XXI Escola Regional de Redes de Computadores (ERRC). DOI: 10.5753/errc.2024.4713

ARTIGO COMPLETO

MosqMon: um sistema distribuído para captura de faixas de áudio para treinamento de RNAs detectoras de mosquito Ae. aegypti

A distributed system for capturing audio tracks for training Ae. aegypti mosquito detection ANNs

Yang da Fontoura Rodrigues • yangrodrigues.aluno@unipampa.edu.br
Universidade Federal do Pampa (Unipampa)
Rafael Nogueira Rodrigues • rafaelnogueira.aluno@unipampa.edu.br
Universidade Federal do Pampa (Unipampa)
Kayuã Oleques Paim • kayua.paim@inf.ufrgs.br
Universidade Federal do Rio Grande do Sul (UFRGS)
Diego Kreutz • diegokreutz@unipampa.edu.br
Universidade Federal do Pampa (Unipampa)
Weverton Cordeiro • weverton.cordeiro@inf.ufrgs.br
Universidade Federal do Rio Grande do Sul (UFRGS)
Rodrigo Brandão Mansilha • rodrigomansilha@unipampa.edu.br
Universidade Federal do Pampa (Unipampa)

RESUMO. O projeto Aedes Vigilance visa ampliar a vigilância do mosquito *Aedes aegypti*, vetor de doenças como Zika, chikungunya e dengue, monitorando seu som característico usando dispositivos de baixo custo, como smartphones. Uma rede neural artificial (RNA) foi desenvolvida para identificar o mosquito a partir do som das asas, mas testada apenas em ambientes controlados, devido à falta de *datasets* adequados. Este trabalho propõe o MosqMon, um sistema distribuído que coleta amostras de áudio em ambientes não controlados, visando gerar *datasets* realistas para o treinamento de RNAs. A arquitetura segue o modelo agente/gerente, no qual *agentes* capturam amostras de áudio e enviam ao gerente, responsável pelo armazenamento das amostras para futuros treinamentos ou classificações automatizadas. A implementação funcional do MosqMon demonstrou a viabilidade do sistema, que se espera fornecer dados essenciais para melhorar a vigilância do *Aedes aegypti*.

ABSTRACT. The Aedes Vigilance project aims to enhance the surveillance of the Aedes aegypti mosquito, the vector of diseases like Zika, chikungunya, and dengue, by using low-cost devices such as smartphones to monitor its characteristic sound. A neural network (RNA) was developed to identify the mosquito based on the sound of its wingbeats, though it has only been tested in controlled environments due to the lack of appropriate datasets. This work proposes MosqMon, a distributed monitoring system designed to collect audio samples in uncontrolled environments, such as universities and residences, to generate realistic datasets for RNA training. The architecture follows an agent/manager model, where agents capture audio samples and send them to a manager, which stores them for training or automated classification. The system's viability has been demonstrated through a functional implementation, and MosqMon is expected to provide essential data for improving RNA-based mosquito surveillance.

PALAVRAS-CHAVE: Sistema Distribuído • Monitoramento • Mosquito • Aedes aegypti
KEYWORDS: Distributed System • Monitoring • Mosquitoe • Aedes aegypti

1 Introdução

O animal mais letal da Terra é o mosquito[1, 2]. O Aedes aegypti, em particular, gera perdas econômicas globais na ordem de bilhões de dólares [3–5]. No Brasil, as enfermidades transmitidas pelo mosquito Aedes aegypti, como Zika, Chikungunya, dengue e febre amarela, têm gerado graves problemas de saúde pública há décadas, e ainda estão longe de serem erradicadas. De acordo com dados do Ministério da Saúde do Brasil [6], apenas no primeiro semestre de 2024, foram registrados mais de 6,3 milhões de casos prováveis de dengue.

