SENAI — CentroWEG · Jaraguá do Sul, SC

Do Sensor à Nuvem: Telemetria Industrial em Tempo Real

Uma jornada técnica com ESP32, MQTT/TLS, PostgreSQL e Grafana

Ryhan Gabriel Schutz

Curso Técnico em CiberSistemas para Automação

Unidade Curricular: Programação para Coleta de Dados em Automação

Docente: Lucas Sousa dos Santos

Turma: T TCPA 2025/1 INT1

Jaraguá do Sul, SC — 24 de fevereiro de 2026

Este documento adota a fonte Helvetica em referência à tradição tipográfica europeia — especialmente à escola suíça de design e aos artigos técnicos franceses e alemães — pela clareza estrutural que proporciona em documentação de engenharia.

Stack Tecnológico

ESP32

HiveMQ

PostgreSQL

Neon

Grafana

Java 17

Maven

GitHub

Wokwi

WEG

Todas as ferramentas acima foram utilizadas na construção deste projeto.
Neon e Grafana Cloud foram descobertas durante o próprio desenvolvimento.

Sumário

1 Introdução

Motores elétricos estão no centro da produção industrial. Bombas, compressores, ventiladores, esteiras — todos dependem de variáveis como temperatura, vibração e corrente para operar com segurança e eficiência. Monitorar essas variáveis em tempo real deixou de ser diferencial e tornou-se requisito.

Essa percepção não vem apenas da teoria. Em painéis de automação modernos, é comum encontrar inversores de frequência da série CFW da WEG, soft-starters SSW, CLPs e IHMs integradas operando em conjunto, onde o monitoramento remoto já é realidade consolidada. Ferramentas como o **WEG WEGScan** — dispositivo com conectividade Wi-Fi que permite acompanhar parâmetros de inversores em tempo real — ilustram bem esse movimento da indústria: dados saindo do painel e chegando onde o operador estiver.

Este trabalho nasce desse contexto e vai além dele. O objetivo inicial era construir um protótipo de telemetria com ESP32, capaz de coletar temperatura, vibração e corrente e transmiti-los via MQTT a um sistema receptor em Java. O que se seguiu, por iniciativa própria, foi a construção de uma arquitetura completa de supervisão em nuvem: banco de dados PostgreSQL hospedado no Neon, dados gravados em tempo real pelo backend Java, e um dashboard temporal no Grafana Cloud — tudo integrado, acessível de qualquer lugar.

Nota do autor

Este documento é ao mesmo tempo relatório técnico e registro de aprendizado. Algumas ferramentas utilizadas aqui — Neon, Grafana Cloud, GitHub Codespaces — foram descobertas no próprio processo de desenvolvimento deste projeto. A ideia é que quem leia reconheça o caminho, não apenas o destino.

2 Arquitetura do Sistema

O sistema é estruturado em quatro camadas funcionais, cada uma com responsabilidade bem definida:

```
ESP32 → MQTT/TLS (8883) → Java Backend → PostgreSQL/Neon → Grafana Cloud
```

2.1 Visão geral das camadas

Camada de coleta — O ESP32 lê os sensores a cada dois segundos e publica o payload no broker HiveMQ Cloud via MQTT com criptografia TLS na porta 8883. O formato da mensagem é uma string simples separada por vírgulas: `"temperatura,vibracao,corrente`

Camada de transporte — O protocolo MQTT opera sobre canal seguro TLS, com autenticação por credenciais no broker HiveMQ Cloud. Isso garante que os dados de processo não trafeguem em texto puro pela rede — uma decisão consciente que vai além do escopo mínimo exigido.

Camada de persistência — O backend Java, ao receber cada mensagem, realiza o parse do payload e executa um `INSERT` na tabela `leituras` do banco PostgreSQL hospedado no Neon. Cada registro carrega o valor dos três sensores e o timestamp automático da chegada.

Camada de visualização — O Grafana Cloud conecta diretamente ao banco Neon via driver PostgreSQL e plota as séries temporais de temperatura, vibração e corrente em um dashboard interativo e atualizado em tempo real.

2.2 Por que essa arquitetura?

A escolha por serviços em nuvem foi motivada pelo ambiente de desenvolvimento disponível: um PC institucional com restrições de instalação e um Codespace do GitHub como ambiente de execução do Java. O PostgreSQL local ficou inviável — o Codespace não acessa `localhost` da máquina física. A solução foi o **Neon**, um PostgreSQL serverless gratuito que resolve o problema com elegância: banco em nuvem, acessível de qualquer origem, com suporte nativo a SSL e latência mínima.

