# *Monitoramento Inteligente da Fome Felina*

Lucas Santos Monteiro
Universidade Estadual do Rio Grande do Sul - UERGS Guaíba / Brasil
lucas-monteiro@uergs.edu.br

**ABSTRACT**

This project presents the development of an intelligent monitoring system designed to detect when domestic cats are hungry and alert their owners in real time. The system uses a Passive Infrared (PIR) sensor integrated with an ESP32 microcontroller to identify the presence of cats near their feeding area. Upon detection, it sends an immediate notification to the owner's smartphone, ensuring the pet is fed at the right time and promoting its well-being. The methodology involved researching sensor operation and integration, circuit assembly, programming the ESP32 for sensor reading and notification sending, followed by practical tests and adjustments to optimize sensitivity and reduce false positives or negatives. The results demonstrate the feasibility of using low-cost microcontroller-based systems for pet monitoring, with potential future expansions such as integration with an automatic feeder to further automate domestic animal care.

**CCS CONCEPTS**

Computer systems organization.
Embedded and cyber-physical systems.
Embedded systems.

**KEYWORDS**

ESP32; PIR sensor; pet monitoring; IoT; cats.

**1. Introdução**

Atualmente, o cuidado com pets envolve não apenas atenção presencial, mas também a utilização de tecnologias que possam auxiliar no monitoramento do bem-estar dos animais domésticos. Gatos, por exemplo, muitas vezes demonstram fome através de sua presença constante na área de alimentação, mas tutores ocupados ou distraídos podem não perceber esses sinais rapidamente. Pensando nisso, este projeto propõe o desenvolvimento de um sistema de monitoramento inteligente utilizando uma ESP32 e um sensor PIR (Infravermelho Passivo) para detectar a presença dos gatos próximos ao local de alimentação e alertar o tutor em tempo real. A ESP32 foi escolhida devido à sua alta capacidade de processamento, conectividade Wi-Fi estável e facilidade de integração com sensores, sendo ideal para aplicações de Internet das Coisas (IoT). Dessa forma, busca-se garantir a alimentação adequada e o bem-estar dos pets, além de demonstrar a aplicabilidade de sistemas embarcados de baixo custo em soluções domésticas automatizadas.

**2. Descrição do sistema**

O sistema desenvolvido consiste em um monitor de presença inteligente para gatos, composto por uma ESP32 e um sensor PIR (Infravermelho Passivo). O sensor PIR é responsável por detectar a presença dos gatos quando eles se aproximam da área de alimentação. Ao identificar movimento, o sensor envia um sinal digital para a ESP32 , que processa essa informação e aciona o envio de uma notificação ao tutor, informando que o pet

deseja comer.

A ESP32 foi escolhida por oferecer conectividade Wi-Fi confiável, permitindo o envio rápido das notificações, além de possuir maior capacidade de processamento e recursos extras em comparação à ESP8266, o que facilita expansões futuras do projeto. A ESP8266 foi originalmente considerada, mas a ESP32 oferece vantagens de processamento e conectividade para o projeto. O sistema é alimentado por uma fonte de 5V, podendo ser adaptado para alimentação por bateria caso seja necessário posicioná-lo em locais sem tomada próxima.

Os componentes são conectados por jumpers durante a prototipagem , e a montagem final pode ser acomodada em uma caixa de proteção para maior segurança. Em operação, o sistema fica em estado de monitoramento contínuo, detectando a presença do gato sempre que ele se aproximar , sem necessidade de intervenção manual , garantindo praticidade ao tutor e promovendo o bem-estar animal.