Para combater o mosquito, medidas simples, como a eliminação de criadouros de água parada em áreas exter-

Yang e Rafael são Bacharelandos em Ciência da Computação na Unipampa. Kayuã é doutorando na UFRGS. Diego e Rodrigo são professores na Unipampa e Weverton é professor na UFRGS.

nas, poderiam ser eficazes, mas requerem um elevado nível de conscientização e engajamento da população. Por outro lado, soluções mais avançadas, como a dedetização e a liberação de mosquitos geneticamente modificados, demandam menor participação popular, porém são relativamente mais onerosas, enfrentam importantes barreiras regulatórias e podem apresentar efeitos colaterais indesejáveis, dificultando sua implementação indiscriminada em grande escala [7, 8]. Portanto, torna-se essencial identificar as áreas de maior risco para a aplicação de campanhas de conscientização e de soluções tecnológicas de maneira localizada e financeiramente viável. A vigilância dessas áreas críticas tem se baseado, até o momento, no monitoramento da disseminação de doenças transmitidas pelos mosquitos, uma abordagem a *posteriori*. Há uma necessidade urgente de desenvolver métodos de vigilância dessas áreas críticas a *priori*, isto é, antes que ocorra a disseminação das doenças.

O projeto Aedes Vigilance visa ampliar o monitoramento do mosquito Aedes aegypti através da captura de seu som natural, utilizando dispositivos comuns. Ele é útil em locais com populações vulneráveis, como grávidas, bebês e idosos, além de áreas públicas com grande fluxo de pessoas, como hospitais e escolas. Contudo, sua proposta de valor está no monitoramento sistêmico: uma abordagem mais eficaz e abrangente de vigilância a *priori*, que as soluções atuais de vigilância a *posteriori*.

Pesquisas recentes desenvolveram uma rede neural artificial (RNA) capaz de identificar o mosquito a partir do som de suas asas, utilizando smartphones de entrada [9, 10]. Porém, essa tecnologia tem sido testada em ambientes controlados (*i.e.*, laboratório), onde ruídos externos (*e.g.*, trânsito, conversas e outros animais) são minimizados. Para viabilizar seu uso no cotidiano em smartphones, bem como em outros dispositivos, como alto-falantes inteligentes (*e.g.*, Amazon Echo), TVs conectadas, relógios inteligentes e dispositivos vestíveis, é crucial treinar e testar a RNA em cenários reais, levando em consideração os ruídos presentes em ambientes comuns.

Neste trabalho apresentamos o MosqMon: um sistema distribuído de monitoramento para capturar faixas de áudio úteis no futuro treinamento de RNAs detectoras de mosquito Ae. aegypti. A arquitetura do sistema segue o modelo tradicional de gerenciamento, composto por agentes e gerentes. Os agentes, com capacidade decisória simplificada, são responsáveis pela coleta de faixas de áudio. Os agentes devem ser acoplados a algum tipo de armadilha de mosquito para permitir que as faixas sejam confrontáveis com outras evidências (*e.g.*, presença de mosquito). Os gerentes supervisionam um conjunto de um ou mais agentes, assegurando a curadoria adequada dos dados coletados. O sistema deve contribuir para a construção de *datasets* em ambientes variados, permitindo um treinamento mais realista das RNAs desenvolvidas no projeto Aedes Vigilance. A Figura 1 mostra o escopo da MosqMon contextualiza em projeto Aedes Vigilance.

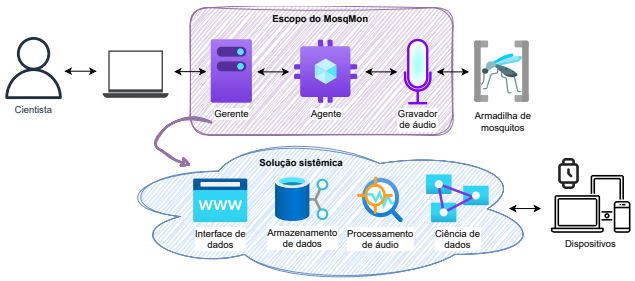

Figura 1. Escopo do sistema.

