Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Utilizando sensores como tecnologia a serviço da prevenção de acidentes envolvendo vazamento de Gás GLP:

Aplicação “vazou!”. Um estudo prático.

Célio Hauck Carreira Júnior[1]
Centro Universitário Academia, Juiz de Fora, MG

Tassio Ferenzini Martins Sirqueira[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: Computação básica

RESUMO

É inegável que o gás (GLP) é vital para o comércio, transporte, agronegócio e residências para diversos fins, por ser combustível e ter ampla utilização no Brasil, um possível vazamento pode colocar em risco a vida das pessoas que estão nas proximidades, gerando prejuízos à saúde e econômicos. Acidentes como explosões, incêndios e asfixia são os casos mais comuns, portanto, um sistema de detecção é fundamental para ajudar na prevenção desses casos e na proteção das pessoas (MOREIRA, 2015). Este artigo consiste no desenvolvimento de um sistema que previne acidentes envolvendo vazamentos de gás (GLP) através do monitoramento de vazamentos por hardware, como sensores, emissores e software como serviços que recebe as informações capturadas pelos sensores e notifica as pessoas responsáveis. A ideia do sistema é identificar e alertar aos responsáveis sobre qualquer ambiente monitorado onde esteja ocorrendo algum vazamento, ajudando assim a prevenir acidentes que possam ocorrer com vazamentos de gás. Como resultado a aplicação demonstrou satisfatória em ambientes controláveis, notificando quando ocorre um vazamento.

Palavras-chave: Vazamento. GLP. ESP32. Sensores. Incêndio

ABSTRACT

It is undeniable that gas (LPG) is vital for trade, transport, agribusiness, and homes for various purposes because it is fuel and widely used in Brazil. A possible leakage can endanger the lives of people nearby, generating health and economic losses. Accidents such as explosions, fires, and asphyxiation are the most common cases, so a detection system is essential to help prevent these cases and protect people (MOREIRA, 2015). This article consists of developing a system that prevents accidents involving gas leaks (LPG) by monitoring leaks through hardware, such as sensors and emitters, and software as services that receive the information captured by the sensors and notifies the responsible people. The idea of the system is to identify and alert the responsible parties about any monitored environment where a leak is occurring, thus helping to prevent accidents that may occur with gas leaks. As a result, the application proved satisfactory in controllable environments, notifying when a leak occurs.

[1] Discente do Curso de Engenharia de software do Centro Universitário Academia – UniAcademia. Endereço: Rua Manoel Alves Sobrinho, número 102, apartamento 302, Graminha, 36242-296. Celular: (32) 98458-6000. E-mail: celiohauck@gmail.com.
[2] Docente do Curso de Engenharia de software do Centro Universitário Academia. Orientador.

1 INTRODUÇÃO

O gás liquefeito de petróleo (GLP) ou comumente conhecido como gás de cozinha, contem gases inflamáveis sendo resultado da separação de frações mais leves do petróleo durante processo de refino. É composto por uma mistura de gases hidrocarbonetos, como propano, butano, propeno e buteno e, devido a essa combinação de gases ele possui alto poder calorífico, de excelente qualidade de combustão, de fácil manuseio e baixo impacto ambiental (CONSIGAZ, 2017).

É inegável a importância do gás (GLP) para diversos usos na indústria, comércio, transporte, agronegócio e residências para diversos fins, conforme os dados do Sindigás (Sindicato Nacional das Empresas Distribuidoras de Gás Liquefeito de Petróleo, 2021), 91% da população do país, em 100% dos municípios, usaram o GLP em 2019, sendo entregues 20 botijões de até 13 kg por segundo, além de gerarem 380 mil empregos diretos e indiretos.

Por se tratar de uma substância altamente inflamável e muito utilizada, já ocorreram muitos acidentes resultados no vazamento desse gás, levando a explosões, incêndios e asfixia, em 2015 foram registrados 3.391 vazamentos com GLP no estado de São Paulo, resultando em um aumento de 6,13% em relação ao ano anterior, nesse mesmo ano ocorreram 358 incêndios com GLP (LIQUIGÁS, 2020). Entre 2000 e 2019, intoxicação acidental por gás matou ao menos 532 pessoas no Brasil, a categoria de gases inclui monóxido de carbono, dióxido de enxofre, oxido de nitrogênio entre outros, alguns desses casos o gás não emitiu odor ou a pessoal não conseguiu sentir (GLOBO, 2019).

Vazamentos de gás ocorrerão inevitavelmente, o que representa um risco para as pessoas próximas ao vazamento. Infelizmente, nem todos os vazamentos têm um odor que ajuda a detectar o vazamento e existem pessoas que são incapazes de sentir o cheiro, por conta dessas possibilidades existem componentes eletrônicos e soluções de software que podem ajudar a identificar vazamentos, uma dessas soluções é a SmartGás, sendo uma plataforma inteligente de monitoramento (MEDEIROS; SANTOS M, 2017).

É inegável que o vazamento de gás (GLP) é perigoso, pois se uma pessoa inalar grandes quantidades de gás, pode ter sérios problemas de saúde e até levar à morte. Além disso, o contato com faíscas pode causar incêndios e explosões, colocando em risco a vida das pessoas.

Devido aos riscos apresentados, este artigo implementa uma solução que inclui tanto a parte de hardware, como sensores e transmissores, quanto a parte de software. Na parte de hardware, foi construído um sistema eletrônico para capturar o vazamento de gás e emitir um aviso sonoro quando o vazamento atinge determinado nível. Quando essa informação é capturada o sistema eletrônico envia a informação via *HTTP* para um serviço Web que trata o dado e realiza a notificações para as pessoas responsáveis.

