Sistemas Embutidos

Câmara na Campainha

Maria Sousa Carreira, up202408787
Ricardo Amorim, up202107843

Maio 2025

Conteúdo

1 Descrição dos Objetivos

Este projeto tem como principal objetivo desenvolver um sistema de vigilância inteligente, integrado numa campainha, destinado a ser instalado à entrada de uma casa ou apartamento. O sistema deverá ser capaz de detetar movimento ou toques na campainha e, em resposta, ativar uma gravação de vídeo (em *stop-motion*). Além disso, pretende-se que o utilizador receba notificações no seu telemóvel e possa aceder a um *stream* de vídeo em tempo real.

2 Análise de Requisitos

Em relação aos requisitos do nosso sistema, decidimos distinguir entre dois tipos principais:

- **Requisitos Funcionais -** definem as funcionalidades que o sistema deve oferecer, ou seja, representam o comportamento esperado do sistema em resposta a diferentes sinais e ações do utilizador ou ambiente.
- **Requisitos Não Funcionais -** especificam as características de desempenho, qualidade, usabilidade, fiabilidade, entre outras, relacionadas com a forma como o sistema executa as suas funções.

Requisitos Funcionais	**Requisitos Não Funcionais**
• O sistema deve detetar movimento e proceder ao início da gravação; • O sistema deve gravar em modo *stop-motion*, interrompendo a gravação após 5 minutos sem deteção de movimento; • O utilizador deve receber uma notificação no *smartphone* sempre que alguém tocar à campainha; • Deve ser possível aceder a um *stream* de vídeo a partir do *smartphone* e aos vídeos gravados e apagá-los; • Deve haver presença de um LED vermelho indicativo de gravação ativa.	• A deteção de presença na porta deve ocorrer em menos de 3 segundos e a uma distância mínima de 50 cm; • O início da gravação após deteção ou toque na campainha deve ser inferior a 3 segundos; • O acesso a um *stream* de vídeo a partir do *smartphone* deve ocorrer com atraso inferior a 10 segundos; • O sistema deverá operar com baixo consumo energético; • A aplicação móvel deve ser de fácil utilização, compatível com dispositivos *Android*.

3 Modelo de Atores

O modelo de atores do sistema, ilustrado na Fig.1, é composto por quatro elementos principais.

O Ator Campainha é responsável por monitorizar a área em frente à porta, detetando movimento ou a ativação da campainha. Sempre que ocorre um evento, este ator comunica com o Ator Intermediário de Sinais, que, por sua vez, recebe os sinais dos sensores e comunica ao Ator Controlador da Câmara a decisão de ativar a gravação e a luz indicadora. Este último executa a gravação em *stop-motion* e notifica sempre o Ator Aplicação Móvel, o qual pode comunicar, a qualquer altura, a intenção de gravação ou *streaming*.

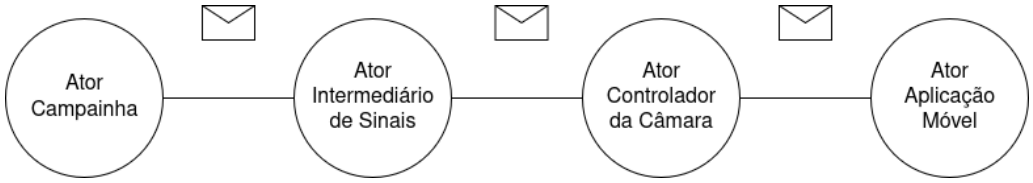


Figura 1: Modelo de Atores.

4 Especificação

Na Fig.2, é possível estabelecer uma correspondência entre os atores da Fig.1 e os componentes de hardware apresentados na Sec.5:

- **Ator Campainha:** irá incluir o sensor de movimento e o botão;
- **Ator Intermediário de Sinais:** refere-se ao Arduino;
- **Ator Controlador da Câmara:** refere-se ao Raspberry Pi, que irá conectar-se à *Camera* e ao LED;
- **Ator Aplicação Móvel:** irá incluir o Android.

