UNIVERSIDADE DO VALE DO ITAJAÍ
CURSO DE MESTRADO EM COMPUTAÇÃO APLICADA

Klaus Dieter Kupper

A Review of Current Sensor Technologies for Environmental Monitoring : For the Era of IoT and Wireless Sensor Networks

Itajaí
2025

Klaus Dieter Kupper

A Review of Current Sensor Technologies for Environmental Monitoring : For the Era of IoT and Wireless Sensor Networks

Trabalho de Conclusão de Curso de Mestrado em Computação Aplicada da Universidade do Vale do Itajaí como requisito para a obtenção do título de Mestrado em Computação Aplicada.
Orientador: Prof. Dr. Jordan Sausen

Itajaí
2025

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Kupper, Klaus Dieter
Estudo e Implementação de um Sistema Web para Vendas e Pagamentos / Klaus Dieter Kupper ; orientador, Carlos Roberto Moratelli, 2023.
52 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2023.

Inclui referências.

1. Engenharia de Controle e Automação. 2. Internet of Things. 3. Automação de Pedidos. 4. Desenvolvimento Web. 5. Sockets. I. Moratelli, Carlos Roberto. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Klaus Dieter Kupper

A Review of Current Sensor Technologies for Environmental Monitoring : For the Era of IoT and Wireless Sensor Networks

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Mestrado em Computação Aplicada" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Itajaí, 10 de Julho de 2025.

Banca Examinadora:

Prof. Dr. Carlos Roberto Moratelli
Universidade Federal de Santa Catarina

Prof. Dr. Jordan Sausen
Universidade Federal de Santa Catarina

Prof. Dr. Ciro André Pitz
Universidade Federal de Santa Catarina

Este trabalho é dedicado aos meus colegas de classe, a
minha namorada e aos meus queridos pais.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão ao meu professor orientador, pela sua orientação acadêmica e apoio ao longo deste trabalho. Suas orientações sábias e paciência foram fundamentais para o seu desenvolvimento.

A todos os professores que tive ao longo da minha jornada acadêmica, sou imensamente grato. Cada aula, conselho e palavra contribuíram para o meu crescimento pessoal e profissional. Em especial, agradeço por terem despertado em mim o interesse pela engenharia e tecnologia, que hoje são a minha paixão e a área em que atuo.

Agradeço também aos meus pais, cujo amor, apoio incondicional e crença em mim foram essenciais para que eu chegasse até aqui. Vocês foram minha inspiração e força motriz em todos os momentos desafiadores.

Aos meus colegas, expresso minha gratidão pelo apoio, compreensão e companheirismo ao longo dessa jornada. Nossos momentos de estudo, discussões e desafios compartilhados foram fundamentais para o meu crescimento e formação.

Por fim, gostaria de estender meu agradecimento a todos que, de alguma forma, contribuíram para a minha formação. Cada gesto e palavra de apoio foram importantes para o meu crescimento como pessoa e profissional.

"O computador é a bicicleta da mente." (Steve Jobs, 1985)

RESUMO

Este trabalho apresenta uma análise comparativa de sensores ultrassônicos, LiDAR e câmeras para a medição do nível de rios. O objetivo é avaliar o desempenho desses sensores em diferentes condições ambientais, considerando fatores como precisão, alcance máximo e limitações operacionais. Os testes experimentais serão conduzidos em um ambiente controlado e em um cenário real, permitindo uma avaliação detalhada das capacidades de cada tecnologia. Com base nos resultados obtidos, serão indicadas as aplicações mais adequadas para cada tipo de sensor e determinada a melhor opção para o monitoramento do nível de rios. Os achados deste estudo podem contribuir para o aprimoramento de sistemas de monitoramento hidrológico, auxiliando na escolha de sensores mais eficientes e adequados para diferentes cenários.

Palavras-chave: Sensores Ultrassônicos; LiDAR; Câmeras; Monitoramento Hidrológico; Sensoriamento Remoto.

ABSTRACT

This work presents a comparative analysis of ultrasonic sensors, LiDAR, and cameras for river level measurement. The objective is to evaluate the performance of these sensors under different environmental conditions, considering factors such as accuracy, maximum range, and operational limitations. Experimental tests will be conducted in a controlled environment and a real-world scenario, enabling a detailed assessment of each technology's capabilities. Based on the obtained results, the most suitable applications for each sensor type will be identified, and the best option for river level monitoring will be determined. The findings of this study may contribute to the improvement of hydrological monitoring systems, helping to select more efficient and appropriate sensors for different scenarios.

Keywords: Ultrasonic Sensors; LiDAR; Cameras; Hydrological Monitoring; Remote Sensing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
IA	Inteligência Artificial
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
JWT	*JSON Web Token*
QR	*Quick Response*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
tRPC	*Typescript Remote Procedure Call*
UUID	*Universal Unique Identifier*
WWW	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

The increasing severity of climate change, pollution, and ecological degradation has elevated the need for continuous, accurate, and scalable environmental monitoring. Understanding the dynamics of air quality, water purity, soil health, and meteorological conditions is essential for informed decision-making in fields such as public health, agriculture, urban planning, and conservation. Traditional environmental data collection methods—largely manual and periodic—are no longer sufficient to capture the spatial and temporal complexity of these phenomena.

Recent advancements in sensor technology, coupled with the rise of the Internet of Things (IoT) and wireless sensor networks (WSNs), have revolutionized the way environmental data is collected, transmitted, and analyzed. These systems enable real-time, distributed, and often autonomous monitoring across vast geographic areas, even in remote or harsh environments. The convergence of low-power sensors, edge computing, and long-range wireless communication protocols has made it feasible to deploy scalable and cost-effective monitoring infrastructures.

This review aims to provide a comprehensive overview of the current state of sensor technologies used in environmental monitoring. It explores the different types of environmental parameters and the corresponding sensor modalities, examines how these sensors are integrated into modern IoT-based systems, and discusses the challenges related to data quality, calibration, power consumption, and communication. Additionally, it highlights emerging trends in sensor development and identifies research opportunities in this rapidly evolving field.

1.1 OBJETIVO GERAL

O objetivo geral deste projeto é avaliar o desempenho de sensores ultrassônicos, sensores LiDAR e câmeras na medição do nível de rios, comparando suas capacidades e limitações em diferentes cenários. Com base nos resultados, busca-se determinar qual tecnologia apresenta melhor custo-benefício e maior adequação para essa aplicação específica. Foram escolhidos três tipos de sensores para análise: sensores ultrassônicos, sensores LiDAR e câmeras.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar uma revisão bibliográfica sobre as principais tecnologias de sensores utilizadas no monitoramento de níveis de líquidos e corpos d'água.
2. Definir um conjunto de critérios para avaliação dos sensores, considerando aspectos como precisão, alcance máximo, resistência a interferências ambientais e custo.

3. Conduzir testes experimentais com sensores ultrassônicos, sensores LiDAR e câmeras em ambiente controlado.
4. Realizar testes em um cenário real, avaliando o impacto de fatores como variação climática, presença de detritos e mudanças no fluxo da água.
5. Comparar os resultados obtidos e identificar as aplicações mais adequadas para cada tipo de sensor.
6. Determinar a tecnologia mais eficiente para o monitoramento do nível de rios, considerando aspectos de viabilidade técnica e econômica.
7. Elaborar recomendações para a implementação de sistemas de medição baseados nos sensores analisados.

