Departamento de Ciências dos Computadores
Mestrado em Engenharia de Redes e S.I.

Câmara na campainha

Sistemas Embutidos

Docente:
Luis lopes

Membros do Grupo:
Rui Santos
Ricardo Sá

Ano académico 2024/2025

Contents

1 Descrição **2**

2 Requisitos **2**

3 Especificação **3**

3.1 Nomenclatura do sistema 3

3.2 Modelo de atores 3

3.3 Diagrama de blocos 4

3.3.1 Arquitetura de Hardware e atores 4

3.3.2 Arquitetura de Software 5

3.3.3 Diagrama de Instalação 6

3.4 Modelação de Software 6

3.4.1 Diagramas de Fluxo 7

3.4.2 Máquinas de estado 9

4 Protótipo **11**

4.1 Hardware 11

4.2 Software 12

5 Integração **13**

6 Avaliação **13**

7 Github **14**

1 Descrição

Neste relatório apresentamos a especificação completa do projeto *Câmara na campainha*, realizado no âmbito da unidade curricular Sistemas Embutidos.

Com este projeto propomos a criação de um sistema de vigilância integrado com uma campainha que oferece duas modalidades de visualização de vídeo: transmissão ao vivo (live-video) e sequência de imagens em stop-motion. A funcionalidade de live-video é disponibilizada a pedido do utilizador através de uma aplicação móvel. Em contraste, a gravação em stop-motion é ativada automaticamente pela deteção de movimento na proximidade da campainha, capturando uma série de fotografias sequenciais com o objetivo de otimizar o uso do armazenamento. Adicionalmente, ao tocar na campainha uma notificação é enviada para a aplicação móvel do utilizador, alertando-o para a presença de alguém na porta. Através da mesma aplicação, o utilizador poderá visualizar tanto o live-video quanto as sequências de imagens gravadas em stop-motion, tendo ainda a opção de remover estas últimas. Para fornecer uma indicação visual clara do estado de gravação do sistema, uma luz vermelha vais ser incorporada no mesmo.

Para realizar este projeto fizemos uso de: Um Arduino Uno R4 Wifi, um Raspberry pi 4 , um smarthphone e outros componentes eletrónicos (sensores,botões,Led, etc.).

São apresentados neste documento os requisitos e a nomenclatura do sistema "Câmara na Campainha", seguidos da descrição da sua arquitetura de hardware e software, suportada por diagramas de blocos e pelo modelo de atores. A modelação do comportamento do sistema é detalhada através de diagramas de fluxo e máquinas de estado. O protótipo desenvolvido é então descrito, tanto no que respeita ao hardware como ao software. O documento termina com a indicação do repositório Github onde o código fonte do projeto está disponível.

2 Requisitos

Os requisitos impostos para o sistema final são os seguintes:

- Deteção de presença na porta deve ser $<$ do que 3 seg
- Início de gravação desde deteção ou toque na campainha deve ser $<$ do que 3 seg
- Pedido de stream de video do smartphone deve ter um atraso $<$ do que 10 seg
- Presença deve ser detetada no mínimo a 50 cm

Todos estes requisitos foram cumpridos no prótotipo final.

3 Especificação

Para demonstrar a especificação do nosso projeto desenvolvemos quatro tipos de diagramas: 1) Diagrama do Modelo de atores, 2) diagrama de blocos, 3) diagrama de fluxo e 4) diagrama de máquinas de estado.

3.1 Nomenclatura do sistema

Por questões de simplificação , decidimos adotar a seguinte nomenclatura para os componentes principais do sistema:

- O arduino R4 Wifi foi designado como **Arduino_W**.
- O conjunto Raspberry mais a câmara a é designado por **RPi_Cam**.
- O smartphone é designado por **Android**.

