INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

FELIPE MIEHE PEREIRA

SISTEMA DE DETECÇÃO DE VAZAMENTOS NA REDE PLUVIAL RESIDENCIAL ATRAVÉS DE APRENDIZADO SUPERVISIONADO

FLORIANÓPOLIS, 2024.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA CÂMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA
CURSO DE GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

FELIPE MIEHE PEREIRA

SISTEMA DE DETECÇÃO DE VAZAMENTOS NA REDE PLUVIAL RESIDENCIAL ATRAVÉS DE APRENDIZADO SUPERVISIONADO

Monografia submetida ao Instituto Federal de Educação, Ciência e Tecnologia no Câmpus de Florianópolis, como parte dos requisitos para a obtenção do título de Bacharel em Engenharia Mecatrônica.

Orientador:
Prof. Dr. Eng. Valdir Noll.

FLORIANÓPOLIS, 2024.

Ficha de identificação da obra elaborada pelo autor.

Pereira, Felipe Miehe
SISTEMA DE DETECÇÃO DE VAZAMENTOS NA REDE PLUVIAL
RESIDENCIAL ATRAVÉS DE APRENDIZADO SUPERVISIONADO /
Felipe Miehe Pereira; orientação de Valdir Noll.
- Florianópolis, SC, 2024.
106 p.

Trabalho de Conclusão de Curso (TCC) **-** Instituto Federal
de Santa Catarina, Câmpus Florianópolis. Bacharelado
em Engenharia Mecatrônica. Departamento
Acadêmico de Metal Mecânica.
Inclui Referências.

1. Detectar vazamentos. 2. Aprendizado de máquinas.
3. Automação residencial. 4. Algoritmo KNN. 5. Controle.
I. Noll, Valdir. II. Instituto Federal de Santa Catarina.
III. SISTEMA DE DETECÇÃO DE VAZAMENTOS NA
REDE PLUVIAL RESIDENCIAL ATRAVÉS DE APRENDIZADO SUPERVISIONADO
.

SISTEMA DE DETECÇÃO DE VAZAMENTOS NA REDE PLUVIAL RESIDENCIAL ATRAVÉS DE APRENDIZADO SUPERVISIONADO

FELIPE MIEHE PEREIRA

Este trabalho foi julgado adequado para obtenção do título de Bacharel em Engenharia Mecatrônica e aprovado na sua forma final pela banca examinadora do Curso de Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 26 de setembro de 2024.

Banca Examinadora:

gov.br Documento assinado digitalmente
VALDIR NOLL
Data: 26/09/2024 20:23:03-0300
Verifique em https://validar.iti.gov.br

Valdir Noll, Dr.Eng
IFSC - Campus Florianópolis

gov.br Documento assinado digitalmente
ROBERTO ALEXANDRE DIAS
Data: 26/09/2024 13:05:02-0300
Verifique em https://validar.iti.gov.br

Roberto Alexandre Dias, Dr.Eng
IFSC - Campus Florianópolis

gov.br Documento assinado digitalmente
RENAN AUGUSTO STARKE
Data: 26/09/2024 13:39:49-0300
Verifique em https://validar.iti.gov.br

Renan Augusto Starke, Dr.Eng
IFSC - Campus Florianópolis

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema para monitorar o consumo de água, com o objetivo de detectar vazamentos de forma precoce e evitar desperdícios. O sistema utiliza acionamentos automatizados e mede o fluxo de água por efeito Hall, incorporando conceitos de Internet das Coisas e casas inteligentes, além de modelos de aprendizado de máquina. Um protótipo foi desenvolvido para medir a vazão de água em sistemas pluviais residenciais, permitindo a detecção de vazamentos antes e depois do reservatório. O sistema possibilita o acionamento remoto, automático e manual de uma válvula de retenção, utilizando como principais componentes um sensor de vazão por efeito Hall e um circuito impresso que minimiza ruídos. Este circuito regula a tensão e se conecta à ESP 32, enquanto a válvula esférica elétrica é acionada por um relé, operando em dois estados: normal aberto e normal fechado. Um Orange Pi Zero 3 atua como servidor para o Home Assistant, executando um modelo em Python para avaliar o consumo.Testes foram realizados para medir a vazão ao longo do tempo e treinar o modelo de detecção de vazamentos. O protótipo promove a automação residencial e assegura que a rede instalada não apresente vazamentos, resultando em economia de recursos financeiros e naturais, além de oferecer uma interface visual para gerenciar o consumo.

Palavras-chave: Detectar vazamentos, IOT, Automação residencial, Aprendizado de máquinas, Efeito *hall*.

ABSTRACT

This work presents the development of a system for monitoring water consumption, focusing on the early detection of leaks to prevent waste. The system employs automated controls and measures water flow using Hall effect sensors, integrating concepts from the Internet of Things and smart homes, as well as machine learning models. A prototype was developed to measure water flow in residential rainwater systems, enabling the detection of leaks both before and after the reservoir.The system allows for remote, automated, and manual operation of a check valve, featuring key components such as a Hall effect flow sensor and a printed circuit board designed to minimize noise. This circuit regulates voltage and connects to the ESP32, while an electric ball valve is controlled by a relay that operates in two states: normally open and normally closed. An Orange Pi Zero 3 acts as the server for Home Assistant, running a Python model to evaluate consumption.Tests were conducted to measure flow over time and train the leak detection model. The prototype enhances home automation and ensures that the installed network is leak-free, resulting in savings of both financial and natural resources, while also providing a visual interface for managing consumption.

Keywords: Detecting Leaks, IOT, Home Automation, Machine Learning, Hall Effect.

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

CASAN	Companhia Catarinense de Águas e Saneamento
MOM	Middleware Orientado a Mensagem
MQTT	Message Queuing Telemetry Transport(Transporte de telemetria de enfileiramento de mensagens)
IoT	*Internet of Things* (Internet das Coisas)
SOA	*arquitetura orientada a serviços*
REST	Representational State Transfer(Transferência de estado representacional)
API	*Application Programming Interface (Interface de Programação de Aplicação)*
KNN	*K-Nearest Neighbors(k-vizinhos mais próximos)*
NO	*Normal aberto*
NC	*Normal fechado*
SQL	*Structured Query Language(linguagem de consulta estruturada)*
COM	*Comum*
FreeRTOS	*Sistema Operacional Livre em Tempo Real*
CPU	*Unidade central de processamento*
HTTP	*Protocolo de Transferência de Hipertexto*
ARM	*Advanced RISC Machine (Máquina RISC Avançada)*
BIT	*Binary digit(dígito binário)*
MHz	*Unidade de medida de frequência (símbolo: MHz)*
Wi-Fi	*wireless field(area sem fio)*

V	*Volt*
IP	*Internet Protocol (protocolo de rede)*
VDC	*Voltagem em corrente contínua*
VAC	*Voltagem em corrente alternada*
SPI	*Serial Peripheral Interface(conversor serial/paralelo)*
I2C	*Inter-Integrated Circuit(Circuito Interintegrado)*
RAM	*Memória de Acesso Aleatório*
GPIO	*Pinos de entrada/saída de uso geral*
KB/MB/GB	*Unidade de armazenamento digital*
JSON	*Java Script Object Notation*
SoC	*System on a Chip*

SUMÁRIO

1 INTRODUÇÃO

No momento atual é comum o abastecimento de água ser concedido a uma empresa, que faz o tratamento e distribuição para as residências e condomínios, passando ser um direito de cada residência ligar sua instalação hidráulica junto a rede da companhia. As redes de abastecimentos são de grande maioria controlados pelas mesmas, porém, a rede dentro da residência é de responsabilidade do proprietário de acordo com o manual do usuário da CASAN (2015). Dentre essas responsabilidades, deve-se evitar que a instalação pluvial não esteja com vazamentos, que por sua vez os meios que existem para se precaver sejam muito manuais ou quando se recebe uma conta elevada de água passando muito tempo com o vazamento na rede. De acordo com a matéria da revista Metrópoles, 39,2% da água distribuída no país não chegam a residências sendo que 60% são de redes domésticas(Dino,2022), então é notável a necessidade de meios para reduzir esses números.

Portanto, esse trabalho foi desenvolvido para oferecer uma solução que consiga detectar vazamentos precoces por meio de aquisição e avaliação em tempo real com modelos de treinamento de máquina. Além disso, aplica um sistema de casas inteligentes. Possibilitando envio automáticos de possíveis vazamento e visualização gráfica do consumo ao longo do tempo.

1.1 Justificativa

Existe uma crescente busca por aprimorar a eficiência e qualidade em qualquer tipo de operação, procurando economizar e reduzir os custos associados. No contexto residencial, onde há uma significativa demanda por água, vazamentos que não foram detectados e que permanecem despercebidos por longos períodos, por um ou dois meses, podem resultar em desperdício e despesas elevadas com o consumo de água. Esses custos excessivos podem se tornar insustentáveis a longo prazo, tanto para residências comuns, quanto para condomínios e seus condôminos. Portanto, é necessário buscar soluções inovadoras para resolver esse tipo de problema.

O desenvolvimento de uma solução, mesmo que inicialmente baseada em uma residência comum, tem o potencial de ser validada e transformada em um

produto desejado por condomínios, já que ela poderá oferecer a detecção precoce de vazamentos.

Ainda para reforçar a justificativa, a matéria da revista metrópoles (Dino,2022) afirma que:

> Vazamentos em tubulações internas de imóveis são comuns de ocorrer, mas muitas vezes difíceis de serem identificados, ocasionando grande desperdício de água e de dinheiro. De acordo com uma pesquisa realizada pelo Instituto Trata Brasil, divulgada em junho de 2021, referente ao ano de 2019, 39,2% de toda a água potável distribuída no país não chega às residências do país, o que representa um volume equivalente a 7,5 mil piscinas olímpicas de água tratada desperdiçadas diariamente – segundo o levantamento, cerca de 60% destas perdas são decorrentes de vazamentos.

Portanto, é significativa a economia de recursos naturais e financeiros na detecção de vazamentos de água precocemente.

1.2 Definição do Problema

Como podemos realizar a aquisição de dados de consumo de água em residências, detectar vazamentos precoces e ocultos por meio de métodos matemáticos, notificar os moradores e evitar contas de água elevadas?

1.3 Objetivo Geral

O objetivo principal é desenvolver uma ferramenta que possibilite o monitoramento de consumo de água, que seja capaz de capturar e enviar dados para um modelo de aprendizado de máquina avaliar se está havendo um vazamento ou não na rede pluvial. Será realizado por meio de um sistema embarcado em microcontrolador, utilizando o conceito Internet das Coisas (IoT). A aquisição do fluxo de água se dará por meio de um sensor de vazão tipo turbina baseado no efeito hall. Os dados coletados serão comparados com modelos previamente treinados sobre o consumo da rede, com a finalidade de fornecer uma análise de consumo de água, alarmes e desligamento automático.

O protótipo exibirá um painel com informações relevantes, incluindo o consumo diário de água. Se houver consumo excessivo diário, fora dos limites permitidos calculados com base em dados previamente estabelecidos, o sistema

emitirá um aviso, indicando a possível existência de vazamentos na rede de abastecimento de água, além de desligar a entrada de água.

1.3.1 Objetivos Específicos

a) Desenvolvimento de um Banco de Dados temporal SQL;
b) Comunicação, via Protocolo MQTT: Desenvolver um sistema que possa consultar e transmitir os dados dos dispositivos de hardware por meio do protocolo MQTT;
c) Elaborar uma interface visual para visualização de dados que exiba, de forma gráfica, os dados coletados, proporcionando uma interface de monitoramento amigável e acessível, e enviando avisos sobre possíveis vazamentos;
d) Bloquear o fornecimento remotamente e automaticamente, caso detecte-se essa necessidade.

1.4 Estrutura do Trabalho

O TCC está organizado em capítulos. O primeiro capítulo apresenta a fundamentação teórica, incluindo trabalhos relacionados, investigações e pesquisas sobre sistemas de medição de fluxo de água. Também aborda conceitos e protocolos de comunicação, além de comparar suas vantagens para o protótipo. Há também a avaliação de sensores e atuadores em questão de benefícios para o projeto e estudos dos conhecimentos essenciais para execução do trabalho e suas grandezas físicas mensuradas.

O próximo capítulo é a estrutura metodológica dividida em metodologia e desenvolvimento do projeto, que cita componentes utilizados para a aquisição dos resultados, escolhas dos protocolos, informações sobre circuitos eletrônicos e seu funcionamento, conexões entre os componentes, programação e comunicação entre os dispositivos e servidor local, bem como o desenvolvimento do protótipo.

No capítulo de resultados ocorre a sintetização dos dados obtidos e coletados durante os testes com o protótipo desenvolvido, estão destacadas as principais reflexões e conclusões da síntese realizada, possibilitando ter uma base de pensamentos sobre o protótipo e sua eficiência e precisão.

Em suma, o capítulo de conclusão aborda relações entre resultados e expectativas em relação ao protótipo, comentários sobre os objetivos alcançados, aprimoramentos e acréscimo de possíveis melhorias e novas funcionalidades.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, mostra-se a pesquisa e revisão da literatura, possibilitando uma base para o desenvolvimento do trabalho.

2.1 Distribuição da água

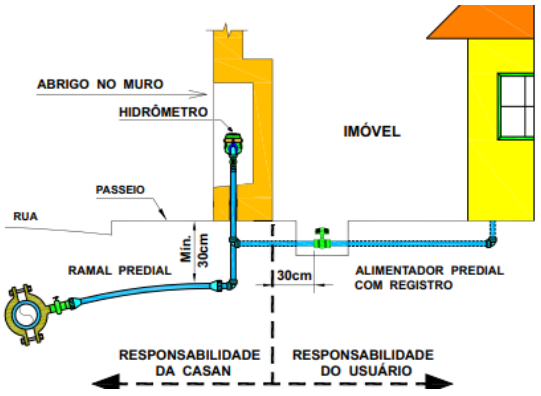
O Manual do Usuário da CASAN (2015) estrutura a distribuição de água em infraestruturas, instalações e serviços com finalidade de abastecimento de água e esgotamento sanitário em Santa Catarina. Sua missão é fornecer água tratada e coletar e tratar esgotos sanitários, promovendo saúde, conforto, qualidade de vida e desenvolvimento sustentável.

O sistema da CASAN é constituído por:

- Captação;
- Tratamento;
- Reservação;
- Distribuição.

A forma em que a CASAN exige e cobra pelos valores referentes à distribuição de água precisa seguir o modelo de instalação, com hidrômetro e abrigo, conforme a Figura 1, e de acordo com o manual do usuário da CASAN (2015). Na Figura 1 pode-se ver quem é o responsável pelos encanamentos, ou seja, o usuário é responsável pela sua ligação depois do hidrômetro, e se houver qualquer vazamento o próprio usuário será responsabilizado.

Figura 1 - Padrão técnico de ligação



Fonte: CASAN(2015)

2.2 Definição vazão

Para Delmée (1995, p. 22) para conseguir estabelecer as leis da hidráulica, usa-se elementos geométricos que simulam as condições reais que se efetua o fluxo, para assim aplicar leis mecânicas, que ele chama de filete líquido. Ainda afirma que o filete líquido só obtém movimentos constantes e ordenados quando estão em contato com superfícies lisas os tangenciando por exemplo, tubulações. A posição dos filetes em relação ao tempo varia, sendo que alguns filetes estão mais à frente e outros mais atrás enquanto fluem em uma tubulação e são chamados de linhas de correntes. Supondo que as linhas de correntes preencham todo o tubo desprezando os efeitos de contração e diferença de posição entre os filetes, que são pequenas, chega-se na Equação 1.

$$Q = S_1 \cdot V_1 \tag{1}$$

Sendo:
$S_1 - \text{seção do tubo qualquer } [m^2]$
$V_1 - \text{velocidade do líquido que atravessa a seção } [\frac{m}{s}]$
$Q - \text{vazão } [\frac{m^3}{s}]$

A variação do volume sobre a variação do tempo, em um tubo, também representa a vazão de um líquido fluindo entre ele, representada na Equação 2. (Brunetti, 2003, p. 72)

$$Q = \frac{\Delta v}{\Delta t} \tag{2}$$

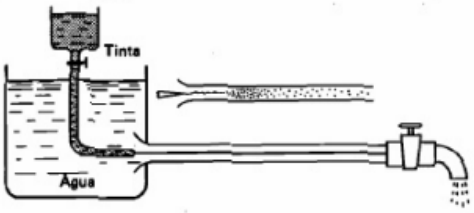
Sendo:
$\Delta v - \text{variação do volume } [m^3]$
$\Delta t - \text{variação do tempo } [s]$
$Q - \text{vazão } [\frac{m^3}{s}]$

As unidades de medidas buscam sempre esclarecer as leituras de vazão, podendo ser em vazão mássica em (Kg/h) ou vazão volumétrica em (m^3/s). Pode ser representado por qualquer unidade de tempo, por exemplo litros por minuto.

O escoamento de um fluido numa tubulação possui dois regimes, respectivamente, laminar e turbulento, sendo o regime laminar um escoamento em camadas planas ou concêntricas, depende do formato do tubo, sem passagem de partículas entre elas e não variam a velocidade, ou seja velocidade constante, já o escoamento turbulento as partículas das camadas se misturam entre si e existe uma intensa variação de pressão e velocidade, gerando trajetos complicados e sem orientações. (Delmée. 1995, p. 27), ainda classifica que assim para ambos regimes a equação 1 consegue uma boa representação da vazão e para melhor precisão deve-se considerar o calor específico, viscosidade, seu regime específico de acordo com o valor de Reynolds, que é uma constante que representa a variação dos regimes dependendo da viscosidade, diâmetro do tubo e velocidade. Para valores inferiores a 2000 considera-se regime laminar, e entre 2000 e 4000 considera-se uma região de transição de regimes, e acima de 4000 regimes turbulentos.

Na Figura 2, mostra o experimento que Delmée fez para obter regimes turbulentos e laminares. A tinta escoa pela torneira com uma abertura pequena, e fica em regime laminar e ordenada. Já quando a abertura aumenta, a água causa uma turbulência e acaba misturando as partículas gerando um escoamento turbulento.

Figura 2 – experimento de regimes laminar e turbulentos



Fonte: Delmée **(**1995. p.27**)**

2.3 Pressão em um tubo

A pressão é uma força aplicada sobre uma superfície que pode ser decomposta em dois efeitos, um tangencial que origina tensões de cisalhamento e

outra a força normal que dá origem à pressão, a força em si não pode ser confundida com a pressão, pois em muitos casos a força aplicada é a mesma porém a área varia, ocasionando diferentes pressões. Se a pressão for uniforme em toda área pode ser representada pela Equação 3 (Brunetti, 2003, p. 18).

$$p = \frac{Fn}{A} \tag{3}$$

Sendo:

$Fn - força\ [N]$

$A - área\ da\ seção\ [m^2]$

$p - pressão[\frac{N}{m^2}]$

Em grandezas no conhecimento popular pressão em (N/m^2) pode assumir a unidade de medida em $Pascal(Pa)$, que é a unidade internacional de medida para pressão que equivale a força em *Newton* sobre a superfície em metros quadrados.

Blaise Pascal (1623-1662) era um físico, matemático e escritor francês, se dedicou a experimentos de pressão atmosférica, inventou a prensa hidráulica a seringa e aperfeiçoou o barômetro, ganhando assim a homenagem no sistema internacional de medidas com seu nome, representando a pressão (Frazão, 2023).

2.4 Instrumentação

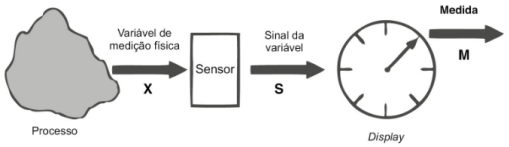
A instrumentação é uma área multidisciplinar que estuda e aplica instrumentos para medição e controle de processos em diversos contextos. Segundo Franchi (2003), essa ciência é fundamental para garantir a precisão e eficiência em operações que vão desde o controle de temperatura em residências até sistemas complexos em indústrias e reatores nucleares.

Os instrumentos de medição são essenciais para converter variáveis físicas, como temperatura, pressão e fluxo, em dados que podem ser analisados. Muitas vezes chamados de sensores, esses dispositivos permitem a automação e otimização de processos, possibilitando a coleta contínua de informações e ajustes em tempo real.

Além disso, a instrumentação envolve o desenvolvimento de protocolos de comunicação que asseguram a transmissão eficaz dos dados coletados. Esses protocolos são cruciais para a integração de sistemas de medição em ambientes conectados, como os encontrados em casas inteligentes e na Internet das Coisas (IoT). Dessa forma, a instrumentação não apenas melhora a eficiência operacional, mas também contribui para decisões informadas e soluções sustentáveis.

Na Figura 3, a variável física é medida pelo sensor, que traduz essas características e transforma ela um um sinal de variável do tipo apropriado para conseguir leituras através de circuitos elétricos, em sua maioria, para uma leitura e armazenamento direto do valor por computadores (FRANCHI, 2003). Para Neri (2022, p. 3), os instrumentos são classificados por função e a associação desses instrumentos se denomina malha.

Figura 3 – Modelo simplificado de um instrumento



Fonte: FRANCHI **(**2003. p.2**)**

O termo "sensor" designa dispositivos que são sensíveis a alguma forma de energia do ambiente, como energia elétrica, térmica ou cinética. Esses dispositivos relacionam uma grandeza a ser medida. Nem sempre as propriedades elétricas de um sensor são bem definidas, o que pode exigir a aplicação de amplificações e tratamentos nos valores obtidos.

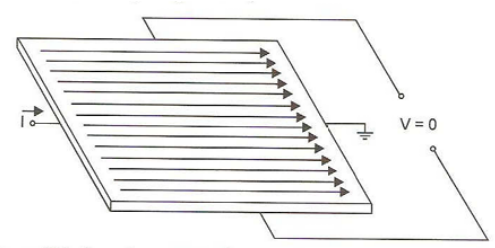
Existem sensores digitais que têm saídas que assumem apenas dois valores. Embora esses sensores não representem diretamente as grandezas físicas, seus dados são convertidos por circuitos eletrônicos, geralmente transdutores. Esses transdutores transformam uma grandeza física em um sinal de tensão ou corrente.

Por outro lado, sensores analógicos geram valores que representam suas grandezas no sinal de saída ao longo do tempo (THOMAZINI, 2005).

Os sensores de efeito hall foram descobertos por Edward E. Hall por volta de 1879, são constituídos por dispositivos semicondutores sensíveis à campo

magnéticos, onde por uma placa condutora passa uma corrente e perpendicularmente forma um campo magnético, que faz gerar uma diferença de potencial nas laterais da placa condutora, o qual se conecta a um circuito de medição, esse diferença de potencial é chamada de efeito Hall THOMAZINI(2005. p.173). O efeito Hall existe em qualquer material condutor, porém é mais intenso em semicondutores e ele possui uma grande vantagem que é medir campos contínuos e campos alternados em um único instrumento e seu funcionamento é bem interessante. De início uma placa com um material semicondutor, de silício ou germânio, percorre uma corrente de forma constante no sentido convencional do polo positivo para o negativo como a Figura 4, e sua distribuição de corrente é uniforme THOMAZINI(2005. p.173).

