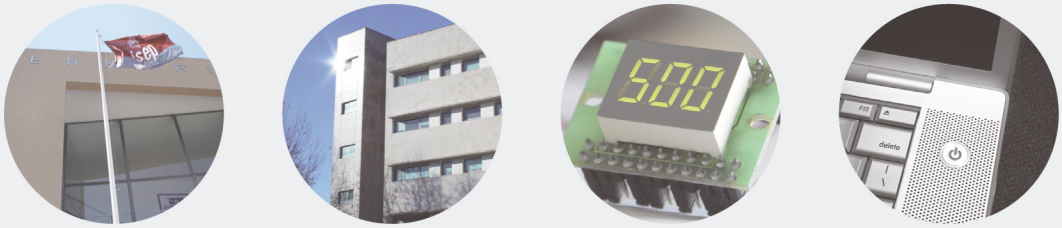
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Sistema de Visão Computacional Low-Cost para Deteção e Contagem de Pessoas e Veículos em Smart Cities

MIGUEL MENDES AMADO
dezembro de 2021

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Sistema de Visão Computacional *Low-Cost* para Deteção e Contagem de Pessoas e Veículos em *Smart Cities*

Miguel Mendes Amado

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas Autónomos

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Novembro, 2021

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Sistemas Autónomos.

Candidato: Miguel Mendes Amado, Nº 1180436, 1180436@isep.ipp.pt

Orientação Científica: André Miguel Pinheiro Dias, apd@isep.ipp.pt

Coorientação Científica: Alfredo Oliveira Martins, aom@isep.ipp.pt

Empresa: CEiiA

Orientador: Tiago Filipe Macedo, tiago.macedo@ceiia.com

Coorientador: Hélder Covas Oliveira, helder.oliveira@ceiia.com

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Novembro, 2021

"Always go with the choice that scares you the most, because that's the one that is going to help you grow."
Caroline Myss

Agradecimentos

A elaboração desta dissertação apenas foi possível graças ao CEiiA - Centro de Engenharia e Desenvolvimento de Produto, nomeadamente ao meu Orientador Tiago Macedo e Coorientador Hélder Covas por me integrarem num projeto que me permitiu adquirir conhecimentos no âmbito da visão computacional. Adicionalmente, possibilitaram a alocação de tempo, assim como facilitaram o aquisição de todo o material necessário para desenvolvimento desta dissertação.

Quero também agradecer a orientação científica por parte do Engenheiro André Dias e do Engenheiro Alfredo Martins e a ajuda disponibilizada relativamente à estruturação e escrita desta dissertação.

Acima de tudo, quero agradecer aos meus pais e irmão por me terem possibilitado a oportunidade de estudar no Instituto Superior de Engenharia do Porto (ISEP), longe da terra natal, assim como agradecer todo o apoio diário por parte da minha avó Maria Egas.

Todo o apoio por parte dos meus amigos mais próximos foi também bastante importante, principalmente, em momentos de maiores dificuldades no decorrer desta dissertação. Agradeço também a toda a equipa do Departamento de Eletrónica e Sistemas Embebidos pela ajuda e conhecimento passado a todos os níveis, para além do excelente ambiente de trabalho existente.

Quero também deixar um agradecimento a todas as pessoas, desde colegas de curso a professores que fui conhecendo e vindo a ter contacto ao longo destes anos no Ensino Superior, por todo o espírito e vivências académicas, como também pela constante troca de conhecimentos da área de Engenharia.

Por fim, quero deixar um grande agradecimento às pessoas com quem partilhei casa, durante todo o período de desenvolvimento da dissertação, por me respeitarem neste período e por procurarem sempre criar um bom ambiente entre todos.

Miguel Amado

Resumo

Devido ao contínuo aumento da densidade populacional no meio urbano, têm-se identificado problemas de congestionamento do tráfego e, consecutivamente, de controlo dos níveis de poluição com vista à descarbonização das *smart cities*. Com o rápido crescimento da área de Inteligência Artificial (IA), nomeadamente no campo da visão computacional, estudos apresentam as vantagens da sua utilização, comparativamente a metodologias mais tradicionais. Assim sendo, é proposto um sistema *low-cost* baseado em visão computacional e IA com foco na contagem de pessoas e veículos. A informação adquirida ajudará no estudo da mobilidade com otimizações da rede de transportes públicos e diminuição do congestionamento do tráfego local. Instalado em zonas estratégicas das cidades, este sistema processa, localmente e em tempo-real (*edge-computing*), as imagens de vídeo e, posteriormente, partilha as contagens através de uma comunicação sem fios, tendo em conta a proteção da privacidade dos dados pessoais. O protótipo desenvolvido demonstrou tratar-se de um produto *low-cost* e de baixo poder computacional. Este utiliza técnicas de IA e de otimização, que demonstraram exigir um baixo consumo energético sem comprometer a performance da contagem. Quanto à eficiência da contagem, utilizou-se um modelo de *Deep Learning* (DL) juntamente com um algoritmo de *tracking*, que apresentam resultados prometedores perante cenários com diferentes condições atmosféricas.

Palavras-Chave: CNN, *deep learning*, Deepstream, *edge-computing*, linha de contagem, *low-cost*, *low-power*, MQTT, Multiprocessos, GPU, IoT, NB-IoT, NvDCF, NVIDIA Jetson Nano, performance, protótipo, ResNet, *smart city*, SSL/TLS, *tracking*.

Abstract

Due to the population's density increase in the urban areas and in order to decarbonize smart cities, traffic congestion problems are being identified alongside the control of pollution's levels. Considering the quickly growing field of Artificial Intelligence (AI), particularly in computer vision, the studies focus on the strengths regarding its use compared to traditional methodologies. Therefore, a low-cost system based on computer vision and AI focusing on counting people and vehicles is proposed. The acquired information will assist the mobility field with optimizations in public transport network and the decrease of local traffic congestions. Installed in strategic areas of cities, this system processes, locally and in real time (edge-computing), the video images and, posteriorly, shares the counting data through a wireless communication, ensuring the protection of personal data. The developed prototype proves to be low-cost and has low computing power. It uses optimized AI techniques which have been shown to require low energy consumption without compromising the performance. As for counting efficiency, Deep Learning (DL) models have been used along with a tracking algorithm, demonstrating promising results in scenarios with different weather conditions.

Keywords: CNN, counting line, deep learning, Deepstream, edge-computing, low-cost, low-power, MQTT, multiprocessing, GPU, IoT, NB-IoT, NvDCF, NVIDIA Jetson Nano, performance, prototype, ResNet, smart city, SSL/TLS, tracking.

Índice

Lista de Figuras

Lista de Tabelas

Listagens

Lista de Acrónimos

2D	*2-Dimensional*
3D	*3-Dimensional*
3G	*3rd Generation*
3GPP	*3rd Generation Partnership Project*
4G	*4th Generation*
ABS	*Acrylonitrile Butadiene Styrene*
AC	*Alternating Current*
AIPD	Avaliação de Impacto sobre a Proteção de Dados
AMQP	*Advanced Message Queuing Protocol*
API	*Application Programming Interface*
APN	*Access Point Name*
ARM	*Advanced RISC Machines*
AT	*ATtention*
BACF	*Background-Aware Correlation Filters*
CA	Entidade Certificadora
CCTV	*Closed-Circuit Television*
CE	Comunidade Europeia
CNN	*Convolutional Neural Networks*
CNPD	Comissão Nacional de Proteção de Dados
CoAP	*Constrained Application Protocol*
COVID-19	Doença do Coronavírus
CPU	*Central Process Unit*

CSI *Camera Serial Interface*

CUDA *Compute Unified Device Architecture*

DAN *Deep Affinity Networks*

DC *Direct Current*

DCF *Discriminative Correlation Filter*

DCNN *Deep Convolutional Neural Networks*

DCPS *Data-Centric Publish-Subscribe*

DDS *Data Distribution Service*

DIN *Deutsches Institut für Normung*

DL *Deep Learning*

DL *Downlink*

DLRL *Data-Local Reconstruction Layer*

DNN *Deep Neural Network*

DR *Data Report*

DT *Data Trigger*

DTLS *Datagram Transport Layer Security*

eDRX *Extended Discontinuous Reception*

ETSI *European Telecommunications Standards Institute*

FA *Function Acknowledge*

FaaS *Function as a Service*

FC *Fully Connected*

FE *Function Execute*

FIFO *First In, First Out*

FLOPS *Floating-Point Operations Per Second*

FN *False Negative*

FoV *Field of View*

FP	*False Positive*
FP	*Floating Point*
FPS	*Frames per Second*
FTP	*File Transfer Protocol*
GPIO	*General Purpose Input/Output*
GPRS	*General Packet Radio Service*
GPU	*Graphics Processing Unit*
GSM	*Global System for Mobile*
HD	*High-Definition*
HTTPS	*HyperText Transfer Protocol Secure*
HW	*Hardware*
I/O	*Input/Output*
I2C	*Inter-Integrated Circuit*
I2S	*Inter-IC Sound*
I&D	Investigação e Desenvolvimento
IA	Inteligência Artificial
IBM	International Business Machines Corporation
ICCID	*Integrated Circuit Card Identifier*
ID	*Identity Document*
IETF	*Internet Engineering Task Force*
IMEI	*International Mobile Equipment Identity*
IMSI	*International Mobile Subscriber Identity*
IoT	*Internet of Things*
IoU	*Intersection over Union*
IP	*Internet Protocol*
IP	Índice de Proteção

ISM *Industrial, Scientific, and Medical*

IV Infravermelhos

IVA Imposto sobre Valor Acrescentado

JDE *Joint Detection and Embedding*

JSON *JavaScript Object Notation*

KLT *Kanade–Lucas–Tomasi*

LED *Light-Emitting Diode*

LiDAR *Light Detection and Ranging*

LIFO *Last In, First Out*

LoRa *Long Range*

LPDR *License Plate Detection and Recognition*

LPWAN *Low-Power-Wide-Area Networks*

LSTM *Long Short-Term Memory*

LTE *Long-Term Evolution*

LwT *Last Will and Testament*

M2M *Machine-to-Machine*

MAE *MovingAvgEstimator*

mAP *mean Average Precision*

MCUT *Multicut*

MIPI *Mobile Industry Processor Interface*

ML *Machine Learning*

MME Média Móvel Exponencial

MOT *Multiple Object Tracking*

MPe *Effective Megapixels*

MQTT *Message Queue Telemetry Transport*

MRCNN *Mask RCNN*

N/A *Not Available*

NB-IoT *NarrowBand-Internet of Things*

NMS *Non-Maximum Suppression*

NTP *Network Time Protocol*

NvDCF *NVIDIA-adapted Discriminative Correlation Filter*

OASIS Organization for the Advancement of Structured Information Standards

OMG *Object Management Group*

ONU Organização das Nações Unidas

OpenCV *Open Computer Vision*

PCB *Printed Circuit Board*

PGIE *Primary GPU Inference Engine*

PLA Poliácido Láctico

PoE *Power over Ethernet*

PSM *Power Saving Mode*

PTZ *Pan-Tilt-Zoom*

PWM *Pulse Width Modulation*

QoS *Quality of Service*

RAM *Random Access Memory*

RCNN *Region Based Convolutional Neural Network*

ReLU *Rectified Linear Unit*

ResNet *Residual Network*

REST *Representational State Transfer*

RGBA *Red-Green-Blue-Alpha*

RGPD Regulamento Geral sobre a Proteção de Dados

ROLO *Recurrent YOLO*

RTS *Request to Send*

SaaS *Software as a Service*

SBC *Single-Board Computer*

SCL *Serial Clock*

SD *Secure Digital*

SDA *Serial Data*

SDK *Software Development Kit*

SGIE *Secondary GPU Inference Engine*

SIM *Subscriber Identity Module*

SLAM *Simultaneous Localization And Mapping*

SN *Serial Number*

SO Sistema Operativo

SORT *Simple Online and Real-time Tracking*

SOT *Single Object Tracking*

SPI *Serial Peripheral Interface*

SSD *Single Shot MultiBox Detector*

SSH *Secure Shell*

SSL *Secure Sockets Layer*

SVM *Support Vector Machine*

SW *Software*

TCP *Transmission Control Protocol*

TIC Tecnologias da Informação e Comunicação

TLS *Transport Layer Security*

TMS *Transportation Management System*

TN *True Negative*

TOPS *Tera Operations Per Second*

TP *True Positive*

TPCAM *Traffic Pattern Collection and Analysis Model*

TPU *Tensor Processing Unit*

UART *Universal Asynchronous Receiver-Transmitter*

UCS *Universal Character Set*

UDP *User Datagram Protocol*

UE União Europeia

UHS *Ultra High-Speed*

UL *Uplink*

URC *Unsolicited Result Code*

URI *Uniform Resource Identifier*

USB *Universal Serial Bus*

UTF-8 UCS *Transformation Format 8*

VAS Sistema de Análise de Vídeo

VCA *Video Content Analysis*

VGA *Video Graphics Array*

VNC *Virtual Network Computing*

VPN *Virtual Private Network*

Wi-Fi *Wireless Fidelity*

YOLO *You Only Look Once*

Lista de Símbolos

Símbolo	Descrição	Unidades
F	frequência	Hz
g	ângulo	°
i	corrente	A
M	massa	kg
P	potência	W
R	resistência	Ω
T	temperatura	°C
t	tempo	s
v	tensão	V
x	dimensão	m

Capítulo 1

Introdução

Com o aumento da densidade populacional que reside nas áreas urbanas (atualmente mais de 50% da população mundial) e visto que, segundo a Organização das Nações Unidas (ONU), este valor tende a aumentar nos próximos anos (para 70% em 2050), torna-se um desafio para as cidades conseguir garantir aspetos como: sustentabilidade; controlo de emissões poluentes; qualidade de vida; níveis de população e mobilidade.