1.3 ESTRUTURA DO DOCUMENTO

Este trabalho está dividido em seis capítulos. O Capítulo 1 apresenta a introdução e os objetivos da pesquisa. O Capítulo 2 trata da fundamentação teórica, abordando os princípios de funcionamento dos sensores analisados. O Capítulo 5 descreve a metodologia e os critérios adotados para a realização dos testes. O Capítulo 6 apresenta os experimentos conduzidos e as tecnologias utilizadas. O Capítulo 7 discute os resultados obtidos e compara as diferentes abordagens. Por fim, o Capítulo 8 traz as considerações finais e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção é dedicada à exploração e discussão dos conceitos e teorias que embasam este trabalho. Neste capítulo, serão apresentados os principais temas e tecnologias envolvidos, como HTTP, NodeJS, *websockets*, e APIs. A discussão será organizada de maneira progressiva, começando pelos conceitos mais gerais e avançando para os aspectos mais específicos e técnicos, como as bibliotecas e *frameworks* utilizados.

2.1 AUTOMAÇÃO DE SERVIÇOS

Automação de processos, se refere ao uso de tecnologia para realizar tarefas rotineiras e repetitivas de maneira eficiente e sem erros, e é um aspecto fundamental da automação de serviços.

A automação de serviços é uma tendência em crescimento nos últimos anos, especialmente na área de comércio eletrônico. Ela busca tornar processos mais eficientes, reduzir custos e oferecer melhores experiências aos clientes. Através da automatização, é possível reduzir erros e aumentar a precisão e rapidez no atendimento ao cliente. Com isso, os estabelecimentos podem oferecer um serviço de qualidade, com maior agilidade e menor tempo de espera.

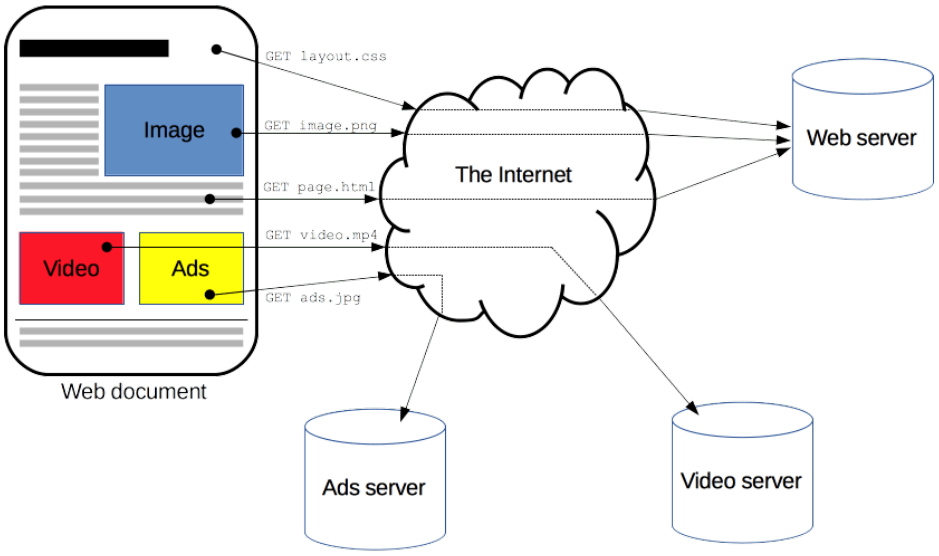
A pandemia de COVID-19 acelerou a necessidade de digitalização em muitos setores. Com as restrições de movimento e o fechamento de lojas físicas, muitos negócios tiveram que se adaptar rapidamente para oferecer seus serviços online. Isso levou a um aumento na demanda por automação de serviços, à medida que as empresas procuravam maneiras de continuar operando de forma eficiente em um ambiente digital. De acordo com um estudo de Sousa, Silva e Araújo (2008), a automação não é apenas uma parte do processo; é um ecossistema em desenvolvimento em que a IA (Inteligência Artificial) e a IoT trabalham juntas para criar uma força de trabalho mais inteligente, eficiente e produtiva (Gourley; Totty, 2002).

2.2 COMUNICAÇÃO HTTP E WEB

O protocolo HTTP (*Hypertext Transfer Protocol*) é um dos principais métodos para transferência de dados via *web*. Trata-se de um protocolo de cliente-servidor, o que significa que as solicitações são iniciadas pelo cliente, geralmente o navegador da *web*, e um documento completo é reconstruído a partir dos diferentes recursos buscados, como texto, descrição de *layout*, imagens, vídeos, *scripts* e mais (HTTP..., s.d.), a Figura 1 ilustra esse processo.

O HTTP é um protocolo sem estado, o que significa que cada requisição é independente das outras. Isso permite que a web seja altamente escalável, pois os servidores não precisam manter informações sobre cada usuário.

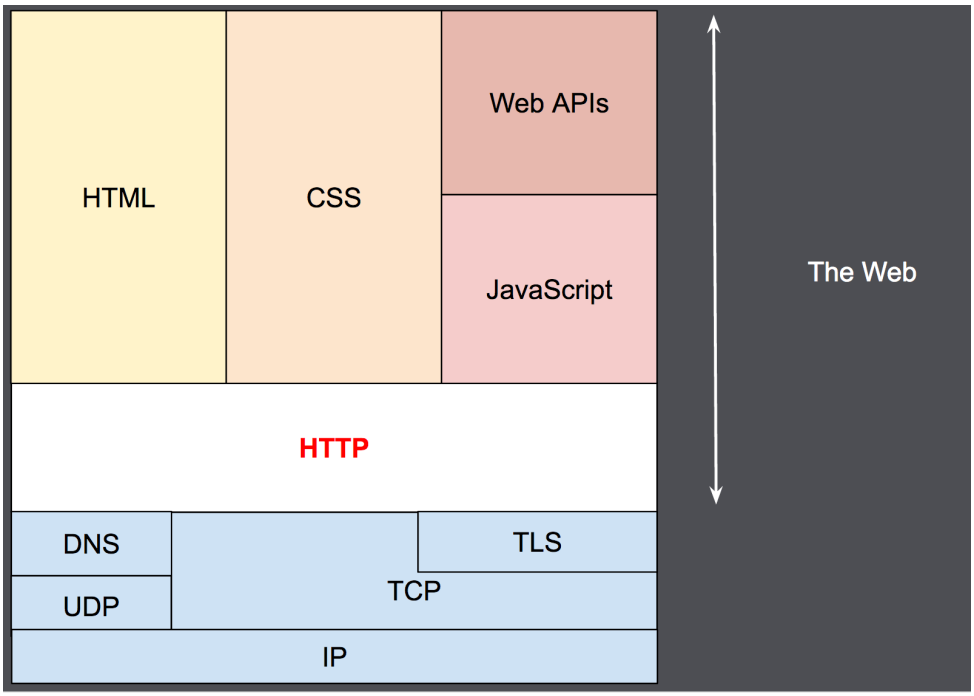
Figura 1 – Buscando dados da Web.



Fonte: (HTTP..., s.d.).

A WWW foi criada em 1989 por Tim Berners-Lee, um cientista da computação britânico que trabalhava no CERN, o laboratório de física de partículas na Suíça. O objetivo era criar uma maneira fácil de compartilhar informações entre os cientistas que trabalhavam em diferentes universidades e institutos ao redor do mundo. O HTML foi a linguagem de marcação que Berners-Lee desenvolveu para criar páginas *web*. A Figura 2 demonstra a conexão entre estas tecnologias e como elas se encaixam para produzir a *web*.

Figura 2 – Camadas da *web*.



Fonte: (HTTP..., s.d.).

Com o tempo, o HTTP evoluiu e novas versões foram lançadas. A versão mais recente é o HTTP/3, que oferece melhor desempenho e segurança em comparação com as versões anteriores. No entanto, o HTTP/1.1 e HTTP/2 ainda são amplamente utilizados na *web* (Elhadef; Grira, 2016).

