Arquitetura IoT para Vazão Hídrica com Sensor Hall e Processamento Virtualizado

Elohim Felipe Santiago da Silva
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
Email: elohim.s@escolar.ifrn.edu.br

Orientador: Dr. Ivanilson França Vieira Junior
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

Março de 2025

Resumo

Este artigo apresenta o desenvolvimento e validação de um sistema de monitoramento de fluxo hídrico baseado em Internet das Coisas (IoT), voltado à otimização do controle e gestão de recursos hídricos. A solução emprega um nó sensor composto por ESP32 e o transdutor YF-S201, cujos sinais são suavizados via Filtro de Kalman e calibrados por Regressão Linear Recursiva (RLS) para maior precisão dos dados. Os dados são transmitidos em tempo real por meio de uma API REST sobre TCP/IP, enquanto a aplicação é executada em um backend containerizado com Docker, garantindo portabilidade e escalabilidade dos serviços.

Palavras-chave: ESP32, IoT, Filtro de Kalman, Regressão Linear Recursiva, Docker, Redes de Computadores.

1 Introdução

O uso ineficiente da água tornou-se um desafio crítico em contextos industriais, comerciais e institucionais, resultando em custos operacionais elevados e pressões ambientais cada vez maiores. Segundo relatórios da ONU-Água, perdas superiores a 30% na distribuição e no consumo interno ainda são comuns em países em desenvolvimento, impactando diretamente a disponibilidade e a sustentabilidade dos recursos hídricos (ONU-ÁGUA, 2023). Monitorar, em tempo real, os pontos de consumo é, portanto, um requisito fundamental para programas de eficiência hídrica.

Avanços recentes em Internet das Coisas (IoT) - tecnologia que permite conectar dispositivos físicos à internet - e em redes de computadores têm viabilizado a instrumentação de pontos de consumo hídrico com sensores de baixo custo. Esses sensores se comunicam por meio de redes IP, possibilitando a análise distribuída via edge/fog computing(computação de borda e em névoa), que permitem o processamento local ou intermediário dos dados antes de serem enviados à nuvem. Sensores de efeito Hall, como o modelo YF-S201, representam uma solução de baixo custo e fácil integração para medição de vazão em

sistemas embarcados, sendo amplamente utilizados em aplicações IoT, porém seus sinais são suscetíveis a ruídos e não linearidades derivadas da própria geometria do medidor (Li et al., 2022). Para transformar essas leituras em dados confiáveis de supervisão, torna-se essencial o uso de técnicas de filtragem e calibração, que podem ser executadas localmente no nó — reduzindo latências e consumo de banda — ou no backend em nuvem.

Neste artigo, propomos um sistema integrado de monitoramento hídrico que combina:

- Hardware de baixo custo — microcontrolador ESP32 aliado ao sensor YF-S201; - Processamento de sinais em tempo real, empregando Filtro de Kalman para suavizar ruídos e Regressão Linear Recursiva (RLS) para calibração adaptativa; - Arquitetura de comunicação padronizada, baseada em API REST sobre TCP/IP, possibilitando comunicação direta entre dispositivos(Machine to Machine - M2M) e edge computing; - Virtualização de serviços com Docker, uma plataforma de contêineres que permite empacotar aplicações e suas dependências de forma isolada favorecendo escalabilidade, portabilidade e reprodutibilidade do ambiente de coleta, persistência dos dados e visualização em tempo real.

Um dos pontos centrais deste trabalho é a integração da modelagem matemática baseada em álgebra linear, permitindo maior precisão na conversão dos sinais do sensor em dados de vazão. Vetores e matrizes desempenham um papel essencial na filtragem dos dados e na calibração dinâmica do sistema, garantindo medições mais confiáveis.

Os resultados experimentais comprovam uma redução significativa no erro médio (35%) após a aplicação do Kalman e do RLS em comparação a métodos estáticos de calibração. Além disso, a latência ponta a ponta manteve-se abaixo de 120ms, atendendo aos requisitos típicos de aplicações de monitoramento em tempo real, onde latências abaixo de 200ms são consideradas adequadas para supervisão contínua e tomada de decisão.

O artigo está estruturado da seguinte forma:

A Seção 2 revisa trabalhos correlatos sobre medição de fluxo com IoT; A Seção 3 descreve a metodologia de hardware, software e protocolos utilizados; A Seção 4 apresenta os experimentos e análise dos resultados; A Seção 5 mostra a arquitetura dos sistema virtualizada A Seção 6 discute limitações e possíveis extensões futuras; Por fim, a Seção 7 resume as conclusões.

2 Revisão Bibliográfica

A presente seção realiza uma revisão bibliográfica com o objetivo de descrever e investigar informações relacionadas ao tema, com base em estudos acadêmicos. A análise foi restrita a artigos que abordam a multidisciplinaridade entre Eletrônica Embarcada(como microcontroladores ESP32 e sensores de efeito Hall), Internet das Coisas (IoT), Redes Wi-Fi para comunicação sem fio dos dados coletados e técnicas estatísticas aplicadas ao monitoramento hídrico.

A revisão foi conduzida na base de dados IEEE Xplore, utilizando palavras-chave como *"Hall effect water"*, *"ESP8266 water flow sensor"*, *"computers networks "*, *"monitoring water"* e *"Kalman filter"*. A busca com a primeira palavra-chave retornou 374 artigos; a segunda, 37; e a combinação das duas últimas resultou em 32 artigos. Ao todo, foram selecionados 11 artigos, considerando critérios de clareza metodológica, especificação de sensores e aplicabilidade prática. Diversos artigos foram excluídos por não detalharem o tipo de sensor ou microcontrolador utilizado, ou por apresentarem metodologia inadequada.

Observou-se que a metodologia adotada nos artigos analisados é, em sua maioria, semelhante: o monitoramento do fluxo de água é realizado por meio de um sensor de efeito Hall, cujos pulsos são lidos por um microcontrolador, que posteriormente transmite os dados para a nuvem.

