integrados a apresentações visuais com o uso de gráficos.

Módulo 2

Controle de dispositivos e aquisição de dados com microcontroladores

Estruturar aplicações e sistemas para controle de dispositivos e aquisição de dados, utilizando práticas de programação voltadas para o uso de sensores e atuadores.

Módulo 3

Sockets e cliente/servidor TCP e UDP

Identificar os conceitos e as funções de sockets na criação de conexões em rede TCP e UDP em programas com linguagem C.

Módulo 4

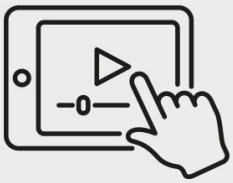
Cliente/servidor HTTP e cliente MQTT para IoT

Desenvolver programas cliente/servidor HTPP e com protocolo MQTT para o envio de dados entre dispositivos pela internet.

Introdução

Neste vídeo, vamos explorar a importância da comunicação serial, o controle de dispositivos por meio de sensores e atuadores, e o papel fundamental dos sockets e protocolos como HTTP e MQTT na conectividade de redes. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

1 - Porta serial e portas de microcontroladores

Ao final deste módulo, você será capaz de desenvolver programas para comunicação via porta serial RS232 integrados a apresentações visuais com o uso de gráficos.

Acesso à porta serial

Na era digital atual, a interface RS232, embora antiga, continua sendo muito importante, especialmente em ambientes industriais e para interagir com sistemas legados. Criada no início da computação, a RS232 era essencial para conectar dispositivos periféricos. Apesar da ascensão de interfaces como o USB, a RS232 se mantém relevante por sua robustez e confiabilidade, enriquecendo o aprendizado sobre comunicação serial e capacitando profissionais a gerir e a integrar tecnologias de diferentes eras.

Explicaremos neste vídeo por que a interface RS232 ainda é importante para a comunicação serial, mesmo com tecnologias modernas como o USB. Vamos destacar seu papel na automação industrial e na manutenção de sistemas legados. Assista!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

No início da computação pessoal, a porta serial RS232 era fundamental para a comunicação de dispositivos periféricos. Com o advento de interfaces mais modernas, como o USB, seu uso em computadores

contemporâneos diminuiu, mas ela continua relevante em contextos profissionais, especialmente na interação com microcontroladores e sistemas legados.

Porta serial RS232.

O RS232, um protocolo de comunicação serial, permitia a transferência sequencial de dados entre dispositivos, como modems e mouses. Embora menos prevalente em novos dispositivos, sua utilidade em automação industrial, eletrônica e robótica, além de sua capacidade de comunicar com equipamentos específicos, mantém sua importância profissional.

Conector DB-9 e conversores USB-Serial

O conector DB-9 de design trapezoidal com nove pinos é bastante utilizado na interface RS232. Veja os pinos principais.

TXD

Transmit data: envia dados sequenciais de um dispositivo a outro.

RXD

Receive data: recebe dados, permitindo comunicação bidirecional.

SG

Signal ground: serve como terra de referência de sinal.

Com a prevalência de portas USB e a escassez de RS232 em dispositivos modernos, os conversores USB-Serial tornaram-se

essenciais, facilitando a comunicação entre dispositivos USB e equipamentos que usam RS232.

Eles são importantes para a integração de equipamentos legados e para o desenvolvimento e a depuração de hardware, como microcontroladores, permitindo programação e diagnósticos por meio de conexões USB.

Relação entre RS232 e UART

Muitas vezes confundidos, o RS232 e o UART (universal asynchronous receiver/transmitter) são complementares na comunicação serial de dados. Vejamos!

UART

É um componente de hardware em microcontroladores e computadores, converte dados entre formas serial e paralela. Na transmissão, transforma dados paralelos em sequência serial; na recepção, faz o inverso.

RS232

É o protocolo que define como esses dados são transmitidos, incluindo os níveis de voltagem para sinais de transmissão e recepção. Assim, o UART prepara e processa os dados, enquanto o RS232 gerencia a transmissão efetiva entre dispositivos.

Vantagens continuadas do RS232

O RS232 é valorizado em ambientes industriais pela confiabilidade frente a condições adversas, como alta temperatura e interferência eletromagnética, além da simplicidade e do baixo custo de manutenção. Sua capacidade de comunicar com equipamentos antigos sem necessidade de conversores ou redesenho também é uma vantagem significativa.

Atividade 1

Qual das seguintes opções melhor descreve uma vantagem significativa do padrão RS232 em ambientes industriais?

A Alta taxa de transferência de dados em comparação a tecnologias mais modernas, como USB e Ethernet.

B Requerimento de componentes de hardware de alta tecnologia para garantir a comunicação eficiente.

C Capacidade de operar de modo eficaz em condições adversas, como alta temperatura e interferência eletromagnética.

D Necessidade de atualizações frequentes de software para manter a compatibilidade com novos dispositivos.

E Utilização de técnicas de criptografia avançada para proteger os dados transmitidos.

Parabéns! A alternativa C está correta.

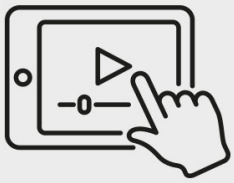
O RS232 é conhecido por sua confiabilidade em ambientes industriais com condições adversas, como altas temperaturas, vibrações e interferência eletromagnética. Sua robustez o torna valioso em situações em que outras tecnologias podem falhar ou necessitar de proteções adicionais. O RS232 não se destaca por alta taxa de transferência, não requer hardware avançado, atualizações constantes de software nem possui criptografia avançada.

Acesso às portas de microcontroladores

Os microcontroladores compactam vastas funcionalidades de um computador em um único chip, integrando CPU, memória e circuitos de comunicação em aplicações que vão de eletrodomésticos inteligentes a sistemas industriais. O conhecimento prático de seu funcionamento e interação com o ambiente externo é essencial para engenheiros e desenvolvedores em TI, permitindo a criação de soluções eficientes e inovadoras em sistemas embarcados.

Vamos detalhar neste vídeo as portas de entrada e saída e como elas permitem interações com o mundo externo, além de discutir o papel dos barramentos I2C e SPI na comunicação entre dispositivos. Não deixe de conferir!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

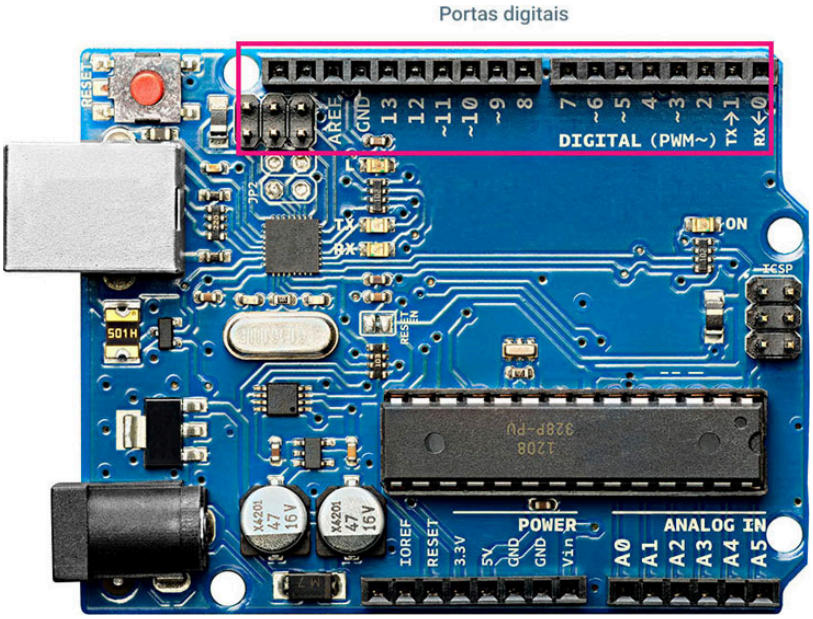


Portas de entrada e saída (E/S)

As portas de E/S nos microcontroladores são usadas para interagir com o mundo externo. Elas coletam dados de sensores e controlam dispositivos, por exemplo, motores e LEDs. Com pinos configuráveis, podem alternar entre entrada e saída conforme necessário. Existem dois tipos principais: digitais e analógicas, cada uma com diferentes usos.

Portas digitais

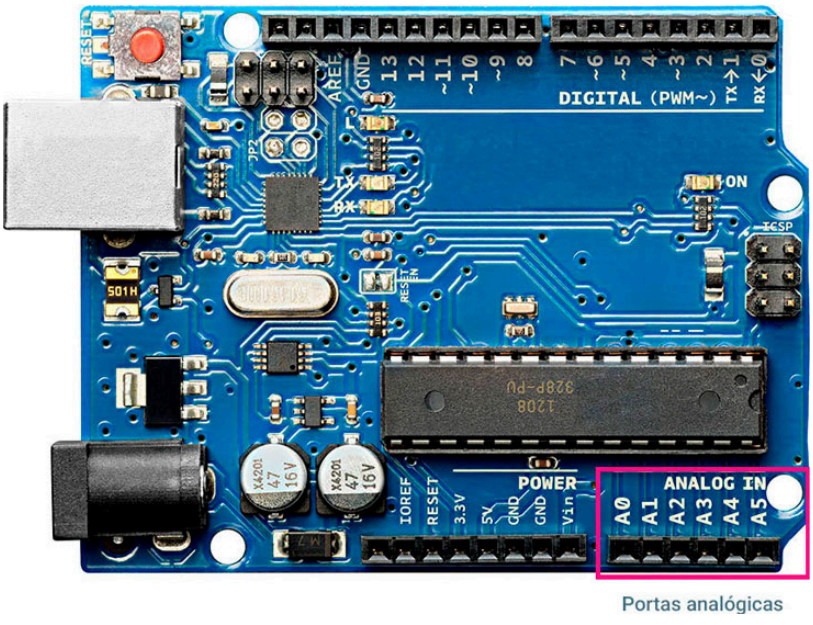
Podem ler e escrever apenas dois estados: alto (High) e baixo (Low). Isso as torna ideais para controlar dispositivos simples como LEDs, relés e interruptores. No Arduino UNO, por exemplo, existem 14 pinos digitais que podem ser configurados individualmente como entrada ou saída, usando funções simples do ambiente integrado de programação (IDE) Arduino, como pinMode(), digitalWrite(), e digitalRead().



Portas digitais destacadas em microcontrolador.

Portas analógicas

São capazes de ler uma variedade de valores, o que é útil para lidar com dados mais complexos de sensores, como temperatura, luz ou pressão. No Arduino UNO, existem seis pinos analógicos que podem ler valores de 0 a 1023, representando uma voltagem de 0 a 5V. No ESP32, essa capacidade é ainda mais avançada, com suporte para resolução maior e mais pinos analógicos.



Portas analógicas destacadas em microcontrolador.

Portas seriais TX e RX e barramentos I2C e SPI

As portas seriais TX (transmissão) e RX (recepção) permitem a comunicação entre microcontroladores e outros dispositivos. No Arduino UNO e ESP32, são usadas para essa finalidade. Os barramentos I2C e SPI conectam dispositivos de hardware. O I2C é adequado para

múltiplos dispositivos de baixa velocidade, enquanto o SPI é mais rápido e usa quatro fios:

- MISO
- MOSI
- SCK
- SS

Cada dispositivo no SPI requer um pino SS separado, o que pode complicar o design com muitos dispositivos conectados.

Atividade 2

Considerando o uso de portas digitais em um Arduino UNO para um sistema de irrigação automática, qual das seguintes afirmações é verdadeira a respeito da função dos pinos digitais?

A Os pinos digitais são usados para medir diretamente o nível de umidade do solo.

B Os pinos digitais podem ser configurados para emitir sinais analógicos para controlar a intensidade do fluxo de água.

C Os pinos digitais são responsáveis por gerar sinais de frequência variável para controlar os solenoides diretamente.

D Cada pino digital no Arduino UNO deve ser configurado exclusivamente como saída para que o sistema de irrigação funcione corretamente.

E Os pinos digitais do Arduino UNO são usados para controlar os relés que ativam os solenoides, baseando-

se em sinais digitais de um sensor de umidade.

Parabéns! A alternativa E está correta.

Em um sistema de irrigação automática com Arduino UNO, os pinos digitais controlam relés para ativar solenoides, baseados nos sinais do sensor de umidade do solo. Os pinos digitais não medem umidade, mas recebem sinais de um sensor específico. Eles não emitem sinais analógicos.