Um conceito importante para a comunicação HTTP são os métodos, os principais métodos HTTP são:

- **GET:** Solicita um recurso do servidor. Este é o método mais comum e é usado para solicitar a visualização de páginas da web.
- **POST:** Envia dados para o servidor para criar um novo recurso. Os dados são incluídos no corpo da solicitação.
- **PUT:** Envia dados para o servidor para atualizar um recurso existente. Os dados são incluídos no corpo da solicitação.
- **DELETE:** Solicita a exclusão de um recurso no servidor.
- **OPTIONS:** Solicita informações sobre os métodos de comunicação disponíveis para um recurso ou para o servidor em geral.
- **PATCH:** Aplica modificações parciais a um recurso.
- **CONNECT:** É usado para abrir uma conexão de rede bidirecional com o recurso solicitado. Geralmente é usado para acesso SSL (HTTPS).

Esses métodos são definidos no protocolo HTTP e são usados para indicar a ação desejada a ser realizada no recurso especificado. Eles formam a base da interação entre o cliente e o servidor na web (Elhadef; Grira, 2016).

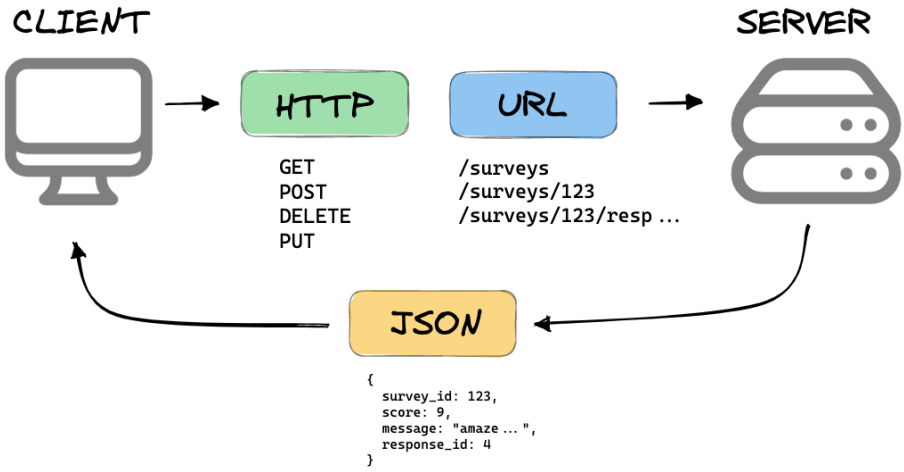
2.3 API

Uma API (*Application Programming Interface*) é um conjunto de rotinas, protocolos e ferramentas para construção de software e aplicações. Ela define como os componentes de software devem interagir entre si, permitindo que diferentes aplicações possam se comunicar e compartilhar informações (Mulloy, 2012).

2.3.1 REST API

REST (*Representational State Transfer*) é um estilo arquitetural para sistemas distribuídos, baseado no protocolo HTTP. Ele define um conjunto de restrições que devem ser seguidas para que as aplicações possam se comunicar de forma eficiente e escalável. Uma das principais características do REST é a separação entre cliente e servidor, onde o cliente faz requisições ao servidor para acessar recursos, e o servidor responde com uma representação do estado atual do recurso solicitado. A Figura 3 mostra a arquitetura de uma REST API.

Figura 3 – API Rest.



Fonte: (**restAPI**).

O REST é um elemento fundamental na construção de APIs modernas, devido à sua simplicidade e eficiência. Ele permite que os desenvolvedores criem APIs que podem ser facilmente consumidas por diferentes clientes, incluindo navegadores web, aplicativos móveis e outros servidores. Além disso, o REST é independente de linguagem, o que significa que pode ser usado com qualquer linguagem de programação que suporte HTTP.

O REST foi proposto por Roy Fielding em sua tese de doutorado em 2000. Fielding é um dos principais contribuidores para o desenvolvimento do protocolo HTTP e co-fundador da Apache HTTP Server Project. Em sua tese, Fielding descreveu o REST como um conjunto de princípios arquiteturais que podem ser usados para projetar sistemas distribuídos que são escaláveis, eficientes e fáceis de modificar e manter.

Os métodos HTTP citados na Seção 2.2 são incorporados ao REST. Eles formam um estilo arquitetural que utiliza métodos com a mesma semântica para permitir a construção de APIS, com rotas do tipo POST, GET, DELETE etc.

Desde então, o REST se tornou o estilo arquitetural mais popular para a construção de APIs na web. Ele é usado por muitas grandes empresas, incluindo Google, Facebook e Twitter, para fornecer acesso programático aos seus serviços (Richardson; Ruby, 2007).

2.3.2 APIs de Pagamento

As APIs de pagamento surgiram como uma necessidade para facilitar as transações online. No início, as APIs de pagamento eram principalmente usadas para processar pagamentos com cartão de crédito. Empresas como a PayPal foram pioneiras nesse campo, fornecendo APIs que permitiam aos comerciantes aceitar pagamentos com cartão de crédito em seus sites.

Com o tempo, as APIs de pagamento evoluíram para suportar uma variedade de métodos de pagamento. Isso inclui não apenas cartões de crédito, mas também débito

direto, pagamentos móveis e até mesmo cripto-moedas. Além disso, as APIs de pagamento também começaram a oferecer funcionalidades adicionais, como suporte para pagamentos recorrentes, reembolsos, e a capacidade de gerenciar várias moedas.

No Brasil, intermediários de pagamento como o Mercado Pago e o PagSeguro oferecem APIs que permitem aos comerciantes aceitar uma variedade de métodos de pagamento, incluindo boleto bancário e transferências bancárias. Recentemente, com a introdução do PIX, um sistema de pagamentos instantâneos operado pelo Banco Central do Brasil, esses intermediários também começaram a oferecer APIs que suportam PIX, permitindo transações quase instantâneas (Aué et al., 2018; ADYEN. . . , s.d.).

2.4 SOCKETS E WEBSOCKETS

Os *sockets e webSockets* são tecnologias fundamentais para a comunicação em tempo real na internet. *Sockets* são um mecanismo de comunicação bidirecional entre dois nós em uma rede, permitindo a troca de dados em tempo real. Eles são amplamente utilizados em sistemas distribuídos e aplicações de rede para estabelecer conexões entre servidores e clientes (Attoui, 2000).

Os *websockets*, por outro lado, são uma extensão dos *sockets*, projetados especificamente para comunicação em tempo real na web. Eles superam as limitações das soluções tradicionais de comunicação em tempo real na web, como *polling e long-polling*, fornecendo um mecanismo eficaz para comunicação bidirecional sustentada entre o cliente e o servidor. A tecnologia *websockets* permite uma comunicação mais eficiente, reduzindo o tráfego de rede e a latência, tornando-a ideal para aplicações que exigem interações em tempo real (Liu; Sun, 2012).

2.5 TOKENS E JWT

A autenticação e a autorização são componentes críticos de qualquer aplicação segura. *Tokens*, particularmente JWT (*JSON Web Token*), são usados para transmitir informações de forma segura entre partes. JWT é um padrão aberto (RFC 7519) que define uma maneira compacta e independente de transmitir informações entre partes como um JSON. Essas informações podem ser verificadas e confiáveis porque são assinadas digitalmente. JWT pode ser assinado usando um segredo (com o algoritmo HMAC) ou um par de chaves pública/privada usando RSA ou ECDSA (INTRODUCTION. . . , 2023).