O **Grafana Cloud**, por sua vez, eliminou a necessidade de instalação local, oferecendo um ambiente de dashboard profissional acessível pelo browser — o mesmo tipo de ferramenta utilizado em sistemas SCADA industriais como o Ignition e o FactoryTalk Historian.

3 Hardware

O circuito foi montado no simulador Wokwi utilizando o ESP32-DevKit-V1. A Tabela 1 apresenta a pinagem adotada.

Tabela 1 – Pinagem do circuito — ESP32-DevKit-V1

Componente	GPIO	Tipo	Função
DHT22	15	Digital I/O	Temperatura (°C)
Potenciômetro 1	34	ADC 12-bit	Vibração simulada (0–100 mm/s)
Potenciômetro 2	35	ADC 12-bit	Corrente simulada (0–50 A)
LCD 16x2 I2C	SDA→21/SCL→22	I2C	Display local de monitoramento
LED Verde	12	Digital OUT	Indicador: envio MQTT ativo

3.1 Circuito simulado no Wokwi

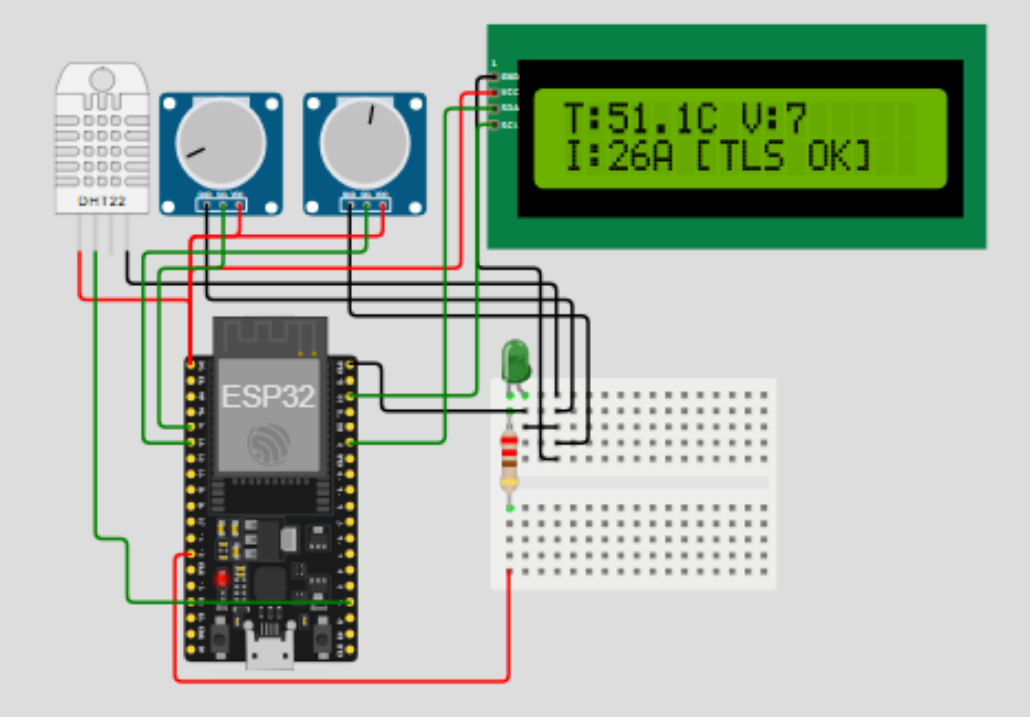

Figura 1 – Protótipo simulado — ESP32 com DHT22, potenciômetros, LCD I2C e LED verde

O uso do Wokwi como ambiente de simulação permite validar o firmware e o comportamento dos sensores sem necessidade de hardware físico, acelerando o ciclo de desenvolvimento e facilitando a documentação visual do circuito.

4 Desenvolvimento de Software

4.1 Firmware C++ — ESP32

O firmware foi desenvolvido em C++ para o framework Arduino. A lógica central compreende conexão segura à rede, leitura e escalonamento dos sensores, e publicação via MQTT/TLS.

