**PROJETO INTEGRADO:**

CAMPAINHA INTELIGENTE - KNOCK



**AUTORES:**

**André Medina(82611),Francisco Pires (82604)**

**DISCIPLINA/PROFESSOR:**

**Projeto Integrado/Fernando Barros**

**Índice**

[1 – Re sumo/Abstract 3](#_mudimd6vdcoz)

[2 – Introdução 4](#_tszm99134as)

[3 - Descrição do Problema 4](#_d6vw2mxgqcq7)

[4 - Desenvolvimento 5](#_cwdvvsyv8slq)

[4.1 Materiais e Tecnologias Utilizadas 6](#_4842usymmx4j)

[4.2 Funcionamento Geral do Sistema 7](#_m5m313ktiluu)

[4.3 Implementação do Circuito de Controlo (Botão) 8](#_m5t4kxis8rrc)

[4.3.1 Estabilização do Sinal Lógico (Pull-down/Pull-up) 8](#_c5ndcjy248jy)

[4.3.2 Proteção do Circuito e Limitação de Corrente 8](#_93l7n0l0vh8g)

[4.4 Aplicação Android e Firebase 9](#_or9c1rw8b5k8)

[4.4.1 Concepção e Prototipagem da Aplicação 9](#_kfkem1sllu7c)

[4.4.2 Desenvolvimento da Aplicação em Android Studio 10](#_ichrbhscfifm)

[4.4.3 Integração com Firebase Cloud Messaging 11](#_qk8q5z1mqg2a)

[4.4.4 Gestão do Token do Dispositivo 12](#_iwtsydjh1rq8)

[4.4.5 Transmição Video entre a Raspberry Pi e a Aplicação Android 12](#_o5btrdse55j9)

[4.4.6 Comunicação entre a Raspberry Pi e a Aplicação Android 13](#_ai5qjif5xllu)

[4.4.7 Testes e Validação da Aplicação 13](#_8j97jjwix674)

[4.4.8 Considerações Finais sobre a Aplicação 14](#_zafczjw7t8f)

[4.5 Processamento por Voz e Integração com ChatGPT 14](#_6jrwuav02vu9)

[4.6 Testes e Validação 15](#_dr4ulxdvx5wk)

[4.6.1 Testes ao Botão Físico e GPIO 16](#_m4g5vhd6pz58)

[4.6.2 Testes ao Envio de Notificações Firebase 16](#_1y2z0ewzww7b)

[4.6.3 Testes de Integração do Sistema Completo 16](#_tngq739yjaut)

[Códigos Principal do Sistema (Python) 18](#_vcd5gbz47f3n)

[Diagrama de Funcionamento do Sistema 22](#_1uh6fk9e8cdr)

[**Link do Projeto Github: 23**](#_pl96ngpviaem)

[6 - Conclusões 24](#_2uzods73luyx)

# 1 – Re sumo/Abstract

O presente relatório descreve o desenvolvimento de uma campainha inteligente com interação por voz e notificações móveis, desenvolvida no âmbito da disciplina de Automação e Controlo. O sistema permite alertar o proprietário quando alguém toca à campainha e transmitir, de forma resumida, a intenção do visitante.

A solução é composta por uma Raspberry Pi ligada a um botão físico (GPIO), um sistema de reconhecimento de voz e uma aplicação Android. Sempre que o botão é pressionado, o sistema envia uma notificação para o telemóvel do utilizador através do Firebase Cloud Messaging e inicia uma interação por voz com o visitante.

A mensagem captada é processada por um sistema de inteligência artificial (ChatGPT), que gera um resumo da informação, o qual é novamente enviado para o utilizador sob a forma de notificação. Este projeto permitiu aplicar conhecimentos de programação, automação, redes e integração de serviços externos.

# 2 – Introdução

O relatório tem como objetivo apresentar o projeto final desenvolvido no âmbito da disciplina de Automação e Controlo, que consiste na conceção e implementação de uma câmara-campainha inteligente. Este sistema foi idealizado com o propósito de aumentar a segurança e a conveniência dos proprietários, permitindo identificar, em tempo real, quem se encontra à porta.

O funcionamento do sistema baseia-se na integração de uma câmara de vídeo com um módulo de inteligência artificial capaz de reconhecer rostos. Sempre que alguém se aproxima ou pressiona o botão da campainha, o dispositivo capta a imagem e executa um algoritmo de reconhecimento facial.

A Raspberry Pi é responsável pelo processamento dos dados e pelo envio de notificações automáticas ao proprietário, garantindo uma comunicação rápida e eficaz, mesmo quando este se encontra fora de casa.

