



# Internet das Coisas em um Mundo Conectado


## UNIDADE 06

### Simulação e Conectividade em Projetos IoT

Nesta unidade, daremos início ao desenvolvimento prático de uma solução de Internet das Coisas (IoT), utilizando duas ferramentas: o simulador do ESP32, e um cliente MQTT para comunicação.

A proposta é tornar acessível o aprendizado sobre dispositivos embarcados e como eles se comunicam em ambientes conectados. Você irá compreender como um microcontrolador pode interagir com sensores e atuar em tempo real, trocando dados via rede com eficiência e baixo custo computacional, característica fundamental em projetos de IoT. A simulação permite validar essas trocas de dados antes da implementação física, reforçando a importância do protocolo MQTT em sistemas distribuídos.

Anteriormente utilizamos os navegadores como dispositivos para acessar o Broker MQTT da plataforma HiveMQ, conforme demonstra a figura a seguir.

#ParaTodosVerem 

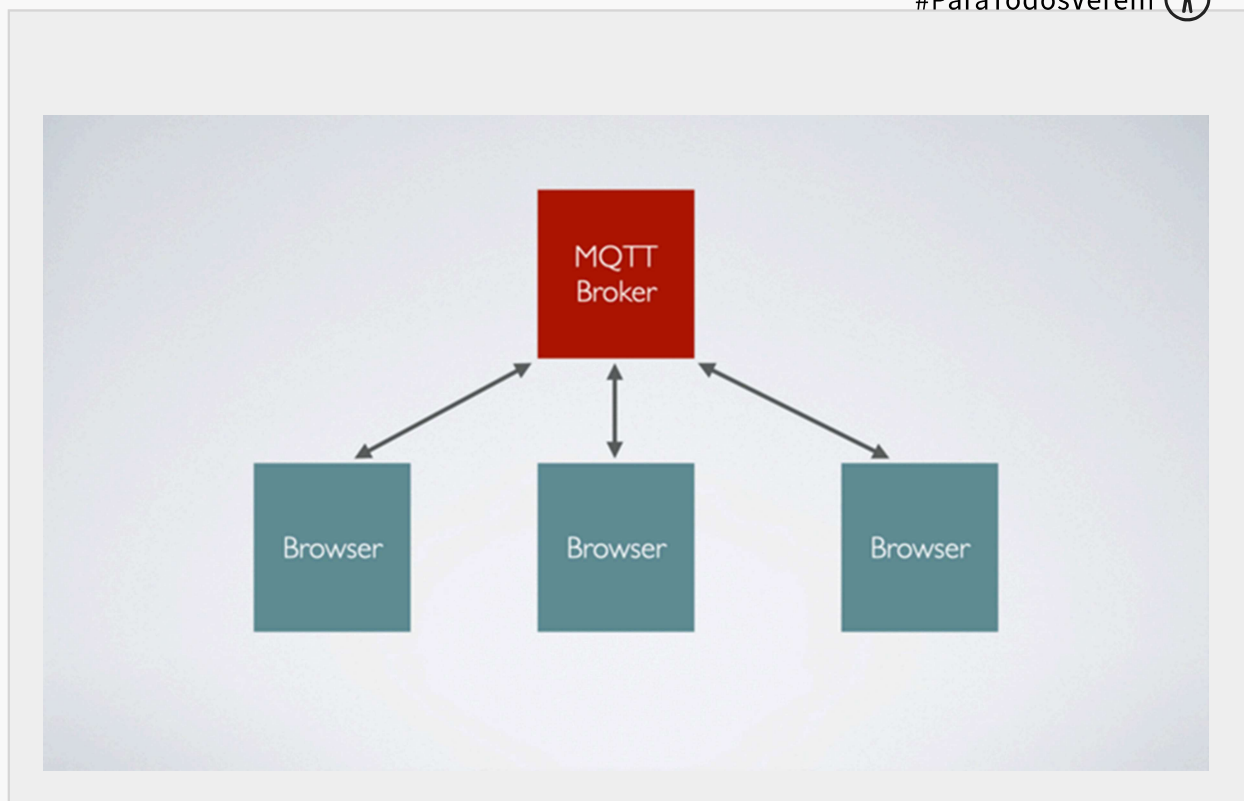


Figura 1: Visão geral da conexão entre o broker MQTT e as possíveis visualizações pelo navegador. O autor

Usaremos um navegador como ferramenta de visualização dos dados, mas nosso foco será o desenvolvimento de um dispositivo ESP32 simulado, em Python que ficará enviando a temperatura, utilizando o protocolo MQTT, sendo possível inserir qualquer outro sensor ou atuador.

A seguir nas videoaulas, você será orientado a montar um circuito virtual com o ESP32 e ao menos um sensor ou atuador – como um LED, botão ou sensor de temperatura DHT22 – e programar a troca de dados com um broker MQTT. Para isso, utilizaremos a linguagem MicroPython e um simulador em nuvem.

