UNIVERSIDADE FEDERAL DO ABC

Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Programa de Graduação em Engenharia de Informação

SISTEMA DE MONITORAMENTO PLUVIAL URBANO

Carlos Augusto Duru Pacheco

IOT e Sistemas Ubíquos

Santo André, Setembro de 2024

Carlos Augusto Duru Pacheco

SISTEMA DE MONITORAMENTO PLUVIAL URBANO

Monografia apresentada ao Curso de Tecnologias e Sistemas de Informação na Universidade Federal do ABC, como requisito básico para obtenção do Grau de Especialista em Tecnologias e Sistemas da Informação.

Orientador (a): Prof. Dr Mario Gazziro

Santo André - SP
2024

Sistema de Bibliotecas da Universidade Federal do ABC

Elaborada pelo Sistema de Geração de Ficha Catalográfica da
UFABC com os dados fornecidos pelo(a) autor(a).

PACHECO, Carlos Augusto Duru

Sistema de Monitoramento Urbano / Carlos Augusto Duru Pacheco — 2024.

43 fls: il.

Orientador: Mario Alexandre Gazziro

Trabalho de Conclusão de Curso — Universidade Federal do ABC, Especialização em Tecnologia e Sistemas da Informação, Santo André, 2024.

1.Enchentes. 2.Simulação 3. Microcontroladores 4. Web Servers 5. AWS I. Gazziro, Mario Alexandre. II. Especialização em Tecnologia e Sistemas da Informação, 2024. III. Título.

Carlos Augusto Duru Pacheco

SISTEMA DE MONITORAMENTO PLUVIAL URBANO

Assinatura do Orientador

_____

Mario Alexandre Gazziro
Orientador

Santo André - SP
2024

AGRADECIMENTOS

Gostaria de expressar minha gratidão ao meu orientador, Mario Gazziro, por seu apoio e orientação ao longo deste trabalho. Sua disponibilidade, suporte com ferramentas experimentais e auxílio nos acessos aos recursos da Amazon Web Services foram essenciais para o desenvolvimento do presente projeto.

Agradeço também aos demais professores do curso de Tecnologia e Sistemas da Informação. Cada um de vocês contribuiu significativamente para minha experiência acadêmica ao longo dos últimos dois anos e sou grato por todo o conhecimento que me proporcionaram.

“A melhor maneira de escapar de uma caixa é
inventando seu próprio caminho para fora.” (Jeff Bezos).

RESUMO

As mudanças climáticas têm se intensificado nos últimos anos, e a falta de um planejamento adequado nos perímetros urbanos agrava ainda mais os riscos de inundações. Diante desse contexto, é crucial desenvolver maneiras mais eficazes de avisar a população sobre o perigo de enchentes e, ao mesmo tempo, encontrar formas de reduzir ao máximo os impactos negativos que esses eventos causam. Para isso, foi elaborado neste estudo um protótipo de Sistema de Monitoramento Pluvial Urbano (SMPU) que integra hardware, software e soluções Cloud para identificar situações de alagamento, alertando a comunidade da região afetada o mais rápido possível. O sistema utiliza sensores conectados a um microcontrolador Arduino, que processa os dados em tempo real, ativando alarmes sonoros e visuais no local, ao mesmo tempo que envia os dados para um servidor Web público. Na nuvem, o armazenamento e a análise dos dados são realizados por meio de serviços da Amazon Web Services (AWS), garantindo a segurança das informações e o envio remoto de notificações automáticas por e-mail. O presente trabalho oferece uma solução alternativa que ajude as pessoas, em caso de enchente, a tomarem decisões rápidas para proteger sua segurança e seus pertences.

Palavras-chave: Enchentes; Simulação; Microcontroladores; Web Servers; AWS.

ABSTRACT

Climate change has intensified in recent years, and the lack of adequate planning in urban areas further aggravates the risk of flooding. Given this context, it is crucial to develop more effective ways to warn the population about the danger of flooding and, at the same time, find ways to minimize the negative impacts that these events cause. To this end, this study developed a prototype of an Urban Rainfall Monitoring System (SMPU) that integrates hardware, software and Cloud solutions to identify flooding situations, alerting the community in the affected region as quickly as possible. The system uses sensors connected to an Arduino microcontroller, which processes the data in real time, activating audible and visual alarms on site, while sending the data to a public Web server. In the cloud, data storage and analysis are performed through Amazon Web Services (AWS) services, ensuring information security and the remote sending of automatic notifications by email. This work offers an alternative solution that helps people, in the event of a flood, to make quick decisions to protect their safety and belongings.

Keywords: Floods; Simulation; Microcontrollers; Web Servers; AWS.

LISTA DE SIGLAS

S3 - Simple Storage Service

APIPA - Endereçamento IP Privado Automático

AWS - Amazon Web Services

CGE - Centro de Gerenciamento de Emergências

E2E - End-to-End

EC2 - Elastic Compute Cloud

HTML - Linguagem de Marcação de Hipertexto

IPCC - Intergovernmental Panel on Climate Change

IAM - Identity and Access Management

IP - Protocolo de Internet

IVCM - Índice de Vulnerabilidade Climática dos Municípios

PHP - Preprocessador de Hipertexto

NoSQL - Not Only SQL

UFABC - Universidade Federal do ABC

URL - Localizador Uniforme de Recursos

UN-ISDR - United Nations International Strategy for Disaster Reduction

SMPU - Sistema de Monitoramento Pluvial Urbano

SMS - Serviço de Mensagens Curtas

SNS - Simple Notification Service

TCP - Protocolo de Controle de Transmissão

TLS - Segurança na Camada de Transporte

XAMPP - Cross-platform, Apache, MySQL, PHP, and Perl

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Com o aumento das mudanças climáticas, as cidades vêm enfrentando desafios crescentes relacionados à gestão de recursos hídricos. Segundo o Intergovernmental Panel on Climate Change (IPCC), existe a previsão de que elevadas temperaturas se façam presentes nos centros urbanos com maior frequência, trazendo consigo a deterioração da qualidade do ar e o aumento da ocorrência de chuvas intensas, que levam a deslizamentos e alagamentos em cidades de clima tropical (IPCC, 2023). Outro motivo é o planejamento urbano ineficiente que permitiu o crescimento desordenado e rápido de ocupações em áreas alagadiças durante os anos, causando interferências na natureza e consequentemente alagamentos. (VASCONCELLOS, 2015).

O Sudeste brasileiro é a região onde se encontram as cidades mais populosas do Brasil. Rio de Janeiro, Minas Gerais e São Paulo são grandes centros urbanos que concentram de forma desordenada boa parte da população do país, essa ocupação somada às fortes chuvas que ocorrem no período do verão são os pontos causadores dos grandes alagamentos da região. (MILANEZ; FONSECA, 2010).

Segundo o Índice de Vulnerabilidade Climática dos Municípios (IVCM), a cidade de São Paulo é mais vulnerável a inundações, enchentes e alagamentos do que outras cidades do Estado (IVCM, 2024). De acordo com dados do Centro de Gerenciamento de Emergências (CGE), durante as chuvas de verão de dezembro de 2022 a março de 2023, foram mais de 300 alagamentos na capital. (CGE, 2024)

Na tentativa de mitigar os riscos associados à inundações em áreas residenciais e comerciais - que vão desde prejuízos à infraestrutura até perdas de vidas humanas - os sistemas de monitoramento pluvial urbano desempenham um papel fundamental, uma vez que atuam como ferramentas importantes para auxílio na redução de danos e prejuízos. (Kobiyama et al., 2006).

Segundo a United Nations International Strategy for Disaster Reduction (UN-ISDR), esses sistemas de alertas requerem a integração da comunidade técnico/científica e da atuação de autoridades e da própria sociedade para que seja possível que a notificação de previsões e detecções e para que essas orientem adequada e corretamente as respostas contra os perigos naturais. (UN-ISDR, 2004).

A própria cidade de São Paulo recebeu novos equipamentos com sensores para monitoramento no período de chuvas, porém é um maquinário caro e em pontos

específicos da cidade. Segundo matéria da (FOLHA, 2023), 100 sensores foram espalhados pelos pontos críticos do município, e eles captam a profundidade da lâmina d'água formada pela chuva.

Mesmo com a existência da CGE, que atua fazendo previsões e indicando áreas sujeitas a alagamentos na cidade quando chuvas intensas acontecem, muitos pontos acabam não aparecendo em seus radares, como locais privados de moradia (condomínios) ou comércios (CGE, 2024).

Dado o cenário apresentado, percebe-se o quão grave são os problemas relacionados às frequentes enchentes em áreas urbanas brasileiras. Portanto, a busca por soluções simples e eficazes, como a notificação dos habitantes e comerciantes dessas regiões, visa minimizar possíveis danos.

Na literatura, são descritos vários trabalhos que foram feitos com a finalidade de monitorar o fluxo pluvial – sendo que há um maior foco nas inundações provenientes da cheia de rios, como é o caso do projeto proposto por Loffi *et al*., 2019, que monitora o nível das águas do Rio Itajaí para a prevenção de enchentes em Santa Catarina.

Seguindo essa mesma linha FRANCESCHINI; FILHO; DANTAS, 2020 apresentam um trabalho onde monitoram possíveis enchentes no município de Caucaia no Ceará utilizando a tecnologia Voice Over IP (Protocolo de Voz sobre Internet), que permite a emissão de alertas e o envio automático de Serviço de Mensagens Curtas (SMS), e-mails e telefonemas em casos de cheias do rio local

Em razão ao exposto, o presente trabalho propõe estudar uma alternativa de alerta para notificar a população em áreas de risco de alagamento, utilizando um sistema inteligente no qual sensores pluviais monitoram o nível da enchente e, em caso de iminência de inundação, o alerta é enviado tanto remotamente quanto no local afetado.

1.2 QUESTÃO DE PESQUISA

Como alertar efetivamente as pessoas quando um alagamento está ocorrendo?

2. OBJETIVOS

2.1. GERAL

O presente projeto tem como objetivo configurar um modelo de sistema capaz de detectar variações no nível da água em regiões com risco de alagamento, proporcionando alertas locais e remotos à população.

2.2. ESPECÍFICOS

Os objetivos específicos deste trabalho incluem o desenvolvimento de um modelo integrado de hardware, software e configuração em nuvem para a detecção de dados pluviais, processamento dessas informações por meio de um algoritmo, emissão de alarmes sonoros e visuais localmente, e, utilizando os serviços da Amazon Web Services (AWS), o armazenamento e envio de alertas por e-mail aos indivíduos. As etapas envolvidas são:

1. Criação de um ambiente simulado com sensor de nível de água e tanque para testes;

2. Configuração de um algoritmo embarcado no microcontrolador Arduino para tomada de decisão com base nos níveis de água detectados;

3. Inclusão de um Display Tela de Cristal Líquido (LCD) e um Buzzer para alarmes visuais e sonoros;

4. Criação de um Web Server (ou servidor Web) local utilizando um módulo Ethernet e transferência dos dados para um servidor Web público;

5. Adequação de roles, funções e políticas na conta AWS através do Identity and Access Management (IAM);

6. Configuração de um algoritmo em Python para o tratamento dos dados provenientes do servidor Web público;

7. Armazenamento do algoritmo Python em um bucket do Amazon Simple Storage Service (S3);

8. Implementação do algoritmo no AWS Lambda, com habilitação do CloudWatch;

9. Criação de um cronograma no Amazon EventBridge para acionamento do Lambda a cada minuto;

10. Armazenamento das mudanças do nível da água no banco de dados DynamoDB;

11. Envio de alarmes por e-mail em caso de alagamento para usuários cadastrados via Amazon Simple Notification Service (SNS).

2.3. RESULTADOS ESPERADOS

O presente trabalho almeja oferecer uma solução alternativa de alerta para a população em áreas com risco de alagamento.

O sistema faz o monitoramento do nível da água, e ao atingir patamares críticos, é feito o envio dos alarmes. Embora o trabalho tenha sido projetado em um ambiente simulado, o modelo proposto, caso seja implementado em uma situação real, além de ser uma alternativa eficaz também pode vir a trazer um impacto positivo na vida das pessoas. O projeto também visa contribuir com avanços nas áreas de tecnologia e sustentabilidade ambiental.