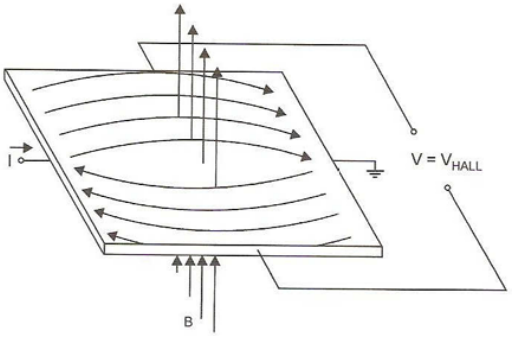
Figura 4 – Corrente constante em um material condutor



Fonte: THOMAZINI**(**2005. p.173**)**

Para existir uma diferença de potencial, o semicondutor sensível a campos magnéticos, é exposto a um campo perpendicularmente a ele, o fluxo da corrente que era constante sofre uma distorção, aparecendo assim uma diferença de potencial entre os terminais, Figura 5 representa a ideia. THOMAZINI(2005. p.174).

Figura 5 – Corrente constante em um material condutor



Fonte: THOMAZINI**(**2005. p.174**)**

THOMAZINI(2005. p.174) define que.

> Uma equação que descreve superficialmente a interação entre campo magnético, corrente e tensão Hall é:

$$V_{hall} = k \cdot I \cdot B \cdot sen\theta \quad (4)$$

Sendo:

$Vhall - Volts\ [V]$
$k - constante\ da\ sensibilidade$
$I - corrente\ [A]$
$B * sen\theta - componente\ do\ campo\ magnético\ [T]$

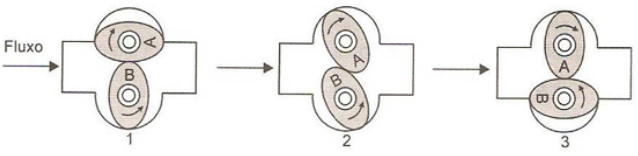
O campo magnético é uma propriedade física que descreve a influência em determinada região no espaço, que pode ser gerado por correntes elétricas em movimento ou a presença de um ímã (TIPLER, 2009).

2.5 Métodos de medição de fluxo de água

Para THOMAZINI (2005. p.147), não existe um só princípio de medição de vazão que atende todas as especificações de uma aplicação, alguns são limitados por pressão, temperatura e outros fatores. Então precisa-se analisar bem qual método é mais vantajoso, levando em consideração preço, faixa de trabalho, precisão e velocidade da resposta.

No método de engrenagens ovais conhecido como método de deslocamento positivo funciona com o movimento das engranagens internas, geralmente pares que se movem na medida que o fluido percorre a encanação, bem simples porém é necessário que a engrenagem que será empurrada por outra tenha seus dados de deslocamento conhecidos para determinar a vazão, conforme a Figura 6 THOMAZINI (2005. p.165).

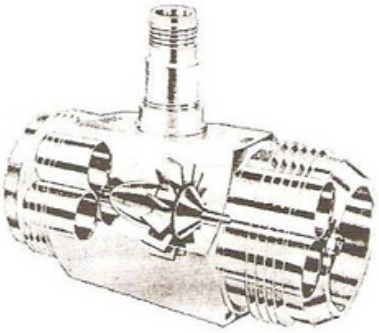
Figura 6 – engrenagens ovais funcionamento



Fonte: THOMAZINI(2005. p.165)

O método de medição por vazão de turbina é utilizado apenas para líquidos. Nesse sistema, uma turbina é fixada dentro da tubulação, onde também se encontra um ímã permanente. Quando o líquido passa, a turbina se movimenta. O rotor da turbina induz uma tensão alternada com frequência variável nos terminais de uma bobina externa feita de material magnético. Essa frequência é proporcional à velocidade do líquido e, consequentemente, à vazão. A Figura 7 ilustra uma turbina (THOMAZINI, 2005, p. 154).

Figura 7 – Turbina



Fonte: THOMAZINI**(**2005. p.154**)**

O hidrômetro utilizado pela companhia CASAN é responsável por medir o consumo de água. Ele combina dois métodos mencionados anteriormente: acoplamentos magnéticos e engrenagens ovais. O sistema é composto por uma turbina com um ímã permanente e um conjunto de engrenagens, que também possui um ímã permanente.

O funcionamento do hidrômetro ocorre quando a turbina se movimenta. Nesse processo, o ímã gira, repelindo o outro ímã que está com o polo invertido. Isso faz com que as engrenagens girem, marcando o deslocamento do fluido.

2.6 Algoritmos de aprendizados de máquinas

Segundo Brandão (2023), que cita KINDERSLEY (2020, p. 301), as técnicas de aprendizado de máquina, um ramo da inteligência artificial, aprendem por tentativa e erro, enquanto os sistemas especialistas utilizam um banco de dados de conhecimento fornecido por programadores. O aprendizado de máquina permite adquirir novos conhecimentos a partir da experiência e realizar tarefas que não

estão explicitamente definidas em suas instruções programadas (RAHMAN, 2022). De forma mais técnica, Brandão (2023) afirma que, segundo MITCHELL (1997), um programa de computador é considerado capaz de aprender quando seu desempenho em uma tarefa melhora com a experiência acumulada. Isso significa que o aprendizado é um processo contínuo. Contudo, a maioria dos algoritmos de aprendizado de máquina é considerada como sistemas especialistas, pois se especializa em tarefas específicas e não aprende além delas, carecendo de consciência sobre suas limitações.

Vários sistemas especialistas desenvolvidos através do treinamento de máquina utilizam algoritmos essenciais, que consistem em modelos estatísticos e matemáticos previamente definidos (Brandão, 2023). Esses algoritmos são utilizados para generalizar os dados e embasar as decisões. Entre os exemplos de algoritmos essenciais, destacam-se *k-Nearest Neighbors(KNN)*, *Naive Bayes*, máquinas de vetores de suporte, redes neurais entre outros. Os algoritmos estão separados de três modelos distintos de treinamento sendo, aprendizado supervisionado, aprendizado não supervisionado e aprendizagem por reforço (Ali, 2022).

No aprendizado supervisionado, a fase de treinamento inclui não apenas o conjunto de dados, mas também informações relevantes chamadas de etiquetas ou rótulos. Essas informações indicam a solução para cada exemplo, permitindo que o modelo aprenda a identificar padrões e relações, além de fornecer um *feedback* sobre seu nível de precisão (Brandão, 2023).

O aprendizado não supervisionado utiliza algoritmos de machine learning para analisar e agrupar dados não rotulados. Esses algoritmos identificam padrões ocultos e agrupamentos sem intervenção humana, tornando-se ideais para análise exploratória de dados. Por exemplo, em marketing, ele pode ser usado para segmentar clientes em grupos com comportamentos de compra semelhantes, ajudando as empresas a personalizar suas estratégias (Ali, 2022). Esse tipo de aprendizado permite abordar problemas sem uma ideia clara de como os resultados devem se parecer e pode ser utilizado para reduzir o número de dimensões em um conjunto de dados, concentrando-se apenas nos atributos mais relevantes ou para detectar tendências. É importante notar que, no aprendizado não supervisionado, não há *feedback* com base nos resultados das previsões (BARROS, 2016).

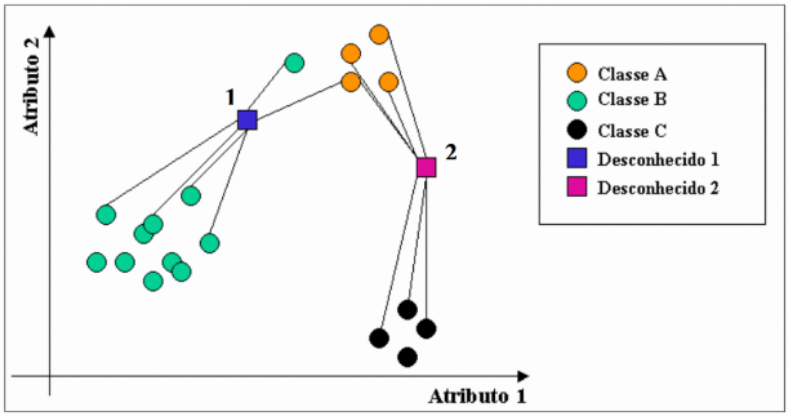
No aprendizado por reforço, o algoritmo atua como um agente que realiza ações em um ambiente externo. Com base nas respostas que recebe, ele avalia se está se aproximando ou se afastando do objetivo desejado. Assim, o agente ajusta seu comportamento para seguir a melhor estratégia, maximizando os estímulos positivos ao longo do processo (Ali, 2022). Um exemplo é avaliar jogadas de xadrez e aprender com milhares de jogadas possíveis qual a melhor estratégia que pode ser aplicada.

2.7 Algoritmo KNN(*k-Nearest Neighbors*)

Segundo Fernandes (2005), o algoritmo KNN (k-Nearest Neighbors) utiliza aprendizado supervisionado. Ele trabalha com sistemas de coordenadas e dados já capturados e processados no modelo desejado. A classificação de novos pontos de entrada é feita com base na distância entre eles e os pontos já existentes. Essa classificação resulta em um índice de similaridade, que varia de 0 (nada similar) a 1 (similaridade exata), permitindo a categorização de imagens ou informações.

A distância entre os pontos no algoritmo KNN pode ser calculada usando diferentes modelos matemáticos. Esse algoritmo é capaz de lidar com uma ou mais classes durante o treinamento, classificando novos pontos com base na concentração dos pontos já existentes. Quando KNN é aplicado a um modelo com uma única distribuição, é possível estabelecer um limite de distância máxima para o novo ponto em relação aos pontos antigos. Na ocasião de um novo ponto que se encontra significativamente distante da concentração dos demais, sua distância média em relação aos vizinhos será elevada, indicando uma anomalia. Essa anomalia sugere um comportamento diferente do que foi aprendido pelo modelo, indicando que o novo ponto pode não se integrar à distribuição existente. Assim, o KNN se torna uma ferramenta eficaz para identificar e prever situações indesejadas. A Figura 8 ilustra as concentrações de classes e novos pontos a serem classificados em relação à distância entre eles.

Figura 8 – Representação de pontos e classes do KNN



Fonte: Monard e Baranauskas (2003)

Os modelos matemáticos utilizados no KNN incluem diversas abordagens, sendo a distância euclidiana uma das mais conhecidas. Esse método mede a distância geométrica em um espaço multidimensional, mostrando-se eficaz para treinar modelos de KNN. Contudo, sua aplicação pode enfrentar desafios em cenários com alta dimensionalidade ou com a presença de valores extremos. Por exemplo, quando há uma grande concentração de pontos semelhantes, as distâncias entre eles podem se tornar longas demais, resultando em classificações menos precisas. Apesar dessas limitações, a distância euclidiana continua sendo amplamente utilizada em aprendizado de máquina, reconhecimento de padrões e em diversas aplicações matemáticas (Fernandes, 2005). A Equação 5 demonstra como calcular a menor distância entre dois pontos, fundamentando essa técnica.

$$ditancia(x,y) = \sqrt{\sum_{k=1}^{n} (x_i - y_i)^2} \quad (5)$$

A abordagem de calcular a distância utilizando o método de city-block (ou Manhattan) mede a distância média entre as dimensões, ilustrando a dinâmica de uma cidade organizada em grades. Ao se deslocar de um quarteirão a outro, não é possível atravessar edifícios; é necessário percorrer as ruas. Assim, esse método não determina a menor distância entre os pontos, mas sim a soma das distâncias na vertical e na horizontal. Essa abordagem pode ser eficaz para minimizar o impacto de valores extremos e respeitar as coordenadas dos pontos, o que pode ser

vantajoso em algumas aplicações (Fernandes, 2005). A Equação 6 demonstra como calcular a distância usando esse método.

$$ditancia(x, y) = \sum_{i=1}^{n} |x_i - y_i| \quad (6)$$

O método de Chebyshev calcula a distância entre pontos ao retornar o maior valor absoluto entre as coordenadas. Isso significa que, ao determinar a distância entre dois pontos em um espaço multidimensional, o método considera a maior diferença entre qualquer par de dimensões (Fernandes, 2005). A Figura 9 ilustra os três métodos de cálculo de distância mencionados, evidenciando as diferenças na representação entre os pontos.

Figura 9 – Representação distância dos métodos citados

Fonte: Do Autor(2024)

Segundo Russel e Norvig (2003), a essência do modelo KNN trabalha na suposição de que a propriedade do ponto de entrada seja provável ser classificada com a referência dos pontos vizinhos mais próximos. Por exemplo, existem 4 pontos que representam uma classificação e o novo ponto de entrada ficou próximo e esses pontos, é bem provável que esse ponto assuma a mesma classificação e ao contrário também é recíproco, está muito longe dos pontos é outra classificação.

2.8 Dispositivos eletrônicos

Segundo Kerschbaumer(2018), microcontroladores são circuitos integrados que possuem no seu interior os componentes necessários para seu funcionamento

com dependência apenas da fonte de alimentação. Ele é composto por uma unidade de processamento CPU, conjuntos de periféricos para seu funcionamento, como, memória para dados, memória para processamento e circuito de *clock,* todos já integrados a ele.

O ESP 32 WROOM 32E é um *System-on-Chip*(SoC) focado em alto desempenho e alta conectividade possui vantagens entre seus concorrentes em relação à memória e processamento. Sendo um sistema em *chip,* possui um software chamado de *FreeRTOS, um sistema* operacional em tempo real que permite criar ambientes com alto grau de paralelização com gerenciamento de memória e tarefas simultâneas (MORAIS, 2023).

O *kit* de Desenvolvimento de *software* do SoC ESP 32 possui uma maneira de trabalhar com temporizadores de *hardware. C*om esses temporizadores é possível gerar interrupções que garantem eventos baseados no tempo. As interrupções são capazes de executar tarefas em intervalos especificados independente do que esteja acontecendo na programação. Para se trabalhar com temporizadores é necessário usar a frequência do *clock* externo, que opera em uma velocidade de 80 MHz por segundo, com isso é possível cronometrar o tempo com um registrador. Esse *timer* conta os pulsos gerados e armazena no registrador, assim ao chegar em um valor que foi estabelecido, através de um bloco comparador é executada a interrupção. É comum que a frequência do processador seja dividida por um prescaler para obter uma relação de pulso por microssegundo, no caso de o prescaler ser de 80 obtém-se uma relação que 1000000 pulsos equivale a 1 microssegundo ou 1 Mhz equivale a 1 segundo. A Figura 10 é um esquemático de *timer*. (Joseph, 2022).

Figura 10 – Esquema de um timer

(Joseph, 2022).

Segundo o fabricante do *chip*, Espressif Systems (Xangai) Co. Ltd (2016-2024), o ESP32 possui dois *timer* de hardware para manter o tempo do sistema. O primeiro é o Temporizador RTC, que permite a manutenção do tempo em modos de hibernação e consegue persistir durante reinicializações, exceto em situações de inicialização por falta de energia. Sua precisão depende das fontes de *clock*, com uma resolução de 6,6667 microsegundo. Já o segundo é o *timer* de alta resolução, que, embora não funcione em modos de hibernação e não persista em reinicializações, oferece maior precisão. Ele utiliza a fonte de *clock* APB CLK, normalmente operando a 80 MHz, e mede o tempo com resolução de 1 microssegundo, apresentando um desvio de frequência inferior a ±10 partes por milhão. Essa combinação de timers permite ao ESP 32 atender a diferentes requisitos de precisão e funcionalidade em diversas aplicações.

Orange PI, é um computador de placa única que suporta sistemas operacionais como Linux, Android e diversas distribuições feitas pela comunidade (Shenzhen Xunlong Software CO. Limited).

A placa é um concorrente da famosa Raspberry PI porém com um custo benefício muito maior, utilizam a mesma arquitetura de processador que é o ARM. A Orange PI vem ganhando muita força para servir de servidor em aplicações de casa inteligentes, pois conta com versões de até 4GB de memória RAM e preços bem baixos para o que ele oferece.

2.9 Atuadores

Os atuadores são, de acordo com Hermini (2007, p. 11):

> … elementos que produzem movimentos, atendendo a comandos que podem ser manuais ou automáticos. São usados em automação para entregar à planta a excitação necessária para seu funcionamento, na forma de energia adequada …

Existem diversas tecnologias de atuadores, cada uma com suas aplicações e características. Os atuadores hidráulicos utilizam um fluido que se desloca em um duto sob pressão adequada, o que gera sua fonte de energia. Geralmente, esse fluido é água ou óleo, o que proporciona uma força considerável.

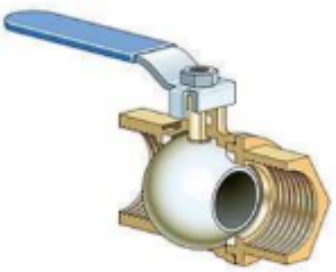
Os atuadores pneumáticos, por sua vez, têm como fonte de energia um gás pressurizado, sendo comum o uso de gás comprimido. Por fim, os atuadores elétricos utilizam a energia elétrica como fonte. Esse tipo de atuador é bastante utilizado devido à presença de motores. (HERMINI, 2007, p. 11)

De acordo com as informações técnicas do catálogo da Parker, o dispositivo eletromecânico válvula solenóide é fundamental em sistemas de controle de fluidos pois é encarregado de regular de forma eficiente e precisa o fluxo de líquidos ou gases. Sua função consiste na interação entre os obturadores, o núcleo móvel(plunger) e a bobina. O princípio do funcionamento é simples, quando a bobina do solenóide é energizada o núcleo móvel é atraído para dentro da bobina, fazendo o obturador se movimentar possibilitando abrir ou fechar, ao cessar a corrente pela bobina, o núcleo volta abrindo a válvula por ação de molas de retornos.

Uma válvula esférica controla o fluxo de fluido por meio de um obturador em forma esférica, que é vazado. Quando a válvula está alinhada à tubulação, o fluido pode passar, e a posição do obturador não apenas regula a quantidade de fluxo, mas também determina se a passagem é permitida; assim, quando a válvula está completamente fechada, não há passagem de fluido (LUCENA, 2024).

O controle do obturador é feito através de uma alavanca que gira para fechar. Nas válvulas esféricas elétricas, um motor acoplado à alavanca permite que ela gire conforme a variação de tensão aplicada. Em alguns modelos, esse processo leva de 10 a 15 segundos para que a válvula feche completamente, a Figura 11 a seguir ilustra um exemplo de válvula com operação manual.

Figura 11 - Válvula esférica manual



Fonte: LUCENA**(2024)**

2.10 MOM (Middleware Orientado a Mensagem) e protocolo MQTT

Middleware Orientado a Mensagem(MOM) é a passagem de dados entre entre um canal de comunicação através de unidades autônomas, chamadas de mensagens, esse canal chamado de, *broker* é responsável pelo roteamento e transferência de dados, além de verificar assinaturas podem arrumar a estrutura da mensagem. As mensagens são enviadas de forma assíncrona e não é essencial que o remetente e o destinatário tenham conhecimento um do outro (ARTEIRO et al.2007).

Bauer e Bolliet (1968) afirmam que middleware é o software de computador que fornece serviços para aplicações além daqueles disponíveis pelo sistema operacional. Um mediador entre sistemas que possibilita a comunicação entre eles.

De acordo com o autor Kale, citado por Faria (2018),

> Os message brokers ou somente brokers, são uma das possíveis implementações de middleware da arquitetura MOM. Eles além de implementar o roteamento das mensagens implementam uma camada que reduz a complexidade de desenvolvimento de alguma funcionalidade, atendendo a requisitos não funcionais específicos e facilitam a reutilização de funções intermediárias. Por exemplo, um broker pode ser usado para gerenciar uma fila de mensagens para vários receptores, fornecendo armazenamento confiável, entrega garantida de mensagens e até mesmo gerenciamento de transações …

Os brokers podem ser utilizados por protocolos MQTT.

O protocolo de comunicação MQTT(*Message Queuing Telemetry Transport*), se encaixa na arquitetura MOM. É um protocolo voltado para troca de mensagens entre dois dispositivos ou duas máquinas que utilizam paradigma de publicar e subscrever em um tópico. É projetado para dispositivos com recurso de baixa largura de banda e alta latência garantindo uma certa confiabilidade entre as mensagens trocadas, realizando o processo com baixo consumo de energia e torna-se um aliado para aplicações IoT. Que contam com recursos escassos de energia e conexão (MQTT.ORG, 2014). Os clientes podem se inscrever em tópicos de publicações como também fazer publicações em tópicos, para enviar as mensagens, possibilitando um desacoplamento entre os dispositivos, ou seja, se caso algum dispositivo estiver com problema os demais continuam funcionando.

2.10.1 REST

A arquitetura REST realiza a troca de informações através de métodos prontos como, GET, PUT, DELETE, POST entre outros através do protocolo *http* que utiliza textos estruturados para enviar mensagens, geralmente com cabeçalhos, tipo de requisição e corpo da mensagem. A comunicação cliente-servidor é importante no REST. O cliente envia uma solicitação ao servidor e o servidor responde fornecendo ao cliente os recursos ou mensagens necessários. REST usa uma abordagem sem estado. Significa que cada solicitação realizada pelo cliente contém todas as informações que o servidor precisa para compreender e processar a solicitação, então ela não armazena o estado da sessão do cliente. Ela é muito utilizada na programação para realização de serviços na *internet* e API REST (FIELDING, 2000). API(Application Programming Interface) Segundo Roy T. Fielding é um conjunto de interface e protocolos que dão a liberdade de comunicação entre diferentes componentes de *software,* essas interfaces e protocolos especificam como os componentes devem interagir entre si possibilitando a não conhecer os detalhes de como foi feita a implementação.

2.11 *Smart Hub*

De acordo com o autor ROUSE 2018 , citado por FARIA (2018),

> Um smart hub é um dispositivo de hardware que conecta objetos inteligentes em uma rede de IoT e controla as comunicações entre eles.

Os *smart hubs* podem ser de empresas grandes como Alexa da Amazon e Google Home da Google. *Home Assistant*, um *smart hub* de código aberto e licença gratuita oferece serviços e integrações feita pela comunidade que atende uma grande parte das demandas de automação residencial.

2.11.1 Home assistant

O *Home assistant* é um *software* de código aberto que transforma o dispositivo onde ele é instalado em um *smart hub*, inclusive em computadores de uma única placa, como, *Raspberry* e *Orange PI*, que torna esse *software* muito

poderoso no meio da automação residencial. Ele possui ferramentas e funções pré -instaladas para o seu funcionamento, algumas delas são as automações e serviços necessitando apenas serem especificadas. No geral, os serviços configuram todas as funções que o usuário pretende realizar na construção da sua aplicação com os diversos tipos oferecidos pelo *Home Assistant*. Os serviços, de maneira geral não funcionam de forma automática salvo os do tipo configuração e do tipo *sensors*, os demais interagem com as automações, que são ações que executam de forma automática os serviços configurados anteriormente, através de gatilhos(*triggers*), que por ventura podem ser disparados por publicações no *broker* MQTT. Todas essas soluções ficam de livre acesso sem a necessidade de configurar portas para o acesso. Porém exige conhecimentos técnicos para instalação e configuração, que por sua vez podem chegar a executar códigos mais complexos escritos em Python. Dentro do *Home Assistant* é possível gerenciar *brokers* MQTT, banco de dados, acesso remoto, consumir serviços de API REST entre muitas extensões feitas pela comunidade, além de todas essas ferramentas é possível montar uma interface gráfica e acompanhar dados em tempo real possibilitando enviar alertas sobre informações relevantes.

