

# DETECÇÃO DE INTRUSÕES RESIDENCIAIS USANDO DRONES E INTELIGÊNCIA ARTIFICIAL

OLIVEIRA Jr, Gilberto  
FERRARI, Allan Christian Krainski – Orientador

## Resumo

A segurança residencial é uma preocupação crescente na sociedade contemporânea, impulsionada pelo aumento das taxas de criminalidade e pela necessidade de proteger propriedades e pessoas. Este trabalho propõe uma solução inovadora para a detecção de intrusões residenciais, utilizando drones e inteligência artificial. O sistema desenvolvido integra sensores PIR (Passive Infrared) conectados a uma rede Wi-Fi, provida pelo SOC ESP32 e Raspberry Pi, que detectam movimentos suspeitos e notificam o operador em tempo real por meio de um aplicativo móvel. O operador tem a opção de acionar um drone, controlado pelo Ardupilot, que segue uma trajetória de voo predefinida para coletar imagens e transmiti-las para uma base de dados. O ecossistema Cube Pilot, um conjunto modular de componentes de hardware e software projetados para UAVs, é utilizado para garantir flexibilidade e escalabilidade ao sistema. Além disso, o sistema pode ser alimentado por módulos Heltec ESP32 com energia solar, assegurando operação contínua. Um website e aplicativo móvel dedicados permitem a análise dos dados coletados, utilizando tecnologias avançadas de vetorização de banco de dados e modelos de linguagem para uso empresarial, como a tecnologia Retrieval Augmented Generation (RAG). Esta abordagem combina tecnologias de ponta para criar um sistema de segurança residencial autônomo, eficiente e sustentável.

**Palavras-chave:** Drones; AI; ESP32; Raspberry Pi Zero; Cube Pilot; Hardware; Microcontrolador; Monitoramento; RAG.

## 1 Introdução

O mercado brasileiro de segurança residencial tem registrado um crescimento notável nos últimos anos, impulsionado pelo aumento da preocupação dos cidadãos com a proteção de suas casas. O aumento das taxas de criminalidade, incluindo roubos e invasões domiciliares, tem levado à crescente demanda por soluções de segurança mais eficientes e tecnologicamente avançadas. Esse sentimento de

insegurança nas residências está incentivando um maior investimento em sistemas de monitoramento e proteção, expandindo significativamente o mercado de segurança residencial no país<sup>1</sup>.

De acordo com uma pesquisa de 2021 da Associação Brasileira das Empresas de Sistemas Eletrônicos de Segurança (ABESE), a sensação de insegurança fez com que o número de residências que investiram em sistemas de segurança aumentasse significativamente nos últimos anos. O estudo também identifica as principais tendências para os próximos anos, tanto no mercado brasileiro quanto global. As tendências incluem a adoção de tecnologias mais avançadas e integradas, como drones e inteligência artificial, para aprimorar a eficácia e a eficiência dos sistemas de segurança residencial, motivação deste trabalho.

## **1.1 Objetivo**

O projeto "DETECÇÃO DE INTRUSÕES RESIDENCIAIS USANDO DRONES E INTELIGÊNCIA ARTIFICIAL" tem como objetivo criar um sistema de segurança residencial robusto, utilizando drones e inteligência artificial. O sistema emprega RainMaker, ESP32, Raspberry Pi Zero (mini-câmera acoplada de 5 megapixels) e quatro sensores PIR estrategicamente posicionados nas entradas da residência para detectar movimentos e acionar protocolos de segurança. Os dados coletados são analisados usando técnicas de IA para identificar potenciais intrusões.

## **2 Fundamentação Teórica**

Nesta seção, serão exploradas as bases teóricas e os conceitos fundamentais que sustentam o desenvolvimento deste trabalho. A seguir, apresentamos subseções detalhadas sobre drones, incluindo seus princípios de aerodinâmica, eletrônica de controle de voo, sistemas de propulsão e regulamentações. Então, discutiremos a solução adotada utilizando o framework Rainmaker da Espressif, o ESP32 e Raspberry Pi Zero com mini-câmera acoplada,

estabelecendo a conexão com drones através do ecossistema ArduPilot – Mission Planner.

## 2.1 Drones

Os drones têm se destacado como uma tecnologia versátil com aplicações modernas em diversos setores, incluindo a segurança residencial. Equipados com câmeras de alta resolução e sensores avançados, esses dispositivos podem monitorar perímetros de propriedades, detectar intrusões e fornecer vigilância em tempo real. Além disso, eles podem ser integrados a sistemas de alarme residencial, permitindo uma resposta rápida a eventos suspeitos. A capacidade de sobrevoar áreas de difícil acesso e transmitir imagens em tempo real torna os drones uma ferramenta valiosa para complementar os sistemas de segurança convencionais. Com o uso e regulamentação adequados, os drones podem contribuir significativamente para melhorar a segurança e tranquilidade dos proprietários residenciais.

## 2.2 Princípios de Aerodinâmica

Os drones, ou veículos aéreos não tripulados (VANTs), são dispositivos aéreos controlados remotamente que operam em uma variedade de contextos, desde entretenimentos até aplicações comerciais e militares. Eles obedecem aos princípios fundamentais da aerodinâmica para voar de forma estável e eficiente. A aerodinâmica estuda as forças e movimentos do ar em torno de objetos em movimento, como os drones. Princípios como sustentação, arrasto, empuxo e peso são essenciais para entender como os drones se movimentam no ar. A forma e o design dos drones são projetados para otimizar esses princípios, garantindo sua estabilidade, manobrabilidade e eficiência energética durante o voo. O conhecimento dos princípios de aerodinâmica é fundamental para o projeto, operação e aprimoramento contínuo dos drones, contribuindo para sua segurança e desempenho em diversas aplicações.

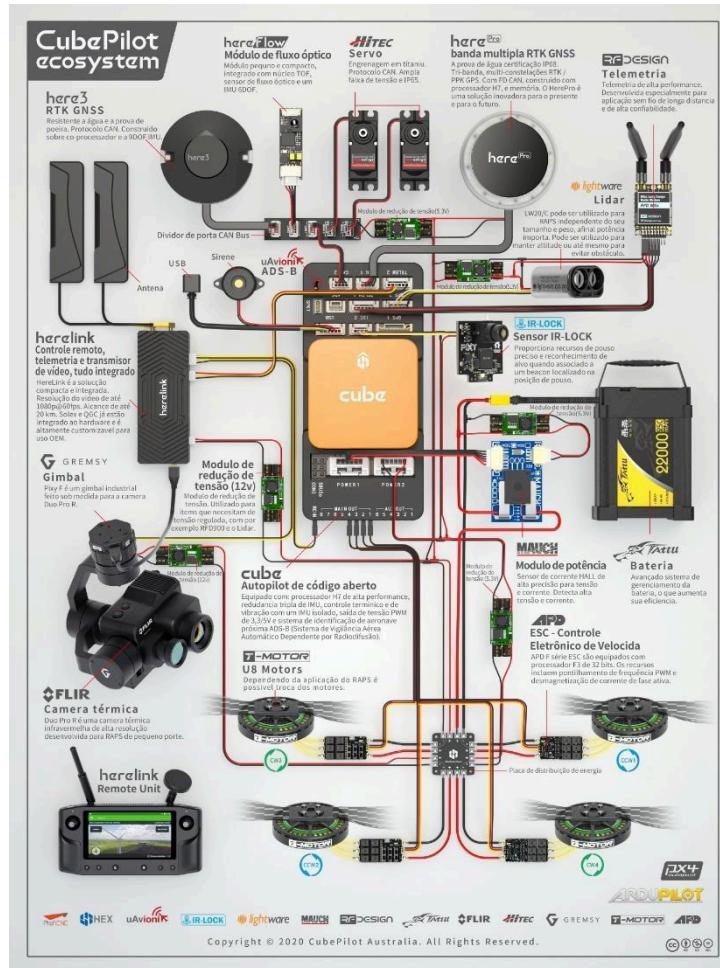
## 2.3 Eletrônica de Controle de Voo

A eletrônica de controle de voo desempenha um papel crucial no funcionamento dos drones, sendo responsável por estabilizar, controlar e orientar o voo do drone. Utilizando sensores como acelerômetros, giroscópios e

magnetômetros, juntamente com algoritmos complexos, a eletrônica de controle de voo processa dados em tempo real para ajustar continuamente a posição e a atitude do drone. Essa tecnologia possibilita manobras precisas, autonomia de voo e segurança operacional, sendo essencial para o desempenho eficiente e a estabilidade dos drones em diversas condições de voo.

Este projeto está baseado no ecossistema Cube Pilot Ecosystem (Figura 1), uma plataforma avançada para soluções em drones, projetada para oferecer alta flexibilidade e desempenho. Composto por hardware como o controlador de voo Cube e acessórios como Here GPS, o sistema é compatível com o software Ardupilot, permitindo personalizações robustas e eficientes. Suas características incluem uma integração suave com sensores e periféricos, além de uma interface amigável para desenvolvedores. Ideal para aplicações desde hobby até usos industriais, o Cube Pilot Ecosystem garante estabilidade, segurança e precisão em operações de voo autônomo.

Abaixo as especificações técnicas do Cube Pilot Ecosystem:



**Figura 1 - Diagrama de fiação do piloto automático do ecossistema Cube Pilot (Fonte: <https://ardupilot.org/>).**

Considerando o alto custo de todo esse equipamento, optamos por utilizar um aplicativo simulador de drones para dar continuidade ao projeto. O **Ardupilot - Mission Planner** foi a opção escolhida.

## 2.4 Sistemas de Propulsão

Os drones são impulsionados por sistemas de propulsão que fornecem a energia necessária para sustentar o voo. Os sistemas de propulsão dos drones podem variar significativamente dependendo do tamanho, finalidade e autonomia desejada da aeronave. Em geral, eles podem ser divididos em sistemas de propulsão elétrica, térmica e híbrida:

- **2.4.1 Propulsão elétrica:** Os drones com propulsão elétrica são alimentados por baterias recarregáveis e motores elétricos. Esses sistemas são populares

devido à sua simplicidade, eficiência e baixo custo operacional. Eles são frequentemente utilizados em drones de pequeno e médio porte, como quadricópteros e drones de asa fixa para fins de fotografia aérea, mapeamento e monitoramento.