3.2 Modelo de atores

O sistema pode ser descrito através do modelo de atores da Fig. 1. O **Agente Video + Storage** é o ator central do sistema, que é responsável por orquestrar as funcionalidades de deteção (movimento ou toque na campainha) e por gerir o acesso ao live-video e as imagem capturadas durante o "stop-motion". O **Visitante** interage com o sistema através do **Agente Campainha** e do **Agente Movimento** (cada um representado o respetivo evento associado). Ambos estes agentes comunicam o seu respetivo evento com o **Agente Video + Storage**. Por sua vez, este comunica com o **Utilizador** para enviar notificações, live-video e imagens obtidos no "stop-motion"

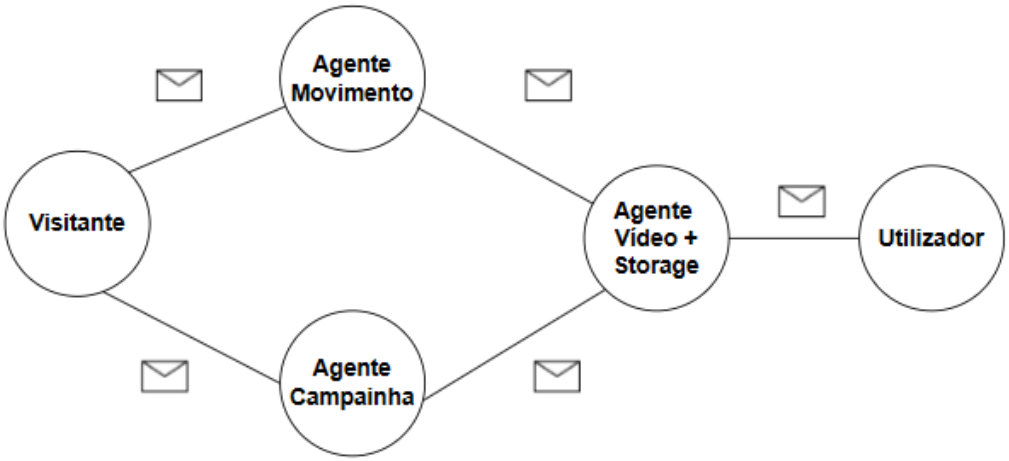


Figure 1: Diagrama do Modelo de atores

3.3 Diagrama de blocos

Os diagrama de blocos apresentados na Fig.2 e Fig.3 são usados para descrever o sistema em termos de unidades lógicas que o constituem. Se consideramos que estas unidades lógicas são componentes de hardware, então o diagrama descreve a arquitetura de hardware. Por outro lado, se as unidades lógicas corresponderem a módulos de software, então dizemos que o diagrama descreve a arquitetura de software.

3.3.1 Arquitetura de Hardware e atores

Na fase da especificação é possível estabelecer uma correspondência entre os atores da Fig. 1 e os componentes de hardware apresentados na Fig. 2. Assim o **RPI_Cam** vai desempenhar o papel de **Agente Video + Storage**. O **Arduino_W** vai desempenhar o **Agente Movimento** e **Agente Campainha**. E por fim, o **utilizador** vai ser representado por **Android**.

O diagrama de arquitetura de hardware (Fig. 2) identifica três componentes centrais:

1. **Arduino_W** (Arduino Uno R4 Wifi)
2. **RPI_Cam** (Raspberry-Pi + Câmera)
3. **Android**

A comunicação entre **Arduino_W** e **RPI_Cam**, bem como a comunicação entre **RPI_Cam** e **Android**, é bidirecional e feita através de um canal Wi-Fi.

