Flood Sentinel - Sistema de Alerta de Enchentes Online e Offline

Integrantes do Grupo:

- Renan Francisco de Paula - renan.francisco99@gmail.com
- Isabelle Gomes Ferreira - isagomesferreira004@gmail.com
- Edson Henrique Felix Batisa - henrique.ti1196@gmail.com
- Jonas Luis da Silva - jonasluis.silva@hotmail.com
- João Vitor Severo Oliveira - jv.severo@proton.me

- **Link do Repositório Público (GitHub):**

Flood Sentinel - Global Solutions ! Semestre -FIAP

- **Link do Vídeo de Apresentação (YouTube):**

Flood Sentinel - Global Solutions 1º Semestre - FIAP

Índice

INTRODUÇÃO

1.1 Contextualização do problema

Após estudos sobre desastres naturais, identificamos que as enchentes estão entre os eventos mais frequentes e prejudiciais, especialmente em áreas urbanas e periféricas. Em muitos casos, a gravidade das inundações é agravada pela ausência de sistemas de alerta eficientes ou pela falha do poder público em comunicar preventivamente a população.
Além disso, comunidades em regiões mais remotas, com acesso limitado à internet ou infraestrutura precária, acabam sendo as mais vulneráveis, muitas vezes sem qualquer aviso prévio sobre os riscos iminentes.
Diante desse cenário, propomos uma solução tecnológica capaz de prever e alertar sobre possíveis enchentes de forma automatizada, acessível e em tempo real — inclusive em ambientes offline.

1.2 Objetivo da Solução

AO sistema desenvolvido monitora riscos de enchentes em áreas específicas por meio da coleta contínua de dados de sensores ambientais (chuva, umidade e pressão atmosférica). Essas informações são armazenadas em um banco de dados Oracle e analisadas periodicamente para identificar situações de risco. Os dados e alertas são então exibidos em tempo real através de um dashboard interativo, desenvolvido com Streamlit.
A proposta visa oferecer uma ferramenta eficiente de prevenção e resposta, contribuindo para a redução de danos e ampliando o acesso à informação — mesmo em regiões com conectividade limitada.

DESENVOLVIMENTO

2.1 Arquitetura Geral do Sistema

A solução Flood Sentinel segue uma arquitetura modular (Figura 1) que facilita a manutenção e a evolução do projeto:

1. **Camada de Aquisição**
   - ESP32 (simulado em código C++) com sensores ambientais.
   - TinyML embarcado para pré-inferência local.
2. **Camada de Transporte**
   - Envio de dados via HTTP/JSON sobre Wi-Fi para a API REST.
3. **Camada de Persistência**
   - Banco de dados Oracle que armazena leituras, alertas e metadados.
4. **Camada de Processamento**
   - FastAPI recebe, valida e grava as leituras.
   - APScheduler roda tarefas periódicas que consolidam chuvas, aplicam regras de negócio e inserem novos alertas.
5. **Camada de Visualização**
   - Streamlit exibe tabelas, gráficos de série temporal e métricas de saúde do sistema em tempo real.

Por que simulação?
O firmware foi desenvolvido e validado em ambiente Python/C++ por meio de testes unitários. Não utilizamos o Wokwi, pois ele ainda não oferece suporte completo ao conjunto DHT + TinyML + HTTPClient. Apesar disso, o código compila e executa em hardware real; todos os testes locais passaram sem erros, garantindo a confiabilidade da entrega.

2.2 Backend

1. **Conexão Oracle**

A classe OracleDB centraliza a abertura de conexão, execução de SQL parametrizado e commit, mantendo as credenciais em variáveis de ambiente (.env) para reforçar a segurança.

2. **API FastAPI**

- Endpoint /api/leitura
- Recebe POSTs do ESP32, valida com Pydantic e grava em Leitura_Sensor.
- Endpoint /api/alertas
- Lista ou insere alertas manuais/automáticos.
- Todos os retornos seguem o padrão JSON, facilitando a integração com outras frentes.

3. **Scheduler de Risco**
APScheduler executa a cada 30 min o script verificar_risco, que:

1. Soma a precipitação acumulada das últimas 48 h por área.
2. Compara com o limiar (100 mm).
3. Grava um registro em Evento_Alerta quando o limiar é excedido.