- **2.4.2 Propulsão térmica:** Alguns drones são equipados com motores de combustão interna, como motores a gasolina ou a diesel. Esses sistemas de propulsão oferecem maior autonomia e capacidade de carga útil em comparação com os motores elétricos, tornando-os adequados para aplicações que requerem longos períodos de voo, como vigilância e patrulhamento.
- **2.4.3 Propulsão híbrida:** Os sistemas de propulsão híbridos combinam motores elétricos e de combustão interna para aproveitar as vantagens de ambos os tipos de propulsão. Esses sistemas oferecem uma combinação de eficiência, autonomia e potência, sendo utilizados em drones de médio e grande porte para missões complexas que exigem longos alcances e tempos de voo prolongados, como transporte de carga e monitoramento de grandes áreas.

Independentemente do tipo de sistema de propulsão utilizado, os drones desempenham um papel cada vez mais importante em uma variedade de setores, desde a agricultura e a segurança até a entrega de mercadorias e o mapeamento de áreas remotas. O contínuo desenvolvimento e aprimoramento dos sistemas de propulsão contribuem para a evolução e expansão das capacidades e aplicações dos drones na sociedade moderna.

## 2.5 Regulamentações e Leis

No Brasil, o uso de drones está sujeito a regulamentações e leis específicas para garantir a segurança e a privacidade. A Agência Nacional de Aviação Civil (ANAC) estabelece requisitos para o registro e operação de drones, incluindo limitações de altitude, distância de áreas sensíveis e necessidade de licenças para operações comerciais. Além disso, o Departamento de Controle do Espaço Aéreo (DECEA) define áreas de exclusão e procedimentos para evitar conflitos com a aviação tripulada. A ANATEL regula o uso de frequências de rádio para controle de

drones. Essas regulamentações visam promover o uso seguro e responsável de drones no país, incentivando a inovação e protegendo os direitos dos cidadãos.

Para o projeto foi utilizado o Mission Planner como controlador de voo do drone. As técnicas e funcionalidades desse sistema serão discutidas a seguir.

### **3 Metodologia**

A metodologia adotada foi a seguinte:

#### **3.1 Natureza da pesquisa**

A pesquisa pode ser classificada como aplicada, pois estamos aplicando tecnologias existentes (Drones, Web Service, ESP32, Raspberry Pi Zero e Inteligência Artificial) para desenvolver um sistema de segurança residencial.

#### **3.2 Abordagem do problema**

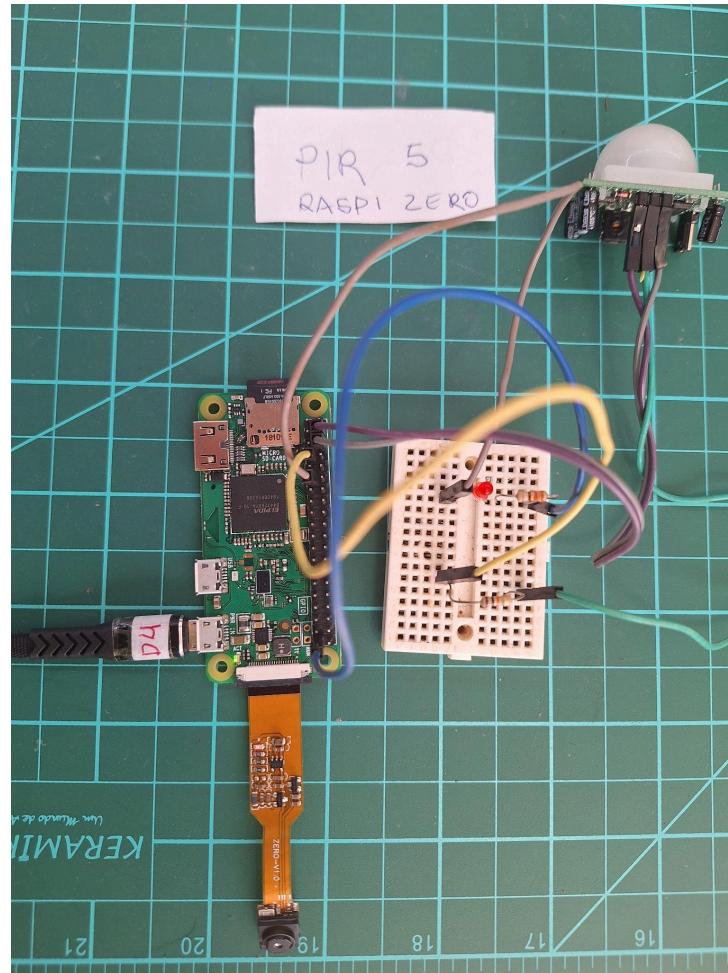
Adotamos uma combinação qualitativa e quantitativa. A abordagem qualitativa é usada para entender as necessidades e expectativas dos usuários em relação à segurança residencial. A abordagem quantitativa é aplicada para avaliar o desempenho do sistema, coletando métricas como taxa de detecção de intrusões, taxa de falsos positivos, entre outras.

#### **3.3 Procedimentos técnicos**

##### **3.3.1 Linha do Tempo do Projeto de Hardware e Modificações Críticas**

O Raspberry Pi Zero é utilizado nos procedimentos técnicos do projeto. O Raspberry Pi Zero é uma versão compacta e de baixo custo da família de microcomputadores Raspberry Pi, que oferece um conjunto robusto de funcionalidades em um formato extremamente pequeno. Ele foi utilizado no projeto "DETECÇÃO DE INTRUSÕES RESIDENCIAIS" para executar as tarefas de detecção de movimento, captura de imagens e execução do servidor web Flask.

Segue parte do protótipo com Raspberry Pi Zero e mini-camera implementado em bancada para testes:



**Figura 2 - Demonstração do Nô PIR#5 com o Raspberry Pi Zero Equipado com Câmera Zero.**  
(Fonte: foto do autor).

A camera Zero V1 com resolução de 5 megapixels. Este conjunto de hardware possibilita a implementação eficaz de projetos, como o sistema de detecção de intrusão residencial, ao combinar desempenho adequado com acessibilidade e flexibilidade. A camera Zero - V1 é especialmente adequada para aplicações que requerem monitoramento visual e análise de imagens, oferecendo uma solução compacta e poderosa para projetos de segurança e automação residencial.

### 3.3.2 Especificações Técnicas

#### 3.3.2.1 Processador Raspberry Pi Zero

O Raspberry Pi Zero é equipado com um processador Broadcom BCM2835, que possui as seguintes especificações:

- Arquitetura: ARM11
- Frequência de Clock: 1 GHz
- Cores: Single-core

Este processador é suficientemente potente para tarefas básicas de computação e é ideal para projetos embarcados que requerem eficiência energética e baixo custo.

### **3.3.2.2 Recursos Técnicos**

- Memória: 512MB de RAM
- Portas: 1x Mini HDMI; 1x Micro USB para dados ; 1x Micro USB para alimentação

### **3.3.2.3 Armazenamento**

- Slot para cartão MicroSD para armazenamento do sistema operacional e dados.

### **3.3.2.4 Conectividade**

- GPIO (General Purpose Input/Output) com 40 pinos para conexão de sensores e outros periféricos.
- Opcionalmente, pode ser expandido com módulos de Wi-Fi e Bluetooth.

### **3.3.2.5 Ecossistema de Apoio**

A Fundação Raspberry Pi, responsável pelo desenvolvimento do Raspberry Pi, promove um ecossistema vasto e de suporte abrangente, que inclui:

- Documentação: Extensa documentação oficial que cobre desde a configuração inicial até projetos avançados.
- Comunidade: Uma grande comunidade global de desenvolvedores, engenheiros e entusiastas que compartilham conhecimento, projetos e oferecem suporte através de fóruns, blogs e eventos.
- Recursos Educacionais: A Fundação oferece recursos educacionais como tutoriais, cursos e guias de projetos que ajudam iniciantes e profissionais a explorar as capacidades do Raspberry Pi.
- Software: Um amplo suporte de software, incluindo sistemas operacionais otimizados como o Raspberry Pi OS, além de

compatibilidade com diversas linguagens de programação como Python, C, e Java.

A seguir relatamos as principais configurações do Sistema de Segurança:

### **3.3.3 Configuração Inicial e Configuração Básica**

#### **3.3.3.1 Configuração de Hardware**

- Integração do RainMaker e ESP32 com quatro sensores PIR posicionados em pontos de entrada crítico;
- Simulação do movimento dos membros da família para criar uma linha de base para atividades normais;

#### **3.3.3.2 Configuração de Software**

- Implementação de um script Python básico (01\_setup\_camera\_send\_email.py) para configuração da câmera e alertas por e-mail.
- Garantia de que o ESP32 estava corretamente integrado com os sensores PIR para registrar dados de movimento.

### **3.3.4 Desenvolvimento do Servidor Web Flask**

#### **3.3.4.1 Inicialização do Servidor Web**

- Criação de um servidor web Flask (02\_flask\_web\_server.py) para gerenciar a interface web do sistema.
- Configuração de rotas para exibir imagens capturadas e gerenciar o diretório da câmera.

#### **3.3.4.2 Modificações Críticas**

- Adição de uma função para verificar a foto mais recente e exibir uma mensagem se novas fotos fossem capturadas.
- Desenvolvimento de um recurso para limpar o diretório da câmera, mantendo apenas a foto mais recente.

### **3.3.5 Integração com Recursos de Segurança**

### **3.3.5.1 Aprimoramentos**

- Integração de recursos de segurança na aplicação Flask, incluindo autenticação de usuário e gerenciamento de sessões.
- Configuração de chaves secretas do aplicativo para segurança aprimorada.

### **3.3.5.2 Modificações Críticas**

- Refinamento da rota check-movement para atualizar dinamicamente e exibir a foto mais recente capturada.
- Melhoria do mecanismo de limpeza de diretórios para notificar os usuários após a limpeza bem-sucedida e redirecionar para a página inicial.

## **3.3.6 Depuração e Tratamento de Erros**

### **3.3.6.1 Depuração**

- Resolução de problemas relacionados a caminhos de arquivos e arquivos de imagem ausentes, garantindo relatórios de erros precisos.
- Correção de discrepâncias em endereços IP e sincronização de tempo para garantir entradas de log precisas.

### **3.3.6.2 Modificações Críticas**

- Garantia de tratamento robusto de erros na aplicação Flask para gerenciar graciosamente erros de arquivo não encontrado e outras exceções.
- Ajuste fino do mecanismo de registro para fornecer logs claros e acionáveis para fins de depuração.