2.4. ORGANIZAÇÃO DO TRABALHO

O desenvolvimento deste trabalho foi dividido em quatro grandes etapas:

1. Apresenta a metodologia utilizada no projeto, incluindo diagramas explicativos e uma descrição detalhada dos materiais, frameworks e serviços que foram essenciais para o desenvolvimento.
2. Foca na implementação do projeto, destacando o ambiente simulado em que foi testado, a lógica de programação aplicada, e como os recursos mencionados na primeira etapa foram utilizados.
3. Mostra os resultados obtidos após a fase de testes, oferecendo uma análise dos dados coletados e das respostas do sistema.
4. Traz as conclusões finais, oferecendo uma visão geral sobre o impacto do projeto, suas principais contribuições e sugestões para melhorias futuras.

3. METODOLOGIA

3.1. VISÃO GERAL

O Sistema de Monitoramento Pluvial Urbano (SMPU) foi desenvolvido com o objetivo de detectar alagamentos e enviar alertas de perigo iminente aos habitantes das áreas afetadas. O sistema monitora continuamente o nível da água e, quando este atinge um patamar crítico, aciona automaticamente os alarmes, garantindo uma resposta rápida e eficaz na região em que o SMPU foi implementado.

Reconhecendo que parte da população pode não ter acesso à internet e, portanto, estar em maior vulnerabilidade em situações de risco, os alarmes do SMPU foram divididos em dois grupos distintos, conforme demonstrado na Figura 1:

Grupo Local: Os alertas são transmitidos por meio de um display LCD conectado ao microcontrolador, que exibe o nível de alagamento, juntamente com um aviso sonoro, alertando as pessoas presentes na área para se afastarem. Para esse grupo, o SMPU não necessita de conexão à internet, garantindo que o alerta seja emitido diretamente no local.

Grupo Móvel: Os dados são transmitidos via um módulo Ethernet conectado ao microcontrolador, que cria um servidor Web local. Esse servidor é replicado para um Web Server em Protocolo de Internet (IP) público. Utilizando os serviços da nuvem AWS, os dados são tratados e é gerado um alerta na forma de e-mail para as pessoas cadastradas no sistema de Grupo Móvel, caso esteja ocorrendo um alagamento.

3.2. DIAGRAMA DO PROJETO

O detalhamento do sistema, ilustrado na Figura 1, fornece uma visão geral da arquitetura do projeto, servindo como referência para as subseções que se seguem nesta seção.

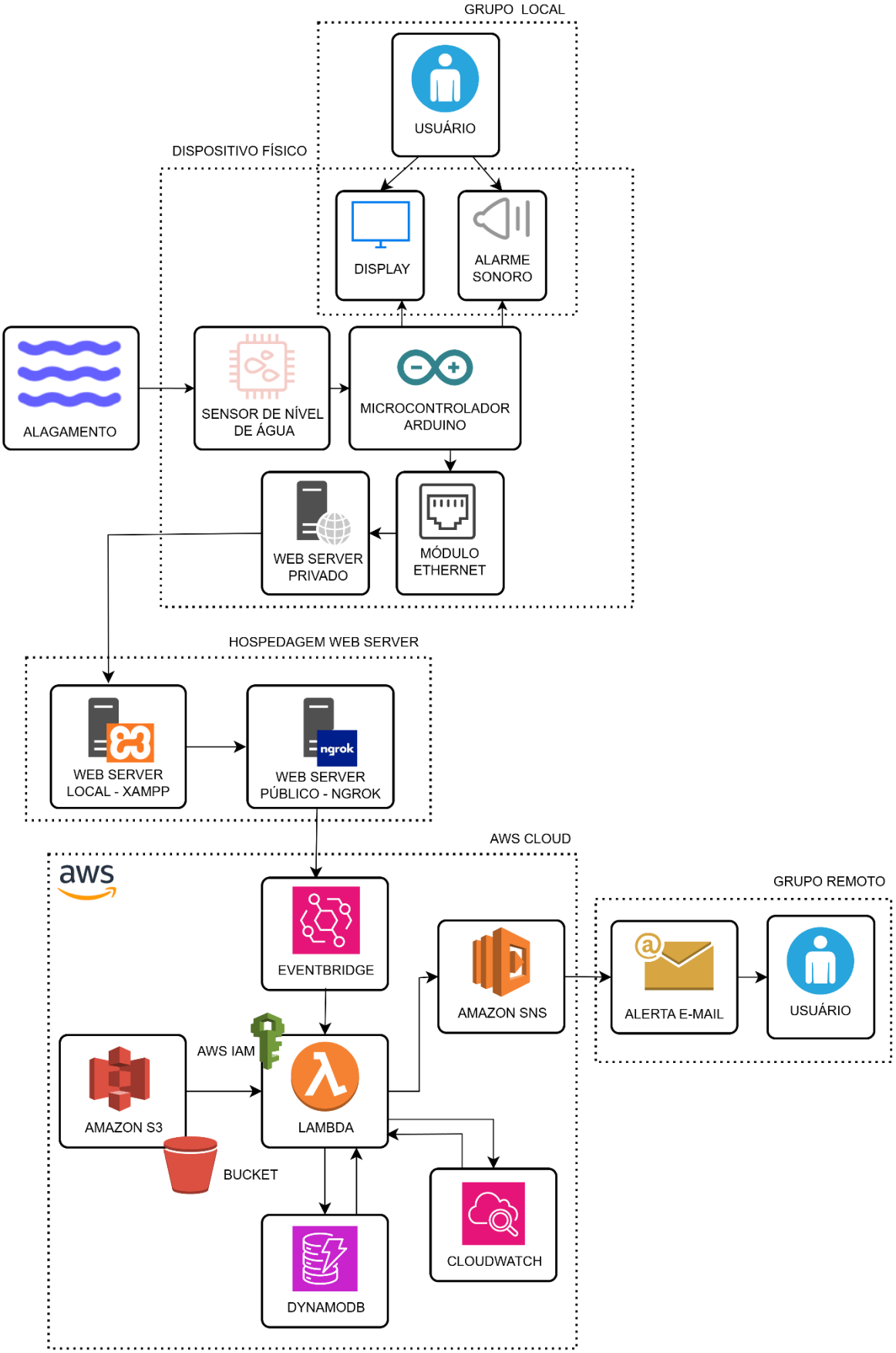

Figura 1: Detalhamento dos componentes do SMPU. Fonte: Autoria própria, 2024

3.3. DISPOSITIVO FÍSICO

Neste estudo, o dispositivo físico foi desenvolvido inteiramente em um ambiente simulado. Contudo, em cenários práticos, devido ao uso de software embarcado, ele poderia ser configurado, com algumas adaptações no módulo para conexão com à internet e instalado, por exemplo, em postes urbanos, conforme ilustrado na Figura 2, onde é apresentada uma esquematização simplificada deste protótipo.

Figura 2: Esquematização de Protótipo SMPU. Fonte: Autoria própria, 2024

3.3.1. HARDWARE

Embora o dispositivo tenha sido desenvolvido em um ambiente simulado, todos os componentes utilizados na simulação são baseados em elementos reais e replicáveis em cenários práticos. Nesta seção, serão apresentados os materiais reais que foram incorporados na biblioteca do simulador durante o projeto.

3.3.1.1. SENSOR DE NÍVEL DE ÁGUA

Um sensor de água é um dispositivo eletrônico utilizado para detectar a presença de água, funcionando por meio da medição da condutividade elétrica do líquido. Conforme descrito por Eletrogate (2024), esses sensores convertem o nível da água em um sinal analógico, que pode ser enviado diretamente para microcontroladores. Eles são amplamente utilizados em aplicações como detecção de água, monitoramento de precipitação e detecção de vazamentos.

Neste trabalho, o sensor foi empregado para monitorar o nível de água resultante de enchentes, sendo conectado diretamente ao microcontrolador Arduino.

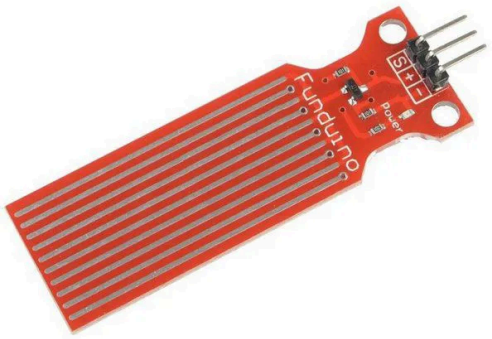

Figura 3: Sensor de nível de água. Fonte: Eletrogate, 2024

3.3.1.2 MICROCONTROLADOR ARDUINO UNO

O microcontrolador escolhido para este projeto é o Arduino Uno, que pode ser programado através do Ambiente de Desenvolvimento Integrado (IDE) Arduino Software. A placa do Arduino Uno possui um bootloader que permite o carregamento direto do código no hardware, eliminando a necessidade de um programador externo (Arduino, 2024). A escolha deste microcontrolador se deve à sua fácil implementação, à sua natureza open-source e ao seu custo relativamente baixo em comparação com outros microcontroladores disponíveis no mercado.

Embora o ESP8266 seja uma alternativa viável e também de fácil implementação, o Arduino Uno foi preferido devido à sua facilidade de integração com a plataforma de simulação utilizada neste projeto.

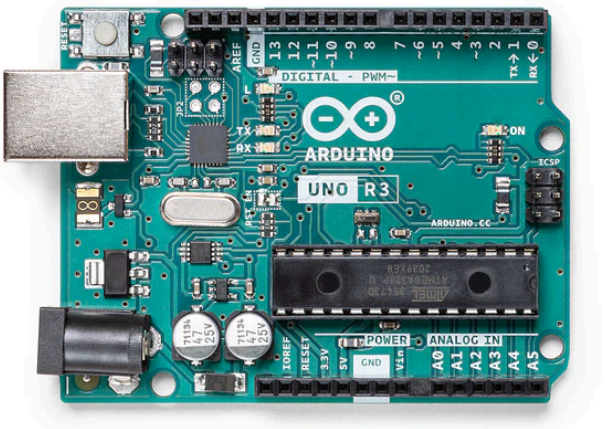

Figura 4: Placa Arduino Uno. Fonte: Arduino, 2024

3.3.1.3 MÓDULO ETHERNET PARA A*RDUINO UNO*

Para que os dados fossem enviados para o servidor Web local e, posteriormente, observados e processados na nuvem, foi utilizado o Arduino Ethernet Shield. Este módulo permite conectar o Arduino à internet em poucos minutos: basta conectá-lo à placa Arduino, configurá-lo na rede desejada e seguir alguns passos simples para iniciar o controle e o monitoramento via internet (Arduino, 2024).

Como o módulo Ethernet foi utilizado em um ambiente simulado, ele gerou um Endereçamento IP Privado Automático (APIPA) em vez de um servidor Web local na porta 80, que seria necessário para enviar as informações para um servidor Web público. Dessa forma, tornou-se necessário o uso de um Framework para replicar as informações do endereço APIPA para o servidor web local.

Em uma implementação em ambiente real, é possível configurar o servidor diretamente com um IP público, permitindo o envio e processamento dos dados na nuvem.

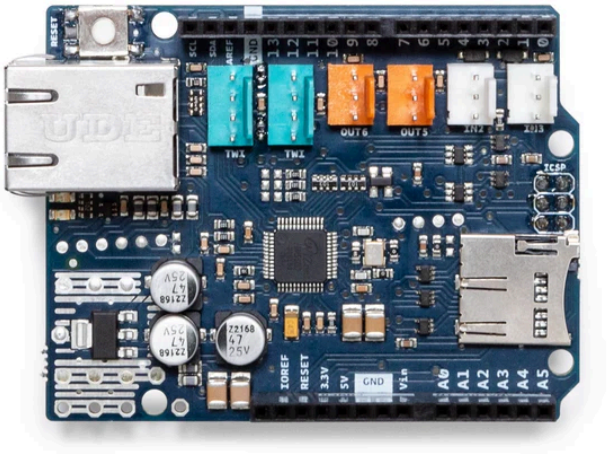

Figura 5: Arduino Ethernet Shield. Fonte: Arduino, 2024

3.3.1.4 DISPLAY LCD

Como parte da validação do sistema e também para exibir avisos sobre o nível de água para o grupo local, foi escolhido um display LCD. Ele foi selecionado por utilizar poucos pinos do microcontrolador, garantindo uma integração eficiente com o sistema. Apesar de seu tamanho compacto, o display LCD é capaz de reproduzir mensagens de forma clara.

