Francisco Turégano Caetano Morão Lourenço

Licenciado em Ciências da Engenharia Eletrotécnica e de Computadores

Deteção de Volume através de Sensores Térmicos

Dissertação para obtenção do Grau de Mestre em

Engenharia Eletrotécnica e Computadores

Orientadora: Anikó Katalin Horváth da Costa, Professora Auxiliar, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa

Júri

Presidente: Prof. Doutor João Francisco Alves Martins, FCT-UNL
Arguente: Prof. Doutor Filipe de Carvalho Moutinho, FCT-UNL
Vogal: Prof. Doutora Anikó Katalin Horváth da Costa, FCT-UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Dezembro, 2019

Deteção de Volume através de Sensores Térmicos

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

À minha família.

Agradecimentos

Em primeiro lugar, um agradecimento à professora Anikó Katalin Horváth da Costa, a orientadora desta dissertação, por toda a disponibilidade demonstrada, assim como todo o empenho e dedicação, e por todo o auxílio prestado, durante o desenvolvimento deste sistema.

Quero agradecer também aos meus pais, que contribuíram sempre para o meu crescimento como pessoa, e que me transmitiram os valores pelos quais me guio diariamente, e em quem tenho um especial orgulho, ao meu irmão, que me acompanha desde sempre, e a toda a minha família que me apoiou ao longo desta etapa da minha vida, e em todas as outras.

Por fim, um agradecimento a todos os meus amigos, tanto aqueles que levo desde sempre, como os que conheci a partir do momento em que ingressei na FCT-UNL, e os quais considero também uma família, em especial aqueles que me acompanharam do início ao fim desta etapa e os que conheci noutros momentos, mas que igualmente me marcaram.

Resumo

O conceito de *IoT*, *Internet of Things*, é um novo paradigma tecnológico relacionado com a existência de uma rede global de dispositivos, em constante comunicação entre si. Sendo esta uma área cada vez mais presente no atual contexto global, existem mais soluções para implementação de sistemas que envolvem uma constante comunicação e transmissão de dados, e cuja utilidade é a melhoria de processos e da qualidade de vida.

Existem aspetos negativos que prejudicam a vida diária urbana e, recorrendo a dispositivos inseridos no conceito apresentado no parágrafo anterior, poderiam sofrer uma melhoria. Um exemplo é a desorganização verificada, diariamente, em transportes públicos urbanos, sem pré-reserva de ingresso. Durante horas de maior congestionamento, existe uma dificuldade acrescida na movimentação nas entradas/saídas dos meios de transporte, como por exemplo o metro.

Deste modo, considerando os factos expostos, verificou-se a necessidade da existência de um sistema, cuja principal finalidade é a monitorização volumétrica, com recurso a sensores de imagem térmica, em tempo real, em áreas cruciais na mobilização diária de massas populacionais de meios de transporte de carruagens, públicos.

Além do mais, pretende-se também que seja garantido o acesso aos dados recolhidos, a partir de plataformas que permitam a exposição da informação, de forma simples e eficaz. Os dados recolhidos pelo protótipo do sistema devem ser armazenados, de modo que a informação seja posteriormente filtrada, e analisada de acordo com as necessidades dos utilizadores, por exemplo na previsão de eventuais flutuações nos fluxos de mobilização, em certos padrões temporais.

Por fim, o sistema de controlo volumétrico descrito nesta dissertação foi implementado e testado, sendo que os resultados obtidos foram os previstos, com o acréscimo de algumas críticas aos mesmos.

Palavras-chave: Controlo Volumétrico, Raspberry Pi, AMG8833, Python, Firebase Real-time Data-base, Firebase Authentication, Aplicação Móvel Android.

Abstract

The concept of *IoT*, Internet of Things, is a new technological paradigm related to the existence of a global network of devices, in constant communication between each other. Being this an area increasingly present in the actual global context, there are more solution to the implementation of systems constant communication and transmission of data, and which goal is to improve processes and the quality of life.

There are negative aspects that harm the daily urban life and, with the resort of devices inserted in the concept presented in the paragraph before, they could suffer an improvement. An example of this is the disorganization verified, day to day, in public urban transport, with no pre-booking of tickets. During the most crowed hours, there is an increase in the difficulty on moving, in the entrances/exists of the public means of transportation, like subway.

That way, considering the exposed facts, it was verified the need of existence of a system, which main goal is the volumetric monotorization, with resource of thermal image sensors, in real time, in crucial areas to the daily mobilization of people masses of the public carriage means of transport.

Furthermore, it's intended that the access to the gathered data is granted, from platforms that can expose the information, simply and effectively. The data gathered by the system's prototype must be stores, in a way that the information is, posteriorly filtered, and analyzed according to the user needs, for example in the prediction of eventual fluctuations in the flow of mobilization, in certain time patterns.

Finally, the volumetric control system, described in this dissertation, has been implemented, and properly tested in the following chapters, being that the results obtained were the predicted, with the addiction of some criticism to them.

Keywords: Volumetric Control, Raspberry Pi, AMG8833, Python, Firebase Realtime Databa-se, Firebase Authentication, Android Mobile Application.

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Siglas

AM	Aplicação Móvel
B/W	*Black and White*
BPMN	*Business Process Model and Notation*
CPU	*Central Process Unit*
CSI	*Camera Serial Interface*
DSI	*Display Serial Interface*
FA	*Firebase Authentication*
FRD	*Firebase Realtime Database*
GPIO	*General Purpose Input/Output*
HSV	*Hue Saturation Value*
HTTP	*Hypertext Transfer Protocol*
I/O	*Input/Output*
IoT	*Internet of Things*
IR	Infravermelhos
JSON	*JavaScript Object Notation*
LDR	*Light Dependent Resistor*
LED	*Light-Emitting Diode*
PIR	*Passive Infrared*

RAM	*Random Acess Memory*
RGB	*Red Green Blue*
ROI	*Region of Interest*
SCL	*Serial Clock Line*
SD	*Secure Digital*
SDA	*Serial Data Line*
UI	*User Interface*
XML	*Extensible Markup Language*

Capítulo 1

Introdução

Ao longo deste capítulo, é apresentada a motivação associada à realização desta dissertação, assim como é colocada uma pergunta de investigação, para a qual é formulada uma hipótese para uma possível solução. A última secção deste capítulo apresenta uma sucinta explicação da organização estrutural desta dissertação, isto é, a temática abordada em cada um dos capítulos.

1.1 Motivação

Atualmente, denota-se uma crescente dificuldade na mobilização das massas populacionais. Por exemplo, em transportes públicos em que não exista a pré-reserva de um lugar fixo ao comprador, existe uma sobrelotação de certos espaços, em detrimento de outros, verificando-se assim uma completa desorganização populacional.

Além do mais, com as possibilidades já existentes para a criação de sistemas eficazes e económicos, a monitorização constante do volume ocupado desses mesmos meios de transporte permite efetuar uma previsão, o que pode ser útil na escolha, por exemplo, do número de carruagens a utilizar num meio de transporte como o metro, num certo dia, a uma determinada hora.

Deste modo, e na perspetiva de finalista do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, surgiu uma oportunidade de criar um sistema autónomo e eficaz, para a deteção e monitorização volumétrica, que pode contribuir para uma melhor organização populacional.

1.2 Objetivos

O grande objetivo desta dissertação é a criação de um dispositivo auxiliar, no que diz respeito à medição volumétrica de espaços, que possa contribuir, em diversos contextos, para o conhecimento do estado ocupacional. Deste modo, e como já foi mencionado anteriormente, pode tornar-se num contributo para que exista uma maior organização do espaço que envolve grandes massas populacionais.

Deve ainda ser considerado o problema da proteção de dados, protegendo qualquer informação sensível acerca dos utentes do espaço.

1.3 Pergunta de Investigação

Considerando os objetivos propostos, foi então desenvolvida uma pergunta de investigação, para a qual é formulada uma hipótese, que cumpra os requisitos para a captação da ocupação e da exposição para a informação: "Como é possível obter informação sobre a ocupação de um espaço, sem infringir a lei de privacidade, e expô-la instantaneamente?"

Portanto, pretende-se o conhecimento da ocupação, através de sensores que não comprometam informação dos utilizadores do espaço onde a ocupação é efetivamente medida. Uma hipótese, estudada ao longo desta dissertação para esta problemática é a seguinte:

Hipótese:

Análise de imagens construídas a partir da informação recolhida por sensores térmicos, e posterior exposição numa aplicação móvel desenvolvida em *Android Studio*.

1.4 Estrutura do Documento

A presente dissertação está organizada de modo a que possa ser acessivelmente compreendida e interpretada, pelo respetivo leitor. Assim sendo, em seguida são apresentados os capítulos (e o seu objetivo) que compõe a estrutura deste documento:

- **Estado de Arte** - Neste capítulo é apresentada uma pesquisa bibliográfica, que visa o estudo das tecnologias utilizadas em sistemas semelhantes ao exposto nesta dissertação., e uma abordagem aos protocolos de comunicação envolvidos, no sistema;
- **Descrição do Sistema Proposto** - Descrição acerca do sistema que se pretende implementar, e requisitos a cumprir;
- **Implementação do Sistema** - Descrição detalhada do procedimento realizado para a implementação, bem como materiais utilizados e as opções tomadas;
- **Testes e Validação** - Cenários de utilização do sistema, que confirmam o seu funcionamento e interações com a interface de utilizador;

- **Conclusão** - Conclusões retiradas pela análise do sistema, já totalmente finalizado, e sugestões de melhoria;
- **Bibliografia** - Referenciação dos documentos e artigos auxiliares, durante a realização desta dissertação;
- **Anexos** - Guia de utilização do protótipo implementado e anexos com detalhes relacionados com a implementação do sistema.

Capítulo 2

Estado de Arte

Neste capítulo, são descritas, em detalhe, tecnologias já existentes, assim como conceitos e projetos semelhantes.

Assim sendo, foi realizada uma pesquisa, essencialmente na área que envolve o projeto proposto, cujo objetivo é a obtenção de conhecimento auxiliar ao desenvolvimento do sistema.

2.1 Conceito de Automação

Pode definir-se automação como uma técnica, método, ou sistema de operação, com recurso a dispositivos eletrónicos, cujo objetivo é reduzir a intervenção humana e, em muitos casos, aumentar o desempenho e rentabilidade [1].

Ao longo dos anos, tem-se verificado um crescimento exponencial na necessidade de integrar sistemas de automação, em fábricas, escritórios, e até em habitações domésticas. Cada vez mais os investigadores e os industrialistas procuram construir sistemas eficientes, automáticos, e com um custo cada vez mais "aceitável", para que se possam controlar e monitorizar diferentes segmentos, como temperaturas ambiente, sistemas de luminárias, entre muitos outros.

Além do mais, a automação permite que seja, não só mais eficiente, como mais económica a utilização de recursos como a eletricidade, água e gás, reduzindo assim os desperdícios [1].

Figura 2.1: Exemplo da variedade de sistemas de automação que são concebíveis para uma habitação

Atualmente, é cada vez mais comum a existência de sistemas, não só a nível industrial, como a nível habitacional recorrendo muitas vezes a soluções que permitam uma implementação facilitada (Figura 2.1). A utilização de plataformas eletrónicas *open-source*[1], como é o *Arduino*[2], cuja placa permite a leitura de entradas, como temperaturas, humidade, entre muitos outros exemplos, e respetiva transformação em saídas, executando instruções para o microcontrolador que está presente na referida placa.

Para um melhor entendimento do tipo de sistema abordado, em seguida é apresentado um exemplo de um sistema de automação, projetado para uma habitação, recorrendo ao *Arduino* (Figura 2.2):

[1] Software para o qual o código-fonte é disponibilizado gratuitamente, e que pode ser redistribuído e modificado [2].

[2] O Arduino é uma plataforma eletrónica open-source, que é baseada numa placa, e no seu software, e que é caracterizada pela sua fácil interação. A referida placa é capaz de ler inputs, e respetivamente transformá-lo numa saída [3].

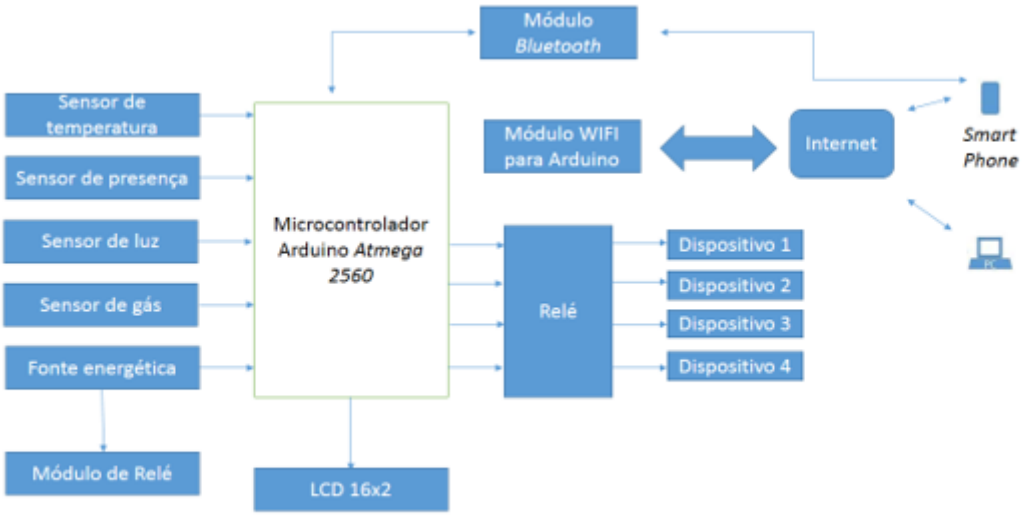

Figura 2.2: Arquitetura do sistema de automação, de uma habitação, adaptado de [4].

A Figura 2.2 demonstra como todos os dispositivos comunicam, não só com o controlador, mas também entre si, e quais os módulos utilizados de leitura dos dados.

Em primeiro lugar, existe um microcontrolador, responsável por obter as medições físicas relevantes, para este caso, através dos sensores que a ele estão conectados. A ativação da luz depende da presença e da necessidade de utilização, numa dada divisão, assim como da intensidade da luz irradiada pelo sol, o que é determinado recorrendo a um LDR[3](*Light Dependent Resistor*). Finalmente, ainda foi integrado um sensor de infravermelhos passivo (PIR, *Passive Infrared*)[4], que deteta movimentação dentro de casa, quando o sistema de segurança está ativo. O intuito do relé, onde estão ligados os quatro dispositivos é enviar sinais de controlo, do microcontrolador, para os dispositivos eletrónicos responsáveis por ativar ou desativar algum dos dispositivos, do sistema.

Para confirmar a identidade de um habitante, foi concebido um portal *web*, com um nome de utilizador e respetiva palavra-chave associada. Este portal interage em dois sentidos, tanto como entrada, como saída, sendo que no primeiro caso serve de controlo a qualquer comando enviado pelo utilizador, e no segundo como auxiliar para representação das leituras, assim como o estado de ativação dos dispositivos, o que se verifica, também, na aplicação desenvolvida para *smartphone* [4].

Assim sendo, concluímos que é possível o desenvolvimento de sistemas, recorrendo a componentes mais económicos, e com duas componentes, de *hardware* e *software*.

[3]Resistência que depende de luz
[4]Sensor que mede o movimento, através da medição de infravermelhos

2.2 Medição Volumétrica

A medição volumétrica é um dado que pode ser muito relevante em sistemas em que exista a necessidade de conhecermos o estado de ocupação, de um determinado espaço.

Para a obtermos a taxa de ocupação volumétrica, existem vários métodos que, enquadrando no contexto da automação, recorrem a sensores e a outras fontes de informação. Dada a existência de variados tipos de sensores para efetuar este tipo de medição, será apresentado um método, através dos qual é possível obter o volume, através da captação de imagens.

Sistema de Visão para Controlo Volumétrico

Neste exemplo, é apresentado um sistema que utiliza um sensor de imagem e um microcontrolador programável da *Mitsubishi*, para a medição volumétrica.

O grande objetivo deste sistema sensorial é a computação automática da área superficial e do volume de produtos agrícolas, assimétricos.

No caso deste sistema, a obtenção destes dados é realizada por aproximação, como uma soma de troncos cónicos. Para o cálculo da área, existe, por exemplo, um método designado de *Tape Method*[5] [5]. Neste procedimento, a fita é dividida em pequenos troços, por forma a que cubra completamente a superfície do objeto. Ao serem retirados os invólucros, pode então ser calculada a área total.

No entanto, a eficácia deste método está totalmente dependente de como são utilizados os pedaços de fita, manualmente. Posteriormente, ainda é necessária a conversão de unidade de área, sendo este método muito dispendioso a nível temporal, além da probabilidade intrínseca de ocorrência de erros humanos. Existe ainda um outro método para o cálculo da área, que a relaciona com a massa do objeto.

Quanto ao volume, em alguns casos, é usado o método do deslocamento de água. O produto seria completamente submerso em água, o que impossibilita o recurso a este procedimento, em diverso tipos de exemplos.

Pelo exposto, e depois inúmeros estudos acerca da utilização de imagem na agricultura, concluiu-se que este método, que recorre à imagem, é eficaz, económico e fiável.

Então, em seguida (Figura 2.3) é apresentado o protótipo para a arquitetura do sistema implementado:

[5]Método que, recorrendo a uma fita, permite o cálculo da área de um objeto

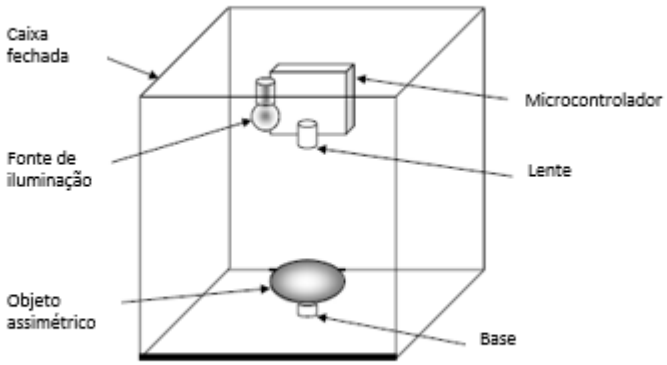

Figura 2.3: Sistema de visão, adaptado de [5]

Aquisição de Imagem

Para efetuar a aquisição da imagem, é utilizado um sensor de imagem, o *OmniVision OV7620 CMOS 1/3"*, que é controlado por um microcontrolador programável, *Mitsubishi MSA0654MEAUST M16C*. A imagem é capturada em modo *B/W* (*Black and White*) e, em seguida guardada numa memória externa, conectada ao microcontrolador.

Para evitar a interferência da luz ambiente no momento da captação da imagem, o dispositivo foi colocado numa caixa, iluminada controladamente através de uma lâmpada de 100W. Além do mais, para precaver qualquer problema que possa surgir na aquisição da imagem, o fundo da caixa protetora é negro, o que causa contraste com o produto e com a sua sombra [5].

Processamento de Imagem

Em primeiro lugar, a imagem foi alisada, recorrendo ao filtro *Sigma*, de dimensão 5x5, com um *sigma* igual a 2, cuja finalidade é a remoção de qualquer ruído existente. Além do mais, com o método do gradiente, as arestas foram também detetadas. Dada a resolução de intensidade, 8-bit, existem 256 níveis de brilho.

Depois de alisada, a imagem é também diferenciada, com a aplicação de um filtro *Sobel*, com operador de dimensão 3x3. Para a deteção dos contornos, é definido um threshold de 90. Deste modo, apenas os píxeis localizados nos contornos são brancos, e os restantes pretos.

Por fim, o contorno que é obtido é utilizado na calibração dimensional, assim como na computação do volume, e da área [5].

Os resultados dos dois processos mencionados estão expostos nas seguintes imagens, onde, em cada uma, é demonstrado o resultado da aplicação das operações, tanto da

imagem capturada, como após o seu alisamento e deteção dos respetivos contornos:

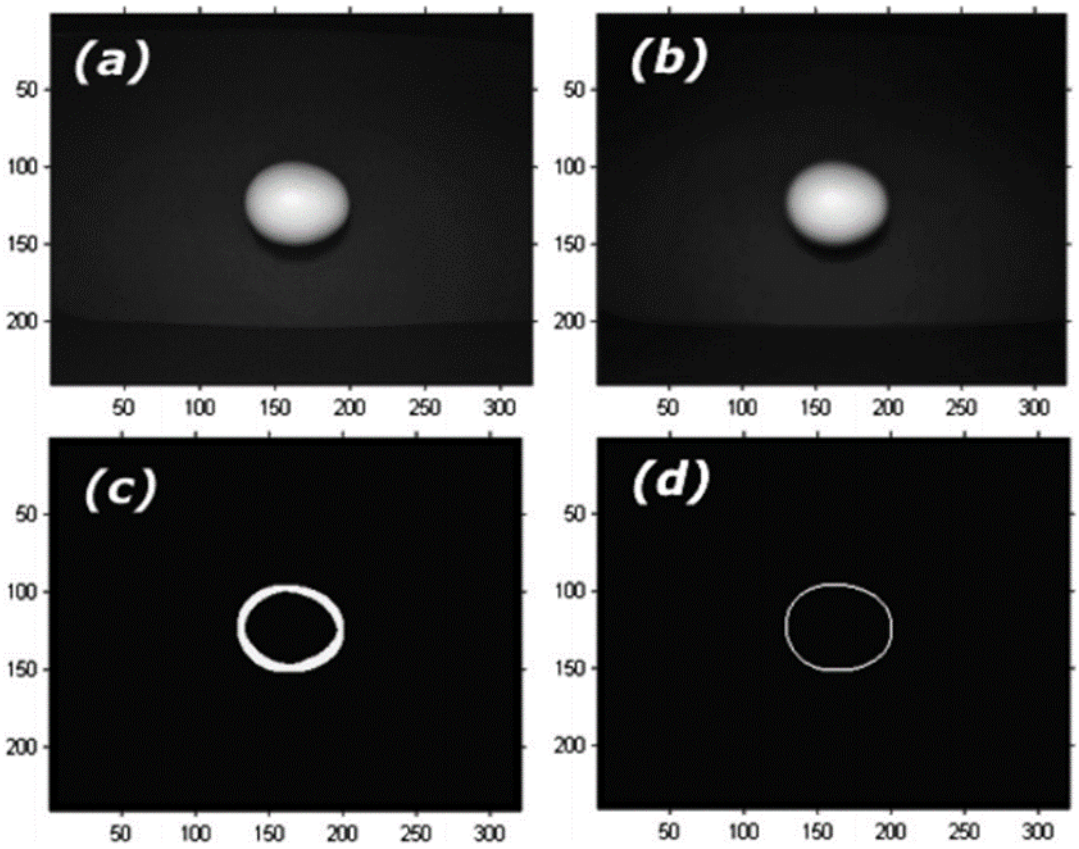

Figura 2.4: Progressão do tratamento de imagem [5]

Na imagem (a) da Figura 2.4 está representada a imagem inicialmente capturada, pelo sensor de imagem, e em modo *B/W* . Após a aplicação do filtro *Sigma*, resulta a imagem identificada pela letra (b).

A imagem representada pela letra (c) diz respeito à aplicação do filtro *Sobel* e, por fim, a imagem (d) demonstra o contorno exterior da imagem, que foi utilizado para calibrações, e também para a computação do volume e da área.

Relativamente ao cálculo da área superficial e do volume, através da calibração dimensional é possível converter o número de píxeis, no contorno da imagem, para uma unidade de distância correspondente. Deste modo, as fórmulas que foram utilizadas, para o cálculo baseiam-se em algumas variáveis, que dependem da forma do objeto. Quanto aos eixos, o de maior dimensão corresponde, sempre, ao eixo z, e todos os cortes transversais de relevância, consequentemente ao plano x-y.

Assim sendo, os objetos são modelados sob a forma de troncos cónicos[6], cujo volume e área podem ser obtidos recorrendo às fórmulas matemáticas representadas pelas equações 2.1 e 2.2 respetivamente e, para tal, é necessário conhecer, tanto os diâmetros inferior e

[6]Porção seccionada de um sólido geométrico, neste caso cónico

superior, como a altura do tronco (a Tabela 2.1 indica o significado representado por cada variável utilizada nas equações 2.1 e 2.2) [5]:

$$V_i = \frac{(\pi * h)}{12} * (D_{b_i}^2 + D_{bi} * D_{ti} + D_{t_i}^2) \tag{2.1}$$

$$A_i = \frac{\pi}{2} * (D_{bi} + D_{ti}) * \sqrt{h^2 + (\frac{D_{bi}}{2} - \frac{D_{ti}}{2})^2} \tag{2.2}$$

Onde:

Tabela 2.1: Definição das variáveis, das equações 2.1 e 2.2

Vi	Volume de um tronco cónico, de índex i
Ai	Área superficial, de índex i
h	Altura do tronco
Dbi	Diâmetro da base de um tronco, de índex i
Dti	Diâmetro do topo, de índex i

2.3 Câmaras Térmicas e Aquisição de Imagem

Uma câmara de infravermelhos é um dispositivo que converte, num sinal eletrónico, a radiação infravermelha, com o objetivo de gerar uma imagem térmica. A temperatura que é capturada por este tipo de câmara pode ser quantificada, para que seja possível observar o comportamento térmico de um corpo.