A literatura apresenta várias técnicas para o monitoramento de mosquitos [11, 12]. Em contraste com soluções completas (*i.e.*, armadilha, sensor e sistema integrados) e sofisticadas, como aquelas baseadas em sensores infravermelho [13], nossa abordagem concentra-se em um sistema distribuído baseado em software e hardwares livres que pode ser integrado a diferentes tecnologias de armadilhas, preferencialmente de baixo custo de aquisição e manutenção.

O restante deste trabalho está organizado em três seções: a Seção 2 descreve a arquitetura do MosqMon, a Seção 3 discute a implementação e os resultados das avaliações funcionais, e a Seção 4 aborda as conclusões e perspectivas para trabalhos futuros.

2 MosqMon

Nesta seção são apresentados os requisitos e as especificações da arquitetura MosqMon.

2.1 Análise de Requisitos

Realizamos um elicitação de requisitos funcionais para a MosqMon. Para cada requisito funcional (RF) anotamos um identificador, nome, descrição e nível de prioridade. As prioridades foram definidas seguindo a escala MoSCoW [14], que possui 4 níveis: *Must Have*, requisito necessário para que o sistema funcione corretamente; *Should Have*, requisito útil porém não crítico; *Could Have*, requisito que seria bom ter, porém não é necessário para o funcionamento básico do sistema; e, *Won't Have*, requisitos que não serão implementados no momento atual, porém podem ser considerados em versões futuras. A Tabela 1 elenca os requisitos funcionais dos usuários com prioridade *Must*.

Tabela 1. Requisitos Funcionais prioritários.

Id. Pri.	Nome	Descrição
RF01 Must	Iniciar gravação	Como cientista, quero solicitar que o agente inicie ciclos de gravações de faixas de áudio, definindo duração, dispositivo de áudio e quantidade de repetições.
RF02 Must	Interromper gravação	Como cientista, quero interromper uma sessão em andamento de gravação de faixas de áudio.
RF03 Must	Listar faixas de áudio	Como cientista, quero listar as faixas de áudio coletados por um agente.
RF04 Must	Obter faixa de áudio	Como cientista, quero solicitar a transferência de faixas áudios do agente para o gerente.
RF05 Must	Obter situação do agente	Como cientista, quero obter a situação do agente (*e.g.*, parado, gravando).
RF06 Must	Apagar faixa de áudio	Como cientista, quero apagar uma faixa de áudio coletado pelo agente.
RF07 Must	Renomear faixa de áudio	Como cientista, quero renomear uma faixa de áudio coletado pelo agente.

Tabela 2. Todos requisitos Não Funcionais (RNFs).

Id. Pri.	Requisito	Descrição
RNF01 Must	Eficiência computacional	O sistema deve rodar de forma estável em computadores de nível de entrada, garantindo um uso eficiente dos recursos de processamento e memória.
RNF02 Must	Abertura	O sistema deve utilizar e aplicar soluções baseadas em software livre ou aberto, facilitando sua adoção e modificação por outros desenvolvedores.
RNF03 Must	Manutenabilidade	O código-fonte deve ser modular e bem documentado para permitir atualizações e correções de maneira eficiente no futuro.
RNF04 Should	Interoperabilidade	O sistema deve ser compatível com outros sistemas que possam utilizar ou fornecer informações, assegurando integração e troca de dados sem falhas.
RNF05 Should	Escalabilidade	O sistema deve ser capaz de atender a um aumento expressivo no número de usuários ou dispositivos, distribuídos geograficamente e operando em diferentes organizações, sem degradação significativa de desempenho.
RNF06 Must	Segurança: confidencialidade	A comunicação entre o gerente e os agentes do sistema deve ser protegida para garantir que os dados trocados sejam acessíveis apenas para as entidades autorizadas.
RNF07 Must	Segurança: integridade de fonte	A origem dos dados (por exemplo, as armadilhas) deve ser validada para assegurar que as informações recebidas são autênticas e provenientes de fontes legítimas.
RNF08 Must	Segurança: integridade de dados	Os dados gerados, como faixas de áudio capturadas pela armadilha, devem ser protegidos contra modificações não autorizadas durante a transmissão e armazenamento.
RNF09 Could	Segurança: disponibilidade	Os dados coletados pelo sistema devem estar disponíveis para os usuários autorizados sempre que necessário, garantindo acessibilidade contínua.
RNF10 Could	Privacidade	Os dados pessoais e sensíveis coletados pelo sistema devem ser tratados de forma a proteger a privacidade dos usuários, seguindo regulamentações como a LGPD, e minimizando a coleta de informações que não sejam estritamente necessárias para o propósito do projeto.