Escalonamento dos sinais analógicos

O ADC do ESP32 opera com resolução de 12 bits, produzindo valores entre 0 e 4095. A função `map()` realiza o escalonamento linear para grandezas físicas reais:

```
// Vibra  o: 0 a 100 mm/s (faixa t pica para motores de indu  o)
int vibra = map(analogRead(PIN_VIBRA), 0, 4095, 0, 100);

// Corrente: 0 a 50 A (faixa t pica para motores de m dia pot ncia)
int corrente = map(analogRead(PIN_CORR), 0, 4095, 0, 50);
```

Listing 1 – Escalonamento ADC — map() para grandezas físicas

Esse processo é análogo ao tratamento de sinais 4–20 mA em sistemas reais, onde o condicionador converte a grandeza física para tensão legível pelo ADC, e o firmware realiza a operação inversa.

Canal seguro TLS

```
// TLS sem valida  o de certificado CA (compat vel com Wokwi)
// Em campo: espClient.setCACert(ca_cert) com certificado raiz
espClient.setInsecure();
client.setServer(mqtt_server, 8883);

// Reconex o autom tica com autentica  o por credenciais
if (client.connect(client_id, mqtt_user, mqtt_password)) {
    digitalWrite(LED_VERDE, HIGH);
}

// Payload: string simples separada por v rgula
String payload = String(temp) + "," + String(vibra) + "," + String(
    corrente);
client.publish("senai/ryhan/motor/dados", payload.c_str());
```

Listing 2 – Conexão TLS e publicação MQTT

4.2 Backend Java — Subscriber e Persistência

O backend foi estruturado como projeto Maven com duas dependências: `org.eclipse.paho.` para a comunicação MQTT e `org.postgresql` para a gravação no banco de dados.

Conexão ao banco Neon

```
static final String DB_URL  =
    "jdbc:postgresql://ep-delicate-cell-ac9rn1t6-pooler" +
    ".sa-east-1.aws.neon.tech/neondb?sslmode=require";

Connection conn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASS);
PreparedStatement stmt = conn.prepareStatement(
    "INSERT INTO leituras (temperatura, vibracao, corrente) VALUES (?,
       ?, ?)"
);
```

Listing 3 – Conexão JDBC ao Neon PostgreSQL

Recepção e persistência em tempo real

```
client.subscribe(TOPICO, (topic, msg) -> {
    String payload = new String(msg.getPayload());
    System.out.println(
        "Dados de Telemetria Coletados com Sucesso: " + payload);

    String[] v = payload.split(",");
    if (v.length == 3) {
        stmt.setDouble(1, Double.parseDouble(v[0].trim()));
        stmt.setInt(2, Integer.parseInt(v[1].trim()));
        stmt.setInt(3, Integer.parseInt(v[2].trim()));
        stmt.executeUpdate();
    }
});
```

Listing 4 – Subscriber MQTT com gravação no banco

4.3 Banco de Dados — Neon PostgreSQL

O Neon é um serviço PostgreSQL serverless que opera inteiramente em nuvem. A escolha foi motivada pela impossibilidade de acessar um banco local a partir do GitHub Codespaces — e se revelou uma descoberta significativa: banco relacional com toda a robustez do PostgreSQL, sem instalação, com suporte nativo a SSL e acessível de qualquer origem.

A estrutura da tabela é simples e eficiente para dados de série temporal:

```
CREATE TABLE leituras (
    id          SERIAL PRIMARY KEY,
    timestamp   TIMESTAMPTZ DEFAULT NOW(),
    temperatura NUMERIC(5,2),
    vibracao    INTEGER,
    corrente    INTEGER
);

CREATE INDEX idx_timestamp ON leituras (timestamp DESC);
```

Listing 5 – Estrutura da tabela de leituras

O índice no campo `timestamp` é essencial para que o Grafana execute queries de janela temporal com performance adequada, mesmo com grande volume de registros.

4.4 Dashboard — Grafana Cloud

O Grafana Cloud é a versão hospedada do Grafana — plataforma de observabilidade amplamente utilizada em ambientes industriais e de TI para visualização de métricas em tempo real. Neste projeto, conecta diretamente ao banco Neon via driver PostgreSQL nativo e plota as três séries temporais em um único painel.

A query utilizada no painel principal é:

```
SELECT
  timestamp AS "time",
  temperatura AS "Temperatura ( C )",
  vibracao    AS "Vibra  o (mm/s)",
  corrente    AS "Corrente (A)"
FROM leituras
WHERE timestamp >= $__timeFrom() AND timestamp <= $__timeTo()
ORDER BY timestamp ASC;
```

Listing 6 – Query Grafana — séries temporais dos três sensores

As variáveis `$__timeFrom()` e `$__timeTo()` são injetadas automaticamente pelo Grafana conforme o intervalo de tempo selecionado no dashboard, permitindo zoom e análise histórica sem alterar a query.