2.6 QR CODE

O código QR (*Quick Response*) é um código de barras bidimensional que pode armazenar informações em um formato legível por máquina. Ele foi desenvolvido para

permitir a leitura rápida e eficiente de informações por dispositivos eletrônicos, como *smartphones e tablets*.

Amplamente utilizado em várias aplicações, o QR Code se tornou muito comum no rastreamento de produtos, gerenciamento de inventário e marketing (QR. . . , 2023). A Figura 4 demonstra um QR que foi gerado com um *link*.

Figura 4 – QR Code com link para loja.

Fonte: Criado pelo autor.

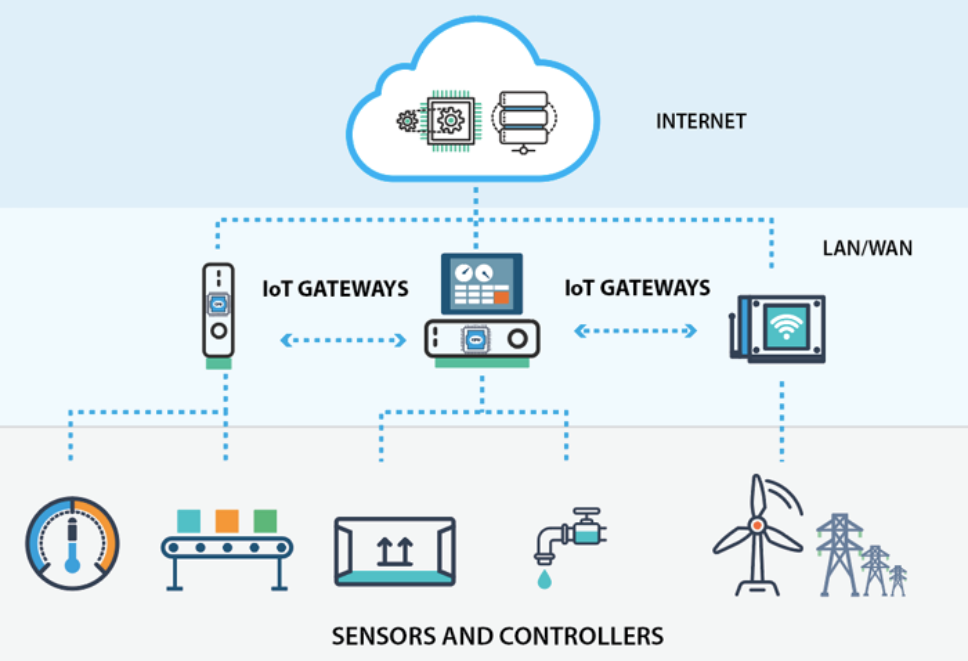
Esta codificação é gerada através de um processo que transforma a informação desejada (como um *link* para um site) em um padrão de pontos pretos e brancos. Este padrão é lido por um *scanner* (geralmente uma câmera de *smartphones* com um aplicativo de leitura de QR Code), que decodifica a informação e realiza a ação correspondente (como abrir um *link* em um navegador).

2.7 INTERNET DAS COISAS (IOT)

IoT é discutido como uma tecnologia emergente que transforma objetos do mundo real em objetos virtuais inteligentes. IoT visa unificar tudo em nosso mundo sob uma infraestrutura comum, não apenas nos dando controle sobre as coisas ao nosso redor, mas também nos mantendo informados sobre o estado dessas coisas (Madakam; Ramaswamy; Tripathi, 2015).

Isso torna possível a criação de dispositivos conectados a internet, capazes de monitorar e controlar processos e equipamentos remotamente. IoT tem um papel crucial na transformação digital e na criação de cidades inteligentes, permitindo a coleta de dados em tempo real e a tomada de decisões baseada em dados. A Figura 5 representa como diversos dispositivos se conectam a internet através da IoT.

Figura 5 – Internet of Things.



Fonte: (IOT..., 2023).

2.8 UUID

Um UUID (*Universal Unique Identifier*) é um identificador único de 128 bits que é usado para identificar informações de forma única em um sistema de computação distribuído (Leach; Mealling; Salz, 2005). Ele é gerado de forma aleatória, tornando-o altamente improvável de ser duplicado. O formato de UUID mais comum é o UUID versão 4, que utiliza a geração aleatória de números para criar uma identificação única. O UUID é amplamente utilizado em sistemas distribuídos, como bancos de dados, sistemas de mensagens e sistemas de arquivos distribuídos (Leach; Mealling; Salz, 2005).

Essa ferramenta permite que cada elemento seja rastreado e gerenciado de forma única, evitando conflitos ou duplicatas no sistema. Além disso, aumenta a segurança, visto que dificulta tentativas, por parte de usuários mal intencionados, de manipular algum elemento do sistema.

2.9 TECNOLOGIAS DA APLICAÇÃO WEB

Neste trabalho, serão utilizadas diversas tecnologias e bibliotecas. As mais importantes são explicadas nas seções a seguir.

2.9.1 TypeScript

TypeScript é uma linguagem de programação que estende o JavaScript, adicionando tipos estáticos. Sua principal motivação é permitir o desenvolvimento de aplicações

JavaScript em larga escala de maneira mais eficiente. O TypeScript introduz um sistema de módulos, classes e interfaces, além de um sistema de tipos gradual e robusto. Essas características permitem que os desenvolvedores escrevam código JavaScript de maneira mais clara e estruturada, facilitando a compreensão e a manutenção do código (Bierman; Abadi; Torgersen, s.d., p. 257).

O TypeScript oferece ferramentas que auxiliam na construção do código, como a capacidade de listar os campos presentes em um objeto ou todos os métodos de uma classe. Isso facilita a navegação e a manipulação do código, especialmente em projetos de grande escala. Além disso, o TypeScript, por meio de seu sistema de tipos estáticos, é capaz de fornecer garantias sobre o comportamento do código, assegurando que os tipos de dados sejam consistentes ao longo do código, o que pode resultar em software mais confiável e de maior qualidade. Essas características tornam o TypeScript uma escolha popular para o desenvolvimento de aplicações web de grande escala, tanto para front-end quanto para back-end.

2.9.2 SQL e Prisma

SQL (*Structured Query Language*) é uma linguagem de programação utilizada para gerenciar e manipular bancos de dados. Ela permite aos usuários criar, ler, atualizar e deletar dados em um banco de dados. SQL é uma linguagem padrão para bancos de dados relacionais e é usada em muitos sistemas de gerenciamento de bancos de dados, como MySQL, Oracle, PostgreSQL e SQL Server. SQL é uma linguagem essencial para desenvolvedores de software, analistas de dados e administradores de banco de dados, pois permite a interação eficiente com os dados armazenados em um banco de dados (SQL..., 2023).

No entanto, trabalhar diretamente com SQL pode ser complexo e propenso a erros. Para resolver isso, os desenvolvedores usam ferramentas chamadas mapeadores objeto-relacional (ORMs). Um ORM permite que os desenvolvedores interajam com o banco de dados usando o paradigma de programação orientada a objetos, o que é mais intuitivo e seguro para muitos desenvolvedores.

O Prisma é um ORM moderno e poderoso que permite o acesso a bancos de dados através de uma interface simples e intuitiva. Ele oferece diversas funcionalidades, como migrações de banco de dados, controle de versão de esquema e consultas otimizadas, permitindo um acesso rápido e eficiente aos dados. O Prisma pode ser usado para acessar bancos de dados de forma eficiente e segura, com suporte para várias funcionalidades, como migrações automáticas, cliente de banco de dados com tipagem segura e navegador de banco de dados visual (PRISMA..., 2023).