# 3 - Descrição do Problema

Atualmente, a segurança residencial é uma preocupação crescente. Os sistemas de campainha tradicionais não permitem a identificação do visitante, obrigando o utilizador a abrir a porta para verificar quem se encontra do outro lado, o que pode representar um risco.

Com o avanço das tecnologias de reconhecimento facial e a acessibilidade de plataformas como a Raspberry Pi, tornou-se possível desenvolver uma solução inteligente, autónoma e acessível que permita identificar visitantes em tempo real e notificar o proprietário de forma imediata.

# 

# 4 - Desenvolvimento

O desenvolvimento do projeto foi orientado para a criação de um sistema de campainha inteligente baseado em interação por voz, notificações móveis e processamento de linguagem natural, sem recurso a sistemas de visão ou deteção por câmara.

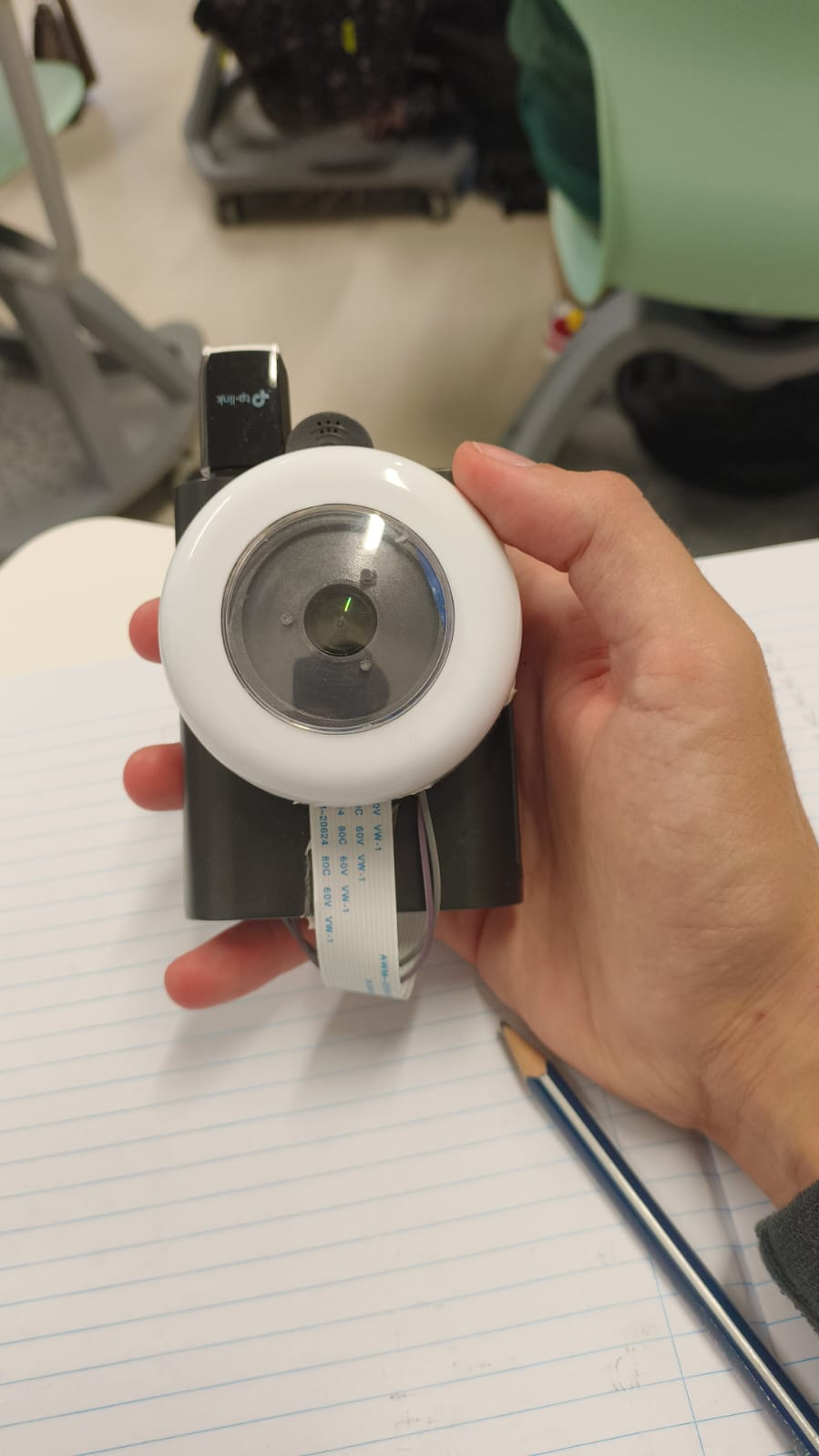
A arquitetura do sistema é composta por três componentes principais: uma aplicação Android, um script em Python executado numa Raspberry Pi e a integração com serviços externos como Firebase Cloud Messaging e a API da OpenAI.