2.12 Trabalhos relacionados

A tese orientada pelo professor Dr. Martim (2022) da UNICAMP dos autores Galhardo, Rocha e Sodek (2022), intitulada “Aplicações de Aprendizado de Máquina para Pesquisa de Vazamentos em Tubulações”, utiliza conceitos e soluções relevantes apresentados no levantamento bibliográfico.

O trabalho propõe um protótipo que executa um algoritmo em uma placa *Raspberry* Pi. Esse protótipo avalia imagens processadas pela biblioteca *OpenCV*, capturadas de ruídos provenientes de um sensor de vibração do fluxo de água nas tubulações até o reservatório. As imagens, que representam gráficos de vibração, são salvas e utilizadas para treinar um modelo de aprendizado de máquina com o algoritmo KNN (*K-Nearest Neighbors*). Com isso, é possível avaliar a distância desses gráficos em relação aos vizinhos próximos e classificar se o ruído detectado indica um vazamento.

Na conclusão, o protótipo foi testado em pontos específicos e apresentou certa imprecisão, especialmente na ausência de uma rede de pontos confiáveis. No entanto, a predição está funcionando, e o estudo aponta para a necessidade de otimização do algoritmo. A Figura 12 ilustra o protótipo instalado em um reservatório.

Figura 12 - Projeto aplicado junto ao reservatório

Fonte : Galhardo, Rocha e Sodek (2022)

A tese de Gamboa-Medina (2017), intitulada “Detecção de vazamentos e alterações em redes de distribuição de água para abastecimento, durante a operação, usando sinais de pressão”, investiga a detecção de vazamentos por meio de séries temporais e aprendizado de máquina, aplicando modelos complexos em redes comerciais. O mecanismo utilizado para obter a vazão foi baseado na pressão da água em diferentes pontos da canalização.

Os métodos empregados incluem cadeias de Markov, controle estatístico de processos e, por fim, a comparação de padrões, que apresentou dados mais conclusivos. Os dados sobre o consumo de água foram fornecidos pela companhia, abrangendo um ano de observações. Na conclusão, os resultados mostraram-se promissores; no entanto, a base de treinamento continha muitos vazamentos, o que

dificultou o aprendizado. Um caso específico destacado na pesquisa foi o de um vazamento que permaneceu aberto por 56 dias.

Para obter resultados melhores, foi necessário realizar estudos em intervalos menores de fornecimento, o que pode servir como ponto de partida para desenvolver uma solução baseada em intervalos que pareçam semelhantes utilizando métodos de aprendizado de máquina em padrões específicos.

3 PROCEDIMENTOS METODOLÓGICOS

Para atender aos objetivos geral e específicos e oferecer uma resposta à problematização da tese, o andamento das ações propostas foi organizado em etapas que refletem os conjuntos de atividades realizadas ao longo da pesquisa, iniciando com escolhas dos componentes e soluções encontradas no referencial teórico; testes, aferimentos dos ruídos e repetitividade; preparação do conjunto do protótipo, coleta de dados; treinamento do modelo; predição e coleta dos resultados; e, por fim, análise dos resultados e conclusão.

3.1 Apresentação das soluções encontradas

No sistema, será adotado um padrão de comunicação entre cliente e servidor. Tanto o cliente quanto o servidor devem aceitar o protocolo de comunicação definido, que é o MQTT, com o broker sendo o Mosquitto MQTT, instalado no *Smart Hub*. Esse protocolo é amplamente reconhecido e já bem conhecido pelo autor desta pesquisa.

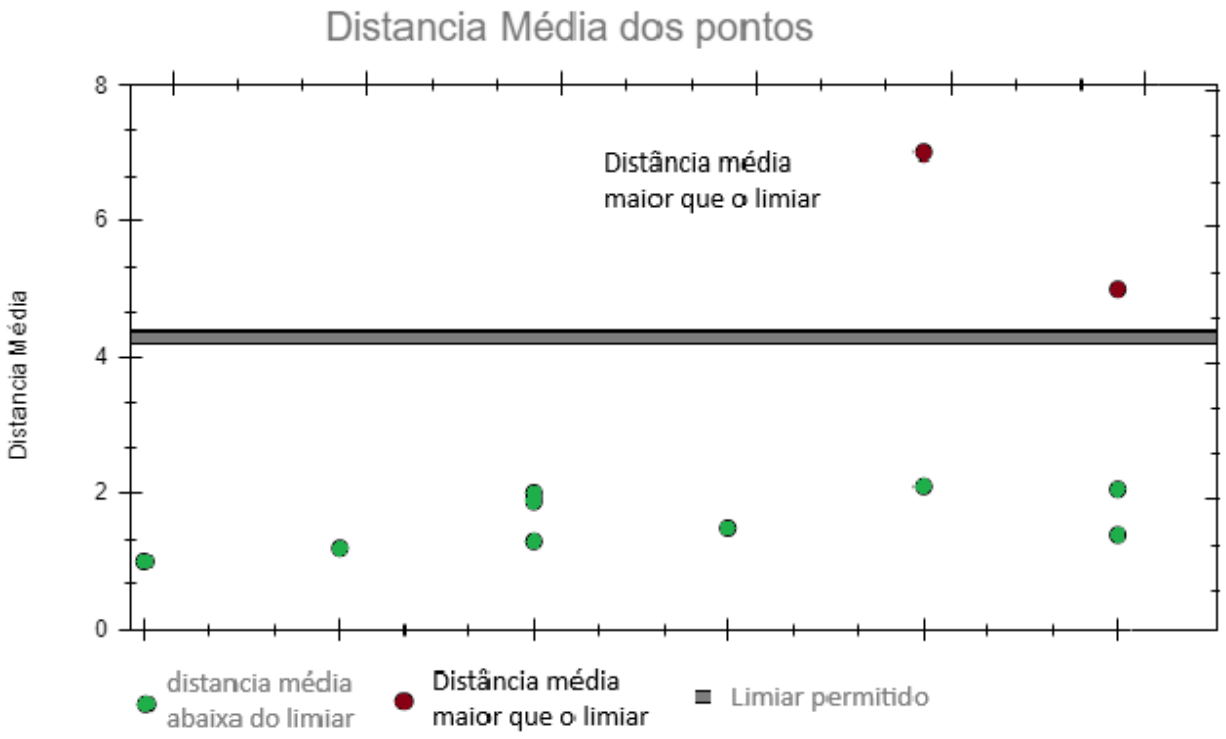
A próxima solução para o armazenamento dos valores junto às datas e horários de aquisição, foi o *InfluxDB*, um banco de dados temporal que pode ser configurado junto aos *smart hubs*, e se torna o candidato ideal por essa característica.

Para o servidor, será utilizada uma aplicação que roda localmente em uma distribuição Linux na placa única *Orange* Pi Zero 3 pela sua disponibilidade. Nesse servidor, será instalado o *Home Assistant Supervised* (*Smart Hub*) e uma API programada em Python e Flask, visando atender ao requisito de avaliar os dados com o modelo treinado. O *Home Assistant* é um *software* gratuito e poderoso para casas inteligentes, que se adapta aos desafios da pesquisa.

Optou-se por uma distribuição *Linux* para o servidor, pois é amplamente utilizada para essa função em diversas aplicações, tornando a implementação mais prática. A escolha da linguagem Python e da API em *Flask* se deve à facilidade de trabalhar com algoritmos de aprendizado de máquina e a capacidade de resolver problemas localizados, sem a necessidade de uma arquitetura muito complexa para criar um padrão REST.

Dentro dos estudos feitos na fundamentação teórica, o algoritmo para treinar o modelo de aprendizado de máquinas é o KNN(k-vizinhos mais próximos). É utilizado apenas uma classe e o método de classificar fica a critério de avaliar a distância média dos novos pontos em relação aos vizinhos mais próximos, utilizados no modelo de treinamento, assim é possível avaliar se os novos pontos pertencem ao modelo ou é uma anomalia. O método de calcular a distância utiliza um sistema de coordenadas, sendo o horário da aquisição e o consumo de água. A equação para o cálculo é a *City-Block* (*Manhattan*), pois ela considera os dois eixos do plano para calcular a distância, permitindo o modelo a levar em consideração o horário e seu consumo específico. A Figura 13 ilustra como o modelo avalia utilizando apenas uma classe de distribuição por meio de um limiar definido com o algoritmo KNN. Quando a distância média em relação aos vizinhos ultrapassa esse limite, indica que é possível que aquele ponto seja um vazamento. Posteriormente, será discutido como foi obtido o limiar de referência. A escolha deste algoritmo é para avaliar se ele consegue identificar vazamentos no contexto da pesquisa.

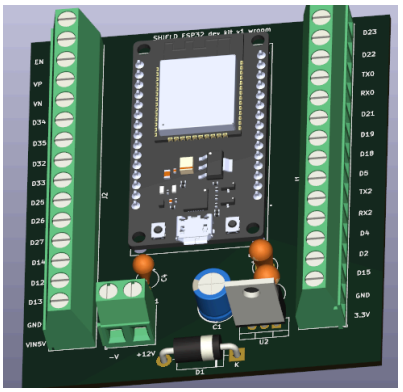
Figura 13 - Como avaliar em relação a distância média



Fonte : Autor(2024)

O cliente da aplicação precisa possuir ferramentas que possibilitam ele a comunicar via MQTT, então é buscado uma plataforma que consiga ler dados de um sensor em relação ao tempo e a possibilidade de acionar uma válvula de retenção. A escolha é a ESP 32, pois no levantando da fundamentação teórica o SoC oferece múltiplas formas para satisfazer esses requisitos, uma delas é o próprio *FreeRTOS* que possibilita executar funções em paralelo de forma contínua, não foi necessariamente um dos requisitos funcionais, porém fica mais interessante utilizar de funções do sistema operacional, para tratar interrupções, conexões contínuas com o *Wi-Fi* e o *broker* MQTT de forma contínua em funções *multitask*. Um problema que surge são os ruídos, conexões limitadas e dificuldade de alimentação do dispositivo em ambientes fora da bancada de teste, por isso surgiu a necessidade de construir uma placa de circuito impresso para lidar com esses empecilhos. Figura 14, é um shield prototipado para a placa ESP 32 no *software Kicad*, onde pode se observar um sistema de alimentação com regulador de tensão e filtros para que ele não sofra interferência da rede externa. Todas as saídas de placa estão recebendo terminais de conexão tipo parafuso possibilitando uma maior segurança entre os chicotes de sensores e a ESP 32, além de facilitar a montagem futura de todos os componentes.

Figura 14 - Shield ESP 32



Fonte : Autor(2024)

Para medir o fluxo da água é utilizado o YF-S201, que é um sensor que mede o fluxo de água através do método de turbina com terminais do tipo efeito *hall*. O fluido percorre e gira a turbina com um imã que induz tensão em uma placa sensível a campos magnéticos, que gera uma diferença de potencial. Essa diferença é medida continuamente obtendo uma frequência, ou pulsos, que representa a

quantidade de fluido que passou por ele. De acordo com o fabricante, conforme o datasheet no anexo (A), é possível obter as informações necessárias para montar a Equação 7, que representa a forma de aplicar os dados capturados para calcular a vazão. Ao dividir o valor do resultado da equação por 60, que corresponde ao número de segundos, obtém-se o volume total em litros.

$$V = \frac{F}{7.5} \tag{7}$$

Sendo:

$V \ - \ Vazão[L/min]$
$F \ - \ frequência\ em\ [Hz]$

O sensor da Figura 15 foi escolhido por atender a especificações de leitura de até 30 litros por minuto, e suportar uma pressão de até 1.75 Mega Pascal que de acordo com art. 133 da resolução nº 19, de 27/03/2019, o fornecimento de água possui uma faixa de 0,1 - 0,50 Mega Pascal, ou 10 - 50 metros de coluna de água e o sensor opera na faixa de 3.3-24V. O sensor utiliza de um método tipo turbina encontrado na pesquisa realizada.

Figura 15 - Sensor vazão YF-S201

Fonte : Autor(2024)

Já citado anteriormente para analisar os dados e avaliá-los, foi confeccionado uma API em Python do tipo *rest*. A API é construída com a biblioteca Flask. No aprendizado de máquina foi utilizado o *sklearn*, uma biblioteca específica para lidar com diferentes tipos de modelos de aprendizado de máquinas, sendo ambos de código aberto. No treinamento é utilizado o *Google collab* devido ao seu alto poder

de processamento e o resultado final é uma API, com seu respectivo endereço executando dentro do servidor *Linux* na *Orange* PI, atendendo o requisito de avaliar novos dados em relação ao modelo.

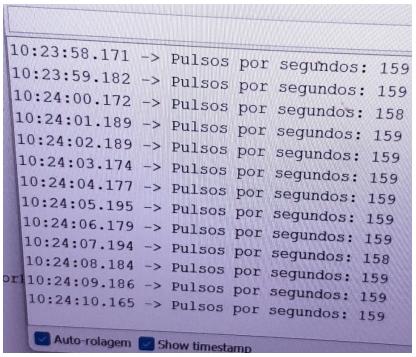
3.2 Testes, aferimentos dos ruídos e repetitividade

Os testes para avaliar os dispositivos são de grande importância para garantir a qualidade e evitar falhas na execução do projeto. Para isso, foi aplicada uma bateria de testes.

A aplicação depende bastante do tempo e, conforme os estudos na fundamentação teórica, é possível utilizar o timer interno para executar uma interrupção do ESP 32 no intervalo de 1 segundo, obtendo o valor da vazão a cada ciclo. Sendo assim, foi realizado um teste de validação do SoC, onde foram gerados pulsos a cada 1 segundo, utilizando um osciloscópio para visualizar a frequência enviada ao longo do tempo por um gerador de onda quadrada, que no caso foi de 158,7 Hertz, para fins de comparações.

Na Figura 16, são apresentados os resultados enviados pelo gerador. Neste caso, o ESP 32 está captando os dados por meio de um IRQ externo, permitindo que o SoC reaja rapidamente a eventos externos através de uma porta digital específica, sem a necessidade de realizar *polling* contínuo, os valores são bem parecidos com os obtidos pelo osciloscópio.

Figura 16 - Leitura da ESP 32



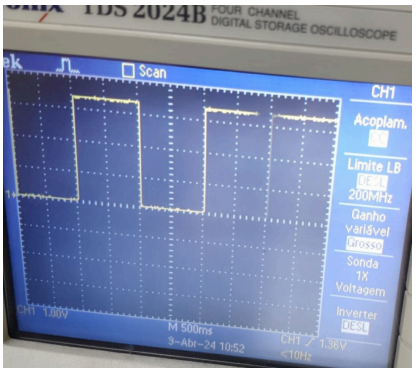
Fonte : Autor(2024)

O segundo teste é necessário para confirmar se o *timer* realmente marca o valor esperado de 1 segundo para disparar uma interrupção. Para isso, foi realizada

a medição de um pino que se ativa ao atingir 1 segundo no *timer* interno, enviando um sinal de 3,3V. No ciclo seguinte do temporizador interno, verifica-se se o pino ainda está ativo. Se estiver, ele retorna ao estado inativo, gerando um sinal de 0V. Dessa forma, é criado um gerador de onda quadrada a cada interrupção do timer interno.

Na Figura 17, o osciloscópio mede essa variação. Cada medida no eixo horizontal corresponde a 500 ms, enquanto cada bloco no eixo vertical representa 1V. O resultado obtido é o esperado, variando entre 3,3V e 0V em um ciclo de 1 segundo. Com esses resultados favoráveis na avaliação dos métodos de aquisição de pulsos e no gerenciamento do tempo, é possível avançar para os testes de repetitividade do sensor.

Figura 17 - Resultado a cada 1 segundo do timer interno

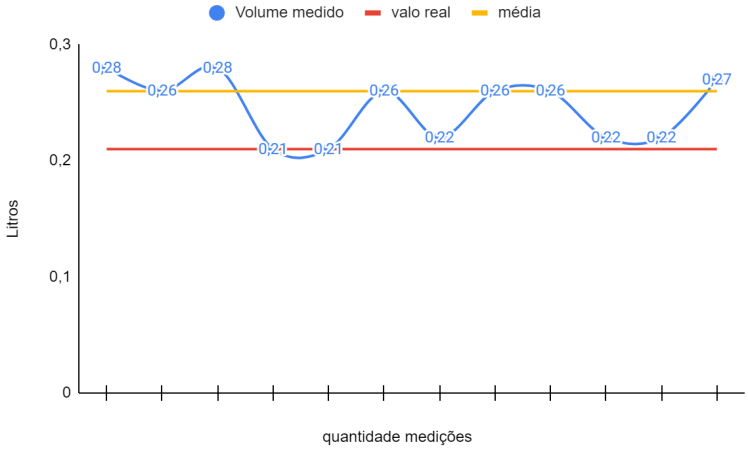


Fonte : Autor(2024)

Uma parte importante do projeto é verificar a repetitividade do sensor. Para isso, foi realizado um teste com 201 mililitros, esse valor teve sua obtenção através de uma balança. Foi pesado 201 gramas que para água, de acordo com Chang e Goldsby (2015) a densidade da água se aproxima de 1 grama por mililitro, ou seja 1 grama representa 1 mililitro.

O teste baseia-se em obter o volume da água que o sensor mede no decorrer do tempo, a programação soma os pulsos no intervalo de 1 segundo aplicando a Equação 7, dividindo por 60 e somando os valores, o resultado é o volume total que foi medido. O gráfico da Figura 18 ilustra os resultados de 12 medidas e a média dos valores.

Figura 18 - Teste dos valores de repetitividade



Fonte : Autor(2024)

No resultado é possível perceber que existe um leve desvio onde o sensor mede com um erro para mais do que o valor real, porém a repetibilidade surpreende pois, entre a média e os valores medidos a distância entre eles é baixa, a exatidão não é algo preocupante pois os dados que serão usados no treinamento serão captados pelo mesmo sistema.

3.3 Preparação do conjunto do protótipo

A construção da aplicação do *smart Hub* vai da necessidade de cada projeto e exige processos de configurações e instalações de *add-ons* específicos. Após a instalação do *Home assistant supervised* no servidor, é preciso descobrir se a aplicação assumiu um endereço de *ip local* e acessar juntamente com prefixo da porta *ip*:8123, conforme a Figura 19 é possível ver o comando utilizado para verificar se o endereço está recebendo pacotes, sendo assim testando a conectividade.

.

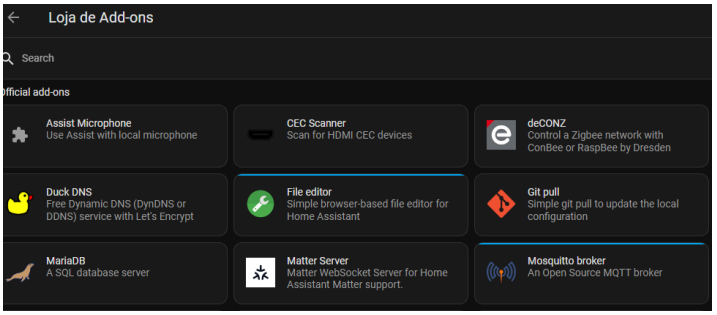
Figura 19 - Comando para teste de conectividade

```
PS C:\Users\felip> ping orangepizero3

Disparando orangepizero3 [192.168.1.12] com 32 bytes de dados:
Resposta de 192.168.1.12: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.1.12: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.1.12: bytes=32 tempo=3ms TTL=64
Resposta de 192.168.1.12: bytes=32 tempo=3ms TTL=64

Estatísticas do Ping para 192.168.1.12:
    Pacotes: Enviados = 4, Recebidos = 4, Perdidos = 0 (0% de
             perda),
```

Fonte : Autor(2024)

Acessando o Home Assistant na porta 8122 pela primeira vez é preciso criar um usuário de acesso e começar a utilizar. Primeiramente para essa aplicação foi a instalação mosquito *broker* MQTT e *InfluxDB* da loja da add-ons do *Home Assistant*, o acesso até a loja fica na aba configurações e posteriormente em *add-ons*, conforme a Figura 20, existem muitas possibilidades dentro da loja.

Figura 20 - Loja de add-ons



Fonte : Autor(2024)

Na configuração do *broker* é necessário um nome, porta de execução e sua palavra passe para permitir publicações, conforme a Figura 21.

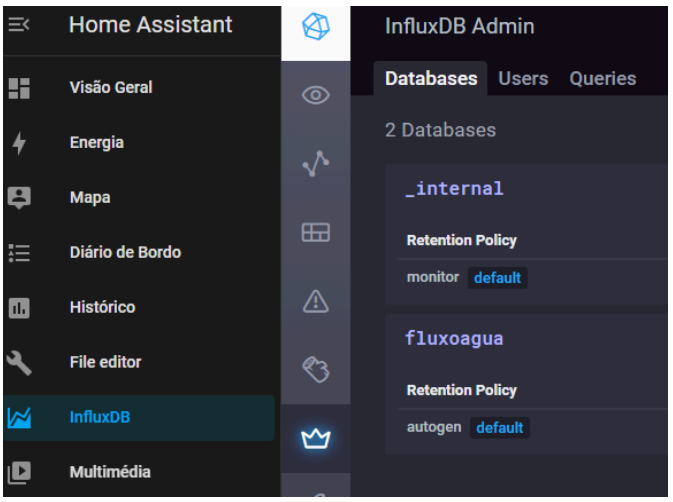
.

Figura 21 - Configuração do broker

Fonte : Autor(2024)

Na configuração do banco de dados, é recomendável criar uma *database* independente, nesse caso conforme a figura 22 se chama fluxoagua e a porta já vem configurada, sendo ela 8086. O próprio *Home assistant* cria um endereço no menu flutuante para acessar a extensão.

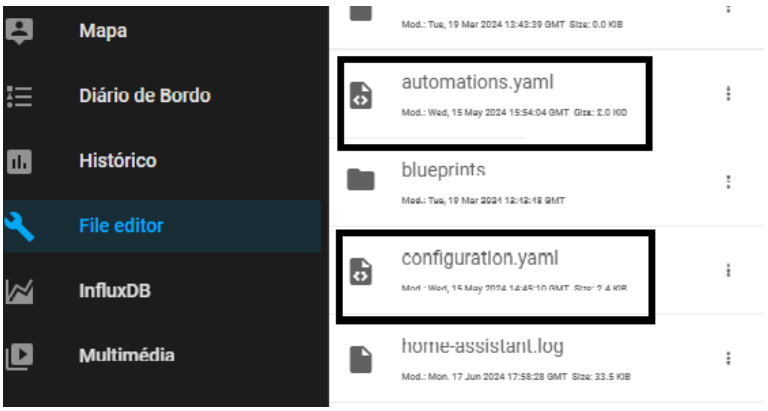
Figura 22 - Banco de dados temporal



Fonte : Autor(2024)

No *Home Assistant,* os serviços, configurações e automações precisam ser configurados dentro dos arquivos chamado de "*configuration.yaml*" e "automations.yaml", respectivamente, que usam uma linguagem de serialização de dados legível para humanos e de fácil interpretação para máquinas, conforme a Figura 23 é possível acessar elas pela aba *file editor*.