Em condições normais, no que diz respeito ao fundo da imagem capturada, quando o objetivo é obter o comportamento térmico de um corpo, o fundo desta mesma imagem pode ser desprezado, em virtude da temperatura do corpo ser representada numa área térmica reduzida. Na seguinte figura, podemos verificar o comportamento deste tipo de câmara, na atividade de captação de um corpo humano [7]:

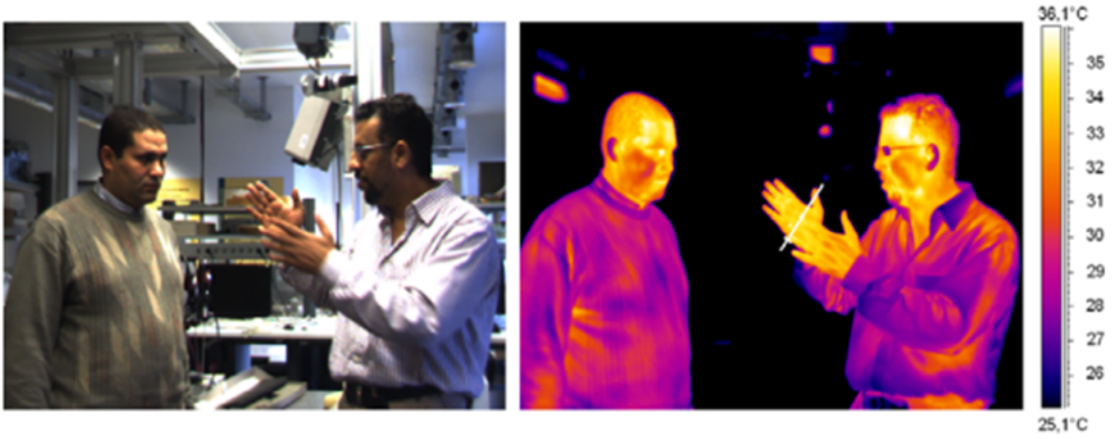

Figura 2.5: Exemplo representativo da imagem apresentada por uma câmara térmica [6]

Pela observação da Figura 2.5, é possível verificar que as áreas respeitantes às mãos e à cabeça estão claramente diferenciadas do fundo da imagem [7].

Em seguida, é apresentado um exemplo da aplicabilidade do mapeamento térmico, num sistema de deteção.

Sistema de Identificação do Estado Afetivo com Câmaras Térmicas

Para obter a informação acerca do estado de espírito de alguns utilizadores de computadores, foi então desenvolvida uma tecnologia, que se baseia na informação obtida da gravação contínua de uma imagem térmica da face, desses mesmos usuários.

Os criadores deste sistema descobriram que o estado de *stress*, de um determinado utilizador, está relacionado com o incremento do fluxo sanguíneo, numa zona específica da cabeça:

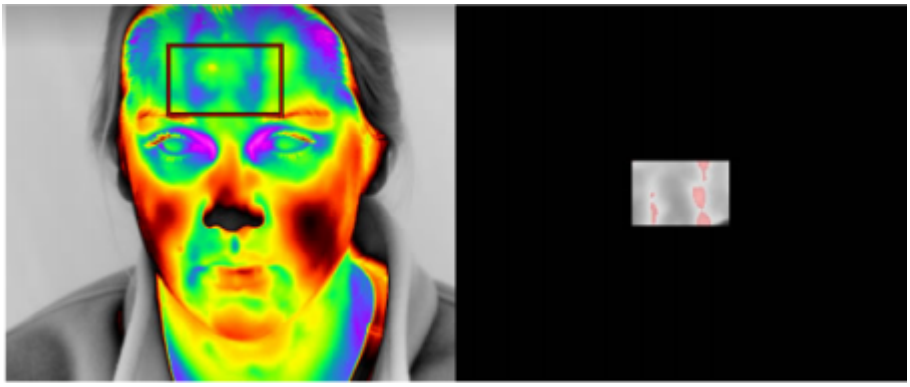

Figura 2.6: Zona de relevância para o sistema apresentado [7]

Foi então conectada uma câmara térmica de infravermelhos ao computador, como um periférico, com o objetivo de monitorizar a zona da face mais adequada para retirar qualquer tipo de conclusão, por ser constituída por uma camada de pele mais fina, o que permite a obtenção de uma imagem de melhor qualidade.

Sendo o foco deste sistema a identificação do estado de *stress*, apenas as zonas cerebrais associadas a este estado de espírito serão relevantes para o estudo.

Na zona da cabeça identificada na Figura 2.6, quando o utilizador é um submetido a algo que suscite o estado de espírito mencionado, existe um crescimento do fluxo da corrente sanguínea, o que resulta num aumento da temperatura (dissipação de calor convectivo, mensurável), que é registado pela câmara. Assim sendo, o algoritmo desenvolvido para a determinação dos níveis de *stress* é apenas baseado nesse dado de temperatura medido, assim como nos níveis de perfusão na vasculatura frontal da cabeça.

Na realidade, para cada sujeito ao estudo realizado, foi selecionada uma Região de Interesse (ROI, *Region of Interest*) que inclui os vasos frontais, e um algoritmo de rastreamento regista esta região, sendo que a computação apenas regista 10% dos "píxeis mais quentes".

Assim, para cada *frame*[7] da gravação térmica é computada a temperatura mediana de 10% dos "píxeis mais quentes"da ROI [7] e, finalmente, foi usado um modelo já existente para tratar a computação da corrente sanguínea nos vasos frontais, baseado na entrada dinâmica de calor.

2.4 Armazenamento e Manipulação de Dados

2.4.1 Definição de *Cloud*

Uma *cloud* pode ser definida como uma metáfora para descrever uma rede global de servidores, cada um com uma função específica, isto é, trata-se de vasta rede de servidores remotos, a uma escala global, interligados, que devem funcionar como um ecossistema único [8].

Os mencionados servidores foram concebidos para o armazenamento e gestão de informação, mas também para a execução de aplicações, ou mesmo para o fornecimento de conteúdos ou serviços.

O grande objetivo do conceito abordado é possibilidade de acesso a ficheiros *online*, tornando a informação disponível em qualquer local, em vez de estar limitada ao acesso exclusivo a partir de um computador, ou outro qualquer dispositivo de armazenamento [8]. Atualmente, diversas empresas já armazenam seguramente a sua informação *online*, sendo que existe uma diversidade de funcionalidades e interações entre as mencionadas opções. Alguns exemplos já existentes são a *Amazon Cloud*[8], a *Apple iCloud Drive*[9], ou a *Dropbox*[10], e cada um oferece ao seu utilizador um número de gigabytes diferente de armazenamento, e inúmeras possibilidades de interação.

A *Firebase*[11], no âmbito das *clouds*, é uma tecnologia que permite o desenvolvimento de aplicações *web*, que não requer qualquer tipo de programação, do lado do servidor, o que facilita e acelera o desenvolvimento, por parte do utilizador. Com o recurso a esta tecnologia, é possível verificar utilizadores, armazenar dados (através da base de dados própria, e gratuita caso as regras de escrita e leitura sejam definidas como públicas), implementar regras de acesso, entre muitas outras opções [9].

No que diz respeito à relação entre este tipo de rede de armazenamento de informação e a automação, em vários tipos de sistema existe a necessidade do armazenamento de informação *online*. O exemplo apresentado em seguida, demonstra a importância do armazenamento da informação *online*, num sistema de uma casa inteligente.

[7] Imagem fixa de uma gravação audiovisual

[8] Armazenamento em "nuvem", criado pela empresa Amazon, em 2011, e que permite o armazenamento de ficheiros, de uma forma segura, além de permitir outro tipo de interações também

[9] Cloud fundada pela empresa Apple, e que permite o acesso a documentos, a partir de diversos dispositivos, não só da empresa fundadora do serviço

[10] Serviço de armazenamento e de partilha de ficheiros, fundada em 2007 e, posteriormente lançada ao público em 2008

[11] É uma plataforma online e móvel de desenvolvimento, fundado pela empresa Firebase, em 2011, e posteriormente adquirida pela Google. Além do desenvolvimento, também tem uma nuvem, que permite realizar operações na cloud

2.4.2 Implementação de uma Casa Inteligente, com recurso à *Firebase*

O seguinte exemplo expõe um sistema de uma casa inteligente, energeticamente eficiente, com base numa rede de sensores sem fios. Para tal, foi usada a *Firebase Realtime Database* como base de dados, o que torna possível o acesso ao sistema da casa, a partir de qualquer local. A Figura 2.7 ilustra o diagrama da arquitetura do sistema:

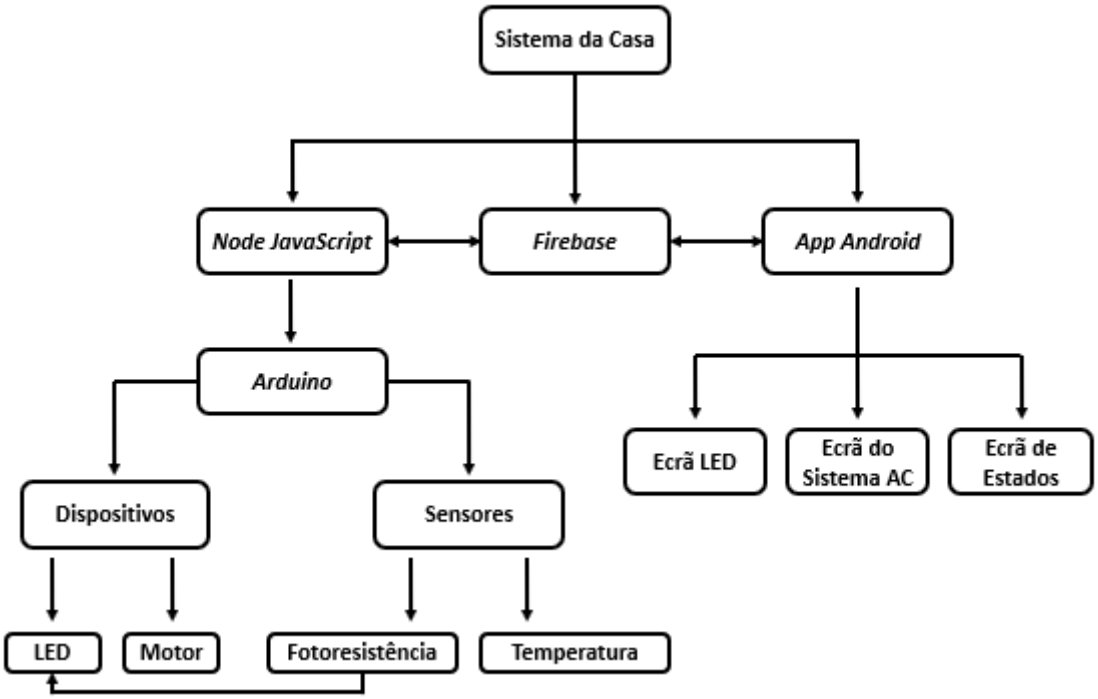

Figura 2.7: Diagrama da Arquitetura deste sistema[9]

Como está implícito no diagrama da figura anterior, este projeto necessita de uma aplicação móvel, que permita ao utilizador interagir com o sistema da sua habitação.

No momento em que se executa algum tipo de interação com a aplicação, a informação é transmitida à *Firebase*, que regista, e modifica os valores lógicos da ação. Depois das mudanças em relação à base de dados, existe um sistema *online* que, sempre que existe qualquer alteração, corre um *node.js*[12], que permite um código *JavaScript*, no servidor. A este sistema está ligado um *Arduino*, que regula os *LED's*(*Light-Emitting Diode*) e a ventilação da casa. A foto resistência e o sensor de temperatura conectados ao *Arduino* obtêm informação acerca das condições meteorológicas exteriores, sendo ainda responsáveis por manter a casa em condições adequadas.

Além do mais, a aplicação móvel ainda está ligada a outra aplicação, esta de previsão meteorológica, que automaticamente efetua alterações, baseadas nestas informações.

Por fim, os dados da foto resistência e do sensor de temperatura são também armazenados na base de dados. Dadas as imensas possibilidades de bases de dados a utilizar, a

[12]Interpretador de código *JavaScript*

selecionada para este sistema foi a da *Firebase* e esta escolha deve-se a vantagens intrínsecas a esta base de dados, comparativamente a outras:

- Reduz o tempo de desenvolvimento;
- É possível assumir que a *Firebase* vai tratar de cada uma das informações;
- Os dados são armazenados como JSON (*JavaScript Object Notation*)[13] nativo;
- Os dados estão protegidos, graças ao tipo de encriptação desta base de dados;
- Têm uma boa coordenação com sistemas como o Angular JS[14] [9].

2.5 Protocolos de Comunicação

A comunicação entre os componentes de um sistema deste tipo é fulcral para o bom funcionamento em todas as etapas do processo. Assim sendo, nesta secção serão abordados os protocolos responsáveis pela mencionada comunicação, e explicitado o seu funcionamento, em cada caso.

2.5.1 I2C

O protocolo I2C admite múltiplos *masters* e *slaves*, o que permite a existência de diversos controladores, num sistema. É composto por um barramento de 2 linhas bidirecionais e um *ground*, tratando-se de um protocolo simples e eficiente, no que diz respeito à troca de informação (Figura 2.8).

Além do mais, cada dispositivo conectado através deste protocolo de comunicação possui um endereço único, independentemente da função (*master*, ou *slave*) e pode operar, tanto como transmissor, como receptor.

Deste modo, este protocolo tornou-se fulcral em muitos tipos de sistemas (como de vigilância, ou de monitorização de fatores, entre outros), incluindo os que envolvem a utilização de um controlador como o *Raspberry Pi*. Muitos sensores compatíveis com este mini-computador comunicam com ele através deste protocolo, utilizando os pinos disponíveis (SDA, *Serial Data Line* e SCL, *Serial Clock Line*)[10].

[13] Linguagem de programação utilizada na troca de dados, de uma forma rápida, e simples
[14] Framework JavaScript, que ajuda na geração de aplicações single-page

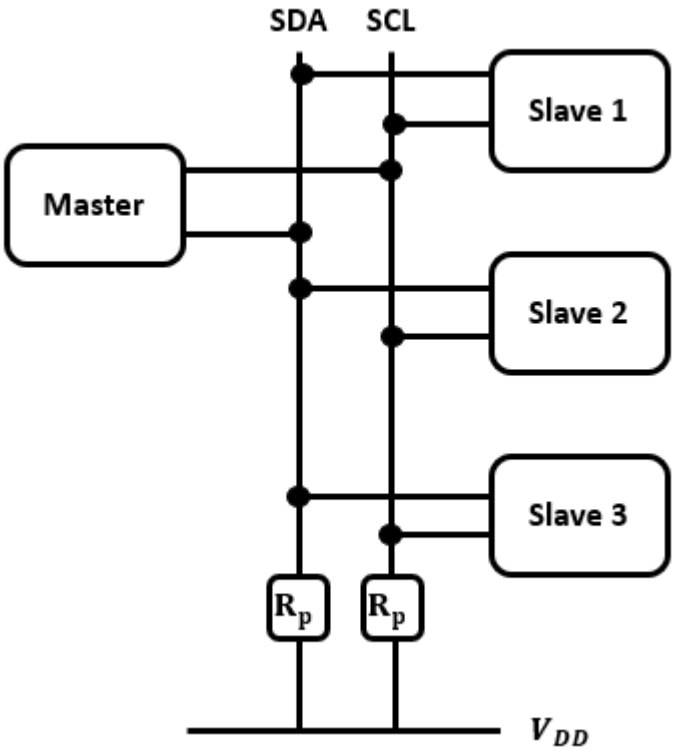

Figura 2.8: Exemplo de possível arquitectura do protocolo de comunicação I2C

2.5.2 1-*Wire*

O sistema de comunicação *1-Wire* foi concebido com o intuito de comunicar com sensores simples ou dispositivos de apenas uma *interface* de uma linha e, portanto, que consumam pouca velocidade e energia [11].

Uma das características importantes deste protocolo é a possibilidade de transferência de informação, entre um dispositivo e o *master* do sistema, enquanto que os outros conectados por este tipo de comunicação estão em "repouso"("*idle state*").

Por fim, existem dois tipos de alimentação possíveis de aplicação, aos *slaves* de um protocolo deste tipo:

- **Energia externa**: O *slave* possui um pino de energia, utilizado para a sua alimentação, no barramento;
- **Energia "Parasita"**: O *slave* é alimentado com recurso à energia fornecida pelo barramento de dados (Figura 2.9), e armazenada pelos condensadores internos dos dispositivos deste tipo.

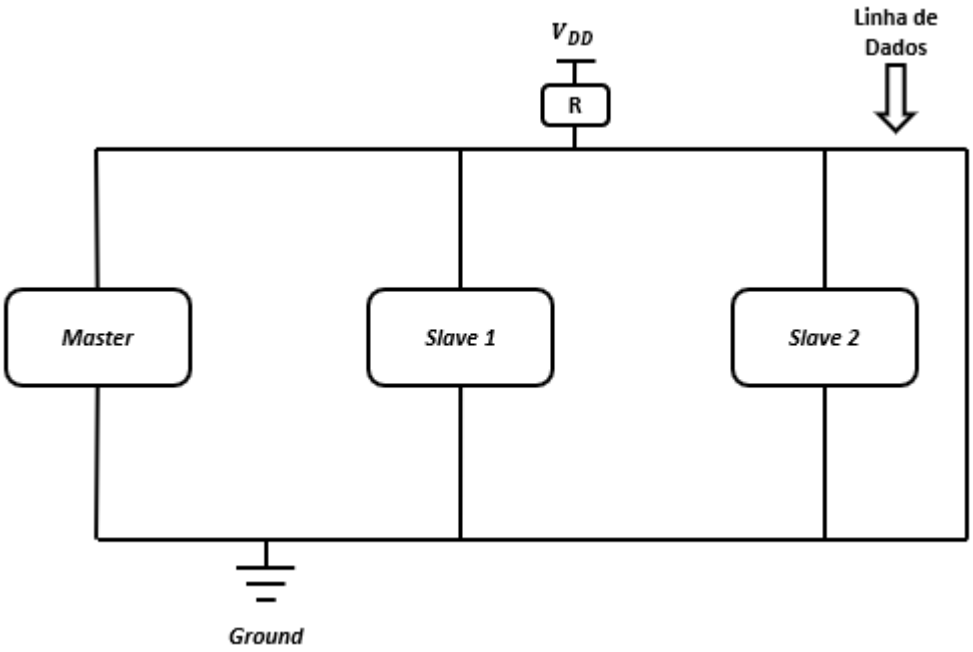

Figura 2.9: Exemplo de possível arquitectura do protocolo de comunicação 1-*Wire* [11]

2.5.3 *WebSocket*

A comunicação *WebSocket* é uma comunicação de *socket* único, *full-duplex* e, por isso, bidirecional (Figura 2.10). Através deste tipo de comunicação, o pedido HTTP (*Hyper Transfer Protocol*) torna-se num pedido único para a abertura de uma conexão *WebSocket*, e vai reutilizando esta mesma conexão para as comunicações, em ambas as direções (servidor -> cliente, cliente -> servidor).

Deste modo, a latência é significativamente reduzida, dado que, assim que é estabelecida uma conexão, o servidor pode enviar mensagens, logo que estas estejam disponíveis [12].

Em síntese, o protocolo *WebSocket* é composto por duas partes, sendo que a primeira é denominada por *handshake*, que consiste na mensagem do cliente e a resposta do servidor. A segunda parte do processo é a transferência de dados [13].

Vantagens

1. **Comunicação em tempo real mais eficiente** - Em comparação aos pedidos HTTP, o *WebSocket* poupa largura de banda, energia do CPU (*Central Process Unit*) e latência;

2. **Maior simplicidade** - As técnicas para a notificação, em tempo real, através de HTTP são de maior complexidade;

3. **Permite a implementação de outros protocolos, por cima do *WebSocket*** - Suporte à aplicação de protocolos de nível mais elevado, o que incentiva o desenvolvimento baseado em componentes reutilizáveis [13].

Pelo exposto, conclui-se que o protocolo HTTP não é o mais indicado para o desenvolvimento de aplicações, e tempo real, contrariamente ao *WebSocket*. Os padrões assíncronos, e de comunicação bidirecional, garantem a flexibilidade concedida pelos protocolos de transporte de camadas, na *Internet* [13].

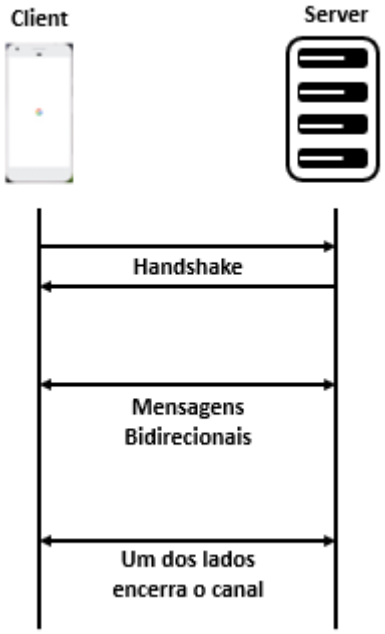

Figura 2.10: Arquitetura da comunicação *WebSocket*

Capítulo 3

Descrição do Sistema Proposto

No presente capítulo é apresentada uma descrição dos requisitos e funcionalidades, que são propostos para este sistema de monitorização volumétrica, recorrendo a sensores térmicos, bem como a enumeração das etapas a realizar para atingir os objetivos delineados.

Deste modo, serão abordados todos os componentes constituintes do sistema, como se conectarão entre si, como se processarão os dados ao nível da base de dados, e todas as funcionalidades disponibilizadas na aplicação móvel.

3.1 Objetivos Propostos

Tal como já foi mencionado anteriormente, atualmente existe um problema de mobilização de grandes massas populacionais, existindo uma necessidade de monitorizar informação auxiliar, num possível planeamento antecipado.

A rápida proliferação do conceito de IoT, no contexto atual, é um fator chave na aquisição deste tipo de dados e, com as ferramentas necessárias, é possível construir um sistema que adquira os dados necessários, e que posteriormente os coloque disponíveis a acesso remoto, para que possam finalmente ser interpretados e representados.

No caso em questão, dos transportes públicos sem lugar definido, existe uma sobrelotação diária dos recursos, em certos períodos do dia, ou em certas alturas, sendo que não existem ainda soluções que permitam uma previsão ou controlo das necessidades.

Pelo exposto, pretende-se implementar um sistema que adquira dados que possam verificar o preenchimento diário, numa região de interesse (neste caso, próximo das portas, como será mais detalhado, em seguida). Para a deteção da presença, isto é, ocupação, é utilizada um sensor de imagem térmica. Este está conectado a um controlador, que por sua vez está ligado a uma *cloud* que, em seguida, envia a informação em tempo real, para uma aplicação móvel.

No que diz respeito às funcionalidades da aplicação móvel, esta permitirá visualizar a informação em tempo real, assim como histórico armazenado, representações gráficas desse mesmo histórico e organizadas de acordo com relevância da informação. Além das interações relacionadas com os dados, também é possível parametrizar o sistema, conforme as necessidades.

Por fim, neste sistema, também é proposta a visualização física do estado atual de preenchimento da zona em torno de uma porta, pela representação sob a forma de *LED's*.

A tabela seguinte demonstra, em síntese, as etapas que são propostas, e as secções seguintes deste capítulo permitem um conhecimento mais detalhado, acerca do funcionamento do sistema e dos objetivos do mesmo.

Tabela 3.1: Procedimento de implementação do sistema

Etapas a Realizar	**Ferramentas Auxiliares**
Aquisição de Dados	Sensor Térmico e Sensor de Temperatura
Envio da Informação e Controlo dos Periféricos	Controlador do Sistema
Armazenamento	*Cloud*
Representação Gráfica	Aplicação Móvel e *LED's*

3.2 Funcionamento do Sistema

3.2.1 Controlador e Respetivos Periféricos

Considerando os objetivos propostos, e sintetizados na Tabela 3.1, pretende-se que o sistema esteja habilitado a recolher informação, e que a torne disponível,instantaneamente, na base de dados *online*.

Recorrendo às capacidades do controlador do sistema, é possível o envio da informação diretamente para uma *cloud*, recorrendo às bibliotecas existentes para bases de dados, e compatíveis com grande parte dos dispositivos que podem atuar como controladores, recorrendo a diversos exemplos de linguagens de programação.

Assim sendo, no que diz respeito aos componentes que fazem parte desta fase de implementação do sistema, o que se pretende é a eficaz leitura, e posterior envio dos dados térmicos recolhidos, tornando possível o conhecimento do estado de ocupação volumétrico, de uma certa área, num ambiente onde as condições de temperatura e humidade são controladas. No entanto, o controlo total do ambiente do espaço recomendado (carruagens de um comboio) pode ser de elevada dificuldade, podendo verificar-se pequenas flutuações que podem influenciar o bom funcionamento do sensor de imagem térmica. Para precaver este problema, é necessária a integração de um periférico habilitado ao controlo das condições de temperatura.