A estrutura deste artigo está dividia na seguinte forma: na seção 2, temos um referencial teórico relacionado ao que vem sendo pesquisado e desenvolvido para reduzir os problemas relacionados ao vazamento de gás (GLP). Na seção 3, estabelecemos o objetivo do artigo dividindo em: objetivos primários e secundários. Na seção 4, detalhamos as tecnologias utilizadas e quais componentes eletrônicos são utilizados no sistema. Na seção 5, apresentamos os resultados da aplicação, realizados em ambientes controlados. Na seção 6, tiramos as conclusões do artigo.

2 REFERENCIAL TEÓRICO

Segundo as informações da Fundacentro[3] publicada por Santos (2019), “O GLP pode trazer riscos à saúde dos trabalhadores e, também, da população e do meio ambiente”. Por causa desses riscos, pesquisas têm sido realizadas para ajudar a prevenir acidentes e garantir a segurança das pessoas.

Na última década foi realizado um estudo para detectar vazamentos de gás usando uma combinação de hardware e software. O resultado da pesquisa é um dispositivo de detecção com a função de enviar mensagens curtas para celulares cadastrados, que visa reduzir a ocorrência de vazamentos (SANTOS J, 2012).

Com a evolução da tecnologia e da Internet, outras pesquisas vêm sendo realizadas utilizando novos recursos, como a Internet das Coisas que pode coletar e processar diversas informações, proporcionando comodidade, segurança, praticidade e economia, principalmente para o ambiente doméstico, onde espaço residencial é transformado em um ambiente inteligente (MEDEIROS; SANTOS M, 2017).

Um estudo que utiliza a Internet das Coisas é a plataforma "smartgás", composta por componentes eletrônicos capazes de captar o peso de botijões de gás e monitorar possíveis vazamentos, o hardware envia essas informações para um serviço em nuvem e exibe essas informações através de um aplicativo móvel (MEDEIROS; SANTOS M, 2017).

Outro estudo ocorreu com o intuito de desenvolver um protótipo de tecnologia vestível para detecção de GLP e monóxido de carbono no meio ambiente. O protótipo vestível pode não apenas monitorar a frequência cardíaca do usuário, mas também capturar informações e enviá-las a um serviço da Web. (ARAUJO; *etc*, 2020).

Como resultado das pesquisas citadas acima, este artigo propõe uma solução para prevenir acidentes de vazamento utilizando hardware e software. A parte do software é toda a Web, incluindo a interface visual. Uma das razões para a escolha do site é o fato de ser construído como um *PWA,* tornando possível juntar as vantagens de um site com a experiência de um aplicativo nativo.

3 OBJETIVOS

Definimos alguns objetivos que precisam ser alcançados para desenvolver este artigo, entre os quais podemos dividi-los em dois: objetivo principal e objetivos secundários. Os objetivos secundários são divididos em tópicos para favorecer a fase de construção.

a. Objetivo principal

O objetivo principal deste artigo é construir um sistema que utilize software e hardware para prevenir vazamentos de GLP através do uso de sensores. Para atingir este objetivo, foram estabelecidos dois sistemas distintos, um é o sistema eletrônico e o outro é o Web.

[3] Fundacentro Entrevista com Alexandre Sá - Especialista da Enesens. Disponível em < https://www.youtube.com/watch?v=FZzPW8imafk>. Acessado em: 02 out. 2021.

Acessado

O sistema eletrônico é composto por módulo ESP32 NodeMCU, conectado com sensor de gás MQ2 e com o emissor sonoro *Buzzer*, esses componentes são úteis para notificar pessoas próximas de onde o sistema está instalado e se comunicar com os serviços da Web para notificar todas as pessoas responsáveis independentemente de onde esteja.

Por sua vez, o sistema Web consiste em um *back-end* em *nodejs*, um front-end em *React* com PWA, um banco de dados não relacional hospedado pelo *Firebase*[4] e um serviço de notificação que também é hospedado pelo *Firebase*.

b. Objetivos secundários

Para auxiliar a parte do desenvolvimento, o objetivo principal está divido em objetivos secundários, divididos em 6 tópicos:

i. Construir o dispositivo eletrônico;
ii. Fazer a detecção de vazamento de gás;
iii. Desenvolver uma interface visual em que seja possível visualizar o nível do gás;
iv. Desenvolver a notificação de quando ocorrer um vazamento crítico;
v. Desenvolver um alerta sonoro para ser executado quando ocorrer um vazamento crítico.
vi. Validar aplicação em um ambiente controlado.

4 METODOLOGIA

O trabalho está dividido em duas grandes partes, responsáveis pelo desenvolvimento do sistema eletrônico de medição de gás, sendo a parte de hardware, e pelo desenvolvimento do sistema de visualização do nível de gás e envio de notificações, sendo a parte de software.