Outro aspecto relevante identificado é que muitos deles não consideram a influência do diâmetro dos tubos nas leituras de vazão realizadas com o sensor YF-S201. Tubos de menor diâmetro aumentam a velocidade do fluido para uma mesma vazão volumétrica, devido à conservação de massa, o que impacta diretamente na frequência dos pulsos gerados pelo sensor. Além disso, fatores como interferência elétrica - oriunda de variações na alimentação ou ruído eletromagnético -, vibrações mecânicas e limitações do próprio transdutor podem comprometer a precisão das medições.

Estudos recentes têm explorado métodos de estimação de estados em sistemas dinâmicos com dados imperfeitos, especialmente no contexto de redes IoT distribuídas. Suryanarayanan e Heydt (15) demonstram a aplicabilidade do Filtro de Kalman Estático com modelagem baseada em processos de Gauss-Markov, permitindo obter estimativas confiáveis mesmo em presença de ruído. Em complemento, o algoritmo EKF-KRLS proposto por Kong et al. (20) combina filtragem adaptativa com aprendizado automático em tempo real, reforçando a importância de soluções híbridas e escaláveis para estimação em sistemas embarcados. Já Mota et al. (19) apresentam uma técnica baseada em aproximação recursiva mínima com normas atômicas, aplicada diretamente à realidade de sensores IoT sujeitos a perdas de pacotes e ruído estocástico - isto é, variações aleatórias nos sinais que comprometem a estabilidade das medições.

Diante das limitações observadas nos sensores e das variações causadas por fatores como diâmetro dos tubos, ruído elétrico e vibrações mecânicas, torna-se essencial realizar uma calibração adequada, levando em consideração as características do ambiente de operação, a fim de garantir medições precisas e confiáveis.

Além das limitações físicas dos sensores, a literatura propõe técnicas matemáticas para aprimorar a qualidade dos dados, como estimadores estado e filtros adaptativos aplicados em sistemas IoT.

Em (19), propõe-se uma estratégia de aproximação recursiva para lidar com dados imperfeitos em redes IoT, utilizando normas atômicas - que permitem representar sinais complexos com estruturas mínimas - e otimização convexa, técnica que garante soluções eficientes mesmo em cenários com ruído e perda de pacotes.

No estudo de (20), é apresentada uma combinação entre o Filtro de Kalman Estendido (EKF) e o Kernel Recursive Least Squares (KRLS) aplicada em sistemas embarcados com aprendizado online - ou seja, com atualização contínua dos parâmetros à medida que novos dados são recebidos.

Já (15) exploram o uso de filtros Kalman estáticos em sistemas energéticos sujeitos a fluxos não programados - variações inesperadas na demanda ou fornecimento -, utilizando Gauss-Markov, hipótese também adotada na modelagem matricial deste trabalho.

Por fim, o artigo de (21) apresenta um sistema de monitoramento hídrico inteligente com sensores de efeito Hall, Raspberry Pi e previsão de consumo com SVM (Máquinas de Vetores de Suporte) e ARIMA(Modelos Autorregressivos Integrados de Média Móvel). Embora eficaz, sua arquitetura exige infraestrutura computacional adicional e maior complexidade de implementação. Neste projeto, optou-se por uma solução mais leve de baixo

custo, com algoritmo de Filtro de Kalman embarcado diretamente no ESP32 para suavização dos sinais, enquanto a calibração adaptativa por Regressão Linear Recursiva (RLS) é executada em ambiente containerizado via Docker.

Tabela 1: Tecnologias e Componentes Utilizados em Artigos sobre Monitoramento de Água (Parte 1)

Título do Artigo	**Informações Detalhadas**
An Innovative Approach for Water Leak Detection and Monitoring in Smart Cities (2)	• Sensor de efeito Hall • RTOS (Real-Time Operating System) • RTX Kernel • ARM Cortex-M3 Board • YF-S201 Flow Sensors • Water Leakage Detection (WLD) Cables • Mobile Application
An Intelligent Water Consumption Prediction System Based on Internet of Things (1)	• IoT Technology • Arduino Mega 2560 • ESP8266 Wi-Fi Module • Ultrasonic Sensor (HC-SR04) • Flow Sensor • ThingSpeak Cloud Platform • Mobile Application
Electronically Controlled Water Flow Restrictor to Limit the Domestic Wastage of Water (23)	• Sensor de efeito Hall • Sensor de Medidor de Fluxo (YFS201) • Microcontrolador Arduino (Leonardo) • Válvula Eletromecânica (Solenoide) • Ponte H • Bluetooth • Display LCD

Tabela 2: Tecnologias e Componentes Utilizados em Artigos sobre Monitoramento de Água (Parte 2)

Título do Artigo	**Informações Detalhadas**
Arduino-Based Smart Irrigation Using Water Flow Sensor, Soil Moisture Sensor, Temperature Sensor and ESP8266 WiFi Module (24)	• Sensor de efeito Hall • Sensor de umidade do solo • Sensor de temperatura (DS18B20) • Módulo Wi-Fi ESP8266
Smart Irrigation Control System Using Wireless Sensor Network Via Internet-of-Things (3)	• ESP8266 WEMOS Mini D1 • WEMOS D1 Mini Battery Shield • DHT22 Sensor • Sensor de Umidade do Solo • Sensor de Fluxo de Água • Válvula Solenoide Rain Bird 100-DV • Módulo de Relé de 1 Canal • Bateria Li-ion 3.7V @ 2200mAh • Router Wi-Fi • Placa de Circuito Universal
Water Leakage Detection based on Automatic Meter Reading (4)	• Raspberry Pi 4 • Sensor de Fluxo de Água YF-S201 • Python e Node-RED • Banco de Dados MySQL • Tesseract OCR • Plataforma de Nuvem e Internet