Demonstração prática de acesso às portas de hardware

Vamos explorar o envio de dados do ESP32 para um computador usando comunicação serial via RS232. O ESP32 é uma plataforma versátil para IoT, e vamos utilizar C para programação devido à eficiência e ao controle.

Iniciaremos configurando o ESP32 no IDE do Arduino para comunicação serial, incluindo os pinos e a taxa de transmissão (baud rate).

Desenvolveremos funções em C para iniciar a comunicação e manipular dados recebidos, permitindo ler e interpretar informações transmitidas pelo ESP32. Isso facilita a integração com outras aplicações ou para monitoramento e teste.

Ao final, você saberá configurar e usar a comunicação serial do ESP32 com um computador.

Neste vídeo, mostraremos como configurar o ESP32 para enviar dados via comunicação serial à porta RS232 de um computador usando a IDE do Arduino. Assista!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

Esse resumo indica o processo de envio de dados do ESP32 para um computador usando a porta serial RS232, com programação no Arduino IDE e funções em C.

Parte 1: configuração do ESP32. Vamos lá!

- Instalação da IDE do Arduino: baixe e instale a IDE, se necessário.
- Adição de suporte ao ESP32: siga as instruções específicas para adicionar o gerenciador de placas ESP32 na IDE.
- Configuração da placa: selecione o modelo de ESP32 em Ferramentas > Placa.
- Programação: escreva código que envie mensagens serialmente a cada segundo, usando Serial.begin() para iniciar e Serial.println() para enviar os dados.

Parte 2: preparação do computador:

- Configuração do ambiente de desenvolvimento C: instale um compilador C, como o MinGW para Windows. Recomenda-se o DevC++ versão 5.11.
- Programação em C: use CreateFile() para abrir uma conexão serial. Configure com SetCommState() e SetCommTimeouts() para ajustar baud rate e timeouts. Use ReadFile() para ler os dados.

Parte 3: teste e depuração:

- Carregamento do código no ESP32: compile e carregue pela IDE do Arduino.
- Execução do programa C: compile e execute no computador. Verifique as configurações de porta COM e baud rate.
- Monitoramento: observe a saída do programa C para confirmar a recepção dos dados.
- Depuração: em caso de problemas, revise conexões, configurações da porta serial e lógica dos programas.

A seguir, confira o código em C para leitura na porta serial, que deve ser compilado no DevC++.

c

Agora, veja o código para escrita na porta serial pelo ESP32, que deve ser compilado e baixado para o ESP32 usando a IDE do Arduino.

c

Atividade 3

Você precisa modificar um programa para o ESP32 de modo que ele crie e envie um número aleatório entre 20 e 40 pela porta serial. Teste o programa enviando dados, com base na prática demonstrada. Qual das seguintes alterações no código original você deveria fazer?

C

A Substitua Serial.println("Olá Mundo"); por Serial.println(random(20, 41));

B Substitua Serial.println("Olá Mundo"); por Serial.println(rand() % 21 + 20);

C Substitua Serial.println("Olá Mundo"); por Serial.println(random(20, 40));

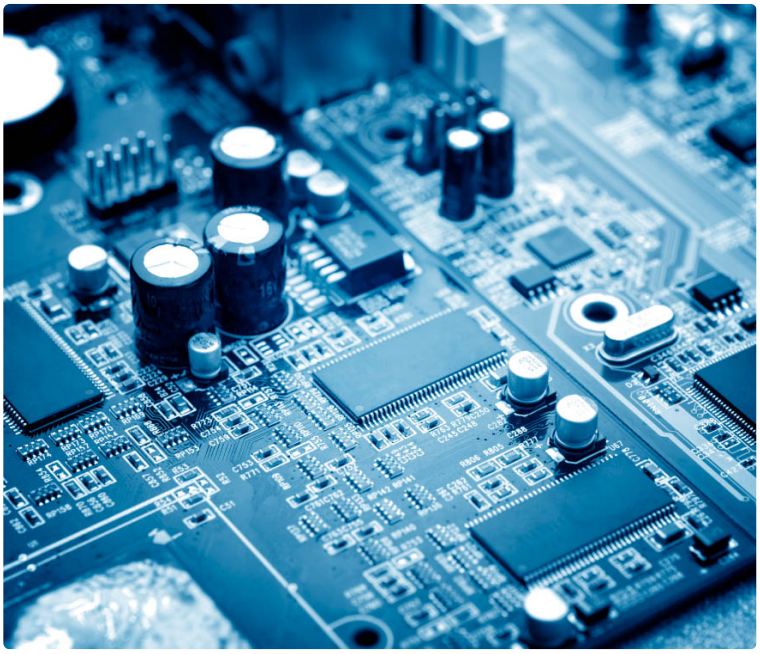
D Substitua Serial.println("Olá Mundo"); por Serial.print("Número: " + random(20, 41));

E

Substitua Serial.println("Olá Mundo"); por Serial.print(rand() % 20 + 20);

Parabéns! A alternativa A está correta.

A função random(min, max) no Arduino gera um número aleatório no intervalo [min, max-1]. Assim, para incluir o 40, usa-se random(20, 41). A função rand() não é nativa no Arduino, e a semente aleatória não foi mencionada. Random(20, 40) gera um número de 20 a 39, excluindo 40. Serial.print não inclui quebra de linha, e "+" para concatenar é inapropriado em C++ sem conversão adequada de tipo. rand() não é nativa, e a expressão não gera números até 40.



2 - Controle de dispositivos e aquisição de dados com microcontroladores

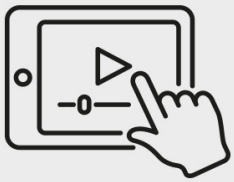
Ao final deste módulo, você será capaz de estruturar aplicações e sistemas para controle de dispositivos e aquisição de dados, utilizando práticas de programação voltadas para o uso de sensores e atuadores.

Controle de dispositivos com microcontroladores

Microcontroladores manipulam dados de sensores e atuam em dispositivos atuadores via técnicas como modulação por largura de pulso (PWM). Entender o funcionamento e a aplicação desses controladores é essencial para projetos práticos e teóricos em eletrônica e automação.

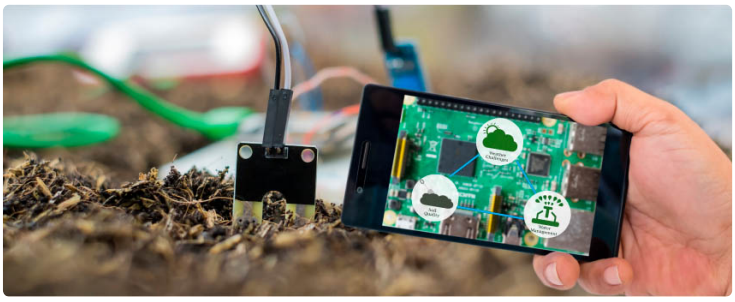
Vamos compreender neste vídeo o papel dos microcontroladores, como o ESP32 e o Arduino, no controle de dispositivos em variados contextos, desde projetos pessoais até aplicações industriais. Acompanhe!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Formas usuais de controle com microcontroladores

Conheça algumas das possibilidades de controle com uso desses dispositivos.



Representação de sensores verificando a umidade do solo.

Controle de sensores e atuadores

Um Arduino pode ler a umidade do solo com um sensor e ativar uma bomba de água automaticamente.

Representação de automação residencial.

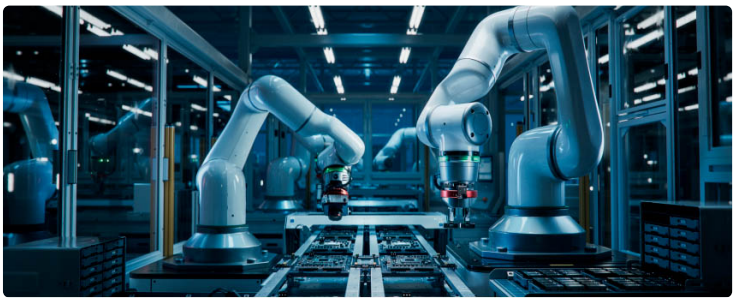
Automação residencial

O ESP32 com Wi-Fi integrado pode monitorar e controlar dispositivos domésticos remotamente via internet, permitindo ao usuário gerenciar luzes, sistemas de segurança e outros com um smartphone.

Modem.

Comunicação entre dispositivos

Microcontroladores facilitam a comunicação entre dispositivos usando tecnologias como bluetooth e Wi-Fi. ESP32s podem coletar dados de sensores em uma fábrica e enviar para um servidor central.



Robôs em esteira de produção.

Robótica

Os microcontroladores são fundamentais para controlar movimentos e ações de robôs, processando informações de múltiplos sensores e controlando motores.

Homem controlando máquina com tablet.

Interfaces homem-máquina (IHM)

Os microcontroladores são utilizados em interfaces de usuário, como painéis de controle e displays LCD ou touchscreens. Por

meio de um Arduino, é possível exibir informações e receber comandos, proporcionando interação entre o usuário e o sistema.

Controle com PWM

A modulação por largura de pulso (PWM) é uma técnica utilizada para controlar a energia fornecida a dispositivos eletrônicos.

> O PWM ajusta o ciclo de trabalho de uma onda quadrada, que representa a proporção do tempo que o sinal fica ativo (alto) em relação ao ciclo total (alto e baixo). Alterando esse ciclo de trabalho, regula-se a energia média enviada ao dispositivo.

Quando o sinal está ativo por mais tempo, mais energia é entregue; se por menos tempo, menos energia é fornecida. Essa técnica otimiza o uso de energia em várias aplicações eletrônicas.

Componentes do PWM

Entenda os componentes do PWM a seguir.

Frequência

Quantidade de ciclos por segundo. Afeta a suavidade com que o dispositivo controlado opera.

Ciclo de trabalho

Percentual do tempo em que o sinal está em nível alto durante um ciclo.

Veja agora aplicações de controle por PWM.

Controle de motores

O PWM pode regular a velocidade de motores DC, ajustando o ciclo de trabalho do sinal. Essencial em robótica, veículos automatizados e sistemas industriais, permite variar velocidades conforme a necessidade operacional.

Iluminação LED

O PWM ajusta a intensidade dos LEDs também alterando o ciclo de trabalho, controlando o brilho sem mudar a tensão. Amplamente utilizado em painéis de instrumentos, decoração e iluminação automotiva.

Controle de temperatura

O PWM regula a energia em elementos de aquecimento, como em ferros de solda e fornos elétricos, mantendo uma temperatura constante. Essencial para precisão térmica em aplicações industriais e comerciais.

Implementação com microcontroladores

IDEs para microcontroladores têm funções para gerar sinais PWM. No Arduino IDE, a função analogWrite(pin, value) gera um sinal PWM em um pino, com value controlando o ciclo de trabalho. Isso simplifica o desenvolvimento de sistemas com controle PWM.

Atividade 1

A técnica de modulação por largura de pulso (PWM) é aplicada em várias configurações eletrônicas, incluindo o controle de intensidade luminosa em LEDs, a velocidade de motores, entre outros.

Considere as seguintes afirmações:

Asserção (A)

Ao aumentar o ciclo de trabalho de um sinal PWM, a quantidade média de energia entregue a um dispositivo aumenta.

Razão (R)

O ciclo de trabalho refere-se à proporção do tempo em que o sinal está em estado alto durante um ciclo completo.

Qual das opções seguintes melhor descreve a relação entre a asserção e a razão?

A Tanto a asserção quanto a razão são verdadeiras, e a razão é uma explicação correta da asserção.

B A asserção e a razão são verdadeiras, mas a razão não é uma explicação correta da asserção.

C A asserção é verdadeira, mas a razão é falsa.

D A asserção é falsa, mas a razão é verdadeira.

E Tanto a asserção quanto a razão são falsas.

Parabéns! A alternativa A está correta.

Aumentar o ciclo de trabalho de um sinal PWM aumenta a energia média entregue ao dispositivo, pois o sinal fica mais tempo em estado alto. A razão, definindo corretamente o ciclo de trabalho, explica esse fenômeno de modo adequado.

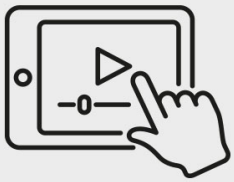
Aquisição de dados com microcontroladores

A aquisição de dados é imprescindível para setores industriais, científicos e cotidianos, envolvendo coleta e análise de informações para decisões e otimização de processos. Microcontroladores são muito importantes nesses processos, atuando como pontes entre dados físicos e informações digitais.