Como forma de combater estes problemas, surgiu o conceito de *smart city* que, apesar de não ter uma definição concreta, pode-se definir como um município que recorre a Tecnologias da Informação e Comunicação (TIC) como forma de aumentar a sua eficiência operacional, compartilhar informações com o público e melhorar tanto a qualidade de vida da população como dos serviços governamentais. O rápido desenvolvimento de novas tecnologias de sensores, processadores e comunicações levou ao aparecimento de sistemas mais eficientes e de baixo custo, capazes de recolher informações do meio urbano. O recurso à Inteligência Artificial (IA), que nos últimos tempos tem tido um desenvolvimento exponencial e promete substituir muita da tecnologia já existente, levando a uma cada vez menor intervenção humana.

O sistema descrito nesta dissertação foi desenvolvido no CEiiA - Centro de Engenharia e Desenvolvimento de Produto, no âmbito do projeto C-TECH. Este projeto tem como objetivo o desenvolvimento de soluções *low-cost* para *smart cities* no âmbito de *Internet of Things* (IoT). Esta organização trabalha no desenvolvimento, implementação e operação de soluções tecnológicas com os seus parceiros como forma de impulsionar a inovação em áreas como aeronáutica, mobilidade, automóvel, mar

e espaço. Em Portugal, o CEiiA é um dos 10 maiores investidores em Investigação e Desenvolvimento (I&D), sendo atualmente uma referência internacional na área de mobilidade sustentável e com reconhecimento no mundo da aeronáutica pelas suas competências em engenharia de estruturas.

O projeto C-TECH surgiu da necessidade de fazer a recolha e a análise de vários parâmetros ambientais, assim como da mobilidade no meio urbano. Posteriormente, pretende ajudar na diminuição do impacto ambiental, a longo prazo, nas grandes cidades. Desta forma, o CEiiA propõe dois produtos distintos de baixo custo, sendo um dedicado à recolha de dados ambientais e o outro a dados de mobilidade. Esta dissertação terá foco no estudo, desenvolvimento e testagem deste último, tendo em consideração o uso de métodos modernos na aquisição de dados, assim como a forma como estes são partilhados com a entidade responsável.

Como forma de atingir os objetivos deste projeto, será necessário proceder ao projeto e desenvolvimento de um sistema capaz de fazer a contagem de pessoas e veículos em determinadas zonas da cidade, reportando os dados de contagem para serem analisados, em tempo-real, por um operador remoto que ajudará a combater problemas das cidades como: gestão dos congestionamentos do tráfego; melhoria da rede semafórica; otimização da logística de transporte; previsão da congestionamentos em diferentes condições climatéricas; estudo da eficácia da rede de transportes públicos e, por fim, a longo prazo contribuir para a descarbonização das cidades. O sistema desenvolvido e apresentado nesta dissertação caracteriza-se por uma solução de visão computacional com modelos de *Deep Learning* (DL) e algoritmos de *tracking*, tendo sido validado o seu desempenho através de um protótipo aplicado num caso real, assim como a respetiva comunicação com um servidor remoto.

1.1 Objetivos

O sistema desenvolvido deve responder aos requisitos do projeto C-TECH. Segue a listagem dos mesmos:

- Desenvolver um sistema de baixo custo por forma a concorrer com produtos existentes no mercado com finalidades semelhantes, nomeadamente para aplicação em grande escala;
- Sistema com baixo consumo energético que, em grande escala, permitirá uma poupança significativa visto que têm um elevado e ininterrupto processamento de imagens de vídeo;
- Sistema autónomo, compacto e leve para a fácil instalação em infraestruturas existentes;

- Realizar, em tempo-real, a contagem de pessoas e veículos (nomeadamente de transportes públicos) ao longo do dia;
- Cumprimento dos termos do Regulamento Geral sobre a Proteção de Dados (RGPD) como forma de garantir a proteção de dados pessoais;
- Reportar, periodicamente, informação para um servidor com recurso a um módulo de comunicação do tipo IoT de forma segura;
- Apresentar possibilidade de receber, remotamente do servidor, instruções de reconfiguração e desativação temporária através do módulo de comunicação.

Face aos requisitos especificados para este produto, pretende-se apresentar uma proposta de arquitetura de baixo custo assim como baixo consumo energético comparado com os produtos já existentes no mercado. Por outro lado, pretende-se fazer a contagem de pessoas e veículos, em tempo-real, com recurso a algoritmos de IA otimizados face às potencialidades computacionais do sistema. Por fim, pretende-se validar a transmissão de dados de forma segura e o cumprimento do Regulamento Geral sobre a Proteção de Dados (RGPD).

1.2 Plano de Trabalho

Para organizar o desenvolvimento deste projeto, foi inicialmente traçado um plano de trabalho que se subdivide em sete fases: Análise do Estado da Arte; Planeamento da Arquitetura do Sistema; Implementação da Solução; Testes de Bancada; Testes de Campo; Apresentação dos Resultados e, por fim, a componente de escrita da Dissertação.

A Figura 1.1 representa, de forma cronológica, a distribuição de todas estas fases ao longo do período de desenvolvimento deste projeto, indicando o início, fim e duração de cada fase.

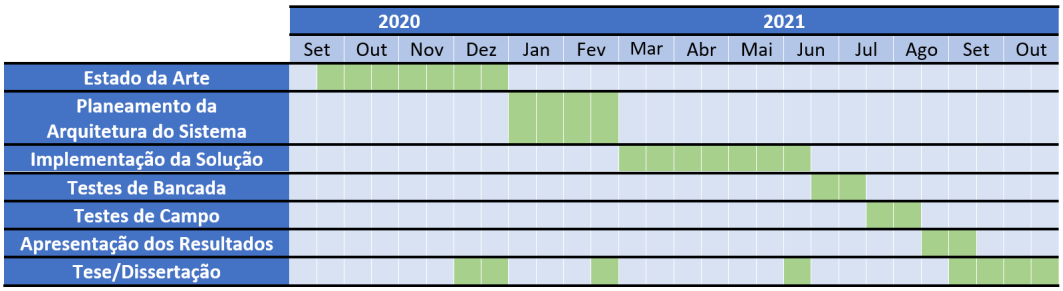


Figura 1.1: Representação da distribuição do Plano de Trabalho.

1.3 Organização da Dissertação

Esta seção apresenta a estrutura desta dissertação, que se encontra dividida em sete capítulos.

Após este capítulo de introdução, o segundo capítulo faz uma análise ao estado da arte, onde apresenta soluções existentes no mercado análise do tráfego em *smart cities* assim como soluções de comunicação IoT.

No terceiro capítulo são apresentados fundamentos teóricos, abordando conceitos relevantes para o posterior desenvolvimento do produto, tanto a nível de análise de dados como da forma como são partilhados.

No quarto capítulo, é descrita a fase de projeto onde se apresenta a arquitetura do sistema desenvolvido tanto a nível de *Hardware* (HW) como de *Software* (SW), tendo em consideração as soluções já existentes assim como o cumprimento dos objetivos inicialmente propostos.

O capítulo cinco aborda a implementação do sistema com a descrição detalhada das técnicas que foram aplicadas ao longo do seu desenvolvimento.

No capítulo seis apresentam-se e discutem-se os resultados dos testes feitos ao sistema desenvolvido.

Por fim, o capítulo sete apresenta as conclusões obtidas e apresentam-se propostas de trabalho futuro para melhorar o sistema proposto.

Capítulo 2

Estado da Arte

Este capítulo começa por fazer uma análise de produtos e plataformas existentes no mercado com características semelhantes aos resultados esperados por esta dissertação, e ainda uma apresentação de soluções de análise de tráfego do âmbito académico. Segue-se uma análise das vantagens e desvantagens dos principais, e mais recentes, equipamentos para aplicações de computação visual. Posteriormente, são apresentadas técnicas de deteção e *tracking* de objetos. Relativamente às comunicações IoT, são comparados os protocolos de comunicação e de mensagens mais comuns. Por fim, é feita uma abordagem ao RGPD.

2.1 Análise do Mercado

Atualmente, começam a existir cada vez mais produtos no mercado destinados à aquisição e tratamento de dados tanto do tráfego rodoviário como também do fluxo de pessoas nas zonas urbanas. Este tratamento tem vindo a recorrer significativamente tanto à visão computacional como a tecnologias de IA para a análise de informação fornecida por sistemas de visão. Desta forma, segue-se a apresentação de soluções com características semelhantes às pretendidas no projeto a desenvolver por esta dissertação. Esta apresentação divide-se em duas partes, sendo a primeira relativa a produtos que recolhem e tratam os dados do tráfego localmente (*edge-computing*) e em tempo-real (Seção 2.1.1), enquanto que a segunda se refere a

plataformas onde o tratamento dos dados é feito fora do local de aquisição (*cloud-computing*), isto é, remotamente, sendo estes dados processados em diferido (Seção 2.1.2).

2.1.1 Produtos

Relativamente aos produtos no mercado, é apresentada uma seleção daqueles que recorrem a tecnologias mais recentes e eficientes. A Figura 2.1 lista 7 produtos, sendo estes, respetivamente: Panomera S-Series; Panomera W-Series; DF5450HD-DN/IR; Autoscope Vision; Miovision TrafficLink; IoT-VT e TrafiCam x-stream2. De seguida, é feita uma breve descrição relativa a cada um dos produtos, finalizando com uma análise comparativa.

(a) Panomera S-Series [1]

(b) Panomera W-Series [1]

(c) DF5450HD-DN/IR [2]

(d) Autoscope Vision [3]

(e) Miovision TrafficLink [4]

(f) IoT-VT [5]

(g) TrafiCam x-stream2 [6]

Figura 2.1: Diferentes produtos no mercado.

- Panomera

Recentemente, a Dallmeier apresentou a tecnologia Panomera [1]. Esta consiste num conjunto de sensores multifocais, num só equipamento, onde todos focam a diferentes distâncias. Desta forma, consegue assegurar grande resolução em toda a imagem, ao contrário da tecnologia das câmaras convencionais em que a resolução, isto é, a densidade de pixeis, é reduzida para grandes distâncias. As câmaras da Panomera recorrem à análise de vídeo com suporte de IA para a deteção, em tempo-real, de movimentos e objetos com diferentes finalidades, como: contagem de pessoas e

veículos; classificação de objetos; reconhecimento facial e ocupação dos parques de estacionamento.

A tecnologia Panomera subdivide-se em duas séries: a S-Series (Figura 2.1a) indicada para cenários necessidade de foco até longas distâncias e a W-Series (Figura 2.1b) destinada a cenários com necessidade de uma vista panorâmica. A S-Series possui um dissipador de calor, assim como um sistema de limpeza de vidros com ar comprimido. Por outro lado, a W-Series possui um sistema de *cooling*, conseguindo uma refrigeração do sistema mais eficaz. Em comum, possuem: um sensor de luz ambiente; um filtro de Infravermelhos (IV); protocolos *Ethernet* como *Internet Protocol* (IP) v4, *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP) e requerem 6 Mb/s de velocidade mínima para *streaming*.

- DF5450HD-DN/IR

O DF5450HD-DN/IR (Figura 2.1c), também da Dallmeier, é um produto concebido para aquisição de imagens tanto de dia como de noite, graças ao seu modo IV [2]. Possui um sistema de *Video Content Analysis* (VCA), em tempo-real, responsável pela deteção de movimentos e de objetos, para além de recorrer à IA para melhoria dos processos de deteção e *tracking* de objetos. Este produto apresenta funcionalidades capazes de fazer a deteção de intrusos, assim como o cruzamento de objetos numa determinada zona. Quanto ao *streaming* de vídeo, este é enviado para a rede e, em caso de falha, é guardado localmente até ao reestabelecimento da ligação. Adicionalmente, este equipamento possui um motor, controlado remotamente, para a alteração do comprimento focal da lente.

- Autoscope Vision

O Autoscope Vision (Figura 2.1d), da Econolite, é uma solução integrada para deteção de veículos com base numa linha de contagem, denominada por *stop bar vehicle detection*, sendo também capaz de diferenciar bicicletas [3]. Este é capaz de calcular tanto a velocidade como o comprimento de cada veículo. Adicionalmente, possui um sistema para prevenção do embaciamento da lente, assim como o controlo remoto do *zoom* da lente. Um módulo externo gere a informação capturada e, posteriormente, partilha tanto dados como *streaming* de vídeo para aparelhos móveis, via *Wireless Fidelity* (Wi-Fi) ou *Ethernet*. Na Figura 2.2 é demonstrada a arquitetura deste sistema, sendo que o módulo Autoscope Vision Comm Manager disponibiliza, para além da comunicação local e remota, a alimentação da câmara. Adicionalmente, o módulo I/O 24 Module consiste numa interface para expansão de entradas e saídas por onde se pode adquirir informação de até 4 câmaras.