## **3.3.7 Testes Finais e Implantação**

### **3.3.7.1 Testes**

- Realização de testes abrangentes para garantir que todos os recursos funcionassem corretamente, incluindo detecção de movimento, captura de imagens e limpeza de diretórios.

- Validação do desempenho do sistema em diferentes cenários para garantir confiabilidade e responsividade.

### **3.3.7.2 Implantação**

- Implantação bem-sucedida da aplicação Flask no Raspberry Pi, garantindo que ela funcione como um serviço para operação contínua.
- Verificação da funcionalidade de ponta a ponta do sistema de segurança residencial, desde a detecção de movimento até as notificações aos usuários.

### **3.3.8 Principais Recursos**

#### **3.3.8.1 Detecção de Movimento**

- Utiliza sensores PIR para detectar movimentos em pontos de entrada críticos;
- Captura imagens ao detectar movimento e registra os eventos.

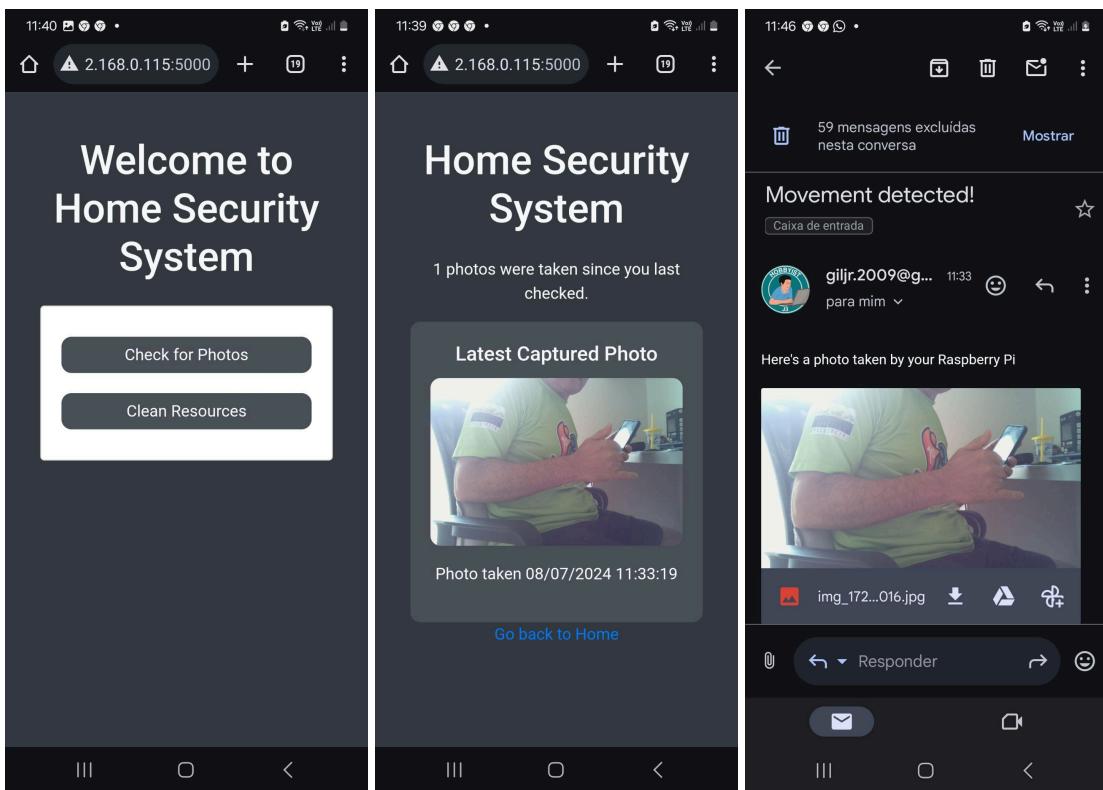
#### **3.3.8.2 Interface Web**

- Fornece uma interface amigável para visualizar imagens capturadas e gerenciar o sistema;
- Apresenta botões para verificar as fotos mais recentes e limpar o diretório da câmera.

#### **3.3.8.3 Segurança**

- Implementada autenticação de usuário e gerenciamento de sessões para acesso seguro.
- Utiliza chaves secretas para segurança aprimorada da aplicação web.
- Tratamento de Erros e Notificações:
- Tratamento robusto de erros para gerenciar arquivos ausentes e outros problemas.
- Envia notificações aos usuários após a limpeza bem-sucedida do diretório e outros eventos significativos.

A seguir, apresentamos as imagens de uso do aplicativo em dispositivos móveis. Essas páginas são fornecidas pelo servidor Flask, que está sendo executado em um Raspberry Pi Zero conectado a uma mini-câmara. Além de receber as imagens via e-mail cadastrado, também recebemos mensagens em tempo real geradas pelo aplicativo RainMaker, que está rodando em cada nó das placas Heltec ESP32. O sistema possui ampla capacidade para acréscimo de novas funcionalidades.



**Figuras 3 - 5: Demonstração da Implementação do Sistema de Detecção de Intrusão Residencial.** (Fonte: aplicativo *Home Security System* do autor).

A interface do sistema é composta por botões de acionamento de serviços. Os dois serviços implementados foram acionados e a leitura em tempo real da foto é também carregada via e-mail configurável. Dessa forma, o sistema notifica imediatamente o proprietário sobre a intrusão detectada. Em produção, o sistema será aperfeiçoado para ser adaptado em conjunto com os sistemas de iluminação, em arandelas em muros e paredes em pontos críticos da planta residencial.

### 3.3.9 Técnicas de Simulação de Voo

#### 3.3.9.1 Simulando o Voo QuadCopter com Mission Planner da Ardupilot

O objetivo do projeto Mission Planner é fornecer um software de estação de controle terrestre (GCS) para drones baseados no ArduPilot<sup>6</sup>. Ele permite que os usuários planejem, controlem e monitorem as missões de voo de seus veículos

aéreos não tripulados (UAVs). O Mission Planner oferece uma ampla gama de recursos, incluindo planejamento de missões, telemetria em tempo real, análise de dados de voo e configuração do veículo.

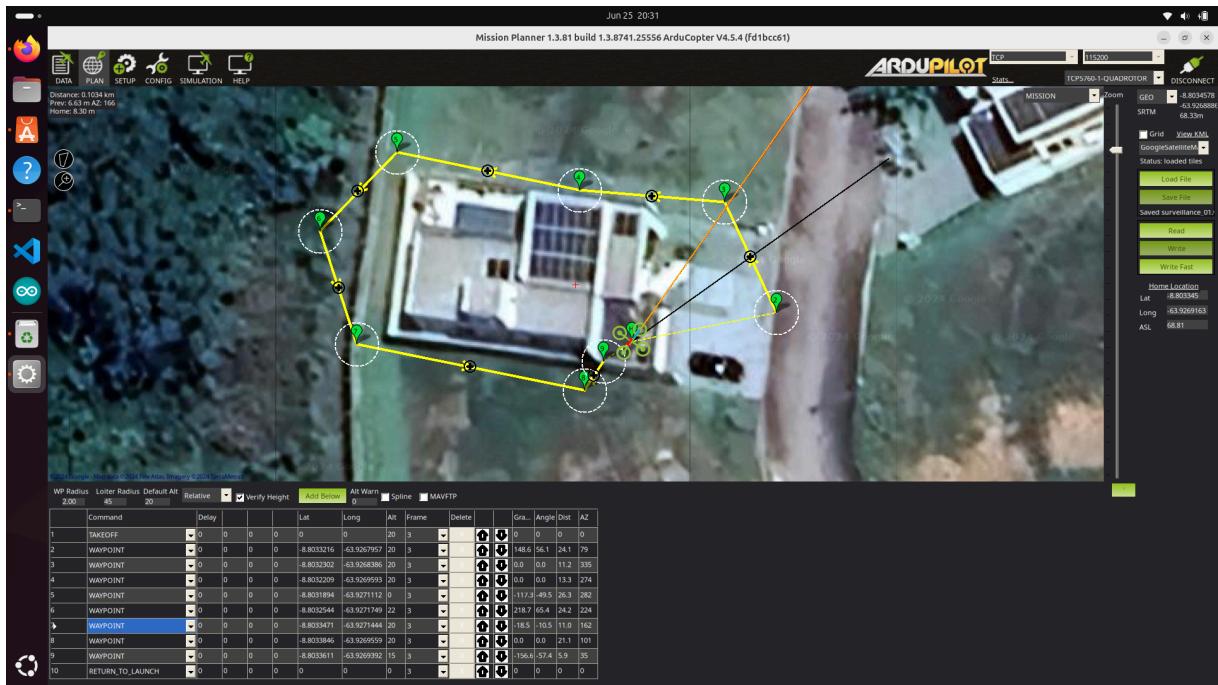
O Mission Planner é uma ferramenta essencial para usuários do sistema ArduPilot, fornecendo uma interface intuitiva para planejamento e monitoramento de missões. Com sua interface gráfica amigável, permite a configuração de parâmetros de voo, definição de rotas e até mesmo a análise de dados pós-voo. Além disso, oferece recursos avançados, como a visualização em 3D do terreno e a simulação de missões. Sua compatibilidade com uma variedade de plataformas e a capacidade de integração com diferentes tipos de aeronaves tornam-no uma escolha popular entre entusiastas, pesquisadores e profissionais da indústria. Graças à sua comunidade ativa, o Mission Planner é constantemente atualizado e aprimorado, garantindo que esteja sempre em sintonia com as necessidades dos usuários. Sua versatilidade e robustez o tornam uma peça fundamental no arsenal de qualquer operador que dependa do ArduPilot para suas operações aéreas.

### **3.3.10 Construção da Interface**

A interface do Mission Planner é construída usando Windows Forms, que é uma biblioteca de classes gráficas (GUI) incluída como parte do Microsoft .NET Framework<sup>7</sup>. A interface é projetada para ser amigável e intuitiva, fornecendo várias ferramentas e recursos, como:

- Vista do Mapa: Para planejamento e monitoramento de missões.
- HUD (Heads-Up Display): Mostrando dados de voo em tempo real.
- Páginas de Configuração: Para configurar os parâmetros do veículo.
- Gráficos: Para analisar logs de voo e dados de telemetria.
- Botões e Menus: Para executar vários comandos e acessar diferentes funcionalidades.

Segue a configuração realizada especificamente para este projeto utilizando o Mission Planner. Utilizamos a localização de uma residência na cidade de Porto Velho, Rondônia. Este mapeamento foi modelado via Google Maps e pode ser replicado para qualquer outra localização no globo terrestre.



