Construção e Validação de um Sistema IoT de Baixo Custo para Detecção de Vazamento de Água em Residências

Luis Barbosa de Assis Jr.
Fabrício Lopes e Silva
Felipe da Rocha Henriques
luis.barbosa@aluno.cefet-rj.br
fabricio.silva@cefet-rj.br
felipe.henriques@cefet-rj.br
CEFET-RJ
Rio de Janeiro, RJ, Brasil

Raphael Pereira de Oliveira Guerra
UFF
Niterói, RJ, Brasil
rguerra@ic.uff.br

Cristiano de Souza de Carvalho
Diego Nunes Brandão
cristiano.carvalho@cefet-rj.br
diego.brandao@cefet-rj.br
CEFET-RJ
Rio de Janeiro, RJ, Brasil

ABSTRACT

The rational use of water is essential to development and economic growth. Data from United Nations (UN) indicates that until 2050 will occur a more significant water scarcity due to the increasing demand for it in emerging countries, mainly with the populational growth of these locals. A conscious consumption by society can softy this effect preserving natural reserves and increasing water security. This premise is used to support the development of this work, which proposes the creation of a low-cost computational system that detects water leakages in residences using Internet of Things concepts. The system includes data acquisition, transmission, processing, and finally displays the information to the user. Empirical assessments about the data transmitting process and a comparison of different architectures and leakage detection techniques are in development to demonstrate proposal viability. The project presents the prototype tested in a real environment.

KEYWORDS

Internet of Things, Sensors, Leakage Detection, Water

1 INTRODUÇÃO

A escassez de água no Brasil, que se agrava a cada ano, atingiu o seu pior índice histórico em 2021. A crise hídrica atual se deve à ausência de chuvas e é potencializada pelo desmatamento da Floresta Amazônica, uma vez que a supressão da floresta reduz a umidade do ar transportado da região amazônica para o centro-sul do país. Essa redução provoca uma menor precipitação de água nas regiões dos reservatórios de água para consumo humano [1].

Além da redução das fontes de captação de água, existe ainda o desperdício de água tanto nas linhas de distribuição quanto nas unidades atendidas. Um estudo recente mostra que no Brasil quase 40% da água tratada é perdida em vazamentos, fraudes e outros problemas [2]. Isto significa que a cada mil litros de água, cerca de 400 litros são perdidos. Tais números são muito altos se comparados com a média global de cerca de 15%. Assim, torna-se importante a promoção de iniciativas que auxiliem na redução do desperdício de água no país.

Em muitas cidades, o consumo de água não é mensurado em tempo real. Uma desvantagem disso é que vazamentos não aparentes acabam sendo descobertos apenas após a recepção da conta de água, que ocorre com frequência média mensal [3]. Os benefícios da detecção de vazamento no pós-medição não se limitam à economia quantitativa da água, impactam ainda os custos com a energia elétrica usada no bombeamento e processamento, bem como os custos de tratamento de esgoto [4].

Nesse contexto, inúmeras pesquisas vêm sendo desenvolvidas visando auxiliar na redução do desperdício de água. Boa parte destas pesquisas abordam o problema dos vazamentos em redes de distribuição de água (*Water Distribution Network* - WDN). No entanto, pesquisas relativas à detecção de vazamento diretamente nos consumidores — ambientes residenciais, escolas, universidades, prédios comerciais e etc. — estão se tornando cada vez mais frequentes.

Algumas propostas de detecção de vazamentos nos consumidores visam a instalação de hidrômetros inteligentes (*smart meters*) para medir o consumo e enviar os dados via rede de telefonia móvel para o consumidor ou para a concessionária de água [5]. Um artigo interessante neste tópico é o de Britton et al. (2013), no qual os autores reforçam a importância do desenvolvimento de sistemas de monitoramento para prevenção a vazamentos e apresentam um relato real da substituição de hidrômetros antigos por hidrômetros inteligentes. Os novos equipamentos avaliavam se houve fluxo ininterrupto nas últimas 48 horas e registravam por telemetria o consumo a cada hora, enviando os dados para dispositivos da concessionária de água. A pesquisa consistiu na detecção de vazamentos no pós-medição dos hidrômetros e na avaliação de formas de notificação de vazamento aos moradores. Após três meses de intervenção, os autores obtiveram uma redução no consumo de cerca de 87,68%.

Outras propostas envolvem a utilização de dispositivos de baixo custo para monitorar o consumo realizado pelo cliente por meio de sensores instalados na rede interna de água [3, 6]. Essas pesquisas utilizam plataformas de prototipagem eletrônica, como Arduino, Raspberry, NodeMCU, dentre outras para o desenvolvimento de sistemas que monitoram a vazão de água no sistema de tubulação de água das residências. Em seu trabalho, Romeiro (2019) desenvolveu um sistema que visava determinar os hábitos de consumo dos moradores de uma residência a partir de um sensor de vazão baseado em efeito Hall, empregando técnicas de regressão polinomial. Já Fuentes and Mauricio (2020) desenvolveram um sistema que identificava vazamentos a partir dos dados de vazão, utilizando para isso a técnica dos k-vizinhos mais próximos (KNN). A partir dessa identificação automática, o sistema enviava uma mensagem ao usuário da residência, solicitando que ele confirmasse a presença do vazamento. Os autores conseguiram atingir uma acurácia de 74% na detecção de vazamentos.

No presente trabalho optou-se por avaliar o consumo da água no pós-medição. Para tanto, um sistema de monitoramento de vazamento de água foi construído e validado. O sistema desenvolvido é baseado em uma arquitetura em névoa/borda [7]. Ele implementa uma regra de associação conhecida como Consumo Zero nas Últimas *x* Horas (do inglês, *Non-Zero Water Consumption* - CNZ) [3, 4]. Busca-se também acrescentar novas perspectivas sobre o consumo residencial de água a partir de dados de relatórios brasileiros. Para tanto um experimento em ambiente residencial foi utilizado. Neste ambiente foram instalados sensores de pressão e vazão conectados a uma placa NodeMCU, sendo esta placa responsável tanto por adquirir os dados, quanto por armazená-los e transmiti-los a um servidor que realiza o processamento dos dados e a exibição para o usuário. O sistema desenvolvido mostra a viabilidade da detecção de vazamentos, além de permitir a identificação de limitações nos sensores de baixo custo utilizados.

