Sistema para monitoramento e detecção de anomalias no comportamento de gatos domésticos

Gabrielle Brambilla, Prof. Me. José Antônio Oliveira de Figueiredo

Instituto Federal de Educação, Ciência e Tecnologia Sul-rio-grandense (IFSul)
99064-440 – Passo Fundo – RS – Brazil

gabriellebrambilla.pf190@academico.ifsul.edu.br,
josefigueiredo@ifsul.edu.br

Abstract. *Este artigo apresenta um protótipo para coleta e análise de dados comportamentais de gatos domésticos, com o objetivo de obter um modelo do comportamento para identificação de anomalias, as quais podem estar relacionadas com o seu estado de saúde e bem-estar. Para isso, foram utilizados componentes de IoT para o monitoramento da frequência de alimentação, hidratação e excreção do felino. O processamento e armazenamento dessas informações foi realizado na nuvem AWS, onde o algoritmo Isolation Forest foi implantado para detecção de anomalias comportamentais. Os resultados obtidos demonstraram que o sistema proposto possui o potencial de auxiliar os tutores nesse acompanhamento cotidiano, contribuindo para manter a qualidade de vida do animal.*

Resumo. *This article presents a prototype for collecting and analyzing behavioral data of domestic cats, aiming to model their behavior and identify anomalies that may be related to their health and well-being. To achieve this, IoT components were used to monitor the frequency of feeding, hydration, and excretion activities. The processing and storage of this information were performed in the AWS cloud, where the Isolation Forest algorithm was implemented to detect behavioral anomalies. The results demonstrated that the proposed system has the potential to assist cat owners in daily monitoring, contributing to maintaining the animal's quality of life.*

1. Introdução

A saúde, qualidade de vida e longevidade dos *pets* é uma preocupação que vem crescendo ao decorrer dos anos, por conta do aumento na importância do relacionamento emocional entre seres humanos e seus animais de estimação [COMAC 2023]. Em relação aos gatos, apesar de também seguirem essa tendência e serem uma das espécies mais populares para adoção, ainda há um desconhecimento sobre as razões e consequências dos seus comportamentos. O conhecimento desses padrões comportamentais, entretanto, é um meio pelo qual os tutores teriam para detectar mudanças e poder associá-las com o estado atual de saúde do felino, além de ser uma ferramenta a mais para diagnósticos e tratamento contra muitas doenças [Scholten 2017].

Nesse cenário, o problema proposto para esta pesquisa é a obtenção de um modelo padrão de comportamento para um gato doméstico, de modo que anormalidades que possam impactar na saúde e bem-estar do animal sejam identificadas. Sendo assim, esse desafio é particularmente importante para que as mudanças nos hábitos diários sejam percebidas antes de impactarem na saúde do *pet*, ou antes que as complicações relacionadas com esses sintomas se agravem sem a procura por ajuda profissional.

A partir disso, foram propostos como objetivo do trabalho a implementação de um protótipo para automatização de coleta de dados e análise de ações cotidianas. Para tanto, o monitoramento do comportamento do gato teve como foco a frequência de três atividades que possuem uma forte relação com diversas doenças e possíveis complicações no quadro de bem-estar do felino: alimentação, hidratação e uso da caixa de areia. O sistema foi integrado com a AWS, de modo a realizar o armazenamento e processamento utilizando serviços da nuvem, para que, então, seja realizada uma análise das informações comportamentais obtidas, e, posteriormente, a notificação em caso de detecção de anomalias.

Este artigo está organizado da seguinte forma: a seção 2 apresenta o embasamento teórico para o trabalho. Em seguida, a seção 3 demonstra os procedimentos metodológicos planejados para desenvolvimento do protótipo. A seção 4 expõe os resultados obtidos em relação ao protótipo e aos testes executados, com base na metodologia apresentada. Por fim, a última seção aborda as conclusões finais, com a discussão sobre os aspectos percebidos com o desenvolvimento desse projeto.

2. Referencial Teórico

Nesta seção será apresentado o embasamento teórico para os principais conceitos e tecnologias aplicadas no desenvolvimento deste projeto, assim como os resultados de um mapeamento sistemático sobre trabalhos relacionados com o problema de pesquisa proposto neste trabalho.

2.1. Comportamento e Saúde dos Gatos Domésticos

Segundo Atkinson (2018), o padrão de comportamento de um gato tem uma notável ligação com vários aspectos da sua saúde e bem-estar. Ainda conforme o autor, esses efeitos comportamentais, frequentemente associados a doenças felinas, manifestam-se através do aumento ou diminuição em atividades como a alimentação, hidratação ou excreção[1]. Ainda assim, apesar dessa forte relação, os sinais demonstrados pelos felinos podem passar despercebidos pelos seus tutores, ocasionando o agravamento de patologias, estresse ou desconforto [Gesthich-Frank 2021].

Conforme aponta Borin-Crivellenti e Malta (2015), por exemplo, os sintomas de poliúria e polidipsia[2] são os principais indicativos de doenças relacionadas com o sistema endócrino de um animal de pequeno porte, como o diabetes. Além disso, outro fator que pode ser indicado por meio de modificações nas ações diárias do gato é o estresse. De acordo com Atkinson (2018), essa reação a mudanças ambientais ou de rotina do felino pode causar o crescimento de tentativas de micção, em consequência da dificuldade de urinar, utilização de lugares não adequados para eliminação (fora da caixa de areia), diminuição de apetite e consumo de água, entre outros. Desse modo, o reconhecimento precoce desses indícios pode ser imprescindível para a procura de ajuda profissional e tratamento eficaz em estágios iniciais [Calhau et al. 2024].

Adicionalmente, esses aspectos físicos e comportamentais devem ser conhecidos e monitorados pelos tutores, de forma a não propiciar o desenvolvimento de distúrbios

[1] Refere-se à eliminação de resíduos do organismo, como urina e fezes

[2] Respectivamente, são os nomes técnicos para o aumento de produção de urina e para a sede em excesso

em consequência da ausência de cuidado com a rotina do animal. A Doença Renal Crônica (DRC), segundo Calhau et al. (2024), é uma das doenças mais comuns nos gatos, por consequência da propensão natural à baixa ingestão hídrica que a espécie tem como característica. Outro caso são as alterações alimentares, que podem resultar em obesidade ou anorexia, por conta da desregulação na ingestão de alimentos ou no "Efeito Monotonia", no qual o animal deixa de se alimentar devido à saturação de um único tipo de alimento [Scholten 2017].

2.2. Internet of Things (IoT)

O termo *Internet of Things* (IoT) é definido por Atzori et al. (2010) como um conjunto de dispositivos capazes de se comunicar entre si através da Internet, trabalhando em um ou mais objetivos em comum. Esses componentes, chamados de *smart* (inteligentes), utilizam tecnologias embarcadas, sensores e atuadores para a transformação de efeitos físicos em digitais, ou digitais em físicos [Marwedel 2021]. Esse paradigma possui uma crescente demanda, tanto na vida doméstica quanto em diversos ramos de negócios, potencializada pela grande gama de aplicações e pelo uso de tecnologias complementares, como a conexão sem fio [Atzori et al. 2010].