**Figura 6 - Interface do Aplicativo Ardupilot, Plano de Voo no *Mission Planner*.** (fonte: aplicativo Mission Planner)

Aplicativo *Mission Planner*: o Plano de Voo nº 1 consiste em um comando de decolagem, oito comandos de *waypoint* e um comando de pouso. Cada comando especifica as coordenadas e a altitude, juntamente com outros parâmetros que são definidos para valores padrões. O plano de voo é detalhado e preciso para guiar o drone autonomamente em missões complexas. Nesse primeiro plano de voo, foi traçada rota ao redor da residência para acionamento automático via plataforma RainMaker. (Fonte: foto do autor).

### 3.3.11 Uso da API Drone Kit para lançamento automático

O Mission Planner e o ArduPilot fornecem APIs para automação e scripting, incluindo lançamento automático e outros comandos<sup>8</sup>.

- Scripting do Mission Planner: O Mission Planner suporta scripting através do uso do IronPython, permitindo que os usuários automatizem tarefas dentro do GCS. Scripts podem ser escritos e executados diretamente no Mission Planner.
- DroneKit: Uma API e SDK para comunicar e controlar drones usando Python. Ele fornece uma maneira fácil de escrever scripts para controlar UAVs alimentados pelo ArduPilot. Inclui capacidades para planejamento de missões, monitoramento do estado do veículo e comando de operações do veículo.

Utilizamos a API do DroneKit para lançar um drone automaticamente. O código está registrado no apêndice, primeiro código - LAUNCH\_01.py.

## 4 Resultado e Discussões

### 4.1 Aplicação Prática

O procedimento prático envolve etapas como a configuração dos drones e da ESP32, o desenvolvimento do firmware e do software, a implementação da inteligência artificial para análise de imagens, a integração com câmeras e sensores, entre outros. Documentamos detalhadamente o desenvolvimento e as etapas de implementação.

Seguem os principais diretórios configurados estabelecidos no Raspberry Pi:

```
j3@j3-LAPTOP:~$ ssh pi@raspberrypi.local
pi@raspberrypi.local's password:
Linux raspberrypi 6.6.36-v8+ #1780 SMP PREEMPT Mon Jul  1 19:39:42 BST 2024 aarch64

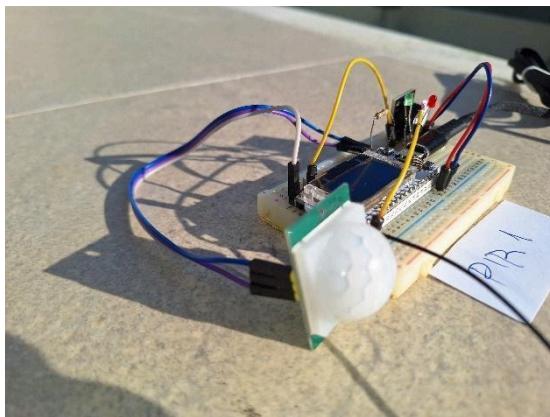
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Jul 10 18:09:47 2024 from 192.168.0.106
pi@raspberrypi:~/camera/
pi@raspberrypi:~/camera $ ls
img_1720399098.4081368.jpg  img_1720446637.6448474.jpg  img_1720449157.7882557.jpg
img_1720399689.5556402.jpg  img_1720446713.0704556.jpg  img_1720452442.4797432.jpg
img_1720430865.5804842.jpg  img_1720446778.7471838.jpg  img_1720452626.1085103.jpg
img_1720434145.7167287.jpg  img_1720446845.4567971.jpg  img_1720452799.7824016.jpg
img_1720445222.163508.jpg   img_1720446910.8129272.jpg  img_1720533155.1116414.jpg
img_1720445290.5906005.jpg  img_1720446976.517244.jpg   img_1720534003.4070358.jpg
img_1720445638.8679683.jpg  img_1720447045.0155106.jpg  img_1720628799.7431762.jpg
img_1720445990.4285123.jpg  img_1720447112.1760077.jpg  img_1720628909.8370125.jpg
img_1720446360.487345.jpg   img_1720447271.271202.jpg   img_1720630926.1542807.jpg
img_1720446427.112843.jpg   img_1720447358.613697.jpg   img_1720715704.9009356.jpg
img_1720446505.1703265.jpg  img_1720447444.5704348.jpg   log
img_1720446571.9948595.jpg  img_1720447510.1168041.jpg
pi@raspberrypi:~/camera $ cd ~/Documents/python_programs/
pi@raspberrypi:~/Documents/python_programs $ ls
01_setup_camera_send_email.py      02_flask_web_server.py
01_setup_camera_send_email.py.save  templates
pi@raspberrypi:~/Documents/python_programs $ cd templates/
pi@raspberrypi:~/Documents/python_programs/templates $ ls
index.html  secure_house.html
```

**Foto 7. Diretórios utilizados no procedimento técnico de testes do protótipo.** (Fonte: figura do terminal da CPU do autor).

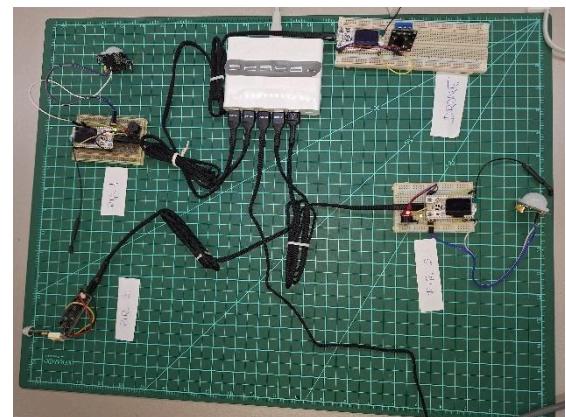
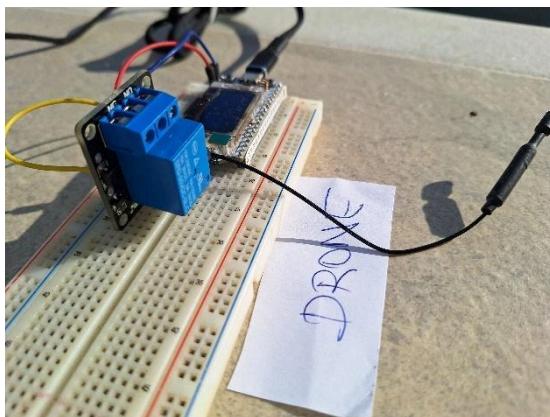
O sistema 'Home Security System' salva as fotos capturadas na raiz `/camera`. Os serviços de envio de e-mails e o servidor web estão localizados no diretório `Documents/python_programs/`. As visualizações renderizadas pelas páginas `index` e `secure_house.html` estão dentro do diretório `templates/`.

Seguem os protótipos para prova de conceito:



**Fotos 8-9 - As fotos apresentam os protótipos com os sensores *PIR 1* e *PIR 3*, integrados com o módulo *ESP32 Heltec* e conectividade *Wi-Fi*.** (Fonte: foto do autor).

Esses protótipos são suportados por antenas que oferecem um alcance de até 100 metros. Estes testes de bancada foram fundamentais para validar os conceitos, funcionalidades e a viabilidade do sistema.



**Fotos 10-11 – Fotos da plataforma de lançamento do Drone e dos testes de bancada.** (Fonte: foto do autor).

Esses foram fundamentais para validar os conceitos e a funcionalidade do sistema. Durante esses testes, os componentes foram conectados e configurados para operar de forma autônoma. O sistema de fiação utilizado nos testes será substituído, na produção, por placas solares e baterias também autônomas. Esta substituição visa proporcionar facilidade na implantação, eliminando a necessidade de infraestrutura complexa. A utilização de fontes de energia autônomas, como placas solares e baterias, também aumenta a sustentabilidade e a eficiência energética do sistema.

#### **4.1.2 Codificação**

Para demonstrar a aplicação prática do sistema, implementamos o código que prepara a plataforma de lançamento do drone e um sistema de segurança usando quatro sensores PIR e ESP32 com integração ao ESP RainMaker. Incluímos a configuração de dispositivos, detecção de movimento via interrupções, e o envio de alertas. Para os códigos vide Apêndice.

##### **4.1.2.1 Código de Preparação da Plataforma de Lançamento do Drone**

01\_LAUNCH.py

##### **4.1.2.2 Código do Sistema de Segurança Sensor PIR e CALLBACK**

02\_PIR\_NO\_BUZ1.ino e 03\_PIR\_CB\_BUZ2.ino

##### **4.1.2.3 Explicação Resumida dos Códigos**

Os códigos fornecidos configuram e implementam um sistema de segurança residencial utilizando um ESP32, um sensor *PIR* (Passive Infrared Sensor) para detecção de movimento e um *buzzer* para alertas sonoros. A integração é feita através da plataforma ESP RainMaker, que permite o controle e a monitorização remota dos dispositivos interligados<sup>9</sup>.

##### **4.1.2.4 Código de Configuração Básica de Segurança**

- Importação de Bibliotecas: As bibliotecas *RMaker*, *WiFi*, e *WiFiProv* são incluídas para habilitar a funcionalidade de ESP RainMaker e *WiFi*<sup>10</sup>.
- Definição de Pinos: Define pinos para *LED*, botão de *reset*, sensor *PIR*, e *buzzer*.
- Configuração de Estados: Variáveis são definidas para armazenar os estados do sensor *PIR* e do *buzzer*.
- Funções de Callback e Eventos: *sysProvEvent* lida com eventos do sistema, como a iniciação da provisão e conexão WiFi. *write\_callback* lida com atualizações de parâmetros do dispositivo.

- Setup Inicial: Configura os pinos, inicializa o nó ESP RainMaker, adiciona dispositivos (interruptor de segurança), e inicia os serviços de OTA, agendamento e provisionamento.
- Loop Principal: Lê o estado do botão de reset para permitir reset de fábrica ou de WiFi, verifica o estado da conexão WiFi e chama as funções para detectar movimento (*detectMotion*) e controlar o buzzer (*controlBuzzer*).

#### **4.1.2.5 Código de Detecção de Movimento e Controle do Buzzer**

- Função *detectMotion*: Lê o estado do sensor PIR. Se o movimento for detectado, aciona um alerta, liga o buzzer e inicia um temporizador.
- Função *controlBuzzer*: Desliga o buzzer após 5 segundos se estiver ligado.