Além desta introdução, este trabalho encontra-se dividido nas seguinte seções: na Seção 2 são apresentados alguns conceitos referentes aos componentes arquiteturais do sistema; a Seção 3, que apresenta as técnicas, ferramentas e conceitos que foram explorados na resolução do problema assim como as justificativas baseadas em estudo de outros trabalhos para estas escolhas. A Seção 4 apresenta os resultados obtidos a partir do cenário experimental; e por fim, as considerações finais são apresentadas na Seção 5.

2 CONCEITOS FUNDAMENTAIS

Esta seção descreve alguns conceitos de Internet das Coisas (*Internet of Things* - IoT) necessários para a compreensão do sistema desenvolvido. Mais detalhes sobre o assunto, assim como uma revisão bibliográfica podem ser obtidos em [3, 4].

2.1 Internet das Coisas (IoT)

O termo IoT se refere à interconexão em rede de objetos do cotidiano geralmente equipados com algum tipo de inteligência [8]. O advento das plataformas de prototipagem de baixo custo permitiu que a IoT se torne cada vez mais popular com aplicações em domótica, agricultura, indústria e muitas outras. Dentre tais plataformas, o Arduino é uma das mais famosas, permitindo o desenvolvimento de soluções de forma simples e com baixo custo econômico [9]. Apesar das inúmeras vantagens, o Arduino necessita que sejam incorporadas novas placas (*shields*) para que seja possível novas funções, como exemplo a comunicação sem fio, gerando um custo maior ao protótipo que deseja-se desenvolver. Assim, um dispositivo similar com um custo menor, foi escolhido para este projeto. O NodeMCU também é uma plataforma de prototipagem que já possui módulos de comunicação, como módulo WiFi e Bluetooth, embarcados, garantindo um menor custo ao projeto a ser desenvolvido. Além disso, ele possui compatibilidade com o ambiente de desenvolvimento do Arduino e mais portas compatíveis com sinais analógicos, facilitando a adição de mais sensores em uma mesma placa [3, 10]. A Tabela 1 apresenta uma comparação sucinta entre os dispositivos.

2.2 Sensores

Os sensores agregam funcionalidades a essas plataformas de prototipagem. Eles são dispositivos que respondem a estímulos físicos ou químicos, transformando a ação sofrida em um sinal analógico ou digital que pode ser transformados em grandeza física para fins de medição ou monitoramento [11]. Exemplos de sensores incluem medidores de temperatura, quantidade de CO_2, luminosidade, velocidade, pressão e etc.

Tabela 1: Comparativo Arduino x NodeMCU.

Característica	Arduino Uno	NodeMCU ESP32
Microcontrolador	ATmega328 Single Core 16MHz	LX6 Dual Core 240MHz
Conexão	USB	USB, Bluetooth, Bluetooth Low Energy, WiFi
Portas	20 GPIO sendo: 6 Analógicas 6 PWM	25 GPIO sendo: 18 Analógicas/digitais 16 PWM
Memória	SRAM: 2KB EEPROM: 1KB	RAM: 520 KBytes ROM: 448 KBytes Flash: 4 MB
BRL R$	62,85	50,00

Quanto mais preciso um sensor maior seu custo, o que inviabiliza sua utilização em certos tipos de projetos. Todavia, os sensores de baixo custo possuem algumas limitações, inclusive de precisão e sensibilidade. É possível medir a vazão por meio de sensores que utilizam ultrassom. Estes são usados quando não é possível furar a tubulação. Geralmente empregados em grandes adutoras, eles têm um custo ainda alto. Por exemplo, o modelo ECR 100F para tubulações de 50-700 mm de diâmetro possui precisão 0,1%, sensibilidade a partir de 0 L/hora e valor de BRL R$ 3.450,00.

No caso de abordagens destrutivas, há os sensores de efeito Hall para situações onde é possível furar a tubulação. Estes possuem internamente seis pás em formato de cata-vento que são movidas com o fluxo de água passante. Esse movimento gira o ímã acoplado. A rotação do campo magnético aciona o sensor Hall, que então emite ondas quadradas (pulsos) [12]. Esses pulsos elétricos são contabilizados pelo NodeMCU que os transforma em valores mensuráveis.

Outra forma de detecção de vazamento se dá pelo estudo da pressão na tubulação. Geralmente usado em WDN. Este sensor, em contato direto com o líquido, sofre ação da pressão da água. De acordo coma força sofrida gera um valor de tensão que será interpretado e convertido para pascal (Pa) pelo NodeMCU. O sinal gerado é analógico. Na Figura 1 é demonstrado a estrutura interna dos sensores, seu funcionamento e o formato de saída dos dados. Uma vez convertidos em valores mensuráveis podem ser mostrados de forma gráfica para o usuário.

2.3 Redes de Sensores Sem Fio

Os dados coletados pelos nós sensores são transmitidos para um nó sorvedouro, que é o nó conectado a uma rede estruturada, por exemplo, a Internet. Cada sensor tem limitação de banda de transmissão, energia, processamento e armazenamento. Nas chamadas Redes de Sensores Sem Fio (RSSFs), inúmeros nós sensores coletam dados que são transmitidos para um ou mais nós sorvedouros

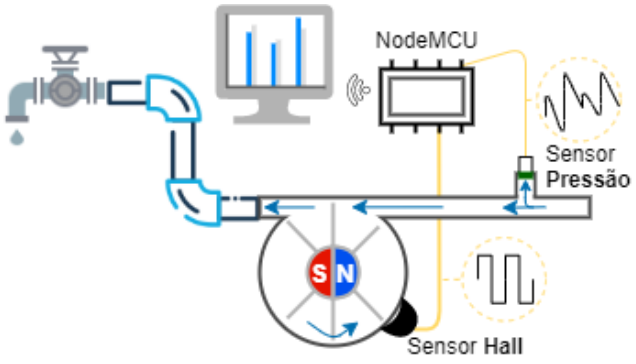

Figura 1: Estrutura interna e funcionamento dos sensores de pressão e vazão, tipo Hall, e formato da saída de dados. Fonte: Próprio autor.

que repassam estes dados ao usuário final ou algum ambiente de armazenamento.

