Revista Ibero- Americana de Humanidades, Ciências e Educação- REASE

doi.org/10.51891/rease.v8i5.5490

UMA APLICAÇÃO WEB PARA MONITORAMENTO E ALERTA DE ALTAS TEMPERATURAS, GAS GLP E FUMAÇA DE UMA RESIDÊNCIA

A WEB APPLICATION FOR MONITORING AND ALERT OF HIGH TEMPERATURES, LPG GAS AND SMOKE FROM A RESIDENCE

Tasso Chaves de Oliveira[1]
Edilson Carlos Silva Lima[2]

RESUMO: Atualmente a grande maioria dos sensores de detecção de gás GLP e fumaça disponíveis no mercado são eficientes somente quando há pessoas na residência, e isso acabando sendo uma grande barreira, uma vez que as pessoas necessitam sair de casa para trabalhar ou para tratar de interesses pessoais. Esse artigo buscou desenvolver uma aplicação *web* baseada em internet das coisas (IoT), com o intuito de garantir uma maior prevenção contra incêndios residenciais, através do monitoramento remoto de temperatura, gás GLP e fumaça, bem como a emissão de alertas por mensagens de WhatsApp. Para tal propósito, foi utilizado a linguagem de programação Java (versão 11) com módulos do Spring *Framework*, juntamente com a placa de desenvolvimento NodeMCU e sensores para Arduino. Além do mais, durante os alertas é registrado em um banco de dados, as altas temperaturas e as detecções de gás e fumaça com data e horário do ocorrido. Permitindo, assim, que o usuário tenha acesso ao histórico de detecção de cada mês. Os testes realizados em uma maquete foram bastante satisfatórios, e a aplicação web demonstrou eficácia tanto no monitoramento quando nos alertas.

Palavras-chave: Aplicação web. Internet das Coisas. Gás GLP.

ABSTRACT: Currently, the vast majority of gas and smoke detection sensors available on the market are efficient only when there are people in the house, and this ends up being a great barrier, since people need to leave the house to work or to deal with personal interests. This article sought to develop a web application based on the Internet of Things (IoT), in order to ensure greater prevention against residential fires, through remote monitoring of temperature, LPG gas and smoke, as well as the issuance of alerts by WhatsApp messages. For this purpose, the Java programming language (version 11) was used with Spring Framework modules, along with the NodeMCU development board and sensors for Arduino. Furthermore, during alerts, high temperatures and gas and smoke detections are recorded in a database, with the date and time of the event. Thus allowing the user to have access to the detection history of each month. The tests performed on a mockup were quite satisfactory, and the web application demonstrated effectiveness in both monitoring and alerts.

Keywords: web application. Internet of Things. LPG gas.

[1] Graduando em Engenharia de Computação na Universidade Ceuma. E-mail: tassochav@gmail.com.
[2] Professor orientador. Universidade Ceuma. E-mail: edilsonlima3@gmail.com.

1INTRODUÇÃO

Em uma rápida pesquisa em portais de comunicações e sites dos bombeiros de vários estados do Brasil, é possível notar que a cada ano é registrado uma grande quantidade de ocorrências de incêndios residenciais. Os motivos para esses acidentes são bastante variados: vazamento de Gás Liquefeito de Petróleo (GLP), mais conhecido como gás de cozinha, curto-circuito na rede elétrica, aparelhos eletrônicos, distrações na cozinha entre outros.

O GLP, por ser inflamável quando em contato com o ar, acaba se tornando em determinadas situações um grande agente causador de incêndio. Ele é o gás mais utilizado em ambiente residencial, representando cerca de 85 % do consumo nacional, uma vez que ele atende as principais necessidades dos residentes, seja na cocção de alimentos ou no cozimento de água. Já o meio industrial é responsável por 15% do consumo, devido a sua utilização como combustível. (GAUTO, 2016, p. 32).

Por mais que o botijão proporciona segurança contra explosões, o vazamento ocasionado por problemas no regulador de gás, mangueira, ou até mesmo por negligencia dos moradores, pode trazer sérias consequências, como o risco de vida e perda patrimonial. Principalmente quando o vazamento é pequeno e não há ninguém no local para agir logo no início.

Tem-se atualmente no mercado, vários detectores de fumaça e gases inflamáveis, com alarme visual e sonoro para residência. Apesar de serem ótimos dispositivos de segurança, detectores desse tipo são eficientes somente quando há pessoas no local. Podemos afirmar que na prática é impossível eliminar todos os riscos de incêndios que existe em um ambiente residencial. No entanto, é de suma importância adotar medidas de prevenção para que seja possível sinalizar o usuário, de modo que consiga agir a tempo para acionar o corpo de bombeiros e evitar que este atinja proporções maiores.

Vivemos na era digital, período em que a sociedade é diretamente beneficiada com a tecnologia. Diante disso, surge o seguinte questionamento: de que maneira podemos usar a tecnologia para melhorar a prevenção contra incêndios residências e evitar grandes desastres causados por vazamento de gás e negligência dos residentes, principalmente quando em situações em que o morador precisa se ausentar da residência, seja por questões pessoais ou de trabalho?

Atualmente é muito comum ouvir falar no conceito de internet das coisas, mais conhecida pela sigla IoT (internet of things), devido a grande quantidade de dispositivos eletrônicos que fazem parte do dia a dia das pessoas. IoT é composto por objetos que fazem parte do cotidiano, que podem ser acessados por dispositivos móveis de forma remota, por meio da internet, como por exemplo a comunicação entre um eletrodoméstico e um aparelho celular. (SACOMANO et al., 2018, p. 58).