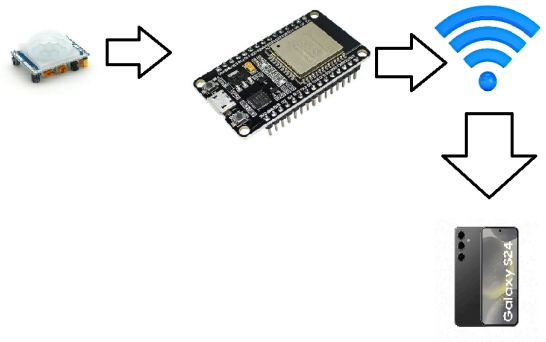


**Figura 1: Visão geral do sistema**

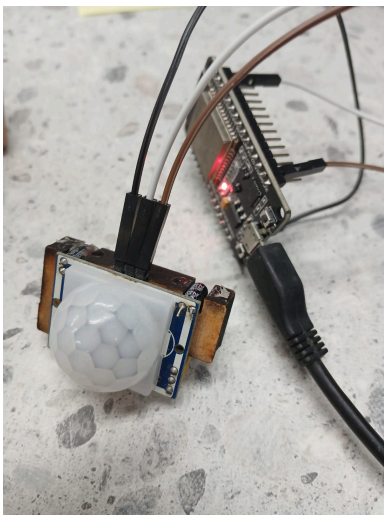

**Figura 2: Projeto montado**

## 3. Trabalhos relacionados

O Intelligent Pet Feeder with Real-Time Camera Monitoring (2024), publicado na IEEE, apresenta um sistema de comedouro inteligente para pets que integra um microcontrolador ESP32S e um aplicativo móvel. A solução emprega um sensor PIR para detectar movimento na área de alimentação e, ao identificar a presença do animal, ativa um módulo de câmera ESP32 para confirmar sua identidade antes da dispensação da ração por meio de um motor de servo. O sistema possibilita o monitoramento e controle remoto dos horários de alimentação, assegurando refeições pontuais e porcionadas. A relevância para o presente projeto reside na utilização combinada do sensor PIR e da ESP32 para a detecção de presença e na implementação de conectividade para notificação ou acionamento de ações.

O Smart Pet Feeder System and Big Data Processing to Predict Pet Food Shortage (2021) detalha um sistema de comedouro inteligente que emprega um ESP32 como componente central, conectando-o a um sensor infravermelho (PIR) para detectar a aproximação do pet e um sensor de peso para monitorar o nível de ração no reservatório. O sistema opera dispensando a ração automaticamente

em horários predefinidos ou mediante a detecção da aproximação do animal. Notificações sobre a dispensação de comida e o baixo nível de ração são enviadas ao tutor via Telegram. Além disso, a coleta de dados sobre o consumo e o nível de alimento é utilizada para prever futuras necessidades. Este trabalho compartilha uma forte similaridade com o projeto proposto, uma vez que ambos utilizam a ESP32 e o sensor infravermelho para a detecção da presença do animal e implementam um sistema de notificação ao tutor.

## 4. Estrutura e funcionamento do sistema

O sistema de monitoramento de fome felina é implementado em uma placa ESP32, utilizando o framework FreeRTOS para gerenciamento de tarefas concorrentes. A arquitetura do software é modular, dividida em funções de configuração e tarefas independentes que interagem para detectar a presença do gato e enviar notificações.

### 4.1. Estrutura Geral do Código:

Inclusões de Bibliotecas: O código inicia com a inclusão das bibliotecas necessárias:

```
1  #include <WiFi.h>
2  #include <HTTPClient.h>
3  #include <UrlEncode.h>
```

**Figura 3: Bibliotecas**

WiFi.h: Para gerenciar a conectividade Wi-Fi da ESP32.
HTTPClient.h: Para realizar requisições HTTP, essenciais para a comunicação com a API do Telegram.
UrlEncode.h: Utilizada para codificar a mensagem a ser enviada ao Telegram, garantindo que caracteres especiais sejam tratados corretamente na URL.
Definições (#define): São definidos parâmetros cruciais para a operação do sistema:

```
6  #define WIFI_SSID "A15 de Lucas"
7  #define WIFI_PASSWORD "[redacted]"
8
9  // Telegram
10 #define TELEGRAM_BOT_TOKEN "[redacted]"
11 #define CHAT_ID "[redacted]"
12
13 #define Sensor 22  // GPIO22 da ESP32
```

**Figura 4: Definições**

Credenciais de rede Wi-Fi (WIFI_SSID, WIFI_PASSWORD).
Token do bot do Telegram (TELEGRAM_BOT_TOKEN) e o ID do chat (CHAT ID) para onde as mensagens serão enviadas.
O pino GPIO da ESP32 conectado ao sensor PIR (Sensor, definido como GPIO22).
Variáveis Globais:

```
14 bool movimentoDetectado = false;
15
16 TaskHandle_t detectarTaskHandle;
17 TaskHandle_t enviarTaskHandle;
```

**Figura 5:Variáveis globais**

Bool movimentoDetectado: Uma flag booleana que controla o estado de detecção de movimento. Ela evita que múltiplas mensagens sejam enviadas para uma única detecção contínua, atuando como um "debouncer" lógico.
TaskHandle_t detectarTaskHandle, TaskHandle_t enviarTaskHandle: Handles para as tarefas do FreeRTOS, permitindo a referência e o controle das mesmas.