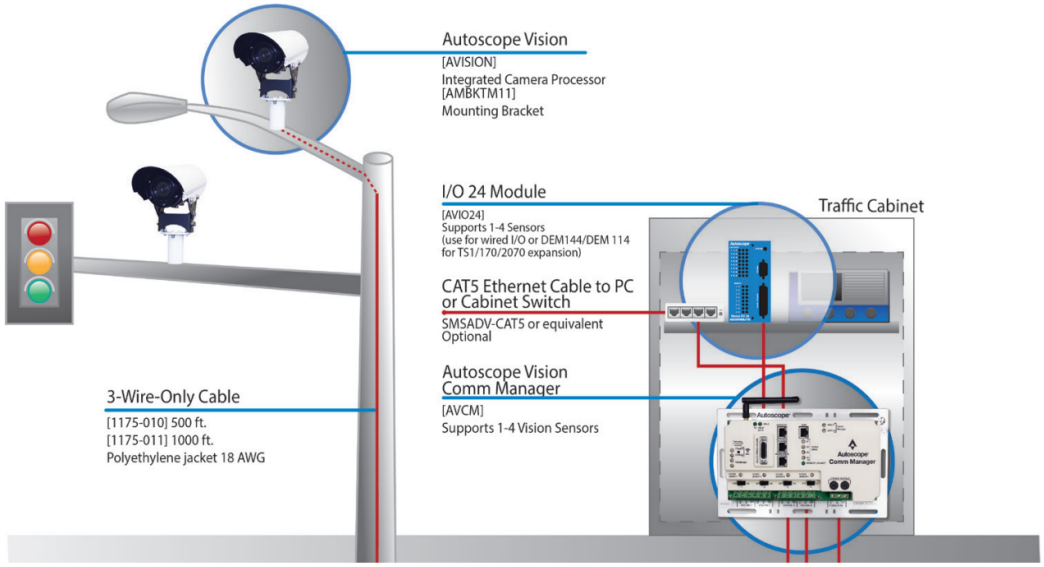


Figura 2.2: Arquitetura do Autoscope Vision [7].

- Miovision TrafficLink

O Miovision TrafficLink (Figura 2.1e) consiste num produto da Miovision que, com recurso à IA e em tempo-real, faz a contagem de pedestres, ciclistas e veículos, mesmo em condições de nevoeiro e precipitação [4]. Tem capacidades para deteção de estacionamento indevido; risco de segurança; incidentes de trânsito e estimação do comprimento das filas de trânsito. Este produto é constituído por três módulos: uma câmara (SmartView 360 camera); um módulo de conectividade via *Long-Term Evolution* (LTE) e, por fim, um processador (SmartSense) composto por 2 *Graphics Processing Unit* (GPU) e 1 *Central Process Unit* (CPU) de 12 núcleos.

Na Figura 2.3 está representada a arquitetura geral deste produto. Esta utiliza um módulo 4G/LTE para conectividade, sendo que, através de uma *Virtual Private Network* (VPN), acede de forma segura a uma *cloud* com ligação através de redes públicas. Assim sendo, o Miovision TrafficLink pode ser controlado remotamente por outro computador que esteja ligado à mesma VPN, assim como disponibilizar, através da *cloud*, os dados recolhidos para uma *Application Programming Interface* (API) [8]. Adicionalmente, toda esta comunicação é encriptada utilizando os protocolos *HyperText Transfer Protocol Secure* (HTTPS) e *Secure Sockets Layer* (SSL).

- IoT-VT

O IoT-VT (Figura 2.1f) é um produto da Real Sensing capaz de fazer a deteção de pedestres, bicicletas e veículos em tempo-real [5]. Esta informação é enviada via *wireless* para *brokers* que a processam e a tornam acessível. Este produto existe em dois formatos: com câmara IP ou analógica. Desta forma, é necessário avaliar a melhor câmara consoante o caso de aplicação, devido às vantagens e desvantagens de

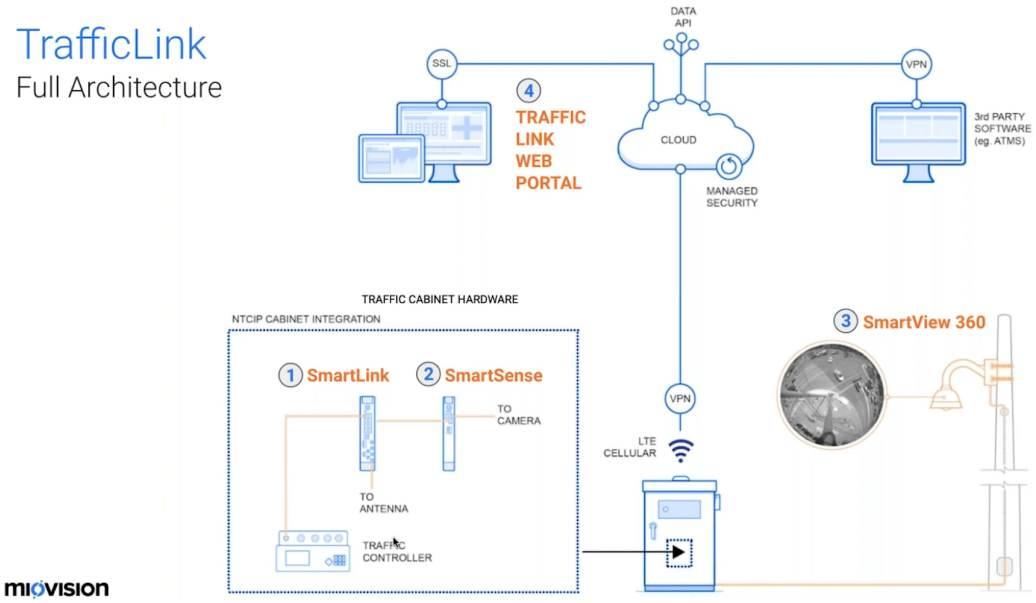


Figura 2.3: Arquitetura do Miovision TrafficLink [8].

cada uma [9]. Quanto ao processamento de imagem, este é feito localmente por um sistema de deteção denominado por *Video Turnstile* e, posteriormente, é enviado via *wireless* [10]. Os vários tipos de comunicações de que dispõe são: *Ethernet*, Wi-Fi, 3G/4G, LoRaWAN, SigFox e *Message Queue Telemetry Transport* (MQTT). Quanto ao preço unitário deste produto, é indicado que começa a partir dos 1000 €.

- TrafiCam x-stream2

O TrafiCam x-stream2, produto da Teledyne FLIR, consiste num sensor de presença de veículos constituído por uma câmara digital e um detetor de veículos, estejam estes em movimento ou parados, ao longo de múltiplas vias ou diferentes direções em zonas de interseção [6]. Transmite, em tempo-real, as informações de presença dos veículos para um controlador de tráfego, através de uma comunicação TCP/IP, com o intuito de ajudar na otimização do fluxo do tráfego local. A sua montagem é adaptável às infraestruturas existentes e a configuração do sensor pode ser feita no local ou remotamente. Quanto à alimentação e transmissão de dados, estes são feitos com recurso à tecnologia *Power over Ethernet* (PoE). Segundo [11], o preço unitário deste produto ronda os 2000 €.

- Análise Comparativa

Feita uma breve descrição do funcionamento e capacidades de cada um dos produtos, recorre-se também à Tabela A.1, disponível no Anexo A, onde agrega um conjunto de propriedades comuns entre estes produtos. De salientar que nesta tabela, os produtos da Panomera abordados consistem nas versões mais completas. Relativamente ao

produto IoT-VT, são indicados os dados com as diferentes câmaras, IP e analógica, respetivamente. No final desta tabela, é salientada uma observação diferenciadora que caracteriza o produto, positiva ou negativamente.

Procedendo à análise da Tabela A.1, percebe-se que existem produtos com características relativamente distintas, podendo fazer-se a distinção destes em seis classes:

- **Resolução de imagem**, sendo os produtos da Panomera os que apresentam uma resolução bastante superior, comparativamente aos restantes produtos. Esta superioridade justifica-se pelo facto de estes produtos estarem equipados com um maior número de câmaras;
- ***Field of View***, sendo o Panomera W-Series e o Miovision os produtos com um *Field of View* (FoV) mais amplo comparativamente aos outros produtos;
- **Alcance**, em que o Panomera S-Series e o Autoscope Vision são os que possuem um alcance significativamente maior comparativamente ao dos restantes produtos;
- **Consumo de energia**, sendo os produtos da Panomera que apresentam maior consumos energético, em grande parte devido à quantidade de câmaras e ao sistema de desembaciamento das lentes. Segue-se também o Miovision, visto os vários módulos instalados na cabine requererem um significativo consumo energético. Em relação aos restantes produtos, a informação do consumo energético não é clara ao ponto de perceber se é referente ao consumo de todo o sistema ou meramente da componente de aquisição de imagem;
- **Peso**, onde existem produtos com um peso significativo visto conterem uma cabine equipada com diversos módulos, como é o caso do Autoscope Vision e do Miovision TrafficLink, embora o Autoscope Vision apenas divulgue o peso de um dos módulos. Quanto aos outros produtos, são significativamente mais leves devido ao seu formato mais modular e compacto, como é o caso dos produtos da Panomera e do TrafiCam x-stream2;
- **Processamento de imagem**, sendo aparentemente apenas no IoT-VT que o processamento é feito remotamente, enquanto que nos restantes produtos este realizado no local de instalação.

Quanto às restantes características abordadas pela Tabela A.1, os Índice de Proteção (IP) são todos destinados a aplicações em ambiente *outdoor* (IP65-IP69). O mesmo se verifica para o *frame rate* da aquisição de imagens que ronda valores na gama dos 15 a 30 *Frames per Second* (FPS). Quanto à alimentação, esta é do tipo alternada, proveniente integralmente da rede elétrica local.

2.1.2 Plataformas

Para além dos produtos descritos anteriormente (Seção 2.1.1), existem plataformas que também se destinam à recolha de dados de tráfego e com diversas funcionalidades, como a contagem de pessoas e de veículos. Estas diferem dos produtos pelo facto de não estarem necessariamente responsáveis pela aquisição de imagens de vídeo em tempo-real. Segue-se a apresentação destas: GoodVision, my.SWISSTRAFFIC e UniTraffic. Na Figura 2.4 é mostrada a interface gráfica de cada uma destas após um processamento significativo de vídeos de análise de tráfego.

- GoodVision

A GoodVision (Figura 2.4a) consiste numa plataforma *Software as a Service* (SaaS) que, com recurso à IA, recolhe , de forma automática e encriptada, informações de análise de tráfego de diversas fontes [12]. Esta informação tanto pode advir de um vídeo em tempo-real como de um previamente adquirido. Esta análise é realizada numa *cloud*, sendo contabilizado o volume de tráfego, assim como a classificação dos objetos detetados até 8 classes distintas. Ao longo desta análise, os dados de vídeo são imediatamente eliminados e, no final, resulta um relatório detalhado do histórico do tráfego. Esta plataforma não necessita de HW adicional, sendo compatível com câmaras IP já instaladas.

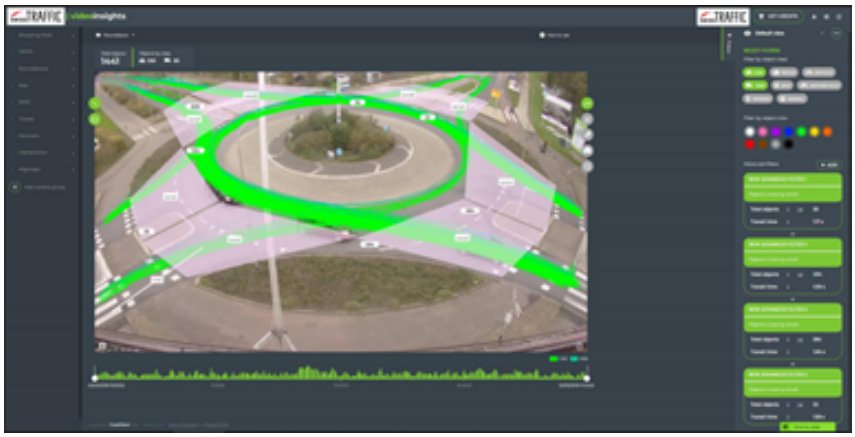
- my.SWISSTRAFFIC

A my.SWISSTRAFFIC (Figura 2.4b) é também uma plataforma SaaS destinada à análise de tráfego, com recurso à IA, sendo que adquire informação de qualquer câmara IP instalada [13]. Este processo pode ser em tempo-real ou em modo *offline*. Caso necessário, a empresa possui câmaras próprias que podem ser instaladas, permitindo a análise, em tempo-real, com recurso a um SW de IA embebido nestas. Neste processo é garantida a privacidade dos dados recolhidos. Na sua interface visual são representados dados como a contagem de volume de tráfego por categorias, assim como um relatório detalhado.