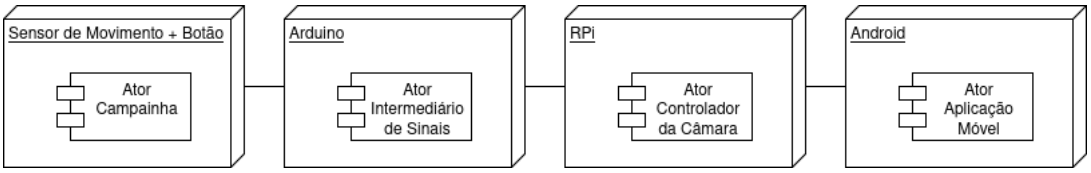


Figura 2: Diagrama de Deployment.

4.1 Arquitetura de Hardware

Na Fig.3, está representado um diagrama de blocos referente à arquitetura de hardware. A solução proposta baseia-se na integração de um Arduino com um Raspberry Pi, que trabalham de forma complementar.

O Arduino é responsável por receber sinais de entrada de dois componentes principais: um sensor de movimento, que deteta a presença de pessoas nas proximidades, e um botão

de campainha, acionado pelo visitante. Quando qualquer um destes eventos ocorre, o Arduino envia a informação para o Raspberry Pi através de uma ligação USB A-B.

O Raspberry Pi atua como o núcleo de controlo do sistema, sendo responsável por processar os sinais recebidos e ativar os dispositivos periféricos associados, como o LED vermelho, que fornece um alerta visual local, e a *Camera*, que capta imagens ou um vídeo do visitante.

Adicionalmente, o Raspberry Pi está ligado a uma rede Wi-Fi, através da qual comunica com um telemóvel Android, permitindo ao utilizador receber notificações e visualizar remotamente quem está à porta.

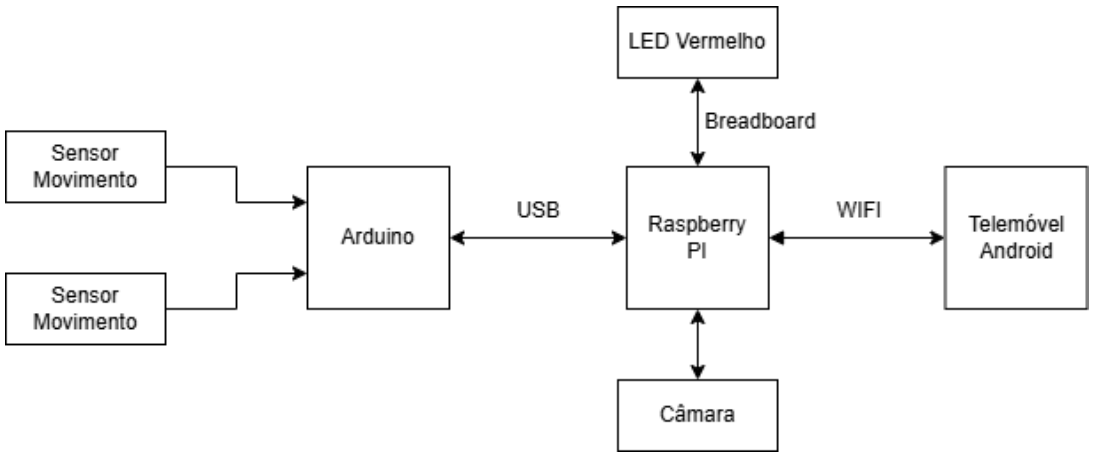


Figura 3: Diagrama de blocos da arquitetura de hardware.

4.2 Arquitetura de Software

A arquitetura de software do sistema organiza-se em torno dos quatro atores principais já referidos e está representada na Fig.4.