O funcionamento do sistema é implementado através de um script em Python, apresentado no Anexo A, responsável pela gestão do botão, interação por voz, processamento da mensagem e envio de notificações.

## 4.1 Materiais e Tecnologias Utilizadas

Para a construção do sistema, foram utilizados os seguintes componentes e ferramentas:

* Botão físico para a campainha ligado ao GPIO 17 + Resistência 1kOhm;
* Microfone e coluna para interação por voz;
* Aplicação Android desenvolvida em Android Studio;
* Prototipagem da interface da aplicação em Figma;
* Firebase Cloud Messaging (FCM) para envio de notificações push;
* Módulo OV5647 - Raspberry Pi
* Linguagem Python;
* Bibliotecas Python: gpiozero, speech\_recognition, gTTS, firebase\_admin, requests;
* API da OpenAI (ChatGPT) para processamento e resumo da mensagem do visitante.



## 4.2 Funcionamento Geral do Sistema

O sistema inicia o seu funcionamento quando o visitante pressiona o botão físico da campainha, ligado ao pino GPIO 17 da Raspberry Pi. Este evento desencadeia uma sequência de ações automáticas:

1. Emissão de um som de campainha;
2. Envio imediato de uma notificação push para o telemóvel do utilizador, indicando que alguém se encontra à porta;
3. Ativação de um sistema de interação por voz que solicita ao visitante que se identifique e indique o motivo da visita;
4. Captação do áudio através do microfone e conversão de voz para texto;
5. Envio do texto para a API do ChatGPT, que realiza um resumo da mensagem;
6. Envio de uma nova notificação push para o telemóvel com o conteúdo resumido;
7. Resposta por voz ao visitante, informando que alguém irá atender.

## 4.3 Implementação do Circuito de Controlo (Botão)

No desenvolvimento do sistema de acionamento da campainha, foi necessária a integração de uma resistência em série com o botão de pressão (push-button). Esta adição não é meramente acessória, mas sim uma exigência técnica para garantir a fiabilidade do sinal digital e a integridade do hardware.

### 4.3.1 Estabilização do Sinal Lógico (Pull-down/Pull-up)

**Estado Flutuante (*Floating*):** Quando o botão não está premido, o pino de entrada não está ligado nem ao VCC nem ao GND, ficando "a flutuar". Isto resulta em leituras erráticas e falsos positivos no acionamento da campainha.

**Função da Resistência:** A resistência força o pino a um estado lógico bem definido (0V no caso de uma configuração *pull-down*) enquanto o botão está em repouso, garantindo que o sistema apenas detete sinal quando houver uma intenção real de acionamento.

### 4.3.2 Proteção do Circuito e Limitação de Corrente

A resistência desempenha também um papel crucial na segurança elétrica do projeto:

* **Prevenção de um Curto-Circuito:** Caso o botão fosse ligado diretamente entre o pólo positivo e o negativo para sinalizar a entrada, o seu fecho criaria um caminho de resistência quase nula, provocando um curto-circuito.
* **Controlo da Corrente:** A resistência limita o fluxo de corrente que atravessa o circuito de entrada, protegendo tanto o botão como os pinos do microcontrolador contra picos de corrente que poderiam danificar permanentemente os componentes.

Desta forma, a soldadura da resistência garante que o sistema de campainha apresente um comportamento determinístico (previsível) e robusto, eliminando interferências ambientais e assegurando a longevidade do equipamento eletrónico utilizado.