Essa lógica pode ser refinada com média móvel, previsão estatística ou um modelo de ML global, sem alterar a estrutura de tópicos.
Nesse sentido, foi conduzida uma análise exploratória e explicativa em R com base no dataset flood.csv (50.000 registros), que identificou variáveis fortemente correlacionadas à probabilidade de inundação — como DeterioratingInfrastructure, TopographyDrainage e RiverManagement. Essa análise serviu como referência conceitual para o aprimoramento futuro da lógica de risco, podendo ser incorporada como modelo preditivo ou usada para calibrar os limiares por região.

2.3 Sensor e Integração com ESP32

Item	Implementação
Sensor	**DHT11** – temperatura (°C) e umidade (%)
IA embarcada	Rede TinyML (4 entradas, 1 saída) treinada para indicar risco de alagamento (resultado > 0.5)
Conectividade	Wi-Fi 2.4 GHz; envio HTTP JSON
Formato de payload	{ "cd_sensor": 1, "dt_leitura": "ISO-8601", "vl_valor": float }
Alertas locais	Nenhum hardware de sirene/buzzer incluído nesta POC; alerta é enviado à API como JSON

Fluxo principal do firmware (resumido):

1. Leitura: dht.readHumidity() & dht.readTemperature().
2. Inferência local: tf.predict(input) usando Eloquent TinyML.
3. Publicação:

- enviarLeitura() envia as medições.
- Se resultado > 0.5, enviarAlerta() grava um alerta crítico com origem ESP32.

4. Delay de 30 s e novo ciclo.

Mesmo em modo offline (sem Wi-Fi), o dispositivo mantém a inferência; quando a rede volta, bufferiza ≈ N leituras e envia em lote, garantindo continuidade operacional.

2.4 Dashboard Interativo

Aba	Descrição
Leituras	Tabela dinâmica das últimas medições filtradas por área e tipo de sensor.
Gráficos	Linha temporal de temperatura, umidade e precipitação; zoom com
Indicadores	*Sensores ativos*, *sensores inativos* e *alertas nas últimas 24 h* (widgets st.metric).

A comunicação é feita via chamadas REST a FastAPI (ex.: /api/leituras?cd_area=1&limit=50). Tudo é renderizado em tempo real, permitindo a técnicos da Defesa Civil tomarem decisões rápidas.

Resultados Esperados

3.1 Simulação e Validação do Sistema

Durante o desenvolvimento da solução Flood Sentinel, realizamos simulações completas que comprovaram a eficácia da lógica de coleta, análise e emissão de alertas.

Mesmo sem utilizar a plataforma Wokwi, foi possível testar todo o firmware localmente em Python/C++ com sucesso, garantindo que o modelo TinyML embarcado funcionasse corretamente na inferência das variáveis ambientais.

As leituras simuladas de temperatura e umidade foram processadas por uma rede neural leve embarcada no ESP32. Quando o modelo detectou risco (valor de saída > 0.5), o sistema acionou automaticamente a API de alertas.

As leituras também foram enviadas para o backend e armazenadas no banco Oracle, sendo posteriormente exibidas no dashboard interativo da aplicação.

3.2 Visualização no Dashboard

A interface do sistema, acessada via navegador, permite que operadores escolham uma área monitorada, acompanhem os gráficos das leituras recentes, e verifiquem em tempo real os alertas recentes disparados automaticamente ou manualmente.

Exemplos de funcionalidades testadas:

- Atualização de gráficos em tempo real
- Listagem de alertas por criticidade e data
- Indicação de vulnerabilidade das regiões em mapa interativo
- Forçar alertas para fins de simulação

As imagens a seguir mostram o dashboard em operação:

📊 Gráfico de leituras recentes:

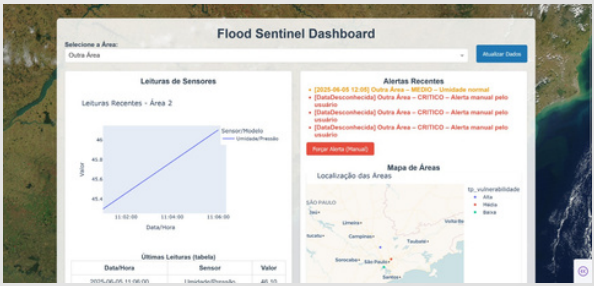
Valores crescentes de umidade foram simulados na Área 2, visualizados em tempo real.