Figura 23 - Acesso as configurações e automações



Fonte : Autor(2024)

No arquivo “configuration.yaml” está a configuração de acesso ao banco de dados temporal e os serviços que vão ser utilizados. Na Figura 24 é possível visualizar a configuração do *influxdb* e os serviços de notificação que executam uma função do tipo *rest*, do *callmebot* e um serviço do tipo *rest command*, que executa a API de analisar os dados com o KNN. Os serviços não funcionam de maneira automática por isso existe a necessidade do arquivo “automations.yaml” para executar os serviços, salvo os que são apenas configurações e sensores. O *callmebot* é uma API externa que possui integração com o *Home Assistant* possibilitando enviar mensagens para o *Whatsapp*.

Figura 24 - Serviços da Aplicação

/homeassistant/configuration.yaml

```
  default: info

# serviço de whatsapp
notify:
  - name: WhatsApp
    platform: rest
    resource: https://api.callmebot.com/whatsapp.php
    data:
      source: HA
      phone: +554898268499   #numero de culular
      apikey: 6364463        #apikey do bot

rest_command:
  chamar_api:
    url: 'http://10.0.0.6:5000'  # Coloque a URL completa do
    method: GET  # Ou POST, dependendo do método HTTP que vo
    content_type: application/json  # tipo JSON
    timeout: 10

influxdb:
  host: 10.0.0.6
  port: 8086
  username: felipe
  password: felipe
  database: fluxoagua
  max_retries: 3
  tags:
    source: HA
  tags_attributes:
    - friendly_name
  default_measurement: "water_volume_L"
  include:
    entities:
      - sensor.water
```

Fonte : Autor(2024)

Ainda na parte de configurações dos serviços, estão sendo utilizados ainda os “*sensors*” e “*inputs*”, o primeiro é para receber dados do sensor de fluxo publicados no broker MQTT, e o segundo é uma chave interruptora de dois estados, ligado e desligado, para representar a válvula e o aviso de vazamento respectivamente. A Figura 25 ilustra as configurações do tipo sensores e *inputs,* tópicos de escrita, seus nomes, unidades de medição e sua classe.

Figura 25 - Serviços de sensores e inputs

```
mqtt:

  sensor:
    - name: "vazao"
      unique_id: vazao123
      state_topic: "caminho/teste"
      unit_of_measurement: "L/min"
      value_template: "{{ value_json.valor_vazao }}"
      json_attributes_topic: "caminho/teste"
      state_class: measurement
      device_class: volume_flow_rate

    - name : "litrosmedidosemseg"
      unique_id: vazao
      state_topic: "caminho/teste"
      unit_of_measurement: "L"
      value_template: "{{ value_json.litrosmedidosemseg }}"
      json_attributes_topic: "caminho/teste"
      state_class: measurement

    - name : "avisosvazamentos"
      unique_id: avisovazamento
      state_topic: "caminho/teste"
      unit_of_measurement: "ocorrências"
      value_template: "{{ value_json.avisovazamento }}"
      json_attributes_topic: "caminho/teste"
      state_class: measurement

input_boolean:
  notify_val:
    name: "Valvula retenção"
    icon: mdi:pipe-valve

  notify_vazamento:
    name: "estadovazamento"
    icon: mdi:home-alert
```

Fonte : Autor(2024)

Na figura 25 a palavra (“mqtt:”) significa que os serviços utilizam o MQTT. Percebe-se também que o tópico assinado para obtenção de dados é caminho/teste e os mesmos estão sendo publicados no formato *JSON* com os seus respectivos nomes no “value_template”. Uma informação importante é que os próprios *sensors* salvam os dados no banco temporal captados no broker. Já os inputs, a configuração deles, basta dizer que são do tipo booleano que o próprio *Home Assistant* consegue tratar os estados.

As automações citadas anteriormente funcionam da mesma maneira das configurações, com o mesmo tipo de serialização, porém diferentes arquivos. Como o próprio nome sugere, são ações que executam de forma automática os serviços configurados anteriormente, através de gatilhos(*triggers)*. Na aplicação todos os gatilhos são manipulados pelas publicações nos tópicos *broker* MQTT, então tanto o ESP 32 quanto a API para avaliar novos dados no modelo conseguem acioná-las. A Figura 26 mostra um exemplo de configuração de uma automação, e as demais automações que o projeto utiliza estarão no apêndice (B). Na automação da Figura 26, ela chama o serviço de “*notify.whatsapp”* configurado anteriormente com a condição do gatilho sendo o tópico caminho/aviso no *broker* com valor “ligar”, ou

seja, quando uma mensagem for publicada neste tópico a automação será executada, que chama o serviço e envia uma mensagem no *whatsapp* com a respectiva mensagem no atributo *“message”*

Figura 26- Exemplo automação

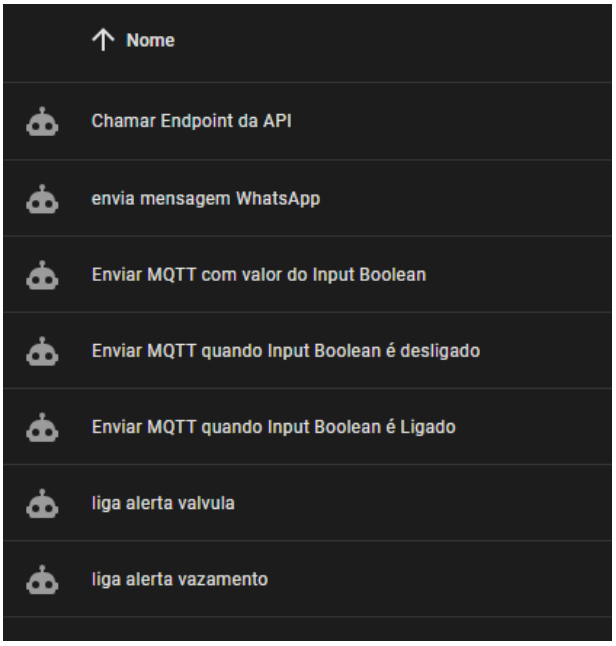
/homeassistant/automations.yaml

```
20
21 - id: Aviso whatsapp via mqtt
22   alias: envia mensagem WhatsApp
23   trigger:
24   - platform: mqtt
25     topic: caminho/aviso
26   condition:
27   - condition: template
28     value_template: '{{ trigger.payload_json.aviso == ''ligar'' }}'
29   action:
30   - service: notify.whatsapp
31     data:
32       message: Possivel vazamento detectado
```

Fonte : Autor(2024)

A Figura 27 é todas as automações que o projeto utiliza e seus nomes.

Figura 27- Automações do projeto



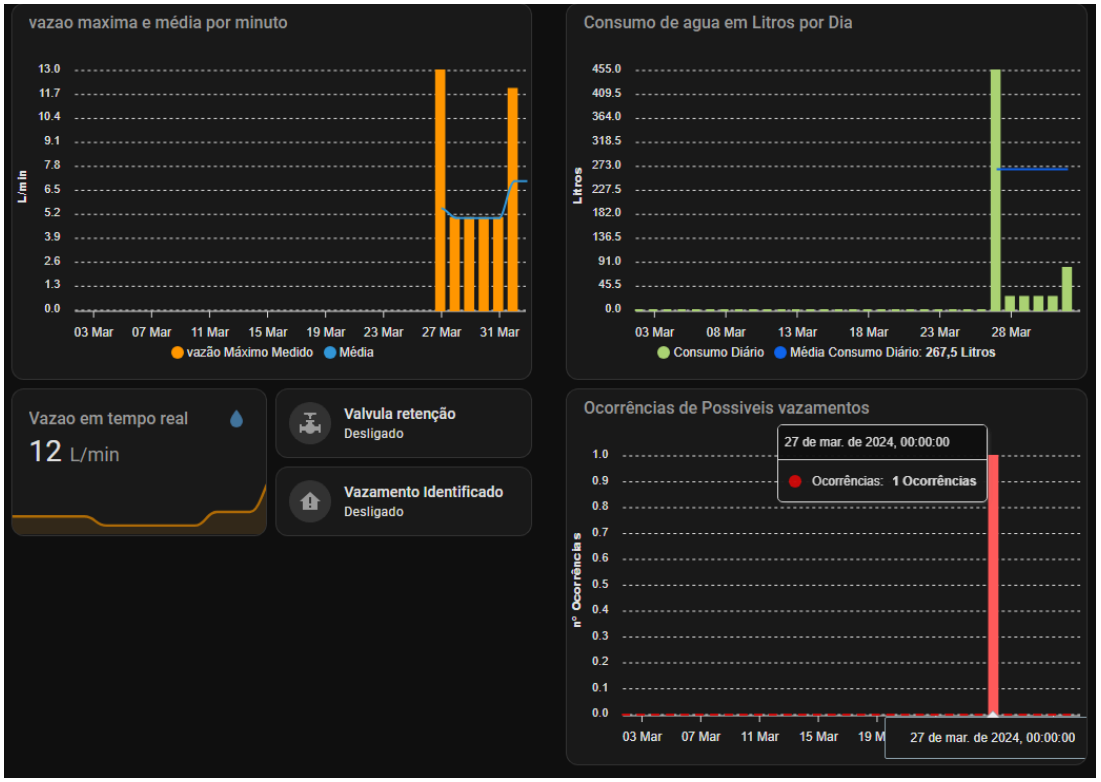
Fonte : Autor(2024)

No total estão sendo utilizadas 7 automações, sendo uma para chamar a API que avalia os dados conforme o modelo, a que envia mensagem para o celular, três delas é para o estados dos *inputs* que sintonizam a ESP 32 com a aplicação visual,

assim como uma para a inicialização do sistema, caso falte energia o próprio sistema consegue se auto sincronizar. As duas remanescentes são para ligar a válvula e o alerta do vazamento, que recebem informações de tópicos diferentes, por isso a necessidade de 2 novas automações. Todas as automações funcionam de maneira bem parecida com a citada anteriormente, através de publicação em seu respectivo tópico e seus gatilhos para executar os serviços.

Na parte de interface visual é possível usar gráficos de bibliotecas de terceiros como *Apex Charts*, ilustrado pela Figura 28.

Figura 28- *Dashboard* concepção com valores fictícios

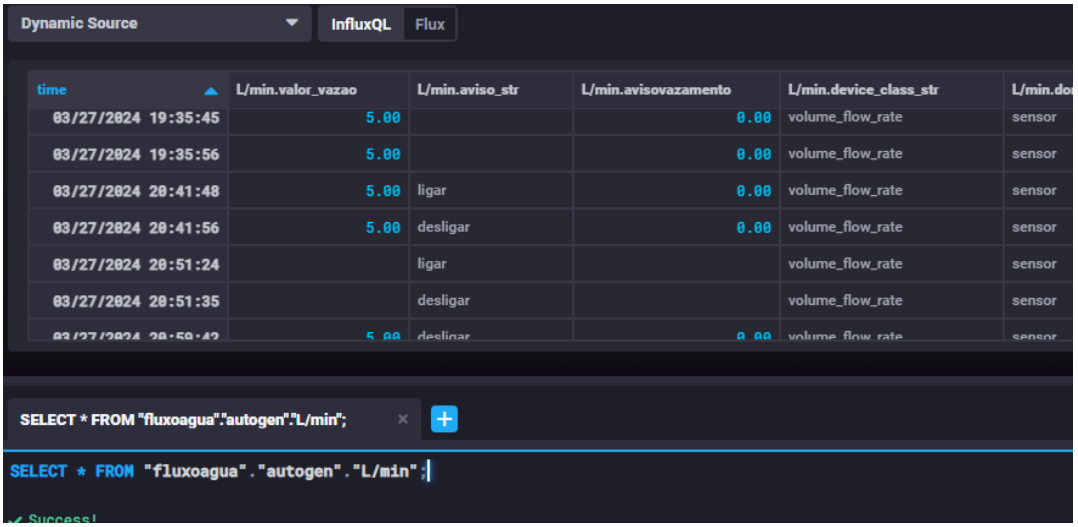


Fonte : Autor(2024)

Os dados dos gráficos são alimentados através de consultas no banco de dados do *Influx* DB. A Figura 28 ilustra a concepção visual do projeto, onde os gráficos são os valores dos sensores que publicaram no banco temporal. Os ícones de válvula e a casa com exclamação são interruptores, um ilustra se existe vazamento na rede e outro fecha e abre a válvula. Existe um cartão que é a vazão em tempo real medida direto do sensor.

As informações capturados pelos sensores publicados no tópico MQTT são salvos no influxDB, com seus respectivos carimbos de data e hora, conforme a Figura 29, que é um dos pontos chaves para o funcionamento da aplicação e um requisito funcional.

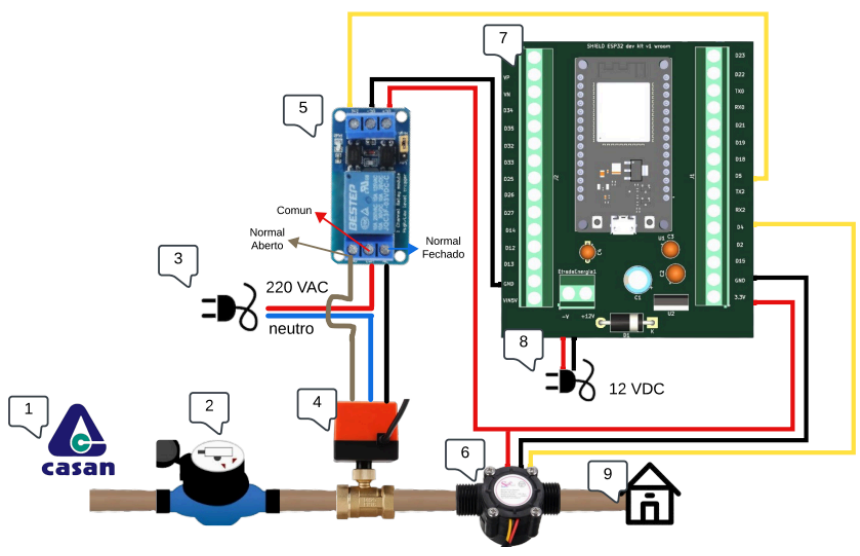
Figura 29 - Banco temporal Influx DB



Fonte : Autor(2024)

Na confecção do sistema físico foi necessário criar um diagrama com os componentes que vão compor o protótipo, como sequência, os componentes e quais que precisam de conexões entre eles, Figura 30.

Figura 30 - Diagrama dos componentes

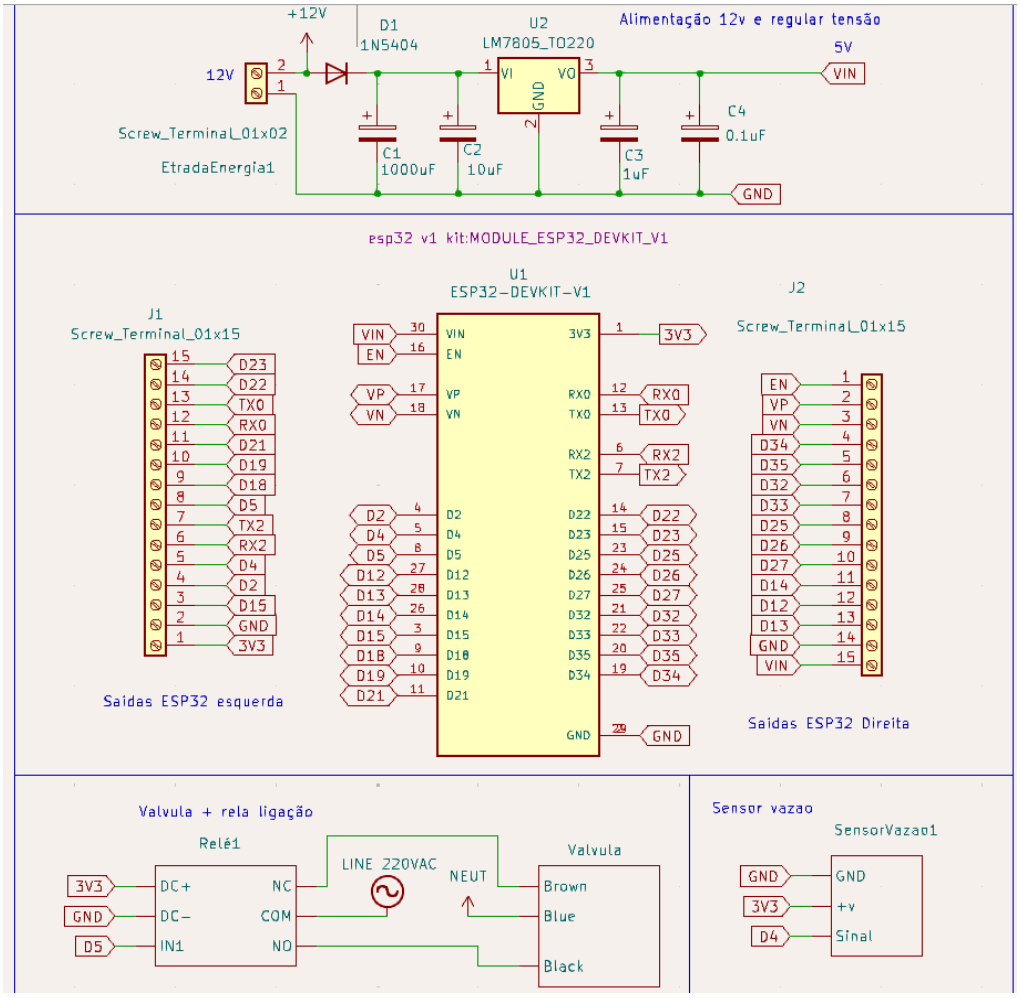


Fonte : Autor(2024)

1. Fornecimento de água CASAN
2. Relógio higrômetro CASAN
3. Entrada energia 220V
4. Válvula esférica elétrica
5. Relé optoacoplador
6. Sensor vazão
7. Shield ESP 32, controlador
8. Fonte linear 12V
9. Fornecimento de água para casa

Partindo do diagrama visual Figura 30, foi possível desenvolver os diagramas elétricos para conexão dos componentes ao *shield* da ESP 32. Nota-se que são necessárias duas fontes de alimentação e um relé isolando as fontes de tensão, ainda percebe-se que foi preciso confeccionar chicotes elétricos para ligação dos, Figura 31.

Figura 31 - Diagrama Elétrico

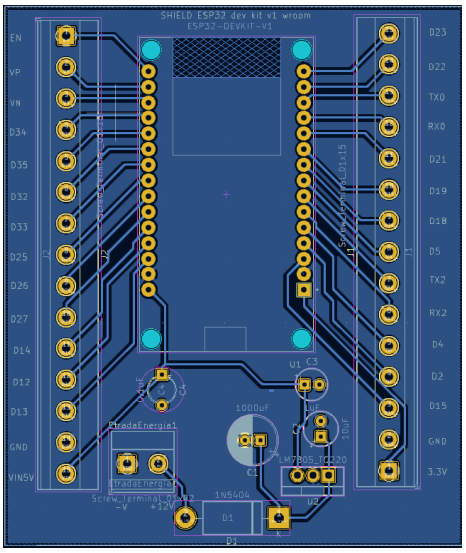


Fonte : Autor(2024)

É possível visualizar que o circuito tem um regulador de tensão com entrada de 12VDC para 5V além de servir como filtro contra variação da rede, todas as saídas do SoC estão sendo expandidas para uma conexão tipo parafuso onde é feita a ligação do relé e do sensor de vazão. No sistema da válvula, o terminal comum (COM) do relé está conectado diretamente à rede de 220VAC da tomada, enquanto os terminais normalmente fechados (NC) e normalmente aberto (NO) recebem o neutro da rede. Dessa forma, o sistema opera em duas vias: quando a ESP 32 envia 3,3V para o IN1 do relé pela porta digital 5, a válvula é acionada; caso o sinal esteja em 0V, o fluxo de água é permitido.

No sistema de captação, a porta digital 4 captura os pulsos enviados pelo sinal do sensor. As demais conexões incluem alimentação de 3,3V e terra. A válvula, o relé e o sensor estão representados por uma caixa para simplificar o diagrama, mas o circuito elétrico completo do módulo relé pode ser consultado no anexo (C). A Figura 32 apresenta o circuito impresso para a manufatura do shield.

Figura 32 - Prototipagem do circuito

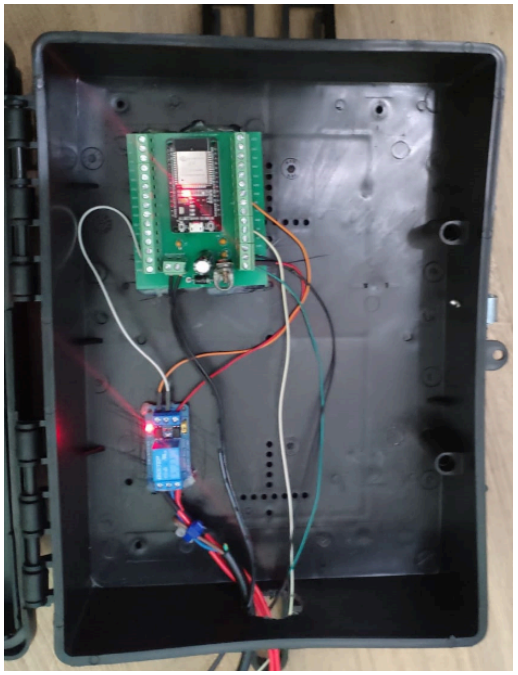


Fonte : Autor(2024)

Com o esquema elétrico e a prototipagem da placa finalizados, foi realizada a soldagem dos componentes eletrônicos. Além disso, foi necessário acomodar tudo dentro de um quadro elétrico, fixando o módulo relé e o circuito impresso com cola quente. Na figura 33 é possível visualizar os componentes ligados uns nos outros e os chicotes elétricos com uma distância de 400 mm, possibilitando uma maior

mobilidade entre o quadro elétrico, o sensor e a válvula. Percebe-se também que as conexões entre os componentes e fontes de tensão estão energizadas e nota-se a quantidade de fios e chicotes que foi preciso para fazer a ligação elétrica.