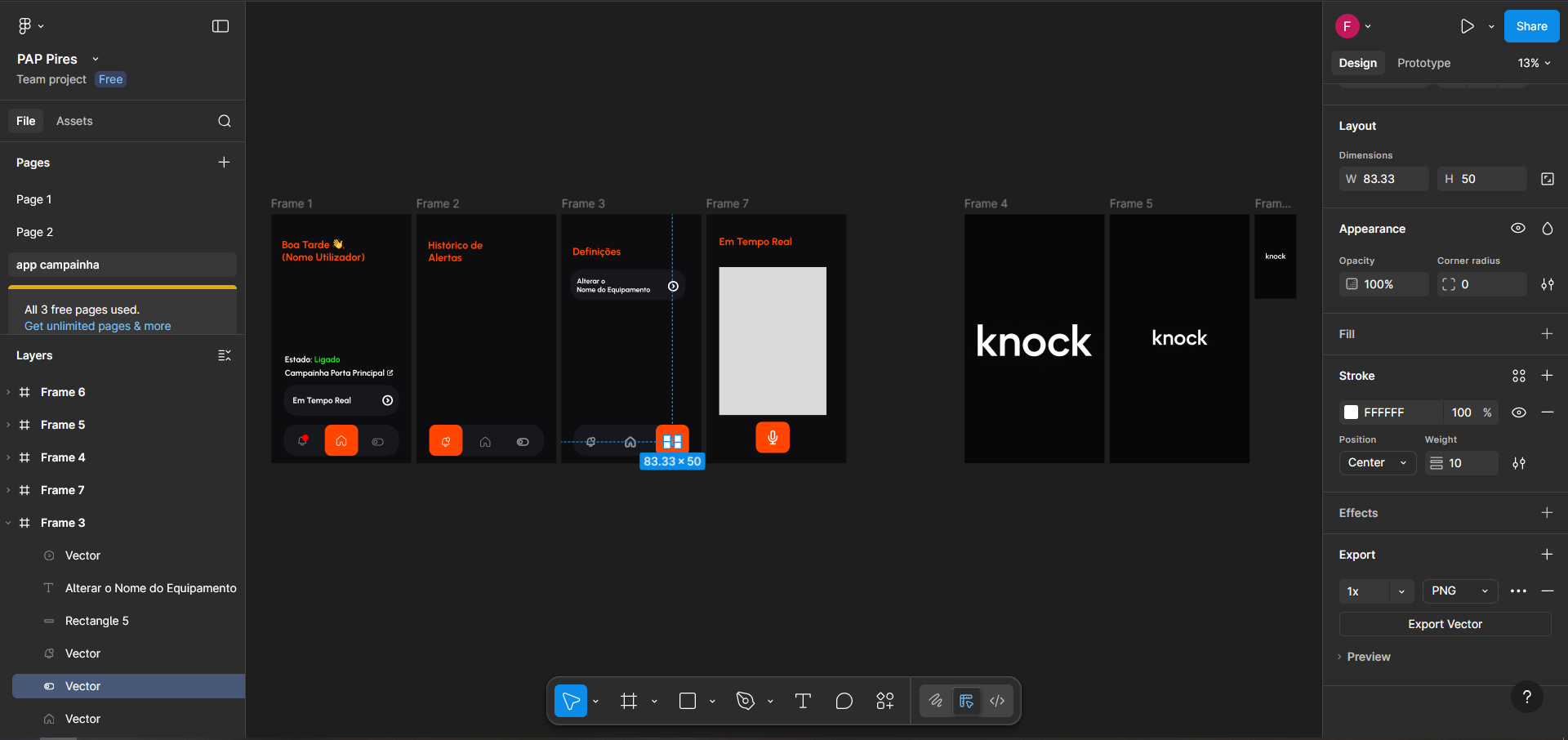
## 4.4 Aplicação Android e Firebase

A aplicação Android desenvolvida no âmbito deste projeto teve como principal objetivo permitir ao utilizador receber notificações em tempo real sempre que a campainha inteligente é acionada. Para tal, foi utilizada a plataforma Android Studio, ambiente oficial de desenvolvimento de aplicações Android, garantindo compatibilidade, estabilidade e facilidade de integração com serviços externos, nomeadamente o Firebase Cloud Messaging (FCM).

### 4.4.1 Concepção e Prototipagem da Aplicação

Antes do desenvolvimento da aplicação, foi realizada a fase de conceção da interface gráfica. Para este efeito, foi utilizado o Figma, ferramenta de prototipagem que permitiu planear antecipadamente a disposição dos elementos visuais da aplicação, como ecrãs, botões e zonas de informação.

Esta fase revelou-se importante, pois possibilitou uma visualização clara da aplicação final antes da sua implementação em código, reduzindo erros e facilitando a organização do desenvolvimento. O protótipo serviu como guia para a criação da interface no Android Studio.



### 4.4.2 Desenvolvimento da Aplicação em Android Studio

Após a fase de prototipagem, a aplicação foi desenvolvida no Android Studio, recorrendo à linguagem Kotlin, atualmente recomendada para o desenvolvimento de aplicações Android. A estrutura do projeto segue o modelo padrão Android, sendo composta por uma atividade principal (*MainActivity*), responsável pela inicialização da aplicação e gestão da interface do utilizador.