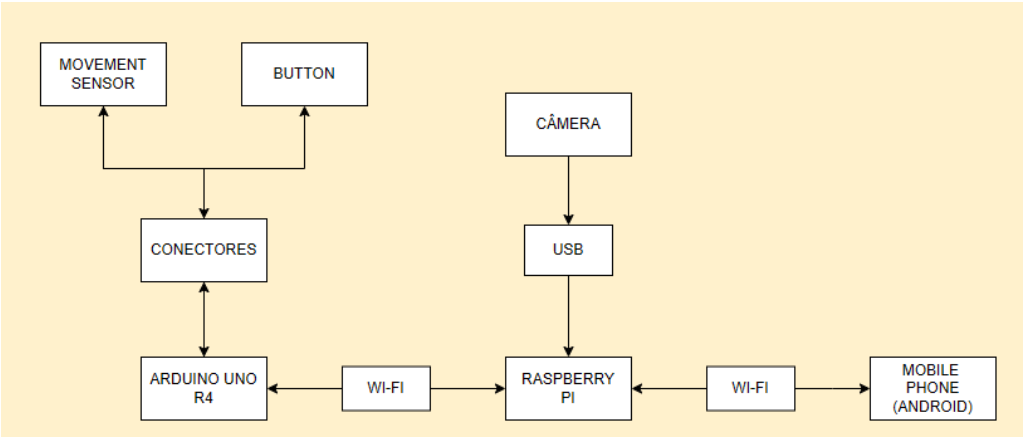


Figure 2: Diagrama de arquitetura de hardware

3.3.2 Arquitetura de Software

A arquitetura de software é composta por blocos que constituem os vários módulos de software a correr nos vários dispositivos. Tendo em conta a Fig. 3, podemos descrever a arquitetura de software com os seguintes pontos:

- **Android**: Esta camada contém a interface do utilizador (UI), um cliente HTTP para comunicar com o **RPI_Cam** e um módulo de gestão de notificações.
- **RPI_Cam**: O Raspberry Pi é o ponto central do sistema, que possui dois servidores HTTP REST: um para armazenamento de dados (Storage) e outro para streaming de vídeo (Vídeo). Também tem incluído um módulo de gestão de mensagens que lida com processamento/resposta de mensagens.
- **Arduino_W**: O Arduino é responsável pela lógica de deteção de movimento e pela lógica para tocar a campainha. Para enviar informações para o servidor REST no **RPI_Cam**.

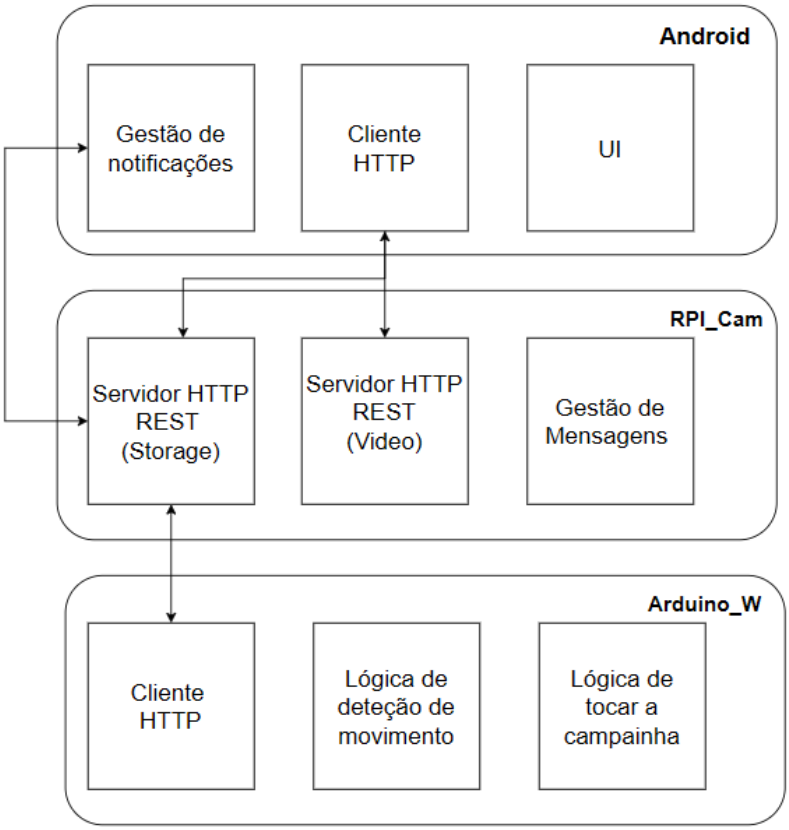


Figure 3: Diagrama de arquitetura de software

3.3.3 Diagrama de Instalação

O diagrama de instalação representa: a configuração, a arquitetura do sistema e a interligação entre os componentes de software e hardware.O diagrama de instalação está representado na Fig. 4.