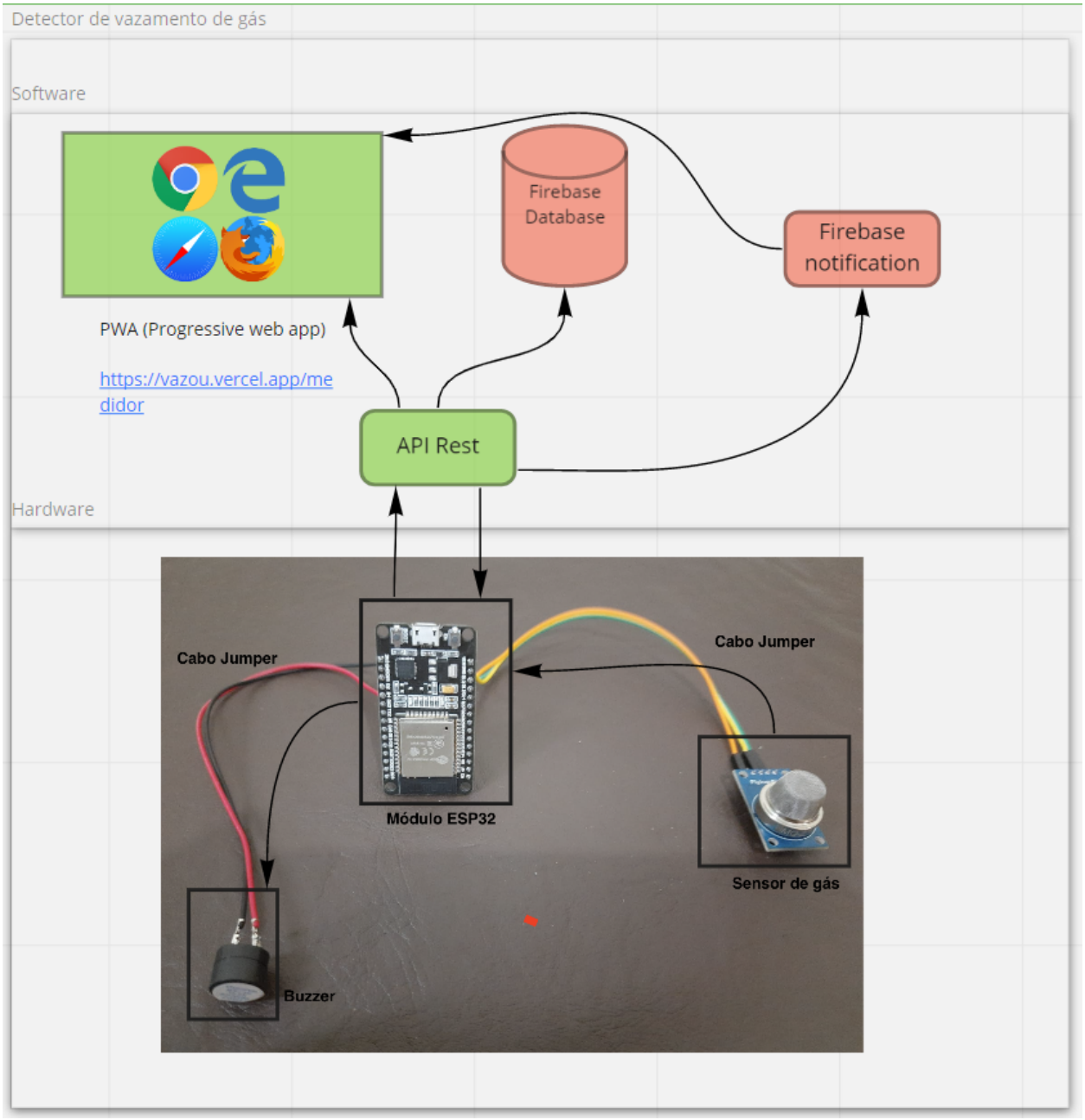
Para facilitar a compreensão do princípio de funcionamento do aplicativo, fornecemos um diagrama do projeto mostrando as principais partes do hardware e software na Figura 1. Na parte inferior temos o sensor de gás, o emissor sonoro *Buzzer* e o módulo EPS32 que conecta todos os dispositivos. Este módulo, por sua vez, faz chamadas *HTTP* para a *API* que processa as informações e as salvas no banco de dados do *Firebase*. Se as informações recebidas forem significativas, a *API* acionará uma notificação para o serviço de notificação do *Firebase*, e a interface visual recebera notificações deste serviço mostrando a mensagem, além disso, a interface enviará solicitações periodicamente para pesquisar o último vazamento válido na *API*.

Em seguida, será explicado o funcionamento de cada componente da Figura 1 e a conexão de cada componente. Para isso essa seção será dividia em dois tópicos diferentes sendo elas o hardware e o software.

[4] Firebase. Disponível em: < https://firebase.google.com/>. Acessado em 25 de novembro de 2021.

Acessado

Figura 1 - Desenho do projeto.



Fonte: Elaboração própria.

a. Hardware

Conforme mostrado na Figura 1, na parte inferior temos todos os módulos e sensores utilizados para capturar vazamentos de GLP. O módulo se comunica com a *API* desenvolvida através de requisições *HTTP*, para isso desenvolveu-se um código que faz toda essa lógica no módulo. Nos próximos parágrafos entraremos em detalhes sobre todos os componentes utilizados por esse sistema eletrônico.

Na Figura 2, temos o ESP32, que é um dispositivo *IoT* (Internet das Coisas) que consiste em um microprocessador *Tensilica Xtensa LX6* de 32 bits dual-core com suporte para rede *WiFi* integrado, *Bluetooth* v4.2 e memória flash integrada. Essa arquitetura permite que seja programado independentemente, sem a necessidade de outras placas microcontroladoras como o Arduino, por exemplo.

A placa contém Wifi e Bluetooth nativos, por isso é amplamente utilizada em projetos de automação residencial, e por ser um módulo *IoT* códigos podem ser inseridos.

O funcionamento do módulo exige que o módulo ESP32 seja conectado à fonte de alimentação. Neste artigo, ele é conectado ao computador através de um cabo micro USB, e a entrada está na parte superior da placa, conforme mostrado em Figura 2.

É importante notar que na Figura 2 existe um botão denominado BOOT próximo à entrada micro USB, que precisa ser pressionado durante a compilação do código para sobrescreve-lo no módulo.

Figura 2 - Módulo ESP32 NodeMCU.

Fonte: TechSuleletronicos Adaptada.[5]

Na Figura 3, temos o MQ-2, sendo um componente eletrônico projetado para detectar a presença de gás / fumaça inflamável em um ambiente fechado. Com base na detecção do sensor, o microcontrolador conectado ao módulo será notificado e poderá realizar uma ou mais ações determinadas pelo usuário.