A aplicação foi desenvolvida com uma interface simples e funcional, uma vez que o seu principal objetivo é apresentar notificações ao utilizador de forma clara e imediata, sem necessidade de interações complexas.



### 4.4.3 Integração com Firebase Cloud Messaging

Para permitir o envio de notificações push, foi integrado o serviço Firebase Cloud Messaging (FCM). O Firebase é uma plataforma da Google que fornece vários serviços para aplicações móveis, sendo o FCM utilizado especificamente para o envio de mensagens e notificações em tempo real.

Durante a configuração do Firebase, foi criado um projeto na consola Firebase e associada a aplicação Android a esse projeto. Após essa associação, o Firebase gera um **token único** para cada dispositivo Android onde a aplicação é instalada.

Este token funciona como um identificador exclusivo do dispositivo e é essencial para garantir que as notificações sejam enviadas corretamente para o utilizador pretendido.

### 4.4.4 Gestão do Token do Dispositivo

Quando a aplicação Android é executada pela primeira vez, o Firebase atribui automaticamente um token único ao dispositivo. Este token é posteriormente utilizado pelo sistema da campainha inteligente para direcionar as notificações.

O token obtido na aplicação foi copiado e configurado no script Python executado na Raspberry Pi. Desta forma, sempre que ocorre um evento, como o pressionar do botão da campainha, o script utiliza esse token para enviar a notificação diretamente para o dispositivo Android correto.

Este processo revelou-se uma das etapas mais sensíveis do projeto, pois qualquer erro na configuração do token resultava na não receção das notificações, exigindo vários testes e validações.

### 4.4.5 Transmição Video entre a Raspberry Pi e a Aplicação Android

O sistema foi configurado para utilizar o protocolo **RTSP**, que garante o menor atraso possível no vídeo.

* Servidor de Streaming: [MediaMTX](https://github.com/bluenviron/mediamtx) (antigamente rtsp-simple-server).
  + Porta RTSP: 8554
  + Porta HTTP (Web): Alterada para 8884 (para evitar conflitos com a porta 8888).
* Captura de Vídeo: Utilização do stack moderno libcamera através do comando rpicam-vid.
* Pipeline de Dados: rpicam-vid (Captura H.264) ➔ ffmpeg (Encapsulamento) ➔ MediaMTX (Distribuição).

Ora Isto é facilmente executado com o seguinte comando:

|  |
| --- |
| /usr/bin/rpicam-vid -t 0 --inline --listen -n --width 1280 --height 720 --framerate 30 -o - | ffmpeg -i pipe:0 -c:v copy -f rtsp rtsp://127.0.0.1:8554/doorbell |

Na Aplicação, a abordagem inicial via WebView foi substituída por uma implementação nativa, uma vez que browsers comuns não suportam RTSP.

* Biblioteca Utilizada: ExoPlayer (Google).
* Componente de UI: StyledPlayerView (integrado dentro de um ConstraintLayout e CoordinatorLayout).
* Protocolo de Rede: RTSP sobre TCP (para maior estabilidade na rede Wi-Fi).
* URL de Conexão: rtsp://192.168.1.101:8554/doorbell.

### 4.4.6 Comunicação entre a Raspberry Pi e a Aplicação Android

A comunicação entre a Raspberry Pi e a aplicação Android não é feita de forma direta, mas sim através do serviço Firebase Cloud Messaging. A Raspberry Pi, através do script Python, envia uma mensagem para o servidor Firebase, que por sua vez encaminha a notificação para a aplicação Android associada ao token registado.

Esta abordagem permite uma comunicação segura, fiável e escalável, evitando a necessidade de ligações diretas entre a Raspberry Pi e o dispositivo móvel, o que poderia representar riscos de segurança e dificuldades de configuração de rede.

### 4.4.7 Testes e Validação da Aplicação

Após a implementação da aplicação e da integração com o Firebase, foram realizados vários testes com o objetivo de validar o correto funcionamento do sistema. Estes testes incluíram:

* Verificação da receção de notificações com a aplicação aberta;
* Testes com a aplicação em segundo plano;
* Testes com o ecrã do dispositivo bloqueado;
* Testes de envio repetido de notificações.

Os resultados obtidos foram positivos, confirmando que a aplicação Android recebe corretamente as notificações enviadas pela Raspberry Pi em diferentes cenários de utilização.