Nesse sentido, no presente trabalho, será apresentado o desenvolvimento de uma aplicação web em linguagem Java com módulos do Spring Framework, juntamente com a placa de desenvolvimento NodeMCU, utilizando o sensor digital de temperatura DS18B20, sensor de gás inflamável MQ-2, e um atuador Buzzer, que basicamente é um pequeno alto-falante que tem a função de emitir alertas sonoros. A aplicação tem a finalidade de monitorar e alertar o usuário em caso de detecção de gás GLP, fumaça no ar e altas temperaturas na residência. Mais especificamente, o usuário da aplicação precisa efetuar um login para poder acompanhar de forma remota a situação do ambiente, e em casos de detecção será emitido

um alerta sonoro no local monitorado, como também será enviado uma notificação por mensagem de WhatsApp, com aviso de detecção. Além disso, todos os alertas serão registrados em um banco de dados, para futuras consultas sobre o histórico de alertas.

2 REFERENCIAL TEORICO

Neste capítulo serão apresentados os conceitos teóricos, bem como os materiais que serão utilizados para o desenvolvimento do projeto.

2.1 Aplicações *web*

Com os avanços tecnológicos, os serviços na internet se tornaram indispensáveis. As aplicações web e os aplicativos móveis englobam uma grande variedade de softwares projetados para serem utilizados através das redes de computadores. Como os navegadores de internet, aplicativos móveis, os diversos softwares que são distribuídos e comercializados e todo os recursos fornecido pela computação em nuvem. (PRESSMAN e MAXIM, 2021, p. 6).

As aplicações, ou sistemas web, recebem esse nome porque são sistemas hospedados em servidores *web*, que podem ser acessadas pelos usuários através de um browser (ou navegador) de internet. Isso ocorre por efeito do modelo arquitetural cliente servidor, que significa que máquinas, chamadas clientes, solicitam serviços ou recursos que são fornecidos por computadores chamados de servidores.

De acordo com Alves (2015, p. 27) a comunicação entre o cliente e o servidor web ocorre por meio de um protocolo de aplicação chamado HTTP (Hyper Text Transfer Protocol), que significa Protocolo de Transferência de Hipertexto. O processo de comunicação ocorre da seguinte forma: o cliente efetua uma conexão (connection) com o servidor, caso seja estabelecida com sucesso, o mesmo solicita recursos para o servidor por meio de requisição (request), e recebe do servidor uma resposta (response). No final de tudo ocorre o fechamento (close) da conexão, quando o cliente é atendido e não solicita mais outro recurso. É dessa forma que funciona quando acessamos qualquer site na internet.

A transferência das informações através deste protocolo, na grande maioria das vezes, ocorre pela porta 80, mas também é possível utilizar outras portas. Este protocolo permaneceu por muito tempo e ainda é possível encontrá-lo em algumas aplicações na internet. Nos navegadores atuais as aplicações web que usam o HTTP são identificadas como não seguras. Isto porque o protocolo HTTP possui algumas vulnerabilidades durante a transferência de dados entre o servidor e o cliente.

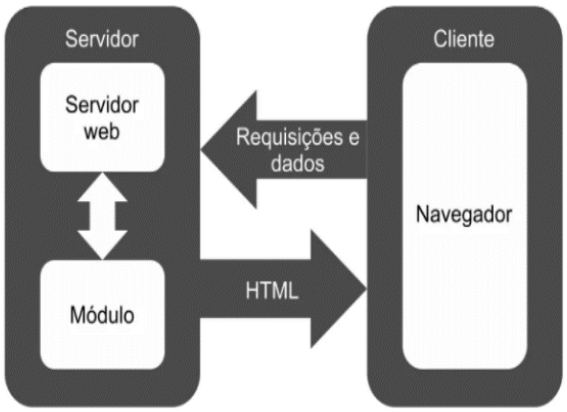
Com o passar do tempo, surgiu a necessidade de garantir uma maior segurança das informações que trafegam na internet. Diante disso, surgiu uma nova versão do protocolo HTTP, chamado de HTTPS (Hypertext Transfer Protocol Secure). Ele possui a mesma funcionalidade do HTTP, a principal diferente entre ambos, está na implementação de uma camada adicional de segurança, representada pela sigla "S" de "Segurança" presente na sigla do protocolo. (LAUREANO e CORDELLI, 2014, P. 22).

Quando acessamos qualquer site na internet que possui um cadeado verde visível na URL (Uniform Resource Locator) ou Localizador Padrão de Recursos, significa que aquele site possui um tipo de segurança digital. Podendo ser o TLS (Transport Layer Security), ou o seu antecessor SSL (Secure Sockets Layer) que são os protocolos de segurança presente no HTTPS. Com isso, os dados são transmitidos por meio de uma conexão criptografada

garantindo alguns pilares da segurança da informação, como a integridade, confidencialidade e autenticação.

Como é possível observar na Figura 1, durante o processo de requisição o cliente solicita um serviço para o servidor, e é retornado um documento HTML que será processado pelo navegador.

Figura 1 – Estrutura básica de uma aplicação *web*.



Fonte: Oliveira e Zanetti (2020, p.44).

Conforme Oliveira e Zanetti (2020, p.44) a linguagem HTML é indispensável para a criação de qualquer página na internet. Contudo, é uma linguagem que dispõe somente de elementos estáticos para a composição e formatação de documentos para a *web*. Com a limitação de conteúdos estáticos na internet, houve a necessidade de criar páginas dinâmicas a fim de proporcionar interação com os usuários. Nesse cenário, surgem as linguagens de programação para servidores, como JAVA, PYTHON, PHP entre outras.

De acordo com Aws (c2022), as aplicações *web* modernas são divididas em duas partes fundamentais para sua construção, o *front-end* e o *back-end*. O *front-end* consiste nas tecnologias voltadas para a interação do usuário com a aplicação, e o *back-end* é responsável pelas regras de negócios da aplicação, ou seja, se preocupa com todas as ações que serão realizadas pelos usuários.