O Ator Campainha é responsável pela deteção de eventos físicos, como o toque na campainha ou a presença junto à porta, convertendo esses eventos em sinais digitais. Estes sinais são enviados ao Ator Intermediário de Sinais, que atua como ponte de comunicação, lendo-os e codificando-os numa mensagem que é enviada via Serial (USB) para o Ator Controlador da Câmara. Este é o núcleo lógico do sistema: lê as mensagens recebidas, gere o início e fim das gravações em modo *stop-motion*, ativa o LED vermelho indicador de gravação e responde a pedidos de *streaming*. Também comunica com o Ator Aplicação Móvel, que fornece ao utilizador uma interface intuitiva para receber notificações via HTTP, aceder aos vídeos gravados no Volume Docker, apagá-los e visualizar a transmissão ao vivo via protocolo RTSP (através do MediaMTX). Por fim, torna-se possível pedir para iniciar ou parar a transmissão (*streaming*), através do Ator Aplicação Móvel, via SSH.

4.2.1 Máquina de Estado

Para descrever o comportamento lógico do sistema proposto, recorreu-se à modelação através de uma *Finite State Machine* (FSM). Este modelo permite representar de forma clara e estruturada as diferentes fases operacionais do sistema, bem como as transições entre essas fases em função dos eventos detetados.

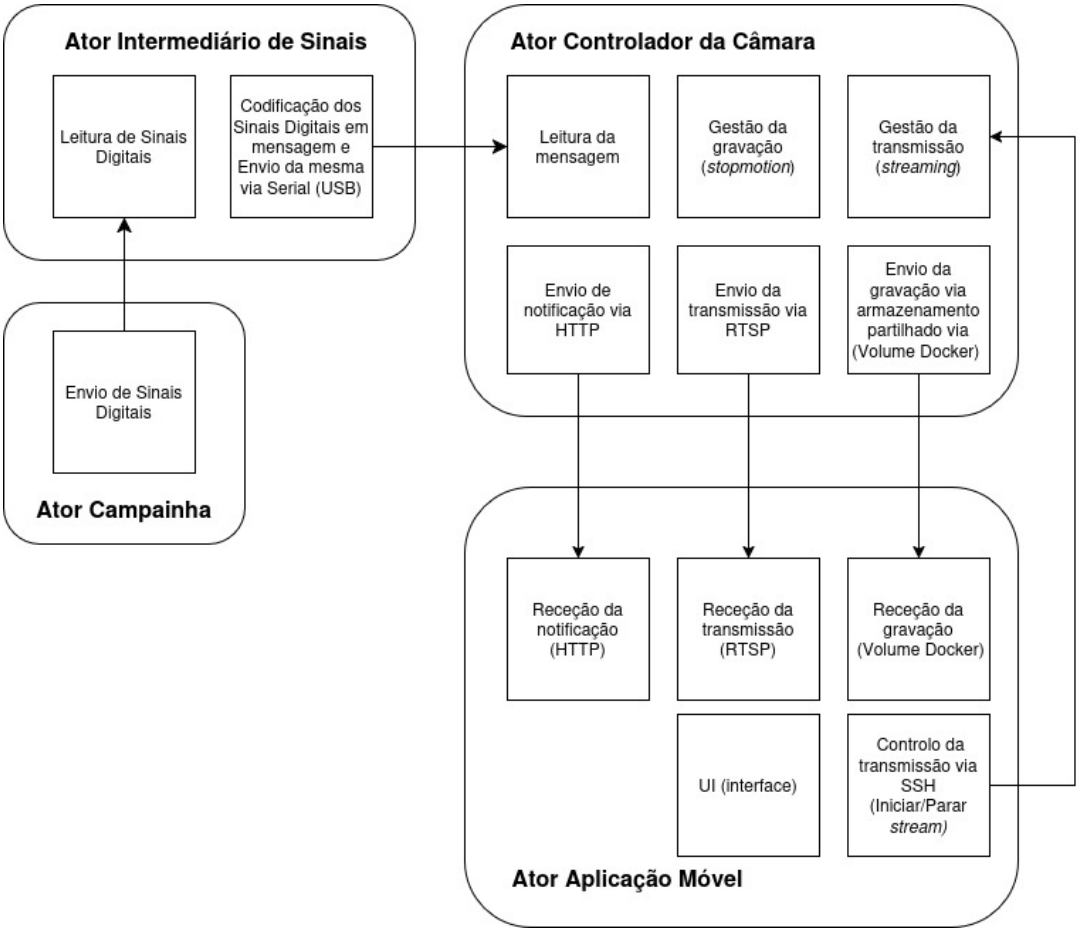


Figura 4: Diagrama de arquitetura de software.