O sensor pode detectar os seguintes gases: metano, propano, butano, gás natural, gás liquefeito de petróleo e hidrogênio. Ele pode ser definido com um componente de *INPUT*, pelo

[5] Modulo ESP32. Disponível em <vazamentos porhardware, como sensores, emissores e software como serviços que recebe as informações capturadas pelos sensores e notifica as pessoas responsáveis.. Acessado em: 08 set. 2021

Acessado

fato dele receber informações e dar possibilidades de ações para quem estiver programando as funcionalidades.

Figura 3 - Sensor MQ2.

Fonte: MarinoStore Adaptada.[6]

Na Figura 4 temos o *Buzzer*, que é um componente eletrônico que emite efeitos sonoros em forma de alarme, é definido como um emissor *OUTPUT* porque não recebe nenhuma informação do módulo conectado, mas sim porque reproduz informações para o módulo.

Figura 4 – *Buzzer*.

Fonte: arducore.[7]

[6] Modulo ESP32. Disponível em <https://www.marinostore.com/sensores/sensor-de-gas-mq-2>. Acessado em: 13 set. 2021
[7] Modulo ESP32. Disponível em <https://www.arducore.com.br/buzzer-5v >. Acessado em: 13 set. 2021.

Acessado

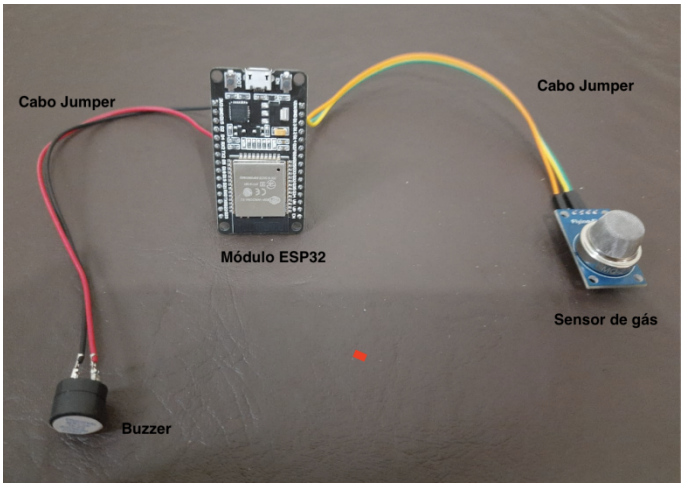
Na Figura 5, temos os cabos *jumper* para prototipagem de componentes eletrônicos, muito adequados para a conexão entre os componentes eletrônicos do projeto, sem eles a conexão não pode ser feita.

Figura 5 - Cabo *Jumper*.

Fonte: casadarobotica.[8]

Após adquirir todos os componentes acima, realizou-se a conexão do cabo *jumper* ao Módulo ESP32, ao sensor de gás e ao *buzzer*, além da conexão de um cabo *jumper* nas entradas VCC e GND do sensor de gás e a outra parte do cabo no pino 34 do Módulo ESP32 e o outro cabo *jumper* no *buzzer* ao pino 18 do Módulo ESP32, com isso a montagem da parte elétrica foi feita, conforme a figura 6.

Figura 6 – Protótipo.



[8] Cabo Jumper. Disponível em < https://casadarobotica.mercadoshops.com.br/MLB-1484347411-cabo-jumper-macho-x-fmea-10-cm-20-unidades-para-arduino-mxf-_JM>. Acessado em: 13 set. 2021 gases inflamáveis sendo resultado da separação d

Acessado

Fonte: Elaboração própria.

O sistema eletrônico é composto por todos os componentes citados, mas o hardware sozinho não pode atingir as funções esperadas do sistema. No próximo tópico, será apresentada a arquitetura de software, mostrando as tecnologias e serviços utilizados, e como construí-los para atender aos objetivos propostos.

b. Software

A parte do Software é dívida em três partes distintas, sendo elas: *back-end, front-end* e *scripts* de módulo. Explicaremos cada parte e suas responsabilidades começando pelo *back-end.*

O *back-end* é responsável pelas regras de negócio, organizada da seguinte forma: *API REST* que é a aplicação centralizadora das regras, *Firebase Realtime Database* que é o banco de dados e o *Firebase Cloud Messaging* que é o serviço de notificação, será explicado as partes envolvida no *back-end* nos próximos parágrafos.

API REST é definida como uma interface de programação de aplicações que está em conformidade com as restrições do estilo de arquitetura *REST* (REDHAT, 2020). Neste projeto, uma *API REST* foi desenvolvida para se comunicar com o *firebase* e ser consumida pelo *front-end* para recuperar valores válidos de vazamento. A *API* verifica as informações recebidas e realiza uma série de verificações para processar as informações. A *API* usa node.js na sua construção por conta da sua estrutura e proposta.

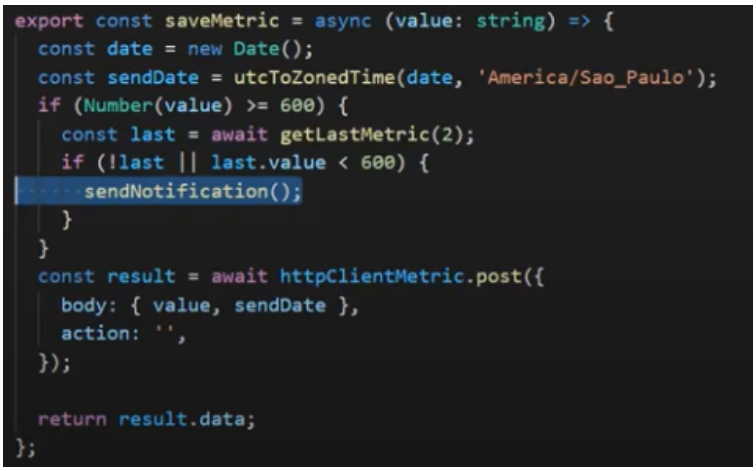
O *node.js* é fácil de usar e manter, ele possui suporte a OO e programação funcional, utiliza *event-driven, no-blocking i/o model* tornando ele leve e eficiente (NODEJS, 2021). Pela natureza da solução, de receber e tratar informações assíncronas e por conta das qualidades do *nodejs*, este software foi escolhido na construção da API.