Dashboard em operação

Figura 2 – Dashboard Grafana Cloud — Temperatura, Vibração e Corrente em tempo real

5 Testes e Validação

Os testes foram conduzidos com o ESP32 simulado no Wokwi, o backend Java em execução no GitHub Codespaces e o banco Neon recebendo as gravações em tempo real. A Tabela 2 registra os resultados.

Tabela 2 – Ficha de verificação de testes

ID	Teste	Procedimento	Resultado
01	Conectividade Wi-Fi	Iniciar ESP32 no Wokwi	**Aprovado**
02	Display LCD	Verificar temperatura no display local	**Aprovado**
03	LED Verde	Confirmar acionamento ao conectar MQTT	**Aprovado**
04	Telemetria MQTT	Girar potenciômetros, observar console Java	**Aprovado**
05	Integridade do payload	Verificar formato `temp,vibra,corrente`	**Aprovado**
06	Gravação no banco	Verificar INSERT no Neon após cada publicação	**Aprovado**
07	Dashboard Grafana	Confirmar plotagem das três séries temporais	**Aprovado**
08	Atualização em tempo real	Girar potenciômetros e observar gráfico mudar	**Aprovado**

5.1 Evidências

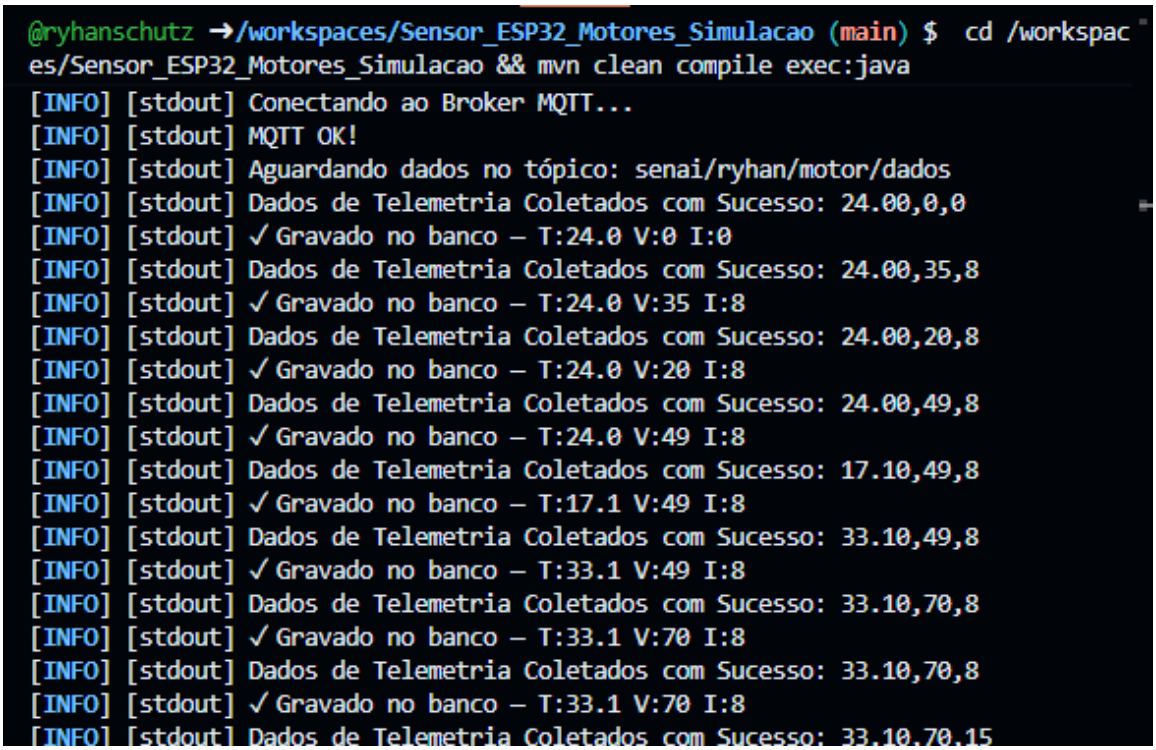

Figura 3 – Console Java — recepção MQTT e confirmação de gravação no Neon PostgreSQL