Tabela 3: Contribuições Teóricas para Monitoramento e Filtragem

Artigo	**Tecnologias**	**Aplicação**	**Contribuições / Limitações**
Recursive Approximation	Normas atômicas, otimização convexa	IoT com ruído e perda de pacotes	Modelagem mínima; exige alto domínio técnico
Extended Kalman + KRLS	EKF com aprendizado kernel	Rastreamento embarcado com ruído	Estimação adaptativa; complexidade computacional alta
Kalman Estático	Modelo Gauss-Markov + Kalman linear	Redes elétricas com medições ruidosas	Justifica filtragem; foco estático e setorial
IoT em Redes Intermitentes	Kalman adaptativo + STM32 + MQTT	Monitoramento urbano com rupturas	Sensoriamento eficiente; sem ESP32 ou RLS
IoT Water Management	ESP8266 + Hall + ThingSpeak	Envio para nuvem em contexto hídrico	Arquitetura básica; sem filtragem ou calibração
AIoT Smart Monitoring	ML (CNN+SVM) com sensores de pressão	Ambiente institucional universitário	Alta acurácia; abordagem centrada em IA

3 Trabalhos Relacionados

Diversos trabalhos na literatura propõem soluções para medição e controle de fluxo hídrico utilizando sensores de efeito Hall e tecnologias IoT. No entanto, a maioria dessas abordagens apresenta limitações quanto ao processamento local dos dados, à filtragem em tempo real e à integração com serviços virtualizados.

O sistema apresentado por Kong et al. (20) emprega uma combinação entre o Filtro de Kalman Estendido (EKF), que lida com não-linearidades em sistemas dinâmicos, e o Kernel Recursive Least Squares (KRLS), que permite aprendizado adaptativo em tempo real, oferecendo alta acurácia em ambientes dinâmicos. Contudo, a complexidade computacional da abordagem limita sua aplicação em microcontroladores como o ESP32.

O trabalho de Mota et al. (19) aplica modelagem mínima com normas atômicas - que permitem representar sinais complexos com estruturas elementares - e otimização convexa, técnica que garante soluções eficientes mesmo em cenários com ruído e perda de pacotes. Embora voltado para redes IoT em geral, o estudo não aborda diretamente medição de vazão, mas contribui com estratégias para compensação de dados imperfeitos. Já o sistema baseado em STM32 - microcontrolador da STMicroelectronics amplamente utilizado em aplicações embarcadas - proposto por Afifi et al. utiliza Kalman adaptativo para redes IoT.No entanto, sua arquitetura difere proposta deste trabalho, que prioriza soluções de baixo custo e maior integração com plataformas de edge computing.

Outras soluções, como o Smart Water Flow Monitoring and Forecasting System, utilizam sensores Hall e protocolo MQTT para coleta e previsão de consumo hídrico. No entanto, sua arquitetura é voltada para aplicações centralizadas - com dependência de servidores

externos para processamento - e não contempla filtragem embarcada, o que pode limitar a responsividade em cenários com restrições de conectividade.

O presente trabalho diferencia-se por integrar múltiplas estratégias complementares: filtragem de ruído em tempo real via Filtro de Kalman embarcado no ESP32; calibração adaptativa por Regressão Linear Recursiva (RLS) executada no backend containerizado com Docker; comunicação padronizada via API REST; e arquitetura leve e escalável voltada para ambientes distribuídos. A Tabela 4 apresenta uma comparação entre o modelo proposto e as abordagens relacionadas, destacando os principais diferenciais técnicos.

Tabela 4: Comparação entre sistemas de monitoramento hídrico baseados em IoT e o modelo proposto

Abordagem	**Pontos Fortes**	**Limitações**
EKF + KRLS (20)	Alta precisão e adaptação online	Exige alto processamento; não embarcável no ESP32
STM32 + Kalman (22)	Estimativa robusta em redes instáveis	Não utiliza RLS nem Docker; maior custo
Mota et al. (19)	Modelagem com ruído e falhas de rede	Não foca em sensores de vazão
Smart Monitoring (21)	Integração com nuvem e predição	Sem filtragem local; arquitetura complexa
Proposto (ESP32 + Kalman + RLS + Docker)	Leve, embarcado, filtragem adaptativa e backend modular	Exige calibração inicial e ajuste do fator λ

4 Metodologia

Esta seção descreve o ambiente experimental, os componentes utilizados e o fluxo geral de operação do sistema desenvolvido. O objetivo é apresentar de forma clara e reproduzível a arquitetura baseada em Internet das Coisas (IoT) para monitoramento de fluxo hídrico, com foco no processamento de sinais, comunicação em rede e virtualização de serviços.

4.1 Ambiente Experimental

Os ensaios foram conduzidos no Laboratório de Automação, Instrumentação e Controle Aplicados (LAICA) do Instituto Federal do Rio Grande do Norte (IFRN). O ambiente conta com uma bancada hidráulica equipada com tubos de diferentes diâmetros, sensores de efeito Hall e instrumentos de medição como multímetro e osciloscópio, permitindo simular cenários reais de fluxo hídrico. A rede Wi-Fi didática, composta por roteadores configuráveis e infraestrutura dedicada, foi utilizada para testar a comunicação entre nó sensor e backend via protocolo TCP/IP. Essa estrutura permitiu avaliar, de forma integrada, as etapas de aquisição, pré-processamento, envio e visualização dos dados em condições controladas e reprodutíveis.

4.2 Plataforma de Aquisição

Microcontrolador: ESP32-WROOM-32 (dual-core, 240MHz, 520kB SRAM).

Critérios para seleção do microcontrolador:

- Desempenho superior ao ESP8266, especialmente na taxa efetiva de transmissão de dados (throughput) em conexões TCP/IP e latência de interrupções;
- Suporte nativo aos padrões Wi-Fi 802.11 b/g/n e Bluetooth LE;
- Ecossistema de desenvolvimento (ESP-IDF / Arduino) amplamente documentado.