A Tabela 2 apresenta todos os requisitos não funcionais. É essencial que o sistema possua eficiência computacional para reduzir os custos de aquisição de hardware e consumo de energia. Além disso, o sistema deve ser aberto, a fim de minimizar despesas e aumentar o impacto, especialmente em regiões mais vulneráveis. A manutenibilidade é crucial para garantir a sustentabilidade do projeto ao longo do tempo. A interoperabilidade será necessária no futuro para permitir que a MosqMon possa ser adotado para outras iniciativas para além do projeto Aedes Vigilance, por exemplo, mas não é fundamental neste estágio de desenvolvimento do projeto.A escalabilidade, em suas diversas vertentes, será fundamental para atender às demandas futuras de crescimento, dispersão geográfica e integração entre diferentes organizações públicas e privadas.

Em relação aos requisitos não funcionais (RNFs) de segurança, é essencial contextualizar o modelo de ameaça adotado. Consideramos seguras as primitivas criptográficas utilizadas, como funções de *hash* (*e.g.*, SHA256) e MAC (*e.g.*, HMAC), além de protocolos como o TLS 1.3. Confiamos na integridade dos administradores do sistema e das armadilhas, ou seja, a possibilidade de ameaças internas (*insider threats*) não é contemplada neste contexto.

Como delimitação do escopo, deixamos para trabalhos futuros aspectos relacionados à segurança com foco em disponibilidade e privacidade. A disponibilidade das armadilhas deve ser garantida caso o sistema atinja um nível de popularização significativo. Já a disponibilidade das faixas pode ser alcançada no futuro com a implementação de uma solução sistêmica robusta. Em relação à privacidade, é necessária uma análise mais aprofundada para prevenir o vazamento de informações sensíveis captadas nos áudios, além de assegurar que os usuários sejam devidamente conscientizados.

2.2 Arquitetura

Para atender os requisitos funcionais e não funcionais descritos na subseção anterior propomos uma arquitetura, denominada MosqMon. Ela é composta por uma (ou mais) armadilha(s) inteligentes para captura de mosquitos e um sistema de monitoramento distribuído. A arquitetura distribuída é baseada no modelo clássico de Agente/Gerente de monitoramento, que se comunicam por meio de um protocolo específico.

A Figura 2 apresenta a arquitetura da MosqMon em

notação C4model[1]. Ela é composta por dois elementos que são interligados por um protocolo de comunicação, em linha com arquiteturas Agente/Gerente de monitoramento. O usuário pode comandar o sistema diretamente através do gerente. O gerente também pode ser conectado a uma federação, atuando localmente (*e.g.*, cidade de Alegrete-RS) e contribuindo regional, nacional ou globalmente para a coleta de dados.

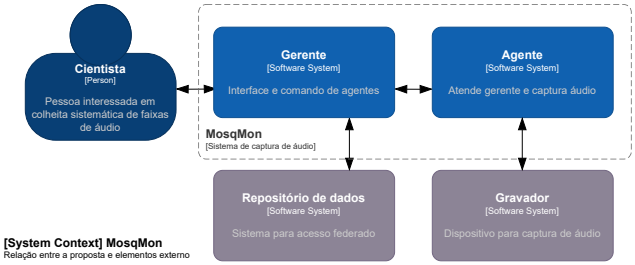

Figura 2. Arquitetura MosqMon

O gerente recebe instruções do usuário por meio de uma interface de usuário, que pode ser baseada em linha de comando (CLI) ou gráfica (GUI), e utiliza o protocolo de comunicação para comandar o agente. O agente, por sua vez, segue o protocolo para atender às solicitações do gerente, controlando o gravador para capturar o áudio de mosquitos. Esse gravador pode ser integrado a uma armadilha para mosquitos, aumentando as chances de obter amostras.