Para uma aplicação prática, é possível aprimorar essa parte do sistema, substituindo o display por uma tela maior, o que facilitaria a visualização das informações a uma maior distância.

Figura 6: Display LCD 16x2. Fonte: Eletrogate, 2024

3.3.1.5 TANQUE DE ÁGUA

Para demonstrar o sistema de monitoramento, utilizou-se um tanque de água que simula a acumulação durante uma enchente, possibilitando o teste dos sensores em tempo real. No entanto, a necessidade de adquirir um tanque físico com 100 cm de altura para ajustar manualmente o nível de água tornou inviável realizar os testes em um ambiente real. Por isso, optou-se por uma simulação, onde é possível alterar os níveis de água com apenas alguns cliques.

No ambiente simulado, o tanque possui quatro níveis de água que foram calibrados de acordo com os níveis estimados de alagamento em São Paulo, conforme mostrado na Tabela I. Em uma aplicação prática, seria possível dispensar o uso do tanque e instalar diretamente os sensores e módulos no local com iminência de alagamentos, ajustando a altura e os níveis de alagamento conforme as diretrizes específicas da região.

É importante destacar que o foco deste trabalho é a monitoração utilizando sensores. Assim, no ambiente de teste, o nível de água do tanque foi ajustado manualmente para simular diferentes condições de enchente. Outro ponto relevante é que, embora o tanque ou a situação de alagamento não estejam representados na área do Dispositivo Físico da Figura 1, optou-se por incluí-los nesta seção devido ao tanque simulado estar dentro do contexto da simulação realizada nos testes.

Tabela I: Estimativa de alagamentos.

Nível de Alagamento	Altura da água (cm)
Sem Alagamento	Abaixo de 15
Alagamento Leve	Entre 16 e 30
Alagamento Moderado	Entre 31 e 60
Alagamento Grave	Acima de 61

Fonte: Autoria própria (2024)

3.3.1.6 ALARME SONORO

Para garantir que a população sem acesso a alarmes via aplicativos seja devidamente alertada, foi implementado um alarme sonoro que emite um som em casos de iminência de alagamento. Na simulação, utilizou-se um Módulo Buzzer, uma campainha eletrônica controlada pelo Arduino, de acordo com as regras estabelecidas na Tabela II.

O Módulo Buzzer é um dispositivo que emite som ao receber um sinal elétrico e opera com uma tensão de 5VDC. Ele é amplamente utilizado em projetos eletrônicos para gerar alertas sonoros e sinais acústicos (Eletrogate, 2024). Em uma aplicação prática, o sistema pode ser aprimorado com a incorporação de um dispositivo de maior potência sonora, capaz de ser ouvido a longas distâncias.

Optou-se por utilizar o mesmo nível de som para qualquer tipo de iminência de alagamento, garantindo que a população seja alertada tanto em casos de alagamentos leves quanto em situações mais graves.

Tabela II: Envio de Alarmes Sonoros.

Nível de Alagamento	Disparo de Alarme Sonoro
Sem Alagamento	Não
Alagamento Leve	Sim
Alagamento Moderado	Sim
Alagamento Grave	Sim

Fonte: Autoria própria (2024)

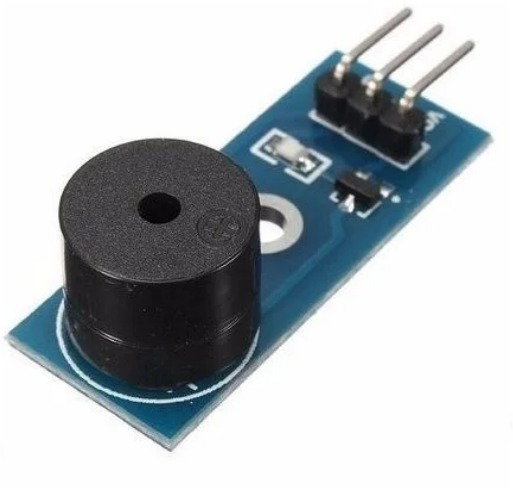

Figura 7: Módulo Buzzer. Fonte: Eletrogate, 2024

3.3.2 SOFTWARE EMBARCADO

Embora o presente projeto tenha sido desenvolvido em ambiente simulado, toda a programação é funcional também para cenários físicos e pode ser refatorada para atender demandas e cenários específicos.

Conforme mencionado na seção 3.3.1, o projeto utiliza uma placa Arduino juntamente com um módulo Ethernet conectado a ela. Dessa forma, toda a programação é incorporada à placa, permitindo que os dados gerados pelo algoritmo sejam enviados para a Internet.

O algoritmo inclui as regras de funcionamento do SMPU. Em resumo, são inicializadas as variáveis, o display LCD, o Buzzer e a conexão Ethernet.

O valor da resistência do sensor de nível de água é recebido pelo pino de entrada, que transforma a altura da água em um valor fixo de resistência, conforme mostrado na Tabela III. Sempre que ocorre uma variação neste valor, o algoritmo diferencia o envio de mensagens de acordo com a informação da Tabela I. Assim, são enviadas descrições no sobre o alagamento, a região afetada e a identificação do SMPU, tanto para o display LCD quanto para o servidor Web privado APIPA no padrão Linguagem de Marcação de Hipertexto (HTML), conforme protocolo da Figura 8.

Para a regra sobre os alarmes sonoros, serão disparados à partir da primeira faixa de alagamentos (Alagamento Leve) conforme Tabela II.

O código contendo o software embarcado está disponibilizado na seção Apêndice A deste trabalho.

Tabela III: Regra de pareamento Resistência x Altura.

Resistência do Sensor (Ω)	Altura da água (cm)
Abaixo de 150	Abaixo de 15
Entre 151 e 300	Entre 16 e 30
Entre 301 e 600	Entre 31 e 60
Acima de 601	Acima de 61

Fonte: Autoria própria (2024)

3.3.3 SERVIDOR WEB PRIVADO

Um servidor Web possibilita a disponibilização de conteúdo online. Esses dados podem estar em diversos formatos, como arquivos HTML, imagens, JavaScript, entre outros.

No contexto deste projeto, o envio de dados para o servidor web utilizando o HTML é adequado, uma vez que ela é amplamente utilizada para definir a forma como os navegadores exibem elementos nas páginas da Web. Além disso, o HTML é considerado um padrão oficial da internet. (HOSTINGER, 2024).

Conforme mencionado anteriormente, na simulação deste projeto, o algoritmo no Arduino gera uma página HTML que é enviada pelo módulo Ethernet para um servidor web privado automático. Essa página contém informações específicas, como a identificação do local, o número de cadastro do módulo SMPU e a descrição da situação atual daquele ponto específico, seguindo o protocolo ilustrado na Figura 8.

De acordo com o algoritmo, as informações no APIPA são atualizadas constantemente, na casa dos milissegundos. No entanto, essa atualização só se torna perceptível quando há uma mudança no estado do nível de água.

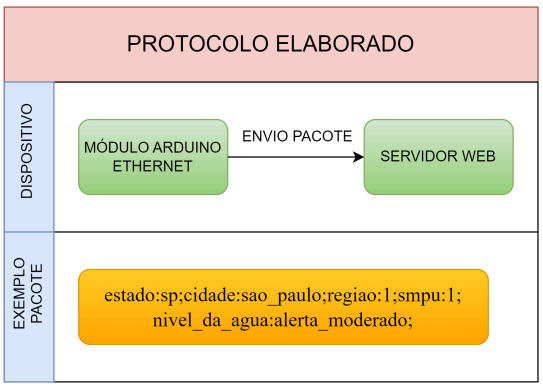

Figura 8: Protocolo Elaborado. Fonte: Autoria Própria, 2024

Para a exemplificação desse determinado SMPU da Figura 7:

Especificação do SMPU:
Estado: São Paulo (SP)
Cidade: São Paulo
Região: Região 1 de São Paulo
Número do SMPU: SMPU 1

Condição do Nível da Água:
Alerta: Moderado

3.4. HOSPEDAGEM WEB SERVER

Esta seção foi implementada devido ao fato de o dispositivo físico simulado ter gerado um APIPA, o que criou a necessidade de replicar os dados de um servidor Web privado para um servidor Web local e, posteriormente, para um servidor Web público, permitindo assim a disponibilização na nuvem.

3.4.1 WEB SERVER LOCAL

Para criar o Web Server local que pode ser reconhecido pelo roteador doméstico, essencial para a formatação de um IP público, foi utilizado o software Cross-platform, Apache, MySQL, Preprocessador de Hipertexto (PHP), and Perl (XAMPP).

O XAMPP oferece uma maneira simples de configurar um servidor Web Apache em PHP (Apache Friends, 2024). Com o XAMPP, foi possível gerar uma réplica do servidor Web originalmente criado no ambiente simulado, com apenas alguns cliques e sem a necessidade de programação.

3.4.2 WEB SERVER PÚBLICO

Para criar o Web Server público, essencial para ser reconhecido pelos serviços da AWS e possibilitar a monitoração remota, foi utilizado o ngrok for development. O ngrok é um proxy reverso que protege e permite a distribuição global de aplicativos e páginas privadas, oferecendo suporte a HTTP, Segurança na Camada de Transporte (TLS) e Protocolo de Controle de Transmissão (TCP). (Ngrok, 2024) Dessa forma, foi possível replicar os dados do servidor local para um servidor com IP público, hospedando essas informações na Internet e garantindo que fossem atualizadas simultaneamente com os servidores privados e locais do sistema.

Um dos desafios do ngrok é que ele gera um novo IP a cada sessão, o que

significa que o endereço IP muda caso o software seja reiniciado. No entanto, foi escolhido para este projeto por ser uma solução gratuita para hospedar as informações da simulação na Internet.

Uma alternativa seria integrar o servidor local diretamente ao serviço Elastic Compute Cloud (EC2) da AWS, que desempenharia a mesma função do ngrok, porém de uma forma mais robusta. Tentativas de implementação dessa solução no EC2 foram feitas durante o trabalho, mas acabaram sendo descartadas devido ao alto custo de hospedagem na AWS.

Para uma aplicação real, seria necessário avaliar a viabilidade da hospedagem de um IP público, seja via EC2 ou por outra alternativa mais acessível, como o ngrok for production.

3.5. AWS CLOUD

Com um IP público refletindo os mesmos dados gerados e processados pelo algoritmo do Arduino no Web Server local, foi possível realizar o tratamento desses dados na nuvem. Para isso, foram escolhidos os serviços da AWS devido ao seu ambiente totalmente integrado, que facilita a implantação de diversas soluções a partir das informações hospedadas no IP público.

A AWS é a plataforma de nuvem mais adotada e abrangente do mundo, oferecendo mais de 200 serviços distribuídos em data centers globalmente. Milhões de pessoas a utilizam para reduzir custos, aumentar a agilidade e acelerar a inovação (AWS, 2024).

Vale destacar que a conectividade com a nuvem poderia ter sido realizada por meio de outras plataformas, como Google Cloud ou Microsoft Azure. No entanto, optou-se pela AWS devido à conta oferecida à Universidade Federal do ABC (UFABC), que facilita o acesso à plataforma para fins de aceleração e estudos acadêmicos.

Nesta seção, serão apresentados os serviços da AWS utilizados neste trabalho e a lógica aplicada para tratar os dados do Web Server público.

3.5.1 AWS EVENTBRIDGE

O AWS EventBridge é um serviço serverless que facilita a criação de aplicativos ao conectar componentes por meio de eventos (AWS EventBridge, 2024).

Neste trabalho, o EventBridge foi utilizado devido à sua funcionalidade de Cronograma, que permite acionar outros serviços da AWS com base em configurações de

tempo. Especificamente, ele foi configurado para acionar o AWS Lambda a cada 1 minuto.

3.5.2 AWS LAMBDA

O AWS Lambda é um serviço que executa código em resposta a eventos, gerenciando automaticamente os recursos de computação sem a necessidade de servidores (AWS Lambda, 2024).

Neste projeto, o AWS Lambda executa o código que trata os dados do nível de água a partir do Web Server público. Quando acionado pelo EventBridge, o Lambda faz orquestramento a execução, interagindo com outros serviços da AWS para enviar o alerta.