2.9.3 Zod

Zod é uma biblioteca para TypeScript que ajuda a garantir que os dados em uma aplicação estejam corretos e seguros. Ele faz isso através do uso de "esquemas", que são como modelos que descrevem como os dados devem ser estruturados, incluindo o tipo de dados e regras. Por exemplo, um esquema pode especificar que um item em uma loja online deve ter um nome (caracteres), uma descrição (caracteres) e um preço (número). O trecho de Código 1 mostra como esse elemento pode ser representado através dessa biblioteca, criando um tipo de dado chamado “itemSchema”.

Código Fonte 1 – Exemplo de uso da biblioteca Zod.

```
1 import z from 'zod';
2
3 export const itemSchema = z.object({
4   id: z.string().uuid(),
5   name: z.string(),
6   description: z.string().min(4),
7   price: z.number(),
8 });
```

Fonte: Criado pelo autor.

Com Zod, é possível definir esses esquemas e usar a biblioteca para verificar se os dados de entrada e saída da aplicação correspondem a esses esquemas. Isso é especialmente útil em aplicações web, onde os dados de entrada geralmente vêm de usuários e podem ser imprevisíveis.

Ao usar Zod para validar esses dados, é possível prevenir muitos erros e vulnerabilidades de segurança, tornando a aplicação mais robusta e confiável (ZOD. . . , 2023).

2.9.4 tRPC

O tRPC (*Typescript Remote Procedure Call*) é uma biblioteca leve que permite a criação de APIs totalmente seguras em termos de tipo (TRPC. . . , s.d.). Ele se apresenta como uma alternativa a outras tecnologias de chamada de procedimento remoto, como REST, GraphQL e gRPC.

Ele permite que os desenvolvedores criem APIs, sem a necessidade de manter manualmente as definições do formato dos dados entre o cliente e o servidor. Isso é conseguido através da geração automática de definições de tipo com base no código do servidor. Em outras palavras, o tRPC entende quais dados devem ser enviados na requisição, com base no código do servidor, eliminando a necessidade de manutenção manual dessas definições (TRPC. . . , s.d.). Complementando o que foi exibido anteriormente com o Zod, o Código 2 demonstra a implementação de uma rota backend que utiliza tRPC e Zod.

O Código 3, corresponde a uma implementação do tRPC no frontend, que utiliza esta rota para enviar informações.

Código Fonte 2 – Exemplo de uso da biblioteca tRPC no backend.

```
1
2  export const itemsRouter = createTRPCRouter({
3    create: publicProcedure
4      .input(
5        z.object({
6          name: z.string(),
7          description: z.string().min(4),
8          price: z.number(),
9        })
10     )
11     .mutation(({ ctx, input }) => {
12       return ctx.prisma.items.create({
13         data: {
14           description: input.description,
15           price: input.price,
16           name: input.name,
17         },
18       });
19     }),
20 });
```

Fonte: Criado pelo autor.

No contexto de desenvolvimento full stack, o tRPC se destaca como uma biblioteca fundamental para a construção tanto do servidor quanto do cliente. De acordo com o trabalho de (Nivasalo, 2022), o tRPC se mostrou uma excelente biblioteca para se basear, pois reduziu significativamente o tempo de desenvolvimento ao tornar extremamente fácil a implementação de chamadas de endpoint da API para o frontend.

Por fim, vale ressaltar que o tRPC é um projeto de código aberto e gratuito para uso, o que o torna uma excelente opção para desenvolvedores e empresas que buscam uma solução eficiente e econômica para a criação de APIs seguras em termos de tipo.

2.9.5 Fastify

Fastify é um *framework* web eficiente para Node.js, projetado para ser o mais rápido possível, tanto em termos de tempo de execução quanto de velocidade de desenvolvimento (FASTIFY:. . . , s.d.). Ele fornece um conjunto robusto de recursos para construir aplicações web e é totalmente extensível com seu sistema de *plugins*. Fastify também oferece um modelo de roteamento fácil de usar e suporte para manipulação de solicitações e respostas HTTP, tornando-o uma escolha popular para muitos desenvolvedores de Node.js.

2.9.6 Next.js

Next.js é um *framework* JavaScript para sistemas baseados em React, que permite a criação de páginas estáticas e dinâmicas, bem como a geração de conteúdo sob demanda, proporcionando uma experiência de carregamento mais rápida e eficiente para o usuário.

Código Fonte 3 – Exemplo de uso da biblioteca tRPC no frontend.

```
export const itemsRouter = createTRPCRouter({
   // importa a rota da api trpc
  const createItem = api.items.create.useMutation({
    onSuccess: (createdItem) => {
      toast.success("Item criado com sucesso", {
        position: "top-right",
        autoClose: 3000,
        theme: "colored",
      });
      },
      onError: (err) => {
      toast.error("Erro ao criar item", {
        position: "top-right",
        autoClose: 5000,
        theme: "colored",
      });
      }
    });

    // utilizacao dela para enviar a requisicao
    createItem.mutate({
        name: values.name,
        description: values.description,
        price: values.price,
      });
```

Fonte: Criado pelo autor.

Este *framework* pode ser usado para criar aplicações web eficientes e escaláveis, com suporte para várias funcionalidades, como roteamento por arquivo, *streaming* de HTML dinâmico e suporte a CSS (NEXT. . . , 2023).

Amplamente utilizado e reconhecido na indústria de desenvolvimento web, o NextJs é o 14º maior projeto no GitHub e é considerado o *framework* ReactJS número 1, com mais de 100.000 estrelas no GitHub. Grandes empresas como Notion e Twitch utilizam o Next.js em suas aplicações, o que demonstra a confiabilidade e a maturidade do *framework* (Vercel, 2023).

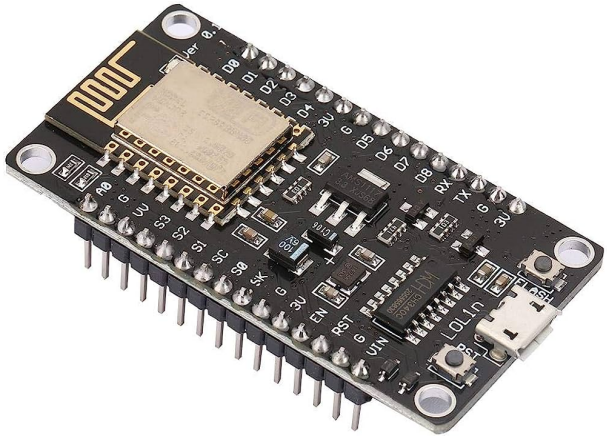
Além disso, o Next.js é recomendado na documentação oficial do React para projetos que se beneficiariam de suas características específicas, como renderização do lado do servidor e otimizações de desempenho. Isso indica a importância e a relevância do Next.js no ecossistema de desenvolvimento web atual. A adoção do Next.js neste trabalho é uma escolha estratégica que visa aproveitar essas vantagens e recursos poderosos para criar uma aplicação web robusta e eficiente.

2.10 NODEMCU ESP8266

O NodeMCU ESP8266 é um microcontrolador baseado na plataforma ESP8266, que possui conectividade Wi-Fi integrada. Ele é bastante utilizado em projetos de IoT

devido à sua facilidade de programação, baixo custo e recursos avançados. O ESP8266, demonstrado na Figura 6, é especialmente adequado para tais aplicações devido à sua capacidade de operar em baixa potência, o que é crucial para dispositivos alimentados por bateria, e sua capacidade de se conectar a redes Wi-Fi, permitindo a comunicação com a internet e outros dispositivos na rede (Kolban, 2016).

Figura 6 – ESP8266 NodeMcu.



Fonte: (NODE. . . , 2023).