A capacidade decisória do agente é intencionalmente mínima, privilegiando um sistema de inteligência centralizada no gerente do MosqMon. Essa arquitetura reduz o custo de processamento no agente, aumenta a robustez do sistema por ser mais simples do que uma abordagem de decisão distribuída e oferece escalabilidade, permitindo uma organização hierárquica de maneira eficiente. O agente pode ser configurado com diferentes capacidades de armazenamento, de forma a equilibrar custos (relacionados ao armazenamento e à comunicação) e cenários de tolerância a falhas (seja na comunicação ou no hardware), dependendo das demandas operacionais.

O protocolo de comunicação coordena o fluxo de dados entre gerente e cada agente seguindo um modelo *Unicast*. Especificamente, ele oferece métodos para gerenciar os arquivos de gravação (listar, renomear, apagar, etc.), iniciar ciclos de gravação (definindo a duração, o dispositivo de áudio e o número de repetições), interromper gravações, e consultar o status das gravações em andamento. Toda a comunicação é criptografada utilizando certificados, que são trocados de maneira segura por meio de um canal auxiliar protegido. A integridade das faixas de áudio é verificada por meio de funções de resumo criptográfico (*hash*), permitindo verificar se os arquivos foram (ou não) alterados durante a transmissão.

3 Prova de Conceito

3.1 Protótipo

A Figura 3 mostra o protótipo implementado da MosqMon. Ela foi instanciada no Laboratório de Estudos Avançados em Computação (LEA[2]. Como armadilha, usamos o dispositivo Mosquito Yg-5623 Ovo De Dinossauro Bivolt da fabricante Nsbao. Para permitir o teste de múltiplos dispositivos de gravação de áudio, substituímos a parte inferior da armadilha por um componente maior. O computador da esquerda atua como agente de monitoramento e o da direita, como gerente de monitoramento (Kernel Linux 6.8.0, CPU AMD 5500U, 6GB memória disponível e SSD NVME).

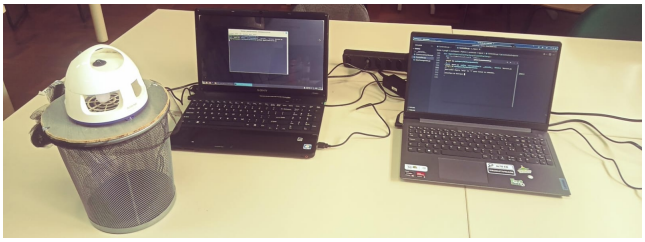

Figura 3. Protótipo do MosqMon

O gerente foi implementado de forma simplificada para testar o protocolo e o funcionamento do agente de monitoramento. Sua interface em linha de comando oferece comandos básicos e eficazes, como: `START_MONITORING` para iniciar um ciclo de gravação com parâmetros configuráveis; `STOP_MONITORING` para interromper o ciclo; `LIST` para listar os arquivos de áudio; `PULL_AUDIO` para transferir arquivos para o servidor; `STATUS` para verificar o status da gravação e da transferência; `DELETE` para excluir arquivos; `RENAME` para renomear arquivos; e `CLOSE_CONNECTION` para encerrar a conexão se não houver tarefas pendentes.

O protocolo de comunicação entre o gerente e o agente, bem como a interface do serviço, foram desenvolvidos em Python, utilizando a biblioteca Twisted para implementar um protocolo de rede personalizado com suporte a SSL/TLS. As classes de protocolo herdaram de `Protocol`, usando métodos como `connectionMade`, `dataReceived`, `transport` e `connectionLost` para gerenciar a comunicação. Classes como `Service-`

[1] https://c4model.com/

[2] https://sites.unipampa.edu.br/lea

`ControllerProtocol` e `AgentControllerProtocol` foram usadas para dados UTF-8, enquanto `Service-DataTransporter` e `AgentDataTransporter` cuidam da transferência de dados brutos, como arquivos WAV. A interface permite o controle remoto do agente, incluindo gerenciamento de gravações e transferência de arquivos, com verificação de integridade dos dados via SHA-256, usando a biblioteca `hashlib`.