3.5.3 AWS S3

O Amazon S3 é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade, segurança e desempenho. (Amazon S3, 2024)

Neste trabalho, foi necessário armazenar o código do algoritmo em um bucket no S3. Como o pacote, incluindo as bibliotecas necessárias, excedeu o limite de 10 MB permitido para upload direto no AWS Lambda, ele foi armazenado no S3. O AWS Lambda foi configurado para apontar para o link privado do pacote no bucket, garantindo sua execução.

3.5.4 AWS IAM

O AWS IAM é um serviço que permite controlar de forma segura o acesso e permissões dos recursos da AWS.(AWS IAM, 2024).

Neste projeto, o AWS IAM foi utilizado para configurar as políticas e funções do serviço AWS Lambda. Como o Lambda interagiu com vários outros serviços da AWS, foi necessário ajustar as permissões e acessos para garantir a execução correta.

3.5.5 AWS DYNAMODB

O Amazon DynamoDB é um banco de dados Not Only SQL (NoSQL) totalmente gerenciado e sem servidor, oferecendo desempenho com latência inferior a dez milissegundos em qualquer escala. (AWS DynamoDB, 2024)

Nesta aplicação, o DynamoDB foi escolhido devido à sua rapidez e fácil integração com o AWS Lambda. O banco de dados é acionado pelo algoritmo para armazenar o status do nível da água, uma informação crucial para a lógica de negócio da função Lambda, conforme detalhado na seção do algoritmo.

3.5.6 AWS CLOUDWATCH

O Amazon CloudWatch monitora em tempo real os recursos e aplicações executados na AWS, permitindo a coleta e o monitoramento de métricas que ajudam a avaliar o desempenho desses recursos e aplicações (AWS CloudWatch, 2024).

Durante a implementação deste trabalho, o CloudWatch foi fundamental para depurar erros, além de fornecer logs em tempo real que garantiram a observação das respostas da aplicação.

3.5.7 AWS SNS

O Amazon SNS permite que os usuários recebam notificações via push, e-mail, SMS, entre outros canais, facilitando o envio de mensagens para destinatários inscritos em um tópico (AWS SNS, 2024).

No contexto deste projeto, o SNS atua como o canal de saída, enviando alertas por e-mail para os usuários inscritos no tópico sempre que o algoritmo da função Lambda detecta uma situação de alagamento, conforme definido nas regras de negócio.

3.5.8 SOFTWARE DE FUNÇÃO LAMBDA

Para que a AWS consiga identificar e tratar os dados do IP público, conforme mostrado na Figura 8, foi necessário criar um algoritmo utilizando o serviço AWS Lambda. O algoritmo foi desenvolvido em Python, uma vez que essa linguagem é compatível com o AWS Lambda e familiar ao desenvolvedor. Para auxiliar na codificação, foi utilizada a IDE Visual Studio Code.

Esse algoritmo realiza uma requisição HTTP para obter os dados do nível da água e os compara com o nível atual armazenado no DynamoDB. Se houver uma alteração no nível da água, a função atualiza o banco de dados e envia um alerta via SNS , que dispara um e-mail para o grupo remoto. Caso ocorram erros na requisição ou na atualização do DynamoDB, a função retorna mensagens de erro que podem ser visualizadas no CloudWatch.

O código contendo a função Lambda está disponibilizado na seção Apêndice A deste trabalho.

4. IMPLEMENTAÇÃO

4.1 IMPLEMENTAÇÃO DO HARDWARE E SOFTWARE EMBARCADO

A implementação do sistema descrito neste trabalho foi realizada em um ambiente simulado utilizando o software Proteus. Este potente Framework facilita o design e a testagem de sistemas embarcados por meio de um layout esquemático intuitivo (Proteus, 2024).

O Proteus destaca-se pela sua excelente compatibilidade com microcontroladores, como o Arduino utilizado neste projeto. Além disso, o software emprega uma biblioteca de materiais que replicam com precisão o comportamento dos materiais reais, o que não só melhora a acurácia dos testes, mas também simplifica a transição para uma implementação física do projeto.

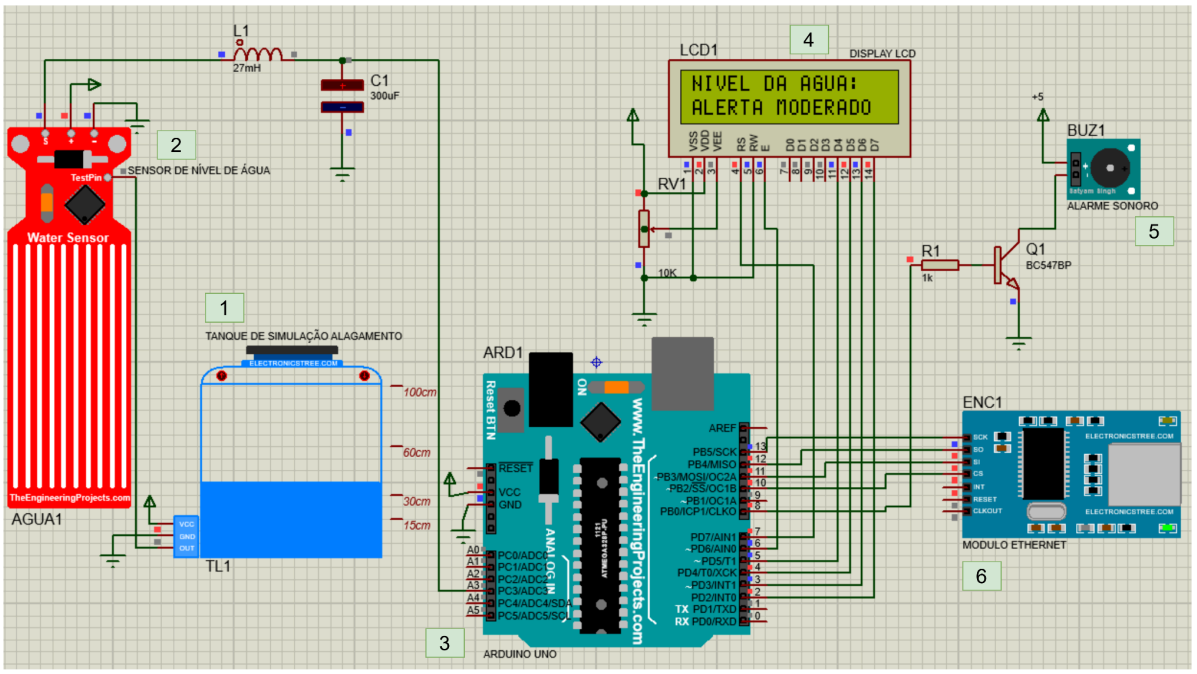

Figura 9: SMPU Simulado no Proteus. Fonte: Autoria Própria, 2024

Conforme Figura 9, o layout da implementação ficou disposto da seguinte forma:

1. Tanque de simulação manual de alagamento: O sistema inclui um tanque para simular diferentes níveis de alagamento, com alturas ajustáveis que vão de 0 a 100 cm.
2. Sensor de nível de água: Equipado com um capacitor de 300 µF e um indutor de 27 mH, este sensor mede a altura da água no tanque e converte essa informação em

uma variação de resistência.

3. Microcontrolador Arduino Uno: O Arduino Uno processa os sinais recebidos do sensor de nível de água, aplicando as regras descritas na Seção 3.3.2. Ele então transmite os dados processados para o display LCD, Buzzer e um módulo Ethernet.
4. Display LCD: Conectado aos pinos de saída do microcontrolador, o display LCD fornece ao usuário uma visualização em tempo real do nível de água no tanque.
5. Buzzer para Alarme Sonoro: Conectado ao pino 8 do microcontrolador, o Buzzer é responsável por emitir um alerta sonoro em caso de alagamento.
6. Módulo Ethernet para Arduino: Este módulo é responsável pela configuração do servidor Web privado, conforme detalhado na Seção 3.3.3.

4.2 IMPLEMENTAÇÃO DO WEB SERVERS

A partir da programação realizada no Arduino e com o uso do módulo Ethernet, foi possível criar um servidor Web local utilizando o APIPA no endereço IP 169.254.199.140. No entanto, essa abordagem apresentou uma limitação, pois a conexão APIPA não é acessível pelo roteador local, impossibilitando de torná-la pública.

Para contornar essa limitação e viabilizar o uso das informações geradas na página HTML do APIPA, foi criado um servidor local na porta 80 com o endereço 192.168.0.5/smpu01/ por meio do framework XAMPP.

O XAMPP foi configurado para copiar automaticamente todas as informações recebidas do APIPA e transferi-las para um endereço IPv4 local do roteador, conforme ilustrado na Figura 10.

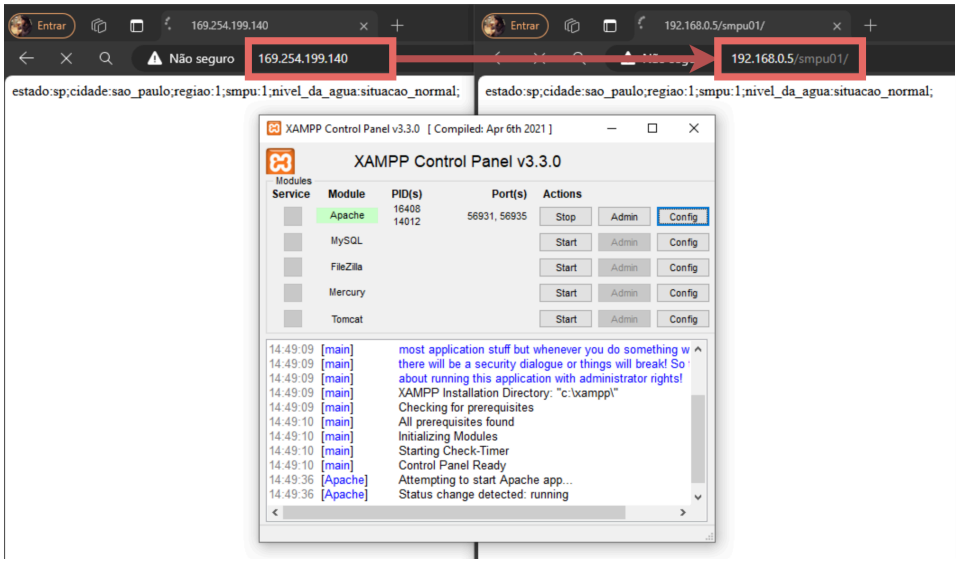

Figura 10: Transferência APIPA para IPv4. Fonte: Autoria Própria, 2024

Para replicar o servidor local em um servidor público e permitir o acesso pela AWS, foi utilizado o Framework ngrok.

Após configurar o *ngrok* para direcionar o IP local e porta utilizada pelo *XAMPP* (http://localhost:80), o *F*ramework gera automaticamente um servidor público que replica os dados do servidor local, conforme ilustrado na Figura 11. Como mencionado anteriormente, a Localizador Uniforme de Recursos (URL) do servidor público tem a duração limitada a uma única sessão no plano gratuito utilizado neste trabalho.

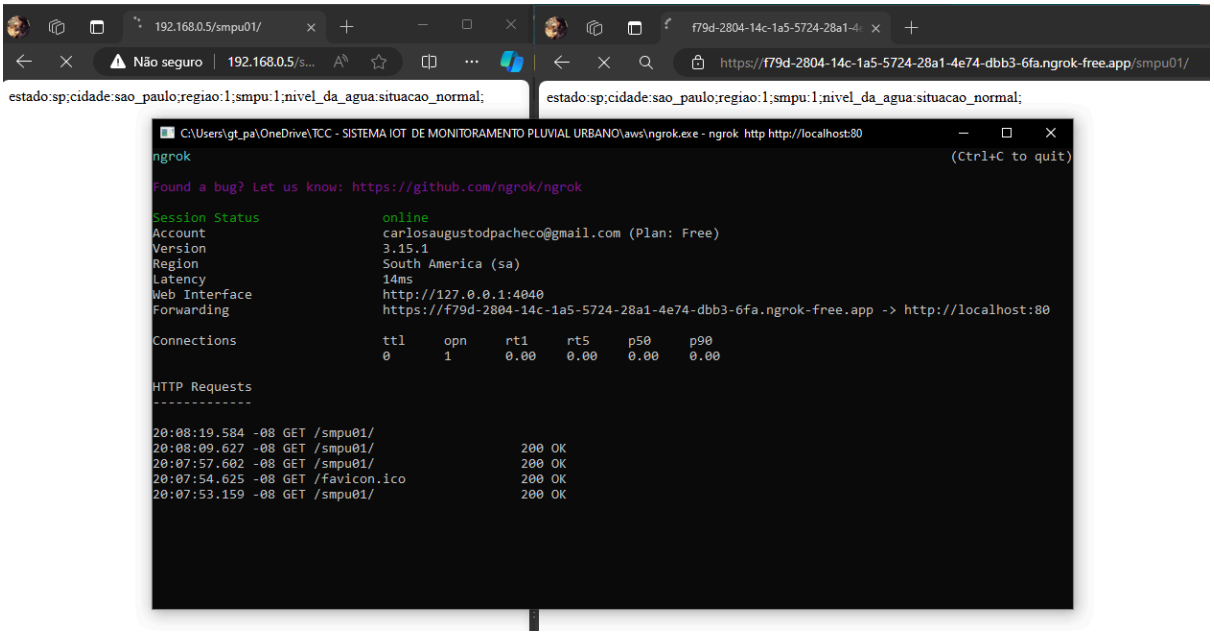

Figura 11: Transferência Servidor Local para Público. Fonte: Autoria Própria, 2024

4.3 IMPLEMENTAÇÃO NA AWS

A partir do servidor público gerado pelo ngrok, foi possível processar os dados utilizando os serviços da AWS. Conforme descrito na Seção 3.5, vários serviços integrados foram empregados para essa aplicação.