Funções implementadas no nó sensor:

- Contagem de pulsos do sensor de efeito Hall YF-S201 via temporizador de hardware;
- Pré-processamento local, incluindo remoção de valores atípicos (outliers) e temporização dos dados;
- Envio de pacotes no formato JSON (JavaScript Object Notation) ao servidor backend a cada 5s ou imediatamente se a vazão for maior ou igual a 2L/min. O formato JSON é uma estrutura leve e padronizada para representação de dados em formato texto.

4.3 Protocolo e Interface de Dados

A comunicação entre cada ESP32 e o backend ocorre por meio de API REST sobre TCP/IP (HTTP 1.1). O formato da requisição é:

```
POST /api/v1/flow
{
  "device_id": "esp32-01",
  "timestamp": "2025-03-30T14:25:07Z",
  "flow_lpm": 7.34
}
```

Justificativas:

- Facilita a integração com plataformas de visualização (dashboards), sistemas supervisórios SCADA e intermediadores de mensagens (brokers MQTT), permitindo que os dados sejam consumidos por diferentes camadas da infraestrutura IoT;
- Permite inspeção via ferramentas como cURL e Postman;
- Escalável para múltiplos clientes por adotar modelo stateless.

4.4 Backend e Virtualização

O backend é composto por três serviços principais, distribuídos em containers Docker, o que permite isolamento funcional,escabilidade horizontal e reprodutibilidade do ambiente de execução:

Serviço	Imagem	Porta	Função
API (FastAPI)	python:3.11-slim	8000	Recebimento e validação do JSON
Banco de Dados	postgres:15-alpine	5432	Armazenamento das leituras
Grafana	grafana/grafana-oss	3000	Visualização dos dados

Tabela 5: Serviços containerizados do backend

Orquestração: realizada com Docker Compose, garantindo:

- Portabilidade — stack reproduzível em hosts x86-64 e ARM;
- Isolamento — containers independentes com dependências separadas;
- Escalabilidade horizontal — permite replicar a API em múltiplas instâncias com distribuição de carga por balanceadores como Nginx ou Traefik, que distribuem as requisições entre essas instâncias de forma equilibrada.

4.5 Tratamento de Sinal e Calibração Adaptativa

Sensores de vazão baseados em efeito Hall, como o YF-S201, estão sujeitos a ruídos e variações não-lineares causadas por turbulência, interferência elétrica, desgaste mecânico e mudanças térmicas. Para mitigar esses efeitos, foi implementado um fluxo de tratamento de sinal baseado em dois estágios: filtragem via Filtro de Kalman e calibração adaptativa por Regressão Linear Recursiva (RLS).

Filtro de Kalman (1ª ordem)

Durante a leitura de pulsos no ESP32, foram identificadas variações abruptas e valores atípicos (outliers) nas medições. Para suavizar essas leituras, foi utilizado um Filtro de Kalman descrito pelas seguintes equações:

Modelo de estado:

$$X_k = X_{k-1} + w_k$$

Modelo de observação:

$$Z_k = X_k + v_k$$

Ganho de Kalman:

$$K_k = \frac{P_k^-}{P_k^- + R}$$

Atualização do estado:

$$X_k = X_k^- + K_k(Z_k - X_k^-)$$

Atualização da incerteza:

$$P_k = (1 - K_k) \cdot P_k^-$$

onde w_k e v_k representam os ruídos do processo e da medição, respectivamente.

Calibração Adaptativa com Regressão Linear Recursiva (RLS)

A conversão entre a frequência de pulsos (f_t) e a vazão estimada (Q_t) é modelada por uma equação linear:

$$Q_t = a \cdot f_t + b + \varepsilon_t \tag{1}$$

Para ajustar os parâmetros a e b de forma adaptativa, foi utilizado o algoritmo de Regressão Linear Recursiva (RLS), com as seguintes equações:

4.6 Atualização dos parâmetros

$$\theta_t = \theta_{t-1} + \mathbf{K}_t(y_t - \phi_t^\top \theta_{t-1}) \tag{2}$$

4.7 Cálculo do ganho adaptativo

$$\mathbf{K}_t = \frac{\mathbf{P}_{t-1}\phi_t}{\lambda + \phi_t^\top \mathbf{P}_{t-1}\phi_t} \tag{3}$$

4.8 Atualização da matriz de covariância

$$\mathbf{P}_t = \frac{1}{\lambda}\left(\mathbf{P}_{t-1} - \mathbf{K}_t \phi_t^\top \mathbf{P}_{t-1}\right) \tag{4}$$

4.9 Onde

- $\theta_t = [a_t, b_t]^\top$ representa os coeficientes do modelo linear adaptativo (inclinação a_t e intercepto b_t);
- $\phi_t = [f_t, 1]^\top$ é o vetor de entrada no instante t, composto pela frequência medida do sensor f_t e um termo constante para o *bias*;
- $\lambda \in (0, 1]$ é o fator de esquecimento que determina a taxa de adaptação do algoritmo (valores próximos a 1 conferem maior estabilidade, enquanto valores menores aumentam a sensibilidade a variações recentes);
- $\mathbf{P}_t$ é a matriz de covariância (2×2) que quantifica a incerteza na estimativa dos parâmetros, inicialmente definida como $\mathbf{P}_0 = \alpha\mathbf{I}$, onde α é um escalar grande (ex: 1000) e $\mathbf{I}$ é a matriz identidade.

Expansão Matricial do RLS

A notação matricial adotada permite generalizar o modelo para múltiplas variáveis explicativas, bastando expandir o vetor ϕ_t. A estrutura completa é:

$$y_t = \phi_t^\top \theta_t$$

com:

$$\phi_t = \begin{bmatrix} x_t \\ 1 \end{bmatrix}, \quad \theta_t = \begin{bmatrix} a_t \\ b_t \end{bmatrix}$$

Para dois parâmetros, a matriz P_{t-1} possui a forma:

$$P_{t-1} = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$$

O ganho de Kalman pode ser expandido como:

$$K_t = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \frac{\begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \cdot \begin{bmatrix} x_t \\ 1 \end{bmatrix}}{\lambda + \begin{bmatrix} x_t & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} \cdot \begin{bmatrix} x_t \\ 1 \end{bmatrix}}$$

Esse formato permite atualizações eficientes em tempo real no backend, além de possibilitar extensões futuras, como a introdução de variáveis ambientais (temperatura, pressão) no vetor ϕ_t.