As RSSFs são úteis em diversas aplicações, tais como monitoramento do meio ambiente, monitoramento de consumo (de água, de energia, dentre outros), monitoramento de idosos ou pacientes em hospitais, segurança, enxame de drones, robôs cooperativos, etc. Elas são essenciais para o desenvolvimento de cidades inteligentes e agricultura de precisão [13], formando uma infraestrutura de comunicação para tais aplicações.

Tais redes podem utilizar diferentes tecnologias de comunicação. Especificamente para o caso de RSSFs dentro do contexto de IoT, as redes mais utilizadas são LPWAN (*Low Power Wide Area Network*), WLAN (*Wireless Local Area Network*) e WPAN (*Wireless Personal Area Network*), NFC (*Near-Field Communication*). As redes LPWAN focam na cobertura de grandes áreas com baixa taxa de transmissão; um exemplo é a tecnologia LoRa. As redes WLAN e WPAN são exemplificadas pela tecnologia Zigbee (baseada no protocolo IEEE 802.15.4), a qual permite a comunicação de diversos sensores e atuadores em redes pessoais, mas com a flexibilidade de implementar diferentes tipos de topologia de rede. Um outro exemplo de tecnologia de rede WLAN é o WIFI (IEEE 802.11), sendo a tecnologia mais utilizada, presente em smartphones, notebooks, tablets e etc. A tecnologia NFC é a menos utilizada, pois garante a comunicação em espaços muito curtos, cerca de poucos centímetros.

Além de compreender sobre as tecnologias de comunicação em redes sem fio, a compreensão dos tipos de topologia de redes é de suma importância para auxiliar na escolha de qual modelo utilizar em sua aplicação. Neste contexto, as principais topologias são: (a) centralizada, onde cada nó sensor se conecta diretamente ao nó sorvedouro; (b) descentralizada, onde os nós sensores transmitem seus dados ao nó sorvedouro por meio de outros nós sensores. Os protocolos Bluetooth e Wi-Fi são muito comuns para conexão em uma rede centralizada. O protocolo Zigbee é o mais popular para compor RSSFs descentralizadas.

Este trabalho adota uma rede estruturada com conexão Wi-Fi. Os nós sensores (NodeMCUs) e o nó sorvedouro (Raspberry Pi) se comunicam por meio de um roteador Wi-Fi que também oferece comunicação cabeada com a Internet.

2.4 *Middleware* de comunicação

Manter a comunicação com as tecnologias mencionadas, além de aspectos de segurança e outras funcionalidades em uma RSSF é uma tarefa árdua, pois seria necessário haver interações diretamente no *hardware*. Para mitigar a necessidade dessas interações e facilitar o desenvolvimento de sistemas, surge o conceito de *Middleware*. Um *middleware* consiste em um *software* que implementa diversas funcionalidades de maneira transparente ao usuário. No caso de RSSF, um *middleware* fornece um serviço de troca de mensagens que garante além da entrega das mensagens entre os elementos da rede, características como a tolerância à falhas, a capacidade de aumento no volume de dados trafegando, a transparência de localização, dentre outras [14].

Segundo Van Steen and Tanenbaum (2017), *Middlewares* Orientados a Mensagem (MOMs) garantem uma comunicação persistente assíncrona entre os elementos do sistema, sem que estes elementos (emissor e receptor) estejam necessariamente ativos durante a transmissão das mensagens. Um MOM implementa uma abstração de filas de mensagens que podem ser acessadas via rede. Assim, ele garante que os nós da rede não precisam necessariamente estar o tempo todo conectados a ela, garantindo com isso economia de recursos como bateria. Além disso, os nós da rede não precisam conhecer os endereços uns dos outros, garantindo também economia de memória. Todavia, o MOM necessita de um elemento central que é responsável por gerenciar as filas de mensagens, tal limitação pode ser superada com a replicação de tal elemento.

Para garantir um melhor gerenciamento das filas de mensagens e do acesso as essas mensagens diferentes políticas podem ser implementadas. Na mais utilizada, *publish-subscriber*, as mensagens são associadas a um determinado tópico, sendo lidas somente por assinantes que optaram por receber mensagens sobre aquele referido tópico. Nesse tipo de política um terceiro elemento é necessário, o *broker*. Ele é um servidor responsável por armazenar as mensagens enviadas, além de filtrá-las, associando cada mensagem a um tópico e garantindo que somente os assinantes daquele tópico tenham acesso a ele.

Neste contexto de *publish-subscriber*, surge o protocolo de comunicação MQTT (*Message Queuing Telemetry Transport*). O MQTT funciona com dois tipos de atores: clientes e o *broker*. Os clientes que enviam mensagens são os já mencionados *publishers* e os que recebem são os *subscribers*. Os tópicos são como se fossem canais onde as mensagens são publicadas. É importante salientar que um cliente pode ser, simultaneamente, *publisher* e *subscriber*, além de ser permitido estar inscrito e um ou mais tópicos [16]. A Figura 2 sintetiza a comunicação entre clientes e *broker*.

O MQTT foi desenvolvido na década de 1990 pela IBM para gerenciar a transmissão de dados assíncronos através de redes intermitentes [17]. Suas principais vantagens são: clientes MQTT são muito pequenos, requerendo poucos espaço na memória dos microcontroladores, seus cabeçalhos são pequenos e otimizados para redes com pouca largura de banda, suporta protocolos de segurança e é escalável permitindo adição de novos nós com facilidade. Tanto Fuentes and Mauricio (2020) quanto Silva Júnior (2017) implementaram este protocolo em seus trabalhos.

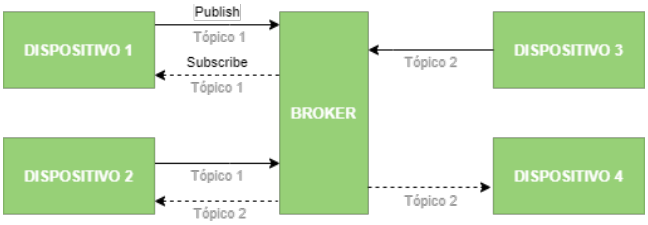

Figura 2: Diagrama de fluxo da comunicação cliente-*broker* no MQTT.

Fonte: Próprio autor.