Para as configurações na AWS, utilizou-se o AWS Management Console, por essa plataforma é possível selecionar os serviços necessários utilizados pela aplicação.

O primeiro serviço utilizado foi o AWS Lambda onde foi criada a função SMPU01 utilizada nesse projeto. Pelo console, além de uma visão geral sobre o serviço é possível o upload do código com o algoritmo desenvolvido, caso ele seja menor à 10MB. Após o upload, o AWS Lambda configura todas as integrações necessárias, como por exemplo, as integrações com o serviços DynamoDB e o Amazon SNS.

A função Lambda, conforme regra de negócio da Seção 3.5.8, foi codificado e

testado no ambiente local e depois implementado na AWS, os resultados obtidos estão disponíveis na Seção 5.

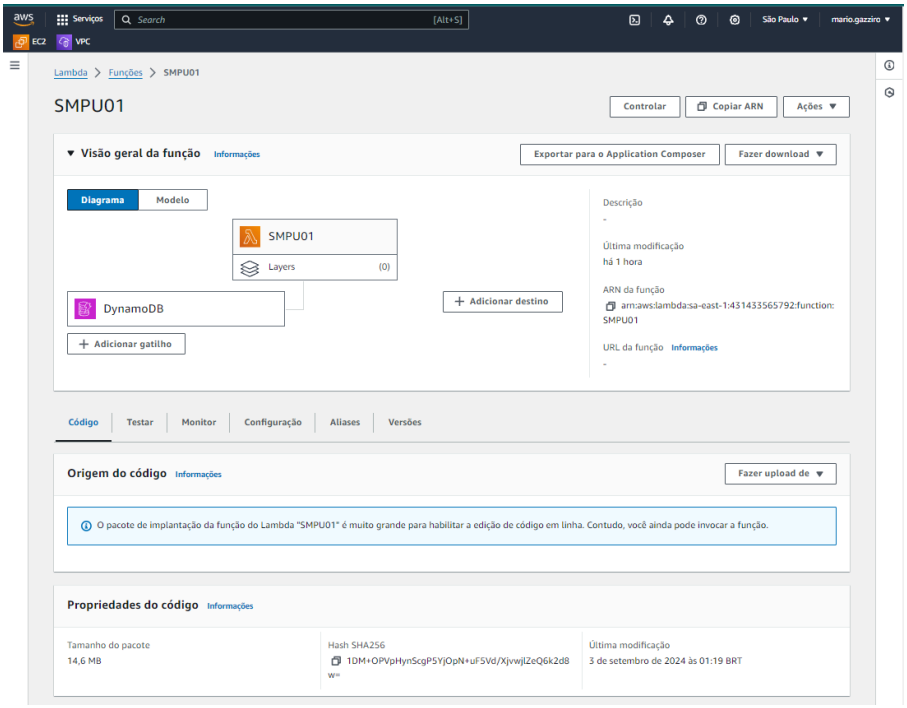

Figura 12: Função SMPU01 no AWS Lambda. Fonte: Autoria Própria, 2024

Conforme demonstrado anteriormente, foi necessário utilizar o serviço Amazon S3 para armazenar o código do algoritmo na AWS. Para isso, foi criado um bucket onde o arquivo lambdasmpu.zip foi armazenado, conforme Figura 13, permitindo que o AWS Lambda fosse redirecionado para esse bucket.

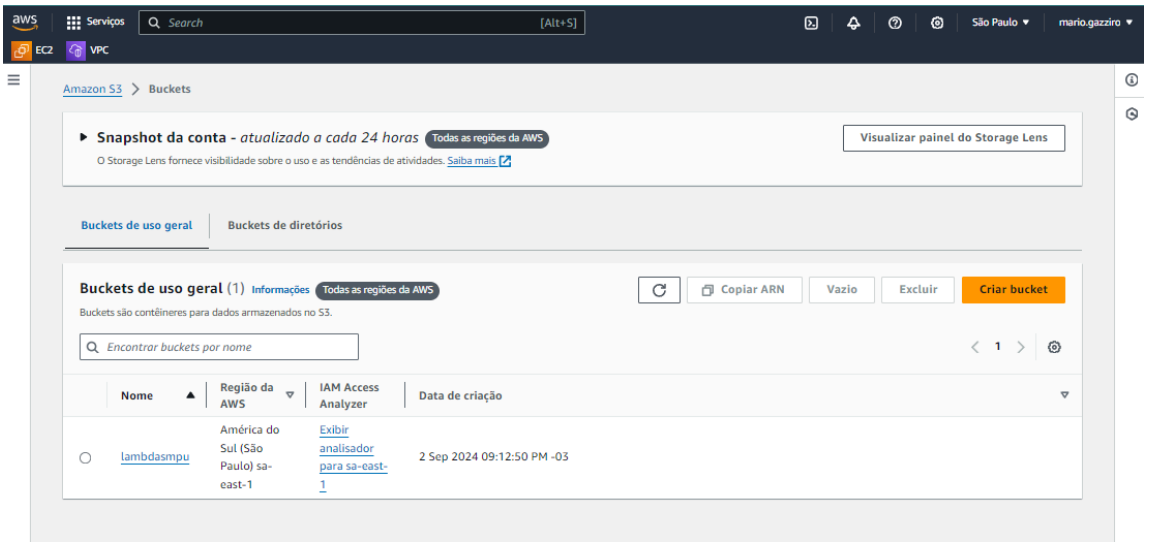

Figura 13: Bucket S3 armazenando o algoritmo. Fonte: Autoria Própria, 2024

Também foi necessário configurar o banco de dados DynamoDB por meio do AWS Management Console. Para isso, foi criada a tabela WaterLevel, referenciada no código da função Lambda.

O console do DynamoDB também auxilia na validação de testes, permitindo visualizar os itens retornados pela tabela. Neste trabalho, foi possível verificar se o nível da água foi atualizado corretamente na tabela, como demonstrado na Figura 14, onde é mostrado que, em determinado momento, o campo “nivel_agua” estava em "alerta_grave".

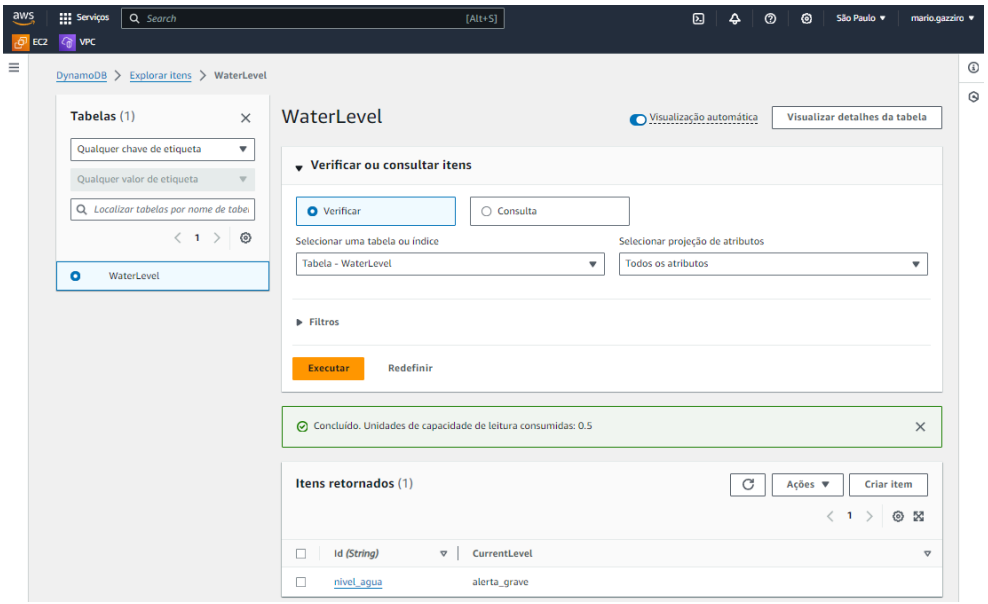

Figura 14: Tabela WaterLevel no DynamoDB. Fonte: Autoria Própria, 2024

Para configurar o envio de e-mails com os alertas, foi utilizado o serviço Amazon SNS no Console AWS. Foi criado o tópico SMPU01, configurado para enviar alertas por e-mail, uma opção gratuita na AWS.

Para que os usuários recebam os e-mails, cada tópico do SNS contém um banco de assinaturas, após à inclusão de um endereço de e-mail válido, o usuário receberá uma mensagem solicitando a confirmação para receber os alertas associados a esse tópico, conforme ilustrado na Figura 15.

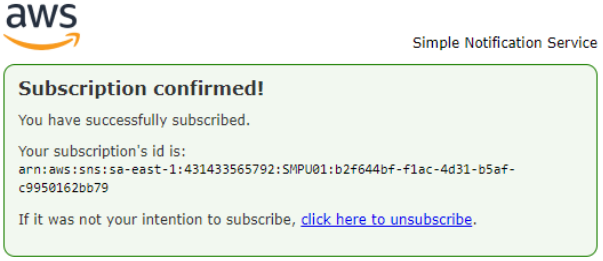

Figura 15: Inscrição para e-mails pelo AWS SNS. Fonte: Autoria Própria, 2024

Uma vez confirmada a assinatura, o usuário será incluído na lista de assinantes confirmados do tópico. A partir da visão geral do Amazon SNS, é possível publicar mensagens diretamente ou associá-las a uma função AWS Lambda, como foi configurado neste trabalho para o envio automático de e-mails caso ocorra um alerta de alagamento.

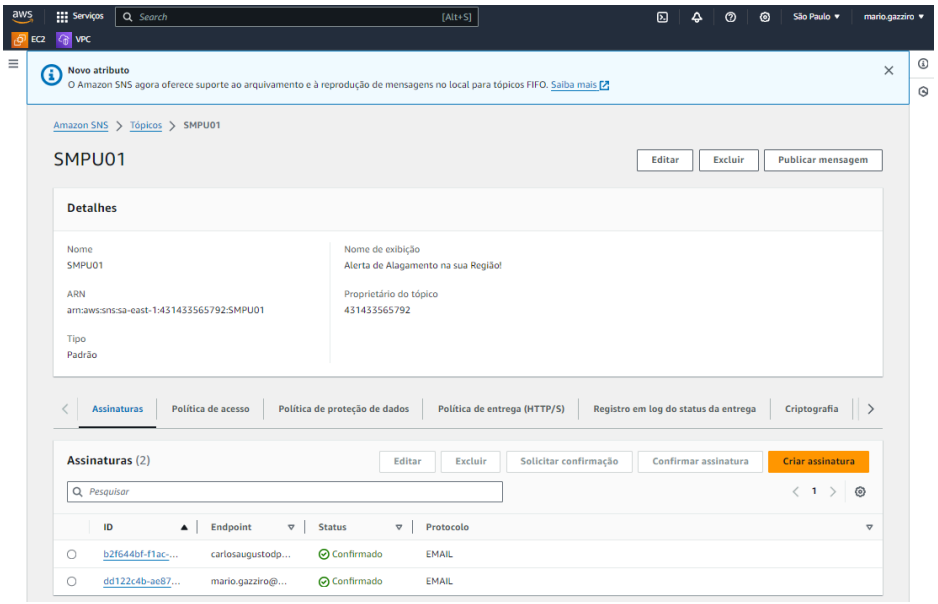

Figura 16: Tópico SMPU01 na AWS SNS. Fonte: Autoria Própria, 2024

Com os serviços configurados no código nos passos anteriores, foi necessário criar a regra de acionamento AcionamentoSMPU no Amazon EventBridge. O destino foi ajustado para a função AWS Lambda SMPU01, com um intervalo fixo de 1 minuto para cada evento acionado.