## **Videoaula 1: Introdução ao Wokwi**

---

Neste vídeo, você conhecerá o simulador Wokwi e aprenderá a criar seu primeiro projeto com o ESP32, adicionando sensores e atuadores e programando em MicroPython.

## **Videoaula 2: Integração Wokwi + MQTTX**

Neste vídeo, veremos como integrar o ESP32 simulado no Wokwi com o cliente MQTTX, enviando e recebendo dados em tempo real via protocolo MQTT.



## MUNDO DO TRABALHO

Vamos ver na prática como o que vimos na videoaula pode ser utilizado no mundo do trabalho.

O setor agrícola tem buscado, cada vez mais, soluções tecnológicas para aumentar a produtividade e reduzir perdas. Um dos grandes desafios é manter o controle adequado de temperatura dentro de estufas, já que variações podem comprometer o desenvolvimento das plantas.

Empresas do agronegócio estão recorrendo a dispositivos IoT (Internet of Things) para criar sistemas de monitoramento e automação que possibilitam a coleta de dados em tempo real e a atuação imediata sobre o ambiente.

Agora imagine que uma startup de tecnologia agrícola convidou você para desenvolver um **protótipo de baixo custo e rápido desenvolvimento** para monitorar e controlar a temperatura em estufas.

Para o projeto, a startup não fornecerá componentes físicos inicialmente, mas propôs que você use **ferramentas de simulação**. As ferramentas escolhidas foram:

- **Wokwi**, para simular um **ESP32** com sensores e atuadores, programado em **MicroPython**.
- **MQTTX**, para validar a comunicação em tempo real entre o dispositivo simulado e um **broker MQTT**, testando envio e recebimento de dados.

Seu desafio para criar o protótipo é:

1. **Leia a temperatura** através de um sensor simulado no Wokwi.
2. **Ative um ventilador (atuador)** quando a temperatura ultrapassar um limite pré-definido.
3. **Envie os dados de temperatura em tempo real para o MQTTX** via protocolo MQTT.
4. **Receba comandos externos pelo MQTTX** para ligar ou desligar o ventilador manualmente, independentemente da leitura do sensor.

Agora, propomos que você faça uma pausa e anote como poderia resolver a estes desafios. Depois, consulte as respostas.



## Respostas do desafio



### Ler a temperatura no ESP32 via Wokwi

No Wokwi, adicione um ESP32 e um sensor de temperatura (DHT22 ou DHT11). Depois, no código em MicroPython, use a biblioteca dht para capturar os valores de temperatura e umidade.

Exemplo simplificado:

</>

```
1 import dht
2 import machine
3 import time
4
5 sensor = dht.DHT22(machine.Pin(15))
6
7 while True:
8     sensor.measure()
9     temp = sensor.temperature()
10    print("Temperatura:", temp, "°C")
11    time.sleep(2)
```

### Ativar um ventilador (atuador) quando ultrapassar limite

No Wokwi, adicione um **LED** ou um **motor DC** para simular o ventilador. Depois, no código, defina um limite (ex.: 28 °C). Se a leitura for maior, ativa o atuador.

Exemplo simplificado:

</>

```
1 ventilador = machine.Pin(2, machine.Pin.OUT)
2
3 if temp > 28:
4     ventilador.on()
5     print("Ventilador LIGADO")
6 else:
7     ventilador.off()
8     print("Ventilador DESLIGADO")
```

## Enviar dados em tempo real para o MQTTX

Configure o **broker MQTT** (pode ser o público [test.mosquitto.org](https://test.mosquitto.org) ou um local). Depois, No código do ESP32 simulado, use a biblioteca `umqtt.simple` para publicar dados.

Exemplo simplificado:

</>

```
1 from umqtt.simple import MQTTClient
2
3 client = MQTTClient("esp32", "test.mosquitto.org")
4 client.connect()
5
6 client.publish(b"estufa/temperatura", str(temp))
```

O próximo passo é no MQTTX. Conecte-se ao mesmo broker. Depois, inscreva-se no tópico `estufa/temperatura`. E por fim, verifique se os valores enviados pelo ESP32 aparecem no cliente.

## Receber comandos externos pelo MQTTX

Programar o ESP32 para **se inscrever em um tópico de controle**, ex.: `estufa/ventilador`. Depois, quando receber a mensagem "ON" ou "OFF", altera o estado do atuador.

Exemplo simplificado:

</>

```
1 def sub_cb(topic, msg):
2     if topic == b"estufa/ventilador":
3         if msg == b"ON":
4             ventilador.on()
5             print("Comando remoto: LIGADO")
6         elif msg == b"OFF":
7             ventilador.off()
8             print("Comando remoto: DESLIGADO")
9
10 client.set_callback(sub_cb)
11 client.subscribe(b"estufa/ventilador")
```

O próximo passo é no MQTTX. Publique ON ou OFF no tópico estufa/ventilador e o atuador deve responder de acordo.



## REFLEXÃO

Depois de analisar o case e verificar as respostas, responda as questões.