Ambos os códigos são configurados para utilizar a plataforma ESP RainMaker, permitindo a integração com serviços de nuvem, configuração via BLE ou SoftAP, e controle remoto via aplicativo.

Quando carregado o código no dispositivo com Android ou iOS, o aplicativo solicita o serviço de credenciamento, em seguida passa para as fases:

#### **4.1.2.6 Carregamento do Código**

O código é carregado no dispositivo, que inclui as bibliotecas necessárias para conectar à plataforma RainMaker.

#### **4.1.2.7 Inicialização**

O dispositivo é inicializado e começa a procurar redes Wi-Fi disponíveis para conexão.

#### **4.1.2.8 Conexão Wi-Fi**

O dispositivo se conecta à rede Wi-Fi configurada, permitindo a comunicação com a plataforma RainMaker.

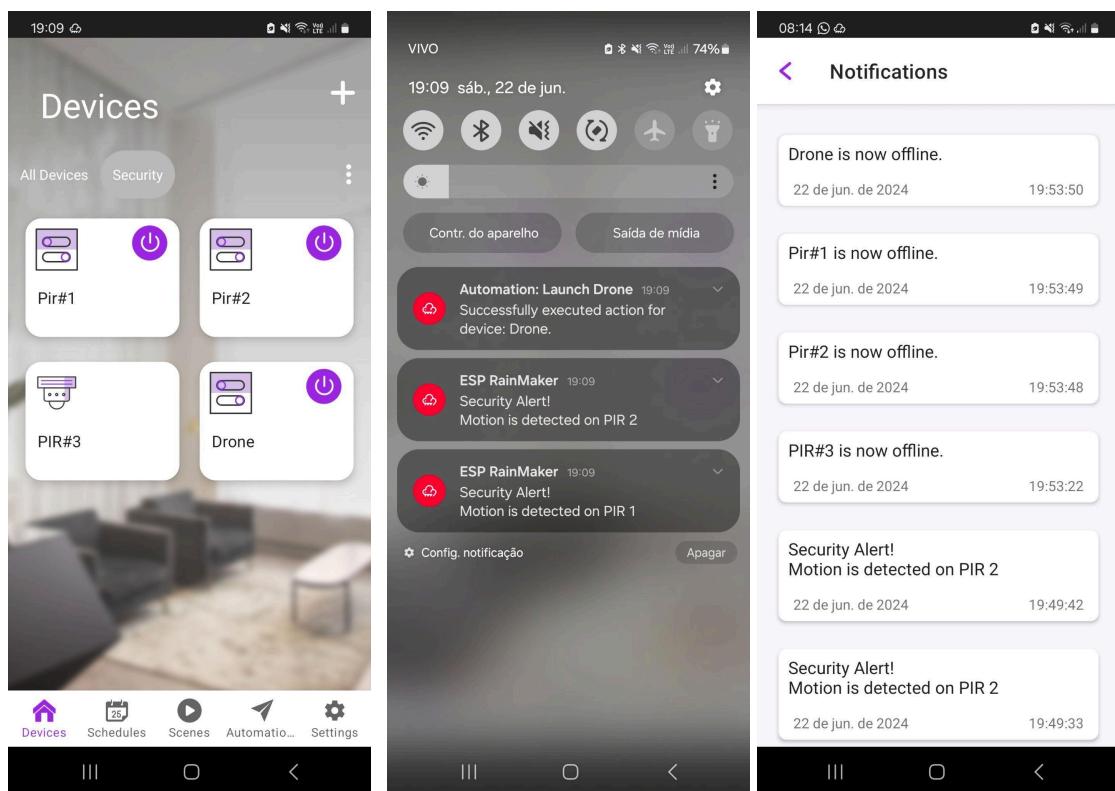
#### **4.1.2.9 Credenciamento**

O dispositivo envia uma solicitação de credenciamento à plataforma RainMaker. A plataforma RainMaker autentica o dispositivo e vincula-o à conta do usuário. Um token de autenticação é gerado e enviado ao dispositivo, confirmando a credencial e a conexão segura.

#### 4.1.2.10 Configuração e Controle

Após o credenciamento bem-sucedido, o dispositivo pode ser configurado e controlado remotamente através da interface do RainMaker. Você pode definir parâmetros, monitorar o status e enviar comandos ao dispositivo.

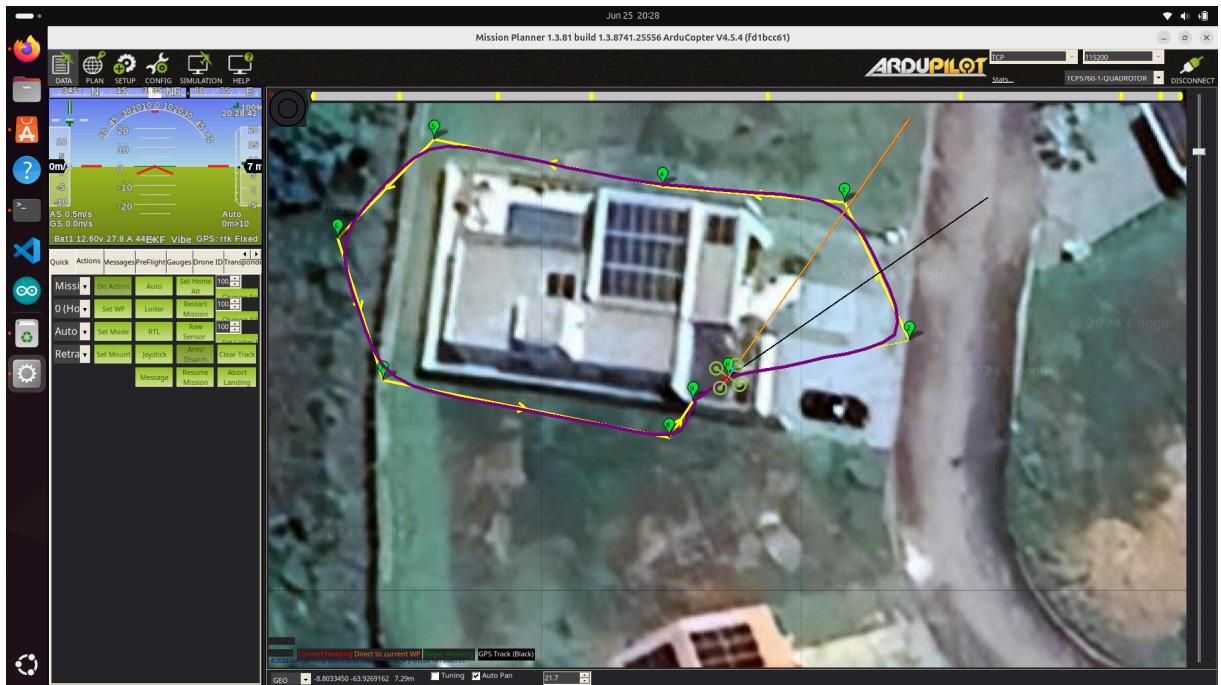
Seguem telas extraídas do dispositivo móvel durante os teste de bancada:



**Figuras 12 - 14: Implementação do Sistema de Detecção de Intrusão Residencial.** (Fonte: aplicativo RainMaker da Expressif).

A interface é composta por quatro componentes *RainMaker*: três sensores *P/IR* (*Passive Infrared Sensor*) monitorando pontos críticos da residência, exemplificando áreas de vigilância. Um desses sensores, identificado como *PIR#3*, situado acima à esquerda, aciona o drone para executar o plano de voo pré-estabelecido quando detecta movimento por mais de 3 segundos. No centro e à direita, são apresentados os sistemas de notificação nativos da aplicação móvel, que avisam o proprietário sobre o evento de intrusão. A prova de conceito funcionou conforme esperado, utilizando a rede *Wi-Fi* da residência para enviar os sinais em tempo real.

Esse processo permite que o dispositivo se integre de forma segura e eficiente à plataforma RainMaker, facilitando o gerenciamento de dispositivos IoT em sua rede.



**Figura 15 - Simulação do Plano de Voo nº 1 registrado na Figura 6.** (Fonte: aplicativo Mission Planner).

Este plano de voo é responsável pelo monitoramento da residência utilizando o sistema de detecção de intrusos. O drone segue uma trajetória que começa no teto da garagem, passando pela frente e ao redor da residência, transmitindo áudio e vídeo em tempo real. A utilização de georreferenciamento e integração com o *Google Maps* são totalmente configuráveis, permitindo que a aplicação funcione em qualquer localidade no globo terrestre.

#### 4.3 Implementação da Inteligência Artificial

O projeto "DETECÇÃO DE INTRUSÕES RESIDENCIAIS USANDO DRONES E INTELIGÊNCIA ARTIFICIAL" utiliza redes neurais recorrentes<sup>3</sup>, especificamente LSTMs, para analisar dados coletados por sensores PIR instalados em pontos estratégicos de uma residência. Os sensores monitoram a movimentação de uma família de quatro membros, gerando um fluxo contínuo de dados de presença. Esses dados são enviados para um servidor, onde são armazenados e processados em tempo real. A rede LSTM é treinada para reconhecer padrões de comportamento normal da família e identificar anomalias que possam indicar uma intrusão. A implementação envolve a configuração de um Raspberry Pi 4 para coleta e envio de

dados, uso de Python para desenvolvimento do modelo LSTM, e integração com serviços em nuvem para armazenamento e processamento dos dados.

## 5 Considerações Finais

O projeto "DETECÇÃO DE INTRUSÕES RESIDENCIAIS USANDO DRONES E INTELIGÊNCIA ARTIFICIAL" integrou com sucesso componentes de hardware e software para criar um sistema de segurança residencial abrangente e inovador. As modificações críticas e os recursos adicionados neste projeto aprimoraram significativamente a funcionalidade e a confiabilidade do sistema. A aplicação está agora pronta para implantação e comercialização, proporcionando uma solução segura e eficiente para a detecção de intrusões residenciais.

A utilização do Raspberry Pi Zero, uma placa de computador de baixo custo e tamanho reduzido, mostrou-se ideal para aplicações compactas e de baixo consumo de energia. Equipado com um processador Broadcom BCM2835, com um núcleo ARM1176JZF-S rodando a 1GHz e 512MB de RAM, o Raspberry Pi Zero oferece um desempenho adequado para as necessidades do projeto<sup>6</sup>. Além disso, a Fundação Raspberry Pi promove um ecossistema de apoio robusto, incluindo documentação abrangente, uma comunidade ativa e uma variedade de bibliotecas e ferramentas de software. A câmera ZERO-V1, uma câmera de 5 megapixels compatível com a linha Raspberry Pi Zero, proporciona alta qualidade de imagem e fácil integração com a placa, permitindo a implementação eficiente do sistema de detecção de intrusão residencial.