2.2 Jquery E Ajax

Como descrito no capítulo anterior, as aplicações e sites fazem requisições para o servidor *web*, ele processa a informação e devolve uma resposta. Contudo, durante esse processo de requisição e resposta, a página inteira é recarregada. Ou seja, toda vez que for solicitado um serviço em um site, a página inteira será renderizada pelo navegador, independentemente do serviço solicitado. Isso implica no processo de comunicação com o servidor, bem como na experiencia do usuário ao solicitar um serviço em uma aplicação da *web*.

Diante disso, podemos utilizar uma combinação de tecnologias com o intuito de realizar chamadas assíncronas para garantir que o fluxo do código não seja interrompido até que a resposta seja obtida. Para tal, utilizaremos uma das principais bibliotecas *JavaScript*, o *jQuery*, juntamente com o conceito de Ajax, que significa *Asynchronous JavaScript and XML*, ou JavaScript e XML Assíncronos. O *jQuery* tem como objetivo tornar as aplicações *web*

mais dinâmicas, como também, facilitar o desenvolvimento de códigos em *JavaScript*, e consequentemente proporcionar uma melhor experiência para o usuário. (SILVA, 2013, p. 28).

Quando combinamos jQuery com Ajax, surge poderosos recursos com o intuito de melhorar o desempenho das aplicações *web*. Com a junção dessas tecnologias será possível realizar requisições para salvar as altas temperaturas e os alertas em um banco de dados da aplicação, sem que haja interrupção do monitoramento.

2.4 Bootstrap

Os frameworks facilitam e automatizam o processo de desenvolvimento de softwares. Bootstrap é um framework para o desenvolvimento de interfaces *web*, que faz uso de elementos HTML, estruturas CSS e JavaScript. Permitindo a criação de interfaces responsivas, com a capacidade de adequação para computadores e dispositivos móveis. (OLIVEIRA e ZANETTI, 2020, p.100).

O Bootstrap possui um kit de ferramentas que atuam no front-end (camada de visualização) da aplicação. É um framework de código aberto (*open-source*), ou seja, seu código-fonte é disponibilizado e licenciado para que qualquer pessoa com conhecimentos de programação, modifique e use para qualquer finalidade.

2.5 Microcontrolador

Os microcontroladores estão presentes em várias invenções tecnológicas do mercado atual. No meio acadêmico é possível encontrar microcontroladores em várias placas de prototipagem para desenvolvimento de projetos de automação e Internet das Coisas (IoT). Microcontroladores são pequenos circuitos eletrônicos, e assim como os computadores, eles são compostos por uma unidade de processamento, memória e alguns periféricos de entrada e saída. (ALMEIDA et al., 2016, p. 177).

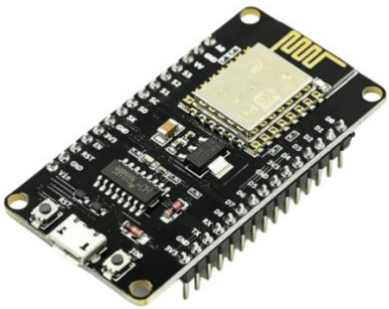
Além de serem pequenos, possuem um ótimo custo/benefício. Outra característica que o torna muito interessante para construir circuitos eletrônicos é o seu baixo consumo de energia. São muito importantes para a criação de eletrodomésticos, Drones, ferramentas elétricas e vários dispositivos inteligentes.

Devido aos avanços e as facilidades tecnológicas, o uso de placas de prototipagem cresceu bastante em ambientes domésticos. Atualmente é possível encontrar vários tipos de placas de desenvolvimento, todas com o objetivo de construir dispositivos eletrônicos, porém cada uma com suas características de adequação às necessidades dos usuários. Todos os modelos de placas são facilmente encontrados em sites e em lojas físicas de eletrônica. Para o desenvolvimento do projeto foi utilizado a placa NodeMCU V3, que nos permite a transferência de informações através do Wi-fi.

2.5.1 Placa NodeMCU V3

O NodeMCU, visto na Figura 2, utiliza um microcontrolador ESP8266, Wi-Fi integrado e um regulador de tensão 3,3V. Assim como o Arduino Uno, o NodeMCU possui uma interface USB-Serial, como também faz o uso do mesmo software de IDE (Integrated Development Environment) ou Ambiente de Desenvolvimento Integrado.

Figura 2 – Módulo NodeMCU.



Fonte: disponível em: <https://www.casadarobotica.com/internet-das-coisas/placas/esp/placa-esp-nodemcu-v3-wifi-802-11-b-g-n>. Acesso em 07 de mai. de 2022.

Monk (2017, p. 146) ressalta que embora o NodeMCU tenha algumas semelhanças e utiliza a mesma linguagem de programação e ambiente de desenvolvimento do Arduino, as placas que utilizam o ESP8266 não são consideradas placas oficiais da família Arduino. Além disso, as placas com o ESP8266 que são caracterizadas por permitir a implementação da conexão Wi-fi em projetos de IoT, também possui a capacidade de atender grande parte dos projetos que são desenvolvidos com Arduino.

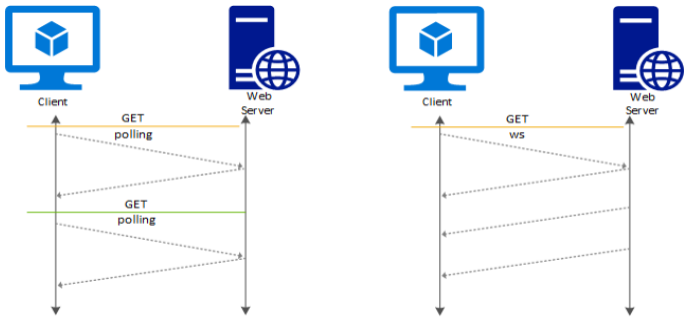
Senso assim, para carregar o código desenvolvido para a placa é preciso um cabo USB para conectá-la a uma porta do computador, selecionar corretamente a porta na IDE e realizar o processo de envio do código. Por se tratar de um dispositivo que precisa enviar as informações das leituras dos sensores para uma aplicação na *web*, de forma simultânea, será utilizado o protocolo WebSocket.