Segundo Hunkeler et al. (2008), em um cenário com uma heterogeneidade de dispositivos, além de uma grande quantidade de dados trafegando pela rede, o *Message Queuing Telemetry Protocol (MQTT)* é um protocolo simples e, por conta disso, particularmente próprio para a utilização na comunicação de componentes com baixos recursos no IoT. Como característica, o protocolo faz o uso de um *broker*, o qual funciona como um mecanismo de desacoplar objetos publicadores e assinantes, sendo, respectivamente, os responsáveis pela produção/envio de dados e os que consomem essas informações [Hunkeler et al. 2008]. O serviço Mosquitto, de código aberto, é um modo de implementação de um *broker* de mensagens com MQTT [Mosquitto 2024].

2.2.1. Sistemas Embarcados e Microcontroladores

Os microcontroladores correspondem a um Circuito Integrado (CI), com todos os componentes de uma arquitetura computacional, sendo, por conta disso, são amplamente utilizados em projetos de sistemas embarcados e, consequentemente, de IoT [Berger 2002]. As instruções para um conjunto específico de tarefas executadas por esse *chip* são programadas em computadores de propósito geral, utilizando plataformas como o Arduino IDE, e posteriormente carregadas na memória *flash* do microcontrolador [Barret e Pack 2006]. Além disso, através de um módulo chamado *General-Purpose Input/Output* (GPIO), esses equipamentos também possuem a capacidade de controlar sensores e atuadores para coleta de medidas ou realizar ações no mundo real [Monk, 2012].

Uma das implementação de microcontroladores mais popular e utilizada para projetos e prototipação é o Arduino, que é um Hardware e Software de código aberto, multiplataforma com uma linguagem de programação baseada em C/C++ e que pode ser expandida com bibliotecas compatíveis [Arduino 2022]. De modo geral, as placas Arduino são constituídas por alguns componentes básicos como Porta USB, para a comunicação e programação através de um computador, pinos digitais e analógicos para

controles de funções e outros pinos para a criação de circuitos e alimentação de partes externas [Arduino, 2024].

2.2.2. Sensores

Os sensores, anteriormente destacados como dispositivos que podem atuar em um sistema de IoT, são componentes que podem gerar dados sobre o mundo real, servindo como uma ponte entre o mundo analógico e o digital [Atzori 2010]. Para isso, utilizam medidas de quantidades físicas, podendo ser aplicados para registro de velocidade, posição, temperatura, luz, entre outras possibilidades, aplicadas em diferentes casos de uso [Lee e Seshia 2017]. Em um sistema embarcado, os sensores, como outros periféricos, são conectados aos pinos GPIO do microcontrolador.

Um sensor ultrassônico, de acordo com Warren et al (2019), é capaz de medir distâncias a partir de ondas sonoras, calculando a medida do espaço até um objeto próximo através da reflexão do som. Desse modo, a distância é calculada a partir da velocidade dessa grandeza, equivalente a 340 m/s em relação ao tempo em que a onda demorou para retornar para o sensor [Dunn 2006]. Um exemplo desse tipo de sensor é o HC-SR04, que pode ser utilizado para medir com precisão distâncias pequenas, entre 2 centímetros e até 4 metros [Eletrogate 2024]. Para cálculo do espaço entre o sensor e uma superfície próxima, esse sensor contém um pino de *TRIGGER*, o qual emite a onda, e um de *ECHO*, o qual recebe a reflexão.

Diferente do modelo HC-SR04, alguns outros sensores podem ser produzidos de forma integrada com microcontroladores, microprocessadores ou apenas com um interface de rede, para, respectivamente, permitir o processamento e o envio de dados através da Internet [Lee e Seshia 2017]. Encaixa-se nessa categoria o ESP32-CAM, que incorpora uma câmera com uma placa ESP32, a qual, além de poder ser programada de forma similar a um Arduino, também possui o recurso de conexão WiFi e Bluetooth [MakerHero 2024] e pode utilizar de bibliotecas para processamento de imagens, como a Eloquent ESP32-CAM, que possui funções e códigos prontos para utilização desse dispositivo [Eloquent Arduino 2024].

2.2.3. Edge Impulse

Edge Impulse é uma ferramenta para a implementação de aprendizado de máquina em diversos tipos e modelos de sensores, placas e dispositivos embarcados [Edge Impulse 2024a]. Para isso, aplica os conceitos de TinyML para execução de modelos de aprendizado de máquina em componentes com restrições de recursos, como os microcontroladores. Por conta dessas limitações tanto de recursos computacionais quanto de armazenamento, as etapas para criação do modelo são realizadas antes da implantação do modelo no dispositivo, com os processos de coleta de dados e treinamento sendo executadas localmente ou na nuvem [Edge Impulse 2020].

Um dos tipos de modelos que podem ser explorados através de sensores de câmeras, como o ESP32-CAM, disponibilizado pela ferramenta é o *Faster Objects, More Objects* (FOMO), para detecção de objetos [Edge Impulse 2024b]. Segundo Zou (2019), a detecção de objetos é uma categoria de visão computacional com o objetivo de identificar uma ou mais classes de itens dentro de uma imagem, resultando em

informações sobre a presença e localização do alvo dentro de um quadro. As etapas de criação de um projeto desse tipo adiciona a necessidade de um passo de classificação (*labeling*) dos objetos nas imagens inseridas no conjunto de treinamento, utilizando uma *bounding box*.

2.3. Computação em Nuvem com AWS

A computação em nuvem é definida pela Amazon Web Services (2024a), conhecida pela sigla AWS, como a "entrega de recursos de TI sob demanda por meio da Internet com definição de preço de pagamento conforme o uso". Dessa forma, a nuvem é a disponibilização de serviços como computação e armazenamento por um provedor, como a AWS, podendo ser provisionados de qualquer lugar ou em qualquer momento. A AWS também dá suporte para diferentes formas de acesso aos serviços providos, que podem ser criados, alterados, utilizados ou deletados através do console, interface de linha de comando ou usando um *Software Development Kit* (SDK), como a biblioteca da linguagem de programação Python chamada Boto3 [AWS, 2024b].

Os serviços abordados no presente estudo, responsáveis pela integração e envio de mensagens entre sistemas e/ou usuários finais, processamento e armazenamento serão detalhados a seguir.

2.3.1. Serviços de Integração

O *Amazon Simple Queue Service* (SQS) é um serviço da AWS para criação de filas de mensagens totalmente gerenciadas pelo provedor, ou seja, sem a necessidade de provisionamento de recursos [AWS, 2024c]. Os produtores, sendo quaisquer componentes ou sistemas distribuídos, enviam uma mensagem para a fila, com um consumidor buscando-a, processando-a e em seguida deletando essa comunicação da fila.

De forma similar ao SQS, o *Amazon Simple Notification Service* (SNS) provê recursos para a criação de tópicos e assinaturas para que mensagens possam ser enviadas por publicadores/produtores, e sejam recebidas por assinantes/consumidores, não sendo focado em N:1 como o serviço de filas [AWS 2024d]. Os assinantes, que podem ser usuários reais ou outros serviços de dentro ou fora da nuvem, podem receber as mensagens através de diversos meios, como HTTP, e-mail, SMS, notificações push, entre outros.

2.3.2. AWS Lambda

O AWS Lambda permite a execução de códigos sem provisionamento de servidores e executando apenas quando receber um gatilho, ou seja, direcionado à eventos [AWS 2024e]. Esse código é organizado em uma função, a qual disponibiliza recursos como variáveis de ambiente, controle de versões de código e compatibilidade com diferentes linguagens de programação para *runtime*. Para aplicações com bibliotecas mais robustas, como de aprendizado de máquina, há também a opção de utilizar imagens conteinerizadas da aplicação com todas as suas dependências, armazenadas, por exemplo, no *Elastic Container Registry* (ECR).