Na *API* foi desenvolvido três *controllers* diferentes, chamados ***health-check***, ***metrics*** e ***token***. Esses *controllers* são os pontos de entrada da *API* e cada controller possui uma série de *endpoints* para executar operações individualmente.

O *controller health-check* contém apenas um *endpoint* para verificar a integridade da *API*, retornando uma mensagem de sucesso.

No *controller* de *metrics*, temos três *endpoints* responsáveis por obter todos os indicadores salvos, salvar os indicadores e obter o último indicador válido salvo, todos esses indicadores são armazenados no *Firebase Realtime Database*, antes de salvar os indicadores algumas verificações são realizadas, primeiro é valido se o valor da métrica recebida é igual ou superior a 600, se sim, é comparado com o valor da última métrica salva, caso seja menor é realizado o envio de uma notificação, informando o vazamento de gás. Podemos visualizar essa funcionalidade na Figura 7.

Figura 7 - Função *saveMetric*.



```
export const saveMetric = async (value: string) => {
  const date = new Date();
  const sendDate = utcToZonedTime(date, 'America/Sao_Paulo');
  if (Number(value) >= 600) {
    const last = await getLastMetric(2);
    if (!last || last.value < 600) {
      sendNotification();
    }
  }
  const result = await httpClientMetric.post({
    body: { value, sendDate },
    action: '',
  });

  return result.data;
};
```

Fonte: Elaboração própria.

O *controller* de token tem apenas um endpoint responsável por salvar o token, utilizando *firebase* como banco de dados, toda vez que o site é aberto pela primeira vez em outro dispositivo ou navegador, um token é gerado e o token gerado é armazenado no banco de dados para ser utilizado no envio de notificação.

A *API* é constituída por todos *controllers* e *endpoints* apresentados e está disponível gratuitamente no *Heroku*[9], que é uma plataforma cloud que faz *deploy* de várias aplicações *back-end*, seja para hospedagem, testes em produção ou para escalar aplicações.

A outra parte incluída como *back-end* é um banco de dados não relacional *Firebase Realtime Database* hospedado pelo Google na nuvem. Os dados são armazenados como *JSON* e sincronizados em tempo real para cada cliente conectado. Quando você cria aplicativos de plataforma cruzada com o *SDKs* para iOS, Android e *JavaScript*, todos os usuários compartilham uma instância do *Realtime Database* e recebem atualizações automaticamente com os dados mais recentes (DUFETEL,2017).

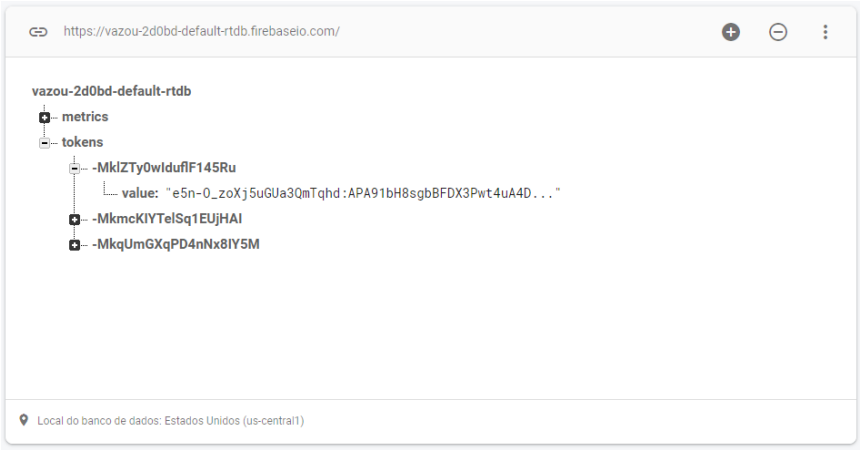
No projeto foi utilizado o *Firebase Realtime Database* em quatro momentos, para salvar todos os indicadores recebidos pela *API*, salvar todos os tokens gerados para o envio de notificações, buscar tokens e indicadores validos para os solicitantes.

Conforme mostrado na Figura 8, neste banco de dados, temos dois objetos principais: *metrics* e o tokens, as informações apresentadas a seguir são dados do banco de dados de teste, onde as mesmas foram inseridas via *HTTP* pela *API*.

[9] Heroku. Disponível em: < https://www.heroku.com/ >. Acessado em 25 de novembro de 21.

Acessado

Figura 8 – *Realtime database.*



Fonte: Elaboração própria.

A última parte que compõem o *back-end* é o serviço de notificação *Firebase Cloud Messaging,* que oferece uma opção de criar uma plataforma de notificação com o mínimo de esforço de codificação, possibilitando a função de *push notification.*

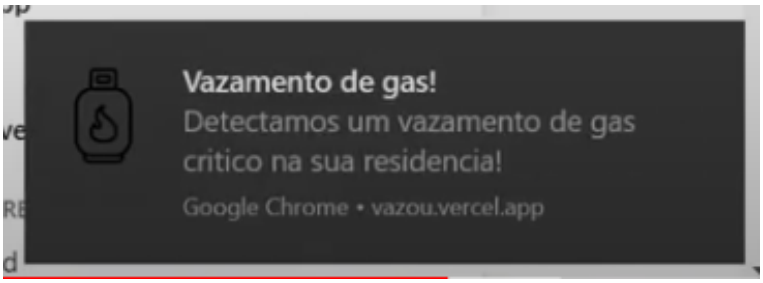
Push notification pode ser definida como um serviço utilizado para enviar notificações curtas para celulares, muita das vezes é utilizado para ajudar a notificar usuários em curto prazo (SIDDIK; NASUTION, 2018).