A API ESP RainMaker é uma ferramenta poderosa para o desenvolvimento de IoT, oferecendo uma variedade de tipos, parâmetros e funções padrão que simplificam a criação, configuração e gerenciamento de dispositivos IoT<sup>11</sup>. Esta ferramenta permite que os desenvolvedores se concentrem na inovação, garantindo um desempenho consistente e confiável, além de facilitar a integração perfeita em um ecossistema IoT mais amplo.

O desenvolvimento de um sistema de segurança residencial utilizando

drones, ESP32 e inteligência artificial demonstrou ser uma abordagem promissora para a proteção de residências<sup>3</sup>. Os exemplos de código apresentados ilustram a implementação prática e a capacidade de integrar diferentes componentes tecnológicos para criar uma solução de segurança eficiente e robusta.

Em conclusão, o projeto "DETECÇÃO DE INTRUSÕES RESIDENCIAIS USANDO DRONES E INTELIGÊNCIA ARTIFICIAL" não apenas alcançou seus objetivos iniciais, mas também abriu novas possibilidades para futuras melhorias e inovações na área de segurança residencial. A combinação de tecnologias avançadas e metodologias de desenvolvimento robustas resultou em um sistema confiável e de alto desempenho, pronto para contribuir significativamente para a proteção e segurança das residências modernas. Este projeto representa um passo importante na evolução das soluções de segurança residencial, evidenciando o potencial das tecnologias emergentes para transformar e melhorar nossas vidas cotidianas.

## Referências

<sup>2</sup>ABESE. Entendendo a Segurança Eletrônica: objetivos e benefícios. **Abese.Org.br**. Disponível em: <<https://www.abese.org.br/>>. Acesso em: 8, Julho 2024.

<sup>8</sup>API - the PiCamera class — picamera 1.13 documentation. (n.d.). **Readthedocs.io**, 2024. Disponível em: <[https://picamera.readthedocs.io/en/release-1.13/api\\_camera.html](https://picamera.readthedocs.io/en/release-1.13/api_camera.html)>. Acesso em: 8, junho 2024.

<sup>8</sup>ARDUPILOT. Cloud EcoSystem documentation. **ArduPilot**, 2024. Disponível em: <<https://cloud.ardupilot.org/>>. Acesso em: 26, junho 2024.

<sup>9</sup>CAMERON, N. . **Electronics projects with the ESP8266 and ESP32**: Building web pages, applications, and WiFi enabled devices. 1st ed. Berlim: APres, 2020.

<sup>6</sup>HOFFMAN, M. . **Raspberry pi**: Setup, programming & developing amazing projects with raspberry pi. Londres: Mike Hoffman, 2023.

<sup>10</sup>Designing Purpose - Built Drones for Ardupilot Pixhawk 2.1: Build drones with Ardupilot. **ArduPilot**, 2024. Disponível em: <<https://cloud.ardupilot.org/>>. Acesso em: 26, junho 2024

<sup>11</sup>ESP32, RainMaker Programming Guide — ESP RainMaker Programming Guide latest documentation - **Espressif.com**. Disponível em:

<<https://docs.espressif.com/projects/esp-rainmaker/en/latest/esp32/index.html>  
Acesso em: 11, julho 2024.

<sup>7</sup>NAKAMOTO, S. . **The Esp32 iot bible**: Android programming for ESP32 IoT devices build your own connected devices with android apps and node communication. Nova Iorque: Independently Published, 2023.

<sup>4</sup>GRANT, J. **Raspberry pi**: A comprehensive beginner's guide to setup, programming (concepts and techniques) and developing cool raspberry pi projects. Londres: Ingram Spark. 2023.

<sup>10</sup>ZULFIQAR, Asim. **Hands-on ESP32 with Arduino IDE**: Unleash the power of IoT with ESP32 and build exciting projects with this practical guide. Londres: Packt Publishing, 2024.

<sup>8</sup>ARDUPILOT. Introduction to plane: plane documentation. **Ardupilot**, 2024. Disponível em: <<https://ardupilot.org/plane/docs/introduction.html>>. Acesso em: 26, junho 2024.

<sup>4</sup>JONES, Michael. **The Versatile Applications of Drones**. London: Tech Books, 2020.

<sup>6</sup>KUMAR, P., Kaur, H., & Adrián, G. . **IOT raspberry pi projects**. LAP Lambert Academic. Pennsylvania: Publishing, 2020.

<sup>1</sup>TAPPARI, S. . **Segurança e automação residencial inteligente baseada em IoT**. São Paulo: Edições Nossa Conhecimento, 2023.

<sup>6</sup>SMITH, John. **Security Systems Using PIR Sensors**. 2nd ed. New York.2024.

<sup>3</sup>WILKINS, N. . **Robotics: What beginners need to know about robotic process automation, mobile robots, artificial intelligence, machine learning, autonomous vehicles, speech recognition, drones, and our future**. Texas: Independently Published, 2019.

## Apêndice

### 1 Código Sistema de Acionamento Do Drone

01\_LAUNCH.py

```
#include "RMaker.h"
#include "WiFi.h"
#include "WiFiProv.h"
#include "AppInsights.h"

#define DEFAULT_POWER_MODE true
```

```

const char *service_name = "PROV_1234";
const char *pop = "abcd1234";

// GPIO for push button
#if CONFIG_IDF_TARGET_ESP32C3
static int gpio_0 = 9;
static int gpio_switch = 16;
#else
// GPIO for virtual device
static int gpio_0 = 0;
static int gpio_switch = 16;
#endif

/* Variable for reading pin status*/
bool switch_state = true;

// The framework provides some standard device types like switch, lightbulb,
// fan, temperaturesensor.
static Switch *my_switch = NULL;

void sysProvEvent(arduino_event_t *sys_event)
{
    switch (sys_event->event_id) {
        case ARDUINO_EVENT_PROV_START:
#if CONFIG_IDF_TARGET_ESP32S2
            Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on SoftAP\n",
                           service_name, pop);
            printQR(service_name, pop, "softap");
#else
            Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on BLE\n",
                           service_name, pop);
            printQR(service_name, pop, "ble");
#endif
            break;
        case ARDUINO_EVENT_PROV_INIT:
            wifi_prov_mgr_disable_auto_stop(10000);
            break;
        case ARDUINO_EVENT_PROV_CRED_SUCCESS:
            wifi_prov_mgr_stop_provisioning();
            break;
        default:;
    }
}

void write_callback(Device *device, Param *param, const param_val_t val,
                   void *priv_data, write_ctx_t *ctx)
{
    const char *device_name = device->getDeviceName();
    const char *param_name = param->getParamName();

    if (strcmp(param_name, "Power") == 0) {
        Serial.printf("Received value = %s for %s - %s\n",
                      val.val.b ? "true" : "false", device_name, param_name);
        switch_state = val.val.b;
    }
}

```

```

        (switch_state == false) ? digitalWrite(gpio_switch, LOW)
        : digitalWrite(gpio_switch, HIGH);
    param->updateAndReport(val);
}
}

void setup()
{
    Serial.begin(115200);
    pinMode(gpio_0, INPUT);
    pinMode(gpio_switch, OUTPUT);
    digitalWrite(gpio_switch, DEFAULT_POWER_MODE);

    Node my_node;
    my_node = RMaker.initNode("ESP RainMaker Node");

    // Initialize Drone on Air device
    my_switch = new Switch("Drone", &gpio_switch);
    if (!my_switch) {
        return;
    }
    // Standard switch device
    my_switch->addCb(write_callback);

    // Add switch device to the node
    my_node.addDevice(*my_switch);

    // This is optional
    RMaker.enableOTA(OTA_USING_TOPICS);
    // If you want to enable scheduling, set time zone for your region using
    // setTimeZone(). The list of available values are provided here
    // https://rainmaker.espressif.com/docs/time-service.html
    // RMaker.setTimeZone("Asia/Shanghai");
    // Alternatively, enable the Timezone service and let the phone apps set
the
    // appropriate timezone
    RMaker.enableTZService();

    RMaker.enableSchedule();

    RMaker.enableScenes();
    // Enable ESP Insights. Insteads of using the default http transport, this
function will
    // reuse the existing MQTT connection of Rainmaker, thereby saving memory
space.
    initAppInsights();

    RMaker.enableSystemService(SYSTEM_SERV_FLAGS_ALL, 2, 2, 2);

    RMaker.start();

    WiFi.onEvent(sysProvEvent);
#ifndef CONFIG_IDF_TARGET_ESP32S2
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_SOFTAP,
WIFI_PROV_SCHEME_HANDLER_NONE,
                                WIFI_PROV_SECURITY_1, pop, service_name);

```

```

#else
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
WIFI_PROV_SCHEME_HANDLER_FREE_BTDM,
                                WIFI_PROV_SECURITY_1, pop, service_name);
#endif
}

void loop()
{
    if (digitalRead(gpio_0) == LOW) { // Push button pressed

        // Key debounce handling
        delay(100);
        int startTime = millis();
        while (digitalRead(gpio_0) == LOW) {
            delay(50);
        }
        int endTime = millis();

        if (((endTime - startTime) > 10000) {
            // If key pressed for more than 10secs, reset all
            Serial.printf("Reset to factory.\n");
            RMakerFactoryReset(2);
        } else if ((endTime - startTime) > 3000) {
            Serial.printf("Reset Wi-Fi.\n");
            // If key pressed for more than 3secs, but less than 10, reset
Wi-Fi
            RMakerWiFiReset(2);
        } else {
            // Toggle device state
            switch_state = !switch_state;
            Serial.printf("Toggle State to %s.\n", switch_state ? "true" :
"false");
            if (my_switch) {
                my_switch->updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME,
                                                switch_state);
            }
            (switch_state == false) ? digitalWrite(gpio_switch, LOW)
            : digitalWrite(gpio_switch, HIGH);
        }
    }
    delay(100);
}

```

## 2 Código Sistema de Segurança com Sensor PIR NO CALLBACK

02\_PIR\_NO\_BUZ1.ino

```

#include "RMaker.h"
#include "WiFi.h"
#include "WiFiProv.h"
#include "AppInsights.h"