2.6 WebSocket

Conforme citado em capítulos anteriores, o protocolo padrão para comunicação entre uma aplicação *web* cliente e um servidor é o HTTP. Como cada requisição desse protocolo possuí um atraso, devido a conexão ser fechada após a resposta do servidor, esse tipo de comunicação acaba sendo um grande problema para aplicações *web* que precisam requisitar ao servidor informações de forma contínua, como é o caso da aplicação *web* proposta pelo presente trabalho, que necessita de uma comunicação em tempo real. E mesmo com tecnologias como *jQuery* e *Ajax* que podem realizam chamadas assíncronas a cada segundo, ainda assim, é ineficiente e sobrecarrega o servidor. O *WebSocket* veio para resolver esse problema de comunicação. Ele faz parte das especificações do HTML5, e é suportado pelas últimas versões dos navegadores. *WebSocket* possui as regras necessárias para se estabelecer uma comunicação bidirecional entre aplicações cliente e servidores, através da conexão full duplex. (IBM, 2021).

Na Figura 3 está apresentado a comparação do fluxo de informações de um navegador cliente para um servidor *web* utilizando HTTP e WebSocket. No primeiro fluxo que é em HTTP, o cliente precisa realizar uma nova requisição do tipo GET (Método HTTP) para o servidor sempre que ele necessita de uma informação. Já no segundo fluxo WebSocket, após iniciado a conexão, as informações são enviadas imediatamente quando estão disponíveis.

Figura 3 – HTTP vs WebSocket



Fonte: disponível em: <https://docs.microsoft.com/en-us/azure/application-gateway/application-gateway-websocket>. Acesso em 01 de mai. de 2022.

Assim sendo, podemos concluir que a velocidade com que as informações trafegam no protocolo *WebSocket* é muito superior ao do protocolo HTTP, pois nesta comunicação não é necessário a criação de uma nova conexão cada vez que se quer obter uma nova informação. Ele é muito utilizado para a criação de *chats* e jogos online, que são aplicações que demandam informações bilateral em tempo real.

2.7 Sensores e atuadores

Existe atualmente no mercado, diversos tipos de sensores para uso na indústria. Esse incrível dispositivo eletrônico, tem provado ser fundamental no processo de transformação das atividades manuais para automáticas.

Junior e Silva (2015, p.24) define sensor como um aparelho que possui a capacidade de detectar as grandezas físicas, e quando essas grandezas são relacionadas entre si, diz-se que o sensor é um transdutor. Em outras palavras, o sensor é responsável por monitorar as grandezas físicas, que pode ser temperatura, força, pressão entre outras. E é caracterizado como transdutor, quando o mesmo for utilizado para transformar essas grandezas em informações compreensíveis por outros dispositivos.

Conforme Thomazini e Albuquerque (2020, p. 11), os atuadores são dispositivos com a capacidade de manipular variáveis de controle. Eles reagem no sistema de acordo com sinais vindo de um controlador. Como exemplo de atuadores podemos citar os motores e relés. Quando sensores e atuadores são conectados a placas de desenvolvimento, surge uma infinidade de possibilidades. No desenvolvimento deste projeto, é utilizado dois sensores, o de gás inflamável e fumaça MQ-2, e o sensor digital de temperatura DS18B20. Como também um atuador *Buzzer*, que basicamente é um pequeno alto-falante que terá a função de emitir alertas sonoros.

2.7.1 Sensor de gás inflamável e fumaça MQ-2

O sensor de gás MQ-2, visto na Figura 4, é um dos dispositivos mais importantes deste trabalho, uma vez que ele é o responsável por detectar a concentração de gases inflamáveis (GLP, gás natural, metano, propano, butano, hidrogênio, álcool e outros inflamáveis) como também fumaça no ar, e transmitir sinais a placa NodeMCU.

Figura 4 – Sensor de gás inflamável MQ-2.

Fonte: disponível em: <https://www.filipeflop.com/produto/sensor-de-gas-mq-2-inflamavel-e-fumaca/>. Acesso em 07 de mai. de 2022.

O MQ-2 possui baixo custo, e é muito utilizado em projetos de automação residencial com Arduino. Como é possível observar na Figura 4, ele possui somente 4 pinos, e é considerado um módulo muito simples de usar.

Conforme consta no datasheet (2018, tradução nossa), as principais especificações técnicas do MQ-2 são as seguintes:

Dados técnicos:

- É um sensor do tipo semicondutor;
- Concentração de detecção: 300 a 10.000 ppm;
- Tensão de operação: 5V;
- Sensibilidade ajustável via potenciômetro;
- Saída Digital e Analógica;
- Dimensões: 32 x 20 x 15mm;

Pinagem:

- VCC: 5V;
- GND: GND;
- Do: Saída Digital;
- Ao: Saída Analógica;

Como pode-se observar nas especificações, o MQ-2 assim como grande parte dos sensores que são utilizados com Arduino, operam a uma tensão de 5V, que é a mesma tensão em que muitas placas de desenvolvimento fornecem em alguns pinos para alimentação de módulos e circuitos externos.

Conforme Junior e Farinello (2019, p. 36), a melhor localização para instalação do sensor MQ-2 em uma residência, é nas áreas que possuem os principais causadores de incêndio, como por exemplo a cozinha, que possui botijão ou tubulação de gás, como também vários eletrodomésticos que podem resultar em incêndios.