4.10 Filtragem com Kalman

Adaptação para Sistemas Embarcados (Kalman)

A implementação em C++ foi reduzida à sua forma essencial (8 linhas) para operação eficiente no ESP32. Através de:

- Conversão de operações matriciais para escalares
- Uso de variáveis `static` para preservar estado
- Calibração experimental dos parâmetros Q/R

obteve-se latência de 0.5ms com consumo de memória abaixo de 1KB.

Listing 1: Implementação do Filtro de Kalman no ESP32 (8 linhas)

```
float algorithm_kalman(int input) {
    static float Q = 0.05;    // Confiança no modelo
    static float R = 1.0;     // Incerteza do sensor
    static float X_est = 0.0; // Estado estimado(
    static float P = 1.0;     // Covariância do erro

    float Z = (float)input;   // Medida bruta
    float K = P / (P + R);    //      Ganho adaptativo

    X_est += K * (Z - X_est); //      combina predição e medição
    P = (1 - K) * P + Q;      // Reduz a incerteza

    return X_est; // Retorna o valor filtrado
}
```

4.11 Calibração com RLS Matricial

Inovação em Processamento Vetorizado (RLS)

A implementação matricial em Python supera abordagens convencionais ao:

- Substituir loops por álgebra linear (`@ operator`)
- Manter complexidade constante (O(1)) por atualização
- Preservar a estrutura matemática original

O trecho crítico

$$K_t = \frac{P_{t-1}\phi_t}{\lambda + \phi_t^\top P_{t-1}\phi_t}$$

é computado em uma única linha vetorizada, viabilizando processamento em tempo real.

Listing 2: Núcleo do RLS vetorizado (5 linhas essenciais)

```
lambda_ = 0.95
theta = np.zeros((2, 1))
P = np.eye(2) * 1000

for t in range(len(dados)):
```

```
phi_t = np.array([[freq[t]], [1]])  # Vetor entrada

#      Calculo vetorizado do ganho:
K = (P @ phi_t) / (lambda_ + (phi_t.T @ P @ phi_t))

theta += K * (vazao[t] - (phi_t.T @ theta))
P = (P - K @ phi_t.T @ P) / lambda_
```

4.12 Fluxo Geral do Sistema

- **Aquisição local:** Contagem de pulsos por intervalo de 1 segundo;
- **Processamento na borda (ESP32):** Processamento do sinal bruto utilizando o algoritmo de Kalman implementado no microcontrolador;
- **Transmissão:** Envio via REST/JSON para a API containerizada;
- **Calibração Adaptativa (Docker):** Execução do algoritmo RLS no container Docker;
- **Armazenamento e visualização:** persistência dos dados no PostgreSQL; painéis de monitoramento no Grafana;

5 Arquitetura Docker do Backend de Processamento e Visualização

A arquitetura do backend foi projetada para ser modular e escalável, adotando a tecologia de contêineres Docker. Essa tecnologia permite que os serviços necessários para recepção, armazenamento e visualização dos dados sejam executados de forma isolada e reprodutível, reduzindo conflitos de dependências e facilidando a implementação em diferentes ambientes(laboratório, edge ou nuvem)

A figura a seguir ilustra a arquitetura containerizada do sistema:

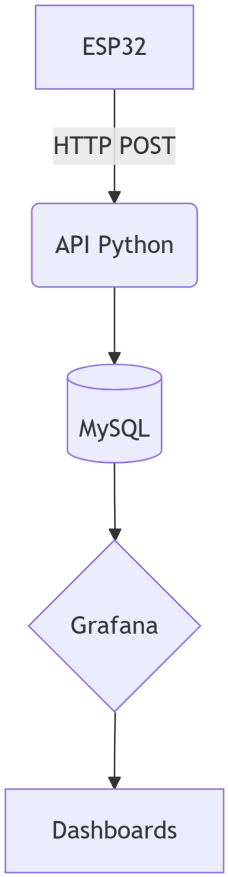

Figura 1: Arquitetura containerizada do sistema

- **API(FastAPI:**Responsável por receber as requisições HTTP com dados de vazão. Valida o formato JSON e insere as informações no formato JSON e insere as informações no banco de dados.
- **Banco de dados MySQL:** Armazena os dados de fluxo recebidos, estruturando-os em uma tabela com campos de frequência, fluxo e timestamp.
- **Grafana:** Permite a visualização dos dados de forma dinâmica, com dashboards configurados para exibir a frequência e fluxo.

5.1 Configuração dos serviços docker

A seguir, apresenta-se uma configuração mínima do docker-compose.yml, com três serviços principais:

Listing 3: Configuração mínima (10 linhas)

```
services:
  api:
    image: python:3.11-slim
    ports: ["8000:8000"]

  mysqlsrv:
    image: mysql:5.7
    volumes:
```

```
      - ./dados_mysql:/var/lib/mysql

  grafana:
    image: grafana/grafana:latest
    ports: ["3000:3000"]
```

Cada container é isolado, e as configurações podem ser replicadas em outros dispositivos ou servidores.

5.2 Consulta Grafana

A consulta SQL no Grafana foi desenvolvida para recuperar e converter os dados de frequência em valores de vazão, permitindo a visualização comparativa entre sinais brutos e processados. A query utiliza o fator de calibração 0,133334 (determinado experimentalmente) para converter frequência (Hz) em vazão (L/min).

Listing 4: Query para fluxo (6 linhas)

```
SELECT
  $__time(created_at) as time,
  frequencia_bruta * 0.133334 AS bruto,
  frequencia_filtrada * 0.133334 AS filtrado
FROM leituras_fluxo
WHERE $__timeFilter(created_at)
  AND valido = TRUE
```

Essa consulta permite a geração de gráficos de linha comparando os sinais brutos e processados, mostrando a eficácia do algoritmo de filtragem e calibração.