Neste projeto, além de avisos sonoros quando o gás é detectado, uma integração com *Firebase Cloud Messaging* também foi adicionada para enviar notificações curtas quando o gás é detectado, um exemplo dessa notificação se encontra na Figura 9.

Para fazer os *push notifications* funcionarem da melhor maneira, é preciso configurar o *SDK* no *back-end* para ele poder obter tokens válidos e acionar mensagens, e usar notificações padrão do navegador no front-end.

Nos próximos parágrafos, apresentaremos a arquitetura *front-end* em detalhes, explicaremos quais tecnologias são usadas e como receber *push notification* com eficácia.

Figura 9 - *Push notification* recebido.



Fonte: Elaboração própria.

Acessado

O *front-end* é a parte visual da aplicação, tendo um foco maior na camada de interface da aplicação, onde o usuário irá visualizar as informações necessárias, neste artigo a parte visual é um site em formato *PWA* que permite receber notificações em segundo plano.

Segundo o Mozilla (2021) o *PWA* é definido como: “aplicativos web desenvolvidos usando várias tecnologias e padrões específicos para permitir que eles aproveitem os recursos da Web e dos aplicativos nativos.” A ideia é combinar o melhor da Web com os aplicativos, permitindo a instalação de sites no formato *PWA*, utilizando funções quando não há conexão com a Internet e assim por diante.

Para a nossa aplicação Web o *PWA é* essencial para realizar a parte de *push notification*, por conta dos seus recursos de poder receber notificações em segundo plano, através do *serviceWork*. Após explicar a motivação para usar o *PWA*, apresentaremos em detalhes as tecnologias na construção do *front-end*.

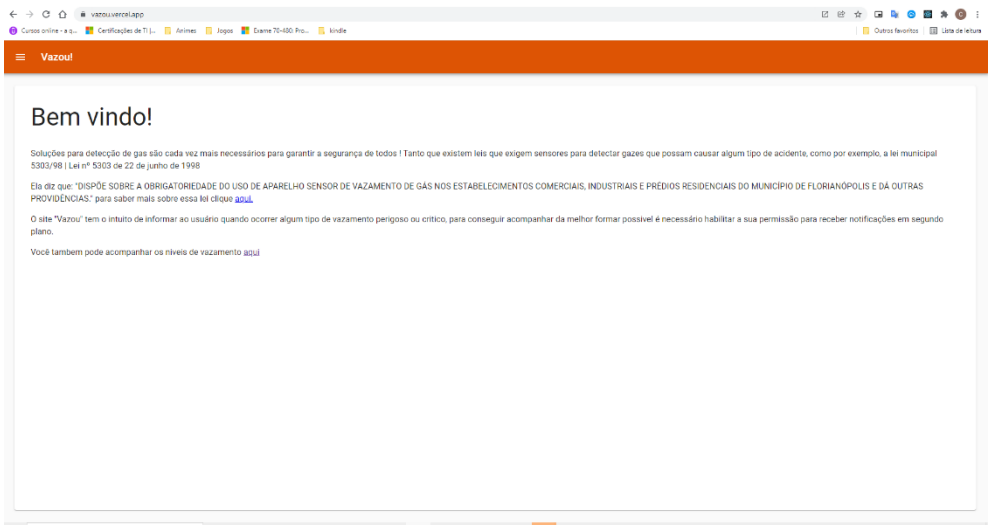
O site é desenvolvido usando *React* que pode ser definida como uma biblioteca para criação de interfaces com usuário de forma dinâmica (REACT, 2021). Uma das razões para usar essa tecnologia é que projetos simples podem ser criados facilmente sem adicionar dependências desnecessárias ao projeto.

No site desenvolvido, *React* é usado com *Material-ui*, que é definido por *Material-ui* (2021) como: “Biblioteca de componentes *React* para um desenvolvimento ágil e fácil”. Por ter um grande número de componentes estilizáveis, ela foi escolhida para acelerar o processo de desenvolvimento.

No site, temos duas telas, uma das quais é a tela principal, que contém informações úteis sobre como o site funciona e o porque dele existir, podemos ver uma imagem dessa tela na Figura 10. A outra é a tela de medição, que contém um multímetro, de 0 a 1000, dividido em três partes. Quando uma medição válida de vazamento de gás é encontrada, o valor do multímetro muda. Podemos observar a imagem relacionada a esta tela na figura 11.

Caso tenha interesse em verificar a aplicação em seu dispositivo móvel ou computador, basta acessar o link: https://vazou.vercel.app/medidor. Se o seu navegador ou sistema operacional suportar *PWA*, você pode instalar o site no seu sistema operacional.

Figura 10 – Primeira tela.



Acessado

Fonte: Elaboração própria.

Figura 11 – Segunda tela.

Fonte: Elaboração própria.

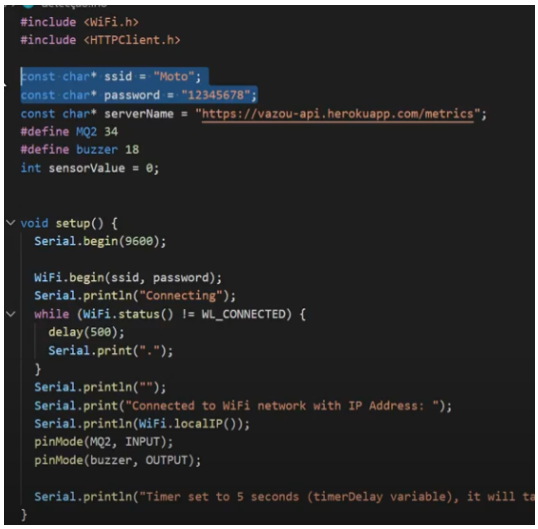
O site está hospedado no *Vercel,* que é uma plataforma em nuvem para sites estáticos e *frameworks front-end*. O principal motivo para a escolha de hospedar um site no *Vercel*[10] é ter um certificado SSL gratuito, e promover *CI* (*Continuous Integration*).