2.5 Arquiteturas de Processamento

Os dados coletados pelos sensores precisam ser processados para serem transformados em informação. De acordo com o local de processamento pode-se classificar a arquitetura do sistema desenvolvido de três maneiras: em borda (*Edge Computing*) [18–20], em névoa (*Fog Computing*) [20, 21] ou em nuvem (*Cloud Computing*) [22]. A mais conhecida, computação em nuvem, é aquela onde os dados captados são transmitidos para serviços em servidores de Internet, local onde são processados. Na opção de computação em névoa, os dados são processados em dispositivos de borda de rede como roteadores, *gateways* e computadores ligados a essa rede. Neste caso, o objetivo é trazer o processamento para arquitetura nativa ganhando desempenho ao reduzir a latência no envio dos dados, enviando-os para a nuvem central apenas para cálculos mais custosos [23]. Assim, a computação em névoa consiste em adicionar poder computacional próximo aos geradores de dados brutos [24]. Por fim, a arquitetura mais próxima aos nós é a computação de borda, onde é explorado parte do processamento nos próprios nós sensores [23].

Amaxilatis et al. (2020) citam que o processamento em nuvem tem como grande vantagem o maior poder computacional, homogeneidade dos dispositivos e acesso de qualquer local do globo, porém costumam ser serviços pagos, são centralizados para reduzir custos, passivos de indesejados atrasos e congestionamentos nas redes. Uma forma de reduzir o tráfego de dados para nuvem e ter redução no tempo de resposta é explorar poder computacional de dispositivos espalhados pela rede interna do usuário. Concluíram em seu trabalho que a escolha por computação em névoa reduziu significativamente a quantidade de dados a serem enviados para nuvem restringindo o envio a apenas 5% do tráfego gerado na rede interna além de prover melhores recursos gerais de computação, segurança e privacidade para o sistema. Os autores também citam que a escolha por essa solução de arquitetura de processamento também gera ganhos no tempo de resposta mais rápido já que as requisições não precisam ir para um sistema de processamento distante sendo processado em dispositivos em névoa tais como: servidores, notebooks, roteadores de borda e *smartphones*. Computação em névoa provê recursos computacionais, armazenamento e controle aos consumidores em uma camada intermediária entre os sensores IoT e os tradicionais *data centers*.

Com o avanço da tecnologia, redução de custo de fabricação e tamanho físico dos dispositivos, uma possibilidade de distribuição de processamento tornou-se cada vez mais atraente, rápida e econômica: a computação na borda, onde parte ou todo o processamento é feito nos dispositivos onde os sensores estão fisicamente conectados como, por exemplo, Arduino, NodeMCU e PIC. Por estes motivos, a computação em névoa e na borda foram selecionadas para an implementação deste trabalho. Após comparações realizadas entre algumas plataformas de prototipagem foi possível escolher o NodeMCU ESP32, que além de possuir mais memória que o Arduino Uno, também possui dois núcleos para processamento, conexão Wifi e Bluetooth embarcados reduzindo o custo de implementação.

3 METODOLOGIA

Na parte de sensoriamento foram selecionados os sensores de vazão e pressão. O sensor de vazão é baseado no efeito Hall. Na Tabela 2, é possível ver as especificações de tais sensores. São computados o consumo e a pressão da residência para detectar vazamentos. Para programar os nós sensores foi escolhida a plataforma Arduino IDE 1.8.13 cuja linguagem de programação baseia-se em *C* + + além de possuir vasto material disponível de forma gratuita em livros, internet e cursos.

Tabela 2: Especificações dos sensores.

Sensor	
Vazão **Seeed YF-S201**	**Pressão**
Funcionamento: DC 4,5 V a 18 V Tensão de trabalho: DC 4,5 V Corrente Máx.: 15 mA (DC 5 V) Vazão de água: 1 a 30 L/min Temperatura de operação: 80 °C Pressão da água: ≤ 2 MPa Diâmetro entrada/saída: 1/2"	Conexão: Rosca 1/4" BSP Tensão de trabalho: DC 5 V Pressão de trab.: 0 a 1,2 MPa Máx. pressão: 2,4 MPa Temperatura: 0 a 85 °C
BRL R$ 53,10	BRL R$ 143,89

A plataforma IoT *ThingsBoard* foi escolhida dentre outras por permitir a instalação no Raspberry Pi, possuir documentação aberta, versão gratuita permitindo a computação em névoa, objetivo deste trabalho. Esta plataforma é responsável por receber as telemetrias dos nós sensores, processá-las e mostrá-las visualmente por meio de gráficos em *dashboards* [25]. O sistema de gerenciamento de banco de dados (SGBD) utilizado é PostgreSQL 11. Além disso, o Raspberry Pi 3B foi escolhido para rodar a plataforma de computação *ThingsBoard*.

A escolha pelo Rapsberry Pi foi reforçada pelo trabalho de Amaxilatis et al. (2020) que também optou por computação em névoa onde foram executados testes de desempenho nos nós de processamento usando um Raspberry Pi, um notebook equipado com processador Intel Core i3-3120M e um servidor Intel E5-2630V4 com diferentes configurações de RAM e armazenamento SD ou SSD para comparação. Consideraram uma limitação do uso de apenas 1 núcleo do processador. Em suas conclusões indicam o Raspberry Pi como solução de baixo consumo e processamento aceitável para projeto de até 30 medidores inteligentes atingindo latência de processamento de 1,62 milissegundos, e conseguindo lidar com operações de criptografia sem oferecer nenhum atraso significativo na comunicação.

No aspecto transmissão de dados a tecnologia utilizada foi a rede sem-fio padrão IEEE 802.11b/g/n por ser embarcado no NodeMCU, logo, sem acréscimo de custo. A topologia de rede utilizada foi a estrela. O protocolo escolhido para comunicação com usuário foi o HTTP, porém o mesmo não foi selecionado para comunicação entre os sensores e o *ThingsBoard*. Essa função ficou a cargo do MQTT. Ambos os protocolos atuam na camada de aplicação e suas escolhas foram feitas baseando-se, também, no estudo de Wukkadada et al. (2018) onde verificou-se que para estabelecer uma comunicação HTTP são necessários nove pacotes a cada envio de mensagem, enquanto que no MQTT são apenas dois pacotes são necessários, o que dificulta as várias transmissões de telemetria devido a possibilidade de sobrecarga da rede caso a primeira opção fosse escolhida. As mensagens MQTT possuem tamanho de 2 bytes, o consumo de energia no envio de mensagens MQTT são sempre muito menores que o consumo no envio de mensagens feito em HTTP contribuindo também para uma duração maior da bateria em casos onde não há alimentação de rede e por fim a recepção de mensagens MQTT possui ínfima taxa de perda de pacotes.