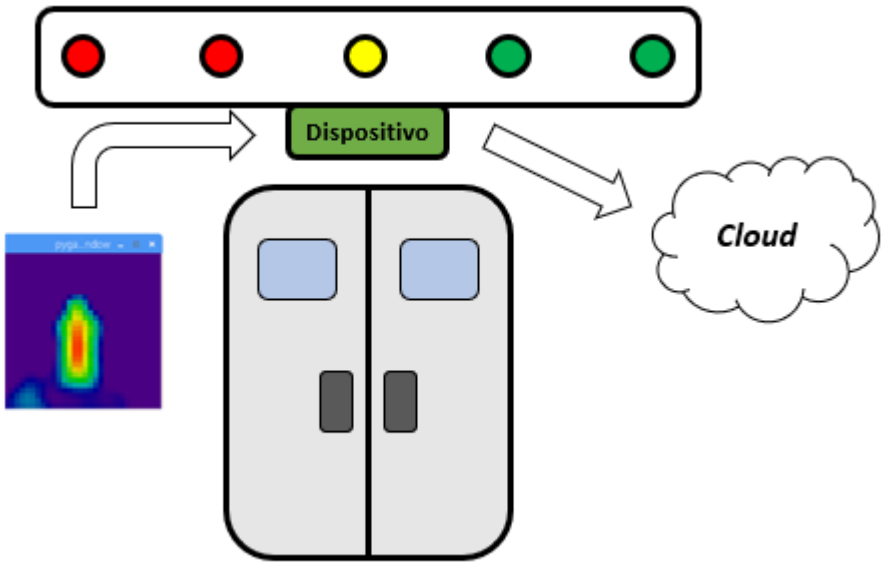

Figura 3.1: Objetivo para a componente física do sistema

A Figura 3.1 representa um possível contexto de utilização para o sistema que se pretende implementar, mais concretamente para a componente que adquire a informação, e que posteriormente a envia para a base de dados *online*. Tal como foi mencionado, as capacidades do controlador permitem que a sua comunicação seja bidirecional, recebendo e enviando a informação.Então, a Figura 3.2, esquematiza os *inputs* e *outputs* envolvidos, no controlador, durante a realização de um ciclo do sistema:

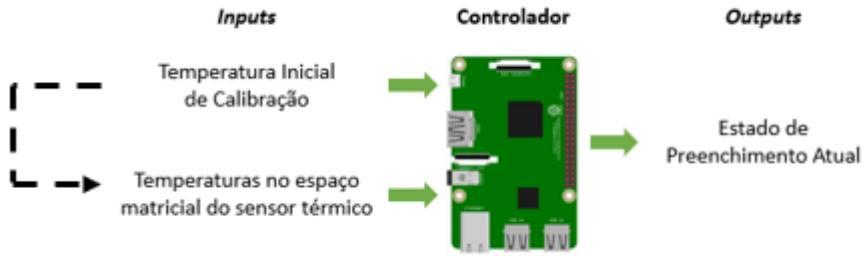

Figura 3.2: Diagrama de entradas e saídas projetado para o controlador

Pela inspeção da Figura 3.2, verificamos que existem duas entradas fulcrais ao bom funcionamento do sistema. A primeira, "Temperatura Inicial de Calibração", é a efetuada pelo periférico auxiliar na calibração e cuja função é a indicar ao sensor de imagem térmica em que intervalo de temperaturas deve operar. Com a disponibilidade deste dado, então é possível obter o segundo *input*, que são as temperaturas efetivamente no espaço de operação do sensor.

Por fim, a saída do sistema, o estado de preenchimento, é enviada para a base de dados que trata de armazenar a informação, nos locais indicados para tal. De salientar que toda a

aritmética relacionada com o cálculo percentual deve ser efetuada ao nível do controlador, para uma melhor eficiência do sistema, evitando a redução do desempenho dos módulos seguintes a esta etapa do processo, e permitindo a interação com os LED's do sistema.

3.2.2 Cálculo Volumétrico

O capítulo anterior abordou alguns conceitos relacionados com o funcionamento de câmaras térmicas e cálculo volumétrico, baseado em captação de imagem, sendo um dos principais objetivos desta dissertação a verificação de uma relação entre ambos.

Na realidade, a ocupação, no raio de acção do sensor de imagem térmica, pode ser determinado com base na imagem térmica adquirida, dado que a presença de um corpo será imediatamente captada, surgindo uma cor característica e identificativa da temperatura atual do corpo.

Considerando que o sensor aplicado neste sistema é composto por uma matriz de infravermelhos (mais detalhado no capítulo seguinte), a ausência de um corpo não irá concretizar alterações na imagem, visto que não existe reflexão aos IR (*Infravermelhos*) emanados pelo sensor.

Assim sendo, esta característica das câmaras térmica pode ser explorada como um método auxiliar de conhecimento da ocupação volumétrica. Através das capacidades de análise de imagem associados ao controlador do sistema, e às suas inúmeras bibliotecas, é possível identificar as diferenças na imagem, com base na identificação de cor do que é captado. Outro método alternativo será recorrer apenas aos valores de temperatura captados pelo sensor, pois é através destes que é construída a imagem térmica, baseado neles, algo que será detalhado no capítulo seguinte.

3.2.3 Armazenamento da Informação

A disponibilidade da informação é uma etapa essencial para o bom funcionamento deste sistema, em virtude da plataforma disponível para o acesso aos dados ser dependente da base de dados.

Para a implementação de uma aplicação de análise de dados, é fulcral a existência de uma base de dados que permita o acesso e, inclusive manipulação da informação, a partir de qualquer local, pelo que a solução passa pela utilização de uma *cloud*, tal como já foi mencionado anteriormente.

Como é exposto na Figura 3.1, deverá existir um envio direto da informação, com recurso ao controlador. Tal como foi efetuado na subsecção anterior, o diagrama seguinte (Figura 3.3) expõe, tanto as entradas, como as saídas, neste caso para a base de dados do sistema.

Figura 3.3: Diagrama de entradas e saídas projetado para a base de dados *online*

Além da informação volumétrica, também será necessário enviar os dados temporais, relativos à recolha da informação, de modo a que sejam organizados e selecionados, de acordo com as necessidades dos utilizadores. Os dados mencionados são armazenados sob a forma de uma árvore JSON, sendo acessíveis remotamente.

Como será mais detalhado nos capítulos seguintes, de acordo com o bom funcionamento deste tipo de sistemas de recolha de informação, deverão existir tipos de utilizadores, distinguidos pelos seus privilégios na aplicação e na manipulação da base de dados, para que a informação seja filtrada consoante necessidade de cada utilizador (administrador e utilizador regular), o que deverá ser verificado numa interação entre a *cloud* e a aplicação móvel.

3.2.4 Aplicação Móvel

O grande objetivo da criação da aplicação móvel, para este sistema é a visualização, em tempo real, da informação recolhida pelo sensor térmico, assim como o histórico desta, organizado de acordo com as necessidades, existindo rotinas que filtram a informação, pela sua relevância.

Uma aplicação móvel deve ser caracterizada pela sua simplicidade na interação com os utilizadores, bem como na sua implementação, para que o seu desempenho seja o mais eficiente possível e a exposição dos dados recolhidos deve facilitar a observação e a obtenção de conclusões.

Na realidade, no desenvolvimento de uma AM (Aplicação Móvel), devem ser consideradas duas componentes essenciais, *frontend* e *backend*. Neste caso, a primeira diz respeito ao que é visualizado pelo utilizador (*views*) e a segunda às operações que envolvem o processamento da aplicação ou qualquer interação realizada.

O histórico do informação deve estar organizado, de acordo com a data em que os dados foram recolhidos, assim como deve ser possível filtrar a informação, para a data que se pretende visualizar. Além do mais, a exposição da informação deve ser também de fácil e rápida visualização, tal como deve acontecer no histórico estatístico dos dados e, pelo exposto, a estatística associada ao sistema deve ser apresentada sob a forma de gráficos, imagens, isto é, componentes que simplifiquem a visualização do utilizador.

Assim sendo, a Tabela 3.2 contém as views da AM a implementar, de acordo com o tipo de usuário, bem como uma sucinta explicação das funções que cada uma deve desempenhar:

Tabela 3.2: Tabela com as views a implementar, ao nível da aplicação móvel

View	Função	Utilizador Regular	Administrador
Registo de Utilizador	Registo de cada utilizador, onde será necessário fornecer um endereço eletrónico válido, e uma palavra-passe	✓	✓
Login	Atividade de login, onde é efetuada uma comunicação com a base de dados para confirmação das credenciais introduzidas	✓	✓
Ecrã Principal	Página principal da aplicação, onde podem ser visualizados os dados referentes a cada comboio do sistema. Contém barra auxiliar para o acesso às restantes Views	✓	✓
Histórico	Composta por uma lista com todos os dados recolhidos pelo sensor e por um campo de pesquisa, editável pelo utilizador	X	✓
Área Gráfica	Contém uma View, em grelha, com várias opções gráficas, com informação filtrada, de acordo com a sua relevância	✓	✓
Ecrã de Iniciação	É constituído apenas por um ícone identificativo da aplicação	✓	✓
Definições	Configuração de parâmetros do sistema	X	✓
Detalhes do Utilizador	Alteração de parâmetros associados à conta de utilizador	✓	✓

As *views* expostas na Tabela 3.2 irão requer a constante comunicação com a base de dados, visto que a atualização da informação é constante. Deste modo, e tendo em conta que a AM apenas irá comunicar, diretamente, com a *cloud* (as parametrizações efetuadas com recurso à aplicação envolvem a base de dados), os seguintes diagramas ilustram as entradas e saídas que envolvem este componente do sistema, em primeiro lugar, para um utilizador definido como administrador do sistema e, depois, para um utilizador comum da aplicação:

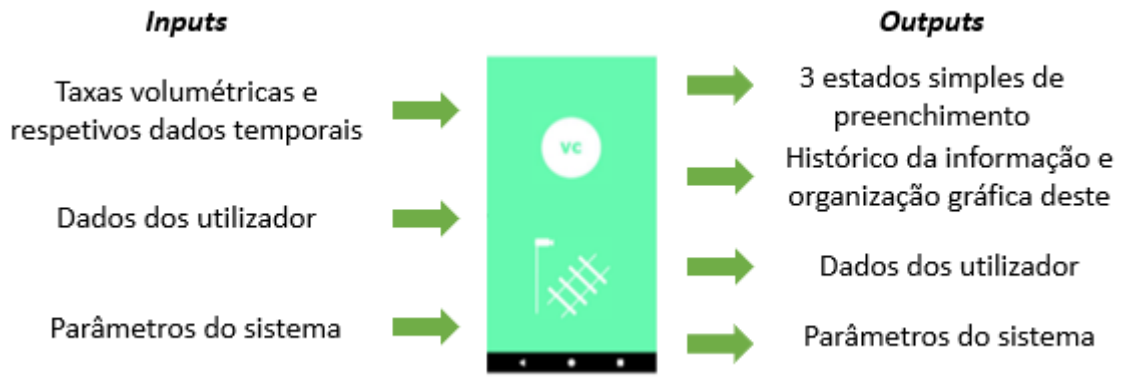

Figura 3.4: Diagrama de entradas e saídas projetado para a AM, de um utilizador classificado como administrador

Pela observação do diagrama da Figura 3.4, deve ser possível, para um administrador, configurar parâmetros do sistema e dados e dados associados à sua conta remotamente. Apenas a segunda opção é possível também para um utilizador regular, tal como acontece relativamente ao seu acesso aos dados do histórico da informação e gráficos associados.

Assim sendo, em seguida é apresentado o mesmo diagrama (Figura 3.5) , para um utilizador classificado como normal:

Figura 3.5: Diagrama de entradas e saídas projetado para a AM, de um utilizador classificado como normal

3.2.5 Arquitetura Proposta

Nas secções anteriores, foram apresentados, e sucintamente detalhados, os diversos módulos constituintes do sistema, a sua função, o objetivo pretendido para cada, e é então de relevância mencionar como todos estes blocos se interligam entre si, e qual a comunicação que estabelecem.

Assim sendo, a arquitetura projetada para este sistema de controlo volumétrico é a apresentada na Figura 3.6 (inclui todos os componentes):

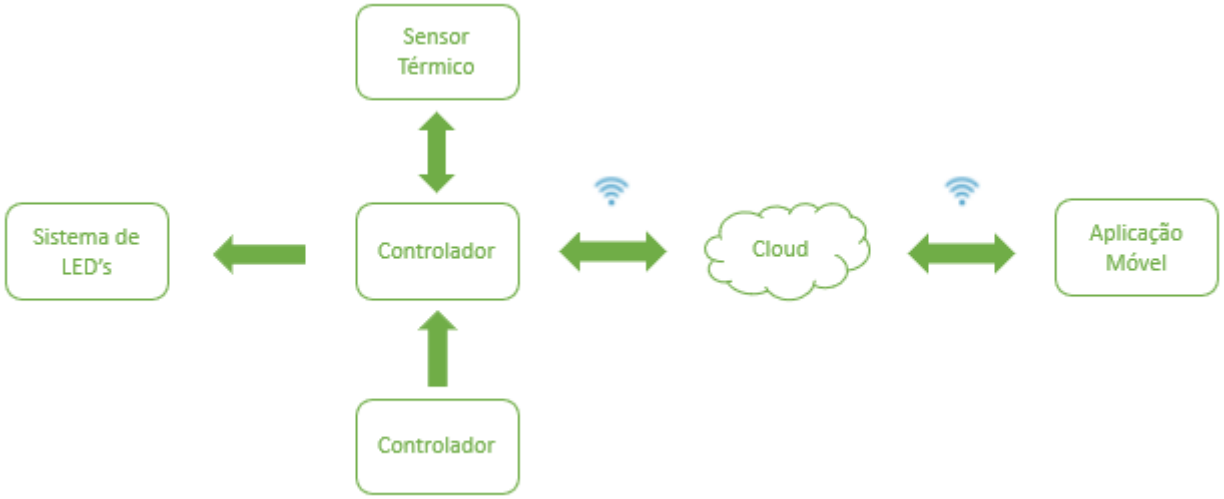

Figura 3.6: Arquitetura Proposta para o Sistema

Os componentes centrais das comunicações existentes, durante qualquer tipo de interação do sistema, são o controlador, a *cloud* e a AM. Todos estes componentes comunicam em ambos os sentidos (bidirecional) entre eles, sendo a base de dados o elemento que conecta a aplicação móvel e o controlador. Mesmo ao nível dos parâmetros do sistema, qualquer alteração efetuada envolverá uma comunicação com a base de dados, que tratará de fornecer essa informação ao controlador, que a tratará da devida maneira, excepto no que diz respeito ao módulos de LED's.

3.2.6 Requisitos não-funcionais

Durante o desenvolvimento e implementação de um sistema deste tipo, pretende-se que o protótipo arquitetado seja funcional e eficaz, pelo que se pretende que a sua instalação seja acessível, e que seja o mais adaptável possível ao meio para o qual foi concebido. Além do mais, apesar do controlo do ambiente do meio, o dispositivo deve ter a capacidade de se adaptar a pequenas variações que possam eventualmente surgir.

No que diz respeito às aplicações digitais constituintes do sistema, deve ter-se em consideração o tipo de público a que se destinam. Neste caso, sendo um público geral, com variadas faixas etárias, as aplicações devem ser simples e concretas, para que a visualização da informação seja o mais abrangente possível.

A aplicação desenvolvida também deve ser concisa nos dados a filtrar, tornando a pesquisa de histórico também precisa. Uma boa acessibilidade à informação permitirá uma rápida procura de soluções e correção de eventuais necessidades verificadas.

Por fim, além dos requisitos relacionados com a informação a apresentar, devem também ser implementadas rotinas relacionadas com a segurança da aplicação móvel (registo de utilizadores, *login*, alteração de parâmetros de registo). Como já foi mencionado, devem existir dois tipos de utilizadores, com acessos distintos a certas áreas da AM e que

terão associados dados introduzidos, durante o processo de registo. Assim, devem ser implementadas rotinas para a alteração de qualquer parâmetro associado a este registo, para precaver qualquer necessidade do utilizador, especialmente relacionadas com a palavra-passe de segurança, também esta definida pelo utilizador, que deve ser possível alterar, seguramente. O esquema seguinte (Figura 3.7) explícita como deve ser tratado o processo de arranque da aplicação, com as rotinas descritas ao longo deste parágrafo:

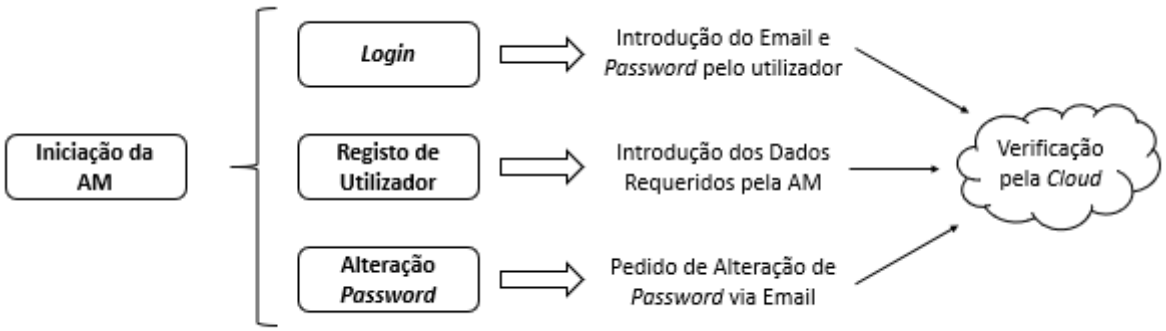

Figura 3.7: Rotina de Segurança dos Utilizadores

3.3 Análise do Sistema Proposto

Em síntese, após a análise das secções anteriores, conclui-se que o sistema implementado se encontra dividido em três componentes principais: o controlador (com os seus periféricos), a base de dados *online* e as aplicações de exposição da informação.

Desta forma, o controlador é o responsável por controlar os dispositivos envolvidos na aquisição da informação, e pelo tratamento desta, sendo possível a sua interligação com um sistema de armazenamento exterior. É também o controlador que trata da ativação dos *LED's*, pelo que, para tal, não é necessária nenhuma instrução exterior, proveniente da base de dados (sistema independente, pois em caso de alguma eventualidade que envolva a *cloud*, o sistema físico pode continuar em operação).

Em segundo lugar, a informação relevante é enviada para a base de dados, que a tornará acessível para todas as aplicações que pertencem ao sistema. Na *cloud*, a informação estará organizada de forma acessível, com o objetivo de facilitar a implementação da exposição dos dados. A comunicação estabelecida entre a base de dados e a aplicação móvel é bidirecional, permitindo a alteração de parâmetros de funcionamento, restritamente, e dados associados aos utilizadores.

Finalmente, a aplicação móvel permite a visualização imediata da informação, pelos utentes registados, e o auxílio na demonstração de padrões de utilização, pela análise estatística implementada.

Capítulo 4

Implementação do Sistema

O objetivo do presente capítulo é descrever o procedimento efetuado para a implementação do sistema, mais especificamente para a elaboração do protótipo projetado e posterior interface de utilizador, para a sua utilização em transportes públicos de carruagens, como auxuliar no controlo da ocupação das entradas / saídas.

Deste modo, serão abordadas todas as tecnologias utilizadas, assim como técnicas de auxílio à implementação, algoritmos gerados e linguagens de programação utilizadas.

Em síntese, este capítulo aborda todos os aspetos relacionados com o controlador, com o respetivo sensor térmico e os restantes periféricos, a base de dados associada e a respetiva aplicação móvel, baseada nos dados enviados pelo controlador e diretamente conectada à base de dados. Além do mais, e com o intuito de demonstrar o princípio de funcionamento do sistema, foram ainda realizados alguns testes, pelo que são também apresentados neste capítulo.

4.1 Instalação dos Dispositivos

Nesta secção, são descritos, pormenorizadamente, dois componentes fulcrais do sistema, o dispositivo utilizado como controlador, o *Raspberry Pi 3 Model B+* e o sensor *AMG8833*, fabricado pela *Panasonic*. Também serão abordados os restantes periféricos constituintes do sistema, o sensor de temperatura auxiliar e os LED's.

A utilização destas tecnologias deve-se essencialmente ao facto de ambas estarem disponíveis a um preço acessível, além da qualidade demonstrada na aquisição dos dados, e da possibilidade de interação entre os periféricos e o controlador, e entre este e a base de dados *online*.

Pelo exposto, esta secção está estruturada, em primeiro lugar, pela exposição do material utilizado, seguida do esquema de montagem e ligação de ambos os componentes

referidos. Por fim, é efetuada a descrição dos algoritmos utilizados para a implementação do controlador e respetiva interligação com os restantes módulos do sistema.

4.1.1 Material Utilizado

Controlador do Sistema

O controlador deste sistema, o *Raspberry Pi 3 Model B+*, é um computador de dimensão reduzida, como o seu custo, que foi desenvolvido pela fundação *Raspberry Pi* . O grande objetivo associado à sua criação foi a possibilidade de oferecer uma ferramenta de teste de capacidades computacionais, permitindo que os seus utilizadores melhorem as suas capacidades de desenvolvimento em linguagens de programação, como por exemplo o *Python* [14].

Assim, para uma melhor compreensão das características associadas a este controlador, em seguida está exposta a Tabela 4.1, onde são enunciadas as mais relevantes deste computador:

Tabela 4.1: Descrição das principais caraterísticas do *Raspberry Pi*

Componente	**Características**
Processador	Broadcom BCM2837B0, com 4 núcleos e opera a 1,4 GHz
Periféricos de Comunicação	Gigabit Ethernet (Via USB, Gigabit/s), Dual-Band Wi-Fi LAN (transita entre 2,4 GHz e 5GHz)
RAM	RAM & 1GB LPDDR2 SDRAM
Bluetooth	Bluetooth 4.2 e BLE (Bluetooth Low Energy)
Armazenamento	Micro SD
GPIO	40 portas programáveis de entrada e saída de dados
Portas	HDMI, conector video-audio analógico de 3,5mm, 4 entradas USB 2.0, CSI (Camera Serial Interface) e DSI (Display Serial Interface)

De salientar que a dimensão deste dispositivo confere inúmeras vantagens, como a sua portabilidade, ocupando 82mm de comprimento, 56mm de largura e 19,5mm de altura, associados a um peso de 50g.

Outro aspeto a realçar é a configuração da cabeça de 40 pinos que compõe o *Raspberry Pi 3 Modelo B+* e que está ilustrada na figura que se segue (Figura 4.1) :

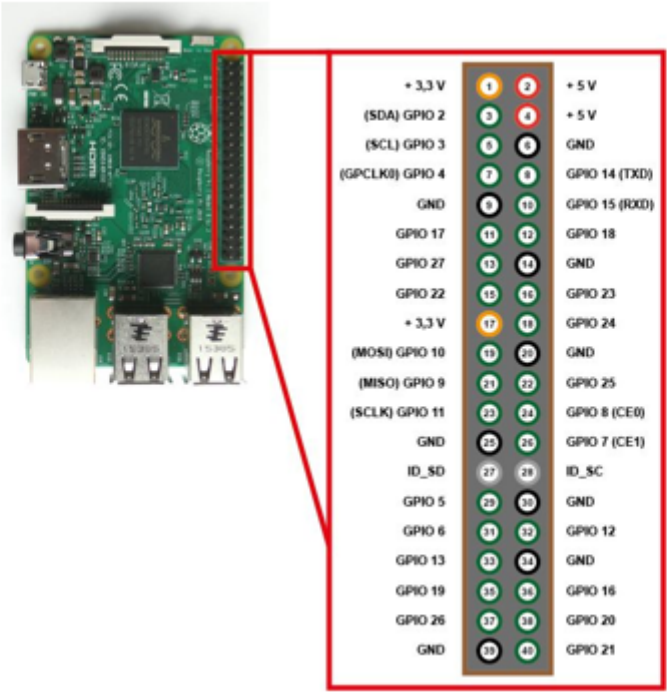

Figura 4.1: Configuração da cabeça de pinos de entrada e saída do Raspberry Pi 3 Modelo B+ [15]

AMG8833

No que diz respeito ao sensor, o *AMG8833* é um sensor térmico, composto por uma matriz 8x8 de sensores térmicos de infravermelhos. Quando ligado ao *Raspberry Pi*, este sensor retorna 64 leituras de temperatura, via I2C (*Inter-Integrated Circuit*). É capaz de efetuar leituras entre os 0ºC e os 80ºC, com uma incerteza associada de (+/-) 2,5ºC. Nas condições ideais de luminosidade, é capaz de detetar um ser humano a cerca de 7m e regista/processa imagens a uma frequência de 10Hz.

Para efetuar a comunicação necessária entre o sensor e o controlador, foi utilizada a linguagem de programação *Python*. Numa perspetiva de auxílio no processamento de imagem, foi utilizada a biblioteca *SciPi*, do *Python* (biblioteca *open-source*, cuja biblioteca central é o *NumPy*). Assim, é possível interpolar a matriz 8x8 e obter melhores resultados.

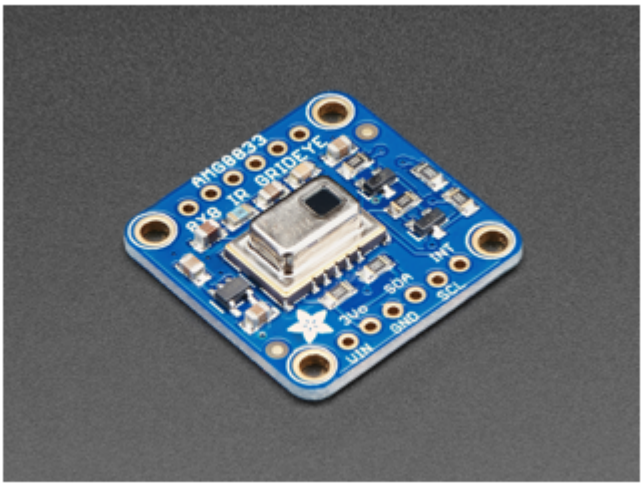

Figura 4.2: Sensor *AMG8833* [16]

O sensor é composto por doze pinos, seis na sua parte superior e seis na inferior, como pode ser verificado na Figura 4.2. Os pinos superiores apenas conferem uma maior estabilidade mecânica ao sensor e apenas a fileira inferior é conectada ao controlador. Nesta sequência de pinos, apenas três são lógicos, sendo os restantes de carácter eletrónico [16].