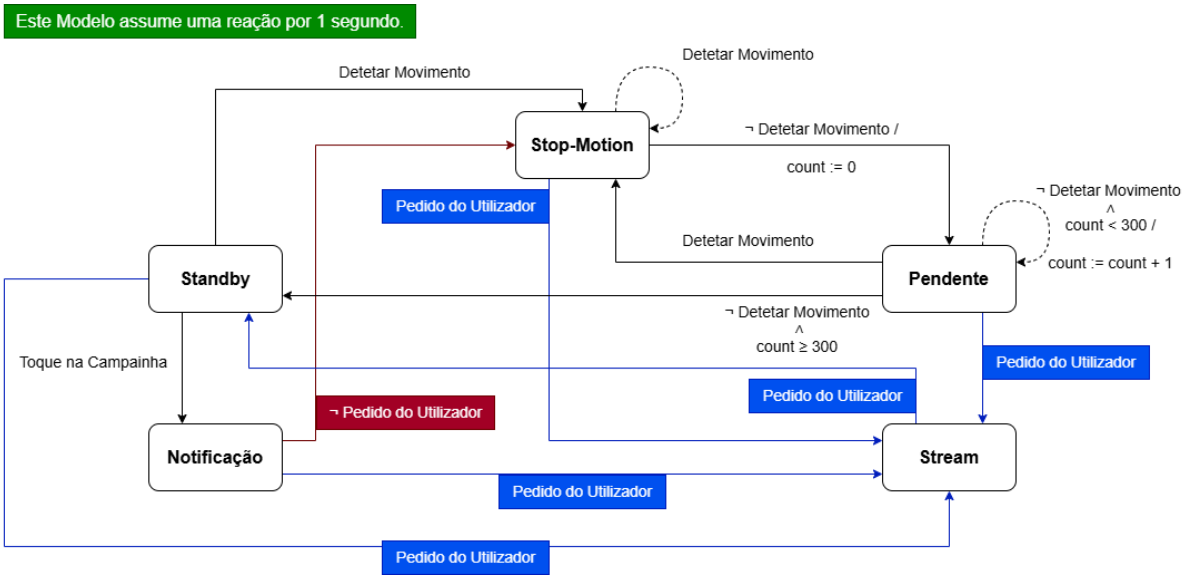


Figura 5: Máquina de Estado.

Na Fig.5, está ilustrada a FSM construída. As interações principais decorrem entre os estado `Stanby`, `Stop-Motion` e `Pendente`. Enquanto é detetado movimento, é executada uma gravação em *stop-motion*. Assim que não se deteta movimento, passamos para o estado `Pendente` e a variável `count` é inicializada. Durante os próximos 300 segundos

(5 minutos), caso se volte a detetar movimento, o sistema volta ao estado `Stop-Motion`, caso contrário (quando `count` atinge o valor 300), o sistema volta a `Standby`.

Por outro lado, as reações do sistema desencadeadas pelo utilizador, estão representadas a azul e vermelho. A partir de qualquer estado, é possível pedir uma *stream* de vídeo (`Stream`), que termina também através do pedido do utilizador, voltando a `Standby`. Para além disso, se se detetar um toque na campainha, o utilizador é notificado (`Notificação`). Neste momento, se o utilizador não efetuar o pedido para uma *stream* de vídeo, o sistema passa para o estado `Stop-Motion`.

5 Protótipos

5.1 Protótipo de Hardware

Na Fig.6 dos Anexos, são ilustrados os principais módulos e dispositivos que compõem o sistema desenvolvido. Entre os elementos mais relevantes destacam-se o Arduino Uno R3 (Fig. 6a) e o Raspberry Pi 3B+ (Fig. 6b), responsáveis, respetivamente, pela aquisição de sinais dos sensores e pelo controlo central e processamento do sistema.