2.10.1 C++ e Bibliotecas

O microcontrolador NodeMCU, pode ser programado em C++ e as bibliotecas relevantes para este projeto estão listadas a seguir.

- **ESP8266WiFi.h:** Esta biblioteca permite que o NodeMCU se conecte a redes Wi-Fi. Ela fornece funções para conectar, desconectar e verificar o status da conexão WiFi. No código fornecido, essa biblioteca é usada para conectar o NodeMCU à rede Wi-Fi especificada pelas constantes “ssid” e “password”.
- **WebSocketsClient.h:** Esta biblioteca permite que o NodeMCU se comunique com servidores *websocket*. *Websocket* é um protocolo que permite comunicação bidirecional em tempo real entre clientes e servidores. No código fornecido, essa biblioteca é usada para conectar o NodeMCU a um servidor *websocket* e enviar/receber mensagens.

3 ESCOLHA DOS SENSORES

3.1 SENSORES UTILIZADOS

Os sensores utilizados neste trabalho foram selecionados com base em suas características técnicas, custo e aplicabilidade no monitoramento do nível de rios. Os sensores analisados são:

- **Sensores LiDAR:** TF-Luna, TF-Nova.
- **Sensores Ultrassônicos:** JSN SR04T, HC-SR04.
- **Sensor com Câmera e IA:** ESP-CAM com modelo de inteligência artificial para medição de nível d'água.

Neste capítulo, serão apresentados os três tipos de sensores que foram analisados neste trabalho. A escolha dos sensores foi baseada em suas características técnicas, custo e aplicabilidade no monitoramento do nível de rios. Os sensores selecionados são: sensores ultrassônicos, sensores LiDAR e câmeras. A seguir, serão descritas as principais características de cada um deles.

3.2 SENSORES ULTRASSÔNICOS

Os sensores ultrassônicos são dispositivos que utilizam ondas sonoras para medir distâncias. Eles emitem um pulso de som e medem o tempo que leva para o eco retornar ao sensor. Essa tecnologia é amplamente utilizada em aplicações de automação e robótica, devido à sua precisão e baixo custo.

3.2.1 JSN SR04T

O JSN SR04T é um sensor ultrassônico de alta precisão, projetado para medir distâncias em ambientes externos. Ele possui um alcance de até 4 metros e é resistente à água, o que o torna ideal para aplicações em ambientes úmidos. O sensor é fácil de instalar e pode ser integrado a microcontroladores como Arduino e Raspberry Pi. Ficha técnica:

- Alcance: 20 cm a 4 m
- Precisão: ± 1 cm
- Tensão de operação: 5 V
- Consumo de corrente: 15 mA
- Temperatura de operação: -20 °C a +70 °C
- Umidade de operação: 0% a 100%
- Resistência à água: IP67
- Taxa de atualização: 10 Hz

- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g

3.2.2 HC-SR04

O HC-SR04 é um sensor ultrassônico amplamente utilizado em projetos de robótica e automação. Ele possui um alcance de até 4 metros e é conhecido por sua precisão e baixo custo. O sensor é fácil de usar e pode ser conectado a microcontroladores como Arduino e Raspberry Pi. No entanto, o HC-SR04 não é resistente à água, o que limita sua aplicação em ambientes úmidos. Ficha técnica:

- Alcance: 2 cm a 4 m
- Precisão: ±3 mm
- Tensão de operação: 5 V
- Consumo de corrente: 15 mA
- Temperatura de operação: -15 °C a +70 °C
- Umidade de operação: 0% a 90%
- Resistência à água: Não
- Taxa de atualização: 40 Hz
- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g

3.3 SENSORES LIDAR

Os sensores LiDAR (Light Detection and Ranging) utilizam luz laser para medir distâncias. Eles emitem pulsos de luz e medem o tempo que leva para o eco retornar ao sensor. Essa tecnologia é amplamente utilizada em aplicações de mapeamento e monitoramento ambiental, devido à sua alta precisão e capacidade de gerar dados tridimensionais. Ficha técnica:

- Alcance: 0,1 m a 12 m
- Precisão: ±1 cm
- Tensão de operação: 5 V
- Consumo de corrente: 100 mA
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Taxa de atualização: 100 Hz

- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g
- Tecnologia: Laser de pulso
- Tipo de saída: Serial TTL
- Protocolo: UART
- Interface: UART
- Ângulo de visão: 270°
- Resolução: 1 cm
- Precisão: ±1 cm
- Taxa de amostragem: 100 Hz
- Distância mínima: 0,1 m
- Distância máxima: 12 m
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g

3.3.1 TF-Luna

O TF-Luna é um sensor LiDAR de baixo custo, projetado para aplicações de medição de distância. Ele possui um alcance de até 8 metros e é conhecido por sua precisão e facilidade de uso. O sensor pode ser integrado a microcontroladores como Arduino e Raspberry Pi, tornando-o uma opção popular para projetos de robótica e automação. Ficha técnica:

- Alcance: 0,2 m a 8 m
- Precisão: ±1 cm
- Tensão de operação: 5 V
- Consumo de corrente: 100 mA
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Taxa de atualização: 100 Hz
- Dimensões: 45 mm x 20 mm x 15 mm

- Peso: 20 g
- Tecnologia: Laser de pulso
- Tipo de saída: Serial TTL
- Protocolo: UART
- Interface: UART
- Ângulo de visão: 270°
- Resolução: 1 cm
- Precisão: ±1 cm
- Taxa de amostragem: 100 Hz
- Distância mínima: 0,2 m
- Distância máxima: 8 m
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g
- Tipo de saída: Serial TTL

3.3.2 TF-Nova

O TF-Nova é um sensor LiDAR de alta precisão, projetado para aplicações de mapeamento e monitoramento ambiental. Ele possui um alcance de até 12 metros e é conhecido por sua capacidade de gerar dados tridimensionais. O sensor pode ser integrado a microcontroladores como Arduino e Raspberry Pi, tornando-o uma opção popular para projetos de robótica e automação. Ficha técnica:

- Alcance: 0,1 m a 12 m
- Precisão: ±1 cm
- Tensão de operação: 5 V
- Consumo de corrente: 100 mA
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Taxa de atualização: 100 Hz
- Dimensões: 45 mm x 20 mm x 15 mm

Código Fonte 4 – Exemplo de um schema no Prisma.

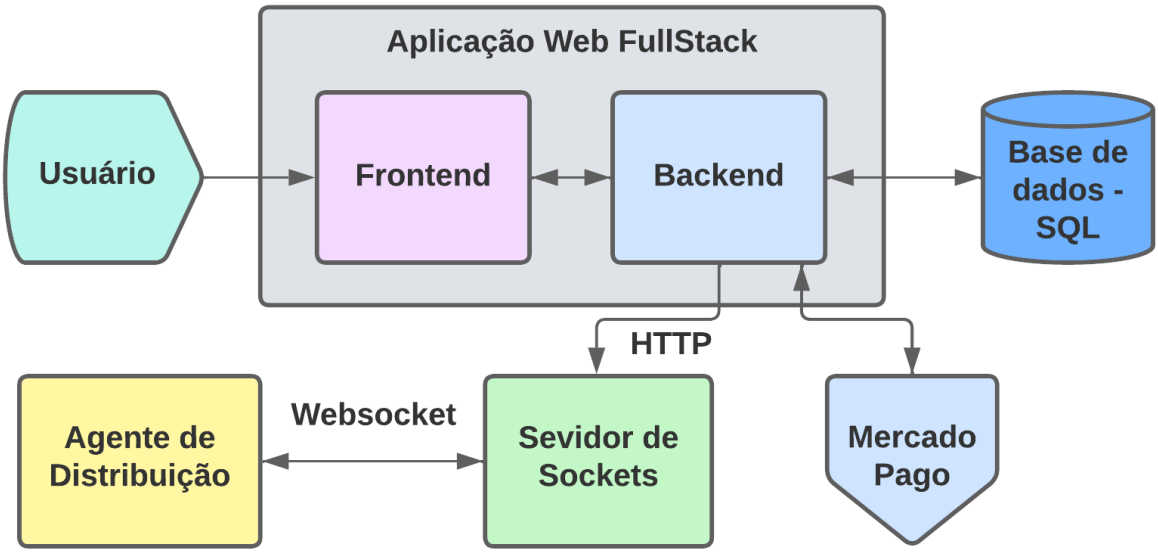
```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id         String        @id @unique @default(uuid())
  email      String
  password   String
  cpf_cnpj   String        @unique
  name       String
  role       RoleEnumType? @default(user)
  createdAt  DateTime      @default(now())
  updatedAt  DateTime      @updatedAt
}

enum RoleEnumType {
  user
  admin
}
```

Fonte: Criado pelo autor.