Na Tabela 4.2 está uma sucinta explicação da função de cada um dos pinos da fileira inferior do sensor:

Tabela 4.2: Legenda dos pinos do sensor AMG8833 [16]

Carácter	**Nomenclatura**	**Função**
Eletrónico	Vin	Pino de energia. Sendo que o sensor utiliza 3.3V, a placa do sensor contém um regulador de tensão que recebe entre 3-5V de tensão contínua e converte-os
	3Vo	A saída de 3.3V do regular de tensão
	GND	Terra
Lógico	SCL	Pino de clock do barramento I2C
	SDA	Pino de dados do barramento I2C
	INT	Pino utilizado para detetar movimentos e alterações, no campo de visão do sensor

DS18B20

Este sensor de temperatura mede temperaturas (9-bit a 12-bit) e comunica através do protocolo de comunicação *1-Wire* que, como o próprio nome indica, requer apenas uma linha de dados e um *ground* para a comunicação com o microprocessador. Além das caraterísticas de transmissão de dados, este sensor é alimentado através do barramento de dados, eliminando a necessidade de alimentação externa (conceito de "energia parasita")

[17].

Por fim, na seguinte tabela (4.3) estão expostos todos os componentes utilizados, durante a implementação deste protótipo:

Tabela 4.3: Tabela do material utilizado durante o desenvolvimento

Componente	Número de Unidades Utilizadas
Raspberry Pi 3 Model B+	1
Cabos de ligação NSR macho-fêmea	13
Sensor Adafruit *AMG8833 Grid-Eye*	1
Micro SD PNY 16GB	1
BreadBoard	1
Termómetro DS18B20	1
Resistência de filme de carvão, de 10kΩ, 0.25W ±5%	1
Resistência de filme metálico, de 330Ω, 0.6W ±1%	3
LED, 4.85mm, verde, 2.9V-3.6V	1
LED, 4.85mm, vermelho, 1.8V-2.6V	1
LED, 4.85mm, amarelo, 1.8V-2.6V	1

4.1.2 Configuração Inicial do Controlador

Em primeiro lugar, com o objetivo da criação da interação entre o controlador e o sensor térmico, foi necessária a instalação do sistema operativo, no controlador do sistema.

Deste modo, procedeu-se à instalação do sistema operativo a operar no cartão SD e, para proceder a uma instalação otimizada deste, o *Raspberry Pi* foi inicializado através do software *NOOBS*[1], que promove uma instalação facilitada do *Raspbian* (o sistema operativo oficialmente suportado pela fundação *Raspberry Pi*).

4.1.3 Instalação das Bibliotecas Necessárias para a Interação com o Sensor

Para que seja possível a interação entre o controlador o sensor que adquire a informação térmica, foi necessário instalar, no *Raspberry Pi*, diversas bibliotecas essenciais à comunicação, através de uma das linguagens de programação mais utilizadas atualmente a nível global, *Python*.

Python é uma linguagem de programação de alto-nível, orientada a objetos, cuja semântica é dinâmica, e é muito associada ao rápido desenvolvimento de aplicações. Além da sua sintaxe simples, é também uma linguagem que permite uma implementação organizada, o que reduz drasticamente os custos de manutenção de um programa. Além do mais, suporta também inúmeros pacotes e bibliotecas, o que o torna um programa completamente configurável, à preferência do seu implementador. Diversas bibliotecas, inclusive a padrão, e o respetivo interpretador, estão disponíveis gratuitamente, o que ainda mais promove a utilização do *Python* [18].

[1] Instalador de sistema operativo de utilização facilitada e que, além do Raspbian, ainda permite a instalação de sistemas operativos alternativos

No caso deste sistema, e apesar de existirem diversas variantes, foi utilizado o *CircuitPython*[2], que permite a interação, em *Python*, com circuitos e periféricos, tal como o AMG8833, e assim como outros tipos de sensores, motores, LED's, entre outros exemplos.

Após a instalação da mais recente versão disponível do sistema operativo mencionado anteriormente, o *Raspbian*, e respetiva instalação do *Python 3*, foi necessário, para a instalação das bibliotecas do *CircuitPython*, habilitar os protocolos I2C e SPI.

4.1.4 Esquema Elétrico da Montagem

As conexões estabelecidas entre os periféricos e o controlador estão demonstradas na Figura 4.3, com excepção das associadas aos LED's (Figura 4.4):

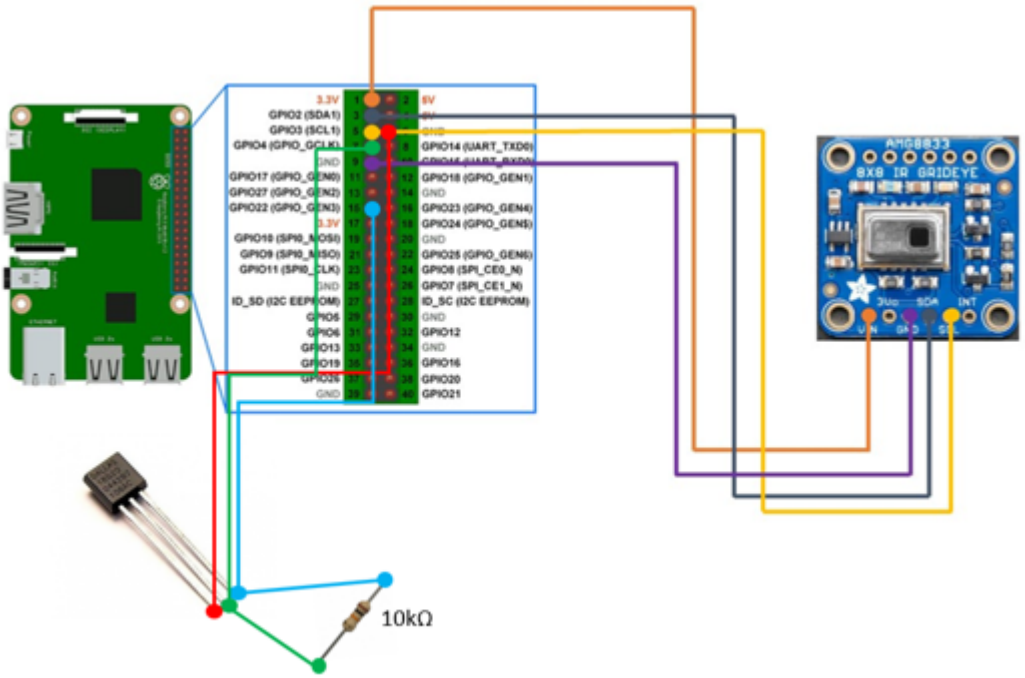

Figura 4.3: Esquema de ligação física dos periféricos de aquisição de informação, ao *Raspberry Pi* [16]

Em primeiro lugar, a alimentação, de 3.3V, é ligada a Vin. Em seguida, é realizado o mesmo procedimento para os pinos de *clock* (SCL), e de dados (SDA), do protocolo I2C e, por fim, o *ground*.

Depois de configurado o sensor, e dada a necessidade da existência de um controlador da temperatura, à medida que são recolhidos os dados, instalou-se o sensor DS18B20. Para tal, como expõe da Figura 4.3, existe a necessidade de instalação de uma resistência de 10kΩ (Tabela 4.3).

Além destes componentes que adquirem informação, existe também o sistema de LED's, que expõe o estado de ocupação, de acordo com os valores que lhe são fornecidos

[2] Adiciona suporte de hardware às inúmeras funcionalidades do Python, isto é, possibilita a utilização de placas com microcontrolador

pelo controlador e, para a sua instalação, foram necessárias também algumas conexões físicas, expostas na figura seguinte:

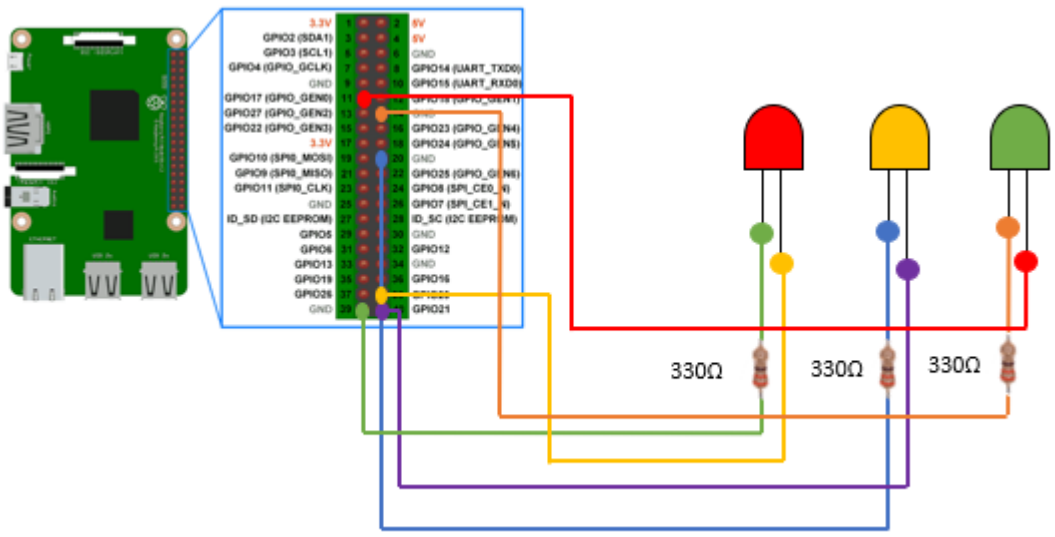

Figura 4.4: Esquema de ligação física dos periféricos de exposição da informação, ao controlador

Cada LED requer também uma resistência, de 330Ω, na sua conexão ao controlador, em virtude do *Raspberry Pi* apenas puder fornecer uma corrente de, aproximadamente, 60mA e na falta desta, pode verificar-se um excesso, comprometendo a segurança do controlador e dos restantes periféricos . A interligação entre cada cada componente deste tipo deve ser efetuada como indica a Figura 4.4, salientando que o maior conector de cada LED (ânodo) está ligado ao lado positivo da fonte de alimentação e o de dimensão mais reduzida (cátodo) ao lado negativo, garantido a correta polarização [19].

Assim sendo, e finalizada a ligação entre o controlador e os respetivos periféricos, procedeu-se então ao teste inicial de funcionamento, que verifica que todas as etapas da instalação foram concluídas, com sucesso, assim como se todas as conexões foram corretamente estabelecidas.

4.1.5 Teste Inicial de Funcionamento do Sistema

Com o objetivo de verificar se todas as ligações e configurações efetuadas, tanto as que incidiram sobre o controlador, como as que envolveram o sensor de imagem térmica, foi então realizado um teste, para que seja possível visualizar o funcionamento inicial. Além do mais, estas leituras serão apresentadas sob a forma de uma matriz 8x8 e, neste caso de teste, a programação do DS18B20 é desprezada.

Para tal, o seguinte diagrama expõe a sequência dos eventos que foram considerados:

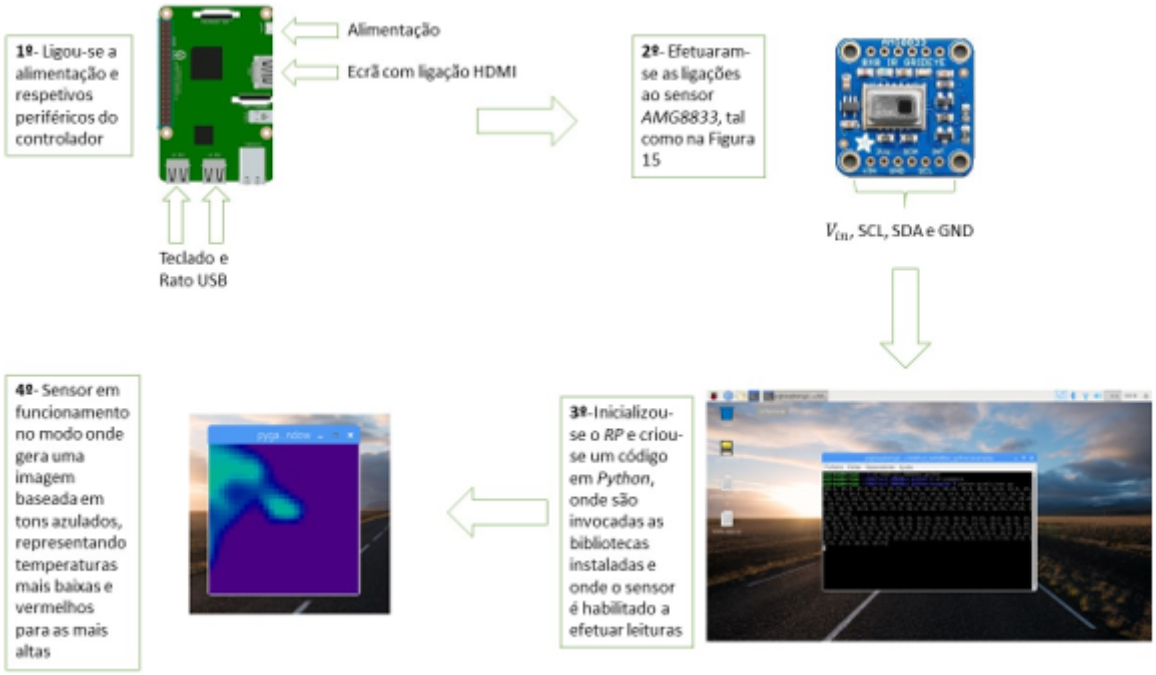

Figura 4.5: Diagrama de procedimento para a realização do teste inicial de funcionamento do sistema

Os dois primeiros eventos do diagrama de sequência representado na Figura 4.5, dizem respeito, única e exclusivamente, a ligações físicas, as mesmas expostas anteriormente (Figura 4.3 e Figura 4.4).

Por outro lado, os eventos seguintes já envolvem a aplicação de algumas instruções sobre o controlador, de modo a que este interaja com o sensor, e solicite o seu funcionamento, em primeiro lugar, obtendo leituras de temperaturas e, em segundo lugar, fornecendo uma imagem baseada nos mesmos valores de temperatura medidos. Estas instruções estão presentes no Anexo III, que contém o algoritmo programado para o controlador.

Numa primeira instância, a criação da imagem é baseada nos valores dos píxeis que são lidos pela matriz de sensores IR, do *AMG8833*. No entanto, sendo esta uma matriz 8x8, e com o objetivo de obter melhores resultados, efetuou-se uma interpolação bicúbica, redimensionando assim a imagem. Além do mais, existem ainda outros parâmetros que podem ser alterados, como o intervalo térmico de leitura, ou o espetro de cores.

4.2 Programação do Sistema

Na presente secção são apresentadas as diversas etapas, desde o arranque do sistema, até à interação com a aplicação móvel, detalhando os motivos pelos quais foram algumas opções tomadas, e também conduziu à utilização algumas ferramentas de suporte à implementação.

Para o conteúdo em seguida descrito, é considerado que a etapa inicial de todo o

processo foi a conexão do controlador à fonte de alimentação, respetivos periféricos e ao sensor térmico.

Em síntese, o seguinte diagrama (Figura 4.6), em notação BPMN 2.0 (*Business Process Model and Notation 2.0*) demonstra a sequência de eventos, entre a recolha de informações pelo sensor e consequente passagem de informação para a base de dados:

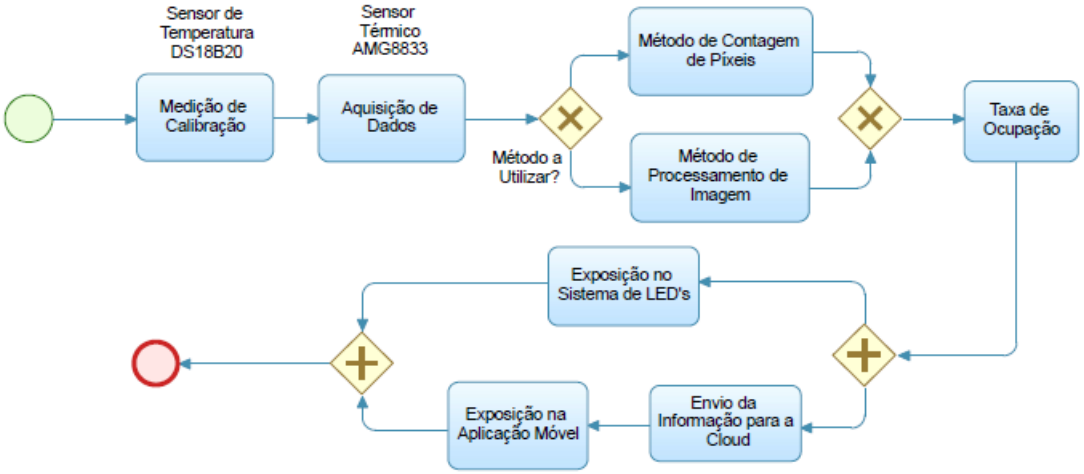

Figura 4.6: Diagrama BPMN de um ciclo do sistema

4.2.1 Leitura de Valores de Temperatura e Representação Gráfica

Deste modo, são então apresentados, em seguida, com detalhe, os processos que envolveram a programação dos sensores, e o respetivo controlo destes, pelo dispositivo controlador, o *Raspberry Pi*.

Matriz de Valores de Temperatura

Em primeira instância, tal como já foi mencionado, a linguagem de programação *Python* é a grande responsável pela interação existente entre o *Raspberry Pi* e o *AMG8833*.

De facto, graças a esta linguagem de alto-nível, é possível ter acesso a uma extensa biblioteca de suporte e também ter um bom desempenho, do ponto de vista da velocidade, pela produtividade do *Python 3*.

Assim sendo, o sensor térmico efetua uma leitura de valores de temperatura, num espaço matricial 8x8 e, com recurso à biblioteca do fabricante do *AMG8833*.

Para que os resultados das leituras sejam interpretados, e respetivamente armazenados (na forma matricial), foi essencial a utilização das propriedades matemáticas incluídas na biblioteca original do Python 3 e o pacote *Numpy*, permite o acesso a operações que envolvem matrizes e vetores [20].

Construção da Imagem Térmica

O processo de representação da informação, sob a forma de uma imagem de propriedades térmicas, está relacionado com as leituras efetuadas pelo sensor.

Em primeiro lugar, sendo a informação recolhida como uma matriz, o que já foi clarificado em capítulos anteriores,foi definido previamente o espetro de cores que é possível visualizar(1024) e parametrizou-se o sistema para mostrar informação numa janela, de dimensão definível.

A temperatura ambiente pode ser relativamente variável, apesar do controlo existente sobre o meio, mas fatores externos como a temperatura ou do número de utilizadores do espaço podem causar pequenas variações. Portanto, é essencial que o sistema se caracterize pela sua adaptabilidade.

A apresentação da imagem (Figura 4.7), requer, novamente, a utilização das bibliotecas do *Python*. Neste caso, com recurso à biblioteca *Pygame*[3] (*Open Source*) [20], e as suas ferramentas, é possível desenhar a imagem e, através de uma interpolação bicúbica, aumentar o seu tamanho. Além do mais, esta biblioteca ainda permite atualizar o ecrã, periodicamente. Através da função *pygame.draw.rect()*, é possível configurar a imagem, em função dos parâmetros de entrada, que neste caso são a superfície, a cor, o retângulo a desenhar e a largura [21].

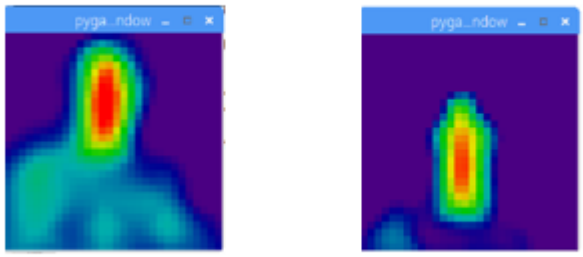

Figura 4.7: Exemplos de atividade da câmara, no mesmo contexto, alterando os parâmetros do algoritmo de calibração

4.2.2 Rotina de Auto Calibração da Câmara Térmica

Como já foi anteriormente mencionado, as medições efetuadas pelo AMG8833 estão dependentes das condições do meio onde são efetuadas, pelo que existem fatores externos ao sistema que podem condicionar o seu bom funcionamento.

[3]Biblioteca que é utilizada em conjunto com Python, normalmente associada à implementação de jogos multiplataforma

Assim sendo, com o objetivo de reduzir ao máximo a influência destes fatores, foi então construída uma rotina, também em *Python*, que permite uma melhor calibração e adaptação do sensor ao meio em questão.

Um fator de extrema influência é a temperatura atual no meio e, para tal, na rotina referida no parágrafo anterior, é efetuada uma primeira medição, pelo DS18B20.

Em seguida, a partir deste valor de temperatura, já é possível definir os limites de representação da imagem térmica, efetuando um cálculo matemático, auxiliado por constantes, cujos valores foram previamente estudados.

Influência do DS18B20

O sensor de temperatura *DS18B20* foi utilizado neste sistema com o objetivo de auxiliar o sensor térmico na sua parametrização, de acordo com as condições do meio. Este sensor, tal como o *AMG8833*, comunica com o controlador através dos pinos de I/O (*Input/Output*).

Deste modo, e recorrendo às funcionalidades do *Python*, foi implementada a função de leitura de temperatura inicial mencionada, por este sensor, e, cada vez que um ciclo do programa é executado, é efetuada uma nova leitura, que garante a adaptação a possíveis alterações, que possam afetar a qualidade no cálculo da taxa de ocupação.

4.2.3 Alcance do Sensor

O primeiro teste a realizar está relacionado com a área de deteção do sensor, em função da altura. Neste teste, ir-se-á verificar se existe perda de qualidade a informação, aumentando a altura, mantendo as condições ambiente constantes.

Não obstante, efetuando uma análise ao documento de especificações do sensor, verificamos que o limite de deteção do sensor é, no máximo, de 5 metros, sendo variável, tendo em conta a tensão de alimentação fornecida pelo controlador (5V ou 3,3V são as hipóteses em causa).

Além do mais, neste documento acerca do sensor ainda consta a informação de que o ângulo de visão do sensor é de 60º, tendo como eixo de referência o que intersecta verticalmente o centro do sensor, tal como demonstra a Figura 4.8 [22]:

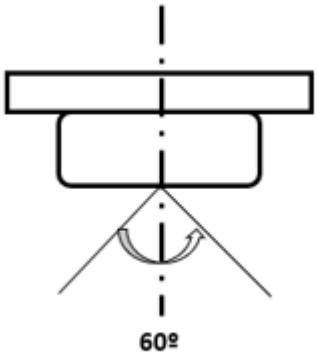

Figura 4.8: Campo de visão do sensor

4.2.4 Análise dos Dados Adquiridos e Cálculo da Ocupação

A informação que se pretende obter, no caso deste sistema, é a ocupação de um dado espaço, a partir da informação recolhida pelo sensor e através de um algoritmo matemático para o cálculo da taxa de ocupação.

Para a obtenção da mencionada taxa de ocupação, foram implementadas duas alternativas, ambas em *Python*, e que envolvem abordagens completamente distintas. Um dos métodos está relacionado apenas com os valores obtidos na matriz 8x8 do sensor, enquanto que o segundo implica o processamento da imagem obtida. O método a utilizar para a análise da informação obtida está dependente da decisão do administrador do sistema, o que pode ser realizado na AM. Este processo é possível, pois está definido um parâmetro para tal, na FRD (*Firebase Realtime Database*) que corresponde ao modo de funcionamento ("Normal", ou "Imagem").

Em seguida, estão apresentadas as implementações de ambos os métodos, bem como os resultados de alguns testes comparativos efetuados.

Método de Contagem de Pixéis

Este algoritmo, referente à taxa de ocupação do espaço, no campo de visão do sensor baseia-se nos valores recolhidos de temperaturas para verificar a presença de um corpo, dentro do espetro térmico visível. As etapas, desde a leitura dos valores, até à obtenção da ocupação estão representadas no esquema da Figura 4.9:

Figura 4.9: Algoritmo desenvolvido, para este método

Em primeiro lugar, após a inicialização do sistema e respetiva autocalibração, as leituras são armazenadas num vetor, que será verificado, e cujos valores são individualmente comparados com o valor mínimo de temperatura definido, isto é, são verificados os píxeis que não estão ocupados. Todos os valores que cumpram este requisito, são armazenados num vetor auxiliar, com dimensão em seguida contabilizada.

Por fim, para obter a taxa de ocupação:

$$taxadeocupao = 100 - (\frac{dimensodovetorauxiliar}{64} * 100) \tag{4.1}$$

Onde o denominador da fração da equação 4.1, 64, é o total de espaços da matriz de valores (8x8).