- Quais vantagens e limitações você enxerga no uso de simuladores em comparação com dispositivos físicos?
- Explique como o protocolo MQTT facilita a comunicação entre o ESP32 e o MQTTX nesse projeto.
- Se esse protótipo fosse implantado em uma estufa real, quais outros sensores ou funcionalidades poderiam ser adicionados para tornar o sistema mais eficiente?
- Como o uso desse tipo de tecnologia pode impactar a gestão agrícola e a sustentabilidade no agronegócio?

Após fazer suas anotações, confira as sugestões de respostas que preparamos para você.



## Sugestões de respostas



### Como o uso do Wokwi pode auxiliar no aprendizado prático sem a necessidade imediata de hardware físico?

O Wokwi permite simular o ESP32, sensores e atuadores em um ambiente virtual, oferecendo uma experiência próxima ao mundo real. Isso elimina a barreira de custos e disponibilidade de componentes, facilitando que os estudantes testem ideias, errem e ajustem sem medo de danificar equipamentos. Assim, o foco fica no entendimento da lógica, da programação e da integração entre dispositivos, criando uma base sólida antes do contato com o hardware físico.

### Quais vantagens o protocolo MQTT oferece na comunicação entre dispositivos IoT nesse cenário?

O MQTT é um protocolo leve, eficiente e projetado para dispositivos com recursos limitados, como o ESP32. No contexto da estufa inteligente, ele permite:

- Comunicação **em tempo real** entre sensores e usuários.
- Organização dos dados em **tópicos**, o que facilita a escalabilidade do sistema.
- **Baixo consumo de rede**, ideal para aplicações de IoT.
- Facilidade para integrar diferentes aplicações e plataformas, como o MQTTX, dashboards ou serviços em nuvem.

### De que forma esse protótipo pode ser expandido para aplicações reais em estufas inteligentes?

O protótipo pode ser ampliado em várias direções:

- **Adicionar novos sensores**, como de luminosidade, umidade do solo e CO<sub>2</sub>, para enriquecer o monitoramento.
- **Automatizar outros atuadores**, como sistemas de irrigação ou cortinas de sombreamento.
- **Integrar com serviços em nuvem**, para armazenar dados históricos e gerar relatórios.



- **Implementar inteligência artificial**, permitindo prever necessidades da planta e otimizar os recursos de forma autônoma.
- **Acesso remoto via aplicativo**, para que o agricultor gerencie sua estufa de qualquer lugar.

**Quais desafios de segurança da informação devem ser considerados em uma aplicação real como essa?**

Em aplicações reais, a segurança é fundamental. Alguns desafios incluem:

- **Proteção do broker MQTT**, evitando acesso não autorizado aos dados e comandos.
- **Criptografia na transmissão (TLS/SSL)** para impedir interceptação de informações.
- **Controle de autenticação e permissões**, garantindo que apenas usuários autorizados enviem comandos.
- **Resiliência contra ataques de negação de serviço (DoS)**, que poderiam paralisar o sistema.
- **Atualização e manutenção dos dispositivos**, para corrigir falhas e vulnerabilidades que possam surgir.



## CURIOSIDADE

### Você Sabia?

Que o **Wokwi** não é apenas um simulador visual: ele executa o código em um **interpretador real de MicroPython ou Arduino C++**, permitindo depuração e testes de bibliotecas de forma quase idêntica ao hardware físico?

Que o **ESP32 simulado no Wokwi** suporta **GPIOs, comunicação serial, I<sup>2</sup>C e SPI**, possibilitando a integração virtual de sensores como DHT22, displays OLED e até motores de passo?

Que diferente de outros simuladores, o Wokwi permite **integração direta com brokers MQTT**, tornando viável testar arquiteturas IoT antes da implementação em dispositivos reais?

Que o protocolo **MQTT (Message Queuing Telemetry Transport)** é leve, baseado em **publicação/assinatura (publish/subscribe)**, e roda sobre TCP/IP? Isso garante baixo consumo de banda e torna-o ideal para IoT em redes instáveis.

Que o **MQTTX** atua como cliente gráfico, mas também suporta **QoS (Quality of Service)**, **retenção de mensagens** e **persistência**, permitindo que o estudante experimente diferentes cenários de entrega de dados?

Que ao conectar o **Wokwi** com o **MQTTX**, você pode simular **fluxos completos de IoT**, como um sensor publicando dados para um broker em nuvem (ex: Mosquitto), enquanto múltiplos clientes (MQTTX, dashboards, outros ESP32) recebem e processam as informações?

## | Ligando conceitos

Nesta unidade, nosso foco foi em compreender como funciona a arquitetura básica de um sistema IoT embarcado. Aprendemos o passo a passo, a simular circuitos utilizando o ESP32 no ambiente online Wokwi, que nos permite trabalhar com sensores e atuadores sem precisar de um protótipo físico. Também vimos como configurar a comunicação MQTT para que o ESP32 envie e receba dados, utilizando o cliente gráfico MQTTX. Colocamos tudo em prática, usando o Wokwi como simulador, o MQTTX para gerenciar as mensagens trocadas pelo protocolo MQTT e o MicroPython como linguagem de programação do ESP32 no ambiente simulado.