- Peso: 20 g
- Tecnologia: Laser de pulso
- Tipo de saída: Serial TTL
- Protocolo: UART
- Interface: UART
- Ângulo de visão: 270°
- Resolução: 1 cm
- Precisão: ±1 cm
- Taxa de amostragem: 100 Hz
- Distância mínima: 0,1 m
- Distância máxima: 12 m
- Temperatura de operação: -10 °C a +50 °C
- Umidade de operação: 0% a 95%
- Resistência à água: Não
- Dimensões: 45 mm x 20 mm x 15 mm

- Peso: 20 g
- Tipo de saída: Serial TTL

Figura 7 – Arquitetura do Sistema.



Fonte: Criado pelo autor.

3.4 CÂMERAS

As câmeras são dispositivos que capturam imagens e vídeos. Elas podem ser utilizadas em diversas aplicações, incluindo monitoramento ambiental e reconhecimento de padrões. As câmeras modernas são equipadas com tecnologia de inteligência artificial, permitindo a análise de imagens em tempo real.

3.4.1 ESP-CAM

A ESP-CAM é uma câmera de baixo custo, projetada para aplicações de monitoramento e reconhecimento de padrões. Ela possui um módulo Wi-Fi integrado, permitindo a transmissão de imagens em tempo real. A câmera pode ser programada para realizar tarefas específicas, como detecção de movimento e reconhecimento facial, utilizando modelos de inteligência artificial. A ESP-CAM é uma opção popular para projetos de automação e monitoramento ambiental, devido ao seu baixo custo e facilidade de uso. Ela pode ser integrada a microcontroladores como Arduino e Raspberry Pi, tornando-a uma opção versátil para diversas aplicações. Ficha técnica:

- Resolução: 640 x 480 pixels
- Tensão de operação: 5 V
- Consumo de corrente: 200 mA

- Temperatura de operação: -20 °C a +70 °C
- Umidade de operação: 0% a 100%
- Resistência à água: Não
- Taxa de atualização: 30 fps
- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g
- Tecnologia: Câmera com IA
- Tipo de saída: Serial TTL
- Protocolo: UART
- Interface: UART
- Ângulo de visão: 90°
- Resolução: 640 x 480 pixels
- Precisão: ±1 cm
- Taxa de amostragem: 30 fps
- Distância mínima: 0,1 m
- Distância máxima: 12 m
- Temperatura de operação: -20 °C a +70 °C
- Umidade de operação: 0% a 100%
- Resistência à água: Não
- Dimensões: 45 mm x 20 mm x 15 mm
- Peso: 20 g
- Tipo de saída: Serial TTL

Tabela 2 – Preços dos planos de hospedagem em nuvem.

Plataforma	**Plano**	**Preço**	**Limites**
Vercel	Hobby	Gratuito	Projetos pessoais ou não comerciais. Deploy a partir do CLI ou integrações git. HTTPS/SSL automático.
Vercel	Pro	$20 por membro da equipe por mês.	1TB de banda. 1000 GB/horas de execução.
Vercel	Enterprise	Personalizado.	Infraestrutura de build isolada, em hardware melhor, sem filas.
Railway	Starter	Gratuito	$5.00 em créditos de recursos todos os meses com tempo de execução de 500 horas.
Railway	Hobby	$10 em créditos por mês	8 GB de RAM / 8 vCPU por serviço
Railway	Pro	$20 por mês	Acesso compartilhado ao espaço de trabalho para equipes.
Railway	Enterprise	Personalizado	Personalizado
ElephantSQL	Tiny	Gratuito	20 MB, 5 conexões concorrentes
ElephantSQL	Simple	$5 por mês	500 MB, 10 conexões concorrentes
ElephantSQL	Enormous	$199 por mês	250 GB, centenas de conexões concorrentes

Fonte: Sites das respectivas plataformas.

Tabela 3 – Tabela comparativa entre os sensores.

Sensor	**Tipo**	**Alcance**	**Precisão**
JNS SR04T	Ultrassônico	20 cm a 4 m	±1 cm
HC-SR04	Ultrassônico	2 cm a 4 m	±3 mm
TF-Luna	LiDAR	0,2 m a 8 m	±1 cm
TF-Nova	LiDAR	0,1 m a 12 m	±1 cm
ESP-CAM	Câmera com IA	0,1 m a 12 m	±1 cm

Fonte: Criado pelo autor.

4 METODOLOGIA EXPERIMENTAL

4.1 METODOLOGIA EXPERIMENTAL

O estudo será conduzido em duas etapas principais: **testes em ambiente controlado** e **testes em condições reais**. O objetivo é comparar a precisão e acurácia dos sensores, verificar se suas especificações técnicas estão corretas e analisar seu desempenho em diferentes condições ambientais.

4.1.1 Testes em Ambiente Controlado

Nesta etapa, os sensores serão avaliados em um ambiente estável e previsível, com iluminação adequada e sem interferências externas.

4.1.1.1 Procedimentos

1. **Configuração e Calibração**
    - Instalar cada sensor em um suporte fixo a uma altura conhecida.
    - Calibrar os sensores conforme as especificações do fabricante.
    - Definir uma referência de medição utilizando instrumentos de alta precisão.
2. **Teste de Precisão e Acurácia**
    - Medir distâncias conhecidas e comparar com os valores obtidos.
    - Repetir medições múltiplas vezes e calcular média e desvio padrão.
    - Determinar a acurácia comparando com a referência.
3. **Teste de Limite de Alcance**
    - Posicionar alvos em diferentes distâncias dentro do alcance especificado.
    - Verificar até que ponto cada sensor fornece medições estáveis.
4. **Teste de Sensibilidade a Condições Variáveis**
    - Testar os sensores com diferentes superfícies de medição (água limpa, água com impurezas, materiais reflexivos, etc.).
    - Alterar a inclinação do sensor para verificar variações nas leituras.

4.1.2 Testes em Condições Reais

Os sensores serão instalados em corpos d'água reais para avaliar seu desempenho em longo prazo, considerando fatores ambientais como variações climáticas e interferências externas.

4.1.2.1 Procedimentos

1. **Instalação dos Sensores**
    - Fixar os sensores em estruturas adequadas na margem ou sobre a superfície da água.
    - Garantir posicionamento adequado para evitar interferências externas.
2. **Coleta de Dados em Diferentes Condições**
    - Realizar medições em diferentes horários do dia.
    - Comparar medições em diferentes condições climáticas.
    - Monitorar a estabilidade das leituras ao longo do tempo.
3. **Validação com Medições de Referência**
    - Utilizar uma régua linimétrica ou outro método manual para comparação.
    - Analisar variações e discrepâncias entre os sensores.
4. **Análise da Durabilidade e Confiabilidade**
    - Avaliar possíveis degradações no desempenho devido a sujeira, umidade ou variações ambientais.
    - Medir o consumo de energia ao longo do tempo.