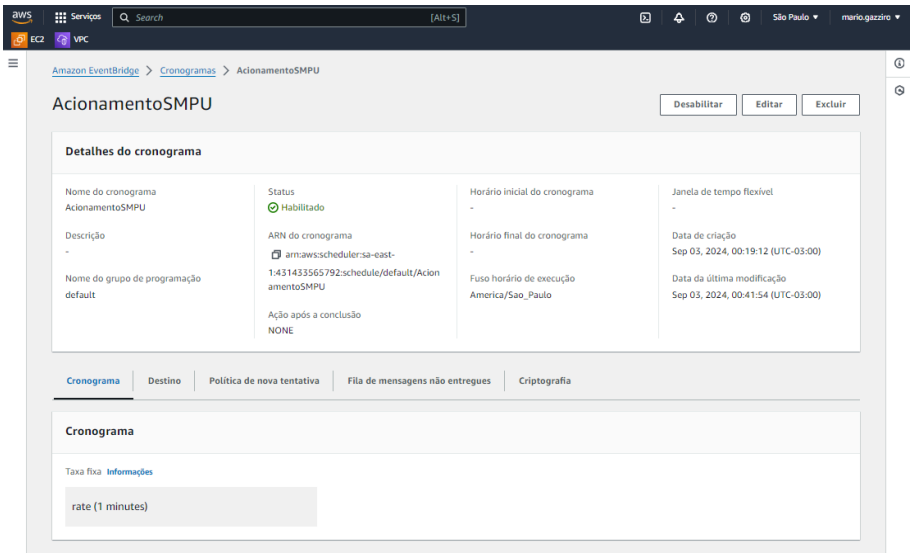

Figura 17: Cronograma AcionamentoSMPU no EventBridge. Fonte: Autoria Própria, 2024

Devido ao elevado número de integrações do AWS Lambda com outros serviços, diversos erros relacionados a permissões foram encontrados. Para configurar e conceder as permissões necessárias, utilizou-se o AWS IAM, seguindo sempre as melhores práticas de segurança, como o princípio de menor privilégio, para garantir a integridade e segurança do projeto.

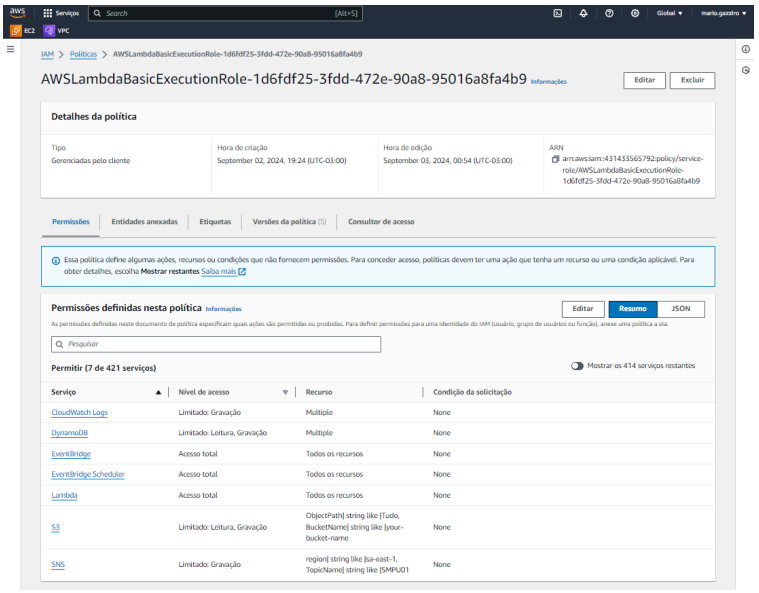

Figura 18: Políticas AWS IAM do AWS Lambda. Fonte: Autoria Própria, 2024

O serviço AWS CloudWatch, integrado ao AWS Lambda, registra os logs de todas as chamadas feitas à função Lambda, nessa implementação o serviço auxiliou na identificação de bugs e na validação dos pontos de erro no código.

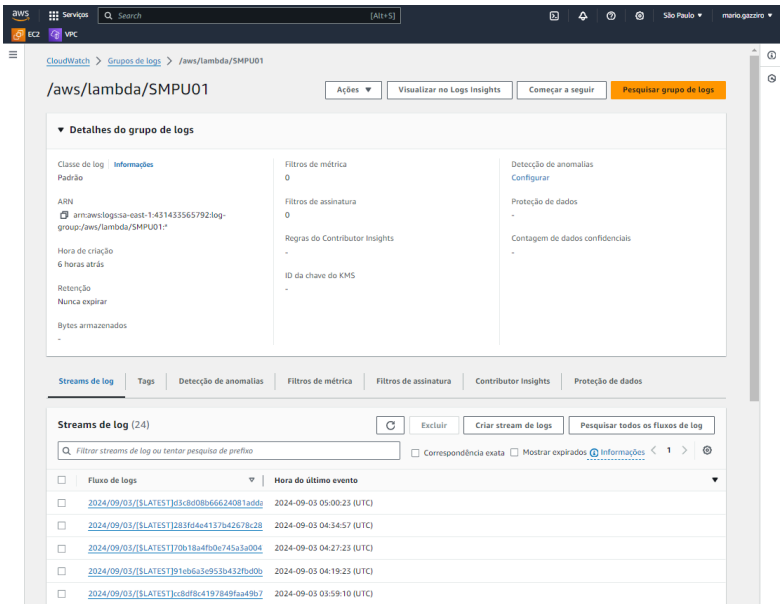

Figura 19: Logs de chamadas pelo Cloudwatch. Fonte: Autoria Própria, 2024

5. RESULTADOS

Para validar a implementação do SMPU, foram realizados testes End-to-End (E2E) comparando as respostas do display LCD e do buzzer no ambiente simulado, o servidor Web local (192.168.0.5/smpu01), o servidor Web público, a tabela WaterLevel no AWS DynamoDB, e, por fim, os e-mails enviados aos usuários apenas em casos de alerta de alagamento. Cada nível de água no tanque foi testado para assegurar que as respostas dos dispositivos estejam alinhadas com as regras definidas na Tabela I.

Os testes E2E visam avaliar o comportamento do sistema de forma integral, "de uma ponta à outra", simulando a atividade do usuário final (Alura, 2024). Eles foram escolhidos para garantir a correta integração de todos os componentes e validar que os fluxos de trabalho funcionem conforme esperado. Os resultados desses testes estão detalhados nesta seção.

O teste inicial foi setar o tanque de de água para 10 cm, o que de acordo com à Tabela I simula uma situação normal, por isso, não tem necessidade de enviar e-mail e acionar o buzzer.

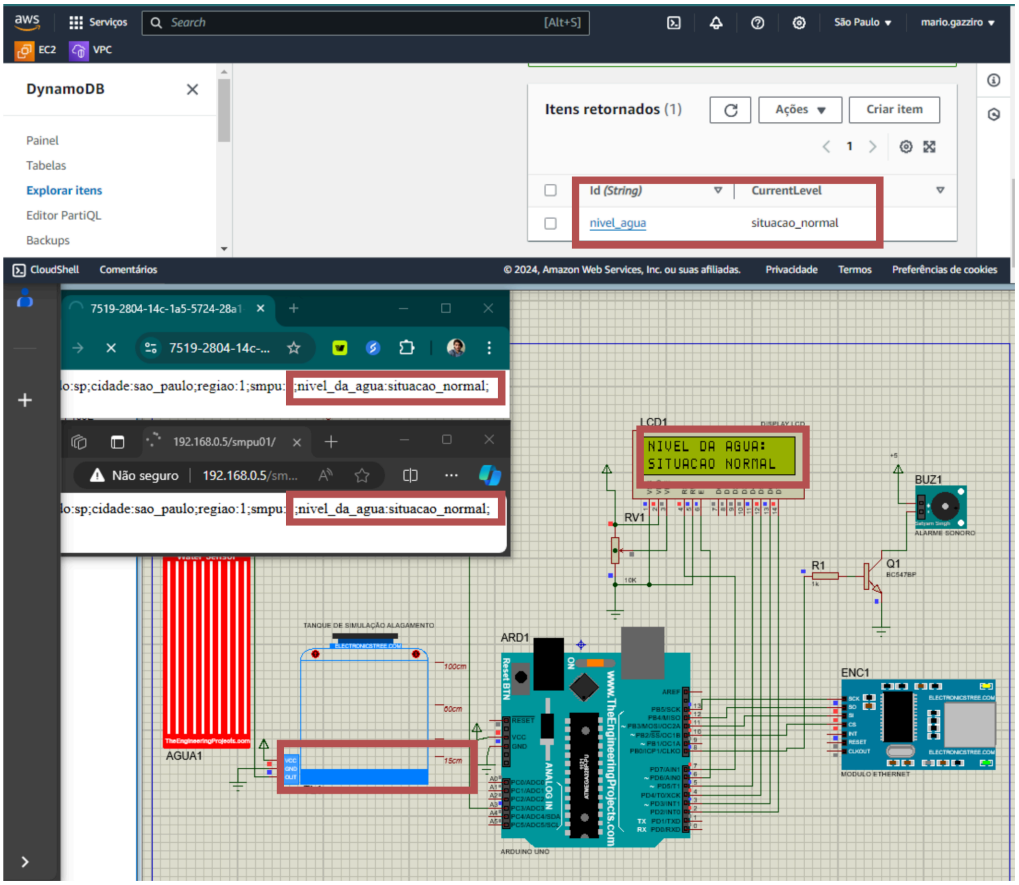

Figura 20: Teste para Situação Normal. Fonte: Autoria Própria, 2024

No segundo teste, o tanque de simulação de água foi preenchido até 15 cm, representando o início de um alagamento. Isso gerou um alerta leve, acionando o envio de um e-mail e ativando o buzzer imediatamente.

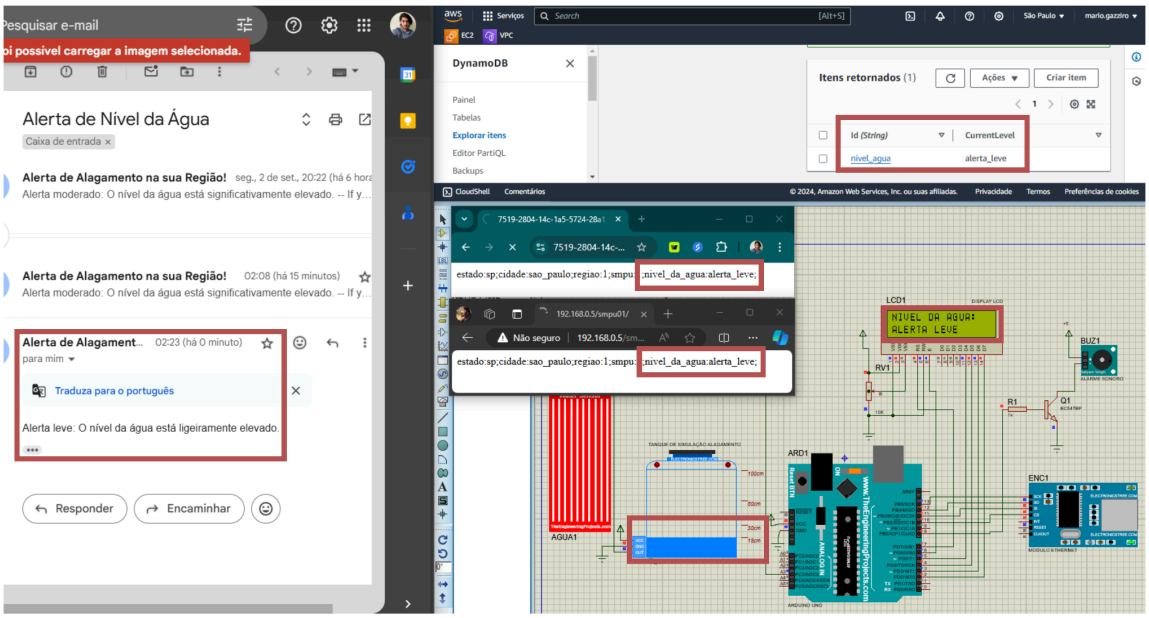

Figura 21: Teste para Alerta Leve. Fonte: Autoria Própria, 2024

No terceiro teste, o nível da água no tanque de simulação foi elevado para 30 cm, o que desencadeou um alerta moderado. Nesse caso, um e-mail foi enviado e o buzzer permaneceu ativado.