2.3.3. Serviços de Armazenamento

O *Simple Storage Service* (S3) é uma solução de armazenamento de arquivos de objetos, os quais correspondem a arquivos de diferentes tipos e formatos, em conjunto com os seus metadados [AWS 2024f]. Esses objetos, que têm a sua alta disponibilidade garantida pela AWS, são organizados em containers lógicos chamados *buckets*, os quais podem ser acessados (ou os objetos contidos neles) através da Internet.

O banco de dados DynamoDB é uma opção não-relacional (NoSQL) e totalmente gerenciada, no qual o espaço necessário para armazenamento de dados, ou recursos para transações também provisionadas pelo provedor [AWS, 2024g]. Fornece suporte para a criação de tabelas, com cada registro contendo chaves primárias para busca e, opcionalmente secundárias, para mais opções de indexação.

2.4. Detecção de Anomalias

A detecção de anomalias é um conceito dentro da área de aprendizado de máquina que possui como objetivo determinar de uma nova observação ou instância está em um mesmo conjunto do que observações obtidas anteriormente [Scikit-Learn 2024]. Citando Liu (2008), esses dados que possuem características diferentes do padrão são chamados de anomalias, ou *outliers*. De acordo com essa natureza, as anomalias, classificadas por alguma dimensão de valor, geralmente se apresentam em uma menor quantidade dentro de um grupo de dados.

O Isolation Forest é uma implementação dessa ideia, utilizando uma estrutura de árvore para a representação de uma sequência de divisões realizadas para isolamento de uma instância dentro de um conjunto de dados [Scikit-Learn 2024)]. Em relação a esse algoritmo, portanto, dados que são mais fáceis de serem isolados (ou que percorrem uma altura menor entre a raiz e as folhas da árvore) são classificados como anomalia [Liu 2008]. Do contrário, um dado que faz necessário uma quantidade dentro ou maior do que a média é considerado normal. Geralmente, um valor anômalo e um valor padrão são representados, respectivamente, pelos valores -1 e 1.

2.5. Trabalhos Relacionados

Para estudo dos trabalhos relacionados com o projeto apresentado neste artigo, um mapeamento sistemático foi realizado com foco em analisar implementações de monitoramento e coleta de dados de gatos domésticos, considerando os mesmos parâmetros de alimentação, hidratação e excreção. Além disso, alguns pontos foram destacados em cada um dos trabalhos relacionados, para entender quais equipamentos e tecnologias foram utilizadas para essa coleta, se esses dados foram processados/analisados e os resultados alcançados.

Desse modo, o mapeamento foi realizado baseado no método apresentado por Petersen et al. (2015). Para a pesquisa dos artigos, foi utilizado um protocolo baseado em trabalhos tanto em língua portuguesa quanto em inglesa, indexados entre o período de 2013 até março de 2024 no Google Acadêmico. Durante o planejamento do mapeamento, foram definidas as seguintes perguntas para servirem como guia da revisão:

a) RQ1: Quais foram os dados coletados considerados relevantes para monitoramento dos animais?
b) RQ2: Quais foram os equipamentos e ferramentas utilizadas para a coleta desses dados?
c) RQ3: Se foi realizado algum tipo de processamento ou análise dos dados coletados, que tecnologia foi utilizada para isso?
d) RQ4: Quais foram os resultados obtidos com o monitoramento proposto?

Foram considerados dentro do escopo da pesquisa trabalhos que tratassem de monitoramento de pelo menos um dos comportamentos definidos no primeiro objetivo específico do projeto: alimentação, hidratação e/ou excreção. Para isso, foram utilizados os termos de pesquisa "monitoramento de comportamento de gatos" ou "*monitoring feline behaviour*", "monitoramento automatizado de comportamento de gatos" ou "*automated monitoring of cats behavior*" e "monitoramento de animais de estimação" ou "*pet monitoring system*".

Own e Teng (2013) implementaram uma coleira utilizando uma *tag* para monitorar a atividade de alimentação, através de uma sensor no comedouro, e a sua localização, por meio do posicionamento de sensores nas portas dentro da residência. Por conta da utilização dessa tecnologia, os autores puderam ter a oportunidade de criar um sistema com a capacidade de identificar mais do que um felino em uma mesma casa, utilizando identificadores diferentes nas *tags* de cada um dos animais. Ademais, o sistema também faz uma análise das informações colhidas, gerando estatísticas sobre o comportamento do gato.

Já tanto Kim (2016) quanto Kumar et al. (2020) e Silva et al. (2008) adicionaram o monitoramento como um complemento para alimentadores automatizados propostos no projeto, com destaque à Kim, que também propôs o uso de um *pooping pad* [3] automatizado. Tanto Kim e Kumar et al. realizaram o monitoramento da alimentação através de um sensor de carga integrado com o comedouro, com o intuito de controlar a quantidade de alimento disponível. Já Silva et al. abordaram esse monitoramento usando um sensor infravermelho, além de ser o único entre os cinco que utilizam um serviço de nuvem. Os três trabalhos também têm em comum a utilização de câmeras para o acompanhamento remoto para uso do tutor, mas nenhum realiza algum tipo de análise nos dados coletados.

Por sua vez, Chen et al. (2021) equipa uma câmera para observação de comportamento do animal, diferente dos autores anteriormente citados. Dentro do projeto, esse sensor tem como objetivo monitorar atividades como alimentação, sono, movimento e uso de caixa de areia em tempo real. Para identificação dessas ações, a aplicação utiliza o modelo *You Only Look Once* (YOLO), executado em um Raspberry Pi. Isso permite com que realize análises durante a ocorrência de algum comportamento considerado indesejado, como utilização da caixa de areia por mais do que 30 segundos, por exemplo.

[3] Não muito comum no Brasil, o *pooping pad* é um tapete em que os *pets*, principalmente cães, são treinados para utilizar para fazer as suas necessidades.

3. Procedimentos Metodológicos

Este segmento tem como foco demonstrar os métodos utilizados para desenvolvimento desta pesquisa, que teve um caráter aplicado e abordagem quantitativa, adotando procedimentos experimentais para desenvolvimento de um protótipo. Dessa maneira, para a prototipação de um sistema para monitoramento e análise do comportamento de gatos domésticos, visando a identificação de anomalias, foi criada a arquitetura apresentada na Figura 1, a qual será explorada nas seções subsequentes.

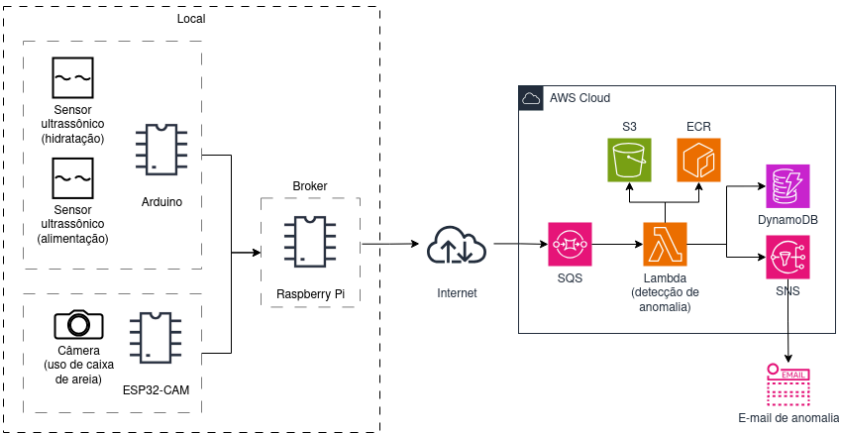

Figura 1. Representação da arquitetura local e nuvem do protótipo.