O sensor PIR (Fig. 6f) permite detetar movimento, enquanto o botão físico (Fig. 6g) atua como campainha. Ambos estão ligados ao Arduino, que envia os eventos para o Raspberry Pi por ligação serial através de um cabo USB A-B (Fig. 6l). A NoIR Camera V2 (Fig. 6e) é utilizada para capturar imagens no modo *stopmotion*, e o LED vermelho (Fig. 6h) funciona como indicador de estado do sistema (ativo/inativo).

Para interligar os componentes, foram utilizados cabos *jumper* (Fig. 6d), uma *breadboard* (Fig. 6n) e resistores de $330\,\Omega$ e $10\,k\Omega$ (Fig. 6i) – o primeiro associado ao LED, e o segundo ao botão, para garantir um *pull-down* fiável.

O telemóvel Android (Fig. 6c) funciona como interface final com o utilizador, possibilitando a receção de notificações e o acesso remoto à stream de vídeo via a aplicação Android.

Em termos de alimentação, o sistema é suportado por um cabo micro-USB (Fig. 6j) ligado a um carregador de 33W (Fig. 6k). Apesar da capacidade nominal do carregador, o Raspberry Pi apenas consome a potência padrão (cerca de 5V/2.5A), uma vez que o modo de carregamento rápido (*fast charging*) não é suportado por este dispositivo. Já o Arduino é alimentado de forma estável e independente através de um adaptador de 5V e 2A (Fig. 6m).

5.2 Protótipo de Software

O protótipo de software foi desenvolvido de forma modular e distribuída, integrando diferentes componentes a correr em dispositivos distintos: o Arduino Uno R3 e o Raspberry Pi 3B+.

No desenvolvimento foram utilizadas várias linguagens e ferramentas:

- **Arduino IDE v2.3.6** utilizada para programação do microcontrolador com a linguagem padrão do Arduino (baseada em `C++`).
- **Bash v5.2.15** — linguagem de *scripting* usada nos *scripts* de sistema do Raspberry Pi.

- **Python v3.11.2** — usada para controlo da câmara e captura de imagens, em conjunto com a biblioteca `picamera2 0.3.10`.
- **ffmpeg v5.1.6-0** — responsável pela geração dos vídeos a partir das imagens captadas.
- **MediaMTX v1.12.0** — servidor multimédia leve utilizado para disponibilizar o *stream* de vídeo via protocolo RTSP.
- **Home Assistant v2025.4.3** — executado em Docker no Raspberry Pi, para gestão da interface, automações e integração geral.
- **pinctrl (sem versão)** — ferramenta de linha de comandos utilizada para controlar o estado dos pinos GPIO do Raspberry Pi (ativar/desativar o LED indicador).
- **curl v7.88.1** — utilizada para enviar notificações ao Home Assistant através da sua REST API.
- **jq v1.6** — usada para análise da resposta JSON da API do MediaMTX.
- **systemd v252.31** — gestor de serviços utilizado para garantir o arranque automático dos scripts e serviços como o sistema de escuta e o MediaMTX.
- **OpenSSH v9.2** — utilizado para execução remota de comandos no Raspberry Pi a partir do Home Assistant (via SSH).

5.2.1 Firmware no Arduino

O Arduino Uno R3 executa um firmware simples, desenvolvido na IDE do Arduino, cuja função é monitorizar continuamente o estado do botão físico e do sensor de movimento (PIR), ambos ligados a pinos digitais. Os sinais elétricos recebidos são interpretados como eventos digitais (`HIGH` ou `LOW`) e, sempre que detetado um evento relevante, como movimento ou toque, o Arduino envia uma mensagem descritiva pela porta Serial (USB). Estas mensagens, como `"motion"` ou `"touch"`, são transmitidas em texto simples e são interpretadas pelo Raspberry Pi como gatilhos para outras ações. O programa no Arduino corre continuamente no ciclo `loop()`, garantindo resposta em tempo real à ocorrência de eventos físicos.