Método de Processamento de Imagem

O método de processamento de imagem, em comparação com o anterior, envolve uma maior capacidade de processamento do ponto de vista do controlador, visto que envolve a representação da imagem, numa matriz, tal como foi explicado anteriormente, e respetiva análise e processamento, através das bibliotecas do *OpenCV* existentes para *Python*. Na Figura 4.10, é possível verificar quais as opções tomadas, para a implementação deste algoritmo que retorna, como o método anterior, uma taxa de ocupação:

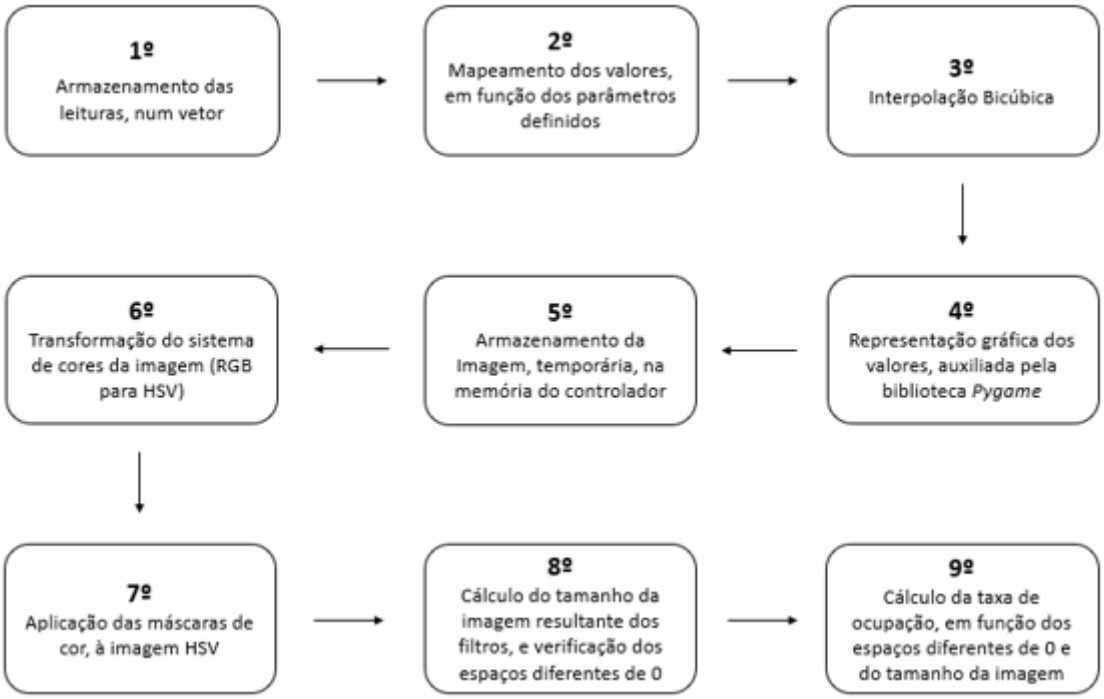

Figura 4.10: Sequência de eventos realizados durante a implementação deste método de análise de imagem

Numa primeira instância, os valores resultantes da leitura são armazenados num vetor, tal como no método anterior. Recorrendo às funcionalidades do *Python*, estes valores são mapeados (*map()*, que aplica uma função a cada item, de um iterável, e retorna uma lista de resultados), de acordo com parâmetros como o intervalo de temperaturas, obtido da leitura do termómetro digital, e o número de cores que é possível observar (*ColorDepth*).

Em seguida, é aplicada uma interpolação bicúbica, através da função *griddata()*, que permite interpolar informação, não estruturada, e D-dimensional. A partir desta interpolação, é apresentada no ecrã a imagem, com recurso à biblioteca *Pygame*, anteriormente detalhada.

O espaço de cores HSV (*Hue Saturation Value*) é um espaço que permite obter uma visibilidade da imagem mais semelhante à do olho humano, quando comparado com o RGB (*Red Green Blue*), em virtude de este não resultar apenas da combinação de três cores primárias. No espaço HSV, tal como o próprio nome indica, *Hue Saturation Value*, são contabilizados três parâmetros, na representação da imagem, a cor, a saturação (quantidade de cor que está combinada com branco), e o valor (quantidade de cor combinada com preto). Assim sendo, o espaço de cores mencionado é mais utilizado em aplicações em que a quantidade de cor presente é um parâmetro de relevância.

Neste momento, é possível aplicar à imagem HSV os filtros de cor, neste caso amarelo e vermelho, que representam os pontos de maior relevância, para o cálculo da taxa de

ocupação. Caso exista alguma alteração que possa afetar a temperatura, abruptamente, ou parâmetro fulcral na obtenção de imagem, poderá ser necessário a atualização dos filtros aplicados.

Para a aplicação destes filtros, é necessário definir o espaço de deteção de ambas as cores e posteriormente guardar a imagem que contém, apenas, as cores detetadas. Por fim, com este valor, é efetuado o cálculo da taxa de ocupação, através da seguinte operação:

$$taxadeocupao = \frac{coeficientes \neq 0}{TamanhodaImagem} * 100 \tag{4.2}$$

Comparação dos Métodos e Testes de Funcionamento

É perceptível que ambos os métodos têm princípios de funcionamento completamente distintos, sendo que, em suma, o primeiro apenas tem em consideração as diferenças verificadas, nos valores dos píxeis, relativamente ao meio, e o segundo realiza uma análise à imagem recolhida.

Verificando-se a existência dos dois métodos, é pertinente efetuar uma comparação de ambos, para confirmar as suas propriedades, em relação a alguns parâmetros inerentes ao sistema.

Assim sendo as tabelas seguintes demonstram os resultados obtidos, variando algumas propriedades, como por exemplo, a temperatura e o intervalo de operação do sensor, baseado no valor inicial de temperatura obtido (definido por L, nas tabelas seguintes). De salientar que os valores percentuais presentes na 5ª e 6ª colunas correspondem à taxa de ocupação, calculada por ambos os métodos.

Tabela 4.4: Tabela comparativa dos métodos, para temperatura de 28º e altura de 2,02m

Temperatura	Altura (m)	Intervalo de Temperatura	Presença de Corpo	Método de Contagem de Píxeis	Método de Análise de Imagem
28º	**2.02**	[L-1, L+2]	✓	22%	36%
				8%	3%
		[L, L+2]	✓	15%	8%
				0%	1%
		[L-1, L+1]	✓	31%	24%
				4%	2%

Tabela 4.5: Tabela comparativa dos métodos, para temperatura de 28,5º e altura de 2,02m

Temperatura	Altura (m)	Intervalo de Temperatura	Presença de Corpo	Método de Contagem de Píxeis	Método de Análise de Imagem
28,5º	**2.02**	[L-1, L+2]	✓	19%	22%
				7%	4%
		[L, L+2]	✓	18%	15%
				0%	7%
		[L-1, L+1]	✓	36%	11%
				1%	8%

Tabela 4.6: Tabela comparativa dos métodos, para temperatura de 28,75º e altura de 2,02m

Temperatura	Altura (m)	Intervalo de Temperatura	Presença de Corpo	Método de Contagem de Píxeis	Método de Análise de Imagem
28,75º	**2.02**	[L-1, L+2]	✓	38%	9%
				7%	4%
		[L, L+2]	✓	15%	12%
				0%	1%
		[L-1, L+1]	✓	30%	20%
				2%	11%

Tabela 4.7: Tabela comparativa dos métodos, para temperatura de 24,5º e altura de 2,02m

Temperatura	Altura (m)	Intervalo de Temperatura	Presença de Corpo	Método de Contagem de Píxeis	Método de Análise de Imagem
24,5º	**2.02**	[L-1, L+2]	✓	15%	6%
				0%	2%
		[L, L+2]	✓	19%	19%
				0%	8%
		[L-1, L+1]	✓	27%	19%
				0%	8%

Tabela 4.8: Tabela comparativa dos métodos, para temperatura de 26,5º e altura de 2,10m

Temperatura	Altura (m)	Intervalo de Temperatura	Presença de Corpo	Método de Contagem de Píxeis	Método de Análise de Imagem
26,5º	**2.10**	[L-1, L+2]	✓	11%	14%
				0%	4%
		[L, L+2]	✓	8%	8%
				0%	4%
		[L-1, L+1]	✓	29%	15%
				2%	9%

Conclusões dos Resultados

Efetuando uma análise aos resultados expostos nas tabelas anteriores, verificamos que o processamento da informação está dependente de algumas variáveis. Considerando que a altura média do tipo de transportes abordado nesta dissertação é de aproximadamente 3 m [23], começou-se por efetuar um estudo, com um valor crescente de altura, começando nos 2,02m.

Nos primeiros casos, considerando temperaturas na ordem dos 28ºC, verifica-se que a qualidade da informação está também dependente de L, o que é um dos objetivos definidos, pois podem existir variações que podem requer adaptações. É o valor recolhido pelo DS18B20, e que auxilia na definição dos parâmetros de aquisição de imagem.

É esperado que, nos casos em que não existe presença de um corpo, o valor obtido seja o mais próximo possível de 0 pois, caso contrário, significará que a câmara está a recolher e processar "ruído". Verificando as tabelas 4.4, 4.5 e 4.6, para uma temperatura próxima de 28ºC, o intervalo de temperatura que oferece maior fiabilidade é de [L, L+2], o que não acontece para temperaturas mais baixas.

De facto, na Tabela 4.7 demonstra os valores obtidos, à mesma altura, mas a uma temperatura de 24,5ºC, concluindo que, segundo o mesmo parâmetro utilizado no caso anterior, o intervalo de temperatura mais indicado é [L-1, L+2].

Outro parâmetro que deve ser considerado são os valores obtidos, na presença de um corpo. Sendo que o sensor estava colocado, exatamente em cima do corpo visualizado (a 2,02m), a taxa de ocupação deve ser coerente com o volume efetivamente ocupado. Para tal, a comparação de ambos os métodos foi utilizada para verificar em qual dos intervalos os valores obtidos foram semelhantes e é imediatamente perceptível que é necessário um intervalo abrangente o suficiente para lidar com o calor resultante da junção de vários corpos, num local de dimensão reduzida. Então, foi selecionado o intervalo [L-1, L+2], para temperaturas abaixo dos 26ºC. Ao realizar os testes, verificou-se que este valor indicado para o efeito e, então, definiu-se o valor de transição de intervalo de temperatura aos 26ºC, considerando as condições normalmente apresentadas no local onde se pretende colocar o dispositivo.

Em seguida estão sintetizadas as conclusões obtidas pela realização dos testes:

- O sensor AMG8833 é influenciado pela temperatura, no que diz respeito à qualidade da imagem adquirida, sendo que, apesar do controlo do meio, deve-se adaptar a pequenas flutuações;

- Para temperaturas mais baixas, é necessário um intervalo de temperatura de maior dimensão, de modo a que o intervalo de operação consiga registar as temperatura de relevância, com o menor ruído possível. Portanto, um intervalo com um ponto máximo de captação elevado;

- No que diz respeito a temperaturas mais elevadas, até 30ºC, a visualização de temperaturas mais baixas criará ruído, que afetará a fiabilidade dos resultados. No entanto, nestas condições, um corpo apresenta valores, no mínimo nesta ordem de valores.
- Para temperaturas de valores muito baixos, foi difícil testar o sistema, considerando o contexto em que o protótipo foi implementado. Portanto, em caso de utilização nesta condições, deve ser efetuado um estudo, com o objetivo de compreender os intervalos de operação. No entanto, estes valores devem elevar o intervalor a temperatura superiores à medida pelo sensor de DS18B20, em virtude do calor emanado, regularmente, pelo corpo humano.

4.3 Funcionamento do Sistema de LED's

O sistema de LED's do sistema é completamente independente de qualquer interação com a FRD, o que evita o mal funcionamento do sistema de exposição físico, em caso de um eventual falha de comunicação com a AM, da parte da *cloud*.

Assim, as condições de ativação de cada um destes díodos emissores de luz são definidas no código implementado para o controlador e são idênticas às apresentadas ao utilizador, na AM do sistema. Em caso de a taxa de ocupação obtida se verificar inferior ou igual a 20%, então é alterado o estado da porta GPIO associada ao LED verde, para "HIGH". O procedimento para as restantes condições é semelhante, sendo que, para uma taxa de ocupação superior ou igual a 80%, é realizado o mesmo processo, mas para a entrada associada ao LED vermelho. Por fim, para valores entre 20% e 80%, é ativado o LED amarelo.

Em síntese, e considerando que apenas cada um desdes componentes deve estar conectado individualmente, para um determinado valor de taxa de ocupação, então em seguida é apresentada a sequência de eventos programada, para o correto funcionamento deste módulo:

1. É definido que se vai evocar os pinos de GPIO pelo *Broadcom SOC Channel*[4];
2. Em seguida, é atribuída aos pinos em questão (GPIO 17, GPIO 20 e GPIO 21) a caraterística de saída e é certificado que todos estão "LOW";
3. Após o cálculo da taxa de ocupação, foram criadas as condições de ativação de um dos LED's, em função do valor da taxa de ocupação;
4. Cada condição inicia-se pela verificação de que os LED's que não são utilizados, para o caso, estão "LOW", para evitar que, quando o ciclo requira que outro LED esteja "HIGH", no processo de troca, o anterior não permaneça "HIGH";
5. Por fim, é alterado o estado do LED pretendido para "HIGH".

[4]O número de baixo nível, que é definido pelo chip do *Raspberry Pi*

4.4 Interação com a *Cloud*

Este sistema utiliza uma base de dados *online*, que lhe permite armazenar os dados, além de possibilitar a posterior exposição dos mesmos, através de uma aplicação móvel.

Assim sendo, nesta secção são descritos os procedimentos e ferramentas utilizadas na implementação desta *cloud*, para o efeito pretendido.

4.4.1 *Firebase Realtime Database*

Um dos principais objetivos do sistema proposto é a comunicação, em tempo real, entre o controlador, a base de dados *online* e a aplicação, para a exposição imediata da informação recolhida pelo sensor.

Este banco de dados hospedado na nuvem *NoSQL*[5] utiliza a sincronização, em vez das recorrentes solicitações HTTP, o que permite a recepção dos dados pelos dispositivos envolvidos, num espaço de tempo mais reduzido, o que confirma a sua utilidade no que diz respeito à comunicação em tempo real. De salientar que a comunicação que é estabelecida entre o controlador e a base de dados, e entre esta e a Aplicação Móvel, estão dependentes da disponibilidade de uma rede de comunicação.

A *Firebase Realtime Database* é um dos diversos serviços gratuitos disponibilizados pela *Google*, ainda que, neste caso, existam algumas funcionalidades que apenas estão disponíveis a utilizadores que contribuam financeiramente.

Deste modo, a FRD é um componente essencial deste sistema, pois armazena todas as leituras efetuadas, possibilitando a sua visualização, através da Aplicação Móvel, o que é possível perceber pela observação do esquema representado na Figura 4.11. Além do mais, alguns dos parâmetros fulcrais ao bom funcionamento do sistema são programáveis, remotamente, pois são editáveis na base de dados, que comunica bidirecionalmente com ambos os componentes do sistema [24].

Figura 4.11: Esquema de Comunicação da Informação

[5]Classe definida de bases de dados que fornecem mecanismos de armazenamento da informação e posterior acesso

4.4.2 Integração no Sistema

A integração da base de dados, tendo em consideração a sua capacidade de comunicação bidirecional, envolve a leitura e escrita de informação através de ambos os componentes, do sistema.

Quanto às configurações ao nível da *Firebase*, foi necessária a alteração das permissões de acesso à escrita e leitura (Figura 4.12, na base de dados, permitindo o seu acesso externo (regras de segurança definidas como públicas):

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Figura 4.12: Regras de segurança, da base de dados

Integração da FRD, no Controlador

A FRD permite a escrita e leitura de informação, no controlador deste sistema, com recurso às bibliotecas existentes desta plataforma, para *Python*. Existem então quatro operações realizadas, duas ao nível de escrita, e duas ao nível de leitura.

Ao executar um ciclo, é definido um período de tempo de espera, para a realização de nova leitura, sendo este período personalizável (em segundos) na AM do sistema, iniciando a aplicação com uma conta com privilégios de administração, o que é visível na Figura 4.13.

A segunda operação de leitura, efetuada sobre a FRD, corresponde à definição do método de análise da informação, tal como indica a Figura 4.12. Tal como foi mencionado na secção onde os métodos são abordados, existem dois modos de funcionamento ("Normal"e "Imagem"), o que é também programável, na aplicação móvel.

Ao efetuar um ciclo, o controlador efetua uma operação de leitura, onde verifica como está definido, atualmente, o *node* correspondente, o que é realizado recorrendo ao comando *firebase.get()*, que é um método assíncrono, que requer dois argumentos. O primeiro, corresponde ao percurso, na árvore principal, até ao *node* pretendido, enquanto que o segundo é o identificativo deste, cujo valor pode ser definido, ou não, retornando, neste caso, todos os dados presentes.

As operações de escrita sobre a base de dados são as responsáveis pela possível visualização da informação, ao nível da AM. Existem então dois tipos de operações deste tipo, que operam de modos completamente distintos:

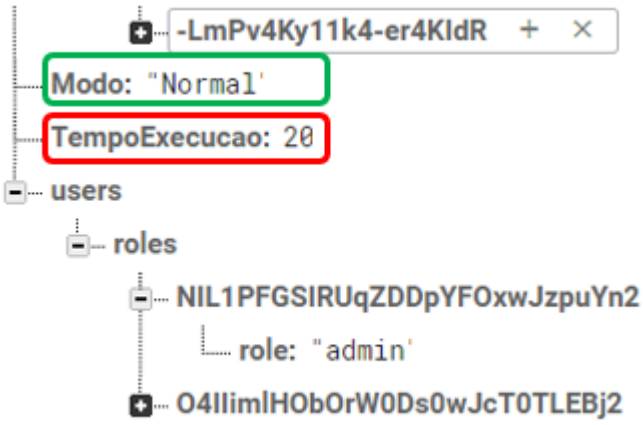

Figura 4.13: *Nodes* de parametrização do sistema
(a vermelho a periodicidade das leituras, e a verde o modo de análise da informação)

- ***Firebase.patch* (*path, snapshot*)** - Cada vez que é efetuado um ciclo, os campos indicados pelo *snapshot*, no node da árvore JSON identificado por *path*, são atualizados. Campo responsável pela exposição da informação, em tempo real;

- ***Firebase.post* (*path, snapshot*)** - Esta é a operação responsável pelo armazenamento do histórico da informação, posteriormente utilizada ao nível da aplicação móvel. Neste campo, as variáveis funcionam de forma semelhante as anteriores, com a diferença que, neste caso, também a data e hora da leitura são enviados para a base de dados. A informação permanecerá armazenada, contrariamente ao que acontece no caso do *patch()*.

Integração da FRD, na Aplicação Móvel

Ao nível da Aplicação Móvel, foi necessário realizar algumas configurações, auxiliadas pelo assistente de conexão à *Firebase*, presente no *SDK Manager*, do *Android Studio*.

No ficheiro *build.gradle*, referente ao módulo, adicionaram-se as implementações que permitem a compatibilização de versões entre as ferramentas e acrescentou-se a dependência dos serviços da *Google*, e da *Firebase*.

Assim, após a realização destas etapas, é possível interligar o ficheiro JSON, da base de dados deste sistema, com a AM. Nas atividades da AM, são aplicadas, tal no controlador, tanto operações de escrita, como de leitura, sobre a FRD. Deste modo, como acontece no *Python*, existem também bibliotecas de auxílio à implementação em *Java*.

Praticamente todas atividades desenvolvidas ao nível da AM envolvem algum tipo de operação, sendo que a maioria envolve apenas leituras, que requerem a criação de eventos que obterão a informação, no caminho especificado, para a referência definida.

Por fim, as operações de escrita sobre a FRD estão presentes apenas nas atividades da AM responsáveis pelas parametrizações das leituras e dos utilizadores. Para tal, basta afixar um valor, no *node* especificado, para a referência e instância definidas (*setValue()*).

4.4.3 Armazenamento e Estruturação da Informação

A informação, em grande parte das bases de dados, é armazenada recorrendo, tanto a tabelas, como a registos. Na *Firebase Realtime Database*, este armazenamento é estruturado com recurso a objetos JSON, que é um formato de organização de dados.

Constituição e Descrição da FRD

Pelo exposto, um banco de dados da FRD é composto essencialmente pela denominada "árvore"JSON (Figura 4.14) e, ao adicionar informação, esta torna-se um *node* na estrutura, ao qual está associada uma chave identificativa, sendo esta chave personalizável, caso seja uma necessidade do utilizador.

No caso deste sistema, a informação encontra-se organizada em cinco secções principais. Em primeiro lugar, para a informação que é exposta, em tempo real, existe um *node*, que contém o estado de ocupação, em cada uma das portas. O *node* seguinte contém a mesma informação que o anterior, com o acréscimo da data e hora em que foi efetuada a medição. Além destes, os dois seguintes contêm parâmetros do sistema, mencionados anteriormente, o período de execução de ciclos, e o modo de processamento da informação.

O último *node* está relacionado com a informação acerca de utilizador, isto é, é o responsável por distinguir o papel do utilizador, no sistema (administrador ou utilizador normal) e armazenar a informação introduzida, pelo mesmo, na instância de registo da conta. Quando um utilizador se regista no sistema, é-lhe atribuído um ID, pela *Firebase Authentication*, ID este que é armazenado na FRD, com o respetivo email associado, papel no sistema e outros dados (número de identificação, data de nascimento e nome completo).

Figura 4.14: Árvore JSON e os respetivos *nodes*, para este sistema

4.5 Aplicação Móvel do Sistema

Nesta secção, incluída ainda na implementação do sistema, mencionam-se os eventos efetuados para a implementação da aplicação móvel, denominada de *Volume Control*.

Todas as ferramentas, configurações e procedimentos, que contribuíram para a implementação, tanto lógica (*backend*), como visual da aplicação (*frontend*), são enunciados, e detalhados nesta secção.

Por fim, de salientar que, para as configurações visuais, foi utilizado XML (*Extensible Markup Language*)[6], enquanto que a lógica, do ponto de vista algorítmico, foi implementada em *Java*, salvo algumas excepções em *Kotlin*.

4.5.1 Configurações Iniciais e Ferramentas Utilizadas

Em primeiro lugar, o *software* que permitiu o desenvolvimento da aplicação, para a plataforma *Android*, foi o *Android Studio*, cuja versão, neste caso, é a 3.4.1. Além do mais, este ambiente de desenvolvimento é *open-source*, estando disponível gratuitamente, sob a licença Apache 2.0[7].

Além do mais, tanto a versão do *compileSdk*[8], como a do *targetSdk*[9], são a 29, enquanto que a *minSdkVersion*[10] é a 15, o que corresponde ao *AndroidX*[11], a mais recente atualização das bibliotecas do *Android*. O Android 4.0.3 (*IceCreamSandwich*[12]) permite que a AM seja compatível com, praticamente, 100% dos dispositivos disponíveis.

4.5.2 Atividades e Fragmentos Implementados

Um projeto, em *Android*, é composto, essencialmente, por *activities* e *fragments*, sendo que um *fragment* representa um comportamento, ou a *interface* de um utilizador, numa *activity*. É então possível a combinação de múltiplos *fragments*, numa *activity*, o que permite a sua reutilização, pelo que, um *fragment* não é mais do que uma secção modular, com ciclo de vida próprio.