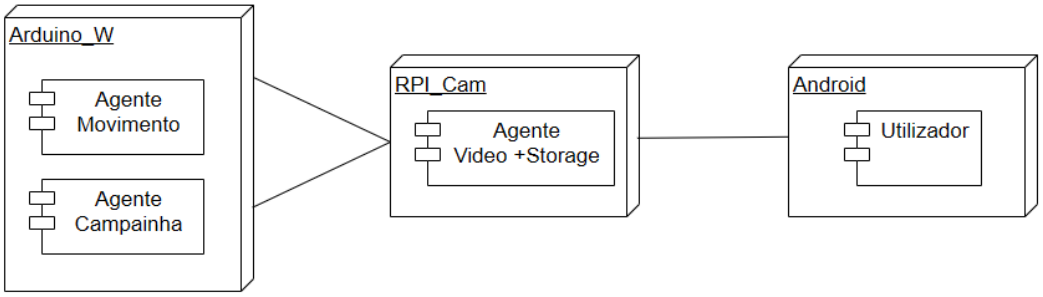


Figure 4: Diagrama de Deployment

3.4 Modelação de Software

O objetivo da modelação de software consiste em especificar o comportamento do sistema em termos das suas funcionalidades. Para isso, são utilizadas linguagens gráficas capazes não só de expressar os requisitos do sistema, como também de expressar as escolhas sobre o design do sistema. Modelamos este comportamento com diagramas de fluxo e máquinas de estado.

Os diagramas de fluxo têm como objetivo especificar a sequência de controlo de um programa ou um processo. Esta sequência identifica as principais funcionalidades do sistema e serve de guia para a implementação do software final. As máquinas de estado têm como objetivo traçar uma visão abstrata do sistema em termos de estados e transições despoletadas por eventos.

3.4.1 Diagramas de Fluxo

O comportamento do **Arduino_W** (Fig. 5) é modelado em dois diagramas de fluxo, um para o sensor de movimento e outro para o botão. Ambos os diagramas têm a sequência comum a qualquer sketch Arduino: primeiro o bloco "Setup"e depois o bloco "Loop". Dentro do bloco "Setup" é feita a inicialização dos canais de comunicação, utilizando as bibliotecas do sistema. Dentro do bloco "Loop" vai ser implementada a lógica de controlo e a gestão das mensagens.