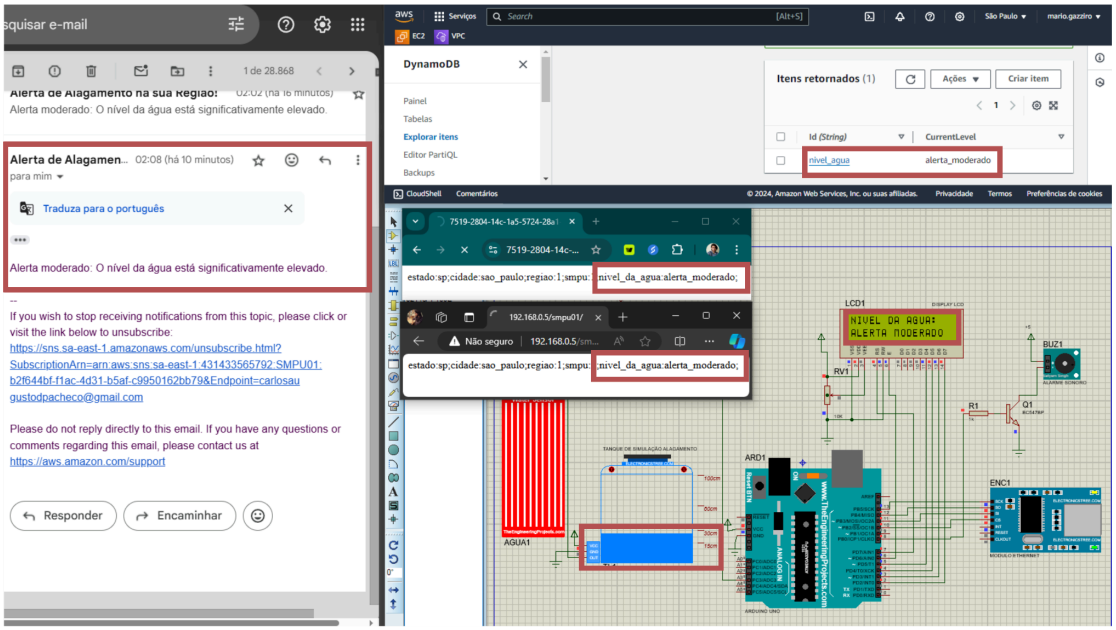

Figura 22: Teste para Alerta Moderado. Fonte: Autoria Própria, 2024

Por fim, foi realizado um teste com o tanque de simulação de água ajustado manualmente para um nível de 80 cm, representando uma situação de calamidade e gerando um nível de alerta grave, é feito o envio do -mail de alerta informando sobre o nível crítico do alagamento, e o buzzer ainda se mantém acionado.

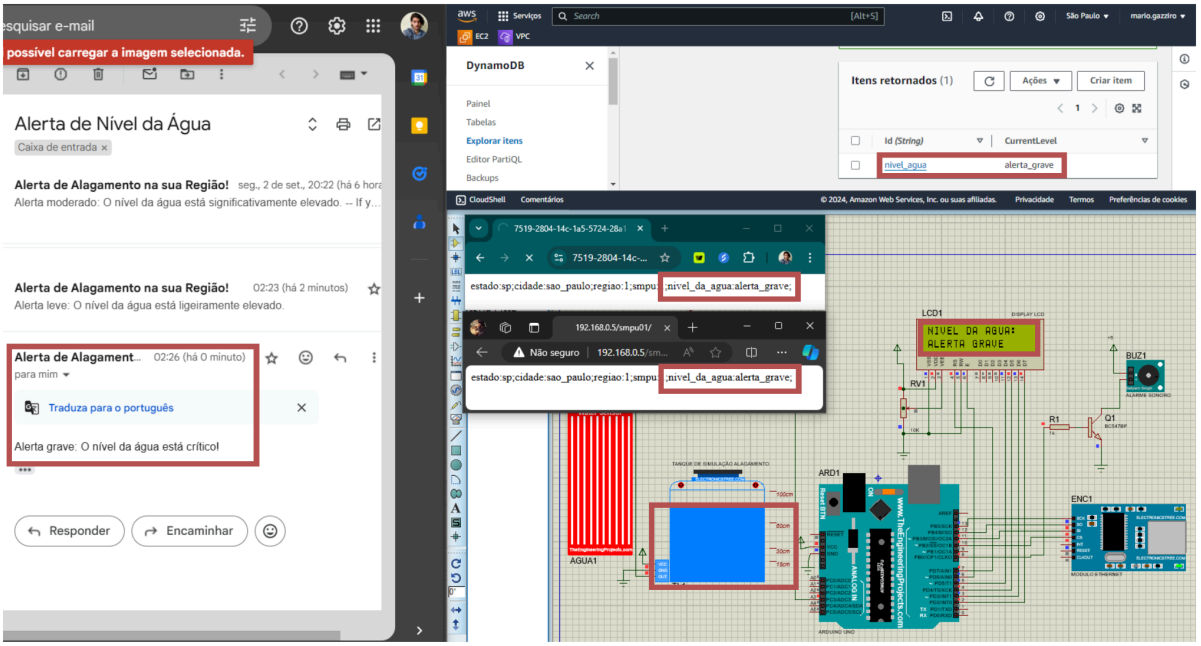

Figura 23: Teste para Alerta Grave. Fonte: Autoria Própria, 2024

Após realizar exaustivamente os testes E2E pode-se afirmar que os resultados foram satisfatórios. O aumento ou diminuição da altura da água foi refletido em todos os estágios do sistema, com alertas enviados ao usuário por e-mail em menos de 2 minutos, e alarmes locais ativados em poucos segundos.

A implantação desse sistema foi feita em um ambiente simulado, e por causa disso, pode ter gerado uma pequena latência. Além disso, foram utilizados servidores públicos gratuitos e serviços AWS básicos para evitar custos maiores para o trabalho. Em um ambiente real, com sistemas de computação e armazenamento mais avançados, a taxa de resposta seria ainda mais rápida.

6. CONCLUSÕES

Os alagamentos em grandes cidades têm impactos significativos e abrangentes, afetando tanto a infraestrutura urbana quanto a qualidade de vida dos seus habitantes. A frequência e a intensidade desses eventos são amplificadas por fatores como o crescimento urbano desordenado e as mudanças climáticas. Esses desafios ressaltam a necessidade urgente de implementar soluções eficazes para proteger a segurança e o bem-estar das populações urbanas. O estudo apresentado neste trabalho demonstrou como um sistema de monitoramento de enchentes pode desempenhar um papel crucial na mitigação desses impactos.

Os resultados obtidos confirmaram a eficiência do sistema como um todo, sendo capaz de monitorar o nível da água em áreas alagadas e alertar os usuários, em caso de iminência de alagamento, locais tanto pelo som quanto pelo display, assim como alertar remotamente por e-mail.

Assim, este trabalho contribui para que futuras pesquisas avancem no desenvolvimento de novos sistemas de monitoramento de alagamentos e até na construção de um modelo físico mais completo, que possa ser aplicado em campo.

7. REFERÊNCIAS

Alura. Tipos de testes: quais os principais e por que utilizá-los? Disponível em: https://www.alura.com.br/artigos/tipos-de-testes-principais-por-que-utiliza-los. Acesso em: 04 de set de 2024.

Apache Friends. XAMPP. Disponível em: https://www.apachefriends.org/pt_br/about.html. Acesso em: 02 de set de 2024.

Arduino. Arduino Uno. Disponível em: https://store-usa.arduino.cc/products/arduino-uno-rev3. Acesso em: 02 de jun de 2024.

Arduino. Arduino Ethernet Shield. Disponível em: https://store-usa.arduino.cc/products/arduino-ethernet-shield-2. Acesso em: 02 de jun de 2024.

AWS. Amazon Simple Notification Service (SNS). Disponível em: https://docs.aws.amazon.com/pt_br/mobile/sdkforxamarin/developerguide/sns.html. Acesso em: 04 de set de 2024.

AWS. O que é o Amazon CloudWatch? Disponível em: https://docs.aws.amazon.com/pt_br/AmazonCloudWatch/latest/monitoring/WhatIsCloudWatch.html. Acesso em: 04 de set de 2024.

AWS. O que é o Amazon DynamoDB? Disponível em: https://docs.aws.amazon.com/pt_br/amazondynamodb/latest/developerguide/Introduction.html. Acesso em: 04 de set de 2024.

AWS. O que é a Amazon EventBridge? Disponível em: https://docs.aws.amazon.com/pt_br/eventbridge/latest/userguide/eb-what-is.html. Acesso em: 04 de set de 2024.

AWS. O que é o IAM? Disponível em: https://docs.aws.amazon.com/pt_br/IAM/latest/UserGuide/introduction.html. Acesso em: 04 de set de 2024.

AWS. O que é o AWS Lambda? Disponível em: https://docs.aws.amazon.com/pt_br/lambda/latest/dg/welcome.html. Acesso em: 04 de set de 2024.

AWS. O que é o Amazon S3? Disponível em: https://docs.aws.amazon.com/pt_br/AmazonS3/latest/userguide/Welcome.html. Acesso em: 04 de set de 2024.

AWS. O que é AWS?. Disponível em: https://aws.amazon.com/pt/what-is-aws/. Acesso em: 03 de set de 2024.

CGE: Centro de Gerenciamento de Emergências Climáticas. Disponível em: https://www.cgesp.org/v3/ Acesso em: 24 de mar de 2024.

Folha Uol. Cidade de SP ganha novos equipamentos e reforça monitoramento no período de chuvas. Folha Uol, 11 out 2023. Disponível em: https://estudio.folha.uol.com.br/prefeitura-de-saopaulo/2023/10/cidade-de-sp-ganha-novos-equipamentos-e-reforca-monitoramento-no-periodo-de-chuva.shtml
Acesso em: 22 de abr de 2024.

ELETROGATE. Display LCD 16x2. Disponível em: https://www.eletrogate.com/display-lcd-16x2-i2c-backlight-azul
Acesso em: 02 de jun de 2024.

ELETROGATE. Módulo Buzzer Passivo 5v. Disponível em: https://www.eletrogate.com/modulo-buzzer-passivo-5v
Acesso em: 02 de set de 2024.

ELETROGATE. Módulo sensor de chuva e nível de agua. Disponível em: https://www.eletrogate.com/modulo-sensor-de-chuva-nivel-de-agua
Acesso em: 02 de jun de 2024.

FRANCESHINI, J.R.R.; FILHO, V.P.S..; DANTAS, V.A.. Sistema de Alerta Para Possíveis Cheias Utilizando Voip e Telefonia Convencional, Para Defesa Civil do Município de Caucaia-Ce, Brasil, 2020. Disponível em: https://www.scielo.br/j/rbmet/a/yY4hmHB7LMM8t7LXvKtcygj/?lang=pt&format=html#
Acesso em: 22 de abr de 2024.

HOSTINGER. Web Server: O que é e Como Funciona? Disponível em: https://www.hostinger.com.br/tutoriais/web-server
Acesso em: 19 de ago de 2024.

HOSTINGER. O Que é HTML: O Guia Definitivo para Iniciantes Disponível em: https://www.hostinger.com.br/tutoriais/o-que-e-html-conceitos-basicos
Acesso em: 19 de ago de 2024.

KOBIYAMA, M.; MENDONÇA, M.; MORENO, D. A.; MARCELINO, I. P. V. O.; MARCELINO, E. V.; GONÇALVES, E. F.; BRAZETTI, L. L. P.; GOERL, R. F.; MOLLERI, G. S. F.; RUDORFF, F. M.; Prevenção de desastres naturais: conceitos básicos. Florianópolis: Ed. Organic Trading, 2006. 109 p. p. 30-47.