```

```

#define DEFAULT_POWER_MODE true
const char *service_name = "PROV_1234";
const char *pop = "abcd1234";

// GPIO for motion sensor
#if CONFIG_IDF_TARGET_ESP32C3
static int gpio_motion_sensor = 7;
#else
// GPIO for virtual device
static int gpio_motion_sensor = 16;
#endif

/* Variable for reading pin status*/
bool motion_detected = false;

// The framework provides some standard device types like switch, lightbulb,
fan, temperature sensor.
static Device *my_motion_sensor = NULL;
// Set the device name (PIR#number)
const char *DNAME = "PIR#3";

void sysProvEvent(arduino_event_t *sys_event)
{
    switch (sys_event->event_id) {
    case ARDUINO_EVENT_PROV_START:
#if CONFIG_IDF_TARGET_ESP32S2
        Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on SoftAP\n", service_name, pop);
        printQR(service_name, pop, "softap");
#else
        Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on BLE\n", service_name, pop);
        printQR(service_name, pop, "ble");
#endif
        break;
    case ARDUINO_EVENT_PROV_INIT:
        wifi_prov_mgr_disable_auto_stop(10000);
        break;
    case ARDUINO_EVENT_PROV_CRED_SUCCESS:
        wifi_prov_mgr_stop_provisioning();
        break;
    default:;
    }
}

void write_callback(Device *device, Param *param, const param_val_t val, void
*priv_data, write_ctx_t *ctx)
{
    const char *device_name = device->getDeviceName();
    const char *param_name = param->getParamName();

    if (strcmp(param_name, "Motion Detection State") == 0) {
        Serial.printf("Received value = %s for %s - %s\n", val.val.b ? "true"
: "false", device_name, param_name);
        motion_detected = val.val.b;
        param->updateAndReport(val);
    }
}

```

```

    }

}

void setup()
{
    Serial.begin(115200);
    pinMode(gpio_motion_sensor, INPUT);

    Node my_node;
    my_node = RMaker.initNode("ESP RainMaker Node");

    // Initialize motion sensor device
    my_motion_sensor = new Device("Motion Sensor",
"esp.device.motion-sensor");
    if (!my_motion_sensor) {
        return;
    }
    // Add standard name parameter
    my_motion_sensor->addNameParam(DNAME);

    // Initialize the motion detection state parameter value
    esp_rmaker_param_val_t motion_detected_val =
esp_rmaker_bool(motion_detected);

    // Add motion detection state switch parameter
    Param motion_switch_param("Motion Detection State",
ESP_RMAKER_PARAM_TOGGLE, motion_detected_val, PROP_FLAG_READ);
    motion_switch_param.addUIType(ESP_RMAKER_UI_TOGGLE);
    my_motion_sensor->addParam(motion_switch_param);

    // Set write callback for motion sensor
    my_motion_sensor->addCb(write_callback);

    // Add motion sensor device to the node
    my_node.addDevice(*my_motion_sensor);

    // This is optional
    RMaker.enableOTA(OTA_USING_TOPICS);
    // If you want to enable scheduling, set time zone for your region using
setTimeZone().
    // The list of available values are provided here
https://rainmaker.espressif.com/docs/time-service.html
    // RMaker.setTimeZone("Asia/Shanghai");
    // Alternatively, enable the Timezone service and let the phone apps set
the appropriate timezone
    RMaker.enableTZService();

    RMaker.enableSchedule();
    RMaker.enableScenes();

    // Enable ESP Insights. Instead of using the default http transport, this
function will
    // reuse the existing MQTT connection of Rainmaker, thereby saving memory
space.
    initAppInsights();
}

```

```

RMaker.enableSystemService(SYSTEM_SERV_FLAGS_ALL, 2, 2, 2);

RMaker.start();

WiFi.onEvent(sysProvEvent);
#if CONFIG_IDF_TARGET_ESP32S2
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_SOFTAP,
    WIFI_PROV_SCHEME_HANDLER_NONE, WIFI_PROV_SECURITY_1, pop, service_name);
#else
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
    WIFI_PROV_SCHEME_HANDLER_FREE_BTDM, WIFI_PROV_SECURITY_1, pop, service_name);
#endif
}

void loop()
{
    // Check motion sensor state
    bool current_state = digitalRead(gpio_motion_sensor);
    if (current_state != motion_detected) {
        motion_detected = current_state;

        // Retrieve the device name
        const char *device_name = DNAME;

        Serial.printf("Motion detected: %s on %s\n", motion_detected ? "true"
        : "false", device_name);

        if (my_motion_sensor) {
            // Update and report the boolean value
            my_motion_sensor->updateAndReportParam("Motion Detection State",
motion_detected);
        }
    }
    delay(100);
}
}

```

### 3 Código Sistema de Segurança com Sensor PIR – CALLBACK

03\_PIR\_CB\_BUZ2.ino

```

//-----
#include "RMaker.h"
#include "WiFi.h"
#include "WiFiProv.h"
//-----
//BLE credentials - no need to change
const char *service_name = "PROV_1234";
const char *pop = "abcd1234";
//-----
static uint8_t WIFI_LED    = 2;
static uint8_t RESET_PIN   = 0;
static uint8_t PIR_PIN     = 16;
static uint8_t BUZZER_PIN  = 21;

```

```

//-----
//PIR Motion Sensor
boolean PIR_STATE_NEW      = LOW; // current state
boolean PIR_STATE_OLD       = LOW; // previous state
//-----
boolean BUZZER_STATE        = false;
unsigned long buzzer_timer  = 0;
//-----
char device1[] = "Security_Pir_2";
/*The framework provides some standard device types
//like switch, lightbulb, fan, temperature sensor.*/
static Switch SecuritySwitch(device1, NULL);
/* the current status of the security switch*/
bool SECURITY_STATE = false;
//-----
uint32_t chipId = 0;
//-----


***** * sysProvEvent Function *****
***** * write_callback Function *****

void sysProvEvent(arduino_event_t *sys_event)
{
    switch (sys_event->event_id) {
        case ARDUINO_EVENT_PROV_START:
#if CONFIG_IDF_TARGET_ESP32
            Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on BLE\n", service_name, pop);
            printQR(service_name, pop, "ble");
#else
            Serial.printf("\nProvisioning Started with name \"%s\" and PoP \"%s\""
on SoftAP\n", service_name, pop);
            printQR(service_name, pop, "softap");
#endif
            break;
        case ARDUINO_EVENT_WIFI_STA_CONNECTED:
            Serial.printf("\nConnected to Wi-Fi!\n");
            digitalWrite(WIFI_LED, HIGH);
            break;
    }
}

void write_callback(Device *device, Param *param, const param_val_t val, void
*priv_data, write_ctx_t *ctx)
{
    const char *device_name = device->getDeviceName();
    const char *param_name = param->getParamName();
    //-----
    if(strcmp(device_name, device1) == 0) {

        Serial.printf("SecuritySwitch = %s\n", val.val.b? "true" : "false");

        if(strcmp(param_name, "Power") == 0) {

```

```

        Serial.printf("Received value = %s for %s - %s\n",
                      val.val.b? "true" : "false", device_name, param_name);
        SECURITY_STATE = val.val.b;
        param->updateAndReport(val);
        //-----
        if(SECURITY_STATE == true){
            esp_rmaker_raise_alert("Security is ON");
        }
        else{
            esp_rmaker_raise_alert("Security is OFF");
        }
        //-----
    }

}
//-----
}

/*****
 * setup Function
 *****/
void setup(){
//-----
Serial.begin(115200);
//-----
pinMode(RESET_PIN, INPUT);
pinMode(WIFI_LED, OUTPUT);
digitalWrite(WIFI_LED, LOW);
//-----
pinMode(BUZZER_PIN, OUTPUT);
pinMode(PIR_PIN, INPUT);
//-----
Node my_node;
my_node = RMaker.initNode("Ahmad_Logs");
//-----
//Standard switch device
SecuritySwitch.addCb(write_callback);
//-----
//Add switch device to the node
my_node.addDevice(SecuritySwitch);
//-----
//This is optional
RMaker.enableOTA(OTA_USING_PARAMS);
//If you want to enable scheduling, set time zone for your region using
setTimeZone().
    //The list of available values are provided here
    https://rainmaker.espressif.com/docs/time-service.html
    // RMaker.setTimeZone("Asia/Shanghai");
    // Alternatively, enable the Timezone service and let the phone apps set the
    appropriate timezone
    RMaker.enableTZService();
    RMaker.enableSchedule();
//-----
//Service Name
for(int i=0; i<17; i=i+8) {
    chipId |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
}

```

```

}

Serial.printf("\nChip ID: %d Service Name: %s\n", chipId, service_name);
//-----
Serial.printf("\nStarting ESP-RainMaker\n");
RMaker.start();
//-----
WiFi.onEvent(sysProvEvent);
#if CONFIG_IDF_TARGET_ESP32
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_BLE,
WIFI_PROV_SCHEME_HANDLER_FREE_BTDM, WIFI_PROV_SECURITY_1, pop, service_name);
#else
    WiFiProv.beginProvision(WIFI_PROV_SCHEME_SOFTAP,
WIFI_PROV_SCHEME_HANDLER_NONE, WIFI_PROV_SECURITY_1, pop, service_name);
#endif
//-----
SecuritySwitch.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME, false);
//my_switch1.updateAndReportParam(ESP_RMAKER_DEF_POWER_NAME, STATE_RELAY_1);

//Serial.printf("Relay2 is %s \n", STATE_RELAY_2? "ON" : "OFF");
//-----
}

/*****************
 * loop Function
*****************/
void loop()
{
//-----
// Read GPIO0 (external button to reset device
if(digitalRead(RESET_PIN) == LOW) { //Push button pressed
    Serial.printf("Reset Button Pressed!\n");
    // Key debounce handling
    delay(100);
    int startTime = millis();
    while(digitalRead(RESET_PIN) == LOW) delay(50);
    int endTime = millis();
    //
    if ((endTime - startTime) > 10000) {
        // If key pressed for more than 10secs, reset all
        Serial.printf("Reset to factory.\n");
        RMakerFactoryReset(2);
    }
    //
    else if ((endTime - startTime) > 3000) {
        Serial.printf("Reset Wi-Fi.\n");
        // If key pressed for more than 3secs, but less than 10, reset Wi-Fi
        RMakerWiFiReset(2);
    }
    //
}
//-----
delay(100);

if (WiFi.status() != WL_CONNECTED){
    //Serial.println("WiFi Not Connected");
}