O gravador de áudio foi implementado em C com foco em alto desempenho, utilizando a biblioteca `SDL2` para simplificar a interface com o sistema operacional e a `sndfile` para leitura e gravação de arquivos de áudio. O código é dividido em quatro componentes: o *parser*, que processa os argumentos de linha de comando; o gravador, que utiliza a `SDL2` para capturar dados de áudio em um *buffer*; o escritor, que usa a `sndfile` para salvar os arquivos de áudio em formato WAV com metadados; e o contexto principal, que coordena essas operações.

3.2 Testes

Para cada requisito funcional do usuário com prioridade *Must*, implementamos pelo menos um teste funcional para o gravador (teste unitário) e outro para o sistema como um todo (teste de ponta a ponta). Utilizamos a bibliotecas `Pytest` para garantir a reprodutibilidade dos testes de forma eficiente. A Figura 4 apresenta os resultados detalhados dessas verificações.

Instrumentamos nossos testes para avaliar o desempenho do nosso protótipo. Na execução da operação `PULL_AUDIO` para transferir uma faixa de áudio (1 segundo, 176,kB) o sistema levou 3,3098 segundos, e teve uso médio de 3,2% da CPU e 22,4 MB de memória.

3.3 Discussão

Observamos uma aprovação consistente em todos os testes funcionais, tanto unitários quanto de ponta a ponta, aplicados ao nosso protótipo, o que comprova a viabilidade da MosqMon. Em relação aos requisitos não funcionais, implementamos um gravador em C e um sistema em Python, priorizando a eficiência computacional (RNF01). Com base nos resultados da avaliação de desempenho, acreditamos que o sistema possa ser executado em microcomputadores de entrada, como o Raspberry Pi ou dispositivos similares. Futuramente, esperamos otimizar o código para que possa rodar em plataformas ainda mais simplificadas, como o Arduino.