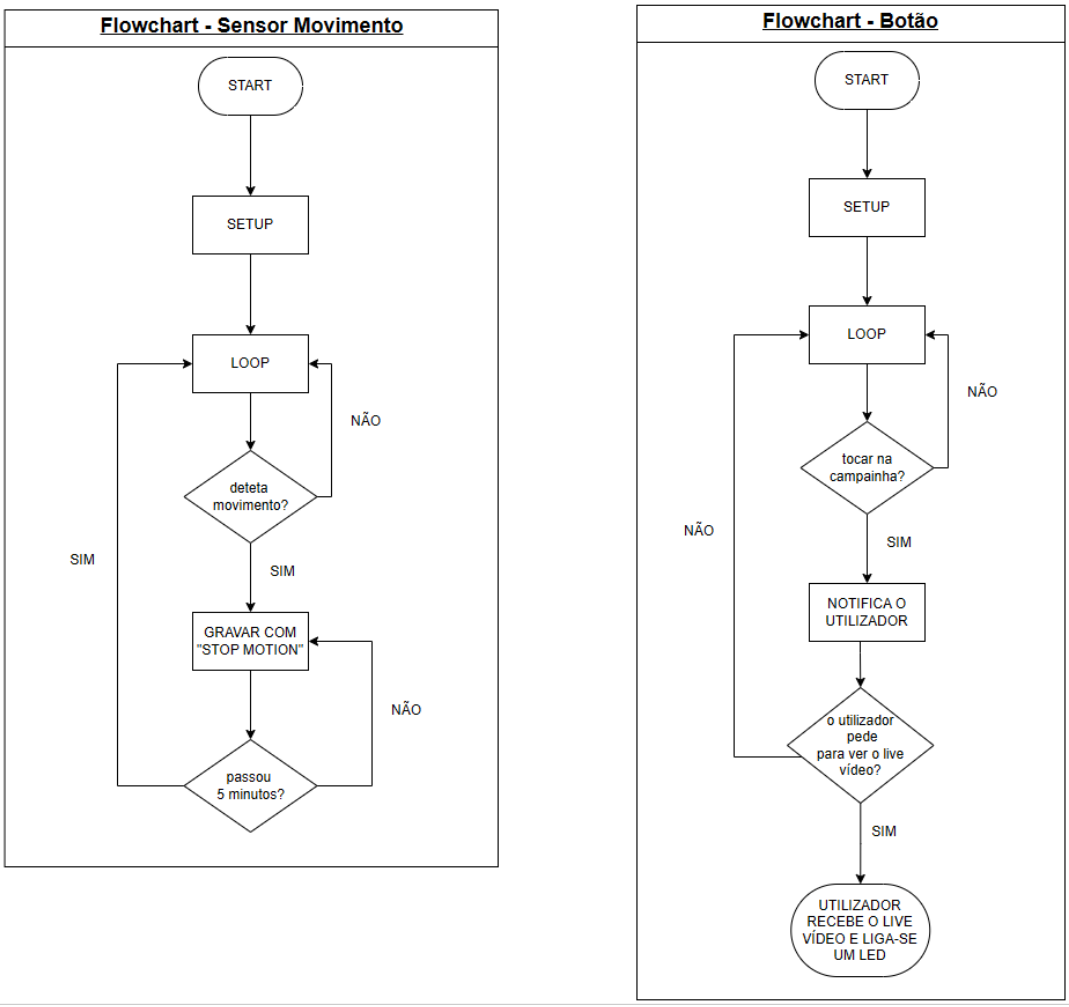


Figure 5: Flowchart de sensor movimento e botão (**Arduino_W**)

O diagrama de fuxo apresentado na Fig. 6 descreve a funcionalidade que é esperada do **RPI_Cam** no que consta à comunicação com **Android**. Neste diagrama pressupomos que **RPI_Cam** contém um servidor HTTP à escuta de pedidos oriundos do **Android** e que responde ao pedido do **RPI_Cam** de forma adequada, consoante o 'tipo' daquele. Caso o pedido não seja válido o servidor responde com `null`.