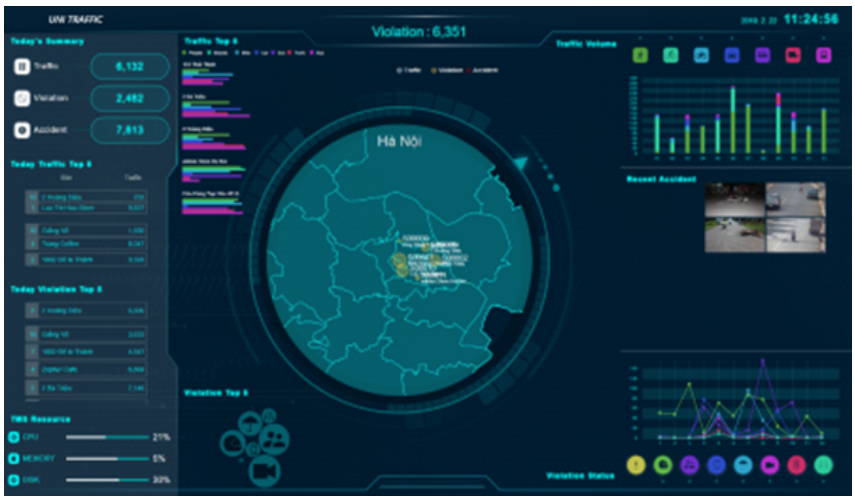
- UniTraffic

A UniTraffic (Figura 2.4c) consiste numa plataforma da UNISEM que recorre à IA para resolver problemas relacionados com o tráfego, em tempo-real, sem necessidade de equipamento adicional [14]. Trata-se de um Sistema de Análise de Vídeo (VAS) com capacidade de reconhecimento de matrículas, *License Plate Detection and Recognition* (LPDR), e gestão de frotas, *Transportation Management System* (TMS), que analisa imagens de câmaras *Closed-Circuit Television* (CCTV), reportando informações como: fluxo de tráfego; contagem de veículos; deteção de violações das regras de trânsito e reconhecimento de matrículas. Para a sua utilização, alguns

(a) GoodVision [12]



(b) my.SWISSTRAFFIC [13]



(c) UniTraffic [14]

Figura 2.4: Diferentes plataformas no mercado.

requisitos são exigidos tanto na configuração das câmaras como dos servidores VAS e TMS.

- Análise Comparativa

Para comparação destas plataformas, a Tabela 2.1 faz um resumo das principais características destas, incluindo o valor de mercado praticado. Todas fazem a deteção e classificação em várias classes de objetos em vídeo, assim como a apresentação de um relatório detalhado das mais variadas formas, como cronológica e gráfica. Todas permitem processamento em tempo-real, podendo o processamento ser no local de aquisição do vídeo, ou remotamente numa *cloud*. O valor deste serviço é cobrado consoante a quantidade de horas de processamento requeridas, excluindo os custo de aquisição e instalação dos equipamentos para aquisição, processamento e envio de dados.

Tabela 2.1: Características das plataformas no mercado [12, 13, 14].

Propriedades	GoodVision	my.SWISSTRAFFIC	UniTraffic
Tipo de Processamento	Tempo-Real/ Modo *Offline*	Tempo-Real/ Modo *Offline*	Apenas Tempo-Real
Local do Processamento	Remoto	Local & Remoto	Local & Remoto
Requisito de Equipamentos	Não	Não	Sim
Preço	199 €/mês	15 €/hora/mês	N/A
Observações	Só processa na *cloud*	Tem câmaras próprias	Com requisitos nos equipamentos

2.2 Soluções de Análise de Tráfego

No processo de monitorização do tráfego existem 2 tipos de sistemas, sendo um deles invasivo, visto interferir no meio, como é o caso de sistemas indutivos instalados sob o pavimento, acabando por exigir elevados custos de operação. Por outro lado, o tipo não invasivo, como é o caso de sistemas de câmaras, sensores *Light Detection and Ranging* (LiDAR) ou sensores radar, são de instalação fácil e económica, visto não necessitarem de intervenção no meio [15]. Desta forma, é imprescindível o levantamento de soluções tecnológicas modernas, autónomas, não invasivas e de baixo custo, nomeadamente modelos de visão computacional para a análise de tráfego no meio urbano.

Das soluções não invasivas, a utilização de câmaras acaba por ser a opção que permite recolher maior informação do meio. Por esta razão, soluções com base no tratamento de imagens têm vindo a ser amplamente estudadas e a apresentar grandes avanços a nível de performance.

Existem abordagens na análise de tráfego em que é feita a remoção do *background* da imagem, seguida da deteção de objetos com base no movimento, como a aplicada por W. AI Okaishi *et al.* [16], assim como a de K. Yang *et al.* [17]. Estas designam-se por métodos tradicionais, tendo por base a utilização de técnicas morfológicas. Em contrapartida, estes métodos estão limitados a situações onde o cenário é estático ou tem pequenas variações [18]. A performance das técnicas morfológicas em ambiente *outdoor* acaba por ser sensível às condições atmosféricas. Com o surgimento de métodos de DL, nomeadamente das redes *Convolutional Neural Networks* (CNN), verificaram-se melhorias no campo de visão computacional comparativamente aos métodos tradicionais.

Existem várias propostas para a análise de tráfego recorrendo a técnicas de DL, sendo que algumas têm em consideração a sua aplicabilidade em HW com limitações em termos de poder computacional. Segue-se a apresentação de alguns trabalhos desenvolvidos no âmbito académico que mais se enquadram no propósito desta dissertação:

- J. Barthélemy *et al.* [19] projetaram um dispositivo IoT do tipo *edge-computing* para *tracking* em tempo-real de pedestres, bicicletas e 4 tipos de outros veículos, garantindo a privacidade dos cidadãos. A sua aplicação encontra-se em condições de ser instalada tanto em ambiente *indoor* como *outdoor*. A aquisição dos dados é feita através de uma *framework* que utiliza o SW OM2M da Eclipse Foundation. Quanto ao HW, consiste num sistema embebido composto tanto por uma NVIDIA Jetson TX2, dedicada ao processamento das imagens fornecidas por câmaras CCTV existentes, como por um módulo Pycom LoPy 4 para comunicações com o exterior. O seu SW integra o algoritmo de deteção *You Only Look Once* (YOLO) v3 juntamente com o algoritmo de *tracking Simple Online and Real-time Tracking* (SORT). Relativamente aos dados resultantes, após processados, são transmitidos segundo o protocolo *Ethernet* ou LoRaWAN. Todo o sistema está integrado numa caixa de alumínio com IP67, sendo alimentado por uma fonte de alimentação de 35 W.

- A. Galletta *et al.* [11] propõem um dispositivo com o paradigma *Function as a Service* (FaaS) capaz de contabilizar veículos e fazer o reconhecimento de matrículas. O sistema é desenvolvido num Raspberry Pi Model 4, adquirindo imagens provenientes de uma câmara *Universal Serial Bus* (USB) com resolução 1920×1080 a 30 FPS. O SW foi desenvolvido na linguagem de programação *Python3* , recorrendo também à biblioteca *Open Computer Vision* (OpenCV) para aplicação de diferentes filtros, consoante a fase do dia. Para a análise do movimento dos veículos, são utilizados algoritmos de deteção de objetos em movimento, tendo por base a remoção do *background*. A informação resultante é guardada num ficheiro local, podendo este ser acedido segundo o protocolo

File Transfer Protocol (FTP). Neste desenvolvimento fez-se a comparação da sua performance com um sensor Traficam (abordado na Seção 2.1.1), apresentando melhor precisão na contagem durante o período do dia;

- D. Dinh *et al.* [20] fazem a introdução de um sistema *low-cost* do tipo *edge-computing* para deteção, *tracking* e contagem de veículos. Vários modelos de DL e métodos de *tracking* são testados tanto na NVIDIA Jetson Nano, como no Google *Dev Board*. Concluíram que a execução do modelo de deteção *Single Shot MultiBox Detector* (SSD) MobileDet, juntamente com o *tracker Deep* SORT, no Google *Dev Board*, apresenta a melhor relação entre precisão ($92,1\%$) e *frame rate* (26,8 FPS). Por outro lado, a Jetson Nano é compatível com uma maior gama de modelos, para além de ser cerca de 25% mais barata que o Google *Dev Board*;

- A. Santos *et al.* [15] propõem um sistema para contagem de veículos utilizando o detetor YOLOv3 com o *tracker Deep* SORT. Utilizando os *datasets* de vídeos de tráfego GRAM e CD2014, os resultados demonstram uma precisão de $99,15\%$. Posteriormente, fizeram a variação do nível de confiança do YOLOv3, assim como de um hiperparâmetro de associação no *tracker*, chegando aos valores de 70% e de 7, respetivamente, como valores ótimos. Com esta configuração, conseguiu-se uma precisão de 90% num cenário real de contagem de tráfego no Brasil. Para a validação deste sistema foi utilizado um computador com as seguintes especificações: Intel Core i5-8300H, NVIDIA GeForce GTX 1050 com 4 GB DDR5 e 8 GB DDR4 de *Random Access Memory* (RAM);

- H. Zhang *et al.* [21] projetaram um sistema com algumas câmaras *Pan-Tilt-Zoom* (PTZ) e um servidor local Xilinx Zynq-7000 *Dev Board*. Este processa as imagens das câmaras com recurso a um modelo pré-treinado, denominado *Support Vector Machine* (SVM), para a deteção de pessoas e veículos na via. O *tracker* que criaram, foi desenvolvido em C++, tendo sido baseado no *tracker Background-Aware Correlation Filters* (BACF), conseguindo apresentar um bom nível de precisão, comparativamente a métodos tradicionais de *tracking*. Este *tracker* destaca-se por exigir pouco poder computacional, nomeadamente pouco espaço de memória, na ordem dos kB, face aos *trackers* baseados em redes CNN que requerem cerca de 300 MB. O processamento de imagens 1080p demora menos de 25 ms. Posteriormente, os dados processados são enviados para uma Google *cloud*, utilizando o Cloud IoT Core. Adicionalmente, o sistema tem um servidor *web* para interface com o utilizador;

- K. U. Sreekumar *et al.* [22] apresentam um modelo inteligente de análise de padrões de tráfego denominado *Traffic Pattern Collection and Analysis Model*

(TPCAM). O modelo, através de uma câmara, deteta e calcula do fluxo de tráfego de veículos e pedestres em tempo-real. Um novo modelo de DL, baseado no YOLOv2, é proposto, sendo adaptado para cenários de deteção de tráfego. O processamento de imagem é feito localmente para evitar a transmissão de grande informação para a *cloud*. Para melhorias de performance, é proposto um *deep tracker* de objetos, robusto a oclusões e variações de luminosidade. Com isto, conseguiram uma melhoria da performance de 3 FPS para pseudo-30 FPS numa NVIDIA Jetson TX2;

- C. Chen *et al.* [23] propõem um sistema de análise de fluxo de tráfego, sendo o detetor de veículos, baseado no YOLOv3, treinado e otimizado com *datasets* de tráfego. Para *tracking* de múltiplos veículos é utilizado o *Deep* SORT, possibilitando, desta forma, a contagem de veículos e a análise do fluxo. A implementação deste sistema foi feito numa NVIDIA Jetson TX2, tendo sido obtidos valores de velocidade de processamento na ordem dos 38 FPS com 92% de precisão;

- S. Yang *et al.* [24] apresentam testes com o *dataset* COSMOS, utilizando uma câmara *bird's eye*. É analisada a precisão da deteção e *tracking* tanto de pedestres como de veículos, recorrendo a algoritmos de DL em tempo-real. Com a utilização de infraestruturas com este tipo de câmaras é garantida uma maior privacidade, visto que o perfil dos pedestres é apresentada com baixa resolução e as matrículas dos veículos não é visível. Os resultados demonstram dificuldades no *tracking* de pedestres, comparativamente com os veículos, devido a razões de escala. Estes testes recorreram a vários algoritmos de deteção (YOLOv3, SSD e *Mask RCNN* (MRCNN)) e de *tracking* (*Deep Affinity Networks* (DAN), *Deep* SORT e *Multicut* (MCUT)), tendo-se observado performances distintas, tanto ao nível de precisão como do tempo de processamento. Por outro lado, afirmam que o *frame rate* aceitável para *tracking* de veículos é de 30 FPS, embora o melhor *frame rate* que conseguiram na deteção foi de 11,8 FPS, com o YOLOv3, e de 3,2 FPS no *tracking*, com o *tracker Deep* SORT;

- A. Marchetti [25] focou-se no desenvolvimento de uma solução de IA para análise de vídeo em tempo-real, recorrendo a uma câmara IP. A plataforma escolhida foi a NVIDIA Jetson TX2. A *framework* foi implementada em *Python*, por questões de rápida prototipagem, recorrendo às bibliotecas OpenCV e TensorFlow. Utilizou o algoritmo de deteção SSD, com o *backbone* MobileNet v2, treinado com o *dataset* UA-DETRAC. Para o *tracking* de veículos, recorreu ao algoritmo *Intersection over Union* (IoU) e, posteriormente, aplicado um contador de veículos. Por fim, os dados resultantes foram enviados para uma base de dados externa, segundo o protocolo MQTT, utilizando a biblioteca

Paho MQTT *client*. O sistema funcionou a 19 FPS com um erro de contagem bastante reduzido;

- Y. Yang *et al.* [26] abordam um método de contagem de pedestres, tendo por base imagens de vídeo em tempo-real. Para a deteção, foi implementado o YOLOv3, tendo sido treinado com imagens reais de pedestres e otimizado com o método de *clustering K-Means*. Quanto ao *tracking*, foi escolhido o algoritmo *Deep* SORT e, por fim, o processo de contagem foi feito segundo uma linha virtual de contagem. Recorrendo a um *dataset* obtido com uma câmara Sony IMX345, foi testada a performance deste método num computador com as seguintes características: CPU AMD R7 2700 e GPU GeForce GTX1070 Ti. Os resultados apresentam $89,2\%$ de precisão na contagem de pedestres a um *frame rate* de 15 FPS, em tempo-real.