4.2 MÉTRICAS DE AVALIAÇÃO

Os sensores serão comparados com base nos seguintes critérios:

- **Precisão:** Variação das medições em relação ao valor real.
- **Acurácia:** Diferença entre o valor medido e a referência.
- **Estabilidade:** Consistência das medições ao longo do tempo.
- **Interferência Ambiental:** Impacto de fatores externos como iluminação, vento e temperatura.
- **Limite de Alcance:** Distância máxima em que o sensor fornece leituras confiáveis.
- **Confiabilidade:** Desempenho após longo período de operação.

4.3 FLUXOGRAMA DA METODOLOGIA

- **Testes em Ambiente Controlado**
    - Configuração e calibração
    - Teste de precisão e acurácia

  - Teste de limite de alcance
  - Teste com diferentes superfícies

- **Testes em Condições Reais**
  - Instalação dos sensores em corpos d'água
  - Coleta de dados em diferentes condições ambientais
  - Comparação com medições de referência
  - Análise da durabilidade e confiabilidade

- **Análise e Comparação dos Resultados**
  - Comparação entre sensores
  - Identificação de limitações e recomendações
  - Conclusões sobre a melhor tecnologia para monitoramento

5 ARQUITETURA DO SISTEMA

6 IMPLEMENTAÇÃO

7 RESULTADOS

7.1 HOSPEDAGEM E CUSTOS

7.2 CONSIDERAÇÕES FINAIS

8 CONCLUSÃO

A proposta deste projeto foi desenvolver um sistema para automação de pedido e compras em pontos de venda, que permita integração com agentes de distribuição, de forma a automatizar todo o processo.

Com base nos resultados apresentados, pode-se concluir que o desenvolvimento do sistema obteve sucesso. A aplicação web desenvolvida permite que os usuários possam fazer seus pedidos, enquanto os funcionários dos pontos de venda têm acesso a informações precisas sobre os pedidos realizados, podendo acompanhar o status da entrega em tempo real. Além disso, o servidor de *sockets* permite que os pontos de venda possam notificar os agentes de distribuição de qualquer atualização no status dos pedidos, de forma rápida e eficiente.

Por fim, a utilização de serviços de hospedagem em nuvem garantiu a disponibilidade e a escalabilidade do sistema, permitindo que ele possa ser usado por um grande número de usuários ao mesmo tempo.

Dessa forma, pode-se afirmar que o sistema de gerenciamento de pedidos para pontos de venda desenvolvido neste trabalho é uma solução eficiente e viável para empresas que desejam aprimorar seus processos de gerenciamento de pedidos e melhorar a experiência dos seus clientes, usando automação e tecnologias web modernas.

Para trabalhos futuros deseja-se incluir melhorias na interface, principalmente para exibir mais informações sobre os pedidos, e de forma geral torna-la mais amigável. Além disso, outro ponto que pode ser trabalhado seria o desenvolvimento de agentes de distribuição, conectando o NodeMCU com uma máquina de café, por exemplo, e distribuindo realmente os produtos.

REFERÊNCIAS

ADYEN Payment Methods. [S.l.*:* s.n.]. https://www.adyen.com/payment-methods. Accessed: 30-05-2023.

ATTOUI, Ilham. **Real-Time and Multi-Agent Systems**. [S.l.]: Springer, 2000.

AUÉ, Joop; ANICHE, Maurício; LOBBEZOO, Maikel; DEURSEN, Arie van. An exploratory study on faults in web API integration in a large-scale payment company. In*:* ACM. PROCEEDINGS of the 40th International Conference on Software Engineering. [S.l.*:* s.n.], 2018. p. 372–383.

BIERMAN, Gavin; ABADI, Martín; TORGERSEN, Mads. Understanding TypeScript, 2014. In*:* ECOOP. EUROPEAN Conference on Object-Oriented Programming. [S.l.*:* s.n.]. p. 257–281.

ELHADEF, M.; GRIRA, S. Performance Evaluation of a Game Theory-Based Fault Diagnosis Algorithm Under Partial Comparison Syndrome. In*:* 2016 IEEE International Conference on Computer and Information Technology (CIT). [S.l.*:* s.n.], 2016. p. 1–6.

FASTIFY: Fast and low overhead web framework, for Node.js. [S.l.*:* s.n.]. https://www.fastify.io/. Acessado em: 20 de junho de 2023.

GOURLEY, David; TOTTY, Brian. **HTTP: The Definitive Guide**. [S.l.]: O'Reilly Media, 2002.

HTTP An overview of HTTP | MDN. [S.l.*:* s.n.]. https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview. Acessado em: 20 de junho de 2023.

INTRODUCTION to JSON Web Tokens. [S.l.]: JWT.io, 2023. Disponível em: https://jwt.io/introduction/.

IOT Architecture. [S.l.*:* s.n.], 2023. https://www.zibtek.com/blog/iot-architecture. Acessado em: 26 de junho de 2023.

KOLBAN, N. **Kolban's book on ESP8266**. [S.l.]: Leanpub, 2016.

LEACH, P.; MEALLING, M.; SALZ, R. **A Universally Unique IDentifier (UUID) URN Namespace**. [S.l.*:* s.n.], 2005. Disponível em: https://www.rfc-editor.org/info/rfc4122.

LIU, Qian; SUN, Xiaojun. Research of Web Real-Time Communication Based on Web Socket. **International Journal of Communications, Network and System Sciences**, v. 5, n. 12, p. 841–846, 2012.

MADAKAM, Somayya; RAMASWAMY, R.; TRIPATHI, Siddharth. Internet of Things (IoT): A Literature Review. **Journal of Computer and Communications**, Scientific Research Publishing, v. 3, n. 5, p. 164–173, 2015.

MULLOY, Brian. **Web API Design: The Missing Link**. [S.l.*:* s.n.], 2012. Disponível em: https://pages.apigee.com/rs/apigee/images/api-design-ebook-2012-03.pdf.

NEXT.JS Documentation. [S.l.]: Vercel, 2023. Disponível em: https://nextjs.org/docs.

NIVASALO, Markus. **Full Stack TypeScript Development with tRPC and React Native**. 2022. Disponível em: https://www.theseus.fi/handle/10024/793117.

NODE MCU. [S.l.*:* s.n.], 2023. https://www.amazon.com.br/ESP8266-CH340G-NodeMcu-desenvolvimento-Internet/dp/B08H26NY16. Acessado em: 26 de junho de 2023.

PRISMA Documentation. [S.l.]: Prisma, 2023. Disponível em: https://www.prisma.io/docs/.

QR Code Tutorial. [S.l.]: W3Schools, 2023. Disponível em: https://www.w3schools.com/whatis/whatis_qrcode.asp.

RICHARDSON, Leonard; RUBY, Sam. **RESTful Web Services**. [S.l.]: O'Reilly Media, 2007.

SQL Tutorial. [S.l.]: W3Schools, 2023. Disponível em: https://www.w3schools.com/sql/.

TRPC Documentation - End-to-end typesafe APIs. Accessed: 30-05-2023. Disponível em: https://trpc.io/.

VERCEL. **AWS and Vercel: Accelerating innovation with serverless computing**. 2023. Disponível em: https://vercel.com/blog/aws-and-vercel-accelerating-innovation-with-serverless-computing.

ZOD Documentation. [S.l.]: NPM, 2023. Disponível em: https://www.npmjs.com/package/zod.