4 CENÁRIO EXPERIMENTAL E RESULTADOS

Como pode ser observado na Figura 3, para a arquitetura utilizada neste estudo foram previstas diferentes combinações de nós sensores. Os nós na borda da rede são constituídos por dispositivos NodeMCU contendo sensores de pressão e vazão. Estes dispositivos enviam as informações adquiridas através dos sensores diretamente ao roteador, que por sua vez redireciona a informação para o servidor Raspberry contendo o Thingsboard. Após o processamento no Thingsboard, a informação é redirecionada ao smartphone do usuário final.

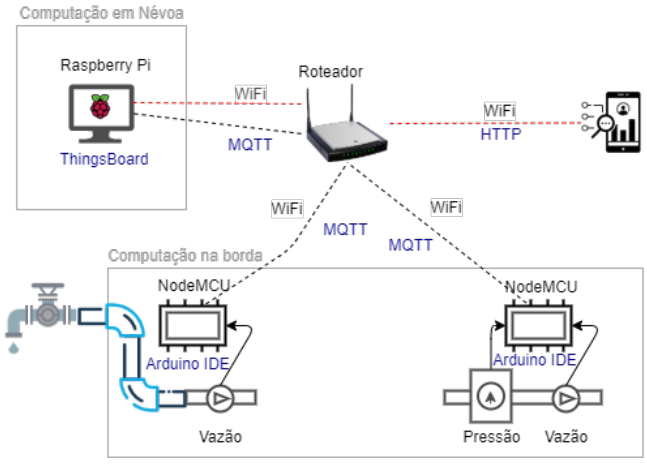

Figura 3: Arquitetura implementada.
Fonte: Próprio autor.

4.1 Dados Coletados

Para o desenvolvimento do projeto foram utilizados dados de vazão e pressão, basicamente. Com o sensor de vazão espera-se medir o volume de água consumido no último minuto. Este dado obtido do sensor corresponde à *flag* $flow\_min$ na telemetria. Para esta variável optou-se por enviar o valor médio da vazão em um intervalo de 1 minuto, considerando que esta vazão pode variar entre zero e seu valor máximo ao se abrir uma torneira no dispositivo consumidor. Para determinar o valor enviado são realizadas cerca de 60 aquisições em um minuto e calculado o valor médio.

A pressão de água na tubulação é medida em KPa (kiloPascal) e é transmitida sob a *flag* $Pressure$. Além dos dados de vazão média e pressão, o dispositivo transmite ainda sua identificação (*flag* $token$) e as informações referentes ao instante da recepção dos dados (*flag* $date\_time$). A *flag* $token$ corresponde a uma chave primária contendo um conjunto de caracteres que compõem uma palavra composta por letras e números, enquanto a *flag* $date\_time$ contém a data e a hora em que os dados foram recebidos.

A utilização de sensor pode ser feita para um só dispositivo consumidor como torneira, chuveiro, filtro e etc, nesta aplicação o método de instalação é considerado não-destrutivo, pois é possível instalar o sensor entre a saída de água e o dispositivo consumidor. Quando o sensor é instalado para um cômodo, é necessário fazer um pequena obra para instalação na entrada de água do cômodo e por isso é considerada como destrutiva. O Thingsboard permite o uso de *tags* para identificar o local de instalação do sensor, não é um dado transmitido na telemetria, mas sim configurado na plataforma IoT.

4.2 Precisão do Sensor de Vazão

Para avaliar a qualidade do sensor de vazão foram realizadas repetidas medições de 500 ml com o auxílio de um recipiente graduado. A partir das medições foi possível calcular um erro de 6,79% e desvio padrão de 69,44 mL. Fuentes and Mauricio (2020), em experimento similar, obtiveram 4,63% de erro nas medições e para isso precisaram calibrar o sensor. A precisão descrita no *datasheet* do fabricante é de 5%.

Neste experimento o sensor não foi calibrado, permanecendo com a constante do algoritmo idêntica à fornecida pelo fabricante. Durante as medições o sensor precisou passar por um processo de manutenção para limpeza do eixo, pois o mesmo parou de girar. Uma substância, semelhante a um tipo de óleo, liberada pela mangueira utilizada nas medições iniciais e as partículas sólidas na água oriundas da WDN podem ter impactado nas leituras.

O sensor possui uma seta em seu verso indicando a orientação do fluxo de água e sua montagem foi feita sempre na horizontal seguindo as orientações de Romeiro (2019). Por recomendação do fabricante foi utilizado um resistor *pull-up* no pino de entrada de dados para estabilização do sinal. Anteriormente, a saída do sensor era ligada a uma mangueira cuja ponta possuía um esguicho. Notou-se que mesmo após o fechamento do esguicho havia fluxo de água devido ao aumento de pressão nas mangueiras do experimento causando erro nas leituras. Este fato motivou substituição desta configuração por um registro imediatamente após o sensor de vazão, como pode ser observado na Figura 4.

Outro fato avaliado neste momento foi a sensibilidade do sensor de vazão que se mostrou insuficiente para detectar pequenos vazamentos como o encher de um copo de medidas contendo 500 mL ao longo de um minuto e treze segundos, resultando numa vazão de 0,41 L/min obtidos pela Equação 1. Nas especificações do fabricante, o mínimo detectável é 1 L/min (Tabela 2). Foi possível verificar na

Figura 4: Experimento para avaliação do sensor de vazão. Fonte: Próprio autor.

prática essa limitação por meio da leitura de telemetrias onde a menor vazão detectada durante o regime contínuo foi 1,1 l/min.

$$vazao = \frac{quantidade(mL)}{tempo(seg)} \times \frac{60}{1000} = L/min \quad (1)$$

Britton et al. (2013) também identificaram dificuldade no registro do consumo de pequena vazão ao utilizar o hidrômetro inteligente Elster V100 que possui 2% de precisão e uma limitação de leitura mínima de 3 L/h. O aparelho usado tem um custo atual de USD $66,00, ou, em conversão direta, BRL R$369,60. Apesar de o equipamento usado por Britton et al. (2013) ser mais preciso que o escolhido neste trabalho, seu custo também se mostrou fora desta proposta e por esse motivo não foi avaliado.