Estas soluções apresentam, nomeadamente a utilização de: sistemas de baixo poder computacional para *edge-computing*; câmaras IP e USB com resolução máxima de 1080p; comunicação IoT com recurso ao LoRa e MQTT; modelos de DL para deteção em tempo-real; algoritmos de *tracking* para contagem dos objetos em tempo-real; e, por fim, o desenvolvimento nas linguagens de programação C++ e *Python*. Em certas soluções, apenas é validado o sistema em bancada, isto é, em computadores de maior poder computacional, com recurso a *datasets* de tráfego. De qualquer forma, resultados revelam performances na ordem dos 5-40 FPS com cerca de $89,2\%$ de precisão.

Relativamente à análise de tráfego com sistemas de visão, a baixa qualidade dos dados (*datasets*), como também dos modelos e plataformas do tipo *edge computing*, são impedimentos para obter informação de qualidade sobre o tráfego [27]. Desta forma, o AI City Challenge foi criado com o objetivo de promover o desenvolvimento de soluções inteligentes para análise de vídeo em *smart cities*, assim como avaliar a performance de tarefas específicas para possíveis implementações no mundo real [28]. Este evento encontra-se já na 5ª edição, contando com a participação de 305 equipas para 5 tipos de desafios de *tracking*, sendo o primeiro dedicado à utilização de dispositivos IoT para contagem de veículos, de vários tipos e com diferentes trajetórias, num conjunto de *datasets* de vídeos de tráfego.

2.3 Equipamentos para Computação Visual

Tendo também por base a informação da Seção 2.2, segue-se um *trade-off* relativo aos principais equipamentos utilizados num sistema com visão computacional, sendo analisadas as vantagens e desvantagens destes. Estes consistem, maioritariamente, em *Single-Board Computer* (SBC) e câmaras. Assim sendo, é feito um estudo destes

componentes, identificando quais os mais utilizados nos dias de hoje, assim como os casos de aplicação particulares de cada um.

2.3.1 SBC

Um *Single-Board Computer* (SBC) consiste num computador de pequenas dimensões, cujos componentes eletrónicos se encontram integrados numa única *Printed Circuit Board* (PCB). Estes elementos consistem em: microprocessadores; memórias; pinos *Input/Output* (I/O); interfaces diversas; entre outros periféricos. Estes computadores destinam-se a várias aplicações, desde o âmbito educacional até a aplicações industriais, como sistemas embebidos para tarefas de controlo.

Apesar de existirem SBC para bastantes aplicações, a Figura 2.5 apresenta três SBC de baixo custo que se adequam a aplicações tanto de visão computacional como de IA, sendo estes: Raspberry Pi 4 Modelo B; NVIDIA Jetson Nano *Developer Kit* e Google Edge *Tensor Processing Unit* (TPU) Coral *Dev Board*, respetivamente.

(a) Raspberry Pi 4 Modelo B [29]

(b) NVIDIA Jetson Nano *Developer Kit* [30]

(c) Google Edge TPU Coral *Dev Board* [31]

Figura 2.5: Diferentes SBC no mercado para aplicações de visão computacional e IA.

O Raspberry Pi 4 Modelo B (Figura 2.5a) é o modelo mais recente das Raspberry Pi, sendo esta a terceira marca *best-seller* de computadores no mundo [29]. Trata-se de um computador destinado, tanto para adultos como para crianças, visto poder ser usado quer na aprendizagem de programação, quer para a criação de projetos de eletrónica. Este novo modelo foi lançado em 2019, apresentando melhor desempenho comparativamente aos modelos anteriores da Raspberry Pi. Comparativamente ao Raspberry Pi 3, resultados de *benchmarks* comprovam significativas melhorias [32, 33]: performance do CPU até 4× mais rápida; performance do GPU com 50% mais FPS; velocidade de leitura e escrita na memória de 9× mais rápida via USB; e alternância nos pinos *General Purpose Input/Output* (GPIO) 3× mais rápida (50 kHz). Em contrapartida, os seus consumos energéticos são relativamente superiores, fruto destas melhorias o Raspberry Pi 4 Modelo B consumiu 7,5 W contra os 5,4 W do seu antecessor, correspondendo a um aumento de 40%.

A NVIDIA Jetson Nano *Developer Kit* (Figura 2.5b) é um pequeno e potente computador desenvolvido pela NVIDIA [30]. Permite que múltiplas redes neuronais

funcionem paralelamente em aplicações de classificação de imagens, deteção de objetos e segmentação. Trata-se de uma plataforma capaz de funcionar a apenas 5 W, sendo de fácil utilização para iniciantes na área de IA e visão computacional, visto que reduz a complexidade no desenvolvimento de SW de IA. Trata-se do elemento mais barato e menos potente da família Jetson da NVIDIA, disponibilizando 472 GFLOPS de performance computacional, enquanto que os seus sucessores Jetson TX2 e Xavier NX possuem tanto maior poder computacional como consumo energético. A NVIDIA possui uma comunidade de utilizadores que tem vindo a crescer bastante, disponibilizando projetos, ferramentas e tutoriais acessíveis para todos [34]. Tendo por base uma série de *benchmarks* da Jetson Nano, comparativamente aos seus sucessores [35, 36, 37, 38, 39, 40], a Jetson Nano é a que apresenta menores consumos energéticos sem comprometer a sua capacidade de computação ao executar a maioria das redes neuronais, sendo, de momento, das mais utilizadas. F. Serzhenko [35] realizou uma *pipeline* de testes *standard*, consistindo no processamento de imagens para aplicações que recebessem imagens de resolução 2K diretamente de câmaras. S. Weiss [36] realizou testes ao utilizar o detetor de objetos MobileNet SSD v2, otimizado pelo TensorRT, *Software Development Kit* (SDK) da NVIDIA para inferência de DL. D. Franklin [37] apresenta o desempenho da Jetson Nano no *streaming* de vídeo *High-Definition* (HD) com diversos modelos de DL. J. Jo *et al.* [38] testam os produtos com uma variedade de redes CNN em tempo-real. S. Ullah e D. Kim [39] fazem testes de classificação de *point clouds 3-Dimensional* (3D), assim como de imagens hiperespectrais, onde obtiveram tempos de inferência e consumos energéticos. Por fim, T. Peng *et al.* [40] testam a performance e os consumos com aplicações de *Simultaneous Localization And Mapping* (SLAM) em robôs móveis.

O Google Edge TPU Coral *Dev Board* (Figura 2.5c) consiste num SBC de prototipagem para sistemas embebidos, incluindo aplicações de *Machine Learning* (ML) [31]. Possui um coprocessador, TPU, que consiste num acelerador de redes neuronais para ML [41]. Desta forma, este SBC consegue um desempenho de 4 *Tera Operations Per Second* (TOPS) com um consumo de 0,5 W/TOPS. É eficiente com modelos de visão como [42], por exemplo, o MobileNet v2 a cerca de 400 FPS [31]. Faz uso do TensorFlow *Lite* para suporte de modelos de DL existentes. A nível de GPU, apresenta o mesmo número de núcleos (128) que a Jetson Nano [43].

Na Tabela 2.2 é feito o levantamento das características mais relevantes, relativamente a cada um destes SBC, para efeitos de comparação, para além do destaque de uma observação, positiva ou negativamente, relativamente a cada um destes.

Para determinar que SBC melhor se enquadra para cada tipo de aplicação, existem uma série de *benchmarks* que resultam de diversos testes feitos. R. K. Gupta e S. D. Senturia [41] chegaram à conclusão que o Google Coral se destacou com o melhor tempo de inferência e consumo energético, embora a Jetson Nano supere

Tabela 2.2: Especificações técnicas dos SBC [29, 30, 31].

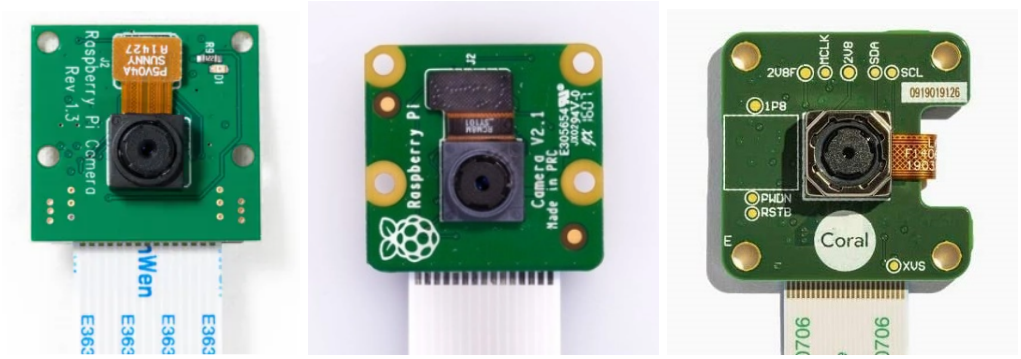
Propriedades	Raspberry Pi 4 Modelo B	NVIDIA Jetson Nano *Developer Kit*	Google Coral *Dev Board*
CPU	Quad-core 64-*bit* ARM A72 a 1,5 GHz	Quad-core 64-*bit* ARM A57 a 1,43 GHz	Quad-core Cortex-A53 e Cortex-M4F
GPU	VideoCore VI 3D Graphics	Maxwell de 128 núcleos	Integrated GC7000 Lite Graphics
RAM	4 GB LPDDR4		
Consumo	5-10 W	5-10 W	10-15 W
Câmara	1× MIPI CSI-2	2× MIPI CSI-2	1× MIPI CSI-2
Dimensões	85×49×17 mm	100×80×29 mm	88×60×24 mm
USB	2× USB 3.0 2× USB 2.0	4× USB 3.0 Micro-USB 2.0	2× USB 3.0 USB 2.0
Periféricos	GPIO, I2C, I2S, SPI, UART		
Preço	50,00 €	109,00 €	140,00 €
SO	Raspbian/Ubuntu	Jetpack (Ubuntu 18.04)	Mendel Linux
Observações	Suporte da Comunidade	Dissipador incluído & Boa qualidade-preço	Coprocessador TPU

com o menor tempo de computação (29, 3%) ao utilizar o modelo MobileNetV2. M. Antonini *et al.* [44] revelam que, ao contrário do Google Coral, a Jetson Nano faz uma alocação mais significativa da memória ao executar o TensorFlow GPU, resultante de problemas de otimização. Assim sendo, poderá implicar a falta de memória ao executar outros processos em simultâneo. Na execução de modelos de redes neuronais, o Google Coral apresenta tempos de inferência significativamente menores, em comparação com a Jetson Nano. Desta forma, o Google Coral é sugerido para sistemas reativos em que vários modelos tenham de ser executados alternadamente. Já a Jetson Nano é mais indicada para a utilização de um único modelo por longos períodos de tempo. A nível energético, a Jetson Nano consome quase 50% menos em modo *idle*, comparando com o Google Coral. Por outro lado, a execução de modelos com o TensorFlow GPU, demonstra um consumo energético 10× superior na Jetson Nano, comparado com o do Google Coral. Para reduzir significativamente este consumo na Jetson Nano, é indicado utilizar o TensorRT da NVIDIA em vez do TensorFlow GPU. Por fim, D. Franklin [37] apresenta resultados de um *benchmark* com estes 3 SBC com algumas das redes de DL mais populares. Conclui que a Jetson Nano é a que apresenta um SW mais flexível para executar todas estas redes com imagens HD que provenham, simultâneamente, de múltiplos sensores de *streaming*. Nos restantes SBC surgem problemas como limitações da capacidade de memória e falta de suporte a determinadas camadas das redes de DL que foram testadas.

2.3.2 Câmaras

Relativamente a câmaras para sistemas de visão computacional, verifica-se, segundo as Seções 2.1.1 e 2.2, que os produtos e soluções de análise de tráfego recorrem a informação proveniente de câmaras *2-Dimensional* (2D). Consoante o caso de aplicação, existem outros aspetos a considerar, como: custo; facilidade de configuração; dimensões; consumo energético e resolução da imagem.

No mercado existem módulos de câmaras (*camera modules*), que consistem em *boards* destinadas a aplicações de sistemas embebidos. Cada módulo integra [45]: um sensor de imagem; uma lente; um controlo eletrónico e uma interface *Camera Serial Interface* (CSI) ou *Ethernet*. Tratam-se de módulos de tamanho reduzido e de baixo custo, comparativamente com as câmaras CCTV de videovigilância. Na Figura 2.6 são apresentados os módulos de câmaras de fácil integração, nomeadamente nos SBC abordados (Seção 2.3.1). Estas câmaras são as Raspberry Pi Camera Modules v1 e v2, da Raspberry Pi, cuja 2ª versão já é compatível com a Jetson Nano e a Coral Camera desenvolvida pela Coral.



(a) Raspberry Pi Camera Module v1 [46]

(b) Raspberry Pi Camera Module v2 [47]

(c) Coral Camera [48]

Figura 2.6: Diferentes módulos de câmaras existentes no mercado para aplicações de visão computacional.

Na Tabela 2.3 é feito o levantamento das características mais relevantes de cada uma das 3 câmaras abordadas.