Pelo contrário, uma *activity* é uma componente da aplicação, direcionada a um único objetivo, e que depende da interação do utilizador, na maior parte dos casos. Além do mais, aqui é composta a *View*, onde são criadas as UI (*User Interface*), além dos métodos que estão também normalmente associados. O diagrama seguinte (Figura 4.15) expõe um modelo BPMN de interação de utilizadores, com a AM:

[6] A sua principal função é o armazenamento e transporte de informação compreensível a nível humano e computacional

[7] Licença de software *open-source*

[8] Versão do Android que as ferramentas de build usam para compilar, ou correr a aplicação

[9] Versão do Android para qual a aplicação é criada

[10] Versão mínima em que ainda é possível instalar a aplicação

[11] Biblioteca do Android cujo objetivo de criação foi a simplificação textual, durante a implementação

[12] Versão do sistema operativo Android

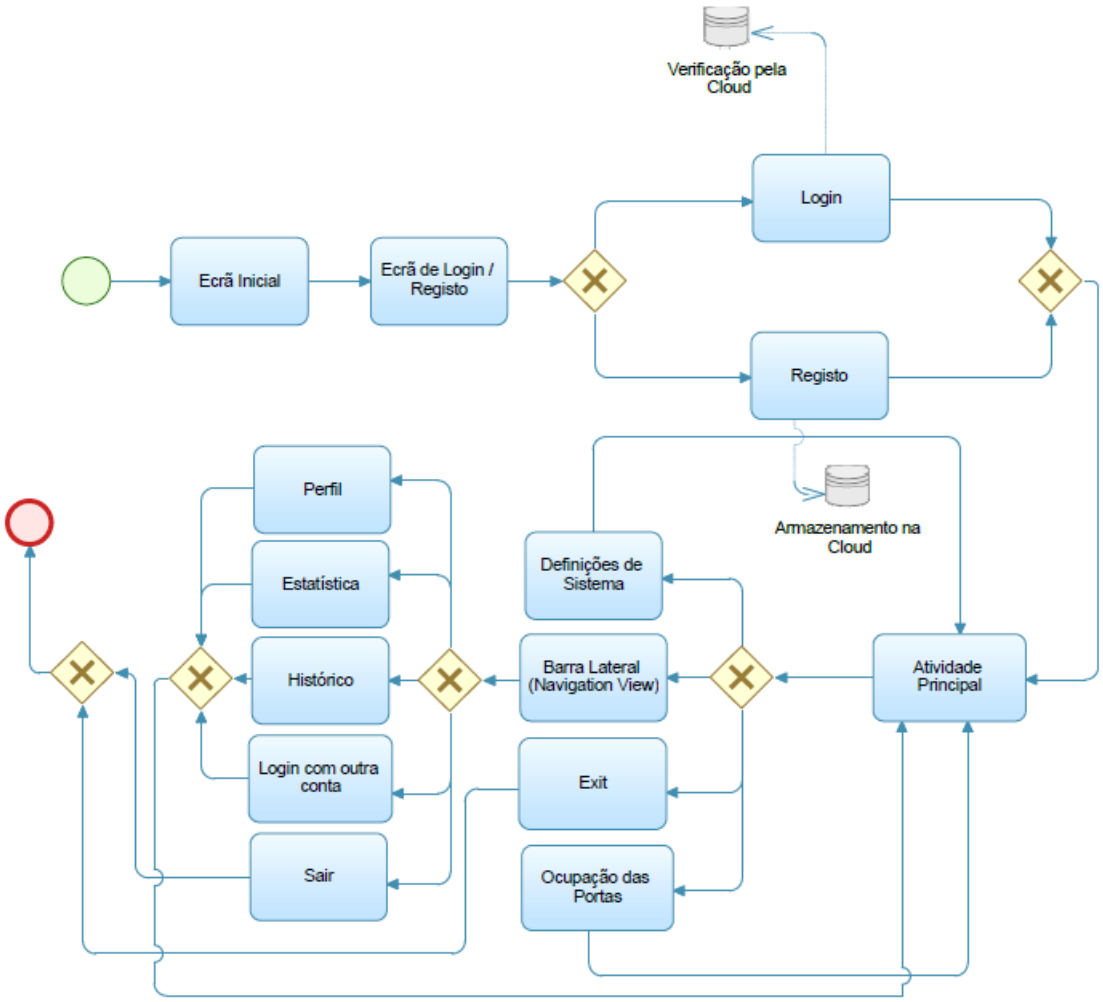

Figura 4.15: Modelo BPMN de interações com a AM

Registo e Autenticação

Com o auxílio da *Firebase*, foram implementadas *activities* para os métodos de Registo e *Login*. A base de dados escolhida para este sistema possui uma ferramenta para a implementação deste tipo de rotinas, a *Firebase Authetication*. Considerando que podem existir dois tipos de utilizadores, o recorrente e o administrador (com privilégios de visualização e interação com a base de dados), foi habilitado o registo de uma conta (são solicitados o email, palavra-passe, número de identificação civil, data de nascimento e nome completo), e posterior *login*, através de um *e-mail*, e da definição de uma palavra-chave. Estas informações, que são utilizadas no início de uma sessão, são guardadas na ferramenta mencionada, que lida com elas, cada vez que forem evocadas.

A implementação dos diferentes tipos de utilizadores é uma operação efetuada com o auxílio da FRD. Tal como já foi mencionado anteriormente, existem cinco *nodes*, na árvore JSON deste sistema, e do último ("users"), deriva um *node* designado de "roles".

Na instância de registo de um utilizador, na AM, é gerado um identificador na FA

(*Firebase Authentication*), com algumas informações acerca deste, como por exemplo, a data do último *login*. Um dos dados mencionados é o *userId* (Figura 4.16), que é atribuído automaticamente pela *Firebase* (fator que pode ser alterado), que é utilizado no *node* "roles"(Figura 4.14).

Deste modo, cada *userId* é guardado na base de dados, com a classificação do seu tipo, que geralmente será "normal". Caso seja pretendida a classificação de um utilizador como "administrador", apenas será necessário alterar este campo, na base de dados, de "normal"para "admin", ou iniciar sessão em *Volume Control*, com uma sessão que contenha o acesso às definições.

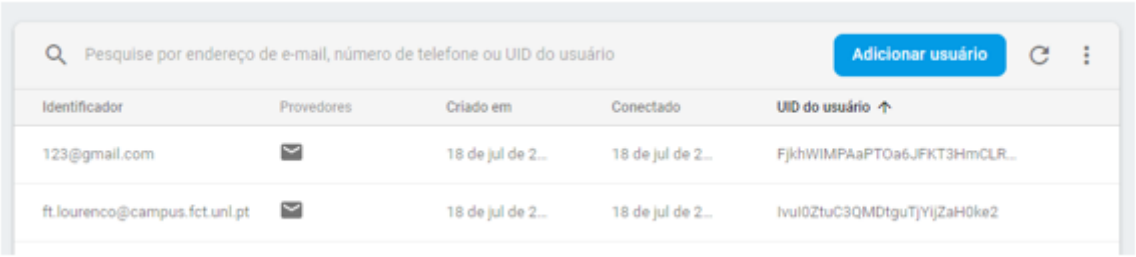

Figura 4.16: Imagem ilustrativa do tratamento da informação dos utilizadores, ao nível da *Firebase Authentication*

De salientar, que esta ferramenta de autenticação permite também a verificação de email, o *login* através de outras plataformas, em que já existe uma conta com informação associada, entre outras funcionalidades de interesse, num outro contexto.

Atividade Principal

A atividade principal da AM é o "centro"de todo o controlo e acesso, às demais atividades. Após efetuar o *login*/registo, e de este ser confirmado pela *Firebase Authentication*, a atividade é iniciada, assim como todos os seus componentes.

Em primeiro lugar, a atividade principal é composta por um menu, configurado por uma *navigation view*, cujo estilo de apresentação é o seguinte (Figura 4.17):

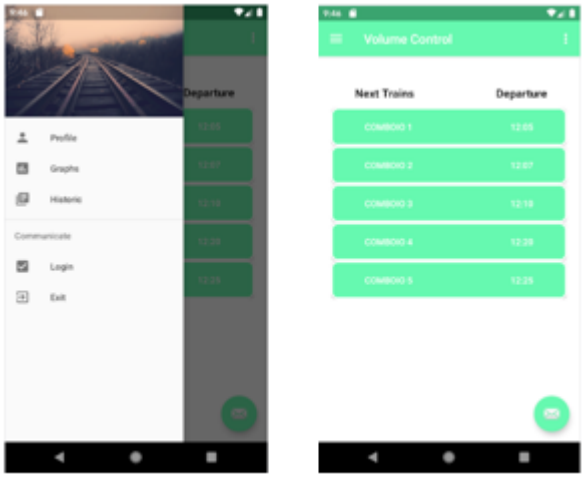

Figura 4.17: *View* da atividade principal, e respetivo menu

Como é possível verificar, o *layout* da atividade é composto pelos botões de iniciação da atividade de visualização dos resultados, em tempo real, que contém informação acerca das horas de partida e o identificativo do respetivo comboio, um menu que permite um acesso às restantes atividades e a barra auxiliar com o nome da aplicação. Além do mais, no canto superior direito, ainda está um botão, que permite o acesso à atividade "Settings".

A existência de dois tipos de utilizadores conduz à criação de dois tipos de menus principais, sendo a única diferença entre ambos a ausência de acesso ao histórico das medições, e às definições.

O processo de iniciação de uma nova atividade é tratado por um objeto que fornece vínculos de tempo de execução, entre componentes separados, neste caso, duas atividades que desempenham funções diferentes, designado por *intent* [25].

De salientar que, esta atividade é a única programada recorrendo a *Kotlin*, dado que, sendo esta a aplicação principal, é necessário que tenha o melhor desempenho possível, em termos de *performance*, recorrendo ao menor número de linhas de código possível, uma das caraterísticas desta variante do *Java* [26].

Representação dos Resultados em Tempo Real

A AM contém uma atividade que apresenta o resultado das leituras efetuadas pelo sensor térmico, em tempo real.

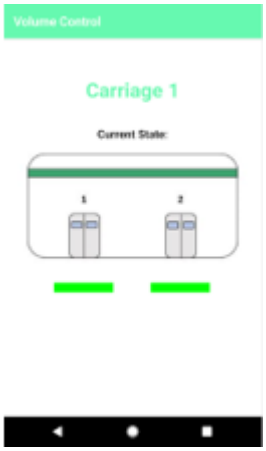

Figura 4.18: *View* da atividade de demonstração dos resultados, em tempo real

Como é possível verificar pela observação da Figura 4.18, no que diz respeito ao *layout* da atividade, esta é composta apenas por uma ilustração de uma carruagem, com as suas portas devidamente identificadas que, sob si, têm uma barra ilustrativa do estado atual de preenchimento volumétrico, junto das entradas numeradas.

Para a implementação da atividade, é necessário recordar que a base de dados, deste sistema, é constituída por alguns *nodes* sendo que, um destes, contém a informação em tempo real, enviada pelo controlador (Figura 4.19), e que é continuamente atualizada.´

Figura 4.19: Esquema de comunicação da atividade, com a *Firebase*

Em primeiro lugar, é necessário criar uma *query* de acesso à base de dados, mais especificamente, ao *node* "Comboio1", realizando uma procura por referência.

Na *query*, sendo que a informação que neste caso é relevante é a disponibilizada no *node* correspondente ao preenchimento da porta, é especificada a leitura deste *child* apenas.

Sendo o objetivo a exposição de apenas três estados de preenchimento, e tendo em conta a simplicidade que deve envolver a AM, optou-se por utilizar barras de progresso como objeto caracterizador da ocupação volumétrica. Para tal, considerando que o estado de preenchimento presente na *cloud* é do tipo *Long*, então é necessário convertê-lo para inteiro, para que seja possível definir as condições de preenchimento da barra de progresso.

O processo é descrito na Tabela 4.9, sendo que, em cada um dos casos, apenas varia a cor com que as mencionadas barras são preenchidas.

Tabela 4.9: Tabela das condições de preenchimento da barra de progresso

Cor	Condição	Algoritmo
Verde	<= 20	A barra de progresso é definida como preenchida a 100%.
Amarelo	Entre 20 e 80	O progresso, do tipo drawable, é aplicado.
Vermelho	>=80	Por fim,a barra é preenchida com a cor correspondente

Área Gráfica da AM

A área gráfica da aplicação contém de um conjunto de diversas atividades, que contribuem uma melhoria na visualização da informação guardada, na base de dados e é iniciada a partir do menu da atividade principal.

Em primeiro lugar, ao abrir esta atividade, surge uma *view*, em grelha, onde estão apresentadas as mais opções gráficas disponíveis.

Nesta grelha, estão disponíveis quatro tipos de gráficos, com informação filtrada, tendo em consideração a sua relevância:

1. **Comparativo do Total das Medidas Efetuadas:**

   Nesta atividade, e em todas as restantes que envolvam a leitura de informação presente na base de dados, é possível restringir os dados, facilitando a implementação, e tornando a programação da AM mais eficaz.

   Assim sendo, para a implementação desta atividade foi necessário, em primeiro lugar, efetuar uma *query* à base de dados, ao node "Comboio1", ramificação de "Historico". É criado um ciclo, onde todos os valores presentes são verificados, e anexados ao objeto "Comboio2", que contém as taxas de ocupação e a data como caraterísticas.

   Em seguida, é verificada a taxa de ocupação, de cada medição, adicionando ao respetivo contador uma unidade, cada vez que o caso se verifique. A partir destes valores de cada contador, é calculada a percentagem, para cada caso, e adicionada ao conjunto de dados associado ao gráfico circular.

   Esta atividade ainda contém o número de medições, em que se verificou cada caso, e o total presente neste *node*(Figura 4.20).

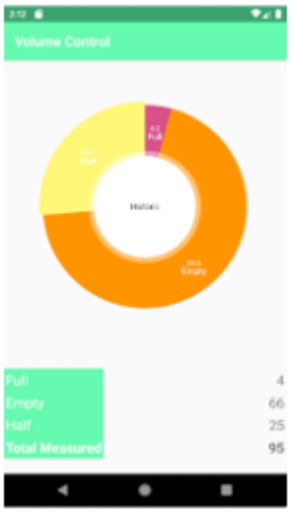

Figura 4.20: *View* da atividade que contém o gráfico comparativo do histórico total

2. **Totais de Cada Resultado, por Ano:**

   O objetivo desta atividade é a demonstração dos totais dos resultados obtidos, por ano, permitindo assim ao utilizador entender a evolução destes, ao longo dos anos de atividade do sistema.

   Para a implementação deste algoritmo, como em todos os casos anteriores, efetuou-se uma *query* à base de dados, mais especificamente ao *node* com o histórico e definiram-se algumas características do gráfico, como o seu tipo, horizontal e de barras.

   Acedendo a todos os resultados presentes na base de dados, criaram-se contadores para todos os casos possíveis (três estados, dados até ao ano de 2023, o que perfaz um total de quinze casos).

   Em seguida, definiu-se um ciclo, no qual todos os *childs* deste *node* serão verificados e armazenados num objeto, para posteriormente comparar com as condições de relevância para o caso.

   Deste modo, com as caraterísticas dos objetos armazenadas, é efetuada uma análise aos segundo e terceiro caracteres (a partir de 0), da *string* "Data", que contém a informação horária da medição (AA**AA**-MM-DD hh:mm:ss), contabilizando assim no contador correspondente, cada caso.

   Em relação aos eixos do gráfico, tanto o eixo horizontal, como o vertical são definidos com recurso à biblioteca *MPAndroidChart*, a responsável pelos gráficos implementados, na AM. Sendo este gráfico de barras, é necessário também definir o tamanho e espaçamento das barras que contém os dados.

   Finalmente, os contadores são enviados para uma lista, cujas entradas são do tipo apropriado para o caso (*BarEntry*), lista esta que será em seguida enviada para os *datasets* criados. As cores correspondentes a cada caso são, neste momento, definidas,

e a informação dos *datasets* é enviada para o gráfico, para ser representada (Figura 4.21).

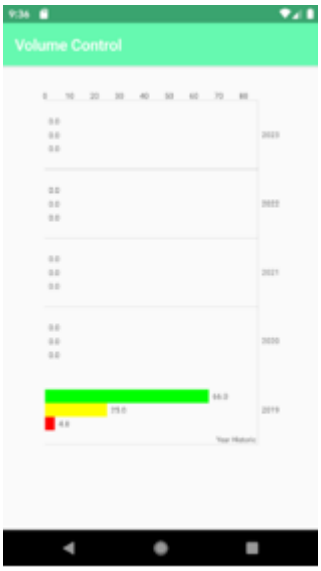

Figura 4.21: *View* do gráfico que contém os totais, por ano

3. **Percentagem de Cada Resultado, por Mês:**

   Na atividade associada aos resultados mensais, foi criado um gráfico para cada caso (cheio, médio e vazio). Os dados correspondentes apenas ao ano de 2019 são filtrados, e dentro desta *query* é efetuada uma leitura aos 5º e 6º caracteres, do campo "Data"(AAAA-**MM**-DD hh:mm:ss), de cada medição.

   Assim, e criados contadores para cada caso, é possível contabilizar o número, para cada mês, e colocar os resultados numa lista de vetores, cujas entradas são do tipo compatível com o tipo de gráfico a apresentar (*PieChart*).

   Este tipo de gráfico utiliza principalmente percentagens, pelo que foi necessário estabelecer uma relação entre o contador de cada mês, e o contador total do ano, para cada um dos três casos existentes. A lista de vetores do tipo correspondente ao gráfico admite entradas do tipo *float*, sendo então necessário converter a percentagem, de *long* para este tipo.

   Por fim, e efetuados alguns comandos de configuração do gráfico, a informação é inserida e, posteriormente representada (Figura 4.22):

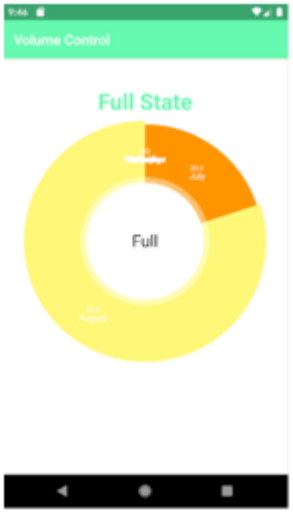

Figura 4.22: Percentagem de cada resultado, por mês

4. ***Top* dos Dias da Semana, por Resultado:**

   Tal como na atividade anterior, existe um gráfico para cada caso, sendo o utilizador a decidir qual visualizar.

   No que diz respeito à implementação do algoritmo, este baseia-se novamente na leitura das características do objeto "Comboio", neste caso o que contém também a informação relativa à data e hora.

   Relembrando novamente o formato enviado pelo controlador à base de dados (AAAA-MM-DD hh:mm:ss), verifica-se que existe um caractere separador, entre a data e a hora, o espaço.

   Assim sendo, realizando uma operação de separação da *string*, é possível obter ambos os dados em separado, guardando o campo referente à data, numa outra *string* auxiliar. Em seguida, considerando apenas este auxiliar, é efetuada uma troca de caracteres, de -"para "/", de forma a que este dado seja reconhecido no formato *DateFormat*.

   Uma vez no formato pretendido, é possível aceder às propriedades do calendário *Android*, sendo que esta instância devolve um inteiro, que corresponde a cada dia da semana. Tal como nos casos anteriores, foram criados contadores, para cada uma das hipóteses, e efetuados todos os cálculos percentuais, para obter a percentagem correspondente a cada dia da semana.

   Por fim, é criada uma condição final que atribui a classificação, em cada caso, a partir de uma análise a uma lista auxiliar, que armazena as percentagens de cada hipótese possível.

   Através das propriedades do *Java* para listas, são avaliados os três maiores resultados, e respetivamente colocados nos lugares correspondentes, no "pódio"apresentado,

assim como numa tabela, onde está também representada a percentagem dos três primeiros classificados, como indica a Figura 4.23.

De salientar a relevância desta atividade, num sistema deste tipo, pois é possível visualizar os dias da semana em que existe maior afluência, sendo que assim é possível precaver eventuais necessidades especiais, para cada dia.

Figura 4.23: *View* da atividade correspondente aos tops dos dias, para cada caso

5. **Histórico da Informação**

   A atividade responsável pela exposição de toda a informação, relativa às medições efetuadas, fornecendo a data e hora a que cada uma foi efetuada, assim como o estado de preenchimento, de cada porta, sob a forma de uma *string* classificativa ("full", "empty", "half").

   Tal como em todos os casos anteriores, nos quais foi necessário efetuar leituras, na base de dados, o procedimento efetuado foi semelhante, diferindo apenas na filtragem da informação relevante, que neste caso é toda, em relação ao *node* "Historico".

   Em seguida, todas as medições são guardadas numa lista de vetores, auxiliar, cujas entradas são do formato de objeto criado para caracterizar cada dado ("Comboio2", neste caso, pois inclui também a informação horária).

   Com o objetivo de auxiliar na implementação da disposição da informação, quando apresentada, foi criado também um adaptador, que, no fundo, é a ponte estabelecida entre a lista de vetores, com os dados, e a *ListView*, implementada no *layout*, desta atividade.

   Pelo exposto, e após a implementação da lista de dados, disponível para a visualização dos administradores, do sistema, é do interesse do utilizador a hipótese de filtragem de dados, conforme a sua necessidade. Assim, foi implementado também um campo onde é possível filtrar, de acordo com a data, e também com o tipo de resultado. Tal é possível, recorrendo às propriedades existentes, no *Android*, para a

implementação de filtros de busca, e adicionando um algoritmo de filtro, à atividade de histórico, e ao adaptador de lista.

6. **Definições**

   O acesso a esta atividade é realizado a partir da atividade principal e está restrito a utilizadores com privilégio de administração atribuído.

   Na realidade, existem três tipos de definições que podem ser parametrizadas a partir da atividade, denominada na AM por "Settings".

   É possível definir, tal como já foi anteriormente mencionado, o tipo de processamento da informação realizado. Ao abrir esta atividade, é possível verificar qual o método atualmente em utilização, o que é implementado efetuando uma *query* à base de dados, mais especificamente ao *node* com esta informação. O caso em utilização é automaticamente selecionado ao abrir a atividade e, em caso de alteração, será selecionado o método escolhido, e a informação será passada à base de dados (Figura 4.24).

Figura 4.24: Esquema de comunicação, no que diz respeito ao parâmetro de análise de informação

   A segunda definição a configurar, a partir da AM, é a periocidade das leituras. Uma vez mais, é possível verificar o valor atual, efetuando uma *query* à base de dados, mais especificamente sobre o *node* "TempoExecucao". Este parâmetro é definido em segundos, e, em caso de alteração, o valor introduzido pelo administrador é enviado para o mesmo *node*.

   Finalmente, é possível atribuir o papel de administrador, a outro utilizador, a partir desta atividade. O esquema de comunicações, durante o processamento desta alteração, é o apresentado na Figura 4.25:

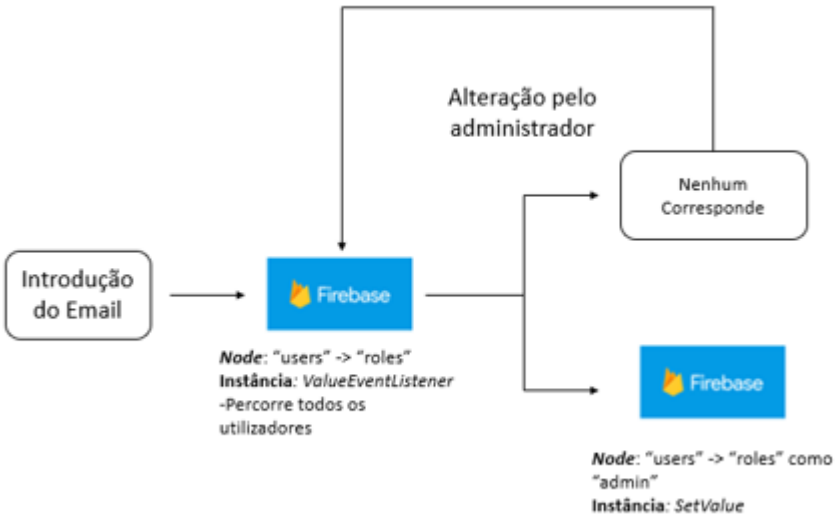

Figura 4.25: Esquema de alteração do papel, no sistema, para um utilizador

7. **Verificação do Perfil**

   Nesta AM foi implementada também uma rotina de verificação dos dados do utilizador, e atualização dos mesmos.

   O acesso a esta atividade é realizado a partir do menu principal da aplicação móvel e é possível conferir todos os dados introduzidos na instância de criação do utilizador em questão, excepto a palavra-passe. Todos os dados visíveis podem ser alterados imediatamente, sendo que os respetivos campos de exposição são também instantaneamente atualizados. Para tal, recordando que existe um *node* atribuído na FRD, para cada utente do sistema, é possível através das simples operações de obtenção de valores da base de dados, aceder às informações pretendidas, excepto a palavra-passe e o email.

   No que diz respeito aos parâmetros associados ao *login*, o processo é relativamente diferente. Neste caso, recorrendo à biblioteca da FA, para *Android Studio*, o email é alterável instantaneamente, através de uma função para o efeito e será enviado um email de confirmação, para a novo endereço atribuído. Quanto à *password*, a segurança que envolve a alteração deste parâmetro dificulta a alteração instantânea, a partir da aplicação móvel e, portanto, optou-se novamente por recorrer à biblioteca de funções da FA. Assim sendo, existe uma função que permite o envio de um *link* de alteração do parâmetro pretendido e fornecendo assim ao sistema uma maior segurança, ao nível das rotinas de utilizadores, como demonstra a Figura 4.26:

Figura 4.26: *View* da atividade de visualização e alteração de dados associados ao perfil

Capítulo 5

Testes e Validação

Para demonstrar o funcionamento do sistema, são apresentados neste capítulo alguns testes e validações, baseados nas interações a realizar por um utilizador, desde o processo de iniciação do dispositivo, à verificação da informação na aplicação móvel.

Assim sendo, para a realização destes testes, o dispositivo foi colocado a uma altura de 2,03m (melhores condições encontradas, para efetuar os testes com a qualidade requerida, para o caso), num ambiente a, aproximadamente, 26,5ºC, segundo o valor fornecido pelo termómetro digital integrado no sistema.

5.1 Inicialização

Este componente do sistema foi inicializado, conectando a fonte da alimentação. O sucesso desta operação é demonstrado na Figura 5.1.

Figura 5.1: Inicialização do Dispositivo

Verificando-se a disponibilidade de uma rede para as comunicações que serão estabelecidas com a base de dados (marcado na Figura 5.1), é possível então iniciar a captação

de imagem.

5.2 Taxa de Ocupação

O caso de uso incide sobre os dois métodos implementados, para demonstrar o funcionamento proposto para cada.

Além do mais, são apresentados também os resultados para diferentes números de presentes, no espaço de aquisição de imagem.

De realçar que, nas figuras apresentadas nesta secção, os valores associados a base de dados, para a ocupação das portas, é duplicado, para que o layout se assemelhe a um contexto mais próximo da realidade do objetivo proposto.

Caso 1 - "Vazio"e Método de Contagem de Píxeis

Em primeiro lugar, para verificar que o sensor está calibrado para possível ruído existente, é demonstrado o funcionamento deste método, com o campo de visão do sensor desimpedido.