```

```

        digitalWrite(WIFI_LED, LOW);
    }
    else{
        //Serial.println("WiFi Connected");
        digitalWrite(WIFI_LED, HIGH);
    }
    //-----
    detectMotion();
    controlBuzzer();
    //-----
}

/*****
 * PirSensor Function
 *****/
void detectMotion() {
    if(SECURITY_STATE == true) {
        PIR_STATE_OLD = PIR_STATE_NEW; // store old state
        PIR_STATE_NEW = digitalRead(PIR_PIN); //read new state
    }
    if(PIR_STATE_OLD == LOW && PIR_STATE_NEW == HIGH) {
        Serial.println("Motion detected!");
        esp_rmaker_raise_alert("Security Alert!\nMotion is detected on PIR 2");
        digitalWrite(BUZZER_PIN, HIGH);
        BUZZER_STATE = true;
        buzzer_timer = millis();
    }
}
//-
}

void controlBuzzer(){
    if (BUZZER_STATE == true) {
        if (millis() - buzzer_timer > 5000) {
            digitalWrite(BUZZER_PIN, LOW);
            BUZZER_STATE = false;
            buzzer_timer = 0;
        }
    }
}

```

#### 4 Código: Configuração e Envio de Email Via Raspberry Pi Zero

`01_setup_camera_send_email.py`

```

import RPi.GPIO as GPIO
import time, datetime
from picamera import PiCamera
import os
import yagmail

PIR_PIN = 4
LED_PIN = 17

```

```

LOG_FILE_NAME = "/home/pi/camera/log/photo_logs.txt"

def take_photo(camera):
    file_name = "/home/pi/camera/img_" + str(time.time()) + ".jpg"
    camera.capture(file_name)
    return file_name

def update_photo_log_file(photo_file_name):
    with open(LOG_FILE_NAME, "a") as f:
        f.write(photo_file_name)
        f.write("\n")

def send_email_with_photo(yagmail_client, file_name):
    yagmail_client.send(to="giljr.2009@gmail.com",
                        subject="Movement detected!",
                        contents="Here's a photo taken by your Raspberry
Pi",
                        attachments=file_name)

# Setup camera
camera = PiCamera()
camera.resolution = (720, 480)
camera.rotation = 180
print("Waiting 2 seconds to init the camera...")
time.sleep(2)
print("Camera setup OK.")

# Remove log file
if os.path.exists(LOG_FILE_NAME):
    os.remove(LOG_FILE_NAME)
    print("Log file removed.")

# Setup yagmail
password = "ogof orem rzqf dppx"
#with open("/home/pi/.local/share/.email_password", "r") as f:
#    password = f.read()
yag = yagmail.SMTP("giljr.2009@gmail.com", password)
print("Email sender setup OK.")

# Setup GPIOs
GPIO.setmode(GPIO.BCM)
GPIO.setup(PIR_PIN, GPIO.IN)
GPIO.setup(LED_PIN, GPIO.OUT)
GPIO.output(LED_PIN, GPIO.LOW)
print("GPIOs setup OK.")

MOV_DETECT_THRESHOLD = 3.0
last_pir_state = GPIO.input(PIR_PIN)
movement_timer = time.time()
MIN_DURATION_BETWEEN_2_PHOTOS = 60.0
last_time_photo_taken = 0

print("Everything has been setup.")

try:
    while True:
        time.sleep(0.01)
        pir_state = GPIO.input(PIR_PIN)

```

```

if pir_state == GPIO.HIGH:
    GPIO.output(LED_PIN, GPIO.HIGH)
else:
    GPIO.output(LED_PIN, GPIO.LOW)
if last_pir_state == GPIO.LOW and pir_state == GPIO.HIGH:
    movement_timer = time.time()
if last_pir_state == GPIO.HIGH and pir_state == GPIO.HIGH:
    if time.time() - movement_timer > MOV_DETECT_THRESHOLD:
        if time.time() - last_time_photo_taken >
MIN_DURATION_BETWEEN_2_PHOTOS:
            print("Take photo and send it by email")
            photo_file_name = take_photo(camera)
            update_photo_log_file(photo_file_name)
            send_email_with_photo(yag, photo_file_name)
            last_time_photo_taken = time.time()
last_pir_state = pir_state
except KeyboardInterrupt:
    GPIO.cleanup()

```

## 5 Código Configuração do serviço da Página Web

### 02\_flask\_web\_server.py

```

from flask import Flask, render_template, redirect, url_for, flash
import os
import glob
import secrets

CAMERA_FOLDER_PATH = "/home/pi/camera"
LOG_FILE_NAME = CAMERA_FOLDER_PATH + "/log/photo_logs.txt"
photo_counter = 0

app = Flask(__name__, static_url_path=CAMERA_FOLDER_PATH,
static_folder=CAMERA_FOLDER_PATH)

# Set the secret key to a sufficiently random value
app.secret_key = secrets.token_hex(16)

def get_latest_photo_info():
    message = ""
    line_counter = 0
    last_photo_file_name = ""
    if os.path.exists(LOG_FILE_NAME):
        with open(LOG_FILE_NAME, "r") as f:
            for line in f:
                line_counter += 1
                last_photo_file_name = line.strip()
    global photo_counter
    difference = line_counter - photo_counter
    message = f"{difference} photos were taken since you last
checked."
    photo_counter = line_counter
else:

```

```

        message = "No photos found."
    return {
        "message": message,
        "latest_photo": last_photo_file_name
    }

@app.route("/")
def index():
    return render_template('index.html')

@app.route("/check-movement")
def check_movement():
    photo_info = get_latest_photo_info()
    return render_template('secure_house.html',
    message=photo_info["message"], latest_photo=photo_info["latest_photo"])

@app.route('/clean-camera-directory')
def clean_camera_directory():
    camera_directory = CAMERA_FOLDER_PATH
    jpg_files = sorted(glob.glob(os.path.join(camera_directory,
    '*.jpg')), key=os.path.getctime)

    if len(jpg_files) > 1:
        latest_file = jpg_files[-1]
        files_to_delete = jpg_files[:-1]

        for file_path in files_to_delete:
            os.remove(file_path)

        flash(f"Deleted {len(files_to_delete)} files, kept the latest
file: {os.path.basename(latest_file)}")
    elif len(jpg_files) == 1:
        latest_file = jpg_files[0]
        flash(f"Only one file found, kept the file:
{os.path.basename(latest_file)}")
    else:
        flash("No files to delete.")

    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```

## 6 Código no Raspberry Pi no diretório *templates/index.html*

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

```

```
<title>Welcome to Home Security System</title>
<!-- Bootstrap CSS -->
<link
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.
min.css" rel="stylesheet">
<style>
  body {
    font-family: Arial, sans-serif;
    background-color: #343a40; /* Dark background */
    color: #f8f9fa; /* Light text color */
    text-align: center;
    padding: 20px;
  }
  .security-icon {
    font-size: 50px;
    margin-bottom: 20px;
    color: #ffc107; /* Gold color */
  }
  .btn-custom {
    width: 100%;
    margin: 10px 0;
    background-color: #495057; /* Darker button background */
    border: none;
    color: #f8f9fa; /* Light text color */
    border-radius: 10px;
  }
  .btn-custom:hover {
    background-color: #6c757d; /* Slightly lighter on hover */
  }
  .alert {
    text-align: center;
    margin-bottom: 20px;
  }
</style>
</head>
<body>
  <div class="container">
    <div class="security-icon">
      <i class="fas fa-shield-alt"></i>
    </div>
    <h1 class="my-4">Welcome to Home Security System</h1>
    <!-- Flash messages -->
```

```

{%- with messages = get_flashed_messages() %}

{%- if messages %}

{%- for message in messages %}

<div class="alert alert-success">
    {{ message }}
</div>

{%- endfor %}

{%- endif %}

{%- endwith %}

<div class="card mx-auto" style="max-width: 400px;">
    <div class="card-body">
        <button class="btn btn-custom"
            onclick="location.href='/check-movement'">Check for Photos</button>
        <button class="btn btn-custom"
            onclick="location.href='/clean-camera-directory'">Clean
            Resources</button>
    </div>
</div>
<!-- Bootstrap JS and dependencies -->
<script
src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.
min.js"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.mi
n.js"></script>
<!-- Font Awesome for security icon -->
<script src="https://kit.fontawesome.com/a076d05399.js"></script>
</body>
</html>

```

## 7 Código no Raspberry Pi no diretório *templates/secure\_home.html*

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">

```

```

<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<title>Home Security System</title>
<!-- Bootstrap CSS -->
<link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.
min.css" rel="stylesheet">
<style>
body {
    font-family: Arial, sans-serif;
    background-color: #343a40; /* Dark background */
    color: #f8f9fa; /* Light text color */
    text-align: center;
    padding: 20px;
}
.card {
    background-color: #495057; /* Darker card background */
    color: #f8f9fa; /* Light text color */
    border: none;
    border-radius: 10px;
}
.card img {
    border-radius: 10px;
}
</style>
<script>
function refreshImage() {
    var img = document.getElementById("latest-photo");
    img.src = img.src.split('?')[0] + '?' + new
Date().getTime();
    updateTimestamp();
}

function updateTimestamp() {
    var img = document.getElementById("latest-photo");
    var timestamp = img.src.match(/img_(\d+\.\d+)\.jpg/)[1];
    var date = new Date(Number(timestamp) * 1000);
    var formattedDate = date.toLocaleDateString('pt-BR') + ' ' +
date.toLocaleTimeString('pt-BR');
    document.getElementById("photo-timestamp").textContent =
'Photo taken ' + formattedDate;
}

```

```
document.addEventListener('DOMContentLoaded', function() {
    refreshImage();
}) ;

setInterval(refreshImage, 60000);
</script>
</head>
<body>
<div class="container">
    <h1 class="my-4">Home Security System</h1>
    <p>{{ message }}</p>
    <!-- Display latest photo if available -->
    <div class="card mx-auto" style="max-width: 600px;">
        <div class="card-body">
            <h5 class="card-title">Latest Captured Photo</h5>
            <div class="photo-container">
                
            </div>
            <p id="photo-timestamp" class="mt-3"></p>
        </div>
    </div>
    <!-- Link to go back to index.html -->
    <a href="/">Go back to Home</a>
</div>
<!-- Bootstrap JS and dependencies -->
<script
src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.
min.js"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.mi
n.js"></script>
</body>
</html>
```