2.4 Técnicas de Deteção de Objetos

Nos últimos tempos, as técnicas de DL têm vindo a dominar significativamente a vertente de processamento de imagem digital. Assim, tem-se verificado uma ligeira diminuição na utilização de técnicas tradicionais de visão computacional, embora muitas ainda sejam aplicadas [50]. Como consequência, têm-se desenvolvido metodologias que combinam ambas as técnicas, tanto para efeitos de melhoria de performance, como para resolução de problemas de otimização presentes nas técnicas de DL. Resultados desta combinação demonstram uma melhoria de 10× no *frame rate*,

Tabela 2.3: Especificações técnicas das câmaras [47, 48, 49].

Propriedades	Raspberry Pi Camera Module v1	Raspberry Pi Camera Module v2	Coral Camera
Sensor	OmniVision OV5647	Sony IMX219	Omnivision OV5645
Resolução	5 MP (2592×1944 píxeis)	8 MP (3280×2464 píxeis)	5 MP (2582×1933 píxeis)
Vídeo	1080p30, 720p60, 640×480p60/90	1080p30, 720p60, 640×480p60/90	1080p30, 720p60
Foco	1 m - ∞	8 cm - ∞	10 cm - ∞
Interface	CSI	CSI	CSI
FoV	53,5°/41,41°	62,2°/48,8°	84,0°/87,4°
Preço	15 €	31 €	24 €

comparativamente com soluções exclusivamente de DL. Por um lado, o DL oferece maior flexibilidade e rapidez, visto que os modelos CNN podem ser retreinados com *datasets* personalizados consoante o caso de aplicação. Por outro lado, as técnicas tradicionais podem resolver certos problemas com grande precisão com recurso a poucas linhas de código, face ao moroso retreinar da rede neuronal [51].

Quanto aos modelos de DL, o tipo de rede neuronal mais utilizada para deteção e reconhecimento de objetos é a CNN [52]. Estes modelos de deteção podem-se dividir em 2 famílias: *two-stage* e *one-stage*. O primeiro tende a demonstrar maior precisão, embora requeira um maior poder computacional, enquanto que o segundo apresenta uma maior inferência e menor complexidade, sendo o mais indicado para casos de aplicação em tempo-real [53]. Relativamente ao conceito de inferência, este consiste num processo que utiliza modelos *Deep Neural Network* (DNN) treinados para fazer previsões com base em dados desconhecidos [54], ao contrário do processo de treino (*training*) que ensina a DNN a desempenhar certas tarefas de IA com base em dados conhecidos (*datasets*). Em contrapartida, os modelos *one-stage* não apresentam a rede *region proposal* existente nos modelos *two-stage* que prevê *bounding boxes* de objetos e lhes atribui uma determinada pontuação. Desta forma, os modelos *one-stage* prevê diretamente as *bounding boxes* e as respetivas suas classificações sem propor regiões [55, 53].

Atualmente, e tendo em conta os algoritmos de deteção abordados na Seção 2.2, o estado da arte dos detetores *one-stage* mais representativos são os seguintes: *You Only Look Once* (YOLO) e *Single Shot MultiBox Detector* (SSD). Estes algoritmos apresentam diversas variantes, diferindo em aspetos como a resolução de imagem de entrada e o *backbone* (*feature extractor*). Relativamente aos *backbones*, os mais utilizados são [53]: *Residual Network* (ResNet), DarkNet e MobileNet. Segue-se agora uma breve introdução aos detetores YOLO e SSD.

2.4.1 YOLO

O *You Only Look Once* (YOLO) consiste num detetor de objetos *one-stage*, publicado em 2015. Surgiu após o *Faster* RCNN, um detetor este do tipo *two-stage* que implementa uma *Region Based Convolutional Neural Network* (RCNN), apresentando melhor precisão da deteção (mAP) e menor tempo de execução [56]. O YOLO destina-se a deteções em tempo-real, adquirindo apenas 100 *region proposals*, 3× menos que o *Faster* RCNN. Por outro lado, o YOLO apresenta uma velocidade de 45 FPS, contra os 5 FPS do *Faster* RCNN. Esta rede utiliza *features* de toda a imagem para, simultâneamente, fazer a previsão de cada *bounding box* e respetiva classe [57]. A rede é composta por apenas 24 camadas de convolução e 2 *fully connected*, afetando a precisão da localização dos objetos [56]. Em contrapartida, o YOLO consegue reduzir em 3× os falsos positivos, em comparação com o *Region Based Convolutional Neural Network* (RCNN).

Já o YOLOv2 consiste numa melhoria do YOLO, tanto ao nível da precisão na deteção, como da velocidade de execução [56]. Como melhoria do *backbone*, este utiliza o Darknet-19 [58]. Adicionalmente, introduziram-se camadas *batch normalization* que aumentaram em 2% a precisão [56]. Com o aumento da resolução da entrada do *backbone* (224×224 para 448×448 píxeis), incluindo alguns ajustes na rede, o valor *mean Average Precision* (mAP) aumentou 4%. A extração de *features* adicionais na menor resolução (224×224) é mantida, aumentando em 1% o mAP. A utilização de *anchor boxes* permite prever a classe, assim como o *score* de cada *bounding box*, melhorando em 7% o *recall*. A sua precisão foi melhorada com a introdução de um método de *clustering* para a previsão do centro de cada *anchor box*. Por fim, adicionou-se o treino da rede em múltiplas escalas, tornando-a mais robusta para imagens de diferentes dimensões.

Quanto ao YOLOv3, este consiste numa versão melhorada do YOLOv2, utilizando como *backbone* o DarkNet-53. Esta versão permite a classificação de múltiplas classes, ideal para cenários mais complexos com objetos de pequenas dimensões. No processo de treino, recolhe *features* a diferentes escalas para previsão das *bounding boxes*. Em contrapartida, esta versão apresenta menor performance para objetos de médias e grandes dimensões.

Por fim, o YOLOv4 é uma versão melhorada do YOLOv3, apresentando uma melhoria do mAP em 10%, assim como 12% nos FPS [59]. O *backbone* passa a ser o CSPDarknet53, onde se faz a separação da camada atual para que uma parte não passe pela camada de convolução, minimizando desta forma a perda de informação ao longo das camadas. Adicionalmente, implementa outras técnicas para melhorar a precisão, durante e após o processo de treino, como: *data augmentation* e *Non-Maximum Suppression* (NMS).

2.4.2 SSD

O *Single Shot MultiBox Detector* (SSD) é um detetor capaz de prever múltiplas classes, sendo que em cada camada são criadas *feature maps* com diferentes escalas [56]. Posteriormente, o SSD deteta objetos em imagens com base numa discretização (*downsamplings*) das *bounding boxes* para conjuntos de *boxes* de diferentes escalas e rácios, em cada *feature map* [60]. De seguida, é determina a probabilidade da presença de cada classe em cada *bounding box*, sem previamente conhecer se esta contém algum objeto [56]. Como *backbone*, este detetor utiliza o modelo VGG-16 com alguns ajustes, para detetar em vez de classificar [61]. Assim sendo, o SSD é capaz de detetar objetos de diversas dimensões através da combinação de previsões de vários *feature maps* a diferentes resoluções [60]. No seu processo de treino efetua uma interseção entre as *bounding boxes* resultantes e as *ground truth* para, posteriormente, selecionar apenas as *bounding boxes* com correspondência [56].

2.5 Técnicas de *Tracking* de Objetos

O conceito de *Multiple Object Tracking* (MOT) consiste num processo computacional de análise de vídeo que identifica e realiza o *tracking* de múltiplos objetos. À medida que estes se movimentam, é atribuído o mesmo *Identity Document* (ID) às *bounding boxes* que contêm o mesmo objeto, ao longo de sucessivas *frames*. Adicionalmente, o *Multiple Object Tracking* (MOT) é capaz de fazer a distinção de objetos da mesma classe ao longo do tempo [62]. Enquanto que no processo *Single Object Tracking* (SOT), o objeto é previamente reconhecido, no MOT é necessário detetar os vários objetos, dificultando o *tracking*, nomeadamente em momentos de oclusão ou até em casos de interseção entre objetos semelhantes. Para lidar com estes desafios, foram propostos alguns métodos baseados em DL para realizar a extração automática de *features*, tanto ao nível da aparências como do movimento dos objetos [63]. Na maioria da aplicações de análise inteligente de vídeo, o MOT é um elemento chave para satisfazer o requisito de análise temporal em relação às mudanças de estado dos objetos.

Os algoritmos MOT podem-se classificar em métodos *offline* ou *online* [64]. O primeiro tipo utiliza, geralmente, resultados de deteção de *frames* anteriores e futuras para identificação do objeto ao longo todas as *frames*, resultando numa boa qualidade do processo de *tracking*. Por outro lado, o método *online* estima a trajetória dos objetos apenas com base nos dados de deteção atuais e anteriores, tornando este método adequado para aplicações em tempo-real, embora com menor qualidade.

Relativamente aos algoritmos MOT do tipo *online*, grande quantidade consiste em variantes de algoritmos, sendo que existem ligeiras modificações na sua arquitectura, com o intuito de melhor se enquadrarem com um determinado caso de aplicação. S. Chen *et al.* [63] apresentam um *benchmark* de diversos algoritmos de

MOT, comparando as vantagens e o nível de robustez de cada um com recurso a *datasets* de MOT. V. Mandal *et al.* [65] abordam uma análise da melhor combinação entre algoritmos de DL para deteção e algoritmos de *tracking* para contagem de tráfego. Contudo, existem algoritmos que recentemente se têm destacado, tendo sido implementados e estudados mais intensivamente devido à sua eficiência e baixo custo computacional, designadamente: o SORT e o *Deep* SORT [65, 66]. Alternativamente, existem [67, 68]: o Tracktor++ com boa precisão, embora não seja indicado para aplicações em tempo-real; o TrackR-CNN que, embora não seja adequado a este tipo de aplicação, apresenta a capacidade de segmentação dos objetos; o *Joint Detection and Embedding* (JDE) que é capaz de funcionar em tempo-real, embora o *Deep* SORT apresente 25% mais FPS e diminuía a performance quando treinado com imagens *Full* HD; e o *Recurrent YOLO* (ROLO), que utiliza o YOLO como detetor e a rede *Long Short-Term Memory* (LSTM) para determinar a trajetória dos objetos. Contudo tem sido superado por outros *trackers* baseados em LSTM [69].

2.5.1 SORT

O *Simple Online and Real-time Tracking* (SORT) foi criado para aplicações de MOT, sendo possível a sua utilização em aplicações em tempo-real, graças à sua simplicidade [66]. Uma vez que o SORT apenas recorre às informações de deteção da *frame* atual e da *frame* anterior [65], utiliza algoritmos de deteção de objetos, baseados na rede CNN.

A cada nova *frame* de vídeo, o SORT faz, para a *frame* atual, a propagação de informação dos objetos até então *tracked*. O processo de previsão das novas posições é baseada num estimador, denominado Filtro de *Kalman*, com um modelo de velocidade constante. Feita a previsão, são detetados possíveis novos objetos na *frame* atual e comparados com os objetos *tracked*. Na fase de correspondência entre objetos de *frames* consecutivas, é analisado o grau de interseção das *bounding boxes*, isto é, o nível de *Intersection over Union*, e, posteriormente, associados segundo o método Hungarian. Caso um objeto não seja associado, é criado um novo objeto *tracked*. Durante o seu funcionamento, nenhuma informação é memorizada, sendo que cada objeto *tracked* é perdido entre *frames* consecutivas.

2.5.2 *Deep* SORT

O *Deep* SORT é uma extensão do SORT que pretende diminuir a troca de identidades [66], permitindo que o *tracking* dos objetos seja conseguido durante um maior período de tempo, mesmo perante períodos de oclusão [70]. Embora o procedimento seja relativamente semelhante ao do SORT, nomeadamente a utilização do Filtro de *Kalman* para estimação do estado, recorre a técnicas adicionais para

fazer corresponder novas deteções a objetos *tracked*. Uma destas é a Distância de *Mahalanobis* que não efetua a correspondência entre objetos de *frames* consecutivos quando a distância não é razoável. Outra técnica baseia-se em descritores de aparência que permitem ao *Deep* SORT reencontrar antigos objetos *tracked* que tenham ficado oclusos durante um certo número de *frames* consecutivas. Para maior robustês perante oclusões, o *Deep* SORT recorre a redes CNN treinadas para obter obter informação mais fiável sobre a aparência e o movimento dos objetos [65].

Segundo R. Kanjee [67], o *Deep* SORT corresponde ao *tracker* mais rápido, graças à sua simplicidade, conseguindo um bom nível de precisão a 16 FPS, sendo assim adequado para deteção e *tracking* de múltiplos objetos em tempo-real.

2.6 Protocolos de Comunicação IoT

Existem diversos protocolos de comunicação *wireless* que, dependendo do objetivo da aplicação, têm as suas vantagens e desvantagens. Vários aspetos devem ser tidos em conta na seleção do protocolo mais adequado como, por exemplo: frequências de operação; consumo energético; custo de operação; alcance e taxa de transmissão dos dados.