3.1. Módulos Locais

Os módulos de IoT foram implantados para realizar o monitoramento dos comportamentos do felino, de modo a possibilitar a coleta de dados para treinamento e análise da frequência das atividades de alimentação, hidratação e uso da caixa de areia (excreção). Sendo assim, ambos os módulos serão detalhados em seguida, juntamente com o *broker* para tratar a comunicação entre o sistema local e a nuvem.

3.1.1. Módulo de Alimentação e Hidratação

O módulo de alimentação e hidratação teve como objetivo coletar a informação da quantidade de vezes em que o gato realizou cada uma dessas atividades, para serem posteriormente analisadas e armazenadas na nuvem. Para esse fim, foram utilizados dois sensores ultrassônicos controlados por um Arduino UNO, com o propósito de enviar um sinal, classificado de acordo com o tipo de dado, para o *broker* registrar o horário da ocorrência percebida pelos sensores. Os sensores ultrassônicos foram fixados acima da posição onde o gato fica para se alimentar ou hidratar.

O algoritmo executado pelo Arduino para a coleta das medidas funciona com a lógica mostrada na Figura 2, em que o sistema mede a distância entre o sensor até o piso e quando um gato se aproxima do comedouro ou bebedouro, esta distância fica menor, indicando que o gato está dentro dessa área, iniciando um evento para ser enviado para análise. Além disso, para evitar que muitos eventos sejam iniciados repetidamente, seja por conta do comportamento do gato durante a alimentação ou por outros fatores

externos, foi implementada uma contagem de tempo mínimo entre as ocorrências, de 5 minutos.

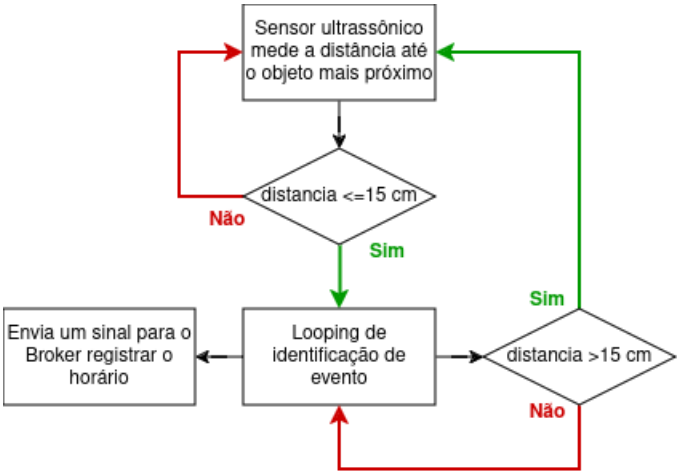

Figura 2. Lógica do módulo de alimentação e hidratação.

3.1.2. Módulo de Uso da Caixa de Areia

O módulo de uso da caixa de areia valeu-se de uma lógica de reconhecimento de eventos similar ao módulo descrito na seção anterior. Adicionalmente, no entanto, utilizou de uma câmera integrada com um microcontrolador, a ESP32-CAM, para coleta de imagens em tempo real e localização do gato.

A detecção do uso da caixa de areia pelo animal foi realizada através de um modelo de detecção de objetos gerada com a ferramenta Edge Impulse, importada para o ESP32-CAM. O processo de criação do modelo teve como início a coleta de imagens para treinamento, utilizando um código da biblioteca Eloquent ESP32, que em seguida foram importadas para o Edge Impulse, usando uma abordagem de 80% de imagens para treinamento e 20% para testes. As imagens definidas para treinamento, então, passaram pelo processo de categorização, para adição do *label* indicando o felino na imagem. Finalmente, o modelo pôde ser treinado e exportado para ser carregado no ESP32-CAM.

3.1.3. Broker

Por fim, o último membro dos componentes locais é o *broker*, que têm a função de receber os dados dos demais módulos, registrar horários e armazenar localmente, até que seja enviado para análise e armazenamento definitivo na AWS. O *broker* foi implementado utilizando um Raspberry Pi, com o Mosquitto instalado para recebimento de mensagens enviadas pelos microcontroladores para o tópico criado para cada módulo, através do protocolo MQTT. Como citado anteriormente, neste projeto os módulos irão enviar sinais para indicar quando uma das categorias de evento foi identificada, sendo registrado pelo *broker* o horário em que a sinalização foi recebida. Esse registro é acrescentado em um arquivo JSON, registrando uma data, relativa ao dia das coletas, e horas e minutos separados por atividade.

Esse armazenamento temporário local acaba quando um agendamento via o utilitário crontab chama um outro script Python para envio dos dados JSON para a AWS, usando um cliente do SQS que envia uma mensagem para o serviço configurado na AWS (detalhado na seção 3.2). Esse procedimento, que ocorre a cada 24 horas, irá finalizar após confirmação do recebimento da mensagem pela fila, apagando o conteúdo do arquivo JSON armazenado localmente, de modo a permitir a gravação de um novo conjunto de dados relativos ao próximo dia. A formatação padrão do arquivo, com os horários para cada uma das categorias de dados, junto com a indicação do dia relacionado com a coleta, é mostrado no Quadro 1.

3.2. Implementação do Isolation Forest

O algoritmo de detecção de anomalias Isolation Forest foi implementado no protótipo através da biblioteca Scikit-Learn, utilizando também a linguagem Python. Inicialmente, para obter os modelos que permitem a análise das coletas, foi necessário um treinamento com um volume considerável de dados. Com isso, o treinamento foi executado levando em consideração a quantidade de horários (frequência) em que cada um dos tipos de dados avaliados ocorreu no registro de cada um dos dias presentes no arquivo JSON inserido, utilizando a função *fit* da biblioteca. Esse processo resultou em três modelos: um modelo para alimentação, outro para hidratação e um terceiro para excreção.

Quadro 1. Formatação do registro da frequência identificada para cada atividade.

```
{
    "timestamp": "2024-01-17",
    "alimentacao": {
        "horarios": [
            "06:00",
            "09:00"
        ]
    },
    "hidratacao": {
        "horarios": []
    },
    "caixa_areia": {
        "horarios": [
            "10:00"
          ]
    }
}
```

Após essa etapa, esses modelos treinados puderam ser utilizados para análise de novas instâncias de dados, através da função *predict*. Dessa maneira, cada uma das categorias de eventos passaram pela predição com base no seu respectivo padrão, retornando um valor igual a 1 caso nada tenha sido encontrado fora do padrão ou -1 caso uma anomalia tenha sido constatada. Por conta da natureza dos dados tratados e de como foi arquitetado o treinamento, portanto, as anomalias estão ligadas com casos em que a frequência de alguma das atividades for menor, maior ou igual a zero, se apresentando como diferente da norma de comportamento diário do gato.