Eles integram diversos sensores, executam protocolos complexos, como o I2C, e realizam conversões analógico-digitais precisas, apoiando o desenvolvimento de soluções inovadoras e eficientes.

Vamos explorar neste vídeo a importância da aquisição de dados em variados setores, enfatizando o papel dos microcontroladores. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Papel dos microcontroladores na aquisição de dados

Para lidar com sinais que variam continuamente, como temperatura ou luz, os microcontroladores utilizam seus conversores analógico-digitais (ADCs) integrados.

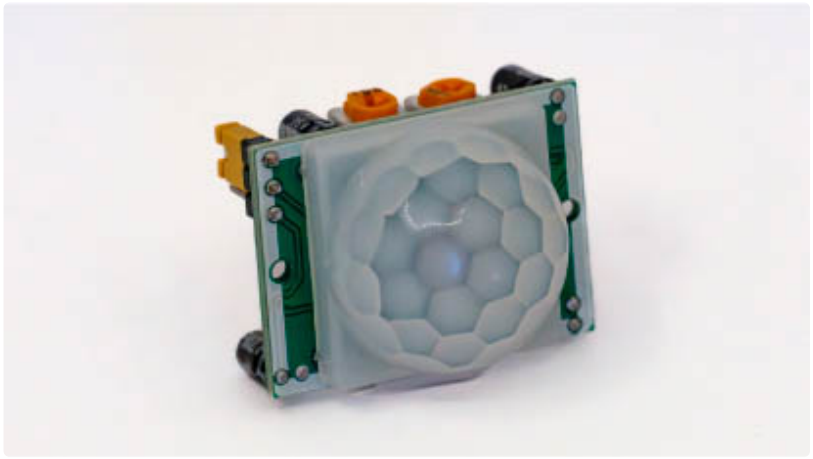
> **Os ADCs transformam sinais analógicos em representações digitais, permitindo ao microcontrolador processar e responder de maneira eficaz e precisa às variações do ambiente físico.**

Essas capacidades fazem do microcontrolador uma peça central em sistemas que requerem monitoramento contínuo, controle e integração de dados de múltiplas fontes.

Exemplo de sensor que opera com estados ON e OFF

O sensor PIR detecta movimento ao captar radiação infravermelha de corpos em sua área de alcance. Sensível a variações de temperatura, muda seu estado de saída quando detecta uma mudança na radiação

entre zonas, indicando movimento, por exemplo, uma pessoa em seu campo de visão.



Sensor de movimento PIR.

Ligação e programação de um sensor PIR no Arduino

Acompanhe o processo!

- VCC do sensor PIR deve ser conectado ao 5V do Arduino.
- GND do sensor PIR deve ser conectado ao GND do Arduino.
- Saída do sensor PIR deve ser conectada a uma porta digital do Arduino (por exemplo, D2).

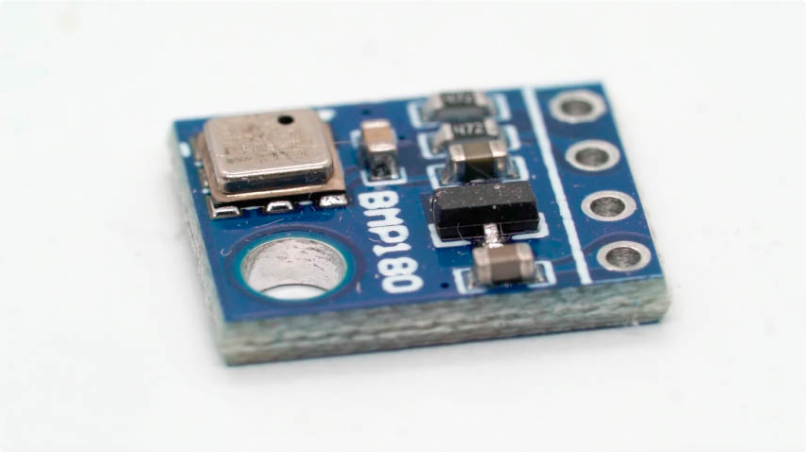
Código simples para ler o estado do sensor PIR

O código exemplo: quando o sensor detecta movimento, o LED na placa do Arduino pisca.

c

Exemplo de sensor que opera com protocolo I2C

O sensor BMP180 opera com protocolo I2C e é um dispositivo que mede a pressão barométrica e a temperatura ambiente. Essas medidas podem ser usadas para calcular a altitude, o que torna esse sensor bastante útil em aplicações como sistemas de navegação para drones, estações meteorológicas, dispositivos de fitness e muito mais.



Sensor BMP180.

Ligação de um sensor BMP180 a um ESP32 e programação

Acompanhe o processo!

- VCC do BMP180 deve ser conectado ao 3.3V do ESP32.
- GND do BMP180 deve ser conectado ao GND do ESP32.
- SCL do BMP180 deve ser conectado ao pino SCL do ESP32 (em geral, é o GPIO 22).
- SDA do BMP180 deve ser conectado ao pino SDA do ESP32 (em geral, é o GPIO 21).

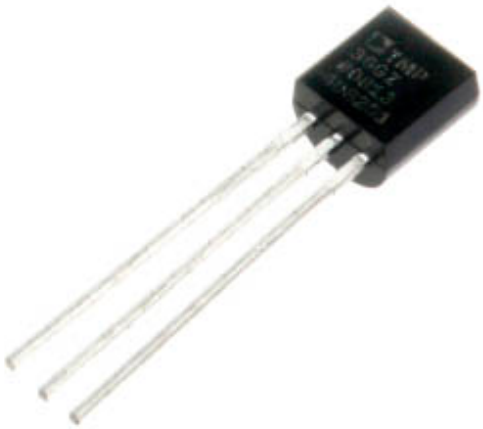
Código para o ESP32 usando o BMP180

Para programar o ESP32 para ler dados do BMP180, você pode usar a biblioteca Adafruit_BMP085_Unified, disponível no Arduino IDE. Veja um exemplo de código!

c

Exemplo de sensor que opera com ADC

O sensor TMP36 é um sensor de temperatura de baixo custo e fácil utilização, que opera com saída em tensão e não requer calibração adicional.



Sensor TMP36.

A saída de tensão aumenta linearmente, com um aumento de 10 mV a cada grau Celsius. Por exemplo, uma saída de 750 mV corresponde a uma temperatura de 25°C. Uma vantagem do TMP36 é que não necessita de tensão de entrada negativa para medir temperaturas abaixo de zero, ao contrário de outros sensores similares.

Ligação de um sensor TMP36 a um Arduino e programação

Acompanhe o processo!

- VCC do TMP36 deve ser conectado ao 5V do Arduino.
- GND do TMP36 deve ser conectado ao GND do Arduino.

- Vout (saída de tensão) do TMP36 deve ser conectada a um pino analógico do Arduino (por exemplo, A0).

Agora, confira o código!

c

Atividade 2

Como os microcontroladores processam sinais analógicos, como os de temperatura ou luz, para utilização em sistemas de monitoramento?

A Usando um osciloscópio embutido para medir diretamente as variações de tensão.

B Convertendo sinais analógicos em digitais por meio de seus conversores analógico-digitais (ADCs) integrados.

C Utilizando uma técnica de amostragem aleatória para estimar o valor do sinal analógico.

D Amplificando os sinais analógicos antes de transmiti-los diretamente para um display.

E Gerando um sinal PWM correspondente ao valor medido do sinal analógico.

Parabéns! A alternativa B está correta.

Microcontroladores convertem sinais analógicos em digitais por meio de ADCs para adquirir dados precisos de sensores. Já microcontroladores não possuem osciloscópios. Amostragem aleatória não é padrão; amplificação é diferente de conversão; PWM é para controle, não para conversão.

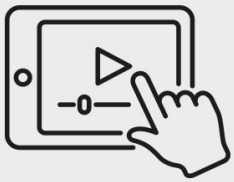
Simulador de microcontrolador Arduino

Simuladores de microcontroladores, como o Tinkercad da Autodesk, são usados no desenvolvimento de sistemas embarcados e em educação em eletrônica. Essa plataforma on-line permite simular circuitos eletrônicos e programar microcontroladores, como o Arduino, sem hardware físico.

Tinkercad é uma ferramenta valiosa para aprender eletrônica e programação de maneira prática e acessível, oferecendo um ambiente seguro para experimentação e facilitando a aplicação prática do conhecimento teórico.

Vamos explorar neste vídeo o uso do Tinkercad como uma ferramenta essencial para a simulação e a programação de circuitos eletrônicos, especialmente com microcontroladores como o Arduino.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Tinkercad

É uma plataforma on-line gratuita, desenvolvida pela Autodesk, que facilita a simulação de circuitos eletrônicos e a programação de microcontroladores como o Arduino.

Acessível a todos os níveis de habilidade, desde iniciantes a profissionais, sua interface intuitiva e o ambiente de desenvolvimento integrado (IDE) permitem a criação, a simulação e o compartilhamento de projetos com facilidade.

> **Ideal para quem deseja aprender eletrônica e programação, o Tinkercad elimina a necessidade de componentes físicos e oferece uma comunidade ativa e recursos educacionais para aprimorar habilidades e realizar projetos complexos.**

Os códigos desenvolvidos podem ser transferidos para a IDE do Arduino, facilitando a transição para hardware físico e a implementação de projetos simulados em dispositivos Arduino sem grandes modificações.

Como acessar o Tinkercad

Acessar e usar o Tinkercad para simular projetos com Arduino é simples. Vamos lá!

①

Passo 1: crie uma conta

- Visite tinkercad.com.
- Selecione Join Now ou Sign Up para criar uma conta.

②

Passo 2: acesse a área de circuitos

- Faça login e escolha Circuits na página inicial.
- Clique em Create new Circuit para começar um projeto.

③

Passo 3: configure o ambiente

Arraste componentes eletrônicos, incluindo o Arduino, para a área de trabalho.

④

Passo 4: adicione componentes e monte o circuito

- Adicione LEDs, resistores e faça conexões.
- Use as ferramentas de fiação para conectar os componentes.

⑤

Passo 5: programe o Arduino

- Acesse Code e escreva ou cole o código em C/C++.
- Use exemplos disponíveis, se desejar.

⑥

Passo 6: simule o código

- Clique em Start Simulation para ver o circuito em ação.
- Ajuste o código e os componentes conforme necessário e repita a simulação.

Atividade 3

Qual dos seguintes aspectos é possibilitado pelo uso de simuladores de microcontroladores, como o Tinkercad, no desenvolvimento de projetos eletrônicos?

A Visualização e teste de projetos em um ambiente virtual.

B Soldagem e montagem física de componentes.

C Fabricação e distribuição de produtos finais.

D Compra e venda de componentes eletrônicos.

E Documentação e patenteação de inovações tecnológicas.

Parabéns! A alternativa A está correta.

Simuladores de microcontroladores permitem a visualização e o teste de projetos em um ambiente virtual, em que usuários podem experimentar e depurar circuitos e códigos antes da execução em hardware real. Isso é importante para identificar falhas de projeto sem os custos da prototipagem física. Eles não estão envolvidos em soldagem, montagem, fabricação, distribuição ou nas áreas de compra e venda, documentação ou patenteação de projetos.

Uso de microcontroladores para o controle de dispositivos

Nesta prática, usaremos a plataforma Tinkercad para explorar o controle de velocidade de um motor DC com modulação por largura de pulso (PWM) e botões push button. Você projetará, simulará e ajustará um circuito eletrônico de forma interativa e segura, sem montagem física. O objetivo é aprender como aplicar o PWM para controlar a velocidade de um motor DC, montando um circuito com um motor DC, botões push button e um Arduino, programando-o para quatro velocidades e parada.

Descubra neste vídeo como montar e programar um circuito usando Arduino para controlar a velocidade de um motor DC com botões push button por meio de PWM.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

Esse resumo guiará você no processo de montagem e programação de um motor DC com velocidade alterada por botão push button utilizando PWM no Tinkercad.

Vamos lá!

Montagem do circuito:

- Arraste um Arduino Uno para a área de trabalho.
- Conecte um motor DC ao Arduino, como positivo em um pino com função PWM, por exemplo, o pino 10.
- Adicione um botão push button para alterar a velocidade do motor.

Programação do Arduino:

- Escreva um código no Arduino para implementar o controle PWM. Defina os pinos correspondentes ao botão e ao motor.
- No loop principal do código, verifique se algum botão é pressionado.

- Ajuste o valor do PWM baseado no botão pressionado: aumente a largura do pulso para controlar a velocidade do motor.