LOFFI, Leandro et al. Monit-RIO - Tecnologia da informação de comunicação para monitoramento de rios em casos de cheias. 1. ed. Resende. AEDB, 2016. Disponível em: https://www.aedb.br/seget/arquivos/artigos16/13424296.pdf. Acesso em: 23 de mar de 2024.

IPCC. Climate Change 2023: Synthesis Report. Disponível em: https://www.ipcc.ch/report/sixth-assessment-report-cycle/. Acesso em: 23 de mar de 2024.

IVCM. Índice de Vulnerabilidade Climática dos Municípios. Disponível em: https://institutovotorantim.org.br/ivcm/#:~:text=O%20IVCM%20tem%20como%20no,outras%20informa%C3%A7%C3%B5es%20sobre%20o%20territ%C3%B3rio. Acesso em: 23 de mar de 2024.

MILANEZ, Bruno; FONSECA, Igor Ferraz da. Justiça climática e eventos climáticos extremos: o caso das enchentes no Brasil. Disponível em: https://repositorio.ipea.gov.br/bitstream/11058/5554/1/BRU_n4_justica.pdf . Acesso em: 21 de abr. de 2024.

Proteus. PCB Design & Simulation Made Easy. Disponível em: https://www.labcenter.com/. Acesso em: 22 de ago de 2024.

NGROK. Overview. Disponível em: https://ngrok.com/docs/. Acesso em: 03 de set de 2024.

UN-ISDR - United Nations International Strategy for Disaster Reduction. Living with Risk: a global review of disaster reduction initiatives. Genebra, Suiça: Inter-Agency Secretariat International Strategy for Disaster Reduction (ISDR), 2004. 457p.

VASCONCELLOS, Andréa Araujo. Infraestrutura verde aplicada ao planejamento da ocupação urbana. Appris Editora e Livraria Eireli-ME, 2015

APÊNDICE A - Repositório do Projeto

Para fornecer uma transparência do que foi desenvolvido neste trabalho, o repositório do projeto foi disponibilizado online na plataforma GitHub.

O repositório contém os códigos-fonte mencionados no Apêndice B, além da simulação Proteus e as bibliotecas utilizadas, como também uma documentação adicional do SMPU.

O acesso pode ser feito através do: https://github.com/carlosaugustodpacheco/smpu-ufabc

APÊNDICE B - Códigos-fonte

Código-fonte do Arduino em C++

```
//  SISTEMA DE MONITORAMENTO PLUVIAL URBANO - TCC UFABC - CARLOS PACHECO
// VERSÃO 1 - Regra de negócio tratando valores captados por sensor e enviando aviso para um Web Server via Ethernet.
// SMPU MODELO PARA (CIDADE:SÃO PAULO/ REGIÃO: 1 / MÓDULO SMPU: 1 )
#include <LiquidCrystal.h>
#include <EtherCard.h>
//inicializa pins de valores do LCD
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
int resval = 0;  // armazena valor da resistencia gerada por sensor de nivel de agua
int respin = A3; // pin do sensor de nivel de água
int buzzer_Pin = 8; // pin do alarme sonoro
// Configuração de IP Estatico
static byte myip[] = { 169, 254, 199, 140 }; // IP do Web Server (169.254.199.140) - usar no navegador para acessar Web Server
static byte gwip[] = { 169, 254, 229, 192 }; // IP do Módulo Ethernet (169.254.229.192) - usar para configurar Módulo no Simulador Proteus
// MAC address do Arduino
static byte mymac[] = { 0x74, 0x69, 0x69, 0x2D, 0x30, 0x31 };
// tamanho do buffer da Ethernet
byte Ethernet::buffer[700];
// Pagina HTML para Situação Normal(armazenada na variavel PROGMEM)
const char situacaoNormal[] PROGMEM =
  "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r\n"
                    "<html>\r\n"
                    "<head>\r\n"
                    "<meta http-equiv='refresh' content='3'>\r\n"
                    "</head>\r\n"
                    "<body>\r\n"
                    "estado:sp;cidade:sao_paulo;regiao:1;smpu:1;nivel_da_agua:situacao_normal;\r\n"
                    "</body>\r\n"
                    "</html>\r\n";

// Pagina HTML para Alerta Leve(armazenada na variavel PROGMEM)
const char alertaLeve[] PROGMEM =
  "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r\n"
                    "<html>\r\n"
                    "<head>\r\n"
                    "<meta http-equiv='refresh' content='3'>\r\n"
                    "</head>\r\n"
                    "<body>\r\n"
                    "estado:sp;cidade:sao_paulo;regiao:1;smpu:1;nivel_da_agua:alerta_leve;\r\n"
                    "</body>\r\n"
                    "</html>\r\n";

// Pagina HTML para Alerta Moderado(armazenada na variavel PROGMEM)
const char alertaModerado[] PROGMEM =
  "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r\n"
                    "<html>\r\n"
```

```
                "<head>\r\n"
                "<meta http-equiv='refresh' content='3'>\r\n"
                "</head>\r\n"
                "<body>\r\n"
                "estado:sp;cidade:sao_paulo;regiao:1;smpu:1;nivel_da_agua:alerta_moderado;\r\n"
                "</body>\r\n"
                "</html>\r\n";

// Pagina HTML para Alerta Grave(armazenada na variavel PROGMEM)
const char alertaGrave[] PROGMEM =
  "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\nConnection: close\r\n\r\n"
                "<html>\r\n"
                "<head>\r\n"
                "<meta http-equiv='refresh' content='3'>\r\n"
                "</head>\r\n"
                "<body>\r\n"
                "estado:sp;cidade:sao_paulo;regiao:1;smpu:1;nivel_da_agua:alerta_grave;\r\n"
                "</body>\r\n"
                "</html>\r\n";

void setup() {
   // Configura o pino 8 como saída para o alarme sonoro
  pinMode(buzzer_Pin, OUTPUT);
  // configura numero de colunas e linhas do LCD
  lcd.begin(16, 2);
  // printa mensagem fixa no LCD.
  lcd.print("NIVEL DA AGUA:  ");
  // Comeca comunicação com Ethenert pelo tamanho do buffer e MAC address
  ether.begin(sizeof Ethernet::buffer, mymac, SS);
  // Configura IP estatico e gateway IP
  ether.staticSetup(myip, gwip);
}

void loop() {
  // Seta cursor para coluna 0 e linha 1
  lcd.setCursor(0, 1);
  resval = analogRead(respin); //Le o dado do pin3 e armazena na variavel resval
  // Maneja pacotes Ethernet e pega a posicionamento do dado
  word pos = ether.packetLoop(ether.packetReceive());

  // REGRAS DE NEGOCIO
  // Menor que 150ohm/Menor de 15cm = SITUACAO NORMAL
  if (resval<=150) {
    // Desliga o alarme sonoro
    digitalWrite(buzzer_Pin, LOW);
    // printa mensagem variavel no LCD.
    lcd.println("SITUACAO NORMAL");
    if (pos) {
      // Extrai o dado do buffer da Ethernet
      char *data = (char *)Ethernet::buffer + pos;
      // copia HTML da situacao normal
      memcpy_P(ether.tcpOffset(), situacaoNormal, sizeof situacaoNormal);
```

```
    // Envia informacao via HTTP para Web Server
    ether.httpServerReply(sizeof situacaoNormal - 1);
   }
  }
  // Entre 151ohm e 300ohm/ Entre 16cm e 30cm = ALERTA LEVE
  else if (resval>150 && resval<=300){
   // Liga o alarme sonoro
   digitalWrite(buzzer_Pin, HIGH);
   // printa mensagem variavel no LCD.
   lcd.println("ALERTA LEVE    ");
   if (pos) {
   // Extrai o dado do buffer da Ethernet
   char *data = (char *)Ethernet::buffer + pos;
   // copia HTML da situacao normal
   memcpy_P(ether.tcpOffset(), alertaLeve, sizeof alertaLeve);
   // Envia informacao via HTTP para Web Server
   ether.httpServerReply(sizeof alertaLeve - 1);
   }
  }
  // Entre 301ohm e 600ohm/ Entre 31cm e 60cm = ALERTA MODERADO
  else if (resval>300 && resval<=600){
   // Liga o alarme sonoro
   digitalWrite(buzzer_Pin, HIGH);
   // printa mensagem variavel no LCD.
   lcd.println("ALERTA MODERADO");
   if (pos) {
   // Extrai o dado do buffer da Ethernet
   char *data = (char *)Ethernet::buffer + pos;
   // copia HTML da situacao normal
   memcpy_P(ether.tcpOffset(), alertaModerado, sizeof alertaModerado);
   // Envia informacao via HTTP para Web Server
   ether.httpServerReply(sizeof alertaModerado - 1);
   }
  }
  // Acima de 601ohm/ Acima de 61cm = ALERTA GRAVE
  else if (resval>600){
   // Liga o alarme sonoro
   digitalWrite(buzzer_Pin, HIGH);
   // printa mensagem variavel no LCD.
   lcd.println("ALERTA GRAVE     ");
   if (pos) {
   // Extrai o dado do buffer da Ethernet
   char *data = (char *)Ethernet::buffer + pos;
   // copia HTML da situacao normal
   memcpy_P(ether.tcpOffset(), alertaGrave, sizeof alertaGrave);
   // Envia informacao via HTTP para Web Server
   ether.httpServerReply(sizeof alertaGrave - 1);
   }
  }
  // Seta delay de mensagem para 100ms
  delay(100);
}
```

Código-fonte da Função Lambda em Python

```
# SISTEMA DE MONITORAMENTO PLUVIAL URBANO - TCC UFABC - CARLOS PACHECO
# VERSÃO 1 - Regra de negócio da função Lambda para tratamento do IP Publico
# SMPU MODELO PARA (CIDADE:SÃO PAULO/ REGIÃO: 1 / MÓDULO SMPU: 1 )
import json
import boto3
import requests
from botocore.exceptions import NoCredentialsError, PartialCredentialsError

# Configurações SNS e DynamoDB
sns_client = boto3.client('sns', region_name='sa-east-1')
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('WaterLevel')

# Função Lambda Handler
def lambda_handler(event, context):
    url = 'https://7519-2804-14c-1a5-5724-28a1-4e74-dbb3-6fa.ngrok-free.app/smpu01/'

    # Buscar dados da URL (necessario alterar a url pra cada nova sessão do ngrok)
    try:
        response = requests.get(url)
        data = response.text

        # Extrair o nível da água
        nivel_da_agua = 'situacao_normal'
        for part in data.split(';'):
            if part.startswith('nivel_da_agua:'):
                nivel_da_agua = part.split(':')[1]
                break

        # Verificar o nível atual armazenado no DynamoDB
        try:
            # Ajuste a chave primária e o valor conforme seu esquema
            response = table.get_item(Key={'Id': 'nivel_agua'})
            stored_level = response.get('Item', {}).get('CurrentLevel', 'situacao_normal')

            if nivel_da_agua != stored_level:
                # Atualizar o DynamoDB com o novo nível
                table.put_item(Item={'Id': 'nivel_agua', 'CurrentLevel': nivel_da_agua})

                # Enviar alerta SNS
                if nivel_da_agua == 'alerta_leve':
                    message = 'Alerta leve: O nível da água está ligeiramente elevado.'
                elif nivel_da_agua == 'alerta_moderado':
                    message = 'Alerta moderado: O nível da água está significativamente elevado.'
                elif nivel_da_agua == 'alerta_grave':
                    message = 'Alerta grave: O nível da água está crítico!'

                sns_client.publish(
                    TopicArn='arn:aws:sns:sa-east-1:431433565792:SMPU01',
```

```
            Message=message,
            Subject='Alerta de Nível da Água'
        )

    except Exception as e:
        print(f'Erro ao acessar ou atualizar o DynamoDB: {str(e)}')
        return {
            'statusCode': 500,
            'body': json.dumps(f'Erro ao acessar ou atualizar o DynamoDB: {str(e)}')
        }

except Exception as e:
    print(f'Erro ao buscar dados: {str(e)}')
    return {
        'statusCode': 500,
        'body': json.dumps(f'Erro ao buscar dados: {str(e)}')
    }

return {
    'statusCode': 200,
    'body': json.dumps(f'Verificação concluída. Nível da água atual: {nivel_da_agua}')
}
```