2.7.2 Sensor de temperatura DS18B20

Existem diversos sensores analógicos e digitais para medição de temperatura para projetos residenciais de automação. O sensor utilizado para o presente projeto é o sensor digital DS18B20, da Dallas Semiconductor (Maxim). Este possui ótima precisão de medição, custo-benefício e é facilmente encontrado em lojas de eletrônica. Existe três modelos deste sensor no mercado, são eles: o tipo sonda, To92 e versão SMD. Está sendo utilizado no projeto o tipo To92 em módulo para Arduino, visto na Figura 5.

Figura 5 – Módulo sensor de temperatura DS18B20.

Fonte: disponível em: <https://www.baudaeletronica.com.br/modulo-sensor-de-temperatura-18b20.html>. Acesso em 11 de mai. de 2022.

A comunicação entre o NodeMCU e o sensor ocorre por um barramento 1-Wire (1 Fio). Isto é, com o sensor alimentado com a tensão necessária no VCC (Tensão em corrente contínua), só é preciso um único pino da placa para o sensor funcionar. Além do mais, com esse mesmo esquema de ligação é possível ligar até 255 sensores do mesmo tipo. (MONK, 2015, p. 136)

O DS18B20, além de detectar a temperatura atual do ambiente, irá desempenhar um papel muito importante no sistema. Pois em situações em que há chamas, se porventura, o MQ-2 falhar ou apresentar algum defeito, a partir da elevação da temperatura podemos garantir que a aplicação *web* consiga enviar notificações ao usuário. A escolha do sensor foi baseada em sua ótima precisão de 0,5ºC e capacidade de medir valores na faixa de -55ºC a +125ºC.

2.8 Plataforma *Twilio*

Twilio é uma plataforma que fornece soluções para comunicação em nuvem. Possui várias ferramentas de comunicação programável para o envio e recebimento de SMS, e-mail, voz, WhatsApp entre outros. As soluções da *Twilio* são pagas e o preço é calculado com base na utilização dos serviços, ou seja, o cliente só paga pelo que usa. (TWILIO, c2022).

Para o presente trabalho foi utilizado o serviço de API (Application Programming Interface), ou interface de programação de aplicações, do WhatsApp. Esta API tem a função de permitir, com poucas linhas de código, a integração da aplicação *web* com o WhatsApp. Uma vez que precisamos notificar o usuário a distância com alertas de incêndio. Optou-se por utilizar o aplicativo WhatsApp para o recebimento das mensagens de alerta, justamente por ele ser o mensageiro mais utilizado no mundo.

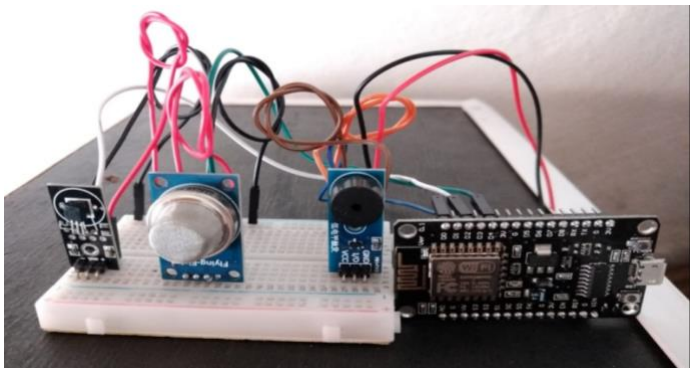
3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo, serão apresentados os passos para a montagem do circuito, implementação do protocolo WebSocket, desenvolvimento da aplicação web, bem como sua interface gráfica e integração com o WhatsApp.

3.1 Montagem do circuito e implementação do protocolo WebSocket

Para a montagem do circuito foi utilizado uma protoboard 400 pontos, com alguns jumpers para conexão entre os componentes e a placa controladora. Foi utilizado os pinos digitais D0, D2 e D4 de GPIO (General Purpose Input/Output), entrada e saída de dados da placa. Como citado em capítulos anteriores, grande parte dos sensores para placas de prototipagem operam com uma tensão de 5V. Por tanto, para alimentação de todo o circuito foi utilizado o GND (Ground) ou aterramento, e pino VIN do NodeMCU. O pino VIN fornece uma tensão de 5V para alimentação externa de periféricos. A Figura 6 mostra o circuito montado antes de ir para o ambiente de testes.

Figura 6 – Circuito completo.



Fonte: O autor (2022)

Conforme abordado em capítulos anteriores, a comunicação entre o NodeMCU e a aplicação *web* será por meio do protocolo *WebSocket*. Para o cliente *WebSocket* foi utilizado o *JavaScript*, devido a sua dinamicidade. Uma vez que para haver comunicação utilizando esse protocolo, é preciso um cliente para abrir uma conexão com o servidor. O NodeMCU serviu de servidor *WebSocket* para a aplicação *web*. Para isso, foi utilizado o ambiente de desenvolvimento do Arduino para programar a placa. Como também, foi realizado a instalação de algumas bibliotecas para a utilização dos sensores, servidor *WebSocket*, comunicação Wi-fi e manipulação de arquivos no formato JSON (JavaScript Object Notation). Utilizou-se como base para o desenvolvimento do código de conexão Wi-fi e construção do servidor *WebSocket*, exemplos disponibilizados pela própria IDE. As informações de leitura dos sensores são processadas pelo NodeMCU, atuando como servidor *WebSocket*, que por sua vez enviará para o cliente *WebSocket* em formato JSON.