Simulação e testes:

- Inicie a simulação no Tinkercad.
- Pressione o botão alternadamente e observe a mudança na velocidade do motor.
- Ajuste o código ou a montagem do circuito conforme necessário para melhorar o desempenho ou corrigir problemas.

Análise de resultados:

- Discuta como as alterações no PWM afetam a velocidade do motor.
- Reflita sobre as aplicações práticas do controle de velocidade em motores DC em projetos de engenharia.

O Tinkercad permite experimentação segura sem as precauções necessárias em circuitos reais. Porém, fora da simulação, é importante usar um driver de motor para proteger o Arduino de correntes altas, utilizando transistores ou módulos especializados que gerenciam a alta corrente sem danificar o microcontrolador.

Veja o código para usar na prática no Tinkercad.

c

Atividade 4

Analise o programa de Arduino no Tinkercad para controlar a velocidade de um motor DC com um botão. Modifique-o para adicionar um segundo botão que reduza a velocidade. Qual alteração no código habilitaria essa função?

A Adicionar pinMode(7, INPUT_PULLUP); no setup() e inserir else if (digitalRead(7) == LOW) { velocidade--; } no loop().

B Substituir if (digitalRead(6) == LOW) por if (digitalRead(7) == LOW) para permitir a diminuição da velocidade com o novo botão.

C Adicionar velocidade--; diretamente após velocidade++; para alternar entre aumentar e diminuir a velocidade a cada pressionamento do botão.

D Inserir uma nova condição if (digitalRead(7) == HIGH) { velocidade--; } antes da condição existente para verificar o estado do novo botão.

E Alterar todos os valores em analogWrite para decrementar em vez de incrementar, refletindo a ação do novo botão.

Parabéns! A alternativa A está correta.

A alternativa adiciona corretamente um segundo botão no pino 7, configurado como INPUT_PULLUP. No loop, adiciona-se uma nova condição que diminui a variável velocidade quando o botão no pino 7 é pressionado.

Uso de microcontroladores para aquisição de dados

Aqui, usaremos o Tinkercad para simular e programar um sensor de distância ultrassônico HC-SR04 com uma placa Arduino. O HC-SR04 é um sensor popular para medir distâncias, emitindo ondas ultrassônicas que refletem em objetos e retornam ao sensor.

> **Com o Tinkercad, simulamos o circuito e o código sem componentes físicos, facilitando o aprendizado em eletrônica e programação de forma segura e acessível.**

O objetivo é desenvolver habilidades com sensores ultrassônicos e microcontroladores, essenciais em robótica, automação residencial e sistemas industriais. A simulação é fundamental para testar e ajustar a aquisição de dados antes da implementação real.

Uso de microcontroladores para aquisição de dados

Acompanhe neste vídeo a montagem e a programação do sensor ultrassônico HC-SR04 com Arduino no simulador Tinkercad, demonstrando como as ondas sonoras são utilizadas para medir distâncias.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

O sensor ultrassônico HC-SR04 mede distâncias ao emitir um pulso ultrassônico que ecoa ao atingir um objeto. Ele possui um transmissor, que emite as ondas, e um receptor, que as capta após a reflexão. A distância até o objeto é calculada pelo tempo que o eco demora a voltar, usando a fórmula: distância = (Tempo do Eco × 340 m/s) / 2. O divisor 2 é usado, porque o som viaja até o objeto e retorna, dobrando a distância percorrida.

Veja agora o roteiro para montagem e programação no Tinkercad.

1 - Montagem do circuito:

Busque e arraste para a área de trabalho um Arduino Uno, um sensor HC-SR04.

2 - Conexões:

- Conecte o pino VCC do HC-SR04 ao pino 5V do Arduino.
- Conecte o pino GND do HC-SR04 ao pino GND do Arduino.
- Conecte o pino TRIG do HC-SR04 ao pino digital 7 do Arduino.
- Conecte o pino ECHO do HC-SR04 ao pino digital 8 do Arduino.

3 - Programação:

- Iniciar código: abra o editor de código no Tinkercad.
- Use o código a seguir para programar o Arduino.

c

4 - Simulação:

- Teste o circuito: clique em Iniciar Simulação para verificar o funcionamento do circuito.
- Observe as leituras de distância no console do Serial Monitor.
- Altere, clicando no sensor na área de trabalho do Tinkercad, a simulação da posição de objetos próximos ao sensor para testar diferentes distâncias e verificar a precisão do sensor.

Comentário

A instrução duration = pulseIn(ECHO_PIN, HIGH); no código Arduino é usada para medir a distância pelo sensor ultrassônico HC-SR04. A função pulseIn() mede a duração de um pulso de entrada (HIGH ou LOW) no pino especificado. O pino TRIG envia um pulso ultrassônico, emitindo um pulso HIGH de 10 microssegundos para iniciar a medição. O pino ECHO, configurado como entrada, recebe o eco do pulso ultrassônico. Após o disparo do pino TRIG, o pino ECHO fica HIGH enquanto o som refletido retorna ao sensor, e pulseIn(ECHO_PIN, HIGH) mede esse tempo, que corresponde à viagem do som até o objeto e de volta ao sensor.

Atividade 5

Você está trabalhando em um projeto utilizando o sensor ultrassônico HC-SR04 e um Arduino no simulador Tinkercad para monitorar a distância até objetos próximos. Atualmente, o programa mede a distância e a exibe no monitor serial.

O objetivo, agora, é adicionar uma funcionalidade ao seu projeto: fazer um LED piscar a cada 100 ms quando a distância medida for menor do que 50 cm. Você deve modificar o código para incluir essa funcionalidade.

Qual alteração você deve fazer no código para que um LED conectado ao pino 13 do Arduino pisque a cada 100 ms sempre que a distância medida for menor do que 50 cm? (Teste sua resposta no Tinkercad).

A Adicione pinMode(13, INPUT); no setup() e if (distanceCm < 50) { digitalWrite(13, HIGH); delay(100); digitalWrite(13, LOW); } no loop().

B Adicione pinMode(13, INPUT); no setup() e if (distanceCm < 50) { digitalWrite(13, HIGH); delay(100); digitalWrite(13, LOW); delay(100); } no loop().

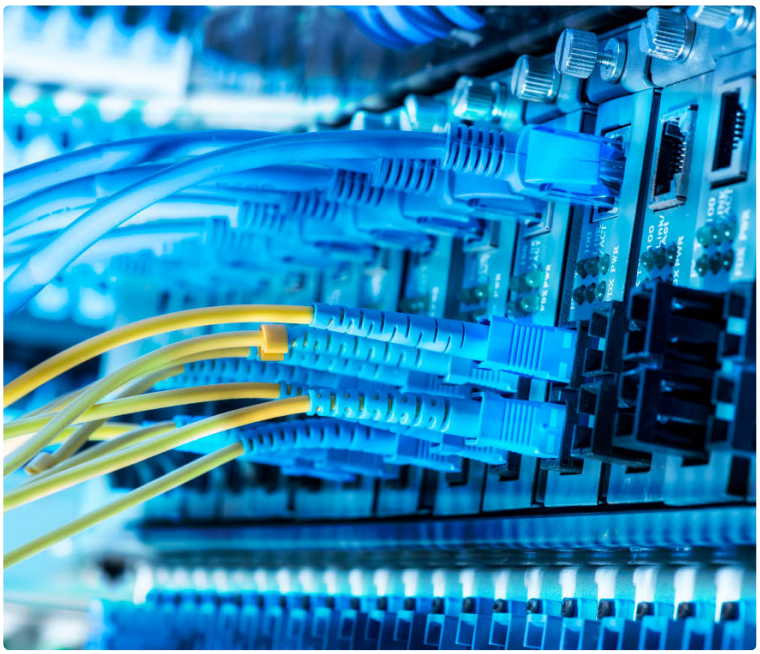
C Adicione pinMode(13, OUTPUT); no setup() e if (distanceCm < 50) { digitalWrite(13, HIGH); delay(100); digitalWrite(13, LOW); } no loop().

D Adicione digitalWrite(13, HIGH); no setup() e if (distanceCm < 50) { digitalWrite(13, LOW); delay(100); digitalWrite(13, HIGH); delay(100); } no loop().

E Adicione digitalWrite(13, HIGH); no setup() e if (distanceCm < 50) { digitalWrite(13, LOW); delay(100); digitalWrite(13, HIGH); delay(900); } no loop().

Parabéns! A alternativa C está correta.

A alternativa garante que o LED piscará a cada 100 ms apenas quando a distância medida for menor do que 50 cm. Essa configuração assegura que o LED não fique aceso continuamente, mas sim que pisque de maneira perceptível, indicando alerta de proximidade.



3 - Sockets e cliente/servidor TCP e UDP

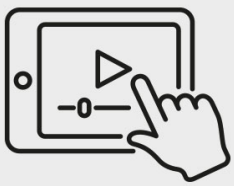
Ao final deste módulo, você será capaz de identificar os conceitos e as funções de sockets na criação de conexões em rede TCP e UDP em programas com linguagem C.

Introdução aos sockets na linguagem C

Os fundamentos da comunicação em rede via sockets são muito importantes para desenvolvedores de software. A compreensão teórica dos sockets em C e suas APIs, como POSIX para sistemas Unix, e Winsock, para Windows, habilita programadores a manipular dados em redes, além de facilitar a integração e a operação de sistemas, criando uma conexão entre teoria e aplicação prática na programação.

Vamos conferir neste vídeo o funcionamento dos sockets como canais de comunicação entre programas, mostrando a distinção entre as APIs de sockets POSIX e Winsock. Assista!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Sockets são uma maneira de estabelecer uma comunicação bidirecional entre dois programas rodando em uma rede; programas estes que podem estar na mesma máquina ou em máquinas diferentes.

Na linguagem C, essa comunicação é feita usando a API de sockets, que permite aos programas em C a criação e o gerenciamento de conexões de rede. Portanto, os sockets em C são uma maneira bastante útil de entender como a comunicação em rede funciona no nível de programação.

Entendendo sockets

São pontos finais para comunicação de rede bidirecional entre programas. Cada programa em uma rede utiliza um socket para enviar e receber dados, associando-se a portas de comunicação na camada de transporte.

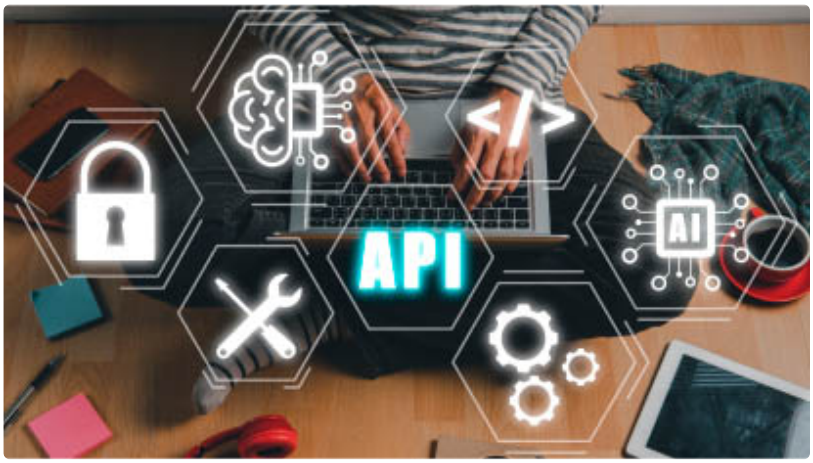
Exemplo

Imagine uma rede como um sistema de correios global, em que casas representam dispositivos, e endereços IP correspondem a endereços reais. Cada casa pode receber diferentes tipos de informação — por exemplo, cartas e pacotes —, que são entregues em portas específicas. Assim, sockets funcionam como caixas de correio, combinando um endereço IP e um número de porta específico.

Ao comunicar, um programa abre um socket vinculado a um número de porta, direcionando o tráfego de rede ao aplicativo correto no dispositivo receptor.

APIs de sockets mais conhecidas para a linguagem C

Uma API (*application programming interface*) é um conjunto de regras que programas seguem para se comunicarem, agindo como um contrato que especifica a interação entre componentes de software.



Representação de API.

Diferentemente de uma biblioteca, que é uma coleção de funcionalidades chamadas dentro de um programa, uma API define

como os sistemas se comunicam, podendo ser fornecida por uma biblioteca, um sistema operacional ou um serviço de rede.

Na linguagem C, as APIs de sockets, como POSIX para sistemas Unix e Winsock para Windows, são fundamentais para a programação de comunicação em rede, sendo bastante usadas por programadores para desenvolver aplicações. Conheça mais sobre elas!