4.3 Precisão do Sensor de Pressão

O experimento consistiu na avaliação da precisão do sensor de pressão. O sensor funcionou com alimentação de 3,3 V e não precisou de resistor *pull-up*. O primeiro passo para sua utilização foi medir a tensão equivalente a pressão zero. Utilizando o próprio código que faz a medição da pressão, o valor obtido passou a ser o zero referencial. A tensão encontrada foi 0,192 V para pressão atmosférica ambiente.

O passo seguinte foi realizar medições *inloco* em andares diferentes do prédio residencial utilizado para os experimentos. Tal edifício é constituído por uma casa de máquinas, seguido de seis andares habitacionais e três andares de uso coletivo (incluindo o térreo). Para as medições, foram selecionados dois pontos de água, em diferentes colunas de distribuição, porém com mesmo diâmetro. Foram desconsideradas a espessura da laje entre andares (aproximadamente 11 cm) e o nível da água na caixa d'água (aproximadamente 1,4 m). A mangueira foi preenchida com água e teve sua pressão liberada para se fazer a medição inicial em cada ponto. Para validação teórica foi adotado o valor de 10 metros de coluna de água para o valor de 98,04 KPa de pressão, bem como a altura da casa de máquina (H_m) de 2,1 m, a altura dos andares (H_a) de 2,75 m, a altura do ponto de medição até o teto (H_p) do 2º andar de 1,6 m e a altura do andar térreo de 1,95 m, conforme a Equação (2). Na Tabela 3 são apresentados os resultados, para os quais se constatou erro de precisão próximo a 5%.

$$P = \frac{H_m + Andares \cdot H_a + H_p}{10} \cdot 98,04 \quad (2)$$

Tabela 3: Valores prático e teórico da pressão nas torneiras escolhidas.

2º Andar			
0,200 V	3,1 KPa	Teórico	Erro
0,606 V	165,5 KPa	171,11 KPa	3,2%
Térreo			
0,206 V	5,7 KPa	Teórico	Erro
0,865 V	268,7 KPa	282,41 KPa	4,8%

4.4 ThingsBoard

O ThingsBoard versão 3.2.2 foi instalado no Raspberry Pi 3B rodando o sistema operacional Linux Raspibian OS versão 3.6 32-bits. ThingsBoard foi configurado para utilizar até 256 MB de memória RAM. O banco de dados (SGBD) usado foi o PostgreSQL 11 por ser a versão mais atual compatível com processadores de arquitetura ARM. Foi necessária a instalação das bibliotecas no Linux toncat9 e mqtt. Essas bibliotecas precisam ser paradas antes da inicialização do Thingsboard, pois o mesmo possui esses serviços internamente.

Foi realizado experimento de recepção de telemetria simultânea a partir de dois nós sensores pelo ThingsBoard com sucesso. Um nó com sensor de pressão e vazão e outro apenas com sensor de vazão. O teste consistiu em assoprar os sensores de vazão sem se preocupar com aferir os valores lidos.

4.5 Funcionamento em Ambiente Real

Nesta etapa foi avaliado o tempo de vida de uma bateria 10.400 mA·h ligada ao NodeMCU via USB enviando telemetria de vazão e pressão. Buscando uma utilização em ambiente real, foram instalados ambos os sensores numa torneira de tanque em apartamento de classe média da cidade do Rio de Janeiro conforme a Figura 5. O término do experimento foi devido ao esgotamento do *power bank* totalizando 3 dias e 7 horas de experimento onde foram enviadas um pouco mais de 4.740 telemetrias, uma a cada minuto. A conexão Wifi ficou sempre ativa e as mensagens foram enviadas usando o protocolo MQTT.

O ThingsBoard permite analisar gráficos em tempo real ou por período, a Figura 6 apresenta toda a telemetria de consumo somado em intervalo de 15 minutos no período mencionado em verde. Observa-se aqui diminutos momentos com leituras de consumo. Por este motivo o algoritmo foi alterado para enviar telemetria apenas quando a vazão é diferente de zero afim de economizar bateria. Esse recurso também economiza espaço de armazenamento no banco de dados do ThingsBoard. Verificou-se que ao adicionar os códigos que ativam o funcionamento do Wi-Fi, o pino 2 do NodeMCU passa a trabalhar de forma anômala. Isso se deve ao fato deste pino ser compartilhado com o chip Wi-Fi e por este motivo foi necessário mudança de porta do sensor de vazão, porém o erro só foi descoberto após análise dos valores apresentados na Figura 6. Nela é possível

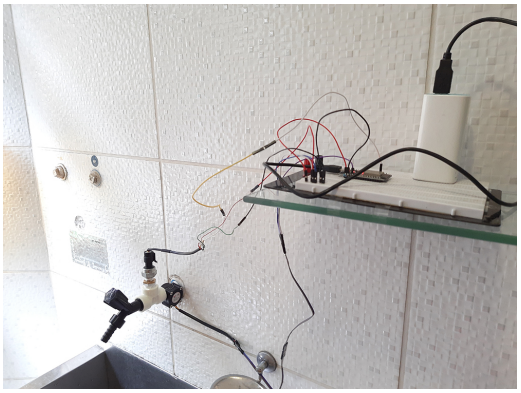

Figura 5: Teste de telemetria em ambiente real.
Fonte: Próprio autor.

observar um consumo médio inferior ao limite de sensibilidade do sensor que é de 1.000 mL/minuto.

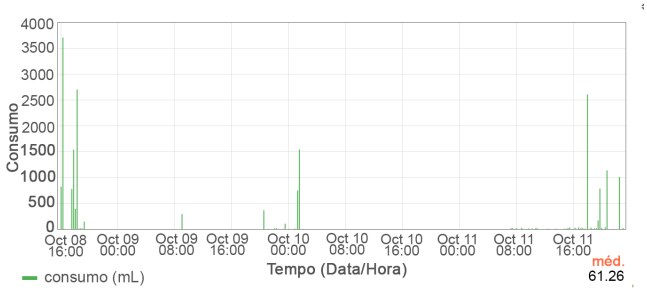

Figura 6: Dados de consumo na telemetria em ambiente real.
Fonte: Próprio autor.