5.2.2 Sistema de escuta no Raspberry Pi

No Raspberry Pi 3B+, corre um *script* central desenvolvido em `bash`, configurado como serviço do sistema através do `systemd`, garantindo o seu arranque automático após reinício ou inicialização do sistema. Este *script* atua como ouvinte permanente da porta Serial `/dev/ttyACM0`, onde aguarda por mensagens enviadas pelo Arduino. Quando um evento é detetado, o *script* inicia um processo de captura em segundo plano, invocando um *script* `Python` que grava imagens em modo *stop-motion*.

Para cada sessão de captura, é criado automaticamente um diretório com *timestamp*, onde são armazenadas as imagens temporárias. No final da sessão, as imagens são convertidas num vídeo através do `ffmpeg`, e o ficheiro final é movido para a pasta `/home/group7/ha_media`, que está montada no Home Assistant como `/media`.

O *script* garante que a gravação para automaticamente após um período de inatividade (definido por *timeout*), ou sempre que a transmissão em direto (*stream*) é ativada,

evitando conflitos no acesso à câmara. Esta verificação é feita com auxílio da ferramenta `jq`, que permite extrair, a partir da API do MediaMTX, o número de clientes ativos da *stream*. Caso sejam detetados clientes ligados, o sistema interpreta que a *stream* foi iniciada e termina a gravação em curso.

Além disso, o *script* ativa e desativa um LED físico através do controlo do pino GPIO 17, usando a ferramenta `pinctrl`, fornecendo *feedback* visual do estado do sistema sempre que a câmara esteja a gravar ou a transmitir. No caso de deteção de toque na campainha, é enviada uma notificação ao Home Assistant via REST API, utilizando a ferramenta `curl` com autenticação através de um *token*.

5.2.3 Captura de Imagens (Python)

O processo de captura de imagens é gerido por um *script* `Python` executado em segundo plano, invocado sempre que o Raspberry Pi deteta um evento vindo do Arduino. Este *script* inicializa e configura a câmara NoIR utilizando a biblioteca `picamera2`, capturando imagens a intervalos regulares enquanto o processo estiver ativo. A paragem do processo é feita de forma limpa através da receção de um sinal `SIGTERM`, garantindo o encerramento adequado da câmara. Após a paragem, os *frames* gerados são utilizados para criar um vídeo em modo *stop-motion* com recurso à ferramenta `ffmpeg`.

5.2.4 Transmissão de Vídeo em Tempo Real

Em paralelo ao sistema de captura, foi também implementada uma solução de transmissão de vídeo em tempo real. Esta funcionalidade é assegurada pelo servidor MediaMTX, que corre como serviço independente no Raspberry Pi. Este servidor é configurado para aceder diretamente à câmara, fornecendo um *stream* no formato RTSP acessível a partir de outros dispositivos na rede local. A ativação ou desativação do serviço é controlada remotamente a partir do Home Assistant, através de comandos SSH, acionados por um botão na interface gráfica.

Importa referir que o sistema garante que o *stop-motion* e a transmissão em direto não ocorram em simultâneo, evitando conflitos de acesso à câmara.

5.2.5 Integração com o Home Assistant

A interface com o utilizador é feita através do Home Assistant, que corre num contentor Docker no próprio Raspberry Pi. Este ambiente fornece uma interface *web* acessível por qualquer dispositivo na rede local, como, no neste caso, um telemóvel Android. Através desta interface, o utilizador pode iniciar ou parar a transmissão de vídeo em direto com um simples botão.

Sempre que é detetado um toque no botão físico (campainha), o sistema envia automaticamente uma notificação para o Home Assistant via REST API, alertando o utilizador do evento. O utilizador tem também acesso direto à galeria de vídeos gerados, disponibilizados através da pasta `/media`, montada automaticamente a partir do diretório local `/home/group7/ha_media`.

6 Integração

6.1 Montagem das Componentes

A montagem das componentes utilizadas pode ser visualizada no seguinte vídeo:

https://youtu.be/guw2_UXaTIw.

No mesmo vídeo, encontra-se a **demonstração** do sistema implementado. Por fim, na Sec.9.2 dos Anexos, incluíram-se alguns tutoriais do YouTube que serviram de base à montagem.

6.2 Instalação do Software

Primeiramente, instalou-se o Arduino IDE v2.3.6 de acordo com a documentação disponível em `https://docs.arduino.cc/software/ide/#ide-v2`.