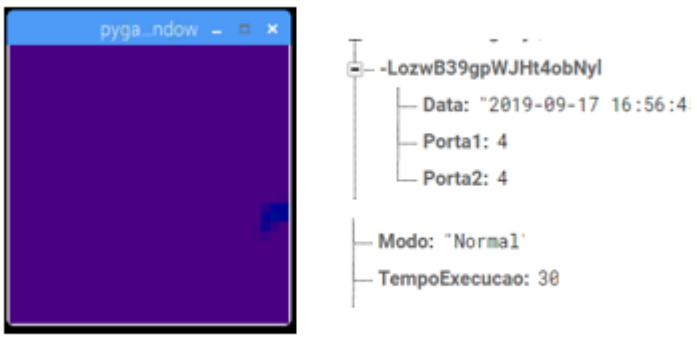

Figura 5.2: Imagem captada e confirmações da base de dados, para o caso de "vazio"

A imagem térmica da Figura 5.2 é afetada por algum ruído, facto comprovado pelos dados recolhidos, nos quais 4% da imagem está preenchida. Isto deve-se ao meio no qual o caso de uso foi aplicado, que não é controlado nas melhores condições, e está sujeito à influência de fatores externos, com relativa frequência. O método de análise da imagem previne a influência este tipo de ruído, da mesma forma que a rotina de autocalibração o minimiza.

Caso 2 - Um Ocupante e Método de Processamento de Imagem

O segundo caso apresentado corresponde à captação da imagem, na presença de um ocupante. Então, a imagem captada pelo sensor de imagem térmica é apresentada em seguida (Figura 5.3):

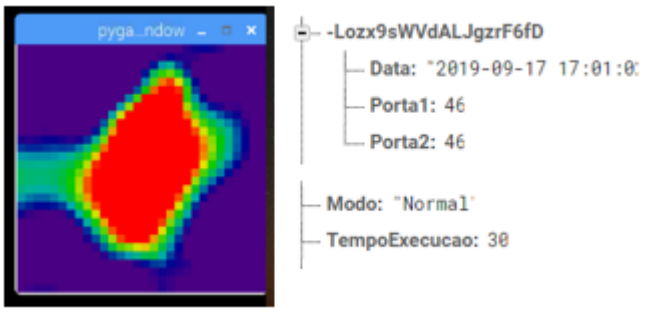

Figura 5.3: Imagem captada e confirmações da base de dados, para o caso 2

Para este caso, considerando a altura a que se encontra o dispositivo, a percentagem obtida é um valor dentro dos parâmetros normais verificados, no capítulo anterior.

5.3 Resultados em Tempo Real

A exposição de resultados na aplicação móvel também se verifica, e é praticamente instantânea, graças ao protocolo e comunicação utilizado pela *Firebase*, e abordado anteriormente.

Pelo exposto, são apresentados as imagens recolhidas da AM, para os dados obtidos nos casos 1 e 2, da subsecção anterior, respetivamente.

Caso 1 - "Vazio"

Figura 5.4: *View* apresentada na AM, para o caso 1

O resultado obtido para este caso (Figura 5.4) é menor que 20% e, por isso, a barra a apresentada deve ser de uma cor elucidativa ao utilizador da falta de ocupação, do espaço.

A mesma condição se aplica à componente física para exposição da informação obtida, sendo que a mesma cor é a emanada pelo LED correspondente. No entanto, ao contrário da

AM, os LED's recebem a condição diretamente do controlador, não requerendo qualquer instrução da base de dados. Então, a imagem seguinte (Figura 5.5) ilustra o funcionamento deste módulo, para o resultado obtido:

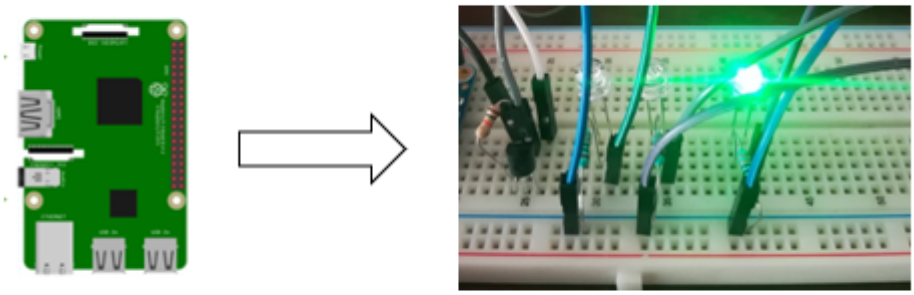

Figura 5.5: Representação do estado de "vazio", através dos LED's do sistema

Caso 2 - Um Ocupante

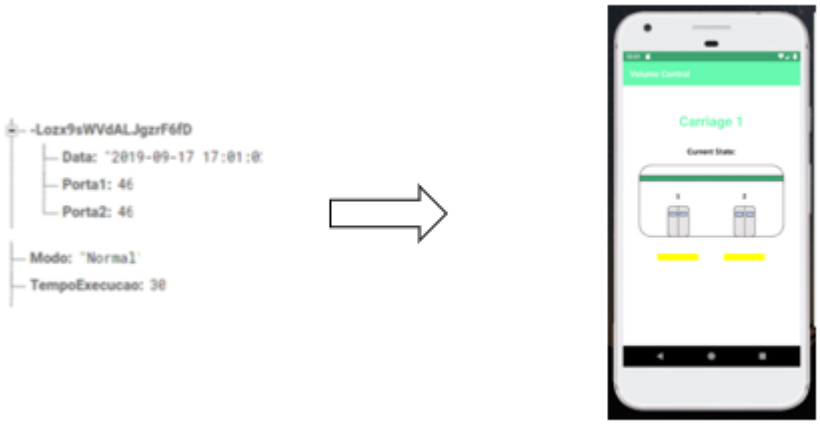

Figura 5.6: *View* apresentada na AM, para este ciclo do sistema

Contrariamente ao caso anterior, neste (Figura 5.6) verificamos uma ocupação de praticamente metade do espaço, o que se deve refletir numa barra amarela, ao nível da AM, o que traduz o sucesso na implementação. Mais uma vez, tal como foi realizado para o caso anterior, a imagem seguinte (Figura 5.7) demonstra o funcionamento da representação gráfica dos dados, a nível físico:

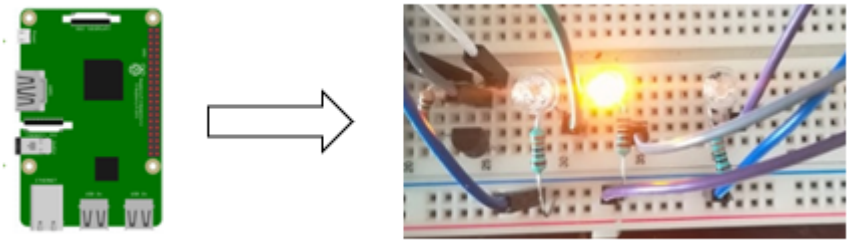

Figura 5.7: Representação física do resultado, para este caso

5.4 Histórico da Informação

Para demonstrar o bom funcionamento da atividade que possibilita a visualização do histórico dos dados recolhidos, confirmaram-se os valores mais recentes presentes na base de dados e a sua correta exposição, demonstrando o filtro implementado para restringir os valores por data (Figura 5.8):

Figura 5.8: Apresentação de todo o histórico presente na FRD, respetivamente filtrada

5.5 Área Gráfica da Aplicação Móvel

Com o objetivo de demonstrar o bom funcionamento destas atividades, da Aplicação Móvel, foram efetuados testes, baseados na informação presente na *cloud*, no momento de realização deste capítulo da dissertação.

Para este efeito, a figura seguinte expõe os gráficos obtidos no contexto especificado, concluindo-se que estão de acordo com o que efetivamente está guardado, na base de dados:

Figura 5.9: Atividades gráficas que compõe a AM

Pela inspeção da Figura 5.9, é possível entender como os números relativos aos dados coincidem entre si, em algumas atividades, comprovando o bom funcionamento da implementação realizada. Por exemplo, na contabilização anual (imagem B), é perceptível que os resultados obtidos coincidem com os apresentados na imagem A, onde o gráfico diz respeito à percentagem de casos, no total da base de dados, que coincide com o total de 2019, pois é o ano decorrente.

5.6 Alteração de Definições do Sistema

A implementação da alteração dos parâmetros de funcionamento do sistema habilita o utilizador à manipulação da aquisição da informação, ao espaço temporal entre cada imagem captada e à atribuição de privilégios de administração.

Em suma, a Figura 5.10 expõe um teste em que são alteradas as propriedades mencionadas, e é efetuada a confirmação da recepção das instruções, ao nível da *cloud*:

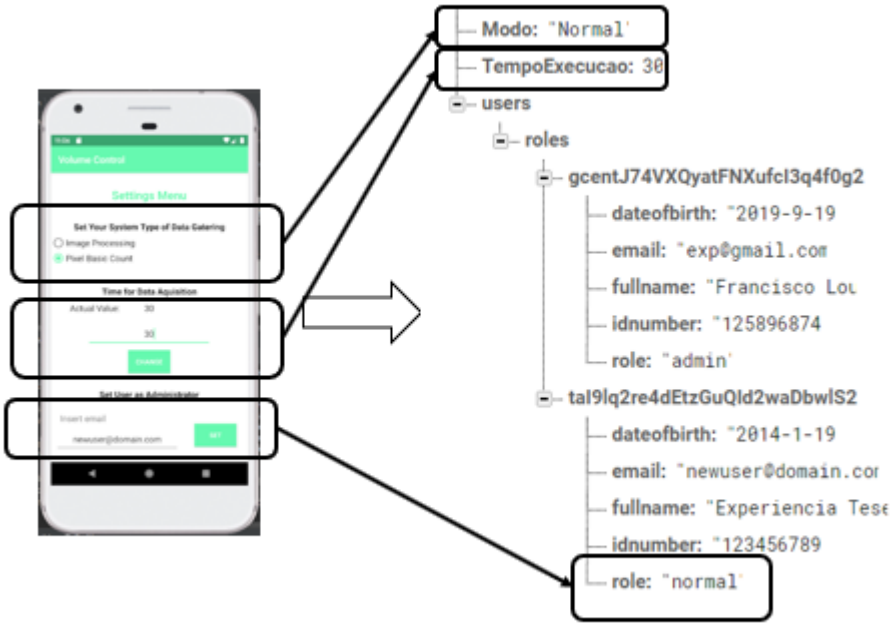

Figura 5.10: Confirmação de alteração de parâmetros do sistema

5.7 *Login* e Registo

As rotinas de *login* e registo já foram clarificadas, no capítulo da implementação do sistema.

Deste modo, a Figura 5.11 demonstra o sucesso verificado no início de sessão, por um utilizador já registado na base de dados:

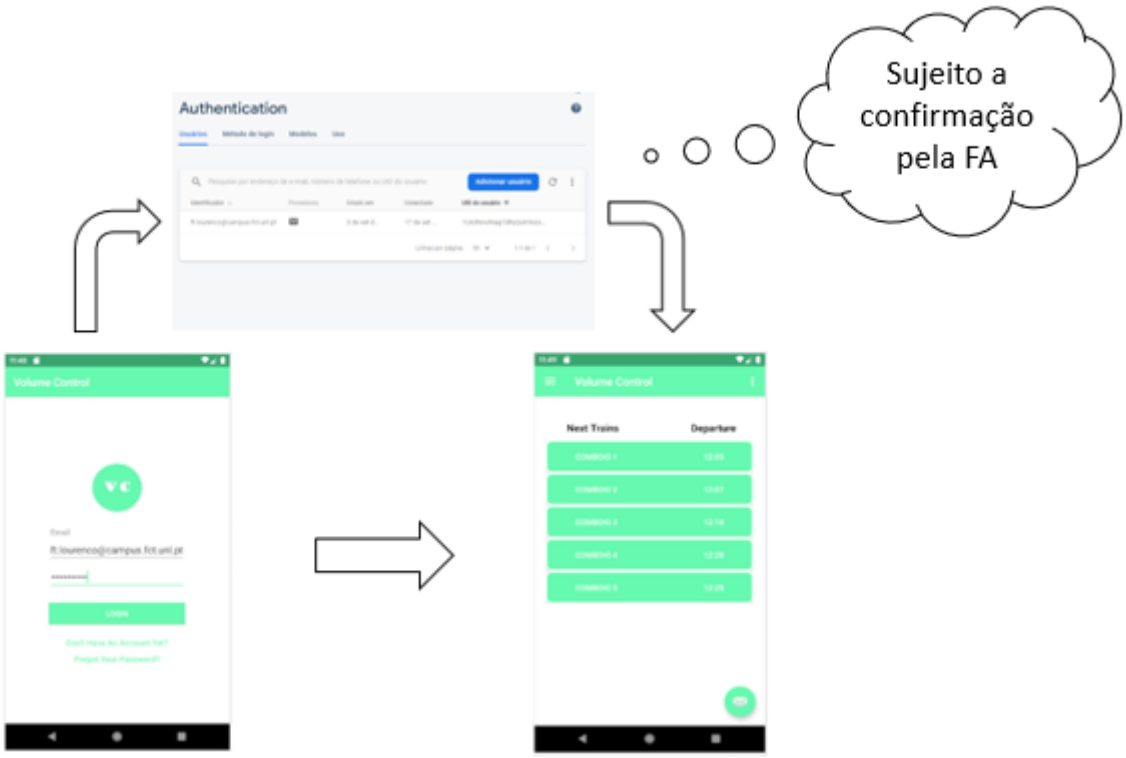

Figura 5.11: Ciclo de confirmação dos dados de utilizador

Na criação do utilizador, como foi mencionado, também é gerado um *node* na FRD, que para o utilizador apresentado na Figura 5.11. Então, na Figura 5.12 são apresentados os que foram introduzidos, para este teste, e que estão associados a este utilizador (data de nascimento, *email*, nome completo, número de identificação civil e papel no sistema).

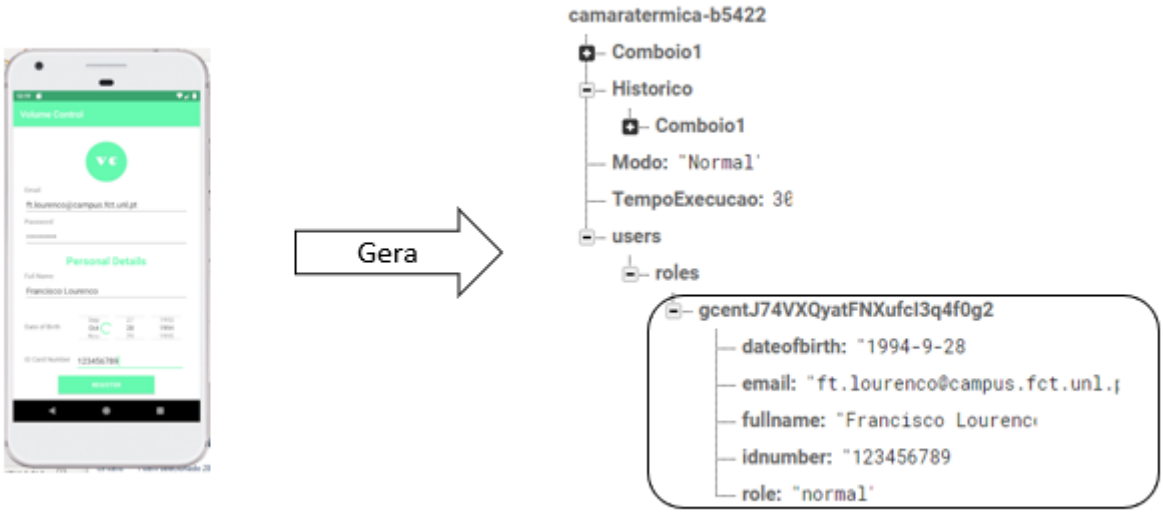

Figura 5.12: *Node* gerado na *FRD*, para este exemplo de registo

5.8 Alteração de Parâmetros do Perfil Associado

A demonstração do sucesso no funcionamento desta atividade da AM é realizada pela alteração dos dados relativos ao utilizador associado aos testes realizados neste capítulo, considerando-se apenas necessário verificar a coincidência dos dados entre os novos introduzidos, e os presentes na base de dados, após este processo. A alteração de informação crucial (*email* e palavra-passe) estão também dependentes das rotinas da FA, como foi mencionado no capítulo anterior e tal como indicam as Figuras 5.13 e 5.14.

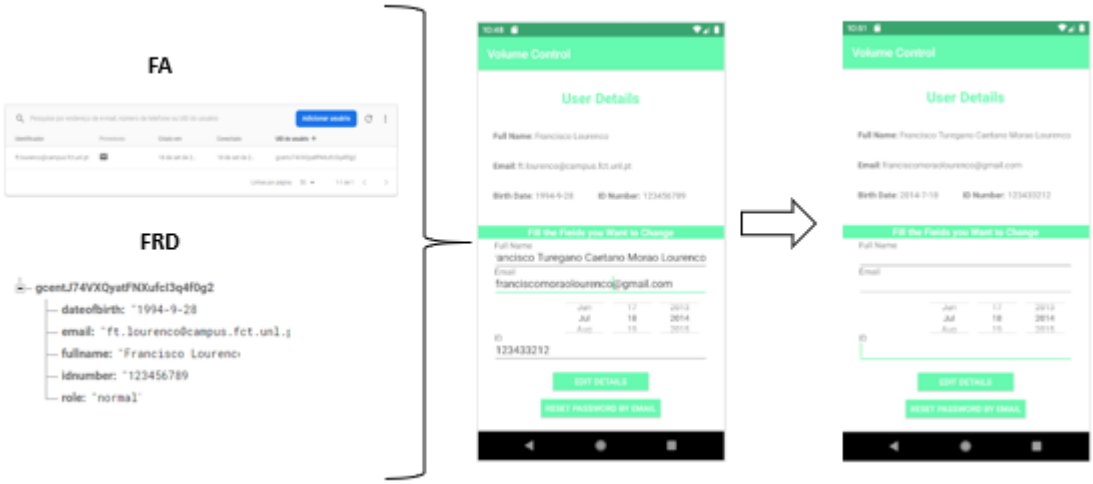

Figura 5.13: Alteração de parâmetros do utilizador associado aos testes realizados, neste capítulo

O único parâmetro cuja alteração ainda não foi confirmada foi a palavra-passe. Para efetuar este processo, e por questões de segurança, implementou-se uma rotina de envio de um pedido de alteração, via email, tal como já foi clarificado mais detalhadamente. Assim sendo, a Figura 5.14 demonstra o funcionamento deste algoritmo, e confirma o sucesso da operação para um utilizador criado para este teste, em ambas as hipóteses de acesso à alteração deste parâmetro:

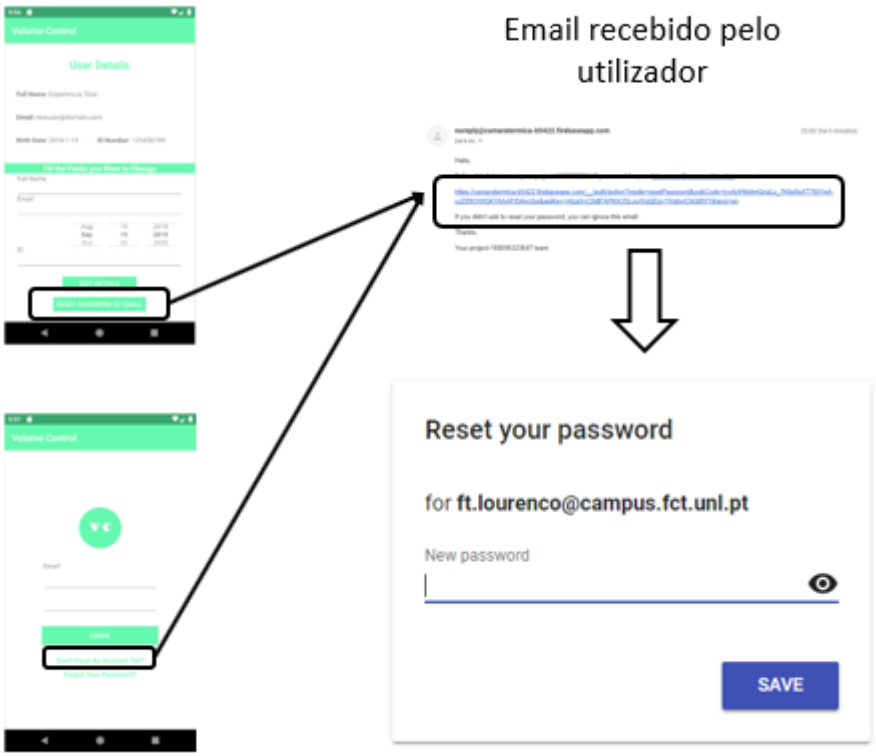

Figura 5.14: Teste ao processo de alteração da *password*, por um utilizador

Capítulo 6

Conclusão

Como foi mencionado nos capítulos iniciais desta dissertação, o objetivo do sistema desenvolvido consistia na obtenção de informação relevante, que pudesse contribuir, significativamente na previsão de necessidades, ao nível dos transportes públicos sem lugar previamente definido e cujo ambiente esteja controlado, no que diz respeito a fatores externos. De salientar que, além destes requisitos, a solução apresentada não deveria captar informação pessoal acerca dos passageiros, impossibilitando a utilização de comuns dispositivos de captação de imagem

Deste modo, foi inicialmente apresentada uma hipótese para uma possível solução para a problemática em questão e consistiu no desenvolvimento de um sistema que, recorrendo a sensores térmicos, identificasse a ocupação volumétrica, nas redondezas das entradas/saídas de um determinado transporte de carruagens.

Assim, na posse destes dados, o sistema deveria enviá-los instantaneamente para uma base de dados *online*, permitindo o acesso remoto à informação, através de uma aplicação móvel desenvolvida.

Pelo exposto, concluí-se, após a análise desta dissertação, que a solução implementada constituí uma opção a considerar para o problema exposto, verificando-se que os principais objetivos propostos na hipótese foram implementados, com sucesso:

1. Identificação da ocupação volumétrica, no espaço de acção do sensor, através de duas soluções para a análise da informação;

2. Recepção da informação, numa base de dados *online*, que viabilize a comunicação com uma aplicação móvel;

3. Interação com o sistema a partir de uma aplicação móvel, na qual é possível conferir a ocupação de um espaço, ou mais, em tempo real, assim como personalizar alguns parâmetros do sistema.

6.1 Crítica

O sistema implementado cumpre os requisitos propostos, fornecendo uma resposta à pergunta de investigação. Verificados os sucessos na obtenção da informação relevante para o caso e na exposição da mesma, numa plataforma acessível a grande parte da população utilizadora de um dispositivo móvel, podem no entanto tecer-se algumas críticas, expostas em seguida:

- Para a obtenção de uma maior qualidade de imagem, seria necessária a utilização de dispositivos com uma maior capacidade de aquisição. Assim, seria possível obter informação mais precisa, e provavelmente seriam necessários menos periféricos, pois a área registada por cada um seria maior.

- Apesar das inúmeras vantagens associadas ao controlador eleito para o protótipo, a capacidade de processamento deste limita o número de sensores de imagem do tipo utilizado, requerendo então, em caso de necessidade, a utilização de mais controladores, aumentando a despesa associada.

- A rotina de autocalibração implementada procura precaver eventuais problemas associados às condições do meio que envolve o dispositivo. No entanto, numa alteração no meio de utilização, o sistema não dispensa uma revisão dos parâmetros de funcionamento.

- A *Firebase* permite todo o tipo de operações, sem qualquer custo associado, pois a informação presente na base de dados e em todos os outros módulos utilizados está definida como pública. Numa implementação definitiva, não deve ser utilizado este tipo de definição, pois poderá colocar em causa as informações obtidas e as respeitantes aos utilizadores.

6.2 Resposta à Pergunta de Investigação

No capítulo inicial desta dissertação, foi colocada uma pergunta de investigação, para a qual foi proposta e testada uma hipótese.

De facto, pela análise do sistema implementado, é perceptível que a hipótese pode constituir uma possível solução para a questão em causa, com as limitações que foram expostas no quarto capítulo.

Em síntese, em resposta à pergunta de investigação, é possível utilizar a captação de imagens térmicas como identificador da ocupação de um espaço, garantindo a devida proteção de dados aos utilizadores do local. É também possível a exposição instantânea da informação, com recurso a uma *cloud*, para o armazenamento e interligação de componentes, e a plataformas indicadas para a visualização, neste caso, o *Android Studio*.

6.3 Trabalho Futuro

O sistema em questão, nesta dissertação, apresenta os resultados de acordo com os objetivos estipulados. No entanto, existem fatores que poderiam aperfeiçoar a aquisição de dados. Com recurso a um controlador com maior capacidade de processamento e maior disponibilidade, ao nível da comunicação I2C, seria possível estender a rede de periféricos, requerendo assim um menor número de controladores.

Além do mais, o *design* da aplicação móvel também poderia sofrer melhorias, com recurso a especialistas neste tipo de trabalho.

Por fim, deixam-se duas sugestões de implementações, que contribuíram, provavelmente, para uma melhoria contínua do sistema. Uma destas seria a implementação do sistema, ao longo da totalidade do espaço, e não apenas nas entradas/saídas, o que poderá conferir um conhecimento total da área. Por fim, a implementação desta informação, os painéis de visualização geral dos horários, seria muito relevante, para uma melhor disposição dos grupos de pessoas, nas entradas para o veículo.