API de sockets POSIX

Derivada dos sockets BSD (Berkeley Software Distribution), é o padrão para comunicações de rede na esfera dos sistemas operacionais tipo Unix, como Linux e macOS. Ela define um conjunto de funções para manipulação de sockets, utilizadas por programas para enviar e receber dados via redes TCP/IP. Com a API POSIX, é possível criar sockets, configurá-los para acessar portas específicas, aceitar conexões e enviar/receber mensagens de forma síncrona ou assíncrona. Essa uniformidade facilita a portabilidade do código de rede entre diferentes variantes UNIX com alterações mínimas.

API Winsock

Embora baseada nos conceitos de sockets do BSD, é adaptada e expandida para melhor se adequar ao modelo de programação do Windows. Ela fornece funcionalidades semelhantes às da POSIX, como criação e gerenciamento de conexões de rede, e incorpora características específicas do Windows, como integração com o modelo de mensagem do Windows e funções específicas para operações de rede assíncronas. Uma vantagem da Winsock é facilitar a portabilidade de aplicativos de rede do UNIX para o Windows, ajudando os desenvolvedores a manterem uma base de código consistente.

Atividade 1

Sockets são fundamentais na comunicação de rede, funcionando como pontos finais para a troca de dados entre programas. Considere as seguintes afirmações a respeito de sockets em uma rede de comunicação:

I. Sockets funcionam como pontos de entrega em que apenas dados recebidos são manipulados.

II. Um socket é definido pela combinação de um endereço IP com um número de porta.

III. Os números de porta em um socket ajudam a direcionar o tráfego de rede para outro dispositivo.

Qual das seguintes opções está correta?

A Apenas I está correta.

B Apenas II está correta.

C Apenas III está correta.

D II e III estão corretas.

E I e II estão corretas.

Parabéns! A alternativa B está correta.

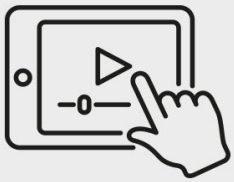
Um socket é identificado pela combinação de um endereço IP e um número de porta, essencial para direcionar o tráfego de rede ao aplicativo correto no dispositivo receptor. Sockets podem enviar e receber dados, e não estão limitados à recepção. Além disso, os números de porta asseguram que o tráfego de rede chegue ao aplicativo correto, e não a outro.

Transferência de dados com sockets

A velocidade e a eficiência na transferência de dados é uma busca constante, e os sockets facilitam e atuam nessas interações imediatas entre cliente e servidor. Dominar a transferência de dados via sockets, incluindo a ordem correta dos bytes, é uma base importante para profissionais e empresas que dependem de tecnologias web e móveis, garantindo interoperabilidade em ambientes profissionais interconectados.

Neste vídeo, você entenderá como sockets funcionam como pontes entre cliente e servidor. Não deixe de conferir!

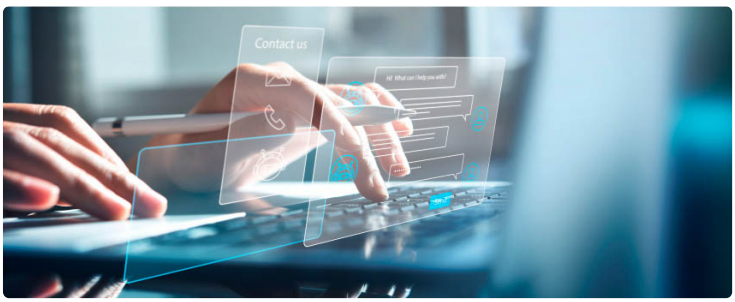
Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Ao acessar um site, o computador (cliente) utiliza um socket para conectar-se ao servidor que hospeda o site. O computador solicita a visualização do site e, o servidor, por outro socket, envia os dados para exibir o site no navegador. Isso ocorre rapidamente, permitindo uma troca complexa de dados por meio do socket, como um tubo que conecta cliente e servidor, possibilitando o fluxo livre de dados.

Exemplos práticos

Fique por dentro de algumas utilidades dos sockets.

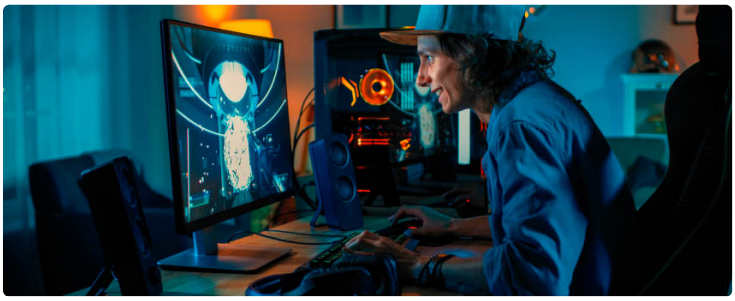


Representação de chatbot.

Chat on-line

Em uma aplicação de chat em tempo real. Quando você envia uma mensagem, ela é rapidamente entregue ao servidor e encaminhada ao destinatário pelos sockets, que estabelecem

uma conexão direta entre o seu dispositivo, o servidor e o dispositivo do destinatário.



Pessoa jogando on-line.

Jogos on-line

Em jogos MMO (massive multiplayer on-line), os sockets são essenciais para sincronizar as ações dos jogadores em tempo real. Movimentos e comandos são enviados ao servidor e distribuídos aos outros jogadores, mantendo o jogo fluido e sincronizado.

Pessoa em transmissão de vídeo ao vivo pelo smartphone.

Transmissão de vídeo ao vivo

Durante uma transmissão, os sockets transmitem o vídeo do servidor para o seu dispositivo em pequenos pacotes de dados, permitindo visualização em tempo real.

Ordem de bytes na comunicação via sockets

Quando falamos de comunicação via sockets, um aspecto técnico a ser considerado é a ordem dos bytes.

O que é ordem dos bytes?

Refere-se à sequência pela qual os bytes de um dado numérico são organizados na memória. Essencialmente, existem dois tipos. Vamos conhecê-los!

Big-endian

O byte mais significativo (MSB) é armazenado no menor endereço de memória, e os bytes subsequentes são armazenados em endereços crescentes.

Little-endian

O byte menos significativo (LSB) é armazenado no menor endereço de memória.

E por que isso importa na comunicação via socket?

Em redes com máquinas de arquiteturas diferentes, ambos os lados devem interpretar os dados de forma idêntica. Diferenças na ordem de bytes, como big-endian e little-endian, podem distorcer o valor dos dados transmitidos.

Para assegurar a interoperabilidade, os dados são convertidos para uma ordem padrão conhecida como "ordem de bytes da rede" (big-endian) antes de serem enviados. Funções de conversão de ordem de bytes da rede são empregadas para ajustar os dados entre a ordem de bytes do host e a ordem de bytes da rede antes da transmissão e após a recepção, garantindo a consistência dos dados entre sistemas diversos. Veja a diferença entre htonl e htons a seguir.

htonl() (host to network long) e ntohl() (network to host)		htons() (host to network short) e ntohs() (network to host short)
Convertem longos inteiros (32 bits), como um endereço IPv4, de ordem de bytes do host para ordem de bytes da rede e vice-versa.	✕	Convertem inteiros curtos (16 bits), como a porta TCP/UDP, de ordem de bytes do host para ordem de bytes da rede e vice-versa.

Atividade 2

A ordem dos bytes, ou endianness, descreve a forma como os bytes são organizados na memória de um computador e pode afetar a interpretação dos dados transmitidos entre sistemas que usam diferentes ordens de bytes.

Considere as seguintes afirmações sobre a ordem dos bytes em comunicação via sockets:

I. A ordem dos bytes tem impacto na comunicação entre computadores da mesma arquitetura.

II. Em big-endian, o byte menos significativo é armazenado no menor endereço de memória.

III. A correta manipulação da ordem dos bytes é essencial para evitar erros de interpretação de dados entre sistemas heterogêneos.

Qual das seguintes opções está correta?

A Apenas I está correta.

B Apenas II está correta.

C Apenas III está correta.

D I e III estão corretas.

E II e III estão corretas.

Parabéns! A alternativa C está correta.

Destaca-se a importância da manipulação adequada da ordem dos bytes para garantir a interpretação correta dos dados numéricos entre sistemas com diferentes ordens de bytes, prevenindo erros de comunicação. A ordem dos bytes não afeta computadores com a

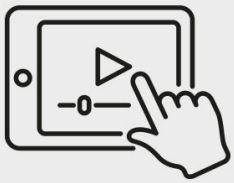
mesma arquitetura. E, em big-endian, o armazenado no menor endereço é o byte mais significativo.

Cliente/servidor: TCP e UDP

Navegar na internet envolve uma rede complexa de processos e tecnologias. Compreender os fundamentos da comunicação na internet, incluindo o modelo cliente/servidor, protocolos como TCP e UDP, e o uso de sockets, desenvolve a capacidade de gerenciar e inovar em aplicações na internet de forma eficiente e segura.

Neste vídeo, vamos discutir a arquitetura da comunicação na internet, destacando o funcionamento dos modelos cliente/servidor, a importância dos protocolos TCP e UDP e como os sockets facilitam a interação entre aplicações e rede. Assista!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Imagine que você (o cliente) esteja em um restaurante e faça um pedido ao garçom (o servidor), que leva seu pedido à cozinha (o sistema de processamento) para preparar sua comida. Após ser preparada, o garçom volta e entrega a comida a você.

No mundo digital, a dinâmica é similar: seu computador ou smartphone (cliente) faz um pedido a um servidor remoto (servidor), solicitando uma página web, por exemplo. O servidor processa esse pedido e envia de volta os dados solicitados, como o conteúdo de uma página.

TCP e UDP: os mensageiros

TCP (Transmission Control Protocol) e UDP (User Datagram Protocol) são dois protocolos usados para enviar bits de dados — conhecidos como "pacotes" — pela internet. Eles são como dois tipos de serviços de entrega, cada um com suas características e usos específicos. Conheça mais sobre eles a seguir.

TCP: o entregador confiável

Funciona como um entregador que assegura a entrega correta e ordenada de pacotes. É ideal para mensagens importantes que não podem ser perdidas ou chegar fora de ordem. O TCP garante isso com: confirmação de recebimento, onde o receptor envia um recibo de volta; ordenação de pacotes, reorganizando-os se chegarem desordenados; e controle de fluxo, ajustando a taxa de envio para não sobrecarregar o receptor. Exemplos práticos do uso do TCP incluem baixar arquivos, navegar em sites e enviar e-mails, onde a confiabilidade e a ordem são fundamentais.

UDP: o rápido e leve

Funciona como um serviço de correio rápido que não requer confirmações de entrega, enviando pacotes rapidamente sem garantir sua ordem ou chegada. Isso o torna mais rápido, porém menos confiável que o TCP. É geralmente usado em streaming de vídeo ou áudio, em que o fluxo contínuo é mais importante, e em jogos on-line, em que a velocidade é crucial, mesmo que alguns pacotes se percam.

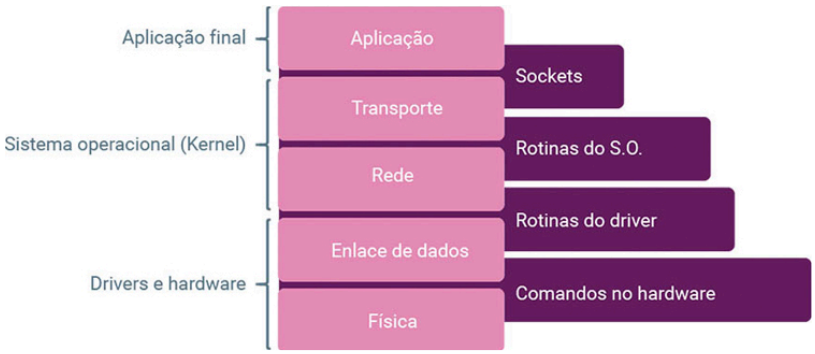
Sockets: a conexão real

Desenvolver um aplicativo que utiliza comunicação via internet implica abrir e gerenciar sockets e escolher entre os protocolos TCP ou UDP conforme a necessidade. Cada socket é identificado por um endereço IP e um número de porta.

> Em um ambiente cliente/servidor, o servidor "ouve" em um IP e porta específicos, aguardando solicitações de clientes. Quando um cliente quer conectar, ele usa o IP e a porta do servidor para estabelecer a conexão via socket.