De seguida, instalou-se o sistema operativo do Raspberry Pi (`https://www.raspberrypi.com/software/`) através do Raspberry Pi Imager. Este programa permite selecionar o cartão SD que irá ser utilizado no Raspberry Pi e onde o SO será instalado, e também configurar o Wi-Fi e SSH para as comunicações. A documentação para estas configurações está disponível em `https://www.raspberrypi.com/documentation/computers/configuration.html`.

A ferramenta MediaMTX foi instalada de acordo com as etapas em `https://github.com/bluenviron/mediamtx?tab=readme-ov-file#installation`. As restantes ferramentas, como `ffmpeg`, instalaram-se através do `apt` do Linux.

O Home Assistant foi instalado no Raspberry Pi utilizando o Docker, com base no ficheiro `docker-compose.yml` submetido, que define a configuração necessária para executar o *container* do Home Assistant (`https://www.home-assistant.io/installation/raspberrypi-other/?utm_source=chatgpt.com`). Posteriormente, a aplicação oficial do Home Assistant foi instalada num dispositivo Android. Para que a aplicação móvel consiga comunicar com o Home Assistant, é necessário configurar o endereço IP do dispositivo onde o serviço está a correr, neste caso, o Raspberry Pi, e introduzi-lo na aplicação.

6.3 Preparação dos Testes

Para garantir uma avaliação eficaz, foram definidos e organizados vários cenários de teste com base nos requisitos funcionais e não funcionais previamente especificados. Antes da execução dos testes, assegurou-se que todos os módulos de hardware estavam corretamente montados e ligados, nomeadamente o sensor PIR, o botão físico, o LED, a câmara NoIR e as conexões entre o Arduino e o Raspberry Pi via USB.

Do lado do software, foi verificada a inicialização correta dos serviços essenciais: o *script* de escuta no Raspberry Pi, o serviço de *streaming* MediaMTX, o sistema de notificações, e o Home Assistant no ambiente Docker. Confirmou-se também o acesso ao Home Assistant através dos dispositivos móveis conectados à mesma rede local.

7 Avaliação

A avaliação do sistema foi conduzida com o objetivo de verificar o cumprimento dos requisitos funcionais definidos para o protótipo. Para isso, foram realizados testes sis-

temáticos que cobriram tanto os módulos isoladamente quanto o sistema como um todo, em condições reais de operação.

Antes de validar os requisitos definidos, foram realizados diversos testes funcionais e não funcionais que cobriram os principais cenários de uso do sistema. Entre os testes efetuados destacam-se:

- Testes de deteção de movimento com o sensor PIR a várias distâncias.
- Testes de resposta do sistema ao toque no botão (campainha).
- Testes de latência entre o evento e o início da gravação.
- Testes de gravação automática e verificação do tempo de paragem por inatividade.
- Testes de envio de notificações via REST API para o Home Assistant.
- Testes da interface do utilizador no Home Assistant (botão de ativação/desativação da *stream*).
- Testes de verificação dos vídeos gerados e acesso aos mesmos através da interface do Home Assistant.
- Testes da transmissão de vídeo via RTSP (verificando estabilidade, atraso e qualidade).
- Testes de verificação do estado do LED indicador (gravação/transmissão).
- Testes de compatibilidade da aplicação com diferentes dispositivos móveis.

7.1 Validação dos Requisitos Funcionais

Em relação aos requisitos funcionais, confirmou-se que o sistema deteta movimento e inicia a gravação automaticamente. A câmara entra em modo *stop-motion* sempre que é detetado um evento (movimento ou toque), e interrompe a gravação após 5 minutos de inatividade, conforme especificado. Verificou-se ainda que o utilizador recebe, sem falhas uma notificação no *smartphone* sempre que alguém toca na campainha. Esta notificação é enviada em tempo real através da REST API do Home Assistant e, na maioria dos casos, é recebida quase instantaneamente, com apenas ligeiros atrasos em situações pontuais.