O software foi disponibilizado e devidamente documentado, atendendo aos requisitos de abertura (RNF02) e manutenabilidade (RNF03). À medida que o sistema for adotado, planejamos desenvolver um API Gateway para garantir a interoperabilidade (RNF04). Considerando o modelo de ameaça proposto, adotamos TLS 1.3 para garantir a confidencialidade da comunicação (RNF06), o uso de certificados para assegurar a integridade da fonte (RNF07) e uma função *hash* considerada segura para atender à integridade dos dados (RNF08). Questões relacionadas à interoperabilidade (RNF04), escalabilidade (RNF05), disponibilidade (RNF09) e à privacidade (RNF10) são deixados para trabalhos futuros.

```
yangfr in protocolo/protocolo on  main [✗!?]
→ ./run_tests.sh
                                    RF01 -- Iniciar Gravação
========================================= test session starts ==========================================
collected 4 items

tests/system_tests_RF01.py::test_system_start_monitoring PASSED                                  [ 25%]
tests/agent_tests_RF01.py::test_agent_audio_recording_with_3_repetitions PASSED                  [ 50%]
tests/agent_tests_RF01.py::test_agent_audio_recording_with_float_number_param PASSED             [ 75%]
tests/agent_tests_RF01.py::test_agent_audio_recording_with_time_less_than_1 PASSED               [100%]

========================================== 4 passed in 17.97s ==========================================
                                    RF02 -- Interromper Gravação
========================================= test session starts ==========================================
collected 2 items

tests/system_tests_RF02.py::test_system_stop_monitoring PASSED                                   [ 50%]
tests/agent_tests_RF02.py::test_agent_stop_audio_recording PASSED                                [100%]

========================================== 2 passed in 10.54s ==========================================
                                    RF03 -- Obter Faixa de Áudio
========================================= test session starts ==========================================
collected 3 items

tests/system_tests_RF03.py::test_system_pull_audio PASSED                                        [ 33%]
tests/system_tests_RF03.py::test_system_get_audio_file_with_incorrect_name PASSED                [ 66%]
tests/system_tests_RF03.py::test_system_data_integrity PASSED                                    [100%]

========================================== 3 passed in 11.36s ==========================================
                                    RF04 -- Obter Situação do Agente
========================================= test session starts ==========================================
collected 1 item

tests/system_tests_RF04.py::test_system_status PASSED                                            [100%]

========================================== 1 passed in 2.37s ===========================================
                                    RF05 -- Listar Faixas de Áudio
========================================= test session starts ==========================================
collected 1 item

tests/system_tests_RF05.py::test_system_list_audio_files PASSED                                  [100%]

========================================== 1 passed in 2.37s ===========================================
                                    RF06 -- Apagar Faixa de Áudio
========================================= test session starts ==========================================
collected 2 items

tests/system_tests_RF06.py::test_system_remove_audio_file PASSED                                 [ 50%]
tests/system_tests_RF06.py::test_system_remove_file_with_incorrect_name PASSED                   [100%]

========================================== 2 passed in 4.77s ===========================================
                                    RF07 -- Renomear Arquivo de Áudio
========================================= test session starts ==========================================
collected 2 items

tests/system_tests_RF07.py::test_system_rename_audio_file PASSED                                 [ 50%]
tests/system_tests_RF07.py::test_system_rename_file_with_incorrect_name PASSED                   [100%]

========================================== 2 passed in 4.73s ===========================================
                                    RF -- Validar Entrada do Usuário
========================================= test session starts ==========================================
collected 5 items

tests/system_input_validation.py::test_system_missing_params_error PASSED                        [ 20%]
tests/system_input_validation.py::test_system_invalid_input_error PASSED                         [ 40%]
tests/agent_input_validation.py::test_agent_file_not_found_error PASSED                          [ 60%]
tests/agent_input_validation.py::test_agent_invalid_input_error PASSED                           [ 80%]
tests/agent_input_validation.py::test_agent_missing_params_error PASSED                          [100%]

========================================== 5 passed in 5.52s ===========================================
yangfr in protocolo/protocolo on  main [✗!?] took 1m 1,2s
→
```

Figura 4. Resultados dos testes funcionais.

4 Considerações Finais

O MosqMon é um sistema distribuído para coleta de faixas de áudios úteis para treinamento de RNAs detectoras de mosquitos aedes aegypit. Ele é o passo inicial em direção a plataformas de monitoramento abertas e acessíveis para combater o animal mais letal do planeta.

Planejamos aprimorar os dispositivos de sensoriamento com a integração de sensores visuais para validar a detecção de mosquitos e dispositivos de atuação, como controle de iluminação e ventiladores. Também investigaremos o impacto do tamanho do recipiente na captação do som dos mosquitos. Além disso, será necessário abordar os desafios relacionados aos requisitos não funcionais restantes.

Declarações complementares

Financiamento

Este estudo foi financiado em parte pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Este estudo também foi financiado em parte pela Fundação de Amparo à Pesquisa do Rio Grande do Sul (FAPERGS) - ARD 10/2020 (21/2551-0000739-7), ARD/ARC 10/2021 (22/2551-0000603-5), Projeto 02/2022 Inova Clusters (22/2551-0000841-0), e Projeto 06/2021 RITEs-RS (22/2551-0000390-7). Também recebemos financiamento do Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brasil (CNPq) - Processo 314506/2023-3. Também recebemos apoio da Nvidia – Academic Hardware Grant.

Disponibilidade de dados e materiais adicionais

O código-fonte para execução e testes funcionais do MosqMon estão disponíveis publicamente [15].

Referências

1 Centers for Disease Control and Prevention. *Fighting the World's Deadliest Animal*. Accessed: 2024-10-17. 2022. Disponível em: https://www.cdc.gov/global-health/impact/fighting-the-worlds-deadliest-animal.html.