3.2 Estrutura e interface gráfica da aplicação *web* desenvolvida

A aplicação foi desenvolvida em linguagem de programação Java da Oracle, juntamente com módulos do Spring, um framework Java de código aberto. O Spring é constituído por vários módulos/projetos prontos para uso, com o intuito de auxiliar no desenvolvimento de aplicações Java. Os projetos que serão utilizados neste trabalho são:

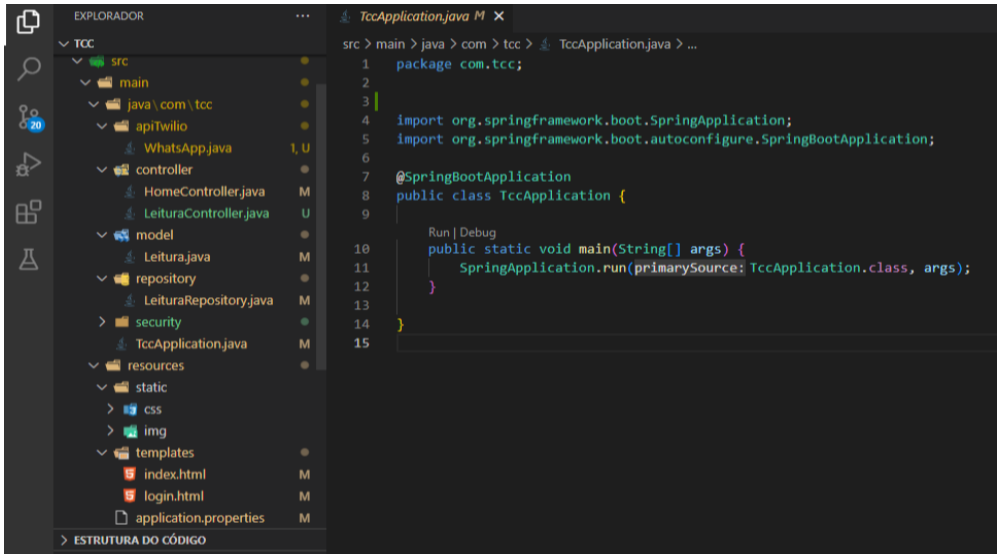
- Spring Boot;
- Spring Data;
- Spring Security;

Inicialmente, foi adicionado as dependências de cada módulo em um arquivo de configuração em formato XML (Extensible Markup Language). Este arquivo fica na raiz do projeto e possui o nome de POM (Project Object Model). Ele é fundamental para a construção de projetos que utilizam o gerenciador de dependência, Maven.

O Spring Boot nos permite ter um projeto mais organizado, uma vez que ele cuida de toda infraestrutura e configurações baseadas em convenções que ele adota. Além disso, possui servidores embarcados, por padrão, vem configurado com o Tomcat, servidor *web* Java. Baseando-se nos modelos de aplicações tradicionais, foi utilizado conceitos básicos para a construção de uma API REST com Spring Boot. Através desse modelo de arquitetura, podemos utilizar melhor os recursos dos métodos HTTP. Como também, a manipulação das mensagens em formato JSON vindas do servidor *WebSocket*.

Como é possível observar na Figura 7, as classes Java estão organizadas em pacotes. Dessa forma, temos um código legível, limpo e de fácil manutenção. Para isso, foi adotado algumas boas práticas de programação. Já os arquivos HTML, CSS, *JavaScript* e imagens que formam a interface da aplicação, ficam localizados no diretório *resource*.

Figura 7 – Estrutura da aplicação *web*.



Fonte: O autor (2022)

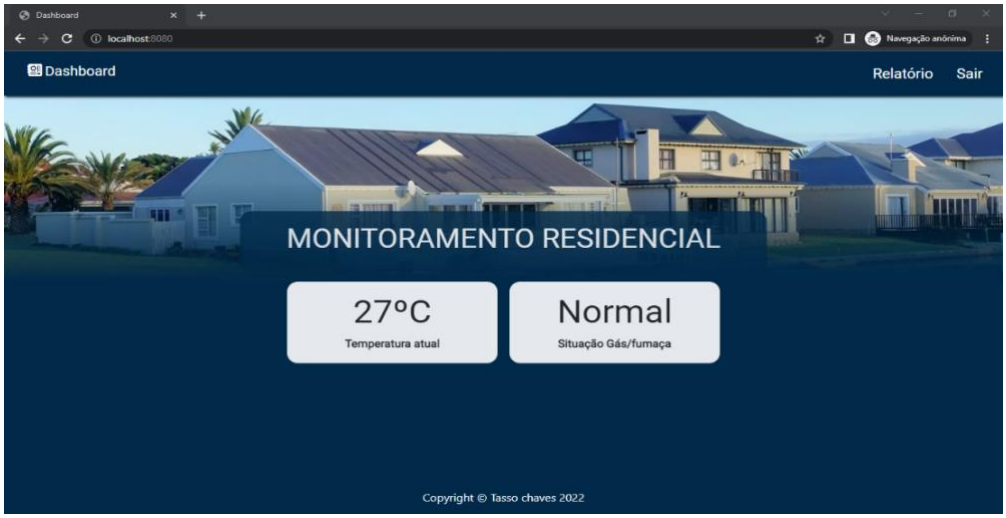
Para se chegar à interface gráfica pretendida, foi utilizado o framework *Bootstrap*, uma vez que já existe vários componentes prontos para uso, como botões, cards, tabelas, entre outros. Como qualquer aplicação *web*, por padrão a primeira tela da aplicação é a de login, visto na Figura 8.

Figura 8 – Login da aplicação *web*.

Fonte: O autor (2022)

Inicialmente, foi pré configurado dois usuários e suas respectivas senhas para acessar o *dashboard* da aplicação. Para criar os usuários e permitir que somente estes tenham acesso ao *dashboard*, foi utilizado recursos de autenticação e autorização do Spring Security. Passando da etapa de login, o usuário será direcionado para o dashboard, onde será possível acompanhar em tempo real, a temperatura atual do ambiente e situação de gás inflamável e fumaça, como é apresentado na Figura 9.