Adicionalmente, a funcionalidade de transmissão de vídeo foi testada com sucesso, sendo possível iniciar e parar o *stream* a partir da interface do Home Assistant e aceder aos vídeos armazenados. Por fim, confirmou-se também a presença do LED vermelho ligado ao GPIO do Raspberry Pi, que acende sempre que a câmara está a gravar ou a transmitir, oferecendo um *feedback* visual do estado do sistema.

7.2 Validação dos Requisitos Não Funcionais

No que diz respeito aos requisitos não funcionais, também se verificou o seu cumprimento. O sensor PIR conseguiu detetar movimento a uma distância mínima de 50 cm em menos de 3 segundos, validando o requisito relacionado com o tempo de deteção. O tempo entre a receção de um evento (por exemplo, toque ou movimento) e o início da gravação foi inferior a 3 segundos, garantindo uma resposta rápida do sistema. O atraso na transmissão do

vídeo em tempo real, fornecido pelo MediaMTX, manteve-se consistente abaixo dos 2 segundos durante os testes realizados, confirmando a boa performance e qualidade da solução de *streaming*.

A eficiência energética foi garantida e validade não apenas em contexto de testes, mas também pela lógica de implementação: a câmara apenas permanece ligada durante a gravação em modo *stop-motion* ou quando a transmissão está ativa, o que reduz significativamente o consumo de energia e evita o uso desnecessário de recursos do Raspberry Pi.

Por fim, a aplicação foi testada num *smartphone* Android, revelando-se funcional, intuitiva e responsiva, cumprindo o requisito de facilidade de utilização e compatibilidade. Adicionalmente, apesar de não ter sido definido nos requisitos, a aplicação foi também testada num dispositivo iOS (iPhone), onde se verificou igualmente um funcionamento correto. Desta forma, foi possível validar a compatibilidade da interface do Home Assistant tanto em dispositivos Android como iOS.

8 Conclusão

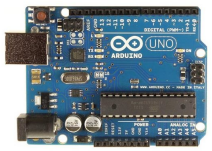
O desenvolvimento deste sistema de campainha inteligente com câmara permitiu aplicar conceitos fundamentais de sistemas embutidos, comunicação entre dispositivos e integração com plataformas domóticas. O projeto combinou hardware e software de forma eficaz, utilizando componentes acessíveis como o Arduino Uno R3 e o Raspberry Pi 3B+, aliados a ferramentas modernas como o Home Assistant e o MediaMTX.

O sistema proposto demonstrou ser funcional e fiável, cumprindo todos os requisitos definidos, tanto funcionais como não funcionais. Foi possível detetar movimento ou toque na campainha, iniciar automaticamente uma gravação em modo *stop-motion*, notificar o utilizador em tempo real e disponibilizar uma transmissão de vídeo em direto acessível a partir de um *smartphone*.

Além disso, o sistema foi concebido com preocupações energéticas, garantindo que a câmara só permanece ativa quando necessário, o que reduz o consumo e melhora a eficiência global.

9 Anexos

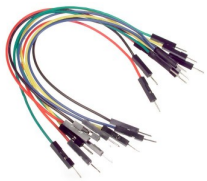
9.1 Componentes de Hardware Utilizados



(a) Arduino Uno R3.

(b) Raspberry Pi 3b+.

(c) Telemóvel Android.



(d) Cabos *Jumper*.

(e) NoIR Camera V2.

(f) Sensor de Movimento.

(g) Botão.

(h) LED Vermelho.

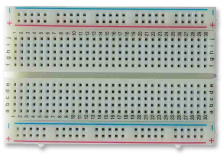
(i) Resistores de 330 Ω e 10 mil Ω.

(j) Cabo Micro USB.

(k) Carregador Xiaomi (33W).

(l) Cabo USB A-B.

(m) Adaptador (5V 2A).



(n) *BreadBoard* (Placa de ensaio).

Figura 6: Componentes de hardware utilizados.

9.2 Referências para a Montagem de Algumas Componentes

- **Montagem do Sensor PIR:** https://www.youtube.com/watch?v=3gj_68ywod4.
- **Montagem do Botão:** https://www.youtube.com/watch?v=yBgMJssXqHY&t=870s.