Além do desenvolvimento do back-end e front-end, o script do módulo ESP32 foi estruturado, a conexão Wifi foi adicionada e a especificação de onde o Sensor MQ2 e o *Buzzer* foram alocados, o Sensor MQ2 foi alocado no pino 34 do Módulo ESP32 e o *Buzzer* no pino 18. O *pinMode* do Sensor MQ2 é configurado como *INPUT*, pois retornará um valor ao Módulo ESP32, e o *pinMode* do *Buzzer* é configurado como *OUTPUT*, pois receberá apenas a informação se será acionado um alarme. Podemos ver essa estrutura na Figura 12.

[10] Vercel. Disponível em: < https://vercel.com/ >. Acessado em 25 de novembro de 2021.

Acessado

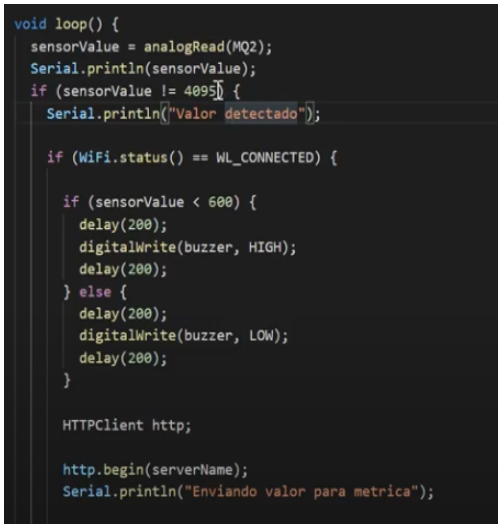
Figura 12 - Função *setup*.



Fonte: Elaboração própria.

No script, temos uma função chamada *loop* (), que verifica o valor de retorno do Sensor MQ2, ativa ou desativa o *buzzer* de acordo com o valor de retorno e envia o valor para a *API Rest* através de requisições *http*, então o valor é salvo e apresentado no *front-end*, por meio de notificação ou exibido na segunda tela do multímetro. Essa função está detalhada na Figura 14.

Figura 14 - Função *loop*.



Fonte: Elaboração própria.

Todos os códigos do projeto podem ser encontrados nos repositórios https://github.com/CelioHauck/vazou e https://github.com/CelioHauck/vazou-api.

No primeiro link temos o site construído e no segundo link temos a *API* junto com o código-fonte que foi compilado para o módulo.

Acessado

5. RESULTADOS E DISCUSSÃO

Após a conclusão de todo o processo de desenvolvimento de hardware, software e liberação do serviço, foi realizado testes de simulação de vazamento de gás com isqueiro. Os resultados desses testes são satisfatórios. Quando o sensor capta o nível de vazamento, o módulo obtém essas informações e as envia para o *endpoint* para registrar o indicador, em seguida, salva as informações no banco de dados e aciona a notificação para as pessoas que acessaram o site e habilitaram a permissão de notificação.

Ao desenvolver e implementar o sistema foi possível notar algumas limitações e dificuldades que podemos encontrar no processo de uso e implementação. Um desses problemas é que os usuários são obrigados a habilitar as permissões de notificação. Sem essa permissão, o *push notification* não pode ser realizado e impacta a eficácia das notificações no caso de um vazamento de GLP. Outra limitação que pode dificultar o processo de implantação é que você precisa compilar o código no módulo toda vez que você muda de *wi-fi*. Outra dificuldade é a falta de suporte do *PWA* com alguns *browsers*, por exemplo, o *PWA* não tem suporte com o Chrome no iOS, mas tem suporte com o Safari, essa incompatibilidade pode afetar a experiência final do usuário caso utilize um *browser* sem suporte.

Uma demonstração detalhada da aplicação, incluindo hardware e software, foi gravada e disponibilizada no YouTube, mostrando os componentes eletrônicos, como se comunicar com o serviço da web e como os usuários finais recebem notificações. A gravação está disponível no link: https://youtu.be/G1ZcOSKEykE.

6. CONSIDERAÇÕES FINAIS

Diante do cenário onde muitas pessoas usam GLP, mesmo com as medidas de segurança que temos atualmente, acidentes continuam acontecendo. Pode ser verificado pelo protótipo desenvolvido que, se houver vazamento, podemos notificar as pessoas de outras formas, seja por notificação ou aviso sonoro, porem melhorias no projeto precisam ser feitas para que mais pessoas possam utilizar, além disso, foi validado o objetivo principal e todos os objetivos específicos do projeto, que se resumia em conseguir construir um sistema eletrônico através do módulo ESP32 NodeMCU que seja capaz de detectar vazamentos de gás (GLP) e emitir um alerta sonoro, um serviço Web que recebe os dados do módulo, trata a informação e caso necessário envia a notificação e um site que mostra através de uma interface visual o nível de gás presente no ambiente e receba notificações enviadas pelos serviço.

Hoje no mercado existem sensores que verificam vazamento de gás, porém a maioria emite alerta sonoro ou visual, caso alguém esteja próximo para ouvir ou ver o sensor emitindo esses alertas, acidentes poderiam ser evitados, porem caso isso não ocorra o vazamento continuará acontecendo sem que ninguém perceba aumentado a chances de acontecer algum tipo de acidente, com essa solução proposta mais pessoas seriam alertadas diminuindo o risco de acidentes envolvendo vazamentos, vale ressaltar que a aplicação foi testa em ambientes controlados, seria interessante no futuro testar em ambientes reais, verificando a cobertura de detecção do sensor, a variação de gases e como pode ser implementado a alimentação do módulo ESP32 NodeMCU.