### 4.4.8 Considerações Finais sobre a Aplicação

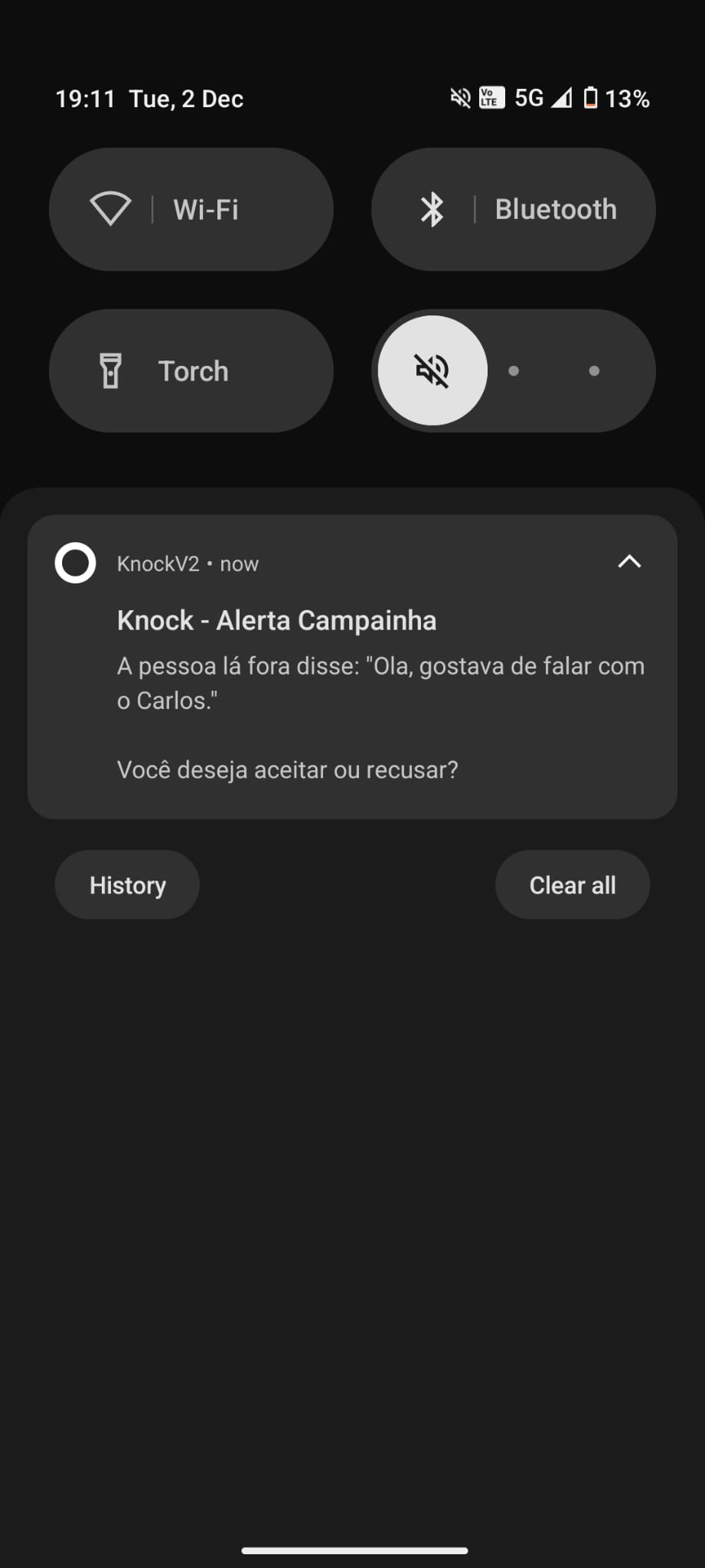
A aplicação Android desenvolvida cumpre de forma eficaz o seu objetivo principal, funcionando como uma interface simples e intuitiva entre o utilizador e o sistema de campainha inteligente. A integração com o Firebase revelou-se uma solução adequada, permitindo notificações rápidas e fiáveis.

Apesar da simplicidade da aplicação, esta desempenha um papel fundamental no funcionamento global do sistema, sendo responsável por garantir que o utilizador é informado em tempo real sobre qualquer evento ocorrido na campainha.

## 4.5 Processamento por Voz e Integração com ChatGPT

Após o toque na campainha, o sistema utiliza a biblioteca SpeechRecognition para captar e interpretar a fala do visitante. O texto obtido é enviado para a API do ChatGPT, que gera um resumo em português de Portugal.

Este resumo é posteriormente enviado para o utilizador através de uma notificação Firebase, permitindo que o proprietário tenha conhecimento claro e rápido da intenção do visitante.