3.3. Solução da Nuvem AWS

O fluxo de comunicação entre os serviços criados na nuvem AWS para o protótipo foi representado no diagrama da Figura 3. Inicialmente, para integração dos módulos locais com a nuvem AWS, como citado anteriormente na seção 3.1.3, uma fila SQS foi configurada, para recebimento das mensagens do *broker*. Para dar continuidade nessa esteira de processamento, essa fila foi configurada como gatilho para a função Lambda, que realiza propriamente a análise dos dados. Essa configuração de gatilho faz com que a função seja acionada sempre que um evento ocorra, o qual, nesse caso, é igual uma nova mensagem na fila, contendo os dados de 24 horas coletados no ambiente local.

A função do Lambda foi configurada para fazer uso de uma imagem de container Docker em um repositório do ECR. Isso foi necessário por conta da utilização da biblioteca Scikit-Learn, que tornou essa aplicação muito maior do que o limite de tamanho que o serviço permite, equivalente a um código e suas dependências somando, no máximo, 250 *megabytes* (descompactado). Para execução da análise, a função também fez uso de um *bucket* do S3 como repositório para os modelos treinados do Isolation Forest, recuperando esses arquivos a cada execução.

Após finalização da predição, a função também está programada para realizar a escrita dos dados recebidos mais resultado dentro de uma tabela do DynamoDB, permitindo consultas futuras. Essa tabela foi criada com a data dos dados como chave primária do tipo string, uma coluna para os dados de cada tipo (alimentação, hidratação e uso da caixa) e a coluna adicional indicando se o dia resultou ou não em uma anomalia. Caso tenha resultado em uma anomalia, o Lambda também causa o envio de uma notificação do SNS, no qual foi criado um tópico e um e-mail assinante deste tópico, para recebimento de uma mensagem detalhando o dia, horários e indicando quais foram as anormalidades encontradas.

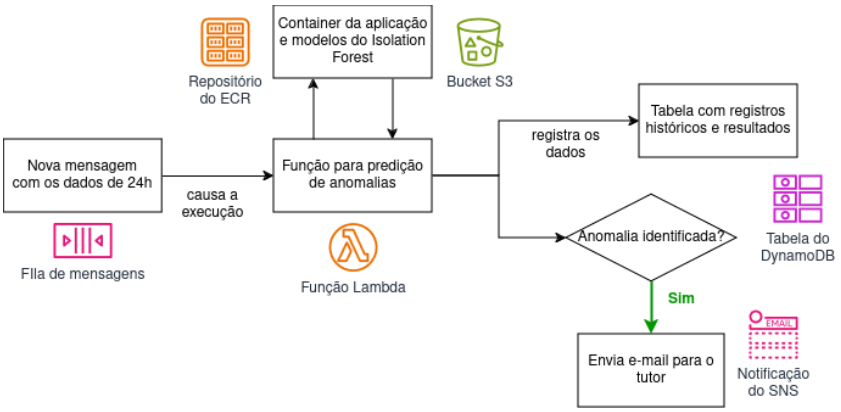

Figura 3. Fluxograma com a lógica de detecção de alimentação/hidratação.

4. Resultados

Nesta seção serão mostrados os resultados obtidos através dos procedimentos metodológicos descritos na seção anterior. Os frutos do trabalho consistem no protótipo

explicado, sendo validado a partir de uma prova de conceito, utilizando testes com dados sintéticos[4]. Ambos serão expostos nos próximos tópicos.

4.1. Protótipo

Em relação ao módulo de monitoramento de alimentação e hidratação, a utilização dos sensores ultrassônicos controlados pelo Arduino Uno foi executada de acordo com a Figura 4. Ambos os sensores foram conectados à suas respectivas protoboards, sendo que o sensor à esquerda corresponde ao responsável pela coleta da atividade de hidratação e o sensor à direita, próximo ao Arduino, ao da alimentação. Essa foi uma abordagem diferente das usadas para coleta desses tipos de dados por trabalhos como Own e Teng (2013), que utilizou *tags* para identificar quando o felino estivesse realizando essas ações; Silva et al. (2008), que mediu a quantidade de comida ingerida através de sensores infravermelhos e Kim (2016) e Kumar et al. (2020), que fizeram uso de sensores de carga.

Esse módulo, em conjunto com a lógica utilizada para o programa executado pelo microcontrolador, mostrou-se satisfatória para o caso de uso. Conforme mostrado na Figura 5, quando a distância detectada por um dos sensores for menor do que a distância para o alimentador/bebedouro, um novo evento será detectado (1). Esse evento, então, é sinalizado para o *broker* e só irá ser considerado finalizado quando essa distância voltar a aumentar acima da quantidade estimada (2). Para evitar os falsos positivos que poderiam ser ocasionados pelo movimento do gato durante um evento, a contagem de tempo também se mostrou apropriada para limitar o intervalo de tempo entre os eventos que podem ser detectados (3).

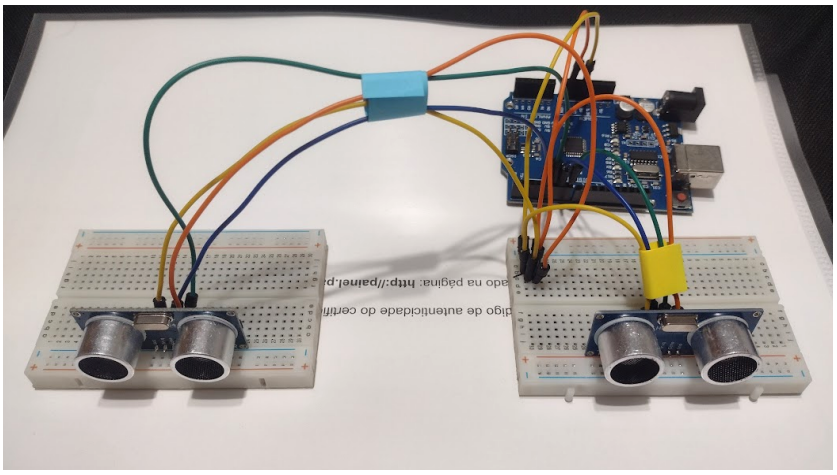

Figura 4. Resultados da montagem do módulo de alimentação/hidratação.

[4] Dados criados com algoritmos e simulações para imitar o mundo real

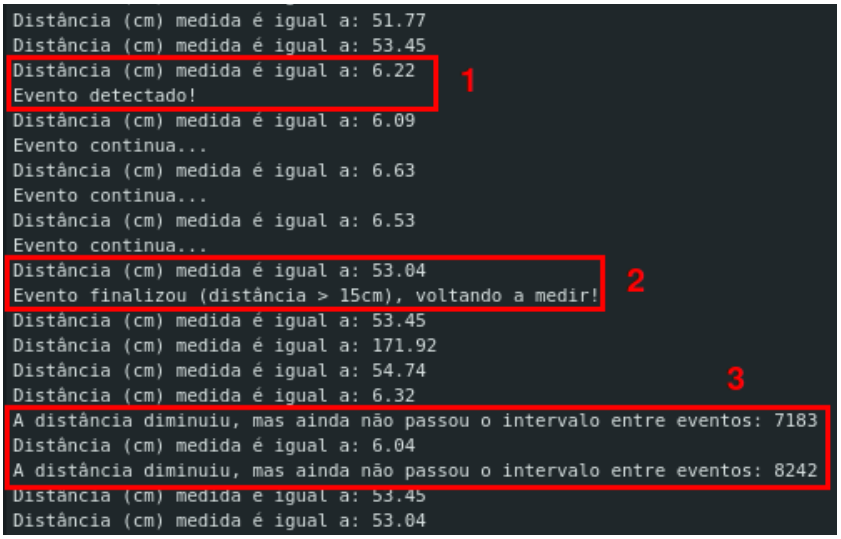

Figura 4. Demonstração da lógica de coleta do módulo de alimentação/hidratação.