Figura 9 – Dashboard da aplicação *web*.



Fonte: O autor (2022)

Além do monitoramento, a aplicação salva automaticamente, com base em condições, as altas temperaturas e a detecção de gás e fumaça em um banco de dados. Este processo de registro, ocorre através de requisições Ajax com jQuery via método POST (método HTTP que envia dados no corpo da requisição). Para facilitar a comunicação com o SGBD (Sistema Gerenciador de Banco de Dados), foi utilizado o Spring Data Jpa, um subprojeto do Spring Data. Com poucas configurações ele nos permite iniciar uma conexão com uma base de dados. Optou-se por utilizar neste projeto o banco de dados relacional feito em Java, o H2.

Outro recurso interessante da aplicação está no botão de relatório, presente na barra de navegação, visto na Figura 9. Ao clicar em relatório, é exibido uma janela modal com seleção de data para buscar o histórico dos dados com as altas temperaturas e detecções de

gás e fumaça que foram salvas no banco de dados, com data e horário de registro. Para que esses dados sejam mostrados na janela, é realizado uma requisição Ajax com jQuery via método GET (método HTTP que passa parâmetros no cabeçalho da requisição).

3.3 Integração da aplicação *web* com WhatsApp

Para ter acesso aos serviços de mensagem do WhatsApp, foi preciso criar um conta gratuita na Twilio. Com está conta podemos usar vários recursos gratuitos para fins de desenvolvimento e testes. Durante o cadastro é configurado o número de WhatsApp que receberá as mensagens. A própria plataforma disponibiliza toda a documentação necessária para os desenvolvedores realizarem a integração das APIs com as aplicações *web*.

Após o desenvolvimento do código foi preciso ativar o recebimento de mensagens. Para isso, foi necessário iniciar uma sessão enviando um comando para um número de WhatsApp disponibilizado pela plataforma. Este serviço de envio de mensagem é invocado automaticamente quando ocorre a detecção e a elevação da temperatura.

4 TESTES E RESULTADOS

Após a montagem do circuito com a placa NodeMCU, sensor digital de temperatura DS18B20, sensor de gás inflamável MQ-2, atuador Buzzer e o desenvolvimento da aplicação *web*, foi possível realizar os testes em um ambiente de simulação. Foi utilizado uma maquete de uma residência e um isqueiro, uma vez que o sensor utilizado é capaz de detectar vários tipos de gases, como o GLP, gás natural, metano, propano, butano, hidrogênio, álcool entre outros. Dentre eles está presente o butano, que é o gás mais utilizado em isqueiros.

Na Figura 10 é possível observar o ambiente preparado para a realização dos testes.

Figura 10 – Ambiente utilizado para teste.

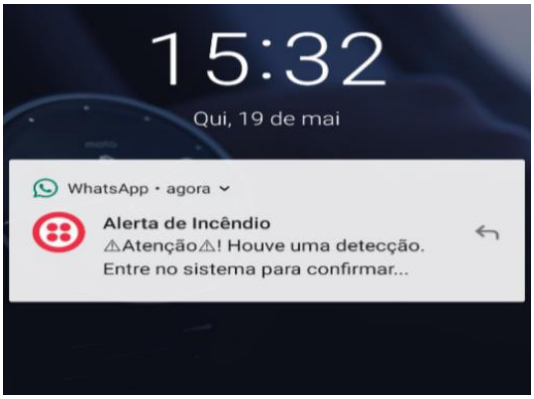
Fonte: O autor (2022)

Para a alimentação da placa NodeMCU foi utilizado um carregador USB 5V de celular. Foram feitos testes com os dois sensores, para verificar as duas situações de alerta. Como o sensor MQ-2 possui duas saídas, uma analógica (AOUT) e uma digital (DOUT), optou-se por utilizar no projeto a saída digital. Com isso, a informação enviada terá apenas dois estados, verdadeiro ou falso. Ou seja, quando detectado envia alertas, caso contrário a aplicação continua no monitoramento.

Antes de tudo, foi necessário ajustar a sensibilidade do sensor MQ-2 para detectar o gás mais facilmente. No primeiro teste foi liberado somente o gás do isqueiro, resultando no

acionamento do atuador *buzzer*, que emitiu um alerta sonoro e com alguns milissegundos é recebido uma notificação no WhatsApp cadastrado na Twilio, como visto na Figura 11.

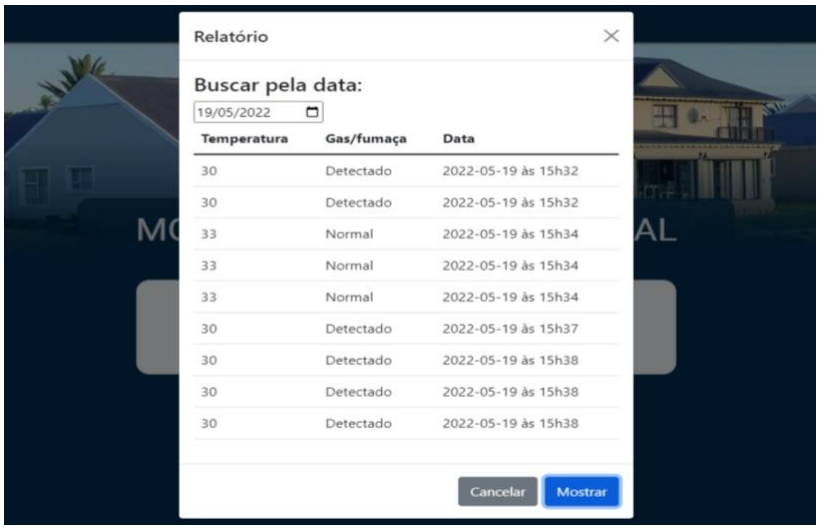
Figura 11 – Notificação de alerta via WhatsApp.