## 4.6 Testes e Validação

Após a implementação de todos os componentes do sistema, foi realizada uma fase extensa de testes com o objetivo de validar o correto funcionamento da campainha inteligente em diferentes cenários de utilização. Esta etapa foi fundamental para identificar falhas, corrigir erros e garantir a estabilidade do sistema antes da sua utilização final.

Os testes foram efetuados de forma progressiva, começando por testes individuais a cada componente e avançando gradualmente para testes de integração do sistema completo.

### 4.6.1 Testes ao Botão Físico e GPIO

Numa fase inicial, foram realizados testes exclusivamente ao botão físico ligado ao pino GPIO 17 da Raspberry Pi. O objetivo era garantir que o pressionar do botão fosse corretamente detetado pelo sistema.

Durante estes testes surgiram alguns problemas, nomeadamente:

* Leituras falsas do botão, mesmo sem este ser pressionado;
* Deteções múltiplas com apenas um toque.

Estes problemas foram resolvidos através da configuração do parâmetro *bounce\_time* e da realização de vários testes até obter um comportamento estável e consistente.

### 4.6.2 Testes ao Envio de Notificações Firebase

Uma das fases mais críticas dos testes foi a validação do envio de notificações através do Firebase Cloud Messaging. Inicialmente, foram encontrados vários problemas, tais como:

* Notificações que não chegavam ao dispositivo Android;
* Erros relacionados com tokens inválidos;
* Falhas de autenticação no serviço Firebase.

Estes erros foram sendo identificados ao longo do desenvolvimento, obrigando à verificação da configuração do projeto Firebase, das credenciais utilizadas e da correta associação do token do dispositivo Android. Após estas correções, o envio de notificações passou a funcionar de forma consistente.

### 4.6.3 Testes de Integração do Sistema Completo

Depois de validados os componentes individualmente, foram realizados testes ao sistema completo. Estes testes consistiram em simular o funcionamento real da campainha, desde o pressionar do botão até à receção da notificação final no dispositivo móvel.

Durante esta fase, foram identificados problemas de sincronização entre os diferentes processos, nomeadamente atrasos entre a interação por voz e o envio da segunda notificação. Estes atrasos foram reduzidos através da otimização do código e da organização das funções em sequência lógica.

**5 - Resultados**

Após a implementação e integração de todos os componentes do sistema, foram realizados vários testes com o objetivo de avaliar o funcionamento, fiabilidade e tempo de resposta da campainha inteligente.

Verificou-se que a deteção do pressionar do botão físico ligado ao GPIO 17 da Raspberry Pi ocorre de forma imediata e consistente, demonstrando uma correta configuração do hardware e do software. Sempre que o botão é pressionado, o sistema reage instantaneamente, iniciando o processo de notificação e interação por voz.

O envio da primeira notificação para o dispositivo móvel, através do Firebase Cloud Messaging, revelou um tempo de resposta reduzido, normalmente inferior a dois segundos. Esta notificação informa o utilizador de que alguém se encontra à porta, permitindo uma reação rápida.

A componente de interação por voz funcionou de forma satisfatória, permitindo que o visitante se identifique e explique o motivo da sua visita. O reconhecimento de voz apresentou bons resultados na maioria dos testes, embora fatores como ruído ambiente possam influenciar a precisão.

O processamento da mensagem pelo sistema de inteligência artificial (ChatGPT) demonstrou ser eficaz, gerando um resumo claro e objetivo da informação transmitida pelo visitante. A segunda notificação enviada para o utilizador apresenta essa informação de forma simples e compreensível.

De um modo geral, os resultados obtidos confirmam que o sistema cumpre os objetivos propostos, funcionando de forma estável e adequada para um contexto doméstico.

# Códigos Principal do Sistema (Python)

Este anexo apresenta o código principal desenvolvido em Python e executado na Raspberry Pi.  
O script é responsável por detetar o toque na campainha através do GPIO, realizar a interação por voz com o visitante, processar a mensagem utilizando inteligência artificial e enviar notificações para o dispositivo móvel através do Firebase Cloud Messaging.