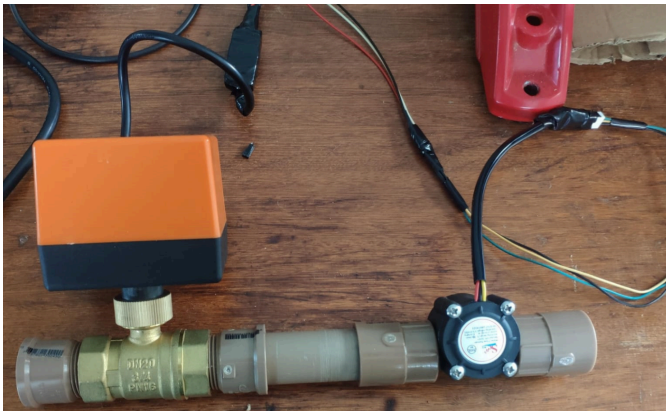
Figura 33 - Sistema eletrônico confeccionado



Fonte : Autor(2024)

Na Figura 34 está a válvula junto com o sensor de vazão com a encenação com as luvas de ¾ de polegada.

Figura 34 - Válvula e sensor



Fonte : Autor(2024)

Na Figura 35 ilustra-se a necessidade de perfurar o solo até achar a encanação da casa perto do relógio, esse buraco possui uma profundidade significante porém o projeto tinha previsto isso e os chicotes elétricos conseguiram satisfazer a distância entre o quadro elétrico e a encanação. Nota-se também que é necessário cortar a encanação e fazer uma conexão com o dispositivo.

Figura 35 - Hidrômetro da casa

Fonte : Autor(2024)

Encenação localizada foi instalado o sistema e a Figura 36 ilustra a implementação do protótipo, já com o corte e a emenda do encanamento.

Figura 36 - Sistema instalado

Fonte : Autor(2024)

A Figura 37 é a placa *Orange* PI conectada na energia elétrica e no modem de *Internet* para prover o servidor da aplicação.

Figura 37 - Orange Pi com servidor

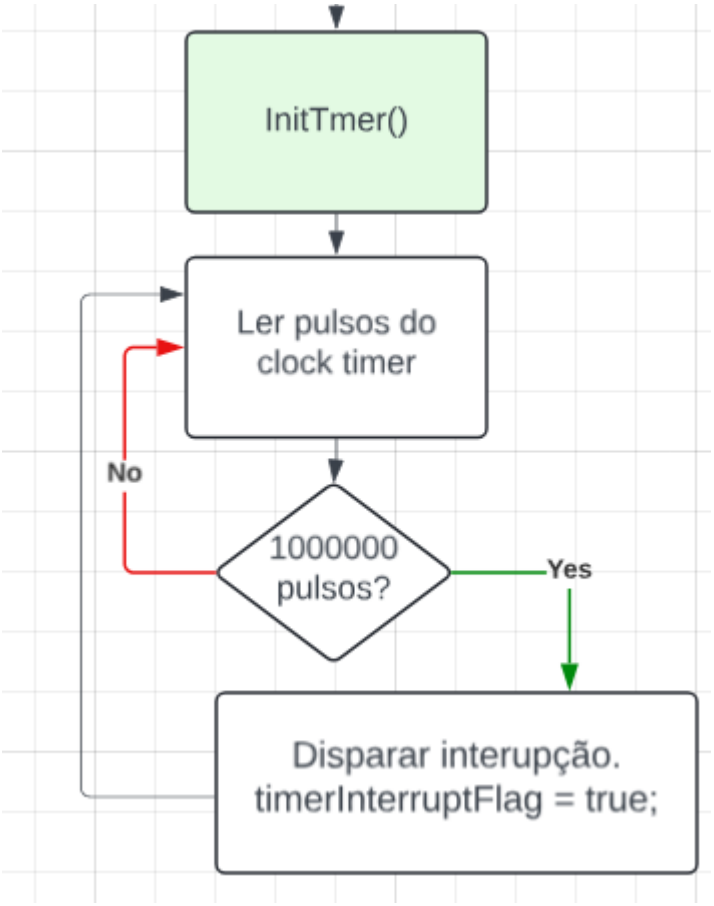
Fonte : Autor(2024)

Na realização da parte lógica do protótipo o sistema está dividido em dois módulos, sendo a ESP 32 que conta com uma programação feita com o Arduino com algumas funcionalidades do *FreeRTOS*. A programação do SoC é dividida em seis funções principais que são executadas concorrentemente, aproveitando a capacidade de multitarefa do *FreeRTOS*. Dessas funções, três são do tipo *task* e três são funções contínuas. É importante ressaltar que existem variáveis globais que manipulam os estados das funções, permitindo que, mesmo quando executadas simultaneamente, algumas delas aguardem um gatilho para cumprir suas respectivas tarefas. As variáveis globais estão descritas abaixo.

```
1. // VAR ---------------
2. const int sensorPin = 4;
3. const int valvulaPin = 5;
4. volatile long pulse;
5. int flag1segundo = 0; // Sinalizador para indicar 1 segundo
6. int flagIniciandoProg = 0;
7. volatile bool timerInterruptFlag = false; // Sinalizador para
   indicar interrupção do timer
8. float totalLiters = 0;
9. float volumeTotal = 0;
10.   float mediaflowrate = 0;
```

Para o entendimento de como o sistema funciona em conjunto, é preciso entender o que cada função cumpre individualmente e as variáveis globais da Figura 38. A primeira função contínua é a função (“initTimer”), essa é a função que gerencia as interrupções através do timer interno, da mesma maneira da fundamentação teórica, ou seja, é definido um prescaler de 80 para transformar a velocidade do clock de 80 Mhz para 1, e cada nova vibração do cristal equivale a 1 microssegundo. Com o bloco comparador é possível medir 1000000 pulsos que equivale a 1 segundo, assim a função simplesmente dispara a interrupção, a interrupção é tratada e a variável global (“timerInterruptFlag”) é definida como verdadeira, conforme o diagrama da Figura 38.

Figura 38 - Diagrama exclusivo da função *timer*

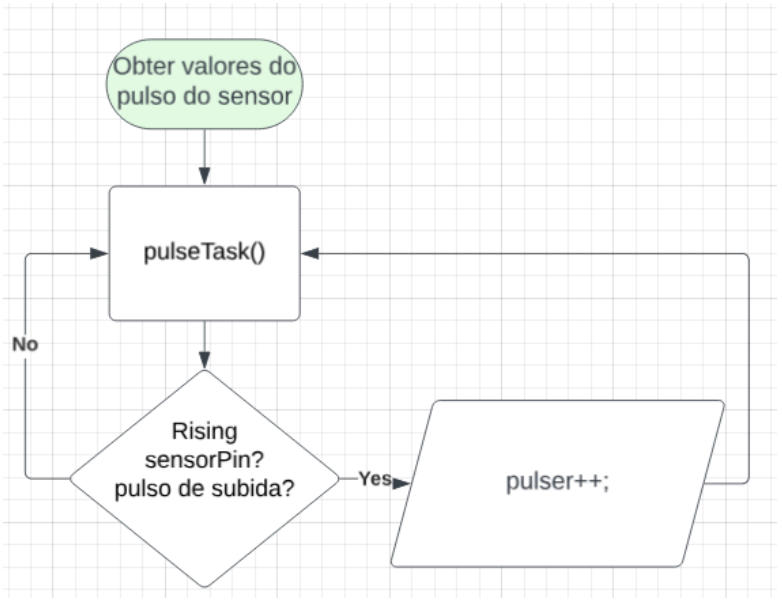


Fonte : Autor(2024)

Seguindo a lógica de que as funções precisam ser executadas concorrentemente para não perder dados ou deixar de seguir tempos precisos, a

primeira função do tipo *task* funciona de uma maneira parecida porém, ela vai captar as interrupções do tipo *RISING*(subida) no pino 4 do microcontrolador, definido como (“SensorPin”), ou seja, a turbina do sensor emite pulsos, então a cada subida de estado o valor vai ser acumulado na variável global (“pulse”). Independente do que estiver ocorrendo na programação, a função apenas acumula os valores obtidos na variável global. Conforme o diagrama de captação dos pulsos da Figura 39.

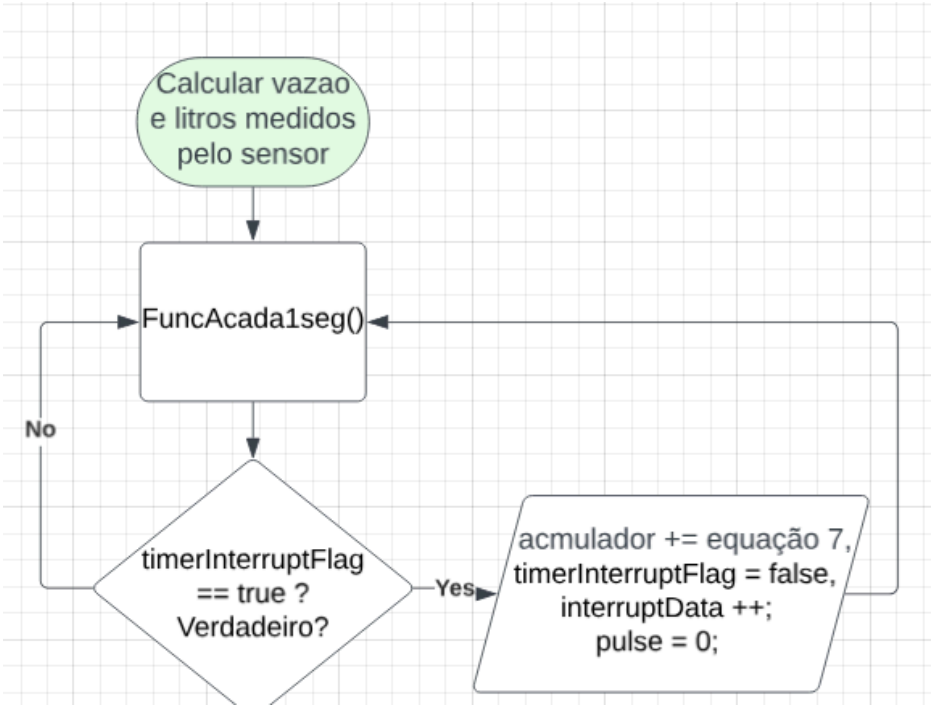
Figura 39- Diagrama exclusivo da função *pulse*



Fonte : Autor(2024)

Na aplicação da Equação 7, que transforma os pulsos captados pela função (“pulseTask”) nos valores de vazão, litros total e volume total, é implementado mais uma função contínua buscando tratar esses valores, (“FuncAcada1seg”), essa função é executada continuamente juntos com as restantes porém necessita de um gatilho de disparo, como mencionado antes. Ela só irá aplicar a equação caso a variável global (“timerInterruptFlag”) for verdadeira, ou seja, a cada disparo da interrupção pelo timer interno que a função é liberada. Quando executada ela usa o valor da variável global (“pulse”), aplica a Equação 7, obtém a vazão em litros/minutos, que é grandeza que o sensor trabalha e divide essa vazão por 60 segundos obtendo o consumo em litros a cada 1 segundo. No final da execução ela armazena esse valor na variável global (“volumeTotal”) que é um acumalador, deixa novamente o estado da (“timerInterruptFlag”) em falso e inicia uma contagem na variável global (“interruptData”), conforme o diagrama da Figura 40, zera os pulsos.

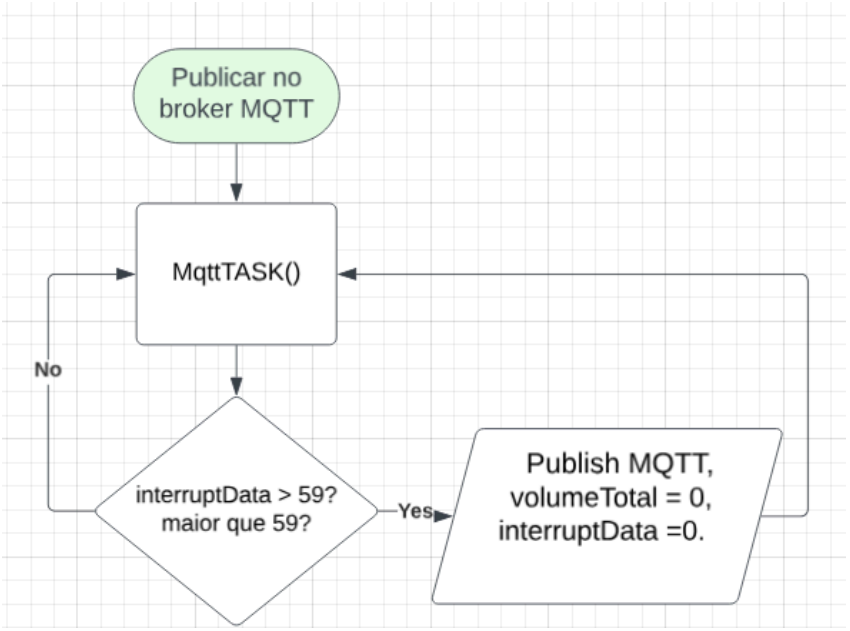
Figura 40 - Diagrama exclusivo da calcular vazão e litros



Fonte : Autor(2024)

Partindo que as três funções citadas anteriormente, que é o núcleo principal do sistema tanto em captar os dados em intervalos e transformá-los em grandezas desejadas estão funcionando, surge a necessidade de comunicação entre a ESP 32 e o *Home Assistant*. A comunicação é através do MQTT e para solucionar esse problema, foi implementado a função (“MqttTASK”) do tipo task, ela segue o mesmo padrão da anterior, necessitando de um gatilho para publicar no tópico. Esse gatilho é a variável global “interruptData”, que aguarda a realização de 60 ciclos de medição para evitar a poluição do banco de dados com valores muito pequenos. Assim, a função acumula 60 valores antes de realizar a publicação no broker. Quando o número de captações de dados é atingido, a função pública no tópico (“caminho/teste”). Ao final da execução, o acumulador (“volumeTotal”) e a variável global (“interruptData”) são zerados para o novo ciclo de 60 valores, conforme ilustrado no diagrama da Figura 41.

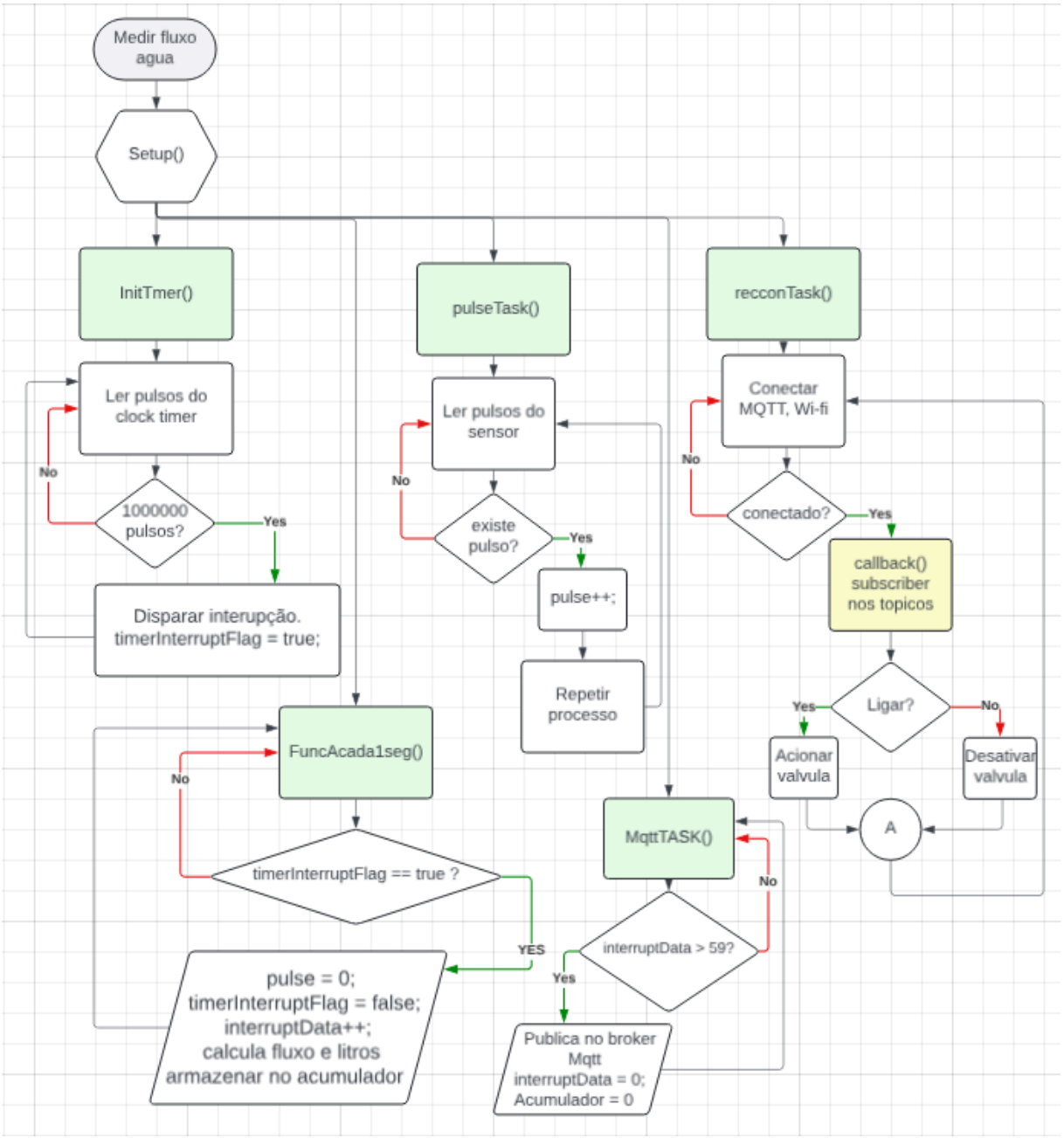
Figura 41 - Diagrama exclusivo da publicar no MQTT



Fonte : Autor(2024)

Para garantir que a conexão fique estável, pois o protocolo escolhido depende de *Internet*, foi implementado uma função para essa finalidade, (“recconTask”). Ela é responsável por ficar analisando se a conexão tanto do MQTT quanto a do Wi-Fi estejam válidas, se em alguma verificação a conexão estiver perdida ela automaticamente irá fazer a conexão novamente, inibindo a possibilidade de perda de dados por conexão. E também como ela está lidando com a comunicação dentro da própria função é chamada a sexta e última que compõem o sistema, (“callback”), que é responsável por ler os tópicos publicados por parte dos outros dispositivos a fim de acionar a válvula. Então com base nas funções apresentadas até agora e seus diagramas de funcionamentos, é possível visualizar como cada função age no sistema de forma simultaneamente e o fluxograma da Figura 42, representa a forma como cada uma delas está sendo executadas assim como os diagramas das últimas duas funções apresentadas.

Figura 42 - Fluxograma Programação ESP 32

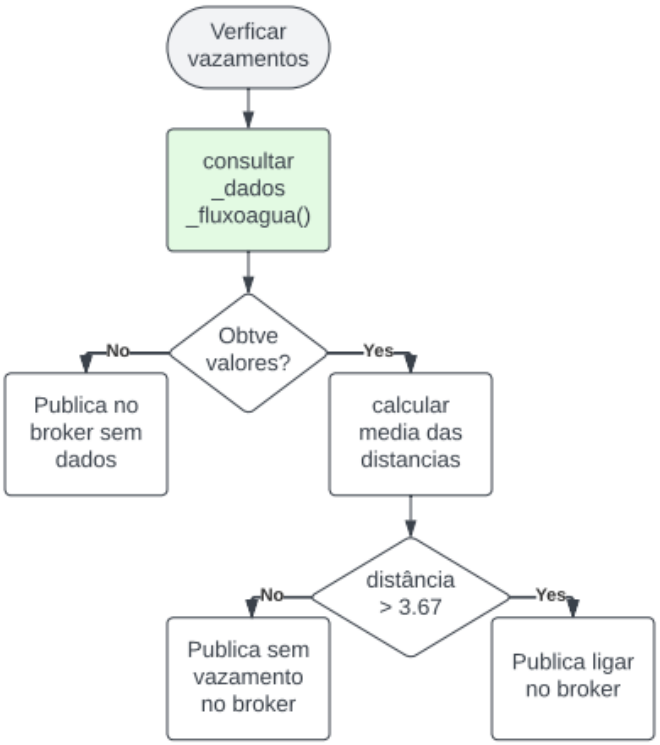


Fonte : Autor(2024)

Na parte da API feita em Python, o fluxograma da função, Figura 43, executa uma consulta REST dentro do banco temporal hospedado no *Home Assistant* obtendo os 30 últimos valores publicados pela ESP 32, ou seja, obtém o padrão de consumo com finalidade de comparar sua distribuição com o padrão do modelo. A função calcula a distância de cada ponto em relação aos seus 200 vizinhos mais próximos, 30 vezes(uma para cada ponto), e desse cálculo é implementado a média

das distâncias, com essa distância é possível analisar se esta distribuição faz parte do modelo. O resultado final é publicado no *broker* e dependendo do valor fecha a válvula ou não a cada 15 minutos, esse tempo foi estabelecido pela automação do *Home assistant*. Vale ressaltar que no intervalo de execução os últimos 15 dados serão novos. O valor 3.67 onde é utilizado para avaliar a distância permitida, é o limiar que é calculado no tópico de treinamento do modelo.

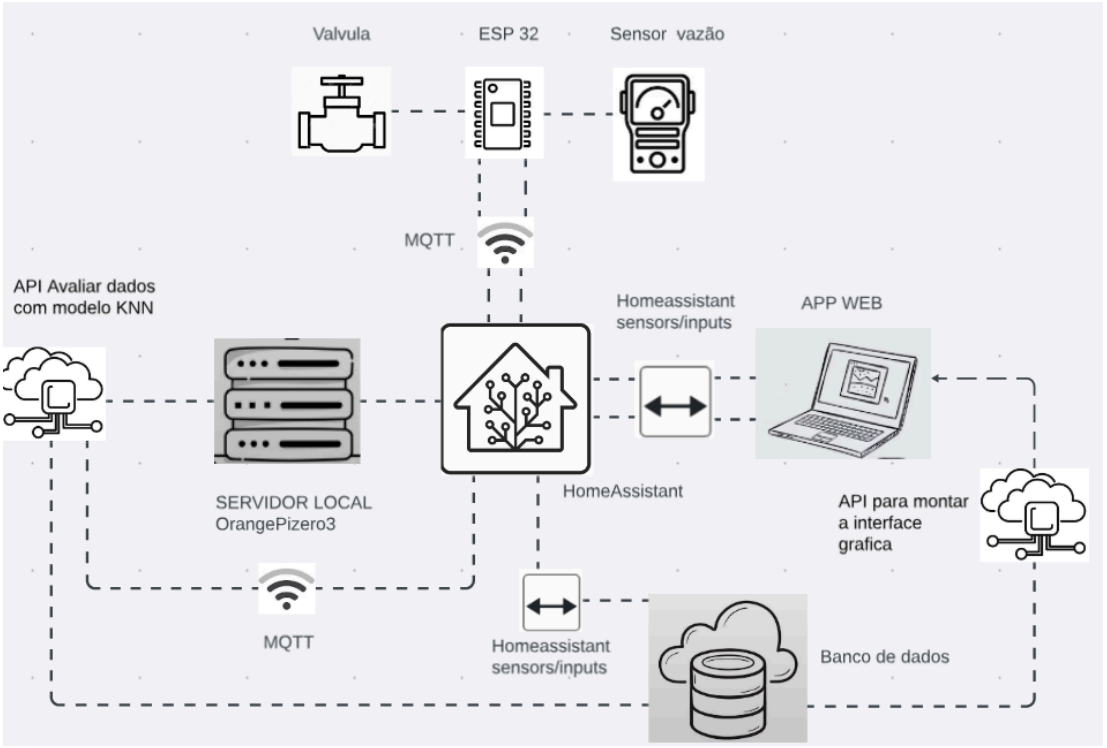
Figura 43 - Fluxograma API python



Fonte : Autor(2024)

O sistema com todos seus módulos juntos foi projetado para funcionar de uma maneira harmônica concentrando todas as comunicações para o broker MQTT utilizando sua característica de suportar inúmeros dispositivos publicando e subscrevendo nos tópicos, cada componente consegue interagir com os próprios tópicos para realizar suas ações programadas. A Figura 44 é um diagrama geral de comunicação entre módulos e ilustra de uma maneira mais detalhada que existe um centralizador de informações que é o *broker* MQTT, a partir dele as informações são direcionadas para cada componente.