Bibliografia

[1] V. S. Gunge. "Smart Home Automation : A Literature Review". Em: *International Journal of Computer Applications* (2016). ISSN: 0975 – 8887.

[2] D. Project. *The Open Source Definition*. URL: https://opensource.org/osd. Visto em 05/03/2019.

[3] Arduino. *Arduino Uno Rev3*. URL: https://store.arduino.cc/arduino-uno-rev3. Visto em 05/03/2019.

[4] D. Nathan, C. Abafor, U. Aronu e O. Edoga. "Design of a Home Automation System Using Arduino". Em: *International Journal of Scientific Engineering Research* 6.6 (2015), pp. 795–801. ISSN: 2229 - 5518.

[5] T. Y. Wang e S. K. Nguang. "Low cost sensor for volume and surface area computation of axi-symmetric agricultural products". Em: *Journal of Food Engineering* 79 (2007), pp. 870–877. ISSN: 0260 - 8774.

[6] A. A.-H. Jörg Appenrodt e B. Michaelis. "Data Gathering for Gesture Recognition Systems Based on Single Color-, Stereo Color- and Thermal Cameras". Em: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 3.1 (2010), pp. 37–50. ISSN: 2229 - 5518.

[7] I. P.J. L. Colin Puri Leslie Olson e J. Starren. "StressCam: Non-contact Measurement of Users Emotional States through Thermal Imaging". Em: *CHI* 79 (2007), pp. 870–877. ISSN: 0260 - 8774.

[8] Microsoft. *What is the Cloud*. URL: https://azure.microsoft.com/en-us/overview/what-is-the-cloud/. Visto em 22/05/2019.

[9] S. K. G. K.N.Manoj Kumar Kailasa Akhi e M. P. Reddy. "Implementing Smart Home Using Firebase". Em: *International Journal of Research in Engineering and Applied Sciences* 6.10 (2016), pp. 193–198. ISSN: 2249 - 3905.

[10] K. T.M. K. Jayant Mankar Chaitali Darode e P. Shahare. "Review of I2C Protocol". Em: *International Journal of Research in Advent Technology* 2.1 (2014), pp. 474–479. ISSN: 2321–9637.

[11] T. I. Incorporated. *1-Wire Enumeration*. Rev. January 2018. 2013.

[12] F. S. Vanessa Wang e P. Moskovits. Apress, 2013. ISBN: 978-1-4302-4741-8.

[13] V. Pimentel e B. G. Nickerson. "Communicating and Displaying Real-Time Data with WebSocket". Em: (2012), pp. 45–53. ISSN: 1089-7801.

[14] C. R. P. Portugal. *Raspberry Pi*. URL: https://www.raspberrypiportugal.pt/raspberry-pi/. Visto em 02/06/2019.

[15] E. Kompendium. *Raspberry Pi: Belegung GPIO (Banana Pi und WiringPi)*. URL: https://www.elektronik-kompendium.de/sites/raspberry-pi/1907101.htm. Visto em 02/06/2019.

[16] D. Miller. "Adafruit AMG8833 8x8 Thermal Camera Sensor". Em: Adafruit Industries, 2018, p. 30. URL: https://learn.adafruit.com/adafruit-amg8833-8x8-thermal-camera-sensor.

[17] *DS18B20 - Programmable Resolution 1-Wire Digital Thermometer*. 19-7487. Rev. 6. Maxim Integrated. Jul. de 2019.

[18] PythonTm. *Beginners Guide / Overview*. URL: https://wiki.python.org/moin/BeginnersGuide/Overview. Visto em 12/07/2019.

[19] T. P. Hut. *Turning on a LED with your Raspberry Pi's GPIO pins*. URL: https://thepihut.com/blogs/raspberry-pi-tutorials/27968772-turning-on-an-led-with-your-raspberry-pis-gpio-pins. Visto em 20/07/2019.

[20] P. Community. *About — wiki*. URL: https://www.pygame.org/wiki/about. Visto em 21/08/2019.

[21] P. Community. *Pygame.draw*. URL: https://www.pygame.org/docs/ref/draw.html#pygame.draw.rect. Visto em 28/04/2019.

[22] *SPECIFICATIONS FOR Infrared Array Sensor*. Rev. 6. Panasonic Corporation Automation Controls Business Unit DESIGNED. Ago. de 2011, p. 26.

[23] T. Logistic. *ML99 - Ficha Técnica*. URL: https://www.trainlogistic.com/pt/Comboios/Gabinete/fich_ml99.htm. Visto em 12/06/2019.

[24] G. Developers. *Estruturar o banco de dados*. URL: https://firebase.google.com/docs/database/web/structure-data?hl=pt-br. Visto em 09/01/2019.

[25] A. Developers. *Iniciar outra atividade*. URL: https://developer.android.com/training/basics/firstapp/starting-activity. Visto em 09/07/2019.

[26] A. Developers. *Learn the Kotlin programming language*. URL: https://developer.android.com/kotlin/learn. Visto em 09/07/2019.

ANEXO I

Manual de Utilização da Aplicação Móvel

1. **Acesso à Aplicação**

   Está disponível um ícone de iniciação de *Volume Control* a partir de qualquer dispositivo compatível com a versão do *Android Studio* implementada:

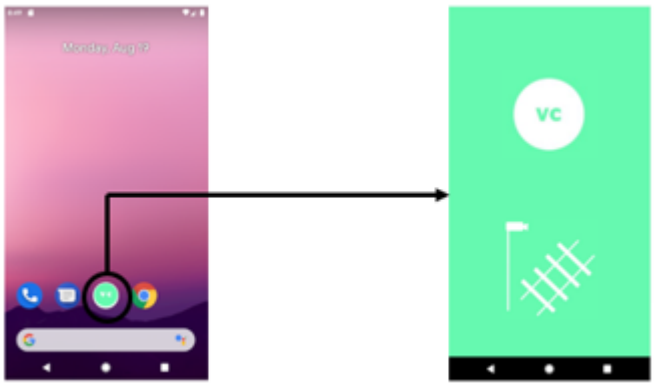

Figura I.1: Inicialização da AM, a partir de um dispositivo móvel

2. ***Login* e Registo**

   Para a criação de um novo utilizador, basta introduzir os dados requeridos (nome próprio, número de identificação civil, data de nascimento, email e password). Posteriormente, o *login* apenas requer o email e password do utilizador.

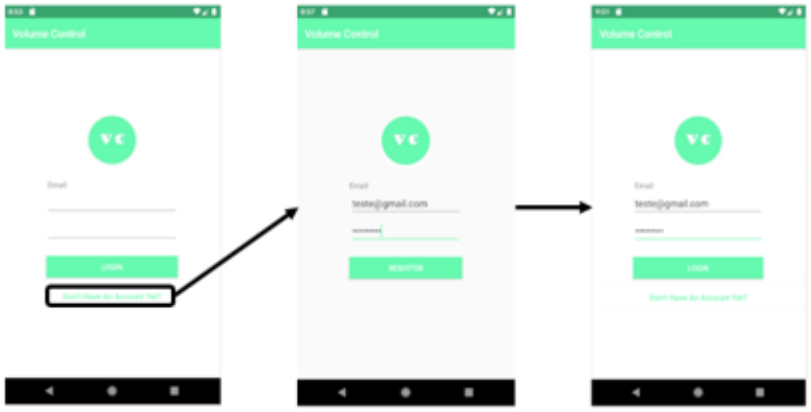

Figura I.2: Criação de um utilizador, e *login* com as respetivas credenciais

3. **Visualização de Resultados, em Tempo Real**

   View que possibilita a visualização dos resultados, em tempo real

Figura I.3: Visualização de resultados, em tempo real

4. **Acesso ao Menu Gráfico**

   Menu onde estão disponíveis todos os gráficos estatísticos da aplicação. Basta premir qualquer uma das opções para que o gráfico pretendido seja apresentado, imediatamente:

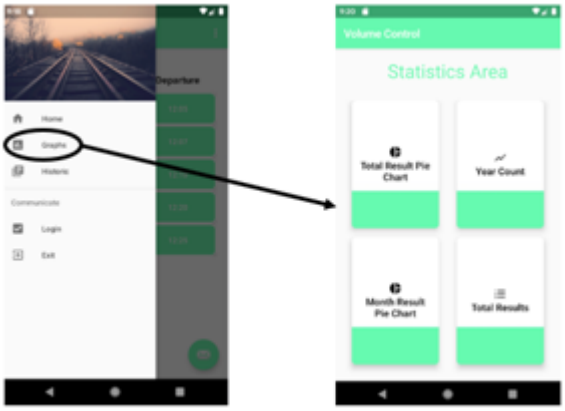

Figura I.4: Interação de acesso à área gráfica

5. **Comparativo do Total dos Resultados**

   Gráfico circular comparativo da percentagem de resultados presente na totalidade da base de dados:

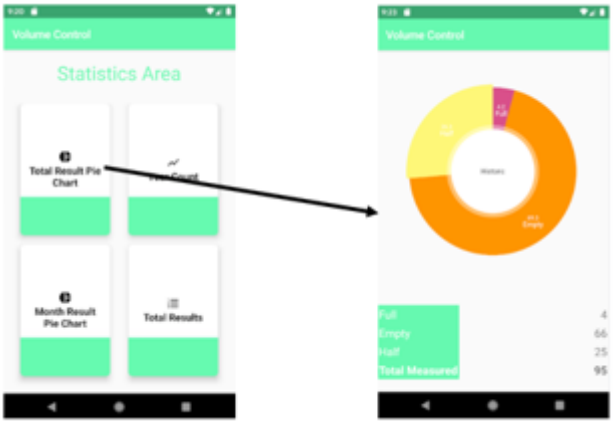

Figura I.5: Histórico do total de resultados

6. **Contagem Anual de Resultados**

   Distribuição anual dos dados presentes na base de dados (parametrizado um limite máximo em 2023):

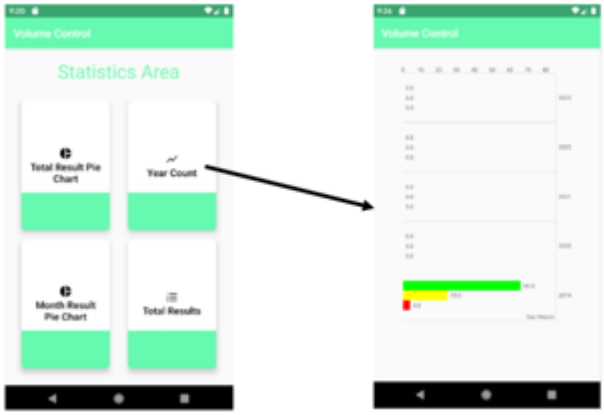

Figura I.6: Atividade que contém o gráfico das contagens, por ano

7. **Comparativo Mensal, em Cada Caso**

   Estão disponíveis os resultados dos três tipos de resultados possíveis, contabilizando a sua percentagem, por mês. Para mudar de caso, basta realizar *scroll*, da direita para a esquerda, no ecrã do dispositivo:

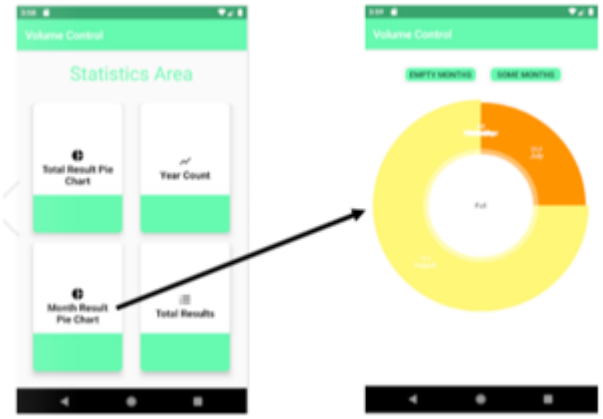

Figura I.7: Pentagem mensal, por ano, para cada tipo de resultado

8. **Ranking de Resultados**

   Como no caso anteriores, os três casos estão disponiveis, realizando o mesmo tipo de *scroll* do caso anterior. Contabilização de quais os dias da semana em que determinado tipo de resultado se verifica:

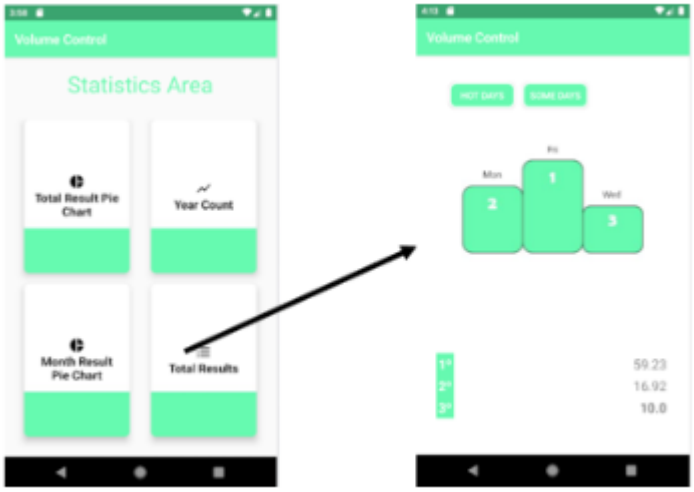

Figura I.8: Top dos dias da semana em que se verificoucou cada resultado

9. **Histórico**

   Histórico do total dos resultados presentes na base de dados, representado sob a forma da sua data, e respetivo resultado obtido. O acesso a esta atividade é realizado através do menu principal, de um administrador do sistema:

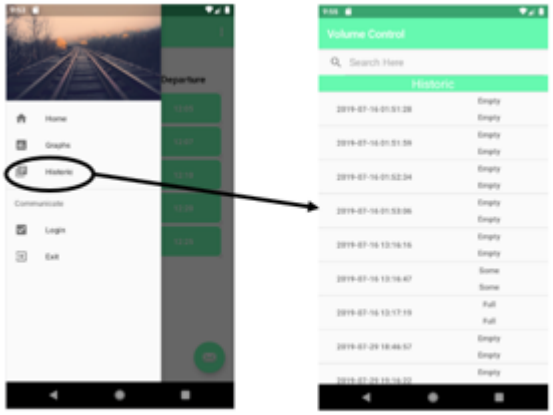

Figura I.9: Histórico presente na base de dados

10. **Definições do Sistema**

    Atividade onde é possivel parametrizar o sistema, de acordo com as necessidades. O botão presente no canto superior direito da atividade principal permite o acesso a esta atividade:

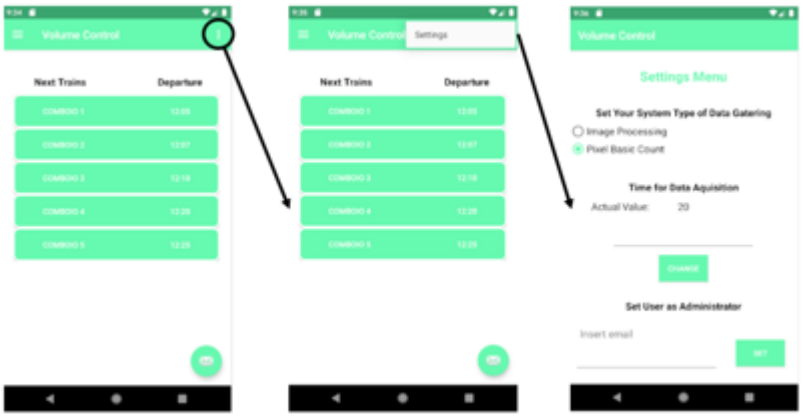

Figura I.10: Procedimento de acesso às definições do sistema

11. **Definição do Método de Análise da Informação**

    Seleção de qual o método a utilizar, na aquisição dos dados. Selecionar a opção pretendida irá instantaneamente alterar o método:

Figura I.11: Alteração do processo de análise de das leituras

12. **Alteração da Periodicidade**

    Periocidade de realização de um ciclo do sistema, em segundos. Indicar o valor em segundos e, em seguida, premir o botão inferior ao campo onde o valor é indicado. Em cima, é indicado o valor atualmente definido.

Figura I.12: Verificação da exposição do valor atual, e respetiva mudança de valor

13. **Atribuição de Privilégios de Administração, no Sistema**

    Basta preencher o campo indicado, com o email do utilizador que é necessário alterar e pressionar o botão à direita.

Figura I.13: Processo de atribuição de privilégios de administração a um utilizador

14. **Encerramento da Aplicação Móvel**

Possibilidades de interação para encerramento da aplicação, quer a partir do menu principal, ou apenas por um simples toque na tecla de retrocesso do dispositivo:

Figura I.14: Interações que permitem o encerramento da AM

ANEXO II

FICHEIRO JSON ASSOCIADO À FRD

Listagem II.1: *Node* que contém as taxas de ocupação volumétrica, em tempo real

```
{
  "Comboio1" : {
    "Porta1" : 2.0,
    "Porta2" : 2.0
  },
```

Listagem II.2: *Node* com o histórico guardado (apenas estão representados alguns *nodes* exemplificativos)

```
  "Historico" : {
    "Comboio1" : {
      "-LjsFouF7Hn1d9gpbRgu" : {
        "Data" : "2019-07-16 01:51:28",
        "Porta1" : 2,
        "Porta2" : 2
      },
      "-LjsFw_ehKycrvPYmpNA" : {
        "Data" : "2019-07-16 01:51:59",
        "Porta1" : 0,
        "Porta2" : 0
      },
      "-LnDf2inwrwhDMrm12v-" : {
        "Data" : "2019-08-26 17:04:49",
        "Porta1" : 2.0,
        "Porta2" : 2.0
      }
```

```
    }
  },
```

Listagem II.3: Configuração do método de análise de dados

```
  "Modo" : "Image",
```

Listagem II.4: Parâmetro que define o tempo de execução, entre ciclos

```
  "TempoExecucao" : 10,
```

Listagem II.5: *Node* que contém a informação acerca dos privilégios de cada utilizador, no sistema

```
  "users" : {
    "roles" : {
      "gcentJ74VXQyatFNXufcI3q4f0g2" : {
        "dateofbirth" : "2011-10-6",
        "email" : "ft.lourenco@campus.fct.unl.pt",
        "fullname" : "Francisco Lourenco",
        "idnumber" : "147852369",
        "role" : "admin"
      },
      "taI9lq2re4dEtzGuQId2waDbw1S2" : {
        "dateofbirth" : "2014-1-19",
        "email" : "newuser@domain.com",
        "fullname" : "Experiencia Tese",
        "idnumber" : "123456789",
        "role" : "normal"
      }
    }
}
```

ANEXO III

Algoritmo, em Python, implementado no controlador

Listagem III.1: Bibliotecas a importar, para o controlador e periféricos

```
from Adafruit_AMG88xx import Adafruit_AMG88xx
import pygame
import os
import math
import time
import datetime
import cv2
import glob

import numpy as np
from scipy.interpolate import griddata
from matplotlib import pyplot as plt
from datetime import datetime
from firebase import firebase
from colour import Color
from PIL import *
import RPi.GPIO as GPIO
```

Listagem III.2: Funções e configurações associadas ao Termómetro Digital e aos LED's

```
#TEMPERATURE SENSOR
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
```

```
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

def read_temp_raw():

    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():

    lines = read_temp_raw()
    while lines [0].strip()[-3:] !='YES':

        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')

    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        return temp_c

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(20,GPIO.OUT)
GPIO.setup(17,GPIO.OUT)
GPIO.setup(21,GPIO.OUT)
GPIO.output(17,GPIO.LOW)
GPIO.output(21,GPIO.LOW)
GPIO.output(20,GPIO.LOW)
```

Listagem III.3: Inicialização do AMG8833, da FRD e definição dos parâmetros de operação do sensor de imagem térmica

```
#THERMAL SENSOR
#how many color values we can have
COLORDEPTH = 1024

os.putenv('SDL_FBDEV', '/dev/fb1')
pygame.init()
```

```
#initialize the sensor
sensor = Adafruit_AMG88xx()

#initialize Firebase
url='https://camaratermica-b5422.firebaseio.com/'
firebase=firebase.FirebaseApplication(url,None)

points = [(math.floor(ix / 8), (ix % 8)) for ix in range(0, 64)]
grid_x, grid_y = np.mgrid[0:7:32j, 0:7:32j]

#sensor is an 8x8 grid so lets do a square
height = 240
width = 240

#the list of colors we can choose from
blue = Color("indigo")
colors = list(blue.range_to(Color("red"), COLORDEPTH))

#create the array of colors
colors = [(int(c.red * 255), int(c.green * 255), int(c.blue * 255)) for c in
    ↪ colors]

displayPixelWidth = width / 30
displayPixelHeight = height / 30

lcd = pygame.display.set_mode((width, height))

lcd.fill((255,0,0))

pygame.display.update()
pygame.mouse.set_visible(False)

lcd.fill((0,0,0))
pygame.display.update()

#some utility functions
def constrain(val, min_val, max_val):
    return min(max_val, max(min_val, val))

def map(x, in_min, in_max, out_min, out_max):
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min

#let the sensor initialize
time.sleep(.1)
```

Listagem III.4: Leitura da informação, construção da imagem térmica e cálculo da taxa de ocupação volumétrica, com exposição de ambos os métodos implementados

```
#read the pixels
pixels = sensor.readPixels()
array = sensor.readPixels()
#now = datetime.now()
pixels = [map(p, MINTEMP, MAXTEMP, 0, COLORDEPTH - 1) for p in pixels]

#perdorm interpolation
bicubic = griddata(points, pixels, (grid_x, grid_y), method='cubic')

#draw everything
for ix, row in enumerate(bicubic):
    for jx, pixel in enumerate(row):
        pygame.draw.rect(lcd, colors[constrain(int(pixel), 0, COLORDEPTH-
            ↪ 1)], (displayPixelHeight * ix, displayPixelWidth * jx,
            ↪ displayPixelHeight, displayPixelWidth))

pygame.display.update()
#print (sensor.readPixels())
espacos_livres=[]

result2 = firebase.get('/Modo', None)
print(result2)

if result2 == "Image":
    #IMAGE PROCESSING
    plt.imsave('Camara Termica/Teste.png', bicubic)
    image=cv2.imread('/home/pi/Desktop/Camara Termica/Teste.png')

    #HSV, for better color detection
    imagehsv=cv2.cvtColor (image, cv2.COLOR_BGR2HSV)

    #Yellow
    low_yellow = np.array([15, 0, 0])
    high_yellow = np.array([36, 255, 255])

    #Red
    low_red = np.array([0, 50, 50])
    high_red = np.array([0, 255, 255])

    #Mask Application
    yellow_mask = cv2.inRange(imagehsv, low_yellow, high_yellow)
    red_mask = cv2.inRange(imagehsv, low_red, high_red)
```

```
        mask = cv2.bitwise_or(yellow_mask, red_mask)
        target = cv2.bitwise_and(image, image, mask = mask)

        plt.imsave('Camara Termica/YellowandRed.png', target)

        #Counts the Size of Image (1024)
        height, width, channels = target.shape
        cntPixels = height*width

        #Non-Zero, after Filter
        nzCount = np.count_nonzero(target)

        taxaOcupacao = round(float(nzCount)/float(cntPixels) * 100)

        cv2.imwrite('/home/pi/Desktop/Camara Termica/hsv.png', imagehsv)

    else:

        #For to check all the matrix
        for i in array:

        #If the variable value is less than lower setpoint
            if i < MINTEMP:

        #Saves the "free spaces" in an aux vector
                espacos_livres.append(i)

        #Measures the size of this aux vector
            espaco_desocupado=len(espacos_livres)

        #Ocupation quote
            taxaOcupacao=100-((espaco_desocupado*100)/64)
```

Listagem III.5: Ciclo de funcionamento dos LED's

```
if taxaOcupacao <= 20:
            #GREEN
            GPIO.output(20,GPIO.LOW)
            GPIO.output(21,GPIO.LOW)
            GPIO.setmode(GPIO.BCM)
            GPIO.setwarnings(False)
            GPIO.setup(17,GPIO.OUT)
            print ("LED on")
            GPIO.output(17,GPIO.HIGH)
```

```
    elif taxaOcupacao >= 80:

        #RED
        GPIO.output(17,GPIO.LOW)
        GPIO.output(21,GPIO.LOW)
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        GPIO.setup(20,GPIO.OUT)
        print ("LED on")
        GPIO.output(20,GPIO.HIGH)

    else:

       #YELLOW
        GPIO.output(17,GPIO.LOW)
        GPIO.output(20,GPIO.LOW)
        GPIO.setmode(GPIO.BCM)
        GPIO.setwarnings(False)
        GPIO.setup(21,GPIO.OUT)
        print ("LED on")
        GPIO.output(21,GPIO.HIGH)
```

Listagem III.6: Envio da informação, para a base de dados, e definição do período de execução de ciclos, a partir da FRD.

```
#DATABASE

#Updates the main child live, controlling the progress bars
result =
    ↪ firebase.patch('/Comboio1',{'Porta1':taxaOcupacao,'Porta2':taxaOcupacao})


#Sends to storage, with datetime as well
result = firebase.post('/Historico/Comboio1', {'Porta1':taxaOcupacao,
    ↪ 'Porta2':taxaOcupacao, 'Data':datetime.now().strftime('%Y-%m-%d
    ↪ %H:%M:%S')})

#Gets the Time Spacing Between Measures, from Firebase
result1 = firebase.get('/TempoExecucao',None)

#From Firebase
time.sleep(result1)
```