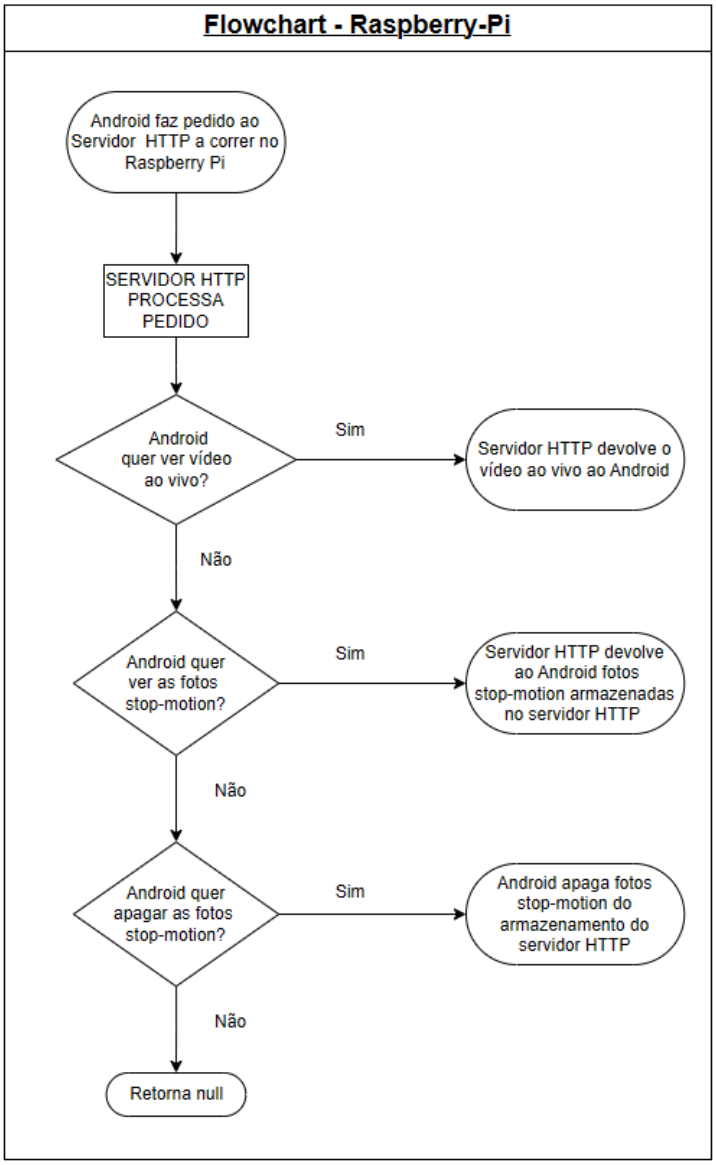


Figure 6: Flowchart de **RPI_Cam**

3.4.2 Máquinas de estado

Os diagramas de máquinas de estado modelam o comportamento do sistema em termos de eventos discretos. A especificação do sistema é feita em termos do estado atual e ocorrências de eventos input/output.

A máquina de estados que contém a visão mais geral do sistema é apresentada na Fig. 7. No pseudo-estado inicial, considera-se que os dispositivos já têm os canais de comunicação inicializados, e que as variáveis de estado "movement", "Passed5", "Ringed", "Live" e "AccessStorage" estão com o valor *False*. Neste caso, o sistema transita automaticamente para o estado <**Idle**>, que representa o estado *default* do sistema.

Existem duas transições principais que podem levar o sistema do estado <**Idle**> para outros estados ativos: a deteção de movimento ("movement" = True) leva ao estado de <**Gravar com 'STOP MOTION'**>, indicando que foi detetado movimento e que foi iniciado o processo de captura de imagens de forma sequencial (i.e. "STOP MOTION"). Por outro lado, se o botão for ativado ("Ringed" = True), o sistema transiciona para o estado <**Notifica Utilizador**>, alertando o utilizador por meio de uma notificação que é entregue a aplicação Android.

Adicionalmente, em qualquer estado do sistema, o utilizador tem a capacidade de interagir de duas maneiras principais: solicitar o vídeo em direto ("Live" = True), o que leva o sistema ao estado <**Utilizador vê vídeo em direto**> e aceder ao armazenamento ("AccessStorage" = True), que move o sistema para o estado <**Utilizador acede ao armazenamento**>.O sistema regressa ao estado <**Idle**> quando:

- A gravação de "STOP MOTION" termina após cinco minutos ("Passed5" = True)
- O utilizador termina o acesso ao armazenamento ("AccessStorage" = False)

Por fim, o estado <**Notifica Utilizador**> apenas tem duas transições possíveis, que representam a decisão do utilizador perante querer ou não aceder ao vídeo em direto.