Figura 44 - Diagrama geral do protótipo



Fonte : Autor(2024)

O protótipo opera com a ESP 32 captando os pulsos do sensor de vazão. Quando a interrupção do timer interno é disparada, aplica-se a Equação 7, cujo resultado fornece o valor em litros e a vazão em litros por minuto. Esses dados são armazenados em um acumulador durante 60 ciclos, ou 1 minuto, conforme o timer interno. Após a coleta de 60 medidas, os valores são publicados no *broker*, onde o *Home Assistant* se inscreve no tópico correspondente, obtendo e armazenando os dados no banco de dados temporal.

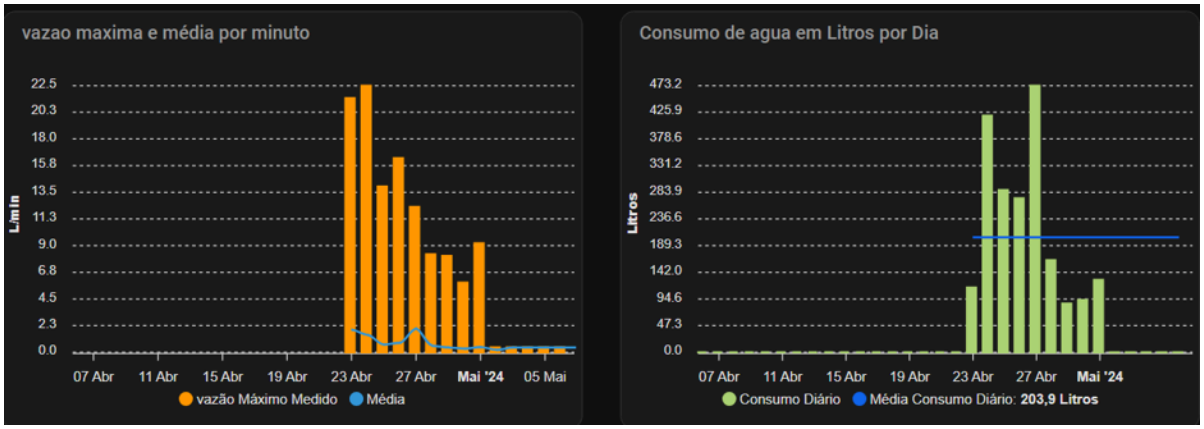
Para manipular as variáveis, utilizam-se automações que publicam no *broker*, tanto para abrir e fechar a válvula quanto para avaliar o consumo a cada 15 minutos. Essa avaliação aciona uma API em Python, que publica os resultados da análise de consumo no *broker*, permitindo ativar a automação de envio de mensagens e fechar a válvula, se necessário.

3.4 Coleta de dados

No levantamento de dados, o objetivo é treinar o modelo, então o procedimento foi de desativar a API.

O sistema agora só obtém os dados e armazena no banco temporal, em um total foram 8 dias de coleta. Para uma maior confiabilidade, no segundo dia, 24 de abril de 2024, foi tirado uma foto do registro hidrômetro da CASAN no início e no final para comparar com os valores mostrados na aplicação gráfica, conforme a Figura 45.

Figura 45- Diagrama comunicação entre dispositivos



Fonte : Autor(2024)

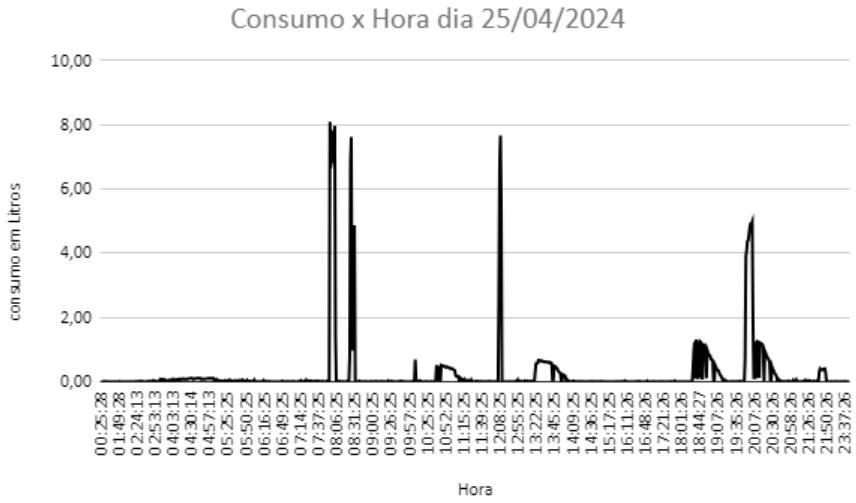
Percebe-se que a média de consumo diário é de 203,9 litros. No dia 24 obteve-se um consumo captado pelo protótipo perto de 425 litros, e conforme a Figura 46, fotos do hidrômetro, ao subtrair os valores marcado pelo relógio, 0415328 e 0415775 o resultado é de 447 litros, comprovando que o sistema está captando valores próximos aos marcados pelo hidrômetro.

Figura 46 - Relógio marcando 1 dia de consumo

Fonte : Autor(2024)

Com o protótipo captando os valores bem próximos aos que são medidos pelo relógio e inserindo eles conforme o tempo no banco temporal, é possível parametrizar os dados e tirar informações importantes. A figura 47, ilustra o consumo entre o início do dia 25 de abril e o final, os valores predominantes são zero, porém existe uma variação em determinados horários.

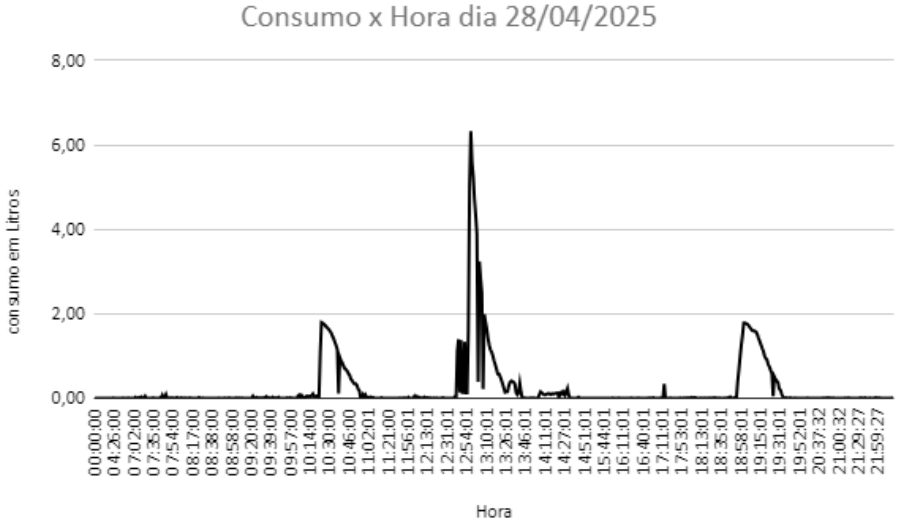
Figura 47 - Fornecimento de água



Fonte : Autor(2024)

Analisando outros dias, o comportamento é semelhante, conforme a figura 48, existe o início do fornecimento e o tempo até encher a caixa.

Figura 48- Fornecimento de água de outro dia



Fonte : Autor(2024)

No gráfico da Figura 47, como existem muitos pontos, há uma distorção da representação gráfica, porém a Figura 49 no intervalo de 30 minutos, é possível observar que o fornecimento possui as características mencionadas.

Figura 49- Fornecimento de água em 30 minutos do dia 25

Fonte : Autor(2024)

Analisando as distribuições dos resultados coletados nos 8 dias e os gráficos das Figuras 47 e 48, é possível visualizar o tempo de fornecimento que varia entre 15 a 20 minutos, e esse padrão possibilita treinar o modelo com a hipótese que o fornecimento de água demora esse intervalo de tempo para encher o reservatório e não existe um fornecimento contínuo, ou seja, os valores ao longo do tempo atingem um limite máximo de consumo e possui características de valores decrescentes a partir de um certo momento, assumindo valores igual a zero.

Ao utilizar o algoritmo KNN é possível avaliar os pontos vizinhos e calcular a média da distância entre os novos pontos e os antigos, assim detectando esse padrão.

3.5 Treinamento do modelo

Para treinar um modelo, primeiramente precisa-se de uma hipótese. Com os gráficos da Figura 47 e 48 analisados anteriormente é possível assumir que o

modelo segue a hipótese que o fornecimento não é contínuo, ou seja, em uma distribuição os dados não assumem valores constantes, sendo que existe um pico com valores de menor concentração e uma base com maior concentração. Na distribuição da Figura 50 é possível ver a concentração de todos os pontos captados que serão utilizados no treinamento do modelo.

Figura 50 - Distribuição dos pontos

Fonte : Autor(2024)

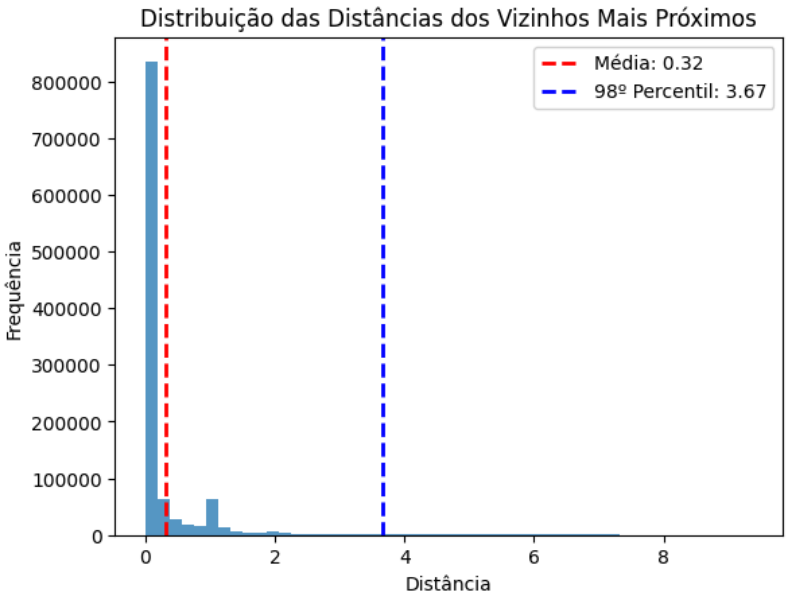
Analisando a distribuição da Figura 50, nota-se que a concentração dos dados fica próxima ao zero, que de fato é o esperado pois o reservatório está cheio a maior parte do tempo, os demais dados quanto maior, menos concentrado está, pois é o abastecimento de água até o enchimento do reservatório.

Como o KNN classifica os novos pontos em relação a duas classes diferentes com base na distância, e no caso do projeto, existe apenas uma classe, a solução é aplicar um limiar de distância entre os dados conhecidos.

Para definir o limiar, é utilizado a distância média do intervalo permitido, ou seja, é definido um intervalo de consumo e é calculado a distância daquele ponto aos seus vizinhos mais próximos e posteriormente pega-se a média da soma do intervalo, o valor escolhido é o 98º percentil da base de dados de todas as distância entre os pontos, 3.67 como ilustrado na Figura 51. Significa que 98% dos valores de

distância dos pontos estão sendo cobertos, possibilitando comparar a distância média dos novos pontos com o limiar, que é uma das maiores distância encontrada entre a média do intervalo dos pontos. Valor bem conservador que pode ser ajustado conforme testes.

Figura 51 - Distribuição de todos as distâncias em relação aos vizinhos



Fonte : Autor(2024)

O modelo treinado analisa a distância média dos últimos 30 pontos de um intervalo, dos quais 15 são novas aquisições. Para cada ponto, calcula-se a distância em relação aos vizinhos, resultando em uma média que é comparada com o limiar treinado da mesma forma. Se os 30 pontos se afastam do padrão esperado — que normalmente inicia próximo de zero, sobe até um pico de abastecimento e, em seguida, desacelera até zero — isso indica um vazamento.

Em uma distribuição contínua, se houver 30 valores acima de 5 litros, não devem existir pontos próximos de zero, o que faz com que a média das distâncias ultrapasse o limiar, indicando um vazamento. Por outro lado, se a distribuição apresentar valores de consumo elevados, como mostrado nas Figuras 47 e 48, os valores subsequentes não serão contínuos e tenderão a decrescer em direção a zero, respeitando o modelo e fazendo com que a média das distâncias fique abaixo do limiar.

O modelo se resume a identificar uma distribuição contínua e analisar a distância média dos pontos que a compõem para determinar se há um vazamento. Isso se baseia na observação de que o fornecimento de água não é contínuo. Em uma situação de vazamento, o reservatório esvazia à medida que o reabastecimento demora, até que o fornecimento de água retorne com pressão, causando uma distribuição contínua que ultrapassa o limiar, pois esse intervalo não respeita os picos e vales esperados

Para treinar o modelo, utiliza-se o *Google Colab*, aproveitando seu poder de processamento com a biblioteca *scikit-learn* (*sklearn*). Os dados são extraídos do banco temporal em um arquivo Excel, e o número de vizinhos próximos é definido como 200. A equação matemática utilizada é o de *Manhattan*, conforme mencionado anteriormente. Ao final do treinamento, o arquivo é salvo em formato binário para uso posterior do modelo, e o código do treinamento pode ser encontrado no apêndice (C). É importante destacar que, se o treinamento usa 200 vizinhos, a avaliação também buscará 200 vizinhos em relação ao novo ponto.

3.5.1 Aplicando o protótipo

Partindo do princípio que na rede não existe um vazamento, é necessário forçar ou replicar algo parecido. Na hipótese de que, ao existir um vazamento e seguindo o padrão do fornecimento de água ocorrer em espaços de tempo variados, o reservatório vai esvaziar ou consumir boa parte. Quando a companhia de distribuição restabelecer o envio de água, o fornecimento não vai respeitar o modelo adquirido, ou seja, esse fornecimento vai ficar contínuo até encher a caixa, porém em uma situação normal o consumo da residência não chega a esvaziar a caixa e mantém um padrão de abastecimento entre 15 a 20 minutos em taxas desordenadas mantendo uma distância média abaixo do limiar de 3.67, nessa hipótese que simula um vazamento o fornecimento de água é contínuo por mais de 20 minutos, sem a devida desaceleração jogando a média das distâncias para um valor acima do limiar.

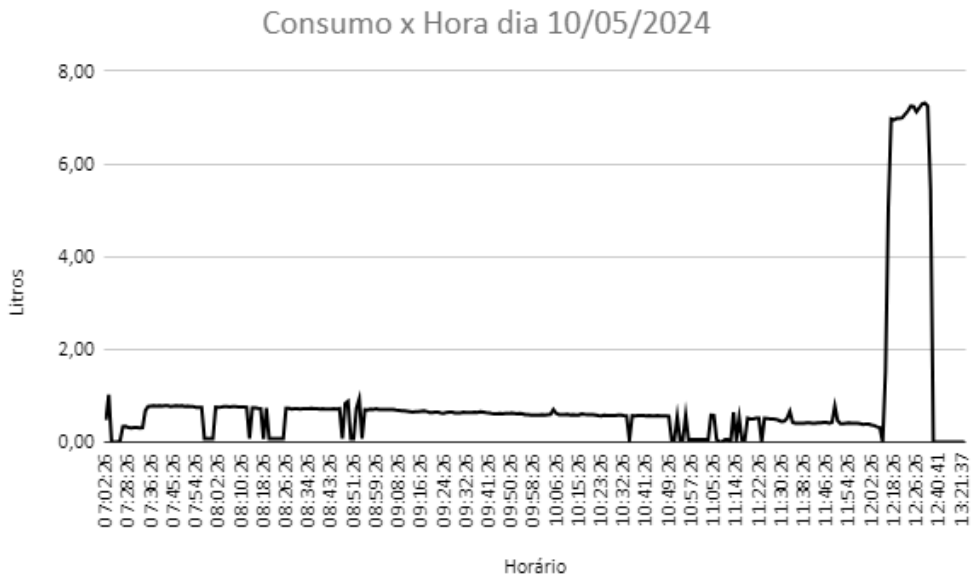
No experimento, buscando realizar um consumo contínuo, simulando um vazamento, foi fechado o registro no dia 6 maio de 2024 até 10 maio, quando foi reaberto novamente, o reservatório está vazio, possibilitando aplicar o modelo e

testar se de fato está funcionando como o esperado. O teste foi baseado em deixar o registro aberto até o sistema identificar um vazamento.

4 PREDIÇÃO E COLETA DOS RESULTADOS

O resultado do teste na aplicação do sistema apresentou resultados interessantes, dia 10 de maio, o registro foi aberto por volta das 7 horas da manhã, e o fornecimento de fato começou, porém como é possível visualizar no gráfico da Figura 52 existe uma certa continuidade do fornecimento de água confirmando a hipótese de fornecimento contínuo, entretanto são valores bem baixos de abastecimento, indicando que naquele momento existe uma possível falta de pressão na rede, esse comportamento prosseguiu até meio dia, que foi quando a pressão na rede foi restabelecida e começou enviar em uma vazão maior porém ainda seguindo o mesmo padrão de antes, de um fornecimento contínuo de água.

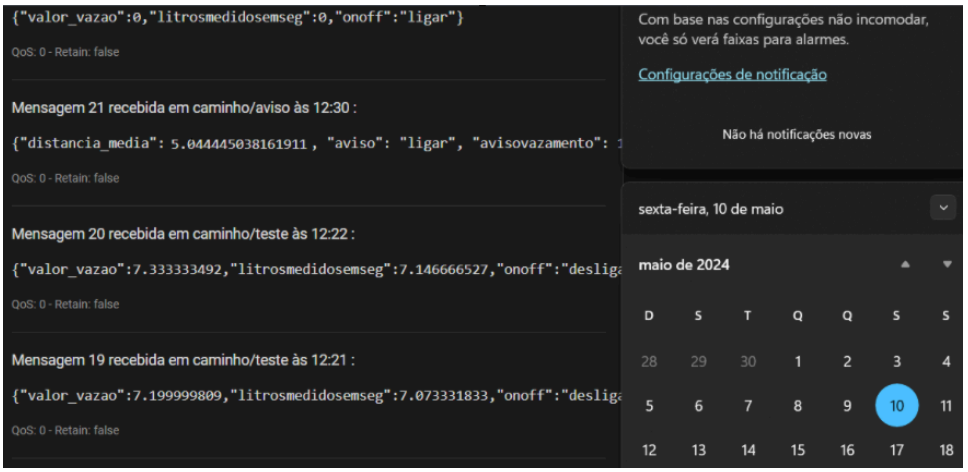
Figura 52 - Distribuição de água no dia do teste



Fonte : Autor(2024)

Durante todo o teste do protótipo foram monitorados os tópicos e valores que estavam sendo publicados no *broker,* e a Figura 53 mostra o funcionamento dos tópicos mencionados anteriormente e o momento que o sistema identifica o vazamento.

Figura 52 - Monitoramento dos tópicos

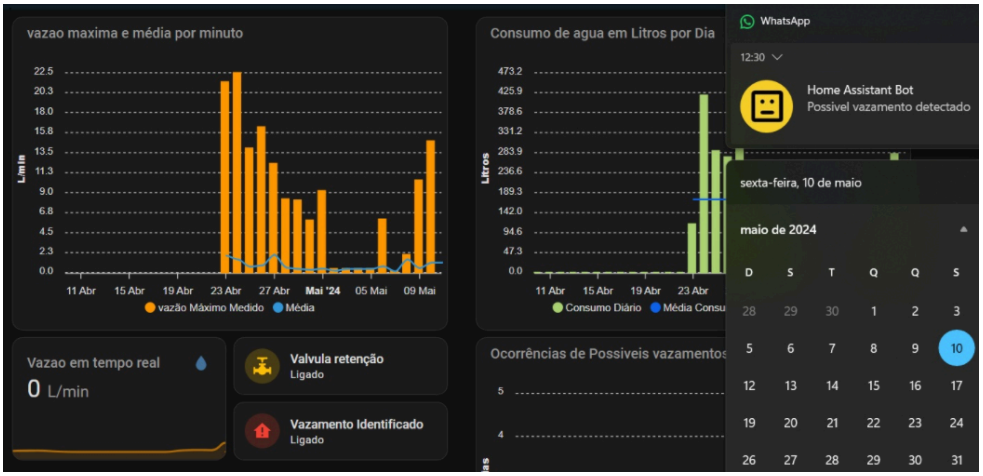


Fonte : Autor(2024)

É possível ver os tópicos caminho/teste para leitura dos sensores, e o tópico caminho/aviso com a propriedade “ligar” indicando que o sistema funcionou tanto na parte de fechar a válvula e identificar o vazamento com o modelo treinado, como na parte eletrônica de captar os dados da vazão. O sistema e a hipótese apresentaram resultados favoráveis, porém gera uma discussão em relação ao modelo, quanto à falta de pressão na rede.

A Figura 53 ilustra o funcionamento de enviar a mensagem de possíveis vazamentos para o proprietário da residência e o *dashboard* final com as duas propriedades *inputs* indicando que estão ligadas junto com a vazão em tempo real no valor de 0, já que a válvula está fechada. O aplicativo do *whatsapp* está instalado no computador, porém no celular foi notificado igualmente.

Figura 53 - Resultado *dashboard* e *whatsapp*



Fonte : Autor(2024)

5 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Referente aos resultados, é possível concluir que o sistema funciona conforme o proposto, identificando vazamentos com base no modelo treinado. No entanto, é importante destacar que pode ocorrer uma falta de abastecimento de água, resultando em falsos positivos. Por exemplo, se o reservatório esvaziar e o fornecimento for restabelecido, isso pode ser interpretado como um vazamento, já que a caixa d'água tende a esvaziar, e os dados subsequentes podem fugir do padrão esperado.

Outra questão relevante é a falta de pressão durante um vazamento. Mesmo que o fornecimento de água pareça contínuo, ele pode não respeitar o modelo de picos de abastecimento e desacelerações de consumo, resultando em uma vazão muito baixa. Nesse caso, a distância média do intervalo avaliado pelo modelo não ultrapassa o limiar de 3,67.