Outro comportamento esperado verificado na prática é a queda da pressão durante o consumo. Neste experimento foram feitas medições a cada 10 segundos. A pressão média sem consumo ficou em 171,46 kPa enquanto durante o consumo chegou a 165,87 kPa variando -3,37% sendo verificado na Figura 7 e melhor visualizada na Figura 8. Todos os gráficos desta subseção foram extraídos por meio de *dashboards* no ThingsBoards.

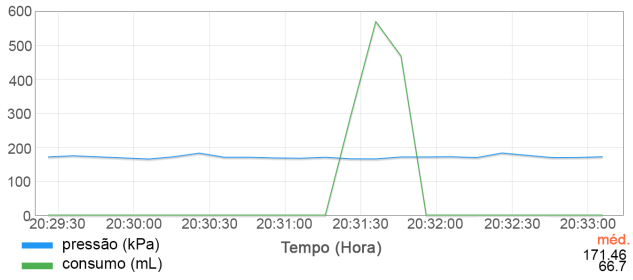

Figura 7: Consumo x Pressão.
Fonte: Próprio autor.

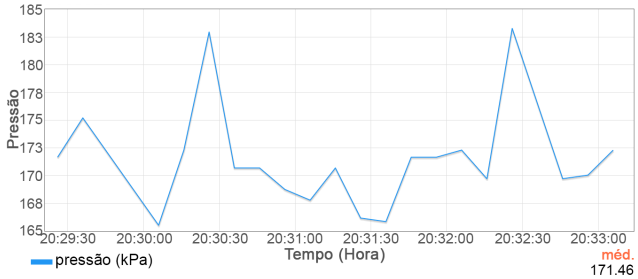

Figura 8: Pressão durante o consumo de água da torneira.
Fonte: Próprio autor.

4.6 Detecção de Vazamento

Foi implementado, em computação na borda (*edge computing*), o algoritmo de detecção de vazamento CNZ. Sua validação se deu por meio de experimentos realizados no aparato apresentado na Figura 5 e com tempo alterado para dois minutos de vazão contínua, passado este tempo o NodeMCU do sensor indica vazamento. No primeiro minuto de vazão zero o alerta de vazamento é desfeito. Também foi implementado o sinal de *alive*, criado com intuito de indicar que o sensor encontra-se conectado a rede e é enviado quando a última telemetria excede o tempo escolhido de uma hora, porém a título de validação da regra de associação foi configurado para ser enviado após dois minutos sem envio de telemetria. A telemetria a seguir demonstra os comportamentos citados.

```
20:06:47.299 -> WiFi connected
20:06:47.352 -> Conectando ao broker MQTT...
20:06:47.453 -> Conectado ao broker!
20:07:46.764 -> Leituras pressão/vazão updated
20:08:47.035 -> Leituras pressão/vazão updated
20:08:47.182 -> Vazamento detectado! Nas últimas 24h
você não teve consumo zero em nenhum momento.
20:09:47.449 -> Leituras pressão/vazão updated
20:10:47.558 -> Vazamento consertado!
20:11:47.636 -> alive = true enviado!
```

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou a construção e validação de um protótipo de um sistema de detecção de vazamento em residências em tempo real a partir de dados oriundos de sensores de vazão e pressão. A abordagem desenvolvida é baseada no paradigma de computação em borda (*edge computing*), onde foram identificadas limitações decorrentes dos próprios sensores. Especificamente, foi identificada a incapacidade do sensor de vazão de baixo custo em identificar fluxos menores do que um litro por minuto, assim vazamentos decorrentes de dispositivos que estejam gotejando por alguma falha ou desgaste, por exemplo, não seriam detectados. A contribuição deste trabalho consiste na implementação da técnica CNZ, originalmente proposta para uma abordagem em névoa, no contexto de computação em borda, permitindo assim uma economia de energia do sistema.

Os desafios tecnológicos e metodológicos para a detecção de vazamento de água em residências são inúmeros, principalmente

em contextos não-destrutivos. Assim, os próximos passos deste trabalho envolvem avaliar outras regras de associação que permitam a detecção de vazamento tanto em arquiteturas baseadas em computação em borda quanto em névoa (*fog computing*). Além disso, novos experimentos são necessários para avaliar o comportamento da pressão envolvendo diferentes dispositivos consumidores (ex: chuveiro, torneira com aerador, descarga e etc). A avaliação de como um vazamento impacta na pressão também será realizada, assim como a distância em que essa variação da pressão é percebida na tubulação, tanto decorrente do consumo quanto de algum vazamento. Uma avaliação sobre a comunicação do sistema também será realizada, visando verificar i nterferências d e r edes W I-FI e dispositivos eletrônicos, além do consumo energético da solução proposta.

ACKNOWLEDGMENTS

Os autores agradecem a Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e ao CEFET/RJ pelo apoio financeiro para o desenvolvimento deste trabalho.

REFERENCES

[1] Bárbara Muniz Vieira. Entenda por que está chovendo menos no Brasil e se há risco de nova crise hídrica em SP, 2021. URL https://g1.globo.com/sp/sao-paulo/noticia/2021/06/14/por-que-esta-chovendo-menos-e-sao-paulo-pode-viver-nova-crise-hidrica.ghtml.

[2] Trata Brasil. Perdas de água 2021 (snis 2019): Desafios para disponibilidade hídrica e avanço da eficiência do saneamento básico. 2021. URL http://www.tratabrasil.com.br/images/estudos/Perdas_d%C3%A1gua/Estudo_de_Perdas_2021.pdf.

[3] Henry Fuentes and David Mauricio. Smart water consumption measurement system for houses using IoT and cloud computing. *Environmental Monitoring and Assessment*, 192(9), 2020. ISSN 15732959. doi: 10.1007/s10661-020-08535-4. URL https://link-springer-com.ez108.periodicos.capes.gov.br/article/10.1007/s10661-020-08535-4.

[4] Tracy C. Britton, Rodney A. Stewart, and Kelvin R. O'Halloran. Smart metering: Enabler for rapid and effective post meter leakage identification and water loss management. *Journal of Cleaner Production*, 54:166–176, 2013. ISSN 09596526. doi: 10.1016/j.jclepro.2013.05.018. URL http://dx.doi.org/10.1016/j.jclepro.2013.05.018.