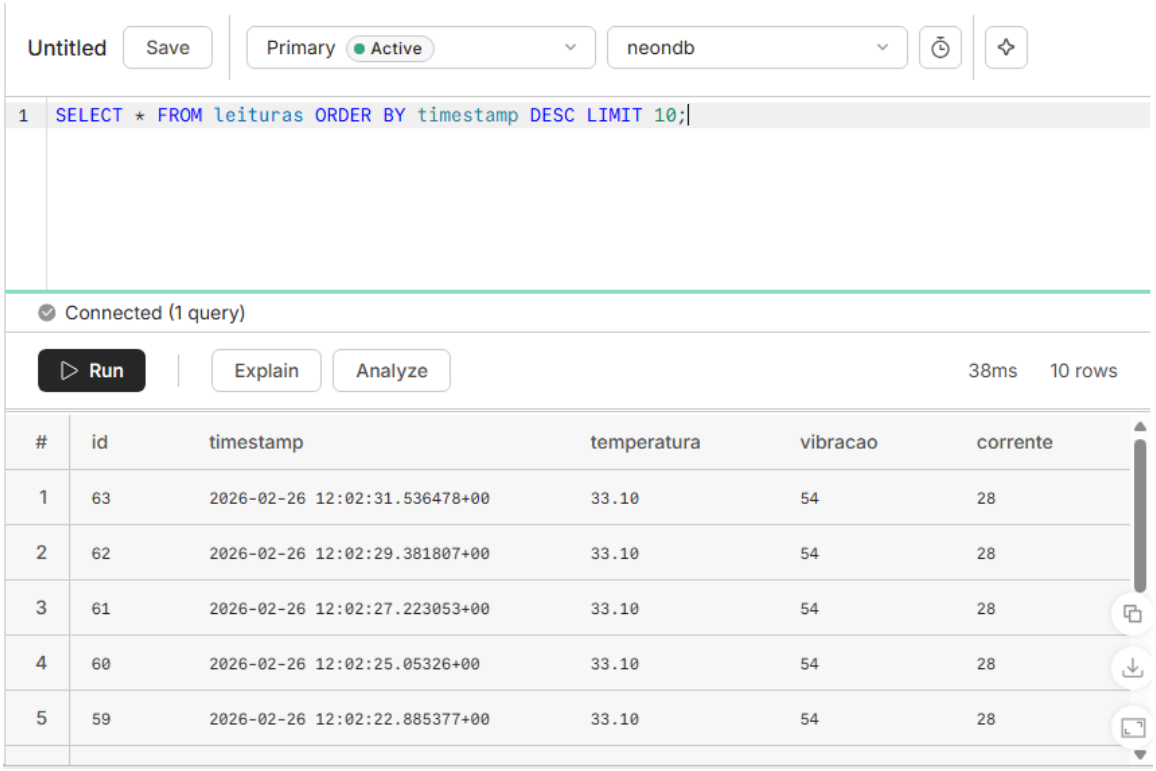

Figura 4 – Neon PostgreSQL — tabela `leituras` com registros de telemetria

6 Planejamento e Processo

O projeto foi executado em uma sessão contínua de desenvolvimento, com duração aproximada de 240 minutos para o escopo original — e horas adicionais dedicadas à extensão com banco de dados e dashboard, realizadas por iniciativa própria.

Tabela 3 – Cronograma de execução — escopo original

Etapa	Estimado	Realizado (aprox.)
Montagem do hardware no Wokwi	40 min	40 min
Programação do firmware C++	60 min	70 min
Configuração do backend Java	40 min	45 min
Testes de comunicação e ajustes	40 min	35 min
Redação do relatório técnico	60 min	50 min
Total — escopo original	**240 min**	**240 min**
Integração com Neon PostgreSQL	—	30 min
Configuração do Grafana Cloud	—	20 min
Dashboard e queries	—	20 min
Total — projeto completo	—	**310 min**

O tempo adicional dedicado à programação no escopo original deveu-se ao estudo de ferramentas que, embora conhecidas conceitualmente, estavam sendo utilizadas pela primeira vez nessa configuração — em particular o `WiFiClientSecure` com TLS no Wokwi e o cliente Paho com `SSLSocketFactory` no Java.

A extensão com banco de dados e dashboard não estava prevista. Surgiu de uma pergunta natural ao ver os dados chegando no console: *e se eu guardasse isso em algum lugar?* O que veio depois foi uma sequência de descobertas — o Neon como solução para o banco em nuvem, o Grafana Cloud como ambiente de visualização profissional acessível pelo browser — que expandiram significativamente o escopo técnico e pessoal deste trabalho.