|  |
| --- |
| ffrom gtts import gTTS  import os  import speech\_recognition as sr  import time  import vlc  from configparser import ConfigParser  import re  import requests  import firebase\_admin  from firebase\_admin import credentials, messaging  import threading  from openai import OpenAI  client = OpenAI(api\_key="sk-proj-wNNd7r5brEQ2XjhepmnspOnNtZ7rMPhiE6cp4SIYzRh4lsVeaf5kej4AENXi4mJub4tyDpgOljT3BlbkFJTvcZoJbqlyEKL0Fr\_biSmsuZdIidycpXLabPtfnix21zqWerOIFo2RC\_pjel1-dJpwt1cTmNgA")  cred = credentials.Certificate("serviceAccountKey.json")  firebase\_admin.initialize\_app(cred)  token\_android = "fYO\_zrBbQGCa-3jGKQ8hjp:APA91bH9Owkr3J5bbgjAhzGkjOnuU6Hbh6eHdBlKVRc08PIILtl2CkLLTmWNSm\_Ppsxib-xVbW7Q5sLGlYGaZs3t5ajl9I-5kk1CGi4WdaMnyU0RhTbcqXc"  def enviar\_notificacao(conteudo):  print("A enviar notificação...")  message = messaging.Message(  notification=messaging.Notification(  title="Knock - Alerta Campainha",  body=conteudo  ),  token=token\_android,  )  try:  response = messaging.send(message)    except Exception as e:  print("Erro ao enviar:", e)  def luminaAi(pergunta):  try:  response = client.chat.completions.create(  model="gpt-4o-mini",  messages=[  {"role": "system", "content": "You are an intelligent, connected doorbell device. make a summary of what the person said. Answer in Portuguese-Portugal. Also say The person outside said (A pessoa lá fora disse:) and use quotation marks. Also ask if the user wants >  {"role": "user", "content": pergunta}  ]  )  content = response.choices[0].message.content  print("DEBUG:" + content)  enviar\_notificacao(content)  falar("Obrigado. Alguem ira atender a porta o mais rápido possivel. Da minha parte é tudo. Fique bem.")  except Exception as e:  print(e)  ativarEscrita()  def falar(text):  tts = gTTS(text=text, lang='pt', tld='pt')  tts.save("output.mp3")  os.system("mpg123 -q output.mp3")  def reconhecer():  portaalerta = "Alguem está á sua Porta!"  enviar\_notificacao(portaalerta)  falar("Bem-Vindo. Por favor. após o sinal diga como se chama e o que pretende. Obrigada.")    recognizer = sr.Recognizer()  with sr.Microphone() as source:  print("Pode Falar.")  audio = recognizer.listen(source, phrase\_time\_limit=20)  try:  dados = recognizer.recognize\_google(audio, language='pt-PT')  except:  print("Erro")  falar("desculpe, não percebi.")  def ativarEscrita():  pergunta = input('FALA: \n')  from gpiozero import Button  from signal import pause  botao = Button(17, bounce\_time=0.02)  print("À espera que carregue no botão...")  def quando\_carregar():  os.system("mpg123 -q campainha.mp3")  reconhecer()  def quando\_largar():  print("-> Botão Largado")  botao.when\_pressed = quando\_carregar  pause() |

# Diagrama de Funcionamento do Sistema

|  |
| --- |
| +--------------------+  | Visitante |  | (pressiona botão) |  +----------+---------+  |  v  +--------------------+  | Raspberry Pi |  | Botão GPIO 17 |  +----------+---------+  |  v  +------------------------------+  | Notificação Firebase |  | "Alguém está à sua porta" |  +----------+-------------------+  |  v  +------------------------------+  | Interação por Voz |  | Pedido de nome e intenção |  +----------+-------------------+  |  v  +------------------------------+  | Reconhecimento de Voz |  | (Speech-to-Text) |  +----------+-------------------+  |  v  +------------------------------+  | ChatGPT (OpenAI API) |  | Resumo da mensagem |  +----------+-------------------+  |  v  +------------------------------+  | Nova Notificação Firebase |  | Mensagem resumida |  +----------+-------------------+  |  v  +------------------------------+  | Utilizador (App Android) |  | Recebe notificação |  +------------------------------+ |

# Link do Projeto Github:

https://github.com/PiresFrancisco/Knock

# 6 - Conclusões

O projeto da campainha inteligente cumpriu com sucesso os objetivos propostos, permitindo a notificação do utilizador em tempo real e a comunicação indireta com o visitante através de interação por voz. O sistema revelou-se funcional, estável e adequado para um contexto doméstico.

A utilização da Raspberry Pi mostrou ser uma solução eficiente e económica, enquanto a integração com o Firebase e com a API do ChatGPT demonstrou a importância da ligação entre sistemas físicos e serviços na cloud.

Este trabalho contribuiu para o desenvolvimento de competências técnicas nas áreas da automação, programação em Python, desenvolvimento de aplicações móveis e utilização de inteligência artificial. Como trabalho futuro, poderá ser adicionada uma câmara ou outras funcionalidades de automação residencial.