Como trabalhos futuros, a aplicação poderia ser aprimorada com o envio de tipos diferentes de notificações, como SMS ou e-mail, além de verificar a possibilidade de uma integração com alguma *API* do corpo de bombeiros, para eles serem notificados caso ocorra algum vazamento por um período maior de tempo. Além disso, criar um cadastro de clientes

seria uma necessidade no futuro, hoje não tem uma referência de qual placa está sendo utilizada por cada usuário e poderíamos ter essa informação para deixar a aplicação escalável para outros dispositivos.

REFERÊNCIAS

CONSIGAZ A CHAMA QUE FAZ SUA VIDA MELHOR. **GLP – GÁS LIQUEFEITO DE PETRÓLEO** - Disponível em <https://www.consigaz.com.br/gas-glp/>. Acesso em: 05 out. 2021

COPLE, Júlia. **Desde 2000, intoxicação acidental por gás matou ao menos 532 pessoas no Brasil,** Globo, 16 jul. 2019, disponível em <https://oglobo.globo.com/politica/desde-2000-intoxicacao-acidental-por-gas-matou-ao-menos-532-pessoas-no-brasil-23811253>. Acesso em: 05 out. 2021.

LIQUIGÁS. 22 Prêmio Fiesp de Mérito Ambiental, 2020, disponível

em: <http://www.fiesp.com.br/arquivo-download/?id=214426>. Acesso em: 05 out. 2021

Documentação,NodeJs, 2021, disponivel em: < https://nodejs.org/en/docs >. Acesso em: 02 out 2021.

Emanuel Mamani Araujo, Mateus. **Tecnologia vestível para detecção de GLP e monóxido de carbono do ambiente**, IFES, 2020, disponível em: <https://downloads.editoracientifica.org/articles/201102092.pdf>. Acesso em: 19 out. 2021.

Introdução, React, 2021, disponivel em: < https://pt-br.reactjs.org/docs/getting-started.html >. Acesso em:01 out. 2021.

LIMA, Cleyson. **Deploy de uma aplicação Spring Boot no Heroku**, TREINAWEB, disponível em: <https://www.treinaweb.com.br/blog/deploy-de-uma-aplicacao-spring-boot-no-heroku>. Acesso em: 14 out. 2021

Márcio Moreira, Alessandro. **Segurança na utilização de gás liquefeito de petróleo**, ufes, 2015, disponível em: <https://ambiental.ufes.br/sites/ambiental.ufes.br/files/field/anexo/seguranca_na_utilizacao_de_gas_liquefeito_de_petroleo_-_alessandro_marcio_moreira.pdf>. Acesso em: 22 out. 2021.

Maria Santos, Débora. **Especialistas comentam sobre os riscos e segurança dos gases industriais**, FUNDACENTRO, 18 out. 2019, disponível em:<https://www.gov.br/fundacentro/pt-br/assuntos/noticias/noticias/2019/10/especialistas-comentam-sobre-o-seminario-sobre-glp-gnv-monoxido-de-carbono-e-gases-naturais>. Acesso em: 02 out. 2021.

MasterWalker electronic shop, **Cabo Jumper Macho-Macho 20cm - KIT com 40pcs**, disponível em <https://www.masterwalkershop.com.br/cabo-jumper-macho-macho-20cm-kit-com-40pcs>. Acesso em: 05 out. 2021

MasterWalker electronic shop, **Sensor (Detector) de Gás Inflamável / Fumaça - MQ-2**, disponível em <https://www.masterwalkershop.com.br/sensor-detector-de-gas-inflamavel-fumaca-mq-2>. Acesso em: 05 out. 2021

MasterWalker electronic shop**. Módulo ESP32 NodeMCU IoT WiFi e Bluetooth**, disponível em

Acessado

<https://www.masterwalkershop.com.br/nodemcu-32s-esp-32s-wifi-bluetooth>. Acesso em: 05 out. 2021

MORONEY, Laurence. **The Firebase Blog, Introducing Firebase Notifications**, Firebase, 14 jun. 2016, disponível em <https://firebase.googleblog.com/2016/06/introducing-firebase-notifications.html>. Acesso em: 15 out. 2021

Nasution, Akmal. **Perancangan Aplikasi Push Notification Berbasis Android**. JURTEKSI, 2018, disponivel em: <http://jurnal.stmikroyal.ac.id/index.php/jurteksi>. Acesso em: 24 out. 2021.

Panorama do setor de glp em movimento, Sindigas, Janeiro 2021, disponível em: <https://www.sindigas.org.br/Download/PANORAMAS/NOVO%20GLP%20EM%20MOVIMENTO_JANEIRO_2021_Rev5.pdf>. Acesso em: 21 out. 2021.

Redhat, 08 maio 2020, **API REST**, disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api> Acesso em: 13 out. 2021

Silva Santos, Jefferson. **Detector de vazamento de gás com aviso por sms**, uniceub,2012 disponível em: <https://repositorio.uniceub.br/jspui/bitstream/235/3697/2/Monografia%20JEFFERSON%20SANTOS%202-2012.pdf >. Acesso em: 02 out. 2021.

Vitor da Silva Medeiros, Gabriel; Ricardo dos Santos, Matheus. **Smartgás: uma plataforma inteligente para monitoramento de gás de cozinha**, IFRN, 2017, disponível em: < https://memoria.ifrn.edu.br/bitstream/handle/1044/1441/SmartG%C3%A1s%20uma%20plataforma%20inteligente%20para%20monitoramento%20de%20g%C3%A1s%20de%20cozinha.pdf?sequence=1 >. Acesso em: 03 out. 2021.