7 Reflexão Técnica e Aprendizado

O que foi além do esperado

Este projeto começou como um exercício de prova. Terminou como um sistema de supervisão em nuvem funcional, com dados reais gravados em banco PostgreSQL e visualizados em dashboard temporal — o mesmo conceito presente em historiadores de dados industriais como o Ignition e o FactoryTalk Historian.

Três descobertas merecem destaque:

Neon PostgreSQL — A existência de um banco PostgreSQL completamente gerenciado em nuvem, gratuito para projetos pequenos, com SSL nativo e compatível com qualquer cliente JDBC, foi uma surpresa genuína. Elimina a necessidade de servidor local e abre possibilidades para projetos que precisam de persistência sem infraestrutura própria.

Grafana Cloud — Conhecer o Grafana apenas pelo nome é diferente de conectar um banco, escrever uma query e ver três curvas temporais aparecerem em tempo real representando dados físicos de sensores. A ferramenta é diretamente comparável ao que se encontra em salas de controle industriais — e está disponível gratuitamente no browser.

TLS no ESP32 — Implementar criptografia de canal em um microcontrolador de $5, com autenticação por credenciais em um broker em nuvem, é algo que demonstra que segurança de dados industriais não é exclusividade de sistemas corporativos caros. Está ao alcance de qualquer projeto embarcado.

A vivência prévia com inversores CFW, soft-starters SSW e ferramentas como o WEGScan deu contexto ao que foi construído aqui. Ver os dados de temperatura, vibração e corrente chegando no dashboard do Grafana — os mesmos tipos de variáveis que se monitora em painéis industriais reais — fechou um ciclo entre o que se aprende na prática de campo e o que se desenvolve em código.

8 Conclusão

O sistema construído cumpre o que se propôs e vai além. A arquitetura completa — ESP32, MQTT/TLS, Java, Neon PostgreSQL e Grafana Cloud — é funcional, segura e acessível de qualquer dispositivo com acesso à internet.

Mais do que o resultado técnico, o processo revelou que a fronteira entre automação industrial e desenvolvimento de software é mais tênue do que parece. Os protocolos mudam — de Modbus para MQTT, de RS-485 para TLS — mas o conceito é o mesmo: coletar, transportar, armazenar e visualizar dados de processo com confiabilidade.

O próximo passo natural é a visualização tridimensional das três variáveis em um gráfico 3D interativo com Three.js — onde temperatura, vibração e corrente ocupam cada um um eixo, e o comportamento do motor ao longo do tempo se torna uma trajetória no espaço. Esse é o horizonte.

Ler, compreender e fazer. Esse princípio orientou cada etapa deste trabalho — do primeiro fio ligado no Wokwi à última linha de SQL no Grafana.

Referências

LAMB, Frank. **Automação Industrial na Prática**. Tradução de Daniel Vieira. Porto Alegre: McGraw-Hill / Bookman, 2015. (Série Tekne).

FRANCHI, Claiton Moro; CAMARGO, Valter Luís Arlindo de. **Controladores Lógicos Programáveis: Sistemas Discretos**. 2. ed. São Paulo: Érica, 2011.

CHAPMAN, Stephen J. **Fundamentos de Máquinas Elétricas**. 5. ed. Porto Alegre: McGraw-Hill / Bookman, 2013.

INTERNATIONAL ELECTROTECHNICAL COMMISSION. **IEC 61158: Industrial Communication Networks — Fieldbus Specifications**. Geneva: IEC, 2019.

NEON. **Neon — Serverless PostgreSQL**. Disponível em: <https://neon.tech>. Acesso em: 24 fev. 2026.

GRAFANA LABS. **Grafana Cloud — Observability Platform**. Disponível em: <https://grafana.com>. Acesso em: 24 fev. 2026.

HIVEMQ. **HiveMQ Cloud — MQTT Broker**. Disponível em: <https://www.hivemq.com>. Acesso em: 24 fev. 2026.

ECLIPSE FOUNDATION. **Eclipse Paho — MQTT Java Client**. Disponível em: <https://www.eclipse.org/paho>. Acesso em: 24 fev. 2026.

WOKWI. **Online ESP32 & Arduino Simulator**. Disponível em: <https://wokwi.com>. Acesso em: 24 fev. 2026.

ANTHROPIC. **Claude — Assistente de Inteligência Artificial**. Versão Sonnet 4.6. Disponível em: <https://claude.ai>. Acesso em: 24 fev. 2026. Utilizado como ferramenta de auxílio na estruturação, revisão e desenvolvimento do código-fonte e documentação técnica.