Em resumo, os sockets operam na camada de transporte, funcionando como um ponto final para a comunicação que acontece entre as aplicações cliente e servidor, utilizando os protocolos TCP ou UDP.

Observe agora a relação entre as camadas de rede e a API de sockets no contexto do sistema operacional e do hardware.



Relação entre as camadas de rede e a API de sockets.

A aplicação final na imagem é a camada na qual usuários interagem com programas, incluindo navegadores e aplicativos de mensagens, que se comunicam via rede usando a API de sockets. Sockets agem como conectores virtuais, permitindo o envio e o recebimento de dados. Essa API é parte do sistema operacional (Kernel), que processa solicitações de comunicação quando uma aplicação acessa um socket.

> A API de sockets trabalha diretamente com a camada de transporte, que usa TCP ou UDP para estabelecer comunicação fim a fim, gerenciando a confiabilidade e a sequência dos pacotes.

As camadas de enlace de dados e física lidam com a transmissão física dos dados através de meios, como cabos de cobre, fibra óptica ou ondas de rádio, garantindo que os dados cheguem ao seu destino final.

Atividade 3

Um profissional de TI está configurando um serviço de transmissão de vídeo ao vivo para um evento on-line. Considerando as características dos protocolos de comunicação, qual protocolo ele deveria escolher para maximizar a eficiência da transmissão?

A TCP, porque garante que todos os pacotes de dados cheguem sem erros e na ordem correta.

B UDP, porque permite a transmissão rápida de dados, o que é essencial para manter a fluidez do vídeo ao vivo.

C TCP, porque ajusta a taxa de transmissão de dados para evitar a sobrecarga do servidor.

D UDP, porque requer confirmações frequentes de recebimento que podem atrasar a transmissão.

E TCP, porque sua prioridade é a velocidade de transmissão, e não a integridade dos dados.

Parabéns! A alternativa B está correta.

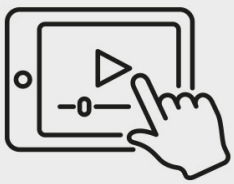
O UDP é ideal para streaming de vídeo ao vivo devido à sua capacidade de enviar pacotes rapidamente sem esperar por confirmações de recebimento, o que mantém a fluidez e reduz a latência; essenciais para a qualidade do streaming em tempo real.

Sockets UDP elementares

O protocolo UDP é ideal para aplicações que exigem transmissão rápida e eficiente, como jogos on-line e chamadas de vídeo. Apesar de não garantir a entrega ou ordem dos pacotes, sua abordagem "envie e esqueça" permite comunicação ágil, valorizando baixa latência sobre integridade completa dos dados.

Neste vídeo, vamos explorar o protocolo UDP, ressaltando sua rapidez e abordagem "envie e esqueça", ideal para contextos em que a agilidade é prioritária, como jogos e streaming. Acompanhe!

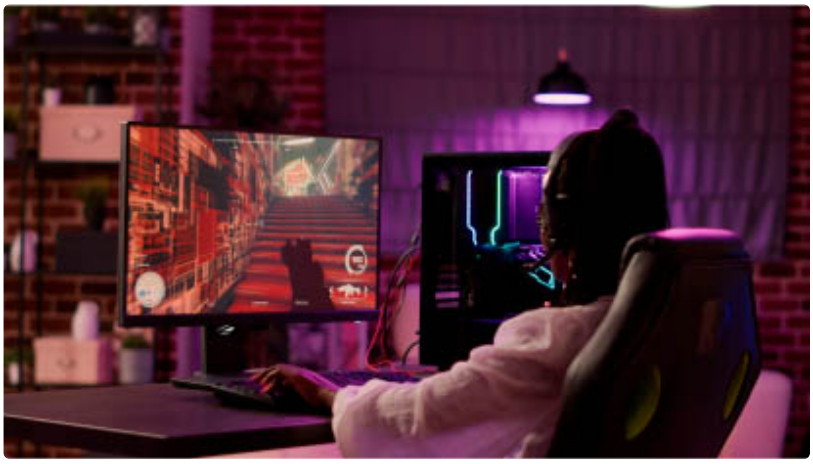
Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



UDP, um protocolo da camada de transporte no modelo OSI e TCP/IP, opera pelo modelo "envie e esqueça". Ao enviar um datagrama, não espera a confirmação de recebimento. Simples e rápido, é ideal quando a velocidade é prioritária, e não a garantia de entrega.

Por que usar UDP?

Ao jogar on-line ou durante uma chamada de vídeo, o UDP é preferido ao TCP para evitar latência.



Pessoa jogando on-line.

O UDP envia informações sem esperar por confirmação, mantendo jogos rápidos e chamadas fluindo mesmo com pequenas perdas de pacotes. É ideal também para streaming de vídeo e áudio, em que é aceitável perder alguns frames ou ter falhas no áudio para evitar atrasos.

Na internet das coisas (IoT) e em aplicações de monitoramento e telemetria, a simplicidade e a eficiência do UDP permitem o envio rápido e com baixa sobrecarga de dados frequentes e pequenos.

Desafios e considerações

O UDP é apreciado por sua simplicidade e velocidade, mas enfrenta desafios como a falta de confirmação de recebimento, perda de pacotes, chegada fora de ordem ou duplicação. Sem controle de fluxo ou congestionamento, pode sobrecarregar rapidamente a rede ou o destinatário. A responsabilidade de gerenciar essas questões, normalmente, recai sobre a aplicação. A escolha entre UDP, TCP ou ambos depende das necessidades específicas do projeto.

Sockets UDP em C

Trabalhar com sockets UDP em C é um processo interessante, que envolve algumas funções específicas da API de sockets, a qual é uma

interface padrão para programação de rede em muitos sistemas operacionais. Confira as funções mais importantes e como elas diferem, quando aplicável, das usadas para sockets TCP.

socket()

UDP: int sockfd = socket(AF_INET, SOCK_DGRAM, 0);

TCP: int sockfd = socket(AF_INET, SOCK_STREAM, 0);

Essa função cria um novo socket. O segundo argumento, SOCK_DGRAM, indica que se trata de um socket datagrama (UDP), usado para enviar pacotes não confiáveis de tamanho fixo. Para TCP, usa-se SOCK_STREAM, que cria um socket de fluxo, representando uma conexão confiável, orientada a byte.

bind()

Na programação de redes, a operação bind associa um socket a um endereço IP e um número de porta específicos para que servidores definam onde escutarão por conexões ou dados. Ao criar um socket com a função socket(), ele ainda não tem um endereço de rede. A função bind() resolve isso ao amarrar o socket a uma combinação específica de IP e porta, permitindo que receba dados enviados para esse endereço.

sendto()

Envia dados para um endereço específico. Essa função é exclusiva do UDP, pois permite enviar um datagrama a um endereço especificado sem a necessidade de estabelecer uma conexão.

recvfrom()

Recebe dados de um endereço. Diferentemente do TCP, em que os dados são lidos de uma conexão estabelecida, no UDP, cada

chamada a recvfrom() pode receber dados de um endereço diferente.

connect()

Em TCP, connect() é usado para estabelecer uma conexão e iniciar a comunicação. No UDP, é opcional e serve para definir o endereço padrão para o qual os dados serão enviados, permitindo usar send() e recv(), em vez de sendto() e recvfrom().

listen() e accept()

Funções específicas do TCP, relacionadas à sua natureza de conexão. listen() coloca o socket em modo de escuta, enquanto accept() aceita uma conexão de entrada, criando um novo socket para essa conexão.

Atividade 4

Você foi escolhido para decidir qual protocolo usar em um novo jogo on-line que sua equipe está desenvolvendo. Examine as afirmações seguintes, com a relação fornecida pelo seu chefe.

Asserção (A)

Com frequência, o UDP é usado em jogos on-line, porque não requer confirmação de recebimento dos pacotes, reduzindo a latência e aumentando a responsividade do jogo.

Razão (R)

Em aplicações como jogos on-line, a velocidade e a fluidez da comunicação são mais importantes do que a garantia de que cada pacote seja entregue ou recebido na ordem correta.

Qual das opções melhor descreve a relação entre a asserção e a razão?

A Tanto a asserção quanto a razão são verdadeiras, e a razão é uma explicação correta da asserção.

B Tanto a asserção quanto a razão são verdadeiras, mas a razão não é uma explicação correta da asserção.

C A asserção é verdadeira, mas a razão é falsa.

D A asserção é falsa, mas a razão é verdadeira.

E Tanto a asserção quanto a razão são falsas.

Parabéns! A alternativa A está correta.

A asserção a respeito do UDP é verdadeira, porque ele não aguarda confirmações de recebimento, reduzindo a latência e melhorando a responsividade; essencial para jogos on-line. A razão confirma e explica corretamente essa asserção, destacando que, em jogos, a velocidade e a fluidez são prioritárias em detrimento da integridade total dos dados.

Transferindo dados com sockets na prática

Construir uma aplicação de mensagens simples usando o modelo cliente/servidor e sockets com Winsock no Windows é uma prática excelente para aprender os fundamentos da comunicação de rede. Esse projeto envolve usar o protocolo TCP, que assegura a entrega confiável de dados entre programas em computadores distintos. Vamos desenvolver tanto o lado do servidor, que escuta e responde às solicitações, quanto o lado do cliente, que inicia a comunicação.

Acompanhe neste vídeo a criação de uma aplicação de mensagens utilizando o modelo cliente/servidor com sockets e Winsock no Windows, destacando a importância da comunicação de rede e do protocolo TCP para transferências de dados confiáveis.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

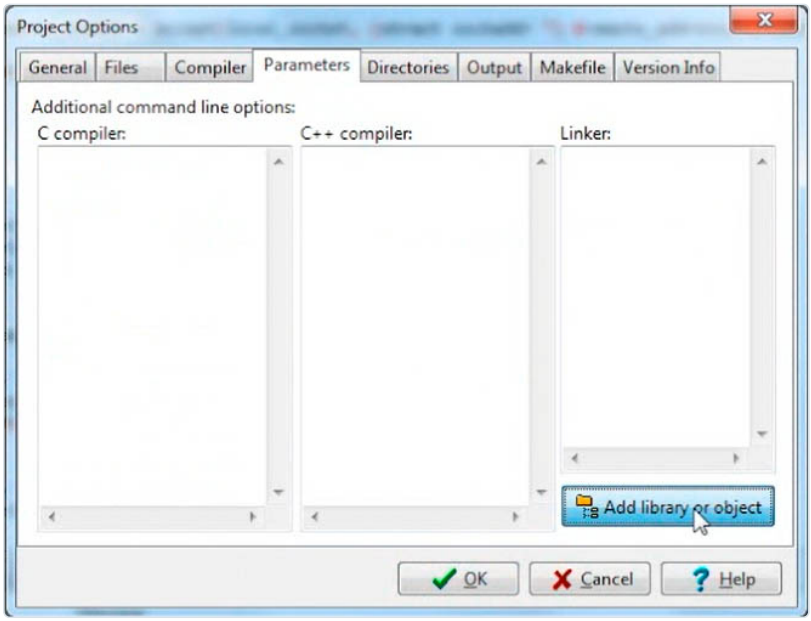
Roteiro de prática

1. Configuração do projeto

Abra o Dev-C++ e crie um novo projeto para cada aplicativo (cliente e servidor). Escolha um projeto de Console C++ para cada um.

Inclua o Winsock: adicione #include <winsock2.h> no topo de seus arquivos de código-fonte. Isso é necessário para acessar as funções da biblioteca Winsock.

Link com a biblioteca Winsock: para que seu programa use Winsock, você precisa adicionar a biblioteca. No Dev-C++, isso pode ser feito nas opções do projeto na aba Parâmetros (imagem a seguir), adicionando o arquivo libwsock32.a à lista de bibliotecas. O arquivo está na pasta do DevC++, em caminho semelhante a C:\Program Files (x86)\Dev-Cpp\MinGW64\x86_64-w64-mingw32\lib32.



Tela para adicionar a biblioteca Winsock ao projeto.

---

2. Roteiro do servidor (estará no endereço local 127.0.0.1)

Inicialize o Winsock: use a função WSAStartup para iniciar o uso da DLL Winsock.

Use socket() para criar um socket que ouvirá as conexões dos clientes.

Utilize uma estrutura sockaddr_in para definir onde seu servidor deverá escutar (por exemplo, INADDR_ANY para qualquer interface de rede) e a

porta.

Associe o endereço ao socket usando bind().

Coloque o socket em modo de escuta com listen(), especificando o número máximo de conexões pendentes.