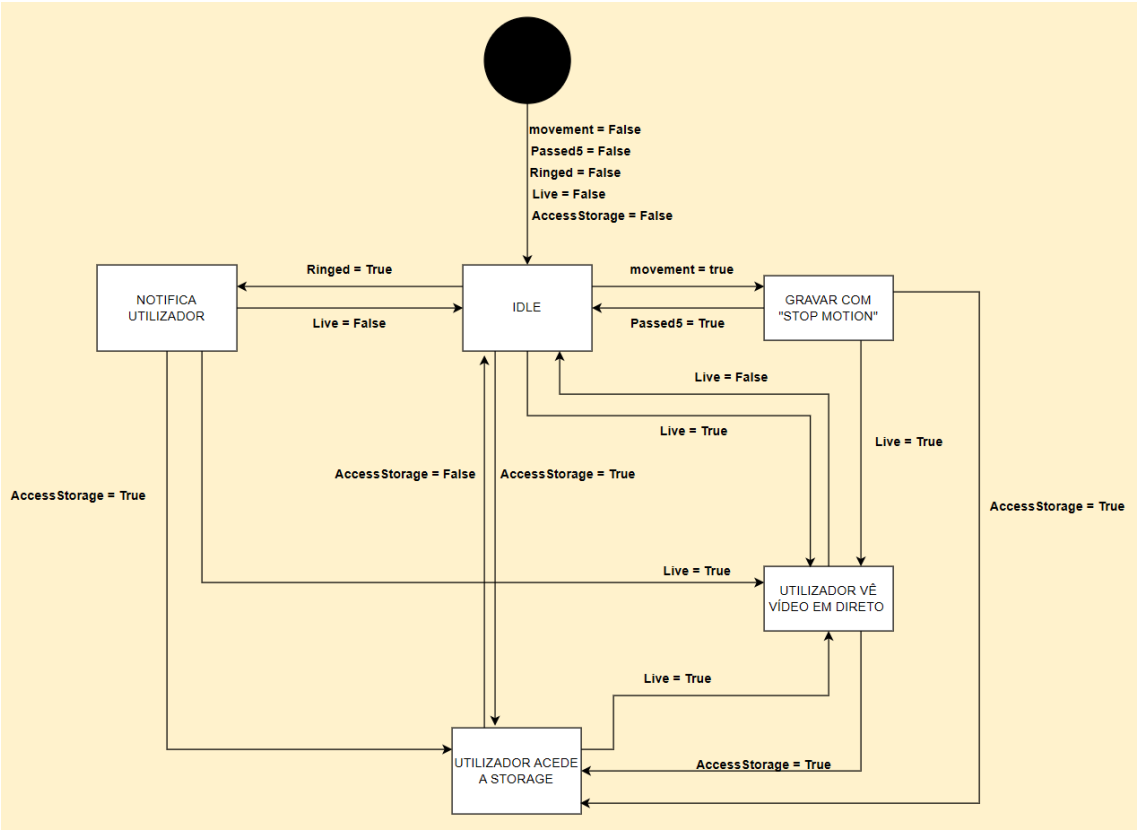


Figure 7: Flowchart de **Máquina de estados geral**

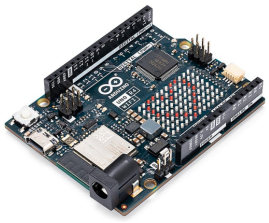
4 Protótipo

4.1 Hardware

A lista de *hardware* utilizada neste protótipo esta demonstrada na Fig. 8. Tendo em conta a Fig. 9 com a montagem do hardware, vemos que:

- O arduino R4 Wifi (**Arduino_W**) tem as ligações correspondentes ao sensor de movimento e ao botão. Em termos de alimentação, temos o **RPI_Cam** que fornece alimentação via USB-C.
- O conjunto **RPI_Cam** (é constituído por uma Raspberry pi conectada a uma Câmara, com 2 led's embutidos). Em termos de alimentação utilizamos uma fonte externa.
- Em termos de ligações *wireless*, **Arduino_W** comunica com o **RPi_Cam** via Wi-fi. Por sua vez, o **RPi_Cam** comunica também via Wi-Fi com o smartphone (**Android**).