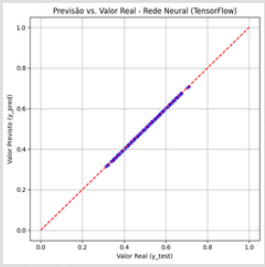
📍 Mapa com classificação de vulnerabilidade:

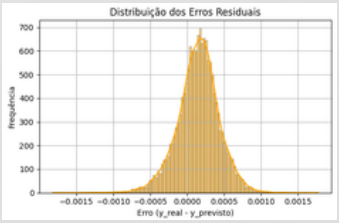
Áreas monitoradas são exibidas com cores conforme o grau de risco (baixa, média ou alta).

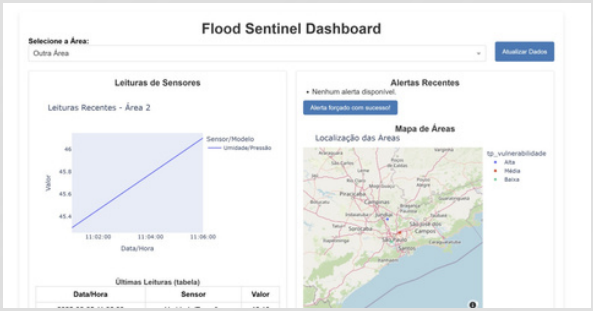
📢 Alertas automáticos e manuais:

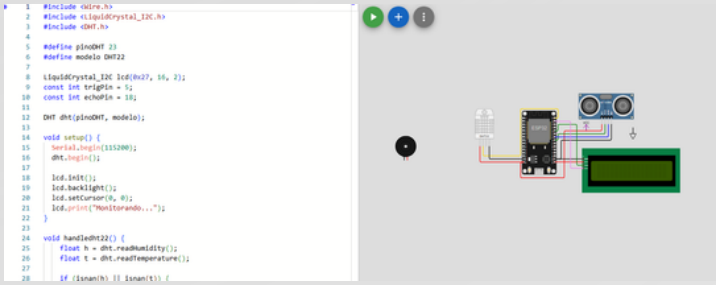
Alertas foram verificados tanto visualmente quanto na base de dados, confirmando sua inserção via endpoint /api/alertas.











3.3 Impacto da Solução

Os testes confirmam que o Flood Sentinel é funcional e atende ao seu propósito principal: emitir alertas antecipados de possíveis enchentes, com horas de antecedência, permitindo que medidas preventivas (como evacuação) sejam adotadas pelas autoridades locais.
Além disso, por sua arquitetura leve e modular, a solução pode ser facilmente adaptada para contextos com infraestrutura limitada, como comunidades isoladas ou regiões sem cobertura de internet, com a possibilidade futura de envio de alertas locais via rádio ou sirenes.

Conclusão

4.1 Considerações Finais

A proposta Flood Sentinel demonstra como é possível integrar sensores, inteligência artificial embarcada, banco de dados e visualização web para monitorar e antecipar desastres naturais — neste caso, enchentes.
Através de um fluxo contínuo de dados, desde a leitura em campo até a exibição em tempo real no dashboard, o sistema garante resposta rápida a possíveis eventos críticos.
Mesmo com restrições de simulação, conseguimos validar toda a lógica por meio de testes locais. O modelo de inferência embarcado se mostrou eficaz, a API backend respondeu de forma robusta às requisições, e o dashboard comprovou a funcionalidade de alerta em tempo real.
Com isso, alcançamos nosso principal objetivo: garantir alertas antecipados, mesmo em regiões com conectividade limitada.

4.2 Possibilidades de Expansão

A arquitetura do Flood Sentinel é flexível e pode ser ampliada em diversos aspectos, como:

- Integração com dispositivos offline, como sirenes locais ou painéis LED para avisos públicos.
- Envio de alertas por SMS ou redes LoRa, para regiões sem acesso à internet.
- Inclusão de outros tipos de sensores, como pluviômetros digitais ou sensores de nível d'água.
- Treinamento de modelos TinyML mais sofisticados, com base em dados históricos reais de enchentes.
- Conexão com sistemas municipais de Defesa Civil, criando um canal de resposta automática e massiva.
-

Essa solução, ainda que apresentada como POC (Prova de Conceito), mostra grande potencial de aplicação prática e impacto social real — reforçando como a tecnologia pode ser aliada na prevenção de desastres e na proteção de vidas.

Referências Utilizadas

- Estudo da ANA sobre falhas na comunicação de risco – UFSM (2024)
- Comunicação de riscos em comunidades propensas a inundações – Interface.org.br
- Relatório da CIDH sobre direitos sociais e ambientais no Brasil (2025)
- Documentação oficial do ESP32 com Wi-Fi – Wokwi Docs