Já o módulo de uso da caixa de areia, como citado anteriormente, utiliza de um ESP32-CAM, com uma câmera integrada, mostrado na Figura 6. O Arduino também contido na figura foi necessário para alimentação. Assim como o trabalho de Chen et al. (2021), o microcontrolador da câmera também é utilizado como "olhos" para o sistema, utilizando um modelo de detecção de objetos para identificação do gato no quadro. O presente protótipo e o sistema apresentado por Chen se diferem, no entanto, pelo modelo e Hardware utilizados, pois enquanto aquele utilizou o YOLO junto com o Raspberry Pi, que possui um poder de processamento maior, enquanto este usou o modelo FOMO, próprio para o ESP32 e outros componentes com menores recursos para esse tipo de carga de trabalho.

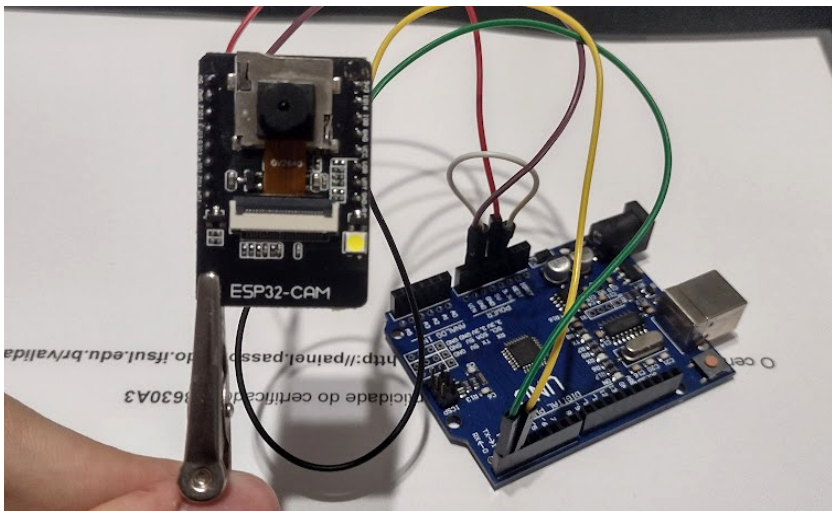

Figura 6. Resultados da montagem do módulo de uso da caixa de areia.

A eficácia do processamento de imagens para detecção de objetos que foi implantado nesse módulo pôde ser medida a partir dos resultados da criação e treinamento do modelo com a ferramenta Edge Impulse, a qual gerou a biblioteca para detecção de objetos utilizada no ESP32-CAM. O Quadro 2 mostra os resultados dos treinamentos realizados para obtenção do modelo, que apresentou, em três iterações de treinamento, uma acurácia de, respectivamente, 94,1%, 96,55% e 100%, em relação às métricas de precisão, recall e pontuação do F1. Esse resultado positivo foi obtido graças à utilização de imagens consistentes em relação ao ângulo e iluminação para treinamento e testes.

Quadro 2. Comparação entre resultados esperados e obtidos dos testes sintéticos.

N.º Treinamento	*Precision*	*Recall*	*F1 Score*	*Accuracy*
1	0,89	1	0,94	94,10%
2	0,97	0,97	0,97	96,55%
3	1	1	1	100%

4.2. Prova de Conceito

Devido a dificuldade em obter o volume de dados reais (coletados pelos sensores) necessários para a modelagem do comportamento do *pet*, optou-se por validar o protótipo usando uma prova de conceito, utilizando dados sintéticos para simulação do comportamento de um gato doméstico. Essa abordagem, além de permitir coletar dados suficientes para treinamento do modelo em um curto espaço de tempo, não contaminando os resultados com falsos positivos por variáveis não controláveis, também tornou possível os testes de diversos cenários de comportamento.

Esses dados sintéticos foram gerados utilizando um script para produção de horários, gerando um arquivo JSON com o registro de 90 dias de um comportamento simulado do animal. De forma a obter uma precisão maior no processo de análise, os dados para treinamento foram gerados seguindo os seguintes parâmetros: 5% de chance de não haver nenhum registro de horário (comportamento não ocorreu); 10% de chance de haver mais registros do que o padrão; 10% de chance de haver menos registros do que o padrão; do contrário, a quantidade de registros gerada está dentro dos limites definidos para cada uma das variáveis. Para traçar o limite do padrão para cada comportamento, por sua vez, foram utilizados os seguintes valores:

a) Alimentação: 2 eventos mínimos e 5 eventos máximos;
b) Hidratação: 4 eventos mínimo e 7 eventos máximos;
c) Uso da caixa de areia: 3 eventos mínimos e 6 eventos máximos.

O treinamento não supervisionado com os dados sintéticos utilizando o Isolation Forest apresentou resultados positivos, visto que os padrões definidos durante a geração dos dados sintéticos foram identificados nos modelos produzidos. Conforme mostrado na Figura 7, as categorias alimentação (a), hidratação (b) e uso da caixa de areia (c) reconheceram as quantidades de frequência menor e maior do que o padrão definido

como anomalias, representadas pelos pontos com a cor vermelha. Do mesmo modo, os valores encontrados dentro do padrão, de acordo com os menos limites, foram classificados corretamente, representados pelos pontos de cor azul nos gráficos gerados.

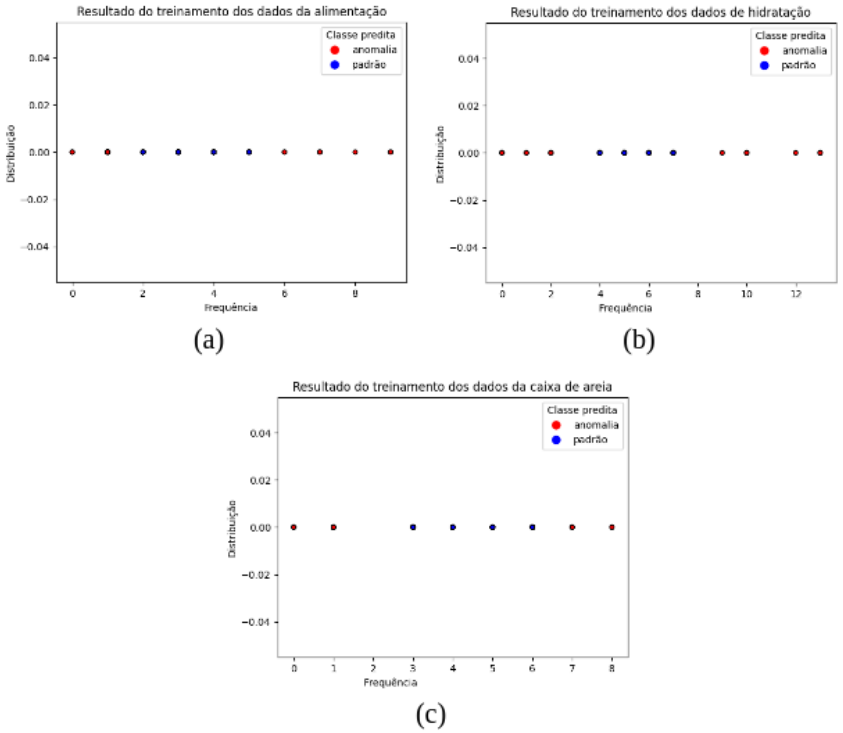

Figura 7. Representação dos resultados obtidos com o treinamento sintético.