Fonte: O autor (2022)

A mensagem tem o objetivo de informar que houve uma detecção, e ao mesmo tempo sugeri que o usuário confirme entrando no sistema, em casos de dúvidas. Em seguida foi realizado o teste de elevação de temperatura. Levando em consideração a temperatura ambiente da maquete que estava a 30º C, foi programado para alertas de altas temperaturas valores a cima de 32ºC. Para a realização deste, o sensor foi pressionado pela mão de uma pessoa. Visto que a temperatura normal do corpo humano está acima da temperatura atual do local de teste. A temperatura alcançada foi de 33ºC, valor acima da temperatura de alerta. E por fim, foi realizado um teste com a fumaça de solda estanho, que também foi bem sucedido. Na figura 12 é possível observar os resultados obtidos, que ficam disponíveis no relatório da aplicação.

Figura 12 – Resultado dos testes



Fonte: O autor (2022)

CONSIDERAÇÕES FINAIS

Pode-se concluir que a aplicação *web* em conjunto com o circuito construído, atingiu as expectativas e mostrou ser eficiente no monitoramento e alerta de forma autônoma, em casos de detecção de gás, fumaça e altas temperaturas. A notificação do usuário por meio de mensagem de WhatsApp é satisfatória para o cenário, uma vez que ele é o mensageiro multiplataforma mais utilizado no mundo. Assim como o alerta sonoro que é muito útil quando houver pessoas no local monitorado. Desse modo, a residência se torna mais segura com esse tipo de prevenção contra incêndio, de tal modo que o residente ao estar despercebido em casa ou quando estiver longe, possa ser alertado para agir rapidamente. Como por exemplo, acionar o corpo de bombeiros de onde estiver, ou até mesmo os vizinhos ou familiares com localidade próxima. Outro recurso interessante que podemos destacar, o fato de a aplicação registrar todos os alertas em um banco de dados, e conceder ao usuário um relatório com o histórico de alertas de cada mês. Vale ressaltar, que ter um sistema com tal capacidade, não significa eliminar do usuário todos os cuidados de prevenção que deve ser seguido durante o dia a dia. Como por exemplo, antes de sair de casa verificar se não tem algum vazamento ou eletrodoméstico ligado sem necessidade, entre outros cuidados. Como ideias para um projeto futuro, destaca-se o envio de alertas para o corpo de bombeiros, como também, a instalação de uma câmera em um ponto estratégico, para envio de fotos em pequenos intervalos de tempo durante os alertas.

REFERÊNCIAS

ALMEIDA, Rodrigo M. A; MORAES, Carlos H. V; SERAPHIM, Thatyana F. P. **Programação de Sistemas Embarcados: Desenvolvendo Software para Microcontroladores em Linguagem C**. 1. ed. Rio de Janeiro: GEN LTC, 2016.

ALVES, William Pereira. **Projetos de Sistemas Web: Conceitos, Estruturas, Criação de Banco de dados e Ferramentas de Desenvolvimento.** 1. ed. São Paulo: Érica, 2015.

AWS. **Criação de um aplicativo Web simples.** Disponível em: < https://aws.amazon.com/pt/startups/start-building/how-to-build-a-web-app/>. Acesso em: 10 de jan. de 2022.

GAUTO, Marcelo Antunes. **Petróleo e Gás: princípios de exploração, produção e refino**. Porto Alegre: Bookman, 2016.

IBM. **WebSocket.** Disponível em: <https://www.ibm.com/docs/pt-br/was-liberty/base?topic=SSEQTP_liberty/com.ibm.websphere.wlp.nd.multiplatform.doc/ae/cwlp_websockets.html>. Acesso em: 6 mai. 2021.

JUNIOR, Sergio Luiz Stevan; SILVA, Rodrigo Adamshuk. **Automação e Instrumentação Industrial com Arduino: Teoria e Projetos.** 1. ed. São Paulo: Érica, 2015.

JUNIOR, Sergio Luiz Stevan; FARINELLI, Felipe Adalberto. **DOMÓTICA: Automação residencial e casas inteligentes com Arduino e ESP8266.** 1. ed. São Paulo: Érica, 2019.

LAUREANO, Marcos Aurelio Pcheck; CORDELLI, Rosa Lantmann; **Fundamentos de software: desempenho de sistemas computacionais**. 1. ed. São Paulo: Érica, 2014.

MONK, Simon; **Programação com Arduino II: começando com sketches.** 2. ed. Porto Alegre: Bookman, 2017.

OLIVEIRA, Cláudio Luís Vieira; ZANETTI, Humberto Augusto Piovesana. **JavaScript descomplicado: Programação para a Web, IOT e Dispositivos Móveis**. São Paulo: Érica, 2020.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de Software: Uma abordagem Profissional**. 9. ed. Porto Alegre: AMGH, 2021.

SACOMANO, José Benedito, et al. **Indústria 4.0: conceitos e fundamentos.** 1. ed. São Paulo: Blucher, 2018.

SILVA, Maurício Samy. **jQuery. A Biblioteca do Programador Javascript.** 3. ed. [S.I]. Novatec, 2013.

THOMAZINI, Daniel; ALBUQUERQUE, Pedro Urbano Braga. **Sensores industriais: fundamentos e aplicações.** 9. ed. São Paulo: Érica, 2020.

TWILIO. **Plataforma de Engajamento com Clientes da Twilio**. Disponível em: < https://www.twilio.com/pt-br/platform> Acesso em: 08 de mai. de 2022.