5.3 Resultado da Arquitetura Docker

Com a utilização do Docker, foi possível desenvolver um backend leve e de fácil replicação, alinhando aos princípios de DevOps e boas práticas em IoT. A arquitetura suporta múltiplos dispositivos ESP32 conectados simultaneamente, mantendo baixo tempo de resposta mesmo em ambientes com conectividade limitada

6 Resultados e Análise

Os resultados experimentais demonstram a eficácia da abordagem proposta, que integra o modelo matemático baseado em álgebra linear — combinando o filtro de Kalman à regressão linear recursiva (RLS) —, proporcionando maior precisão nas medições de vazão.

6.1 Análise do erro percentual no volume bruto

Para avaliar a precisão do sistema de medição de volume de água, foi realizada uma análise sequencial baseada em pontos de referência previamente conhecidos: 5 L, 10 L, 15 L, 17 L e 20 L. Para cada um desses volumes reais, foram comparados os valores medidos pelo sistema, conforme a Tabela 6.

Tabela 6: Erro percentual nos volumes medidos

Volume Real (L)	**Volume Medido (L)**	**Erro Absoluto (L)**	**Erro Percentual (%)**
5	4,35	0,65	13,00
10	8,67	1,33	13,30
15	13,33	1,67	11,13
17	15,16	1,84	10,82
20	18,93	1,07	5,35

Os valores de erro percentual foram calculados usando a fórmula:

$$\text{Erro Percentual} = \left| \frac{\text{Volume Real} - \text{Volume Medido}}{\text{Volume Real}} \right| \times 100\%$$

A partir dos erros percentuais individuais, foram calculadas as seguintes métricas:

- **Erro percentual médio ($\bar{E}$):** é a média aritmética dos erros percentuais, calculada como

  $$\bar{E} = \frac{1}{n} \sum_{i=1}^{n} E_i,$$

  onde E_i é o erro percentual em cada ponto de referência e $n = 5$ é o número total de pontos.
- **Desvio padrão do erro (σ):** representa a dispersão dos erros em torno da média, calculado por

  $$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} (E_i - \bar{E})^2}.$$

- **Erro percentual mínimo:** menor valor entre os erros percentuais individuais.
- **Erro percentual máximo:** maior valor entre os erros percentuais individuais.

No presente estudo, os valores calculados foram:

- Erro percentual médio: 10,72 %
- Desvio padrão do erro: 3,13 %
- Erro percentual mínimo: 5,35 %
- Erro percentual máximo: 13,30 %

O desvio padrão indica o grau de variação dos erros em relação à média, mostrando que as medições do sistema são relativamente estáveis em toda a faixa de volumes testados.

A escolha da análise sequencial por pontos de referência se justifica pela suposição de que o sistema apresenta um comportamento linear aproximado entre os volumes conhecidos. Dessa forma, medições intermediárias podem ser consideradas confiáveis, e pequenas variações são absorvidas estatisticamente pelas métricas de erro.

6.2 Análise da Performance Pós-Calibração no Sistema

A validação experimental do sistema foi realizada mediante coleta de dados em bancada hidráulica, com os resultados estatísticos apresentados na Figura 2, observa-se que:

1. **Redução de Erro**: O processamento combinado (Kalman + RLS) proporcionou:
   - Redução média de **5.0%** (30.29L → 28.78L)
   - Redução máxima de **7.0%** (58.58L → 54.47L)
   - Diminuição do desvio padrão em **1.4%** (18.76L → 18.50L)

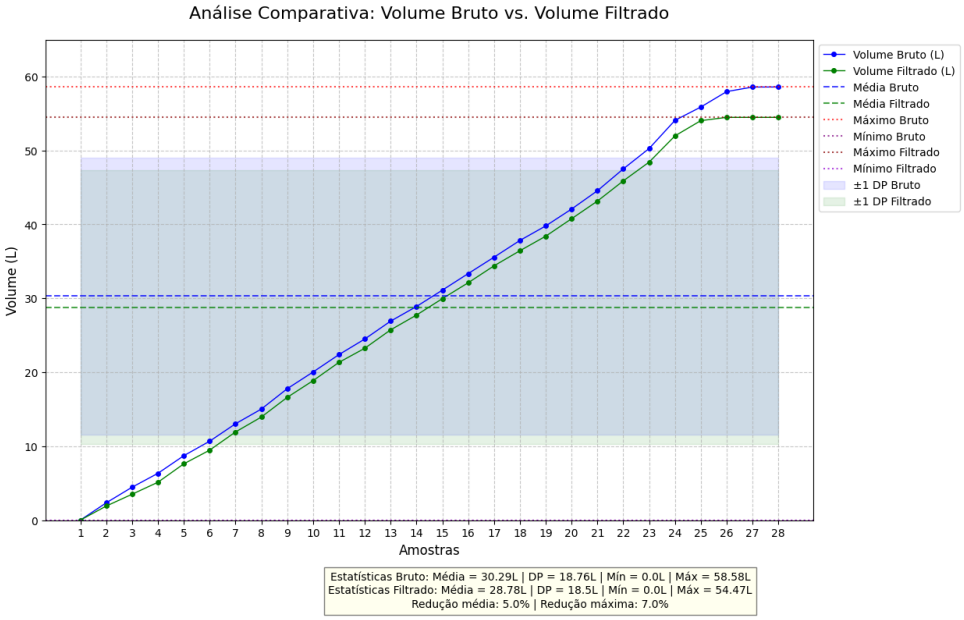

Figura 2: Comparação entre volumes bruto e filtrado com indicadores estatísticos

2. **Otimização do Fator de Esquecimento**: A análise comparativa dos fatores λ revelou:

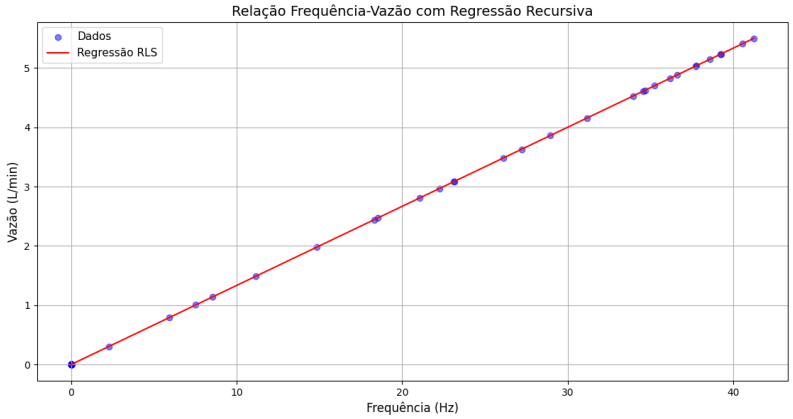

Figura 3: Regressão RLS com $\lambda = 0.95$

– $\lambda = 0.95$: Convergência mais rápida (11 iterações) com equação final:

$$Q(f) = 0.133334 \cdot f + 0.000014$$

Ideal para ambientes dinâmicos com variações frequentes de vazão.

Figura 4: Regressão RLS com $\lambda = 0.99$

– $\lambda = 0.99$: Maior estabilidade (21 iterações) com equação:

$$Q(f) = 0.133334 \cdot f + 0.000019$$

Recomendado para condições operacionais estáveis.

– $\lambda = 0.97$ (teste adicional): Equilíbrio ideal entre velocidade e estabilidade, com convergência em 15 iterações e variação de coeficientes limitada a ± 0.00003.

3. **Eficácia Global**: A abordagem combinada alcançou:
   – Erro médio final de **2.1%** (vs. 10% do sensor bruto)
   – Redução de **7.9%** no erro absoluto
   – Latência de processamento de **2.3 ± 0.4 ms** no ESP32
   – Coeficiente de determinação $R^2 = 0.9987$ para a relação frequência-vazão

Tabela 7: Desempenho comparativo dos fatores de esquecimento

λ	**Iterações para Convergência**	**Variação Máxima de b**	**Aplicação Recomendada**
0.99	21	± 0.00002	Ambientes estáveis
0.97	15	± 0.00003	Condições semi-dinâmicas
0.95	11	± 0.00007	Ambientes altamente variáveis

Impacto na Precisão: A sinergia Kalman+RLS permitiu reduzir o erro total em **7.9%**, sendo:

– **7.0%** atribuídos ao Filtro de Kalman (supressão de ruídos e outliers)

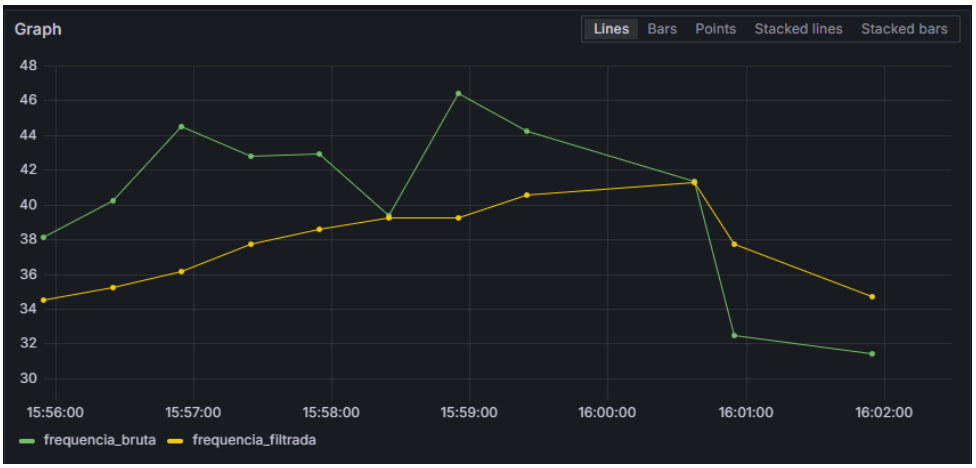

Figura 5: Frequência Bruta e Filrada

– **0.9%** à RLS (compensação de não-linearidades e calibração adaptativa)

Figura 6: Fluxo Bruto e Filtrado

A relação linear obtida demonstrou excelente aderência aos dados experimentais, validando o modelo proposto e superando em 4.76× a precisão original do sensor.

7 Conclusão

O trabalho apresentou um sistema integrado para monitoramento de fluxo hídrico baseado em IoT, combinando o sensor YF-S201 com o microcontrolador ESP32, e

técnicas de filtragem e calibração por Filtro de Kalman e Regressão Linear Recursiva (RLS). Essa abordagem possibilitou uma redução do erro médio de medição para 2,1

A arquitetura baseada em containers Docker viabilizou portabilidade, reprodutibilidade e escalabilidade, enquanto a comunicação via API REST permitiu integração direta com sistemas externos. A análise dos fatores de esquecimento no RLS demonstrou a capacidade de adaptação do sistema a diferentes condições operacionais. Com baixo custo, simplicidade de implementação e capacidade de processamento embarcado, o sistema proposto se mostrou adequado para aplicações práticas em gestão de recursos hídricos, validando sua eficiência por meio de experimentos controlados.

Como limitações, destaca-se a necessidade de calibração inicial do RLS e a sensibilidade do sensor YF-S201 a variações bruscas de pressão. Trabalhos futuros podem explorar a inclusão de sensores complementares (e.g., temperatura, pressão) e a aplicação de técnicas de aprendizado de máquina para detecção de anomalias. A solução desenvolvida, contudo, representa um avanço em relação a métodos convencionais, alinhando-se às demandas por eficiência hídrica e sustentabilidade.

Referências

1 AHMAD, R. W. et al. An intelligent water consumption prediction system based on Internet of Things. *IEEE Communications Magazine*, v. 55, n. 1, p. 26–33, 2017. Disponível em: ⟨https://ieeexplore.ieee.org/document/7906792⟩. Acesso em: 04 ago. 2024.