A partir disso, os testes foram esquematizados de acordo com as descrições mostradas na Tabela 2, testando um comportamento normal, um comportamento totalmente anômalo e outras combinações de normalidade e anomalia para cada um dos tipos de eventos analisados relativos à unidade de 24 horas. Os resultados esperados (hipóteses) foram escritos antes da realização dos testes, utilizando a nomenclatura padrão mostrada na seção 2.4, com -1 para divergências do padrão comportamental (fora dos limites definidos na geração dos dados sintéticos) e 1 para uma frequência de eventos dentro do padrão delimitado. Como observável, os resultados obtidos tiveram uma ótima acurácia em comparação com os resultados esperados.

O sucesso do treinamento também foi comprovado ao realizar os testes através da AWS, enviando os dados através da fila SQS para serem processadas pela função Lambda, armazenadas no DynamoDB e, caso identificada uma anomalia, enviar uma notificação via SNS. Como mostrado na Figura 8, os logs de saída da execução da função Lambda para um dos testes sintéticos aponta o recebimento dos dados do SQS, gravação dos dados e a menção ao envio da mensagem pelo SNS, visto que dois resultados foram identificados como anômalos: a hidratação e uso da caixa de areia. Essa mensagem enviada, junto com a análise realizada da causa da anomalia (ambas as ações ocorreram em uma frequência maior do que o comum para o comportamento padrão), é mostrada na figura Figura 9.

Quadro 3. Comparação entre resultados esperados e obtidos dos testes sintéticos.

Nº	Descrição dos Dados	Resultados Esperados			Resultados Obtidos		
		Alimentação	Hidratação	Caixa de Areia	Alimentação	Hidratação	Caixa de Areia
1	*Alimentação = 0*	**-1**	1	1	**-1**	1	1
2	*Hidratação = 0*	1	**-1**	1	1	**-1**	1
3	*Uso da caixa de areia = 0*	1	1	**-1**	1	1	**-1**
4	*Alimentação < padrão*	**-1**	1	1	**-1**	1	1
5	*Hidratação < padrão*	1	**-1**	1	1	**-1**	1
6	*Uso da caixa de areia < padrão*	1	1	**-1**	1	1	**-1**
7	*Alimentação > padrão*	**-1**	1	1	**-1**	1	1
8	*Hidratação > padrão*	1	**-1**	1	1	**-1**	1
9	*Uso da caixa de areia > padrão*	1	1	**-1**	1	1	**-1**
10	*Apenas alimentação dentro do padrão*	1	**-1**	**-1**	1	**-1**	**-1**
11	*Apenas hidratação dentro do padrão*	**-1**	1	**-1**	**-1**	1	**-1**
12	*Apenas uso da caixa de areia dentro do padrão*	**-1**	**-1**	1	**-1**	**-1**	1
13	*Todas as variáveis dentro do padrão*	1	1	1	1	1	1
14	*Todas as variáveis fora do padrão*	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**

Portanto, através desses resultados, é notável o sucesso do sistema na realização do seu objetivo de identificação de anomalias, além da integração com a AWS, que ocorreu de forma a agregar mais recursos para o protótipo sem causar maior complexidade no provisionamento. A análise dos dados utilizando aprendizado de máquina para encontrar padrões diferiu-se, por exemplo, da abordagem de Own e Teng (2013), que teve como resultados as estatísticas dos comportamentos do felino. O emprego da nuvem no protótipo também foi diferente do usado por Silva et al. (2008), em que uma nuvem foi utilizada somente para armazenamento das coletas.

Figura 8. Logs de saída de uma execução do Lambda com resultados anômalos.

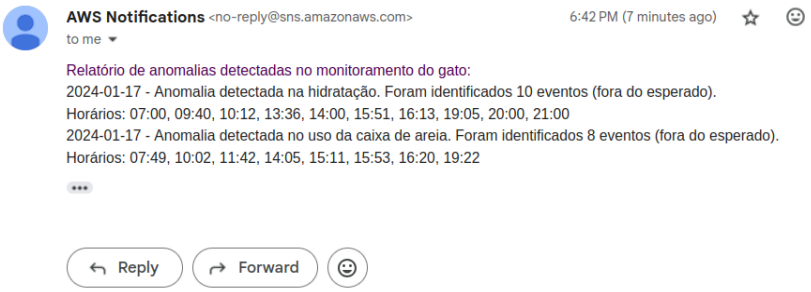

Figura 9. Envio de notificação com informações sobre as anomalias identificadas.

5. Considerações Finais

Em conclusão, o trabalho apresentou como resultado um protótipo capaz de realizar a coleta de dados sobre alimentação, hidratação e excreção realizados por um gato doméstico, através da utilização de sensores IoT para monitoramento de forma automatizada. Também foi possível integrar a computação em nuvem com a AWS para processamento e armazenamento das coletas, agrupadas em períodos de 24 horas. De forma a analisar o comportamento e identificar ações fora do padrão, foi implementado o Isolation Forest, para detecção de anomalias na frequência da realização dessas atividades cotidianas, notificadas ao tutor. Com isso, pôde-se concluir que o sistema proposto é eficaz para resolver o problema da pesquisa, pois tem potencial para realizar a análise do comportamento padrão e identificar anormalidades, as quais podem estar relacionadas com o estado de saúde e bem-estar do *pet*.

Um ponto de dificuldade encontrado durante a implementação e testes do sistema foi o tempo necessário para coleta do volume de dados para treinamento do modelo de detecção de anomalias. Tal aspecto foi solucionado neste trabalho através do uso de dados sintéticos. É previsto para os trabalhos futuros o desenvolvimento dessa fase de coleta em um intervalo de tempo maior, contando com um tempo para acostumar o felino com os objetos novos perto dos seus locais rotineiros. Outra abordagem possível é realizar a fase de coleta em paralelo ao treinamento, tendo um modo de medir ou estimar quando a acurácia aumentasse o suficiente para gerar um modelo definitivo. No entanto, caso ainda adotasse a nuvem como plataforma, poderia vir a gerar um alto custo de utilização.

Ademais, outro aspecto que será explorado no futuro é que mais tipos e dimensões dos dados sejam considerados para a análise do comportamento do animal, além da frequência que foi considerada neste trabalho. Apesar de ser um fator que poderia ajudar a identificar mudanças comportamentais como a poliúria e polidipsia, ainda assim não poderia medir, respectivamente, qual foi o tipo da excreção que ocorreu durante o uso da caixa de areia ou a quantidade de água que foi bebida durante o dia, em mililitros. Para tanto, é necessário o emprego de diferentes tipos de sensores, como sensor de carga e de nível de água. Uma pesquisa futura, além de outros dados, também poderia prever com um ambiente domiciliar com mais de um gato, que foi implementada com a utilização de tags do trabalho de Own e Teng (2013).

Referências

Arduino (2022). What is Arduino? https://docs.arduino.cc/learn/starting-guide/whats-arduino/. Acesso em: 26 abr 2024.