### 4.2. Configuração inicial:

```
19  void setupWiFi() {
20    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
21    Serial.print("Conectando ao WiFi");
22    while (WiFi.status() != WL_CONNECTED) {
23      delay(500);
24      Serial.print(".");
25    }
26    Serial.println("\nConectado ao WiFi!");
27  }
```

**Figura 6:Função setupWiFi**

setupWiFi(): Esta função é responsável por inicializar a conexão Wi-Fi da ESP32 com a rede definida. Ela tenta conectar e espera até que a conexão seja estabelecida antes de prosseguir, garantindo que a comunicação com a internet esteja disponível para o envio de notificações.

```
70  void setup() {
71    Serial.begin(115200);
72    setupWiFi();
73
74    xTaskCreatePinnedToCore(
75      detectarMovimentoTask,
76      "Detectar Movimento",
77      2048,
78      NULL,
79      1,
80      &detectarTaskHandle,
81      1
82    );
83
84    xTaskCreatePinnedToCore(
85      enviarTelegramTask,
86      "Enviar Telegram",
87      8192, // stack aumentada para evitar StackOverflow
88      NULL,
89      1,
90      &enviarTaskHandle,
91      0
92    );
93  }
```

**Figura 7:Função setup**

setup(): A função setup() é o ponto de entrada principal após a inicialização do hardware.
Inicia a comunicação serial (Serial.begin(115200)) para depuração e monitoramento.
Chama setupWiFi() para estabelecer a conexão de rede.
Cria e gerencia as duas tarefas principais do sistema utilizando o FreeRTOS (xTaskCreatePinnedToCore):
detectarMovimentoTask: Fixada no Core 1.
enviarTelegramTask: Fixada no Core 0, com uma pilha de memória maior (8192) para lidar com as operações HTTP.

### 4.3. Funcionamento do Sistema (Tarefas FreeRTOS):

O sistema opera através de duas tarefas concorrentes, o que permite que a detecção de movimento e o envio de mensagens ocorram de forma independente e não bloqueante:

```
29  void detectarMovimentoTask(void *parameter) {
30    pinMode(Sensor, INPUT);
31    for (;;) {
32      int Porta = digitalRead(Sensor);
33      if (Porta == HIGH && !movimentoDetectado) {
34        movimentoDetectado = true;
35      }
36      vTaskDelay(100 / portTICK_PERIOD_MS); // Verifica a cada 100ms
37    }
38  }
```

**Figura 8:Task para detectar movimento**

detectarMovimentoTask(void *parameter): Monitora continuamente o estado do sensor PIR (GPIO22). A cada 100 milissegundos (vTaskDelay(100/ portTICK_PERIOD_MS)), lê o valor do pino do sensor. Se o sensor estiver em estado HIGH (indicando movimento) e a flag movimentoDetectado for false (significando que um novo movimento foi detectado e ainda não processado), a flag movimentoDetectado é definida como true. Isso garante que o sistema registre a detecção e prepare-se para o envio da notificação.
Estado: O pino do sensor é configurado como INPUT.

**Figura 9:Task para enviar a mensagem via telegram**

enviarTelegramTask(void *parameter):

Responsável por enviar a notificação para o Telegram quando um movimento for detectado. Esta tarefa verifica periodicamente (vTaskDelay(200/ portTICK_PERIOD_MS)) se a flag movimentoDetectado está true.
Se estiver true, constrói a mensagem ("🐱 Gato quer comida!") e a URL para a API do Telegram, utilizando o token do bot e o ID do chat. A função urlEncode() é aplicada à mensagem para garantir compatibilidade com a URL.
Inicia uma conexão HTTP (HTTPClient http; http.begin(url);) e realiza uma requisição GET.
Verifica o código de resposta HTTP (httpResponseCode). Se for positivo, a mensagem foi enviada com sucesso; caso contrário, um erro é registrado na Serial.
Após o envio da mensagem (ou tentativa), a tarefa entra em um loop de espera (while (digitalRead(Sensor) == HIGH)) até que o movimento detectado pelo PIR cesse. Isso é crucial para evitar o envio repetitivo de mensagens enquanto o gato permanece na área de detecção.
Uma vez que o movimento cessa, a flag movimentoDetectado é redefinida para false, permitindo que um novo ciclo de detecção e notificação seja iniciado quando o gato se aproximar novamente.