Isso não significa que o algoritmo seja incapaz de identificar essas situações. Uma solução seria aumentar o número de amostras analisadas e diminuir o limiar. Atualmente, o modelo avalia 30 medições a cada 15 minutos; aumentar esse valor para 100 aumentaria a probabilidade de identificar padrões, embora também eleve o risco de falsos positivos. Esse risco pode ser mitigado ao incluir mais pontos vizinhos no treinamento do modelo ou ajustando o limiar para um valor menos conservador.

Em casos de um vazamento ser muito pequeno será difícil o modelo acusar ele, pois esse vazamento não chega a ocasionar uma diminuição grande no reservatório.

6 CONSIDERAÇÕES FINAIS

No geral a implementação foi bem sucedida, a interface visual com *apex charts* e Home Assistant utilizando API para consulta no banco temporal se provou bem agradável de visualizar e funcional. Os sensores, inputs e automações junto com o *broker Mosquitto* centralizando as informações enviadas entre a ESP 32, Home Assistant e API validando o modelo, funcionaram como esperado. O banco temporal do *influxDB* para armazenar os dados, o treinamento de modelo KNN com seu carimbo de data e hora da publicação mostrou-se confiável e seguro.

Na parte de aquisição de dados, a interrupção disparada a cada 1 segundo por meio de temporizadores, junto à captura dos pulsos enviados pelo sensor de vazão de efeito *Hall*, demonstrou ser um método confiável para trabalhar com a ESP 32. A utilização de funcionalidades do *FreeRTOS* em conjunto com o Arduino potencializa essa abordagem, permitindo o gerenciamento eficiente de funções multitarefas. Assim, o protótipo pode operar de forma independente, assegurando que as medições sejam realizadas com precisão.

Como se trata de uma automação residencial, a placa *Orange* Pi, hospedando o *smart hub*, é uma solução custo-benefício em relação à *Raspberry* Pi. Ela funciona muito bem, permitindo a instalação do *Home Assistant* e possibilitando a execução de APIs dentro do sistema operacional. Isso permite avaliar modelos de aprendizado de máquina que podem ser aplicados em diversas áreas, tanto no âmbito residencial quanto industrial.

No contexto comercial essa aplicação satisfaz os quesitos, porém a parte física e de instalação não é muito apreciável pois, necessita de achar a encanação, realizar a escavação e a instalação do protótipo. O buraco é um incômodo e pode encher de água caso chova.

6.1 Sugestões para trabalhos futuros

Com base nos resultados e aprendizados adquiridos durante a execução do projeto, é possível sugerir algumas sugestões para pesquisas futuras. Buscar aproveitar mais o sistema de fornecimento de água disponibilizado pela CASAN, o hidrômetro em específico, automatizando a válvula manual. É interessante obter o

fluxo de água de maneira menos agressiva possível, um sistema de aquisição por ultrassônico ou por sistema de visão são boas opções. Outra sugestão é contornar erros de falsos positivos por falta de abastecimento de água, implementando uma lógica na API que avalia o consumo do dia antes de aplicar o modelo, ou seja, se não tiver acontecido abastecimento durante o dia ela consiga identificar que é um falso positivo de falta de água, outra opção é estudar modelos com redes neurais mais complexas para evitar falsos positivos tanto da falta de água quanto falta de pressão. Uma outra sugestão é fazer com que o modelo treine sozinho durante um período para depois começar aplicar as análises, pois cada residência ou condomínio tem um padrão de consumo.

Um ponto positivo do sistema *smart-hub*, é a possibilidade de montar uma cadeia de automação residencial, analisar consumo de energia, câmeras com inteligência artificial, sensores de presenças entre outras aplicações de casas inteligentes.

REFERÊNCIAS

Companhia Catarinense de Águas e Saneamento (CASAN). **2ª Edição, Revista e Atualizada. Divisão de Políticas Comerciais-DIPCO/GCO**. Florianópolis (SC), outubro de 2015.

FRAZÃO, Dilva. **Resumo da biografia de Blaise Pascal**. Blog ebiografia, 2023. Disponível em: <https://www.ebiografia.com/blaise_pascal/.> Acesso em: 07/03/2024.

NERIS, R. F. . **INSTRUMENTAÇÃO E CONTROLE DE PROCESSOS NA INDÚSTRIA. Revista Ibero-Americana de Humanidades, Ciências e Educação, [S. l.], v. 8, 1269–1276, 2022. DOI: 10.51891/rease.v8i3.4698**. Disponível em: <https://periodicorease.pro.br/rease/article/view/4698>. Acesso em: 07/03/ 2024.

PRADO, J. Alcindo do. **Controle de Processos Industriais**, 2010.

DINO. **Desperdício de água: vazamentos são responsáveis por 60%**. 13 de janeiro de 2022, atualizado em 13 de janeiro de 2022. Disponível em: <https://www.metropoles.com/dino/desperdicio-de-agua-vazamentos-sao-responsaveis-por-60>. Acesso em: 9 de março de 2024.

THOMAZINI, Daniel. ALBUQUERQUE, Pedro U. B. **Sensores Industriais - Fundamentos e Aplicações**. 5ª ed. São Paulo: Érica, 2005.

TIPLER, Paul A. **Física para Engenheiros**, Sexta edição, Vol. 2, 2009.

KERSCHBAUMER, Ricardo. **Engenharia de Controle e Automação - Microcontroladores**. 2018. notas de aula. 180 p.

FERNANDES, Anita M. da R. **Inteligência Artificial noções gerais**, 1ª ed. São Paulo: Visual Books, 2005.

MORAIS, José V. S. **O guia profissional ESP32 com IDF**, 1ª ed. São Paulo: Agbook, 2023.

FRANCHI, Claiton Moro. **Instrumentação de Processos Industriais – Princípios e Aplicações**. 1ª edição , Érica; 18 junho de 2018.

Shenzhen Xunlong Software Co. Limited. **Manual da Orange PI.** Disponível em: <https://geekmatic.in.ua/pdf/OrangePi_PC_user_manual_v0.9.1.pdf.> Acesso em: 14/03/2024..

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial: Uma Abordagem Moderna**. 2. ed. Rio de Janeiro: Elsevier, 2004. 1082 p.

GUIMARÃES, Paulo Ricardo Bittencourt. **Métodos Quantitativos Estatísticos**. Curitiba: IESDE Brasil S.A., 2008. 245 p.

HEMINI, **Helder Anibal. Atuadores. Rio de Janeiro: PETROBRAS – Petróleo Brasileiro S.A**., 2007. 103 p.

PARKER HANNIFIN CORPORATION. **Válvulas e Acessórios para Controle de Fluido. Catálogo 4201-5 BR. Controle em processos críticos e segurança**. 1ª ed. 2022.

GOUVEIA, M.; GOUVEIA, V. **Service oriented architecture (soa): Desafios para o processo de desenvolvimento de software**. INESC, 2004. p.33.

MQTT.ORG. **Frequently Asked Questions. MQTT.ORG**, 2014. Disponível em: <http://mqtt.org/faq>. Acesso em: 03 mar. 2024.

FARIA, Artur Bersan de. **Automação Residencial com Foco em Cenários**. Orientador: Dr. Renato Coral Sampaio. Brasília, DF, 2018.

FIELDING, Roy Thomas. **Architectural Styles and the Design of Network-based Software Architectures**. PhD diss. University of California, Irvine, 2000.

GAMBOA-MEDINA, M. **DETECÇÃO DE VAZAMENTOS E ALTERAÇÕES EM REDES DE DISTRIBUIÇÃO DE ÁGUA PARA ABASTECIMENTO, DURANTE A OPERAÇÃO, USANDO SINAIS DE PRESSÃO**. 2017. 130p. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2017.

ARTEIRO, R. D. et al. **Utilizando redes de petri para modelagem de desempenho de middleware orientado a mensagem**. WPerformance, 2007. p. 34.

KALE, V. **Cloud computing for business and technology managers: from distributed computing to cloud aware applications**. [S.l.]: CRC Press, 2015. v. 1.

ROUSE, M. **Smart home hub (home automation hub)**. 2018. Online; Acesso em: 16 mar. 2024. Disponível em: <https://internetofthingsagenda.techtarget.com/definition/smart-home-hub-home-automation-hub>.

BAUER, P. D. F. L. BOLLIET, D. H. J. H. P. L. **Software engineering. In: . [S.l.]: Peter Naur and Brian Randel**l, 1968. (NATO SCIENCE COMMITTEE). p.32 p.33.

MILANI DE LUCENA, Marcos. **Sistema de Visão Computacional para Automação do Processo de Fabricação de Válvula Esfera Tripartida**. 2024.

JOSEPH, Jobit. **ESP32 Timers & Timer Interrupts.** 23 de junho de 2022. Online; Acesso em: 11 abr. 2024. Disponível em: <https://circuitdigest.com.com/microcontroller-projects/esp32-timers-and-timer-interrupts>.

BRUNETTI, Franco. **Mecânica dos Fluido**s. 2. ed. São Paulo: LTC, 2003.

B'FAR, Reza; FIELDING, Roy T. **Mobile Computing Principles: Designing and Developing Mobile Applications with UML and XML**. New York: Cambridge University Press, 2009.

JESUS, Edmilson dos Santos de. **Modelos de aprendizagem de máquina para previsão da demanda de água da Região Metropolitana de Salvador, Bahia.** 2023. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal da Bahia, Salvador, 2023.

GALHARDO, Victor; ROCHA, Jéssica Oliveira; SODEK, Daniela Bonazzi. **Aplicações de aprendizado de máquina para pesquisa de vazamentos em tubulações.** Orientador: Prof. Dr. Andre Luis Sotero Salustiano Martim. 2023. Trabalho de conclusão de curso (Graduação em Engenharia Civil) – Faculdade de Engenharia Civil, Arquitetura e Urbanismo, Universidade Estadual de Campinas, Campinas, 2023.

ALI, Nidal Martins. **Utilizando o Algoritmo K-Nearest Neighbors (KNN) para um sistema de recomendação de mapas gerados proceduralmente**. 2022. Trabalho de conclusão de curso (Graduação) – UNIVERSIDADE FEDERAL DE PELOTAS, Pelotas, 2022.

BRANDÃO, Diogo Alves. **Inteligência Artificial e Aprendizado de Máquina: da teoria ao algoritmo pronto no Ensino Médio.** 2023. Dissertação (Mestrado Profissional em Matemática em Rede Nacional) – Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Matemática, Brasília, 2023.

RAHMAN, W. ***Inteligência Artificial e Aprendizado de Máquina.*** [S.l.]: Editora Senac São Paulo, 2022. ISBN 978-85-396-3278-7.

APÊNDICES

APÊNDICE A – Configurações e serviços Home Assistant

```
1. # Loads default set of integrations. Do not remove.
2. default_config:
3.
4. # Load frontend themes from the themes folder
5. frontend:
6.  themes: !include_dir_merge_named themes
7.
8. automation: !include automations.yaml
9. script: !include scripts.yaml
10.   scene: !include scenes.yaml
11.
12.   python_script:
13.
14.
15.
16.   logger:
17.    default: info
18.
19.   # serviço de whatsapp
20.   notify:
21.    - name: WhatsApp
22.      platform: rest
23.      resource: https://api.callmebot.com/whatsapp.php
24.      data:
25.        source: HA
26.        phone: +5548955555555   #enter your phone number here
27.        apikey: 6364463        #enter your apikey here (see
   Setup above)
28.
29.   rest_command:
30.    chamar_api:
31.      url: 'http://10.0.0.6:5000'  # Coloque a URL completa do
   seu endpoint aqui
```

```
32.      method: GET  # Ou POST, dependendo do método HTTP que
   você precisa usar
33.      content_type: application/json  # Se necessário
34.      timeout: 10  # Ajuste conforme necessário
35.
36.   influxdb:
37.    host: 10.0.0.6
38.    port: 8086
39.    username: felipe
40.    password: felipe
41.    database: fluxoagua
42.    max_retries: 3
43.    tags:
44.      source: HA
45.    tags_attributes:
46.      - friendly_name
47.    default_measurement: "water_volume_L"
48.    include:
49.      entities:
50.        - sensor.water
51.
52.   sensor:
53.    #INFLUX DATABASE READ
54.    - platform: template
55.      sensors:
56.        influxdb_read:
57.          friendly_name: "Influx DB Read"
58.          value_template: ""
59.    - platform: template
60.      sensors:
61.        influxdb_read_2:
62.          friendly_name: "Influx DB Read 2"
63.          value_template: ""
64.
65.
66.   mqtt:
67.
```

```
68.  sensor:
69.   - name: "vazao"
70.     unique_id: vazao123
71.     state_topic: "caminho/teste"
72.     unit_of_measurement: "L/min"
73.     value_template: "{{ value_json.valor_vazao }}"
74.     json_attributes_topic: "caminho/teste"
75.     state_class: measurement
76.     device_class: volume_flow_rate
77.
78.
79.   - name : "litrosmedidosemseg"
80.     unique_id: vazao
81.     state_topic: "caminho/teste"
82.     unit_of_measurement: "L"
83.     value_template: "{{ value_json.litrosmedidosemseg }}"
84.     json_attributes_topic: "caminho/teste"
85.     state_class: measurement
86.
87.
88.   - name : "avisosvazamentos"
89.     unique_id: avisovazamento
90.     state_topic: "caminho/teste"
91.     unit_of_measurement: "ocorrências"
92.     value_template: "{{ value_json.avisovazamento }}"
93.     json_attributes_topic: "caminho/teste"
94.     state_class: measurement
95.
96.
97.  input_boolean:
98.   notify_val:
99.     name: "Valvula retenção"
100.    icon: mdi:pipe-valve
101.
102.  notify_vazamento:
103.    name: "estadovazamento"
104.    icon: mdi:home-alert
```

APÊNDICE B – Automações Home Assistant

```
1. - id: chamar_endpoint
2.  alias: Chamar Endpoint da API
3.  trigger:
4.    platform: time_pattern
5.    minutes: /15
6.  action:
7.  - service: rest_command.chamar_api
8.
9. - id: envia_notificacao_MQTT_ON
10.    alias: Enviar MQTT quando Input Boolean é Ligado
11.    trigger:
12.    - platform: state
13.      entity_id: input_boolean.notify_val
14.      to: 'on'
15.    action:
16.    - service: mqtt.publish
17.      data:
18.        topic: esp/test
19.        payload: ligado
20.
21.   - id: envia_notificacao_MQTT_OFF
22.    alias: Enviar MQTT quando Input Boolean é desligado
23.    trigger:
24.    - platform: state
25.      entity_id: input_boolean.notify_val
26.      to: 'off'
27.    action:
28.    - service: mqtt.publish
29.      data:
30.        topic: esp/test
31.        payload: desligado
32.
33.   - id: envia_notificacao_MQTT_INICIO
34.    alias: Enviar MQTT com valor do Input Boolean
```

```
35.    trigger:
36.    - platform: mqtt
37.      topic: caminho/inicio
38.    action:
39.    - service: mqtt.publish
40.      data:
41.        topic: esp/test
42.        payload_template: "{% if
  is_state('input_boolean.notify_val', 'on') %}\n  ligado\n{%
43.          else %}\n  desligado\n{% endif %}"
44.
45.   - id: Aviso whatsapp via mqtt
46.    alias: envia mensagem WhatsApp
47.    trigger:
48.    - platform: mqtt
49.      topic: caminho/aviso
50.    condition:
51.    - condition: template
52.      value_template: '{{ trigger.payload_json.aviso ==
  ''ligar'' }}'
53.    action:
54.    - service: notify.whatsapp
55.      data:
56.        message: Possivel vazamento detectado
57.
58.   - id: Aviso via mqtt aviso
59.    alias: liga alerta vazamento
60.    trigger:
61.    - platform: mqtt
62.      topic: caminho/aviso
63.    condition:
64.    - condition: template
65.      value_template: '{{ trigger.payload_json.aviso ==
  ''ligar'' }}'
66.    action:
67.    - service: input_boolean.turn_on
68.      target:
```

```
69.         entity_id: input_boolean.notify_vazamento
70.
71.   - id: Aviso via mqtt vauvula
72.    alias: liga alerta valvula
73.    trigger:
74.    - platform: mqtt
75.      topic: caminho/aviso
76.    condition:
77.    - condition: template
78.      value_template: '{{ trigger.payload_json.aviso ==
   ''ligar'' }}'
79.    action:
80.    - service: input_boolean.turn_on
81.      target:
82.        entity_id: input_boolean.notify_val
83.
```

APÊNDICE C – Treinando o modelo KNN com google colab

```
1. import pandas as pd
2. from sklearn.neighbors import NearestNeighbors
3. import joblib
4. import numpy as np
5.
6.
7. # Carregar o arquivo Excel
8. df = pd.read_excel('dados.xlsx')
9.
10.   # Filtrar as linhas onde as colunas 'Hora' e 'dados' não são
   nulas
11.   df = df.dropna(subset=['Hora', 'dados'])
12.
13.
14.   df['Hora'] = df['Hora'].str.split(':').str[0]
15.   # Preencher valores ausentes com um valor padrão (0, por
   exemplo)
16.   df['Hora'] = df['Hora'].fillna(0)
17.
18.   # Converter a coluna 'dados' para float, caso necessário
19.   df['dados'] = df['dados'].astype(float)
20.   df['Hora'] = df['Hora'].astype(float)
21.
22.   # Criar uma lista de tuplas (hora, dados_somados)
23.   lista_pontos = list(zip(df['Hora'], df['dados']))
24.
25.   # Inicializando o modelo KNN sem rótulos
26.   modelo_knn = NearestNeighbors(n_neighbors=200,
   metric='manhattan')  # Definindo o número de vizinhos
27.
28.   # Treinando o modelo
29.   modelo_knn.fit(lista_pontos)
30.
31.   # Salvando o modelo treinado
32.   joblib.dump(modelo_knn, 'meu_modelo.pkl')
```

```
33.
34.   # Avaliando a distância dos vizinhos de todos os pontos no
   conjunto de dados
35.   distancias, _ = modelo_knn.kneighbors(lista_pontos)
36.
37.   # Calcular a média das distâncias
38.   media_distancias = np.mean(distancias)
39.   print(f"Média das distâncias dos vizinhos de todos os
   pontos: {media_distancias:.2f}")
40.
41.   # Novos dados para avaliar a distância dos vizinhos
42.   novos_dados = [(12, 5.45), (12, 5.04),(12, 6.97),(12,
   6.94),(12, 6.98),(12, 6.98),(12, 6.99),(12, 7.07),(12,
   7.24),(12, 7.23),(12, 7.12),(12, 7.20),(12, 7.29),(12,
   7.31),(12, 7.24)]  # Convertendo para segundos
43.
44.   # Avaliando a distância dos vizinhos dos novos dados
45.   distancias, indices_vizinhos =
   modelo_knn.kneighbors(novos_dados)
46.
47.   mediadistanciz = 0;
48.   print("Distâncias dos vizinhos dos novos dados:")
49.   for i, (distancia, vizinhos) in enumerate(zip(distancias,
   indices_vizinhos)):
50.       print(f"Novo dado {i+1}:")
51.       distancias_vizinhos = []
52.       for j, (dist, indice) in enumerate(zip(distancia,
   vizinhos), 1):
53.           print(f"Vizinho {j}: Distância = {dist:.2f}, Índice =
   {indice}, Dado = {lista_pontos[indice]}")
54.           distancias_vizinhos.append(dist)
55.       media_distancias_vizinhos = sum(distancias_vizinhos) /
   len(distancias_vizinhos)
56.       print(f"Média das distâncias dos vizinhos:
   {media_distancias_vizinhos:.2f}")
57.       print()
```

```
58.      mediadistanciz = mediadistanciz +
   media_distancias_vizinhos;
59.   print(f"media total")
60.   print(mediadistanciz/len(novos_dados));
```