Com o rápido crescimento do mercado de IoT, as redes *Low-Power-Wide-Area Networks* (LPWAN) tornaram-se populares. Trata-se de uma tecnologia de comunicação rádio de longo alcance e de baixa taxa de transmissão [71]. Atualmente, o Sigfox, *Long Range* (LoRa) e *NarrowBand-Internet of Things* (NB-IoT) são os líderes desta tecnologia, competindo entre si no mercado de IoT. Por outro lado, as tecnologias rádio de curto alcance têm a desvantagem de não se destinarem a transmissões de longa distância, assim como as comunicações móveis que apresentam consumos energéticos significativos para o âmbito IoT. Na sua grande maioria, no contexto da IoT, pretende-se são pequenas quantidades de informação a longas distâncias e com baixos consumos energéticos. Na Figura 2.7 são apresentados, de forma sucinta, estes 3 tipos de tecnologias, tanto ao nível de taxa de transmissão como do alcance da respetiva comunicação.

Segue-se uma breve descrição dos protocolos de comunicação mais populares para o âmbito de IoT: Sigfox, LoRa e NB-IoT [71].

2.6.1 Sigfox

A tecnologia Sigfox surgiu em 2010, sendo também uma operadora de rede LPWAN que opera em soluções de IoT *end-to-end* em 31 países em colaboração com outras operadoras móveis [71]. As suas estações base são equipadas com rádios cognitivos que se conectam através de uma rede IP aos servidores. Os dispositivos adotam uma modulação de faixa muito estreita (100 Hz) nas faixas *Industrial, Scientific, and Medical* (ISM) não licenciadas. A comunicação é bidirecional e com baixos

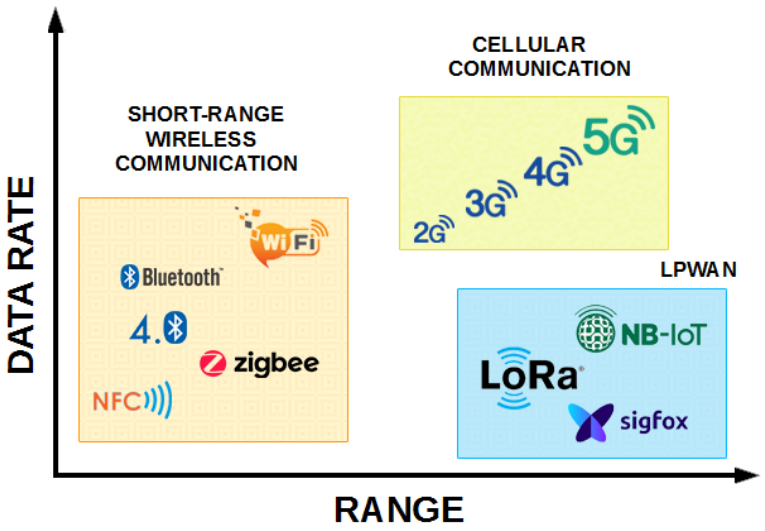


Figura 2.7: Representação dos 3 tipos de tecnologias (curto alcance; longo alcance e comunicações móveis) segundo os níveis de taxa de transmissão e alcance da comunicação [72].

níveis de ruído, podendo o dispositivo escolher de forma aleatória a frequência do canal para enviar as mensagens, daí a utilização dos rádios cognitivos. Por outro lado, a comunicação é limitada a 140 mensagens/dia e a um *payload* de 12 B em *Uplink* (UL), enquanto que em *Downlink* (DL) o limite é de 4 mensagens/dia com um *payload* de 8 B.

2.6.2 LoRA

O *Long Range* (LoRa) consiste num tecnologia de camada física desenvolvida em 2009, tendo sido padronizada pela LoRa-Alliance. Encontra-se disponível em 42 países, estando em fase de implementação noutros em regime de colaboração com outras operadoras móveis. Esta tecnologia utiliza também as faixas ISM não licenciadas para realizar comunicações bidirecionais. O LoRa utiliza 6 fatores de propagação para adaptar a transmissão de dados ao raio de alcance. Esta seleção leva em conta que o aumento de um destes parâmetros implicará a diminuição do outro. Dependendo do fator, a taxa de transmissão pode variar entre 0,3 kb/s a 50 kb/s e, em caso de uso de diferentes fatores, as mensagens transmitidas conseguem ser todas recebidas na estação LoRa, assegurando que o *payload* de cada uma não exceda os 243 B. Com recurso ao LoRaWAN, a utilização de múltiplas estações LoRa permite uma maior redundância na receção das mensagens, assim como fazer a localização dos dispositivos ligados na rede. A LoRaWAN consiste num protocolo de camada construído sob a modulação LoRa, sendo que define a forma como os dispositivos fazem uso do HW LoRa [73].

2.6.3 NB-IoT

O *NarrowBand-Internet of Things* (NB-IoT) baseia-se numa tecnologia de rádio de faixa estreita padronizada pela 3^{rd} *Generation Partnership Project* (3GPP) em 2016 [71]. O NB-IoT pode coexistir tanto com as comunicações *Global System for Mobile* (GSM), como com as LTE em faixas de frequência licenciadas e com uma largura de faixa de 200 kHz. Para que o NB-IoT seja suportado, é necessário uma melhoria de SW nas infraestruturas LTE. O protocolo de comunicação NB-IoT consiste numa simplificação do protocolo LTE para aplicações IoT e permite conectividade de até 100 mil dispositivos. A taxa de transmissão está limitada em DL a 200 kb/s, em UL a 20 kb/s e em 1600 B o *payload* das mensagens.

2.6.4 Comparação

Tendo como referência tanto a Tabela 2.4, que apresenta as características principais de cada uma destas tecnologias LPWAN, como a Figura 2.8, que as classifica segundo fatores a considerar em aplicações de IoT, seguem-se algumas conclusões sobre os casos onde cada um destes protocolos melhor se adequa.

Tabela 2.4: Visão geral das tecnologias LPWAN: Sigfox, LoRa e NB-IoT [71].

Propriedades	Sigfox	LoRa	NB-IoT
Frequência	Faixas ISM não licenciadas (868 MHz na Europa)	Faixas ISM não licenciadas (868 MHz na Europa)	Faixas de frequência LTE licenciadas
Largura de faixa	100 Hz	125 e 250 kHz	200 kHz
Taxa de transmissão	<100 b/s	<50 kb/s	<200 kb/s
Tipo de transmissão	Bidirecional/ Unidirecional	Bidirecional/ Unidirecional	Bidirecional/ Unidirecional
Mensagens/dia	140 (UL), 4 (DL)	*duty cycle* < 1%*	Ilimitadas
Payload	12 (UL), 8 (DL) B	243 B	1600 B
Alcance	10-40 km	5-20 km	1-10 km
Encriptação	Não	Sim	Sim
Módulos	<2 €	3-5 €	>20 €

* Segundo os regulamentos do *European Telecommunications Standards Institute* (ETSI) [74].

A nível de *Quality of Service* (QoS), o NB-IoT destaca-se por funcionar em faixas de frequência licenciadas, sendo a melhor solução para aplicações com fiabilidade na comunicação, apesar do seu elevado custo [71]. Em contrapartida, o NB-IoT acaba por ter maiores consumos devido ao *Quality of Service* (QoS), tornando o tempo de vida dos módulos mais reduzido. Para aplicações indiferentes à latência, o Sigfox será a melhor opção, caso contrário o NB-IoT é o que apresenta menor latência. Quanto à escalabilidade, o NB-IoT é o mais indicado, visto que permite uma maior

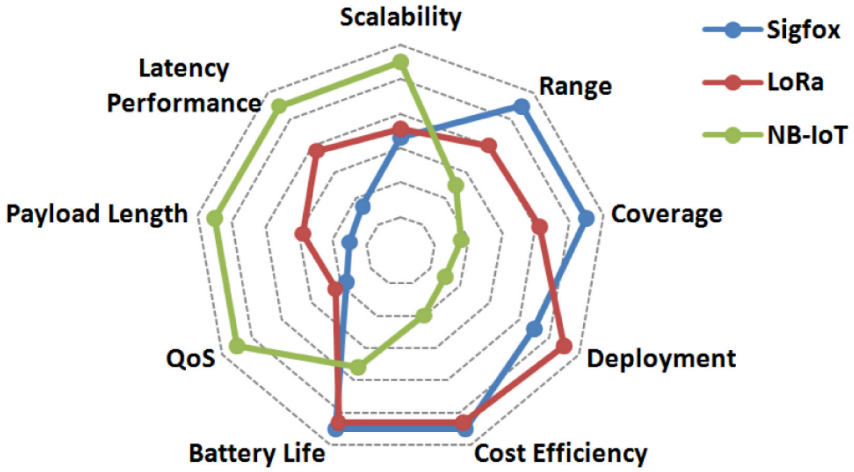


Figura 2.8: Comparação entre os protocolos de comunicação Sigfox, LoRa e NB-IoT, segundo fatores de IoT como: alcance; relação custo-eficiência; QoS e latência [71].

quantidade de dispositivos ligados (100 mil por antena LTE). Adicionalmente, o NB-IoT é também a tecnologia que permite maiores *payloads*, útil para aplicações que necessitem de enviar um elevado número de bytes. Quanto à cobertura e alcance, o NB-IoT está limitado à cobertura LTE, podendo ser comprometedora, em espaços rurais, visto ser a tecnologia com menor alcance. Por outro lado, o Sigfox é a opção que exige menor número de estações base, graças ao seu alcance de 40 km. O ecossistema LoRa tem a vantagem de ser flexível, visto permitir o desenvolvimento tanto numa rede LoRa privada como na pública. Por fim, quanto a custos, o NB-IoT é significativamente mais dispendioso devido a questões de licenciamento e de custos unitários de cada módulo, ao contrário do Sigfox que é a solução mais barata.

2.7 Protocolos de Mensagens IoT

Numa rede de IoT todos os dispositivos estão em constante comunicação entre si. Para além do protocolo de comunicação, um outro grande fator, que determina a performance da comunicação *Machine-to-Machine* (M2M), é o protocolo de mensagens utilizado [75]. Existem dezenas destes protocolos, que devem ser escolhidos conforme os requisitos da aplicação. De entre os diversos tipos de protocolos disponíveis [75, 76, 77, 78, 79], destacam-se: MQTT, AMQP, CoAP e DDS.

2.7.1 MQTT

O *Message Queue Telemetry Transport* (MQTT) é um *standard* da Organization for the Advancement of Structured Information Standards (OASIS) desde 2013, tendo sido introduzido pela International Business Machines Corporation (IBM) em 1999 [76]. Trata-se de um protocolo de transporte de mensagens leve e simples, com uma

arquitetura do tipo *publish-subscribe*, sobre o protocolo TCP/IP para comunicação M2M numa rede restrita comum [77]. O MQTT fornece um QoS de 3 níveis que asseguram a entrega da mensagem. Uma infraestrutura MQTT é composta por 3 tipos de elementos: *publishers*, *subscribers* e um *broker*. Os *publishers* são dispositivos que enviam mensagens para o *broker*, sendo estas categorizadas por tópicos segundo o tipo de dados. Já os *subscribers*, são dispositivos que subscrevem certos tópicos do *broker*, recebendo as respetivas mensagens assim que são enviadas pelos *publishers*. O *broker* MQTT pode exigir uma autenticação aquando da conexão de clientes e, para assegurar a privacidade, a conexão TCP deve ser encriptada com o protocolo de segurança SSL/TLS [77].

2.7.2 AMQP

O *Advanced Message Queuing Protocol* (AMQP) é um *standard* da OASIS destinado à indústria financeira, funcionando sobre o protocolo TCP/IP [77]. Este suporta a arquitetura *publish-subscribe* e possui 3 níveis de QoS [77]. O *broker* é dividido num *exchange* seguido de *queues*, sendo que o primeiro recebe as mensagens publicadas pelos *publishers* e as distribui pelas *queues* de cada tópico. Os *subscribers*, estes ligam-se às *queues* dos tópicos pretendidos e adquirem as mensagens que são publicadas.

2.7.3 CoAP

O *Constrained Application Protocol* (CoAP) é um protocolo criado dentro de um grupo de trabalho do *Internet Engineering Task Force* (IETF), que vem adaptar a lógica *Representational State Transfer* (REST) para aplicações de IoT. Isto significa que funciona segundo o paradigma *request-response*. É construído sobre o protocolo UDP e inclui um mecanismo que fornece fiabilidade. A sua arquitetura é dividida em 2 camadas, sendo a *messaging* responsável pela fiabilidade e duplicação das mensagens e o *request/response* pela comunicação [77]. O publicador envia mensagens para o *Uniform Resource Identifier* (URI) e o subscritor subscreve o mesmo URI, por onde receberá as mensagens publicadas ao longo do tempo [75]. Possui 2 níveis de QoS e, para fornecer segurança, conta com o *Datagram Transport Layer Security* (DTLS).

2.7.4 DDS

O *Data Distribution Service* (DDS) consiste num *standard* de mensagens desenvolvido pelo *Object Management Group* (OMG) para comunicações M2M [76, 77]. Este recorre à arquitetura *publish-subscribe*, utilizando o *multicasting* em vez de um *broker*. Trata-se de um protocolo com excelente fiabilidade e com 23 níveis de QoS. Esta arquitetura define 2 camadas: *Data-Centric Publish-Subscribe* (DCPS) e *Data-Local*

Reconstruction Layer (DLRL). A primeira tem a função de enviar mensagens para os subscritores, enquanto que a segunda consiste numa interface às funcionalidades do DCPS. O publicador faz a distribuição dos dados, no qual o *data writer* dá a aprovação dos dados, assim como as alterações a enviar para os subscritores. Os subscritores são os recetores destes dados publicados. Os *data readers* os lêem e enviam os dados para os subscritores, segundo tópicos que caracterizam o tipo de dados. Assim sendo, os *data writers* e os *data readers* desempenham o papel do *broker*.