### 4.4. Ciclo de Operação do Sistema:

O sistema opera em um ciclo contínuo de monitoramento e notificação:
A detectarMovimentoTask está sempre ativa, observando o sensor PIR.
Quando um gato se aproxima, o sensor PIR aciona, e a detectarMovimentoTask eleva a flag movimentoDetectado para true.
A enviarTelegramTask, ao perceber que movimentoDetectado é true, inicia o processo de envio da mensagem via Telegram.
Após o envio, a enviarTelegramTask aguarda até que o gato saia da área de detecção (sensor PIR retorne ao estado LOW) antes de redefinir a flag movimentoDetectado.
O sistema então retorna ao estado de espera, pronto para detectar a próxima aproximação do gato e alertar o tutor.
Essa estrutura com FreeRTOS garante a responsividade do sistema, permitindo que a detecção de movimento e a comunicação de rede sejam gerenciadas eficientemente sem interferência mútua.

## 5. Análise dos Resultados

Em termos de resultados, o sistema demonstrou alta eficácia na detecção de presença, envio confiável de notificações e estabilidade operacional. Isso contribui diretamente para o bem-estar animal, ao garantir a alimentação no momento certo, e para a conveniência dos tutores, que recebem alertas proativos em seus dispositivos móveis. Trabalhos relacionados corroboram a relevância e as possibilidades de expansão de sistemas IoT para o cuidado de pets.

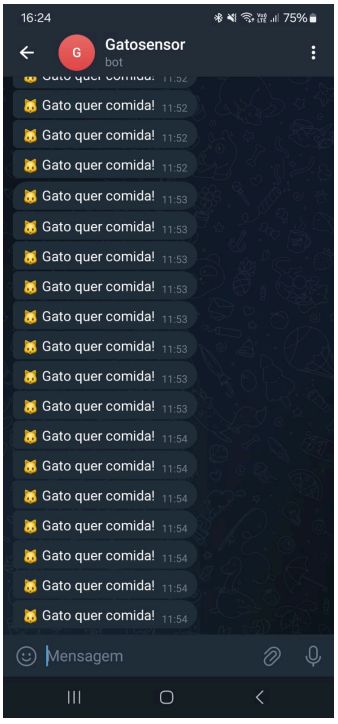


**Figura 10:Telegram com as mensagens**

## 6. Conclusão

O desenvolvimento do sistema de monitoramento inteligente da fome felina, utilizando a ESP32 em conjunto com um sensor PIR, demonstrou ser uma solução viável e eficaz para o acompanhamento das necessidades alimentares de gatos. A pesquisa abordou aspectos fundamentais, incluindo a seleção e integração de componentes, a programação do microcontrolador para processamento de dados e o desenvolvimento de um sistema de notificação via Telegram.

Os resultados obtidos indicam que o sistema é capaz de coletar dados de presença com alta precisão e enviar alertas confiáveis e em tempo real aos tutores. A conectividade Wi-Fi da ESP32 e a arquitetura de tarefas concorrentes baseada em FreeRTOS garantiram a robustez e a prontidão da comunicação.

Trabalhos futuros incluem a integração de funcionalidades adicionais, como um comedouro automático, o monitoramento de padrões de alimentação ou a análise de dados comportamentais. Com isso, o sistema poderá contribuir ainda mais para o bem-estar animal e aprimorar a interação entre pets e tutores.

## REFERENCES

[1]https://ijaem.net/issue_dcp/Intelligent%20Pet%20Feeder%20with%20Real%20Time%20Camera%20Monitoring.pdf

[2]https://pdfs.semanticscholar.org/c75f/f58f2bb079dd0839edd98eecce9d01c93565.pdf