APÊNDICE D – Código feito em Arduino com uso do FreeRTOS para ESP 32

```
#include <WiFi.h>
#include <ArduinoJson.h>
#include <PubSubClient.h>

const char* ssid = "seuwifinomedarede";
const char* password =  "suasenha";
const char* mqttServer = "10.0.0.6";
const int mqttPort = 1883;
const char* mqttUser = "homeassistant";
  const char* mqttPassword = "rohshe1as4owukeero5Cuechoo4iuxeijooThouNgee3eize4wi8Iev5nilaeQue";
  WiFiClient espClient;
  PubSubClient client(espClient);
  // Defina o tamanho do buffer de JSON conforme necessário
  const size_t bufferSize = JSON_OBJECT_SIZE(5) + JSON_ARRAY_SIZE(5) + 30;
  DynamicJsonDocument JSONbuffer(bufferSize);
  // VAR ---------------
  const int sensorPin = 4;
  const int valvulaPin = 5;
  volatile long pulse;
  int flag1segundo = 0; // Sinalizador para indicar 1 segundo
  int flagIniciandoProg = 0;
  volatile bool timerInterruptFlag = false; // Sinalizador para indicar interrupção do timer
  float totalLiters = 0;
  float volumeTotal = 0;
  float mediaflowrate = 0;
  // Definindo o identificador das tarefas
  TaskHandle_t Task1;
  TaskHandle_t Task2;
  TaskHandle_t Task3;
  // Variável global compartilhada para armazenar os dados coletados pela Task1
```

```
31.   volatile int interruptData;
32.   SemaphoreHandle_t dataMutex;
33.   // var time
34.   hw_timer_t * timer = NULL;
35.   // habilitante ele tem que ter para falar dele
36.   portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;
37.
38.   // chamado a 1000000 de micro segundos
39.   void IRAM_ATTR onTimer()
40.   {
41.    portENTER_CRITICAL_ISR(&timerMux);
42.    timerInterruptFlag = true; // Configura o sinalizador de
   interrupção do timer
43.    portEXIT_CRITICAL_ISR(&timerMux);
44.   }
45.
46.   void initTimer()
47.   {
48.    // funçao interna que inicializa o time 0 e devolve um
   ponteiro e devolve um timer de hardaware
49.    timer = timerBegin(0, 80, true);
50.    // associar hardware ao ontimer
51.    timerAttachInterrupt(timer, &onTimer, true);
52.    // configura 1 segundo
53.    timerAlarmWrite(timer, 1000000, true);
54.    // alerta de timer dipasra
55.    timerAlarmEnable(timer);
56.   }
57.   void setup() {
58.    Serial.begin(115200);
59.    // Inicialização da variável global
60.    interruptData = 0;
61.    // mqtt e wifi
62.    WiFi.begin(ssid, password);
63.    client.setServer(mqttServer, mqttPort);
64.    // RELE
65.    pinMode(valvulaPin, OUTPUT); // seta o pino como saída
```

```
66.    digitalWrite(valvulaPin, LOW); // seta o pino com nivel
   logico baixo
67.
68.    // Inicialização do mutex
69.    dataMutex = xSemaphoreCreateMutex();
70.    if (dataMutex == NULL) {
71.      Serial.println("Erro ao criar o mutex!");
72.      while (true); // Travar o programa em caso de erro
73.    }
74.    // Criação da Task1
75.    xTaskCreatePinnedToCore(
76.        recconTask,   /* Função da tarefa */
77.        "recconTask",     /* Nome da tarefa */
78.        10000,        /* Tamanho (bytes) */
79.        NULL,         /* Parâmetro da tarefa */
80.        1,            /* Prioridade da tarefa */
81.        &Task1,       /* Observa a tarefa criada */
82.        0);           /* Tarefa alocada ao núcleo 0 */
83.    // Criação da Task2
84.    xTaskCreatePinnedToCore(
85.        MqttTASK,   /* Função da tarefa */
86.        "Task2",      /* Nome da tarefa */
87.        10000,        /* Tamanho (bytes) */
88.        NULL,         /* Parâmetro da tarefa */
89.        1,            /* Prioridade da tarefa */
90.        &Task2,       /* Observa a tarefa criada */
91.        1);           /* Tarefa alocada ao núcleo 1 */
92.    // Criação da Task3
93.    xTaskCreatePinnedToCore(
94.        pulseTask,   /* Função da tarefa */
95.        "Task3",      /* Nome da tarefa */
96.        10000,        /* Tamanho (bytes) */
97.        NULL,         /* Parâmetro da tarefa */
98.        1,            /* Prioridade da tarefa */
99.        &Task3,       /* Observa a tarefa criada */
100.       1);           /* Tarefa alocada ao núcleo 2 */
101.   // Configura a interrupção do sensor
```

```
102.   attachInterrupt(digitalPinToInterrupt(sensorPin),
   sensorInterrupt, RISING);
103.   initTimer();
104.  }
105.
106.  void loop() {
107.   // Processa a lógica de sincronização fora da interrupção
   do timer
108.   if (timerInterruptFlag) {
109.     FuncAcada1seg();
110.     timerInterruptFlag = false; // Limpa o sinalizador da
   interrupção do timer
111.     client.loop();
112.   }
113.  }
114.  // Função intermediária para a interrupção
115.  void sensorInterrupt() {
116.   pulseTask(NULL);
117.  }
118.
119.  // Função da tarefa Task3
120.  void pulseTask(void *pvParameters) {
121.   pulse++;
122.  }
123.
124.  void FuncAcada1seg() {
125.   noInterrupts();
126.   if (xSemaphoreTake(dataMutex, portMAX_DELAY) == pdTRUE) {
127.     interruptData++;
128.     float flowRateLPM = (pulse / 7.5);
129.     totalLiters = flowRateLPM / 60;
130.     volumeTotal += totalLiters;
131.         if(mediaflowrate == 0 || flowRateLPM > mediaflowrate
   )
132.     {
133.         mediaflowrate = flowRateLPM;
134.     }
```

```
135.     pulse = 0;
136.     xSemaphoreGive(dataMutex);
137.     Serial.print("Dados coletados: ");
138.     Serial.println(interruptData);
139.   }
140.
141.   interrupts(); // Reabilita as interrupções
142.
143.   Serial.print(volumeTotal);
144.   Serial.println(" L ");
145.  }
146.
147.  // Função da tarefa Task2
148.  void MqttTASK(void *pvParameters) {
149.   (void)pvParameters; // Evitar aviso de parâmetro não
   utilizado
150.   for (;;) {
151.     // Espera 60 segundos antes de enviar os dados via MQTT
152.     if(interruptData > 59){
153.         // Zera a variável global de forma segura usando o
   mutex
154.         Serial.print("Media flow rate: ");
155.         Serial.println(mediaflowrate/60);
156.
157.         if (xSemaphoreTake(dataMutex, portMAX_DELAY) ==
   pdTRUE) {
158.           JsonObject JSONencoder =
   JSONbuffer.to<JsonObject>();
159.           JSONencoder["valor_vazao"] = mediaflowrate;
160.           JSONencoder["litrosmedidosemseg"] = volumeTotal;
161.
162.           if(digitalRead(valvulaPin) == LOW){
163.               JSONencoder["onoff"] = "desligar";
164.           } else {
165.               JSONencoder["onoff"] = "ligar";
166.           }
167.
```

```
168.           char JSONmessageBuffer[bufferSize];
169.           serializeJson(JSONbuffer, JSONmessageBuffer,
   sizeof(JSONmessageBuffer));
170.
171.           if (client.publish("caminho/teste",
   JSONmessageBuffer)) {
172.             Serial.println("Sucesso");
173.           } else {
174.             Serial.println("Error");
175.           }
176.           interruptData = 0;
177.           volumeTotal = 0;
178.           mediaflowrate = 0;
179.           xSemaphoreGive(dataMutex);
180.           Serial.println("Variável global zerada.");
181.         } else {
182.           Serial.println("Erro ao adquirir o mutex!");
183.         }
184.     }
185.   }
186. }
187. // mqqt task
188. void recconTask(void *pvParameters) {
189.   (void)pvParameters; // Evitar aviso de parâmetro não
   utilizado
190.
191.    for (;;) {
192.     reccon(); // Chama a função reccon() para reconexão WiFi
   e MQTT
193.     vTaskDelay(pdMS_TO_TICKS(5000)); // Espera 5 segundos
   antes de tentar novamente
194.   }
195. }
196. // pegar dados do mqtt
197. void callback(char* topic, byte* payload, unsigned int
   length) {
198.
```

```
199.   String messageTemp = "";
200.
201.   for (int i = 0; i < length; i++) {
202.     Serial.print((char)payload[i]);
203.     messageTemp += (char)payload[i];
204.   }
205.
206.   Serial.println("");
207.
208.   if (String(topic) == "esp/test") {
209.     Serial.print("Changing output to ");
210.     if(messageTemp == "ligado"){
211.       Serial.println("on");
212.       digitalWrite(valvulaPin, HIGH);
213.
214.     }
215.     else if(messageTemp == "desligado"){
216.       Serial.println("off");
217.       digitalWrite(valvulaPin, LOW);
218.
219.     }
220.   }
221.   Serial.println();
222.   Serial.println("-----------------------");
223. }
224.
225. // Função da tarefa WIFI E MQTT
226. void reccon() {
227.     // Tenta conectar ao WiFi
228.     while (WiFi.status() != WL_CONNECTED) {
229.       Serial.println("Connecting to WiFi..");
230.       vTaskDelay(pdMS_TO_TICKS(500));
231.     }
232.     Serial.println("Conectado ao WiFi");
233.     // Tenta conectar ao MQTT
234.     while (!client.connected()) {
235.       Serial.println("Conectando no MQTT...");
```

```
236.       if (client.connect("ESP32Client", mqttUser,
   mqttPassword )) {
237.         Serial.println("Conectado ao MQTT");
238.         client.subscribe("esp/test");
239.         client.setCallback(callback);
240.         if(flagIniciandoProg == 0){
241.           vTaskDelay(pdMS_TO_TICKS(5000)); // delay para nao
   bugar a valvula
242.           client.publish("caminho/inicio", "inicio");
243.          }
244.         flagIniciandoProg == 1;
245.       } else {
246.         Serial.print("Falha na conexão MQTT, estado: ");
247.         Serial.println(client.state());
248.         vTaskDelay(pdMS_TO_TICKS(500));
249.       }
250.     }
251.  }
252.
```

APÊNDICE E – Código feito em python para avaliar com KNN

```
1. from flask import Flask, jsonify
2. import paho.mqtt.publish as publish
3. import json
4. import requests
5. from datetime import datetime, timedelta
6. from sklearn.neighbors import NearestNeighbors
7. import joblib
8.
9. app = Flask(__name__)
10.
11.   def consultar_dados_fluxoagua():
12.       try:
13.           # Consulta InfluxDB para obter os últimos 30
   resultados
14.           url =
   "http://10.0.0.6:8086/query?db=fluxoagua&q=SELECT+%22litrosme
   didosemseg%22+AS+litrosmedidosemseg+" \
15.
   "FROM+%22fluxoagua%22.%22autogen%22.%22L%2Fmin%22+ORDER+BY+ti
   me+DESC+LIMIT+30;"
16.           response = requests.get(url)
17.
18.           # Verifica se a solicitação foi bem-sucedida (código
   de status 200)
19.           if response.status_code == 200:
20.               # Processa os dados da resposta e retorna um
   JSON com os resultados
21.               dados = response.json()
22.               litros_medidos = [item[1] for item in
   dados['results'][0]['series'][0]['values']]
23.               return litros_medidos
24.           else:
25.               print('Erro:', response.status_code)
26.               return None
```

```
27.      except Exception as e:
28.          print("Erro durante a solicitação:", e)
29.          return None
30.
31.  @app.route('/')
32.  def index():
33.      mqtt_host = '10.0.0.6'
34.      mqtt_port = 1883
35.      mqtt_topic = 'caminho/aviso'
36.      mqtt_username = 'homeassistant'
37.      mqtt_password =
   'rohshe1as4owukeero5Cuechoo4iuxeijooThouNgee3eize4wi8Iev5nila
   eQue'
38.
39.      dados_fluxoagua = consultar_dados_fluxoagua()
40.
41.      if dados_fluxoagua is not None:
42.          horario_atual_utc = datetime.utcnow() -
   timedelta(hours=3)
43.          horario_atual = horario_atual_utc.strftime('%H')
44.
45.          modelo_carregado =
   joblib.load('/root/meu_modelo.pkl')
46.
47.          todas_distancias = []
48.
49.          for litro in dados_fluxoagua:
50.              novos_dados = [(litro, int(horario_atual))]
51.              distancias, _ =
   modelo_carregado.kneighbors(novos_dados)
52.              todas_distancias.extend(distancias[0])
53.
54.          if todas_distancias:
55.              soma_distancias = sum(todas_distancias)
56.              media_distancias = soma_distancias /
   len(todas_distancias)
57.
```

```
58.             if media_distancias >= 3.67:
59.                 message = {
60.                     'distancia_media': media_distancias,
61.                     'aviso': 'ligar',
62.                     'avisovazamento': 1
63.                 }
64.             else:
65.                 message = {
66.                     'distancia_media': media_distancias,
67.                     'vazamento': 'sem vazamento',
68.                     'aviso': ''
69.                 }
70.
71.             payload = json.dumps(message)
72.             publish.single(mqtt_topic, payload=payload,
    hostname=mqtt_host, port=mqtt_port,
73.                            auth={'username': mqtt_username,
    'password': mqtt_password})
74.
75.             return jsonify(message)
76.         else:
77.             return jsonify({'erro': 'Nenhuma distância
    calculada'})
78.     else:
79.         message = {
80.             'distancia_media': 'sem dados',
81.             'vazamento': 'sem dados',
82.             'aviso': ''
83.         }
84.         payload = json.dumps(message)
85.         publish.single(mqtt_topic, payload=payload,
    hostname=mqtt_host, port=mqtt_port,
86.                        auth={'username': mqtt_username,
    'password': mqtt_password})
87.
88.         return jsonify(message)
89.
```

```
90.  if __name__ == '__main__':
91.      app.run(host='0.0.0.0')
92.
```

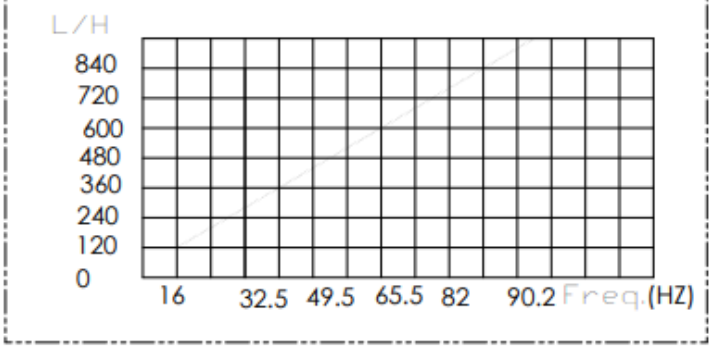
ANEXOS

ANEXO A – Datasheet sensor vazão

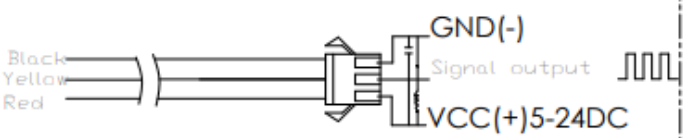
Flow-Pulse | Flow-Curve | Connection method

一.Flow Range:100L/H-/1800H-L/H

Flow (L/H)	Frezq.(HZ)	Erro range
120L/H	16	±10
240L/H	32.5	
360L/H	49.3	
480L/H	65.5	
600L/H	82	
720L/H	90.2	

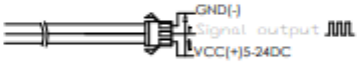


Connection method:



YIFA the plastics Ltd
Prodcut Introduction

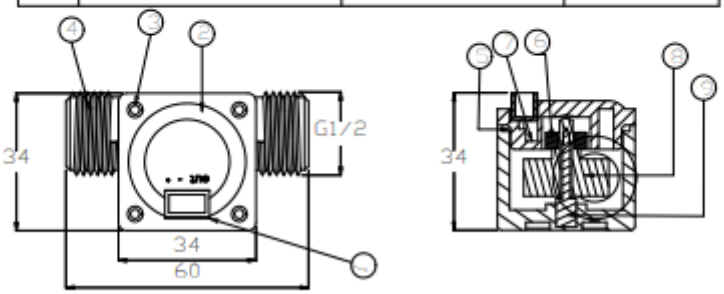
1.Modle:YF-21
2.Product Name:Hall sensor
3.Flow Range: 1-30L/MIN
4.(1)Connection Method



(2)Voltage Range 3.5-24VDC, Pluse Characteristic:F=7Q(L/MIN).
(3)Extent of error:±5%.
(4)Flow-Pulse
2L/MIN=16HZ 4L/MIN=32.5HZ 6L/MIN=49.3HZ
8L/MIN=65.5HZ 10L/MIN=82HZ

5.Bom

No.	Item	Material	Qty.
1	Connection wire		1
2	Bonnet	PA	1
3	Screw		4
4	Valve body	PA	1
5	Leak press valve		1
6	Magnet		1
7	Hall		1
8	Impeller	POM	1
9	Rustless steel axis	SUS304	1
10			
11			

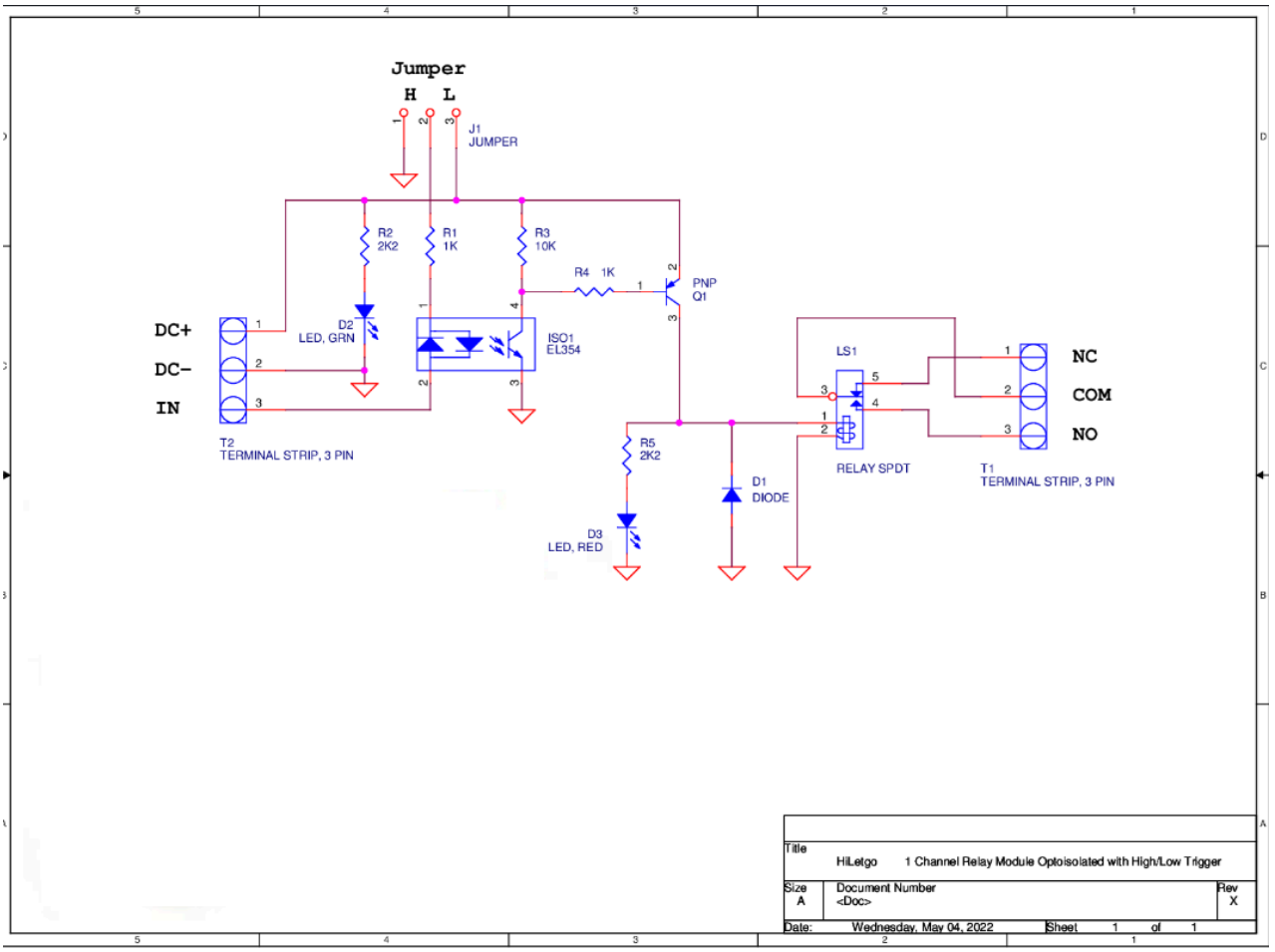


ANEXO B – Especificações valvula elétrica

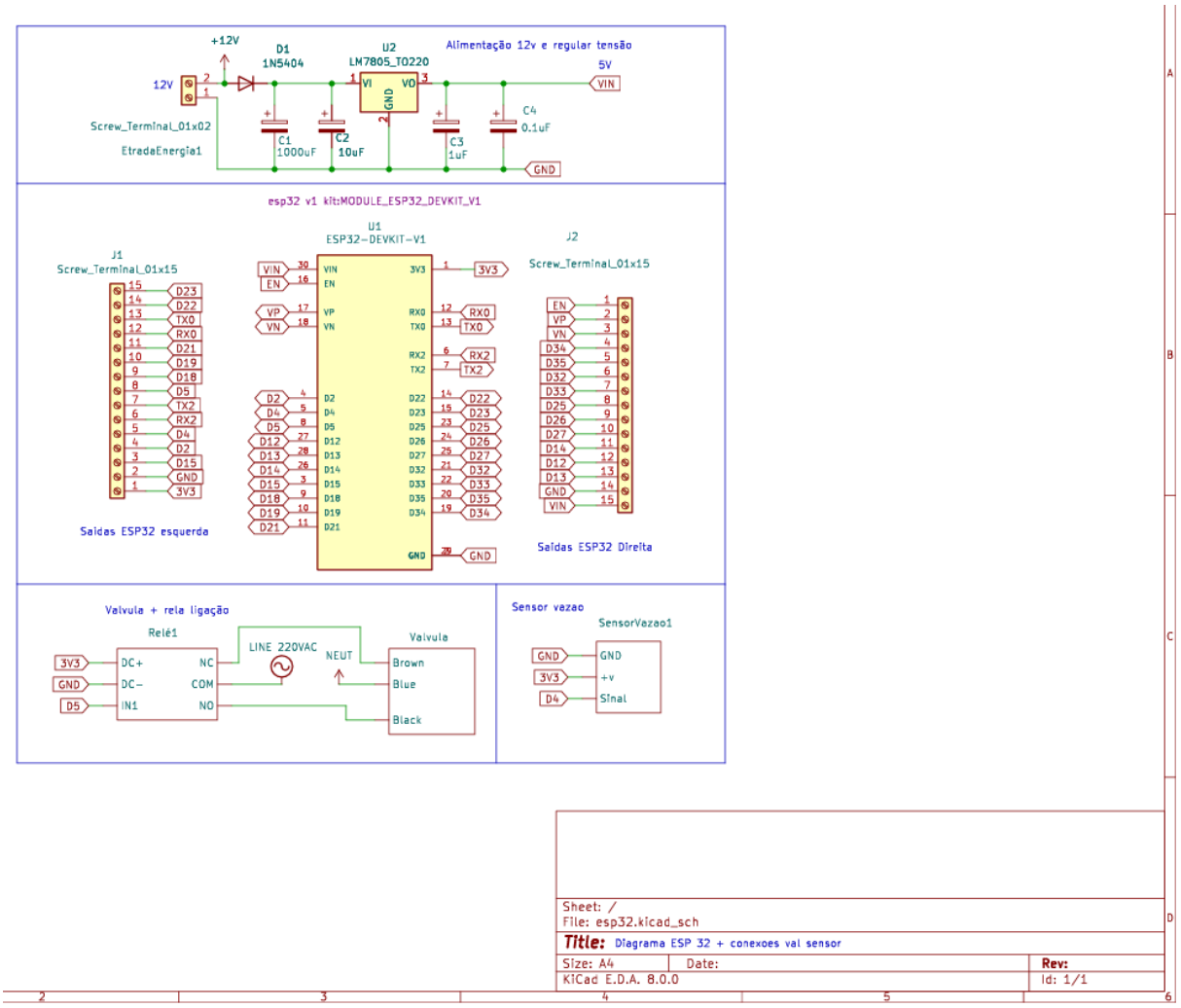
Key Specifications/ Special Features:

Warranty	1 year
Type	BALL VALVES
Customized support	OEM
Place of Origin	Guangdong, China
Brand Name	Acarps
Model Number	2W
Application	General, Water
Temperature of Media	Normal Temperature
Power	Solenoid, (Only when the valves open and close)
Media	Water
Port Size	1/2 to 2
Structure	BALL
Body material	Brass
Voltage	12VDC, 220VAC, , 24VAC, 24VDC
Connection type	BSP (G)
Working pressure	0-16bar
Operation	Three Wire Two Control
Temperature	<95°C
Running Time	5-15 Seconds
Structure Forms	Two Ways

ANEXO C – Circuito elétrico relé



ANEXO D - Diagrama elétrico



ANEXO E - Código da aplicação visual Home Assistant e bibliotecas

Codigo *Front end Home - Assistant:*
https://github.com/felipemiehe/front-Home-assistant/blob/main/front%20end%20home.txt

ArduinoJson: https://arduinojson.org/
PubSubClient: https://pubsubclient.knolleary.net/
WIFI: https://www.arduino.cc/reference/en/libraries/wifi/
Flask: https://flask.palletsprojects.com/en/3.0.x/
Joblib: https://joblib.readthedocs.io/en/stable/
Sklearn: https://scikit-learn.org/stable/
paho-mqtt: https://pypi.org/project/paho-mqtt/