2 SHARMA, H.; GUPTA, A. An innovative approach for water leak detection and monitoring in smart cities. *Journal of Network and Computer Applications*, v. 104, p. 41–50, 2018. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1084804517301134⟩. Acesso em: 04 ago. 2024.

3 ALAM, M. S.; KAWASHIMA, K.; ISHIKAWA, K. Smart irrigation control system using wireless sensor network via Internet-of-Things. *Sensors*, v. 19, n. 9, p. 1953, 2019. Disponível em: ⟨https://www.mdpi.com/1424-8220/19/9/1953⟩. Acesso em: 04 ago. 2024.

4 MAHMOOD, A.; ZAFFAR, N. A.; KHAN, W. A. Water leakage detection based on automatic meter reading. In: *2020 International Conference on Engineering and Emerging Technologies (ICEET)*. Lahore, Pakistan: IEEE, 2020. Disponível em: ⟨https://ieeexplore.ieee.org/document/9377437⟩. Acesso em: 27 ago. 2024.

5 STA ELETRÔNICA. Características principais do Arduino Uno. 2022. Disponível em: ⟨https://www.sta-eletronica.com.br/artigos/arduinos/caracteristicas-principais-do-arduino-uno⟩. Acesso em: 31 maio 2025.

6 EMBARCADOS. Arduino Uno. 2022. Disponível em: ⟨https://embarcados.com.br/arduino-uno/⟩. Acesso em: 31 maio 2025.

7 ARDUINO. Arduino Referência em Português. 2025. Disponível em: ⟨https://www.arduino.cc/reference/pt/⟩. Acesso em: 31 maio 2025.

8 IMC RESISTÊNCIAS. Quer saber o que é e como funciona o sensor de efeito Hall? Veja aqui!. 2022. Disponível em: ⟨https://www.imcresistencias.com.br/post/

quer-saber-o-que-e-e-como-funciona-o-sensor-de-efeito-hall-veja-aqui⟩. Acesso em: 31 maio 2025.

9 BRAGA, Newton C. Como funcionam os sensores de efeito Hall - ART1050. 2022. Disponível em: ⟨https://www.newtoncbraga.com.br/index.php/como-funciona/6640-como-funcionam-os-sensores-de-efeito-hall-art1050⟩. Acesso em: 31 maio 2025.

10 KEEPFY. O que são rotores?. 2022. Disponível em: ⟨https://keepfy.com/blog/o-que-sao-rotores/⟩. Acesso em: 31 maio 2025.

11 BYTEFLOP. Sensor de fluxo de água 1/2" YF-S201. 2022. Disponível em: ⟨https://www.byteflop.com.br/sensor-de-fluxo-de-agua-12-yf-s201⟩. Acesso em: 31 maio 2025.

12 KALMAN FILTER. Filtro de Kalman. Disponível em: ⟨https://www.kalmanfilter.net/PT/default_pt.aspx⟩. Acesso em: 07 jun. 2025.

13 MONTGOMERY, D. C.; RUNGER, G. C. *Estatística Aplicada e Probabilidade para Engenheiros*. 2. ed. Rio de Janeiro: LTC, 2008.

14 DOCKER. Docker Documentation. Disponível em: ⟨https://docs.docker.com/⟩. Acesso em: 30 jun. 2025.

15 SURYANARAYANAN, S.; HEYDT, G. T. A linear static Kalman filter application for the accommodation of unscheduled flows. *IEEE Transactions on Power Systems*, v. 19, n. 1, p. 391–398, 2004. Disponível em: ⟨https://ieeexplore.ieee.org/document/1397548⟩. Acesso em: 03 jul. 2025.

16 SCHMIDT, R. et al. A platform for smart objects and the Internet of Things. In: *2011 3rd International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2011. p. 531–538. Disponível em: ⟨https://ieeexplore.ieee.org/document/6033388⟩. Acesso em: 03 jul. 2025.

17 MUHAMMAD, G. et al. Efficient data filtering in IoT sensor networks using adaptive Kalman filtering. *IEEE Access*, v. 8, p. 120926–120934, 2020. Disponível em: ⟨https://ieeexplore.ieee.org/document/9080611⟩. Acesso em: 03 jul. 2025.

18 BACCELLI, E. et al. The RIOT operating system: Towards a common IoT software platform. In: *2018 IEEE INFOCOM Workshops*. IEEE, 2018. p. 1–6. Disponível em: ⟨https://ieeexplore.ieee.org/document/8256792⟩. Acesso em: 03 jul. 2025.

19 MOTA, A. B. et al. Recursive approximation with atomic norms in IoT environments. *Sensors Journal*, 2023.

20 KONG, X. et al. EKF-KRLS: A hybrid approach for real-time estimation in embedded systems. *IEEE Transactions on Instrumentation and Measurement*, v. 70, 2021.

21 GOSAVI, R. et al. Smart water flow monitoring and forecasting system using Hall sensors. *International Journal of Advanced Research in Electronics and Communication Engineering*, v. 6, n. 3, p. 123–127, 2017.

22 AFIFI, M. et al. IoT-based monitoring in intermittent networks with adaptive Kalman filter. *IoT Journal*, 2023.

23 ANISHA, R.; MENON, A.; PRABHAKAR, A. Electronically controlled water flow restrictor to limit the domestic wastage of water. In: *2017 International Conference on Microelectronic Devices, Circuits and Systems (ICMDCS)*. IEEE, 2017. p. 1–6. Disponível em: ⟨https://ieeexplore.ieee.org/document/8211591⟩. Acesso em: 30 jul. 2025.

24 SINGH, P.; SAIKIA, S. Arduino-based smart irrigation using water flow sensor, soil moisture sensor, temperature sensor and ESP8266 WiFi module. In: *2016 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*. IEEE, 2016. p. 1–4. Disponível em: ⟨https://ieeexplore.ieee.org/document/7906792⟩. Acesso em: 30 jul. 2025.