(a) Raspberry-Pi 4



(b) Arduino Uno r4 Wifi

(c) Android

(d) Câmera

(e) Sensor de Movimento

(f) Botão

(g) Conectores

Figure 8: Componentes de hardware utilizados

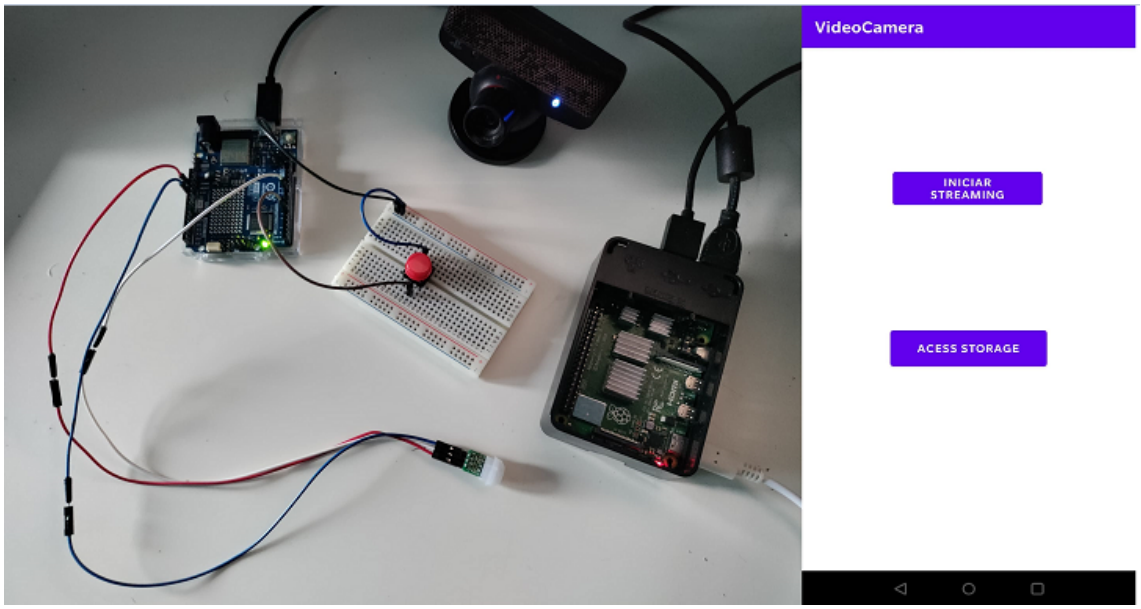


Figure 9: Montagem do hardware

4.2 Software

Em termos de módulos de software e respetiva correspondência, utilizamos as seguintes tecnologias e bibliotecas:

- **Notificações**: Implementámos as notificações utilizando o Firebase e respetiva API.
- **Aplicação Android**: O desenvolvimento da aplicação móvel para Android foi realizado no Android Studio, utilizando a linguagem de programação Java.
- **Servidores HTTP REST**: Para implementar ambos os servidores HTTP REST (Storage e Vídeo), optámos por utilizar a linguagem de programação Python em conjunto com a biblioteca Flask.
- **Streaming de live-video**: A funcionalidade de streaming de vídeo em tempo real foi implementada através da biblioteca aiortc, que oferece o suporte necessário para a tecnologia WebRTC (Web Real-Time Communication).
- **Processamento de Vídeo**: Para as tarefas de processamento de vídeo, como captura e codificação de frames, recorremos à biblioteca OpenCV (Open Source Computer Vision Library).

5 Integração

Para demonstrar a montagem dos componentes, instalação de software e preparação dos testes, decidimos realizar dois vídeos. O Video-demo representa um demo do prototipo final, e nele demonstramos a montagem dos componentes e a execução/preparação dos testes.O Software-install mostra a instalação dos principais modulos python utilizados.

6 Avaliação

Para garantir que o nosso protótipo cumpre todos os requisitos, realizamos um conjunto de testes que permite provar que o sistema atingiu os requisitos definidos na secção 2.

- Deteção de presença na porta deve ser < do que 3 seg
  - Realizámos várias simulações em que movimentávamos-nos a frente ao sensor. Em todos os casos, a deteção ocorreu em menos de 3 segundos, cumprindo o requisito.
- Início de gravação desde deteção ou toque na campainha deve ser < do que 3 seg
  - Efetuámos diversos testes em ambas as situações (movimento e toque na campainha) e verificámos que a gravação iniciava sempre em menos de 3 segundos.
- Pedido de stream de video do smartphone deve ter um atraso < do que 10 seg
  - Verificámos que o atraso depende fortemente da qualidade da ligação à internet. No entanto, em todas as situações testadas com uma ligação estável, o atraso foi inferior a 10 segundos. Em contextos com ligação muito fraca, este requisito poderá eventualmente não ser cumprido, mas em condições "normais" o desempenho foi satisfatório.
- Presença deve ser detetada no mínimo a 50 cm
  - Este requisito foi mais difícil de validar de forma precisa, pois o sensor apresentou alguma inconsistência na deteção, mesmo a distâncias curtas. Suspeitamos que possa haver algum mau funcionamento do próprio. Contudo, segundo as especificações do sensor, o seu alcance pode chegar até 7 metros, pelo que, teoricamente, o requisito é cumprido.

7 Github

O código fonte deste projeto encontra-se disponível para consulta no seguinte repositório Git: Câmara na campainha