Use accept() para aceitar novas conexões dos clientes. Isso bloqueará o servidor até que uma conexão seja estabelecida.

Após a conexão, use send() e recv() para a comunicação entre cliente e servidor.

Feche o socket com closesocket() após a conclusão da comunicação.

Chame WSACleanup() para finalizar o uso da DLL Winsock.

Veja o código completo em C para o servidor.

c

3. Roteiro do cliente

Siga os passos a seguir.

- Inicialize o Winsock: similar ao servidor, use WSAStartup.

- Crie o socket do cliente.
- Use uma estrutura sockaddr_in para definir o endereço IP e a porta do servidor.
- Conecte-se ao servidor com connect().
- Use send() para enviar dados e recv() para receber dados do servidor.
- De modo semelhante ao servidor, feche o socket com closesocket() e chame WSACleanup().

Veja o código completo em C para o cliente.

c

Atividade 5

Você foi solicitado a alterar o programa cliente da prática para que o valor enviado ao servidor seja um número aleatório entre 250 e 270, a

cada segundo. Faça testes no programa cliente e indique qual alteração deve ser feita para que o programa atenda a esse requisito.

A Modificar o código para incluir a biblioteca time.h e utilizar a função time(NULL) para gerar o número aleatório. Remover a leitura da mensagem do usuário (gets(message)).

B Alterar o laço de repetição para executar o envio a cada segundo, usando a função Sleep(1000), e substituir a entrada manual por uma geração automática de números usando rand() % 21 + 250. Remover a leitura da mensagem do usuário (gets(message)).

C Inserir uma chamada à função srand() no início do programa para inicializar o gerador de números aleatórios.

D Substituir o código de entrada de mensagens por um loop que gere um número aleatório entre 250 e 270, e o envie, utilizando a função rand() apenas. Remover a leitura da mensagem do usuário (gets(message)).

E Adicionar uma chamada à função system("pause 1"), após cada envio, para garantir que o envio ocorra a cada segundo.

Parabéns! A alternativa B está correta.

Essa opção substitui a entrada manual por uma geração automática de números aleatórios entre 250 e 270, além de adicionar a função Sleep(1000) para garantir que as mensagens sejam enviadas em intervalos de um segundo. A chave aqui é a remoção da chamada gets(message), que permite que o programa gere, automaticamente, a mensagem, em vez de aguardar uma entrada do usuário.

4 - Cliente/servidor HTTP e cliente MQTT para IoT

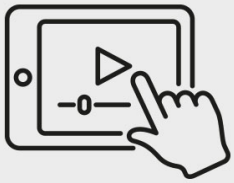
Ao final deste módulo, você será capaz de desenvolver programas cliente/servidor HTPP e com protocolo MQTT para o envio de dados entre dispositivos pela internet.

Conceitos sobre a arquitetura cliente/servidor HTTP

Formam a base da web, por meio do diálogo entre cliente e servidor, pelo protocolo de transferência de hipertexto (hypertext transfer protocol). Essa arquitetura facilita a interação dinâmica e a distribuição global de conteúdo e serviços, destacando-se por sua escalabilidade e eficiência.

Confira neste vídeo a essência da arquitetura cliente/servidor HTTP, pilar fundamental na engenharia de software e na comunicação de dados, especialmente na internet.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Como funciona o HTTP?

O funcionamento do HTTP começa quando você digita um URL e pressiona Enter, desencadeando os passos a seguir.

Iniciando a conexão: cliente e servidor no protocolo HTTP

O cliente (navegador) inicia uma conexão TCP com o servidor, que pode incluir segurança SSL/TLS para HTTPS. A conexão, normalmente, ocorre na porta 80, para HTTP, e na 443, para HTTPS.

Requisição HTTP: método, caminho e cabeçalhos

O navegador envia uma requisição HTTP ao servidor, incluindo o método (GET ou POST), o caminho do recurso e cabeçalhos com informações adicionais.

Processamento da requisição no servidor: ações e respostas

O servidor processa a requisição, podendo acessar arquivos, interagir com bancos de dados ou realizar cálculos para preparar a resposta.

Resposta do servidor: status, cabeçalhos e corpo da mensagem

O servidor responde com uma linha de status (por exemplo: 200 OK, 404 Not Found), cabeçalhos sobre o servidor e o tipo de conteúdo, e um corpo de mensagem com os dados solicitados.

Gerenciamento da conexão TCP: fechamento e persistência

Dependendo do protocolo HTTP e dos cabeçalhos de controle, a conexão TCP pode ser fechada ou mantida aberta para futuras requisições. Em HTTP/1.1, conexões persistentes permitem múltiplas trocas antes do fechamento.

Processamento da resposta no navegador: renderização e recursos adicionais

O navegador processa a resposta, apresentando o HTML ou solicitando recursos adicionais, como imagens e scripts, até que o conteúdo esteja carregado por completo e interativo para o usuário.

Por que a arquitetura cliente/servidor HTTP é importante?

Você pôde notar que a arquitetura cliente/servidor HTTP é a base da web. Ela permite a entrega de conteúdo web e a construção de aplicações interativas que podem ser acessadas de qualquer lugar do mundo. Confira as várias vantagens oferecidas por essa arquitetura.

Escalabilidade

Facilita o atendimento de milhares ou até milhões de clientes simultaneamente.

Flexibilidade

Permite o uso de diferentes tecnologias e linguagens de programação no lado do cliente e do servidor.

Manutenção

A separação entre cliente e servidor facilita a manutenção e a atualização dos sistemas.

Eficiência

Otimiza o uso de recursos, permitindo que os servidores lidem com a lógica de negócios e o armazenamento de dados, enquanto os clientes focam a apresentação e a interação com o usuário.

Atividade 1

Qual das seguintes etapas descreve corretamente o que acontece quando um usuário digita um URL no navegador e pressiona Enter, seguindo o fluxo de funcionamento do protocolo HTTP?

A O navegador compila todas as informações da página em um arquivo único e envia diretamente para o servidor como uma requisição única.

B O navegador, agindo como cliente, começa uma conexão TCP com o servidor, que pode ser segurada por SSL/TLS se o HTTPS estiver sendo utilizado.

C De imediato, o servidor fecha a conexão TCP após receber a requisição para garantir maior segurança na transmissão dos dados.

D O navegador envia uma requisição HTTP ao servidor incluindo um código de criptografia para confirmar a

identidade do usuário.

E Após receber a resposta do servidor, o navegador bloqueia futuras requisições HTTP para garantir que os dados recebidos não sejam alterados.

Parabéns! A alternativa B está correta.

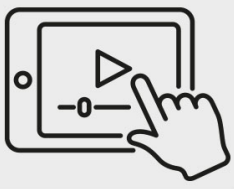
A inicialização da comunicação web ocorre da seguinte maneira: o cliente (navegador) estabelece uma conexão TCP com o servidor, usando SSL/TLS para HTTPS, assegurando comunicação segura. O navegador envia várias requisições conforme o HTML; o servidor não fecha imediatamente a conexão; a requisição HTTP não carrega código de criptografia de identidade; e o navegador continua solicitando recursos conforme necessário.

Conceitos sobre a arquitetura MQTT para Iot

O protocolo MQTT (message queuing telemetry transport) é muito utilizado para a internet das coisas (IoT). Entender o MQTT enriquece o conhecimento teórico e abre caminho para aplicações inovadoras em IoT, em campos como automação residencial, monitoramento agrícola e cidades inteligentes.

Neste vídeo, vamos destacar a importância do MQTT como protocolo essencial para a IoT, enfatizando a origem e o propósito de facilitar comunicações em redes com restrições. Confira!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



O MQTT é um protocolo de mensagens leve e fácil de implementar, desenhado especialmente para dispositivos de baixo poder de processamento e ambientes de rede limitados. Tal descrição se encaixa perfeitamente no perfil da maioria dos dispositivos IoT.

Estrutura do MQTT

O broker MQTT é o núcleo da arquitetura MQTT, atuando como um servidor central que gerencia a troca de mensagens entre dispositivos. Ele recebe mensagens publicadas, filtrando-as por tópicos, e distribui os dispositivos subscritos a esses tópicos.

Os clientes, que podem ser dispositivos ou aplicativos, operam como publicadores ou subscritores. Observe a diferença entre eles.

Publicadores		Subscritores
Enviam mensagens ao broker associadas a tópicos específicos.	✕	Recebem mensagens de tópicos nos quais estão inscritos, possibilitando comunicação bidirecional.

Os tópicos, hierarquicamente estruturados, funcionam como canais em que clientes publicam ou se inscrevem, organizando a comunicação de maneira eficaz, como em sistemas de automação residencial com tópicos segmentados por funções da casa.

Qualidade de serviço (QoS)

O MQTT oferece três níveis de QoS para entrega de mensagens. Vejamos!

QoS 0

Entrega a mensagem no máximo uma vez, sem garantia de recepção. Usado quando a perda de mensagens é tolerável.

QoS 1

Assegura que a mensagem chegará ao menos uma vez, podendo haver duplicatas.

QoS 2

Garante cada mensagem recebida exatamente uma vez, sendo o modo mais seguro, porém o mais lento.

Por que MQTT é ideal para IoT?

O MQTT é adequado à internet das coisas (IoT) devido ao seu design eficiente, que atende aos desafios de dispositivos IoT. Vamos conhecer alguns deles.

Eficiência de banda larga

Projetado para redes com largura de banda limitada, como os celulares, o MQTT reduz overhead de rede com um modelo de publicação/assinatura e pacotes compactos, maximizando a transmissão de dados úteis.

Baixo consumo

Ideal para dispositivos alimentados por bateria, suporta "sleep mode", ativando-os apenas para comunicação, minimizando o uso de energia.

Escalabilidade

Adapta-se ao crescimento do número de dispositivos IoT, permitindo que muitos se conectem sem degradar o desempenho.

Segurança

Oferece criptografia SSL/TLS e autenticação de clientes.

Flexibilidade e facilidade de integração

A estrutura de tópicos flexível e o protocolo simples facilitam a integração com uma variedade de dispositivos e sistemas, promovendo a comunicação eficiente entre dispositivos diversificados.

Atividade 2

Na arquitetura MQTT, qual componente funciona como um servidor central, que gerencia a troca de mensagens, e quais componentes podem operar como publicadores ou subscritores, utilizando uma estrutura hierárquica para a troca de mensagens?

A O cliente serve como servidor central e os brokers atuam como publicadores ou subscritores por meio de canais.

B O tópico serve como o servidor central, enquanto os brokers são os publicadores ou subscritores usando canais para comunicação.

C O servidor é o componente central, com tópicos atuando como publicadores ou subscritores, comunicando-se por meio dos clientes.

D O broker serve como servidor central, enquanto os clientes podem ser publicadores ou subscritores, utilizando tópicos para comunicação.

E O canal funciona como servidor central, com servidores operando como publicadores ou subscritores, enviando e recebendo mensagens por meio dos clientes.

Parabéns! A alternativa D está correta.

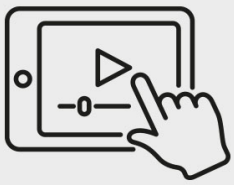
O broker é o servidor central que gerencia as mensagens, ao passo que os "clientes" são dispositivos ou aplicações que se conectam ao broker, atuando como publicadores ou subscritores. "Tópicos" são identificadores hierárquicos que organizam a comunicação entre eles. De outro modo, os termos específicos do MQTT estariam misturados ou usados incorretamente.

Interação entre clientes e servidor web (HTTP)

Abordaremos agora a internet das coisas (IoT) usando o ESP32, uma plataforma de desenvolvimento com wi-fi, para criar um servidor web. Esse servidor permitirá controlar um LED via interface web acessível por qualquer navegador em uma rede local. Utilizaremos a linguagem C, fundamental para o desenvolvimento de sistemas embarcados, para aprofundarmos no mecanismo de comunicação de rede.

Neste vídeo, vamos configurar o ESP32 como servidor web para controlar um LED, destacando a importância da linguagem C na comunicação por rede e na IoT. Assista!

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Roteiro de prática

Esse resumo descreve como configurar um servidor web no ESP32 para controlar um LED via página web.

1. Preparação do ambiente de desenvolvimento: instale o IDE do Arduino e adicione suporte ao ESP32. Certifique-se de que todos os drivers estão instalados para a comunicação com o ESP32.
2. Configuração do servidor web: use a biblioteca WiFi.h para iniciar o modo estação (STA) no ESP32 e conectá-lo à rede Wi-Fi. Configure o servidor web no ESP32 utilizando a classe WebServer.