Arduino (2024). Getting Started with Arduino. https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/. Acesso em: 26 abr 2024.

Atkinson, T. (2018). Practical feline behaviour: Understanding cat behaviour and improving welfare. Oxfordshire, UK ; Boston, MA: CABI.

Atzori, L. et al (2010). The Internet of Things: A survey. Elsevier. https://www.sciencedirect.com/science/article/abs/pii/S1389128610001568. Acesso em: 18 abr 2024.

AWS (2024a). O que é a computação em nuvem? https://aws.amazon.com/pt/what-is-cloud-computing/. Acesso em: 16 mai 2024.

AWS (2024b). AWS SDK para Python (Boto3). https://aws.amazon.com/pt/sdk-for-python/. Acesso em: 17 out 2024.

AWS (2024c). SQS Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/. Acesso em: 16 mai 2024.

AWS (2024d). SNS Developer Guide. https://docs.aws.amazon.com/sns/latest/dg/welcome.html. Acesso em: 10 out 2024.

AWS (2024e). Lambda Developer Guide. https://docs.aws.amazon.com/lambda/latest/dg/. Acesso em: 16 mai 2024.

AWS (2024f). S3 Developer Guide. https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html. Acesso em: 07 out 2024.

AWS (2024g). DynamoDB Developer Guide. https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/. Acesso em: 07 out 2024.

Barrett, S. F. e Pack, D. J. (2006). Microcontrollers Fundamentals for Engineers and Scientists. 1. ed. [S.l.]: Morgan & Claypool.

Berger, A. S. (2002). Embedded Systems Design: An introduction to processes, tools, and techniques. Berkeley, CA: CMP Books.

Borin-Crivellenti, S. e Malta, C. A. S. (2015). A endocrinologia da Poliúria e da Polidipsia. Investigação, 14 (6): 22-25. ISSN 21774080.

Calhau, D. S. et al (2024). Doença renal crônica em gatos. Pubvet, [S. l.], v. 18, n. 02, p. e1551. https://ojs.pubvet.com.br/index.php/revista/article/view/3493. Acesso em: 14 mar 2024.

Chen, R. C. et al (2021). Monitoring the behaviours of pet cat based on yolo model and raspberry pi. International Journal of Applied Science and Engineering. v. 18, p. 1–12, 01. https://www.researchgate.net/publication/354535148_Monitoring_the_behaviours_of_pet_cat_based_on_YOLO_model_and_raspberry_Pi. Acesso em: 27 mar 2024.

COMAC (2023). Radar Pet 2023: O salto emocional na relação tutor-pet. https://sindan.org.br/wp-content/uploads/2023/12/PET-Talks_Apresentacao-Radar-Pet-2023.pdf. Acesso em: 4 mar 2024.

Dunn, W. C. (2006). Introduction to Instrumentation, Sensors, and Process Control. Norwood: Artech House.

Edge Impulse (2020). Edge Impulse Brings TinyML to Millions of Arduino Developers. https://www.edgeimpulse.com/blog/edge-impulse-brings-ml-to-arduino. Acesso em: 20 nov 2024.

Edge Impulse (2024a). Edge Impulse: For beginners. https://docs.edgeimpulse.com/docs/readme/for-beginners. Acesso em: 28 set 2024.

Edge Impulse (2024b). Detect objects with FOMO. https://docs.edgeimpulse.com/docs/tutorials/end-to-end-tutorials/object-detection/detect-objects-using-fomo. Acesso em: 28 set 2024.

Eletrogate (2024). Módulo Sensor de Distância Ultrassônico HC-SR04. https://www.eletrogate.com/modulo-sensor-de-distancia-ultrassonico-hc-sr04. Acesso em: 09 out 2024.

Eloquent Arduino (2024). ESP32 cam Quickstart. https://eloquentarduino.com/posts/esp32-cam-quickstart. Acesso em: 26 set 2024.

Gestrich-Frank, M. I. (2020). Impacto dos transtornos comportamentais na saúde física e bem-estar dos felinos domésticos. UFRGS. https://lume.ufrgs.br/handle/10183/271381. Acesso em: 14 mar 2024.

Hunkeler, U. et al (2008). MQTT-S: A publish/subscribe protocol for wireless sensor networks. UCSB. https://sites.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/mqtt-s.pdf. Acesso em: 22 abr 2024.

Kim, S. (2013). Smart pet care system using internet of things. International Journal of Smart Home. v. 10, p. 211–21. https://www.researchgate.net/publication/301727610_Smart_Pet_Care_System_using_Internet_of_Things. Acesso em: 27 mar 2024.

Kumar, K. et al (2020). Development of smart pet monıtorıng system. International Journal of Advanced Science and Technology. v. 29, p. 4253–4260. http://sersc.org/journals/index.php/IJAST/article/view/22928. Acesso em: 27 mar 2024.

Lee, E. A. e Seshia, S. A. (2017). Introduction to Embedded Systems: A cyber-physical systems approach. 2. ed. MIT Press. https://ptolemy.berkeley.edu/books/leeseshia/. Acesso em: 10 abr 2024.

Liu, F. T. et al (2008). Isolation Forest. IEEE. https://ieeexplore.ieee.org/document/4781136. Acesso em: 06 jun 2024.

MakerHero (2024). Placa ESP32 CAM com Câmera OV2640 2MP. https://www.makerhero.com/produto/modulo-esp32-cam-com-camera-ov2640-2mp. Acesso em: 24 set 2024.

Marwedel, P. (2021). Embedded System Design: Embedded systems foundations of cyber-physical systems, and the internet of things. 4. ed. Cham, SW: Springer. https://link.springer.com/book/10.1007/978-3-030-60910-8. Acesso em: 10 abr 2024.

Monk, S. (2012). Programming Arduino: Getting started with sketches. 1. ed. [S.l.]: McGraw-Hill.

Mosquitto (2024). Eclipse Mosquitto: Documentation. https://mosquitto.org/documentation/. Acesso em: 26 out 2024.

Petersen, et al. (2015). Guidelines for conducting systematic mapping studies in software engineering: An update. Information and Software Technology, v. 64, p. 1–18. ISSN 0950-5849. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950584915000646/>. Acesso em: 27 mar 2024.

Own, C. e Tang, C. (2013). The study and application of the iot in pet systems. Advances in Internet of Things. v. 03, p. 1–8, 01. https://www.researchgate.net/publication/270850816_The_Study_and_Application_of_the_IoT_in_Pet_Systems. Acesso em: 27 mar 2024.

Scikit-Learn (2024). 2.7.3.2. Isolation Forest. https://scikit-learn.org/1.5/modules/outlier_detection.html. Acesso em: 12 out 2024.

Scholten, A. D. (2017). Particularidades comportamentais do gato doméstico. UFRGS. https://lume.ufrgs.br/handle/10183/170364. Acesso em: 18 mar 2024.

Silva, B. G. T. B. et al (2018). PetFeeder: Sistema remoto de alimentação e monitoramento de animais. http://silverio.net.br/heitor/disciplinas/oficina3/relatorios/EEX23_RT_PetFeeder.pdf. Acesso em: 27 mar 2024.

Warren, J. D. et al (2019). Arduino para Robótica. São Paulo: Blucher.

Zou, Z. et al (2019). Object Detection in 20 Years: A Survey. https://www.researchgate.net/publication/333077580_Object_Detection_in_20_Years_A_Survey. Acesso em: 21 nov 2024.