[5] Ayse Muhammetoglu, Yalçın Albayrak, Mustafa Bolbol, Simge Enderoglu, and Habib Muhammetoglu. Detection and Assessment of Post Meter Leakages in Public Places Using Smart Water Metering. *Water Resources Management*, 34 (9):2989–3002, 2020. ISSN 15731650. doi: 10.1007/s11269-020-02598-1. URL https://link.springer.com/article/10.1007/s11269-020-02598-1.

[6] Lucas de Araújo Wanderley Romeiro. *Técnicas De Desagregação De Consumo D'Água a Partir Da Dinâmica De Vazão Em Residências*. PhD thesis, UNIVERSIDADE FEDERAL DA BAHIA, 2019. URL https://repositorio.ufba.br/ri/handle/ri/29647.

[7] Sergej Svorobej, Patricia Takako Endo, Malika Bendechache, Christos Filelis-Papadopoulos, Konstantinos M Giannoutakis, George A Gravvanis, Dimitrios Tzovaras, James Byrne, and Theo Lynn. Simulating fog and edge computing scenarios: An overview and research challenges. *Future Internet*, 11(3):55, 2019. URL https://www.mdpi.com/1999-5903/11/3/55.

[8] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International journal of communication systems*, 25(9):1101, 2012. URL https://doi.org/10.1002/dac.2417.

[9] Ipin Prasojo, Andino Maseleno, Nishith Shahu, et al. Design of automatic watering system based on arduino. *Journal of Robotics and Control (JRC)*, 1(2):59–63, 2020. URL https://journal.umy.ac.id/index.php/jrc/article/view/7736.

[10] Bammidi Deepa, Chukka Anusha, and P Chaya Devi. Smart agriculture using iot. In *Intelligent System Design*, pages 11–19. Springer, 2021. URL https://link.springer.com/chapter/10.1007/978-981-15-5400-1_2.

[11] A. S. E. Moris. *Measurement and Instrumentation Principles*. Butterworth-Heinemann, Oxford, 3a edition, 2001.

[12] Seeed. How to use Water Flow Sensor / Meter with Arduino, 2020. URL https://www.seeedstudio.com/blog/2020/05/11/how-to-use-water-flow-sensor-with-arduino/.

[13] Srikanta Patnaik, Siddhartha Sen, and Magdi S. Mahmoud. *Smart Village Technology: Concepts and Developments*. Springer, Gewerbestrasse, Suíça, 17 edition, 2020. ISBN 978-3-030-37794-6. doi: https://doi.org/10.1007/978-3-030-37794-6.

[14] Preeja Pradeep and Shivsubramani Krishnamoorthy. The mom of context-aware systems: A survey. *Computer Communications*, 137:44–69, 2019. URL https://www.sciencedirect.com/science/article/pii/S0140366418309472.

[15] Maarten Van Steen and Andrew S Tanenbaum. *Distributed systems*. Maarten van Steen Leiden, The Netherlands, 2017.

[16] Ravi Kishore Kodali and Sree Ramya Soratkal. MQTT based home automation system using ESP8266. *IEEE Region 10 Humanitarian Technology Conference 2016, R10-HTC 2016 - Proceedings*, 2017. doi: 10.1109/R10-HTC.2016.7906845. URL https://ieeexplore.ieee.org/abstract/document/7906845.

[17] João Ferreira da Silva Júnior. *Detecção De Perdas Em Sistemas De Distribuição De Água Através De Rede De Sensores Sem Fio*. PhD thesis, UFPE, 2017. URL https://repositorio.ufpe.br/handle/123456789/29444.

[18] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016. URL https://ieeexplore.ieee.org/abstract/document/7469991.

[19] Nour Alhuda Sulieman, Lorenzo Ricciardi Celsi, Wei Li, Albert Zomaya, and Massimo Villari. Edge-Oriented Computing: A Survey on Research and Use Cases. *Energies*, 15(2):1–28, 2022. ISSN 19961073. doi: 10.3390/en15020452.

[20] Mohammed Laroui, Boubakr Nour, Hassine Moungla, Moussa A. Cherif, Hossam Afifi, and Mohsen Guizani. Edge and fog computing for IoT: A survey on current research activities & future directions. *Computer Communications*, 180 (September):210–231, 2021. ISSN 1873703X. doi: 10.1016/j.comcom.2021.09.003. URL https://doi.org/10.1016/j.comcom.2021.09.003.

[21] Amir Vahid Dastjerdi, Harshit Gupta, Rodrigo N Calheiros, Soumya K Ghosh, and Rajkumar Buyya. Fog computing: Principles, architectures, and applications. In *Internet of things*, pages 61–75. Elsevier, 2016. URL https://doi.org/10.1016/B978-0-12-805395-9.00004-6.

[22] Anthony T Velte, Toby J Velte, and Robert Elsenpeter. Cloud computing. *A practical approach*, pages 135–140, 2010.

[23] Nitinder Mohan and Jussi Kangasharju. Edge-Fog cloud: A distributed cloud for Internet of Things computations. *2016 Cloudification of the Internet of Things, CIoT 2016*, pages 1–6, 2017. doi: 10.1109/CIOT.2016.7872914.

[24] Dimitrios Amaxilatis, Ioannis Chatzigiannakis, and Christos Tselios. applied sciences A Smart Water Metering Deployment Based on the Fog Computing Paradigm. *Applied Sciences*, pages 1–28, 2020. doi: 10.3390/app10061965. URL https://www.mdpi.com/2076-3417/10/6/1965.

[25] Matthew Henschke, Xinzhou Wei, and Xiaowen Zhang. Data Visualization for Wireless Sensor Networks Using ThingsBoard. *2020 29th Wireless and Optical Communications Conference, WOCC 2020*, 2020. doi: 10.1109/WOCC48579.2020.9114929. URL https://ieeexplore.ieee.org/abstract/document/9114929.

[26] Bharati Wukkadada, Kirti Wankhede, Ramith Nambiar, and Amala Nair. Comparison with HTTP and MQTT in Internet of Things (IoT). *Proceedings of the International Conference on Inventive Research in Computing Applications, ICIRCA*, pages 249–253, 2018. doi: 10.1109/ICIRCA.2018.8597401. URL https://ieeexplore.ieee.org/document/8597401.