3. Criação da página web: desenvolva uma página web simples no código do Arduino, que inclua botões ou links para ligar e desligar o LED. Use o exemplo SimpleWifiServer.ino, ajustando o pino do LED para o correto na sua placa.

4. Teste e acesso: carregue o código no ESP32 e acesse o servidor via navegador em um dispositivo na mesma rede Wi-Fi, usando o IP atribuído ao ESP32. A interface deve permitir o controle do LED.

5. Depuração: se o LED não funcionar como esperado, revise as conexões, o código e as configurações de rede, incluindo o IP do ESP32.

Para criar botões no exemplo SimpleWifiServer.ino, altere o código como indicado.

Comente ou retire as instruções:

c

Acrescente, no lugar, as seguintes instruções:

c

Atividade 3

No código HTML e CSS fornecido, usado para criar uma interface de servidor web com botões de controle On e Off, qual alteração deve ser feita para que o botão On seja exibido na cor vermelha? Faça os testes no programa para se certificar do resultado.

A Alterar a cor de fundo na classe .button para #FFFFF0 no CSS.

B Adicionar style="color: red;" diretamente no elemento <button> do botão ON.

C Substituir #4CAF50 por red na propriedade background-color da classe .button.

D Inserir a classe .button3 {background-color: red;} e aplicá-la ao botão ON.

E Modificar a classe .button2 para background-color: red; e usá-la no botão ON.

Parabéns! A alternativa C está correta.

Alterar o valor de background-color para red diretamente na classe .button afetará o botão ON, alterando sua cor para vermelho. Isso porque a classe .button é aplicada ao botão ON e modificar seu background-color para red (ou #FF0000, que é o código hexadecimal para vermelho), mudará a cor de fundo desse botão.

Sistema de monitoramento de temperatura usando MQTT

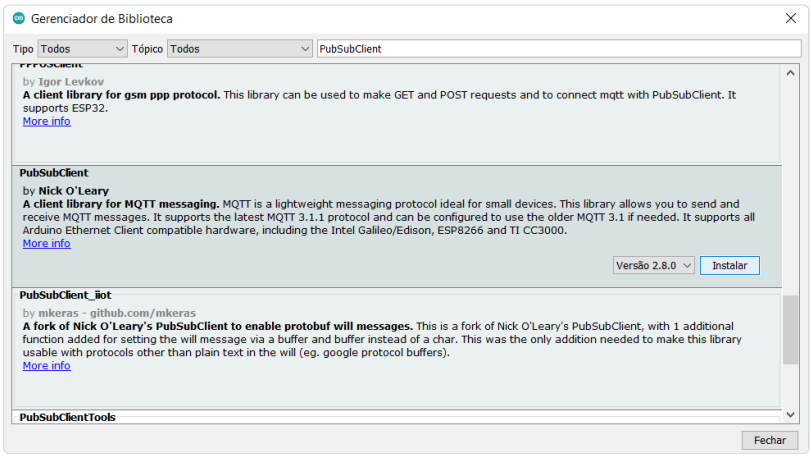
Aqui, vamos acompanhar o desenvolvimento de um sistema de monitoramento de temperatura utilizando o ESP32 e o protocolo MQTT; uma solução robusta e flexível para aplicações de internet das coisas (IoT). O ESP32 será empregado para coletar dados de temperatura através do sensor BMP180. Esses dados serão, então, transmitidos via WI-FI a um broker MQTT gratuito, facilitando o monitoramento remoto em tempo real.

Confira neste vídeo o desenvolvimento de um sistema IoT para monitoramento de temperatura usando o ESP32 e o protocolo MQTT.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

Roteiro de prática

1. Preparação: garanta que todas as bibliotecas necessárias para o wi-fi, o MQTT e o sensor BMP180, estejam instaladas pelo gerenciador de bibliotecas. Com isso, você pode adicionar as bibliotecas , e no topo do seu código. Observe a tela do gerenciador na instalação da biblioteca PubSubClient.



Instalação da biblioteca PubSubClient.

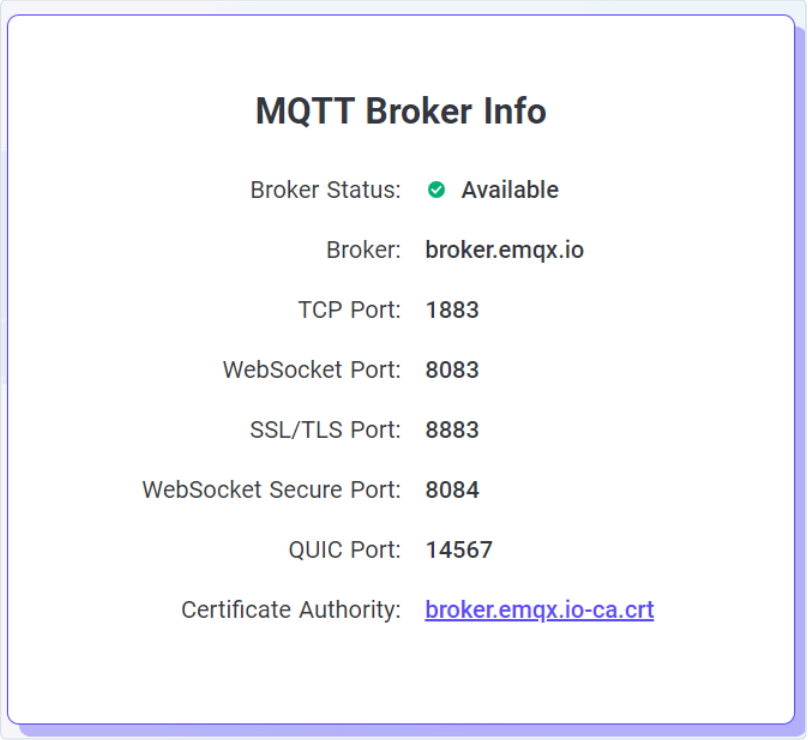
---

2. Montagem do circuito: conecte o sensor BMP180 ao ESP32 usando a comunicação I²C. Ligue os pinos VCC e GND para alimentação, e os

pinos SCL e SDA para os dados com os pinos correspondentes na placa ESP32 e no sensor BMP180.

3. Configuração do software:

- Wi-Fi: no seu código, configure o ESP32 para se conectar à sua rede Wi-Fi, incluindo as credenciais SSID e senha.
- MQTT: defina as informações do broker MQTT, como o endereço do servidor e a porta. Para fins de teste, pode-se usar um broker público, por exemplo, o EMQX. Procure por “emqx mqtt public” no buscador e use as informações sugeridas, conforme mostrado na próxima imagem.
- BMP180: utilize a biblioteca do sensor para inicializar o BMP180 e prepare o código para ler a temperatura.



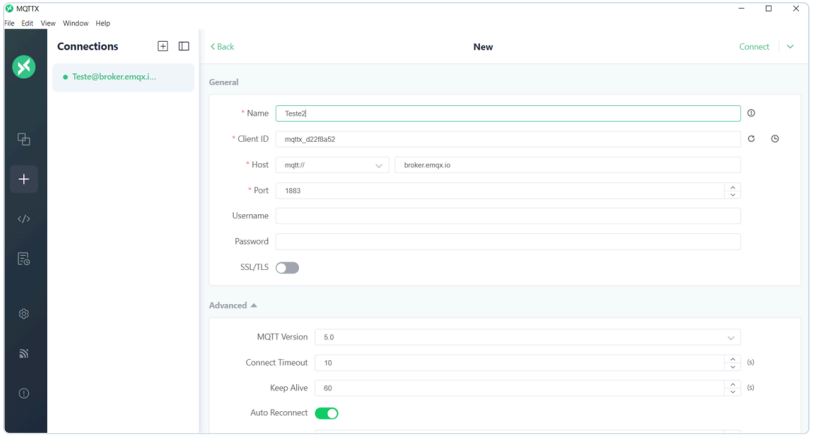
Tela capturada da página do broker público EMQX.

---

4. Implemente a função que lê a temperatura do BMP180.

5. Configure o ESP32 para publicar as leituras de temperatura em um tópico específico do MQTT.

6. Teste e monitoramento: utilize o monitor serial do Arduino IDE para verificar o fluxo de dados e assegurar que o ESP32 está lendo e enviando os dados de temperatura corretamente. Para visualizar os dados enviados, use um cliente MQTT ou um dashboard IoT que possa se inscrever no tópico escolhido (emqx/esp32). Uma opção é o MQTTX.APP cuja tela é mostrada na imagem a seguir.



Tela do programa MQTTX.APP para visualizar os dados enviados.

Veja o código do programa para testes.

c

Atividade 4

Você está desenvolvendo um sistema de monitoramento de temperatura usando um ESP32 e um sensor BMP180. O sistema, atualmente, lê a temperatura ambiente e a envia a um broker MQTT a cada intervalo de tempo. Agora, deseja-se adicionar uma funcionalidade para que o sistema envie uma mensagem de alerta para o mesmo tópico no broker sempre que a temperatura ultrapassar 28 graus Celsius. Acrescente o código necessário ao original que você testou e experimente.

Qual das seguintes adições ao código realizará essa tarefa corretamente?

SHELL SCRIPT

A

SHELL SCRIPT

SHELL SCRIPT

B

SHELL SCRIPT

SHELL SCRIPT

C

SHELL SCRIPT

SHELL SCRIPT

D

SHELL SCRIPT

SHELL SCRIPT

E

SHELL SCRIPT

Parabéns! A alternativa A está correta.

A alternativa implementa, adequadamente, a lógica condicional para verificar se a temperatura lida pelo sensor BMP180 é maior do que 28 graus Celsius. Se a condição for verdadeira, a função client.publish é chamada para enviar uma mensagem de alerta para o tópico específico no broker MQTT (emqx/esp32). As demais alternativas apresentam falhas em sua lógica ou aplicação.

O que você aprendeu neste conteúdo?

- Aprendizado de comunicação serial para integração de dispositivos.
- Criação de sistemas que gerenciam e controlam ativamente dispositivos.
- Técnicas para coletar e processar dados de sensores para monitoramento.
- Conceitos e práticas em conexões de rede em linguagem C.
- Desenvolvimento de habilidades para transferência de dados pela internet.

- Representação e análise de dados adquiridos por meio de interfaces seriais.
- Instalação e configuração de DevC++ e IDE Arduino para programação eficiente.
- MQTT para comunicação eficiente entre dispositivos em redes IoT.
- Técnicas para criar servidores web e clientes que se comunicam por HTTP.

Explore +

Leia o artigo **Protótipo de baixo custo de analisador RS232 síncrono e protocolo HDLC para linhas de radiodeterminação** e saiba como o autor, Derik Adan do Nascimento Maia, aborda o desenvolvimento de um dispositivo eletrônico de baixo custo para avaliação de sinais RS232. Confira no site do Repositório Institucional UEA.

No artigo **Desenvolvimento de um sistema remoto de aquisição de dados de tensão e corrente utilizando sockets**, a autora, Fernanda Fernandes de Oliveira, delineia a criação de um sistema de monitoramento remoto do consumo de energia, enfatizando a integração de dispositivos por meio da internet das coisas (IoT) e outros protocolos de comunicação. Para saber mais, acesse o site Repositório Digital do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba.

Confira o artigo **Aplicação do protocolo MQTT para integração de dispositivos IIOT a sistemas de automação industrial em um ambiente de testes**. Esse artigo apresenta a integração de dispositivos com tecnologia Industrial Internet of Things (IIoT) em sistemas de automação industrial, utilizando o protocolo MQTT para facilitar testes de integração em um software SCADA. Faça a leitura acessando a Revista Brasileira de Mecatrônica.

Referências

DONAHOO, M. J.; CALVERT, K. L. **TCP/IP sockets in C:** a practical guide for programmers. 2nd ed. Massachusetts: Morgan Kaufmann, 2009.

HILLAR, G. C. **MQTT essentials:** a lightweight IoT protocol. Birmingham: Packt Publishing, 2017.

MANZANO, J. A. N. G. **Linguagem C acompanhada de uma xícara de café.** São Paulo: Érica, 2015.

OLIVEIRA, C. L. V.; ZANETTI, H. A. P. **Arduino descomplicado:** como elaborar projetos de eletrônica. São Paulo: Érica, 2015.

ONER, V. O. **Developing IoT projects with ESP32.** 2nd ed. Birmingham: Packt Publishing, 2023.

THAI, T. L. **Learning DCOM.** Massachusetts: O'Reilly Media, 1999.