2 ISGlobal. *Mosquito: el animal más letal del mundo*. Accessed: 2024-10-17. 2023. Disponível em: https://www.isglobal.org/en/-/mosquito-el-animal-mas-letal-del-mundo.

3 Bradshaw, C. J. *et al.* Massive yet grossly underestimated global costs of invasive insects. *Nature communications*, Nature Publishing Group UK London, v. 7, n. 1, p. 12986, 2016.

4 Diagne, C. *et al.* High and rising economic costs of biological invasions worldwide. *Nature*, Nature Publishing Group UK London, v. 592, n. 7855, p. 571–576, 2021.

5 Ahmed, D. A. *et al.* Managing biological invasions: the cost of inaction. *Biological Invasions*, Springer, v. 24, n. 7, p. 1927–1946, 2022.

6 Saúde, M. da. *Painel de Monitoramento das Arboviroses — Ministério da Saúde*. 2024. https://www.gov.br/saude/pt-br/assuntos/saude-de-a-a-z/a/aedes-aegypti/monitoramento-das-arboviroses. Acessado em: 19 de setembro de 2024.

7 Panjwani, A.; Wilson, A. What is stopping the use of genetically modified insects for disease control? *PLoS Pathogens*, Public Library of Science San Francisco, CA USA, v. 12, n. 10, e1005830, 2016.

8 Legros, M. *et al.* Gene drive strategies of pest control in agricultural systems: Challenges and opportunities. *Evolutionary applications*, Wiley Online Library, v. 14, n. 9, p. 2162–2178, 2021.

9 Paim, K. O. *et al.* Acoustic identification of Ae. aegypti mosquitoes using smartphone apps and residual convolutional neural networks. *Biomedical Signal Processing and Control*, v. 95, p. 106342, 2024. ISSN 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2024.106342. Disponível em: https://www.sciencedirect.com/science/article/pii/S1746809424004002.

10 Fernandes, M. S.; Cordeiro, W.; Recamonde-Mendoza, M. Detecting Aedes aegypti mosquitoes through audio classification with convolutional neural networks. *Computers in Biology and Medicine*, v. 129, p. 104152, 2021. ISSN 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2020.104152. Disponível em: https://www.sciencedirect.com/science/article/pii/S0010482520304832.

11 Spitzen, J.; Takken, W. Keeping track of mosquitoes: a review of tools to track, record and analyse mosquito flight. *Parasites Vectors*, v. 11, p. 123, 2018. Received: 03 October 2017; Accepted: 21 February 2018; Published: 02 March 2018. DOI: 10.1186/s13071-018-2735-6. Disponível em: https://doi.org/10.1186/s13071-018-2735-6.

12 Javed, N.; Paradkar, P. N.; Bhatti, A. An overview of technologies available to monitor behaviours of mosquitoes. *Acta Tropica*, v. 258, p. 107347, 2024. ISSN 0001-706X. DOI: https://doi.org/10.1016/j.actatropica.2024.107347. Disponível em: https://www.sciencedirect.com/science/article/pii/S0001706X24002298.

13 Lai, Z. *et al.* Development and evaluation of an efficient and real-time monitoring system for the vector mosquitoes, Aedes albopictus and Culex quinquefasciatus. *PLOS Neglected Tropical Diseases*, Public Library of Science, v. 16, n. 9, p. 1–14, set. 2022. DOI: 10.1371/journal.pntd.0010701. Disponível em: https://doi.org/10.1371/journal.pntd.0010701.

14 Kravchenko, T.; Bogdanova, T.; Shevgunov, T. Ranking Requirements Using MoSCoW Methodology in Practice. *In*: Silhavy, R. (Ed.). *Cybernetics Perspectives in Systems*. Cham: Springer International Publishing, 2022. P. 188–199. ISBN 978-3-031-09073-8.

15 Rodrigues, Y. d. F.; Mansilha, R. B. *MosqMon: Intelligent Mosquito Trap System*. 2024. https://github.com/Y4ngfr/MosqMon. Accessed: 2024-10-21.