2.7.5 Comparação

A Tabela 2.5 apresenta uma análise comparativa entre os protocolos abordados.

Tabela 2.5: Análise comparativa de protocolos de mensagens para IoT: MQTT, AMQP, CoAP e DDS [76, 77].

Propriedades	MQTT	AMQP	CoAP	DDS
Protocolo de Transporte	TCP	TCP	UDP	TCP/UDP
Modelo *Publish/Subscribe*	Sim	Sim	Não	Sim
Modelo *Request/Response*	Não	Não	Sim	Não
Segurança	TLS	TLS	DTLS	TLS ou outros
Níveis de QoS	3	3	2	23

A escolha de um destes protocolos está dependente da aplicação visto que cada um se destina a aplicações bastante específicas [77]. O MQTT é, de momento, o mais utilizado devido a características como o baixo *overhead* e o baixo consumo energético, indicado para aplicações sobrecarregadas computacionalmente e de baixo consumo energético. Para um caso de aplicação que exija funcionalidade REST, o *Constrained Application Protocol* (CoAP) será a opção mais indicada. J. Dizdarević *et al.* [78] referem que o MQTT é o preferido dos *developers*, não só pela sua maturidade, estabilidade e simplicidade na configuração, como também pela documentação *online* e implementações bem sucedidas. Assim, o MQTT tem vindo a demonstrar uma performance excelente em dispositivos no âmbito de IoT. Adicionalmente, é destacado o rápido crescimento do CoAP que, num futuro próximo, poderá chegar a uma maturidade semelhante à do MQTT.

Segundo a avaliação de N. Naik sobre estes protocolos [75], o CoAP apresenta menor tamanho da mensagem e *overhead* por ser o único que não funciona sobre TCP. O *Advanced Message Queuing Protocol* (AMQP) apresenta o maior *overhead* e tamanho da mensagem devido ao seu suporte de segurança e fiabilidade. Quanto ao consumo energético e recursos, o CoAP apresenta os valores mais baixos, sendo que tanto o MQTT como o CoAP foram projetados para funcionar a pequenas larguras de faixa e em dispositivos com recursos limitados, enquanto que o AMQP apresenta piores resultados a este nível. O CoAP envolve menor latência e largura de faixa,

visto que os restantes protocolos funcionam sobre TCP, utilizando mais largura de faixa no início da conexão, levando ao congestionamento na rede e ao aumento da latência. Quanto ao QoS, o *Data Distribution Service* (DDS) é o que apresenta mais níveis, seguido do MQTT [77], permitindo a definição de limites e prioridades de diferentes categorias de dados [80]. Por fim, o AMQP apresenta um maior nível de segurança, enquanto que o MQTT é o que compromete mais a segurança na autenticação, caso não funcione com SSL/TLS.

2.8 RGPD

O Regulamento Geral sobre a Proteção de Dados (RGPD) (Regulamento (UE) 2016/679 do Parlamento Europeu e do Conselho) estabelece as regras relativas ao tratamento de dados pessoais de pessoas na União Europeia (UE), assim como a livre circulação destes por parte de uma pessoa, empresa ou organização [81]. A sua publicação foi feita em 2016 e a sua aplicação começou em 2018, em todos os países membros da UE [82], tendo sido assegurada a sua execução na ordem jurídica portuguesa através da Lei n.°58/2019, de 8 de agosto.

Esta lei pretende defender princípios e direitos fundamentais da liberdade, principalmente o respeito pela vida privada e familiar, proteção de dados pessoais e a diversidade cultural, sendo aplicável, não só às organizações que se encontrem na UE, como também às organizações externas que tenham efeito na UE e façam o tratamento de dados pessoais de residentes na UE [82].

No que toca à recolha de imagens através de dispositivos de vídeo, segundo o RGPD, os responsáveis pelo tratamento devem assegurar que os dados extraídos de uma imagem digital para a construção de um modelo devem conter apenas as informações necessárias para a finalidade especificada, de forma a que não tenham repercussões no comportamento dos indivíduos, através de tratamentos adicionais. Uma das formas de garantir o cumprimento destas regras é através da execução de uma Avaliação de Impacto sobre a Proteção de Dados (AIPD) (artigo 35° do RGPD) e da minimização dos dados pessoais, cumprindo o disposto no artigo 5° do RGPD, assim como estabelecer onde ficam armazenados os dados recolhidos e o respetivo prazo de conservação. A anonimização dos dados pessoais deverá ser uma das soluções para garantir, por exemplo através da desfocagem da imagem. Dessa forma, para garantir que um serviço ou produto possa entrar no mercado sem colocar em causa as regras relativas à proteção de dados pessoais, é necessário adoptar uma abordagem baseada na proteção de dados desde a conceção, de acordo com o artigo 25° do RGPD, que possa garantir a aplicação das medidas técnicas e organizativas adequadas.

Situação paradigmática da importância do tratamento de dados pessoais aconteceu durante a pandemia da Doença do Coronavírus (COVID-19) em que vários

países recorreram a aplicações móveis de rastreamento para tentar conter a propagação do vírus [83, 84]. Com base legal do artigo 9°, n°2 alínea i) do RGPD, admitiu-se o tratamento de dados de saúde por motivos de interesse público, e foi com este fundamento que se efetuou o tratamento dos dados pessoais através da geolocalização dos seus titulares. Ainda assim, nos vários países foram levantadas várias questões acerca do tipo de tratamento que era feito, assim como se esses dados eram reutilizados pelas autoridades de saúde pública. Estas questões potenciadas pela falta da realização de uma AIPD, como no caso Português, conforme a deliberação da Comissão Nacional de Proteção de Dados (CNPD) 2020/262, que demonstra a falta de sensibilidade sobre a proteção de dados pessoais, originando desconfiança dos titulares na sua utilização.

Neste capítulo foi feito o levantamento de produtos no mercado e o estudo de técnicas, tecnologias e protocolos mais utilizados em aplicações no âmbito das *smart cities*. No próximo capítulo é abordada uma componente teórica relativa às técnicas, tecnologias e protocolos a utilizar para combater, de forma mais eficiente, problemas emergentes nas *smart cities*.

Capítulo 3

Fundamentos Teóricos

Este capítulo introduz e descreve os principais fundamentos teóricos aplicados ao longo das fases de projeto (Capítulo 4) e implementação (Capítulo 5) do sistema de análise de tráfego desenvolvido. Assim sendo, os conceitos abordados são: *Convolutional Neural Network*; *Tracker* NvDCF; *Queue*; Multiprocessamento; GPU, MQTT e, por fim, SSL/TLS.

3.1 *Convolutional Neural Network*

Uma *Convolutional Neural Networks* (CNN) consiste num tipo específico de rede neuronal artificial para o reconhecimento e processamento de imagens bidimensionais. Esta rede identifica padrões com base na informação dos pixeis da imagem, como por exemplo, a cor e o brilho, que permitirão fazer a identificação de objetos na imagem [85].

Tipicamente, uma CNN é constituída por uma série de 3 tipos de camadas (camada de convolução; camada de *pooling* e camada *Fully Connected*), assim como de uma função de ativação não-linear [86]. A Figura 3.1 representa um exemplo da arquitetura CNN para classificar a imagem de um cão, passando pelas 4 fases da rede CNN e terminando numa classificação probabilística para cada tipo de classe [87].

Tendo como referência a Figura 3.1, segue-se a descrição de cada uma das 4 fases constituintes da arquitetura CNN [86, 87]:

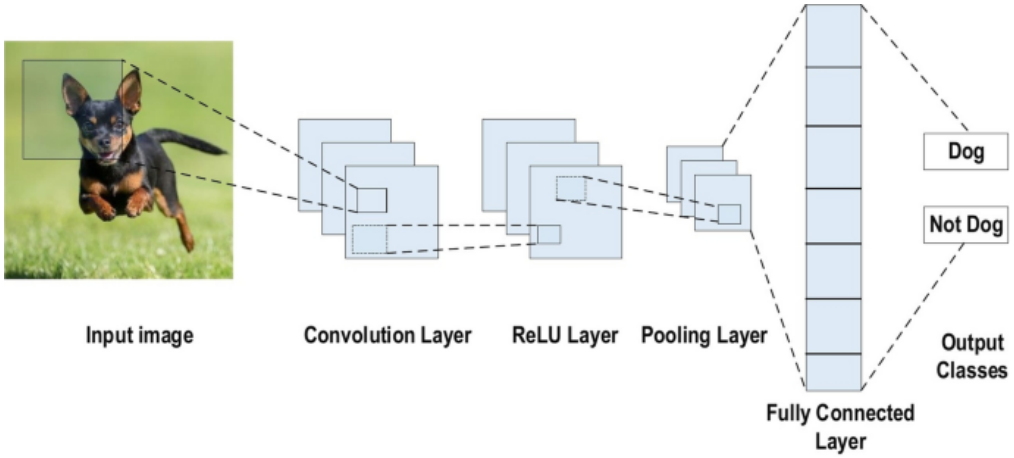


Figura 3.1: Arquitetura de uma rede CNN para classificação de imagens [87].

- **Camada de Convolução** - corresponde à camada principal da arquitetura CNN, consistindo num conjunto de filtros de convolução (*kernels*) formados por uma série de *weights* e *bias*. Nesta camada, é feito o produto escalar entre a imagem de entrada, de dimensão N, com estes filtros, resultando num *feature map*;

- **Função de Ativação Não-Linear** - esta função faz a decisão não-linear, sendo os neurónios da rede amplificados ou atenuados, tendo em conta o seu valor. Dos vários tipos de funções de ativação não-lineares, o mais utilizado no contexto CNN é o *Rectified Linear Unit* (ReLU), convertendo para zero todos os números negativos do *feature map*. De todos, o ReLU é o que envolve menor carga computacional, o que possibilita à rede CNN aprender processos mais complexos. Em contrapartida, a sua utilização na rede gera problemas aquando da aplicação do algoritmo da retropropagação de erros;

- **Camada de *Pooling*** - é uma camada que faz uma redução das dimensões do *feature map*, embora garanta a maioria das *features* mais relevantes. Dos vários métodos de *pooling*, os mais utilizados são o *average* e o *max pooling*. Por um lado, o *pooling* ajuda na aceleração do treino da rede, assim como a correção de problemas de *overfitting*, mas, por outro lado, este poderá reduzir a performance da rede CNN;

- **Camada *Fully Connected* (FC)** - consiste, normalmente, na camada da arquitetura CNN, no qual cada neurónio está ligado a todos os neurónios da camada anterior, permitindo assim a classificação da imagem. Desta forma, a sua entrada consiste num vetor resultante do processo de *flattening* do *feature map* da camada anterior. Da camada *Fully Connected* (FC), resultam

N neurónios, sendo N a quantidade de classes, que ajudam no processo de classificação da imagem.

Ao longo dos últimos anos, tem-se vindo a verificar um aumento significativo do número de camadas nas redes CNN, passando a denominar-se *Deep Convolutional Neural Networks* (DCNN) [88]. Embora este aumento proporcione uma extração de *features* mais complexas que melhoram a classificação e reconhecimento de imagens, tornou não só mais difícil o seu treino, como também afeta a sua precisão [89]. É o caso do Problema da Dissipação do Gradiente que, ao longo das camadas da DCNN, perde informação do gradiente.

Uma *Residual Network* (ResNet), que consiste num tipo específico de rede neuronal, veio resolver certos problemas nas redes CNN [89]. Esta segue também a arquitetura da rede CNN, em que o seu fator diferenciador está na adição de uma *identity connection* entre camadas da rede. A Figura 3.2 apresenta o bloco residual usado nesta rede, onde a seta curva representa a ligação *identity connection*, fazendo a ponte de ligação da entrada do bloco residual para a saída. Nesta estrutura, x representa o valor de entrada que passa por um conjunto de convoluções (*weight layers*) e funções de ativação (ReLU), resultando no mapa residual $F(x)$ [87, 88]. Desta forma, o resultado do bloco residual $H(x)$ corresponde ao somatório de $F(x)$ com a identidade da sua entrada x.

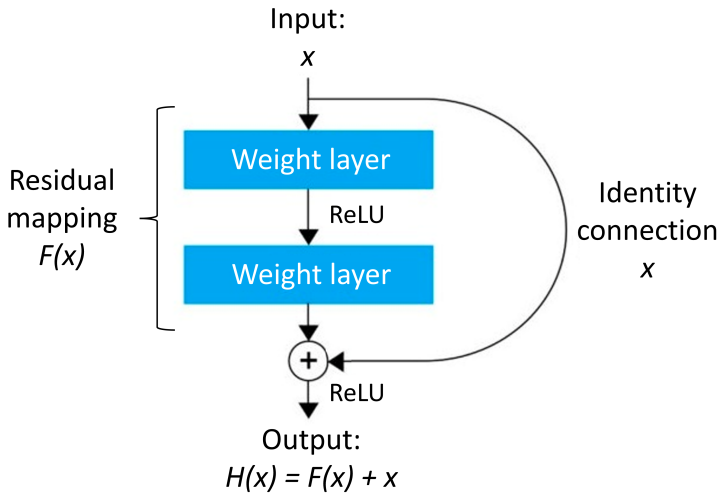


Figura 3.2: Estrutura do bloco residual da ResNet (adaptado de [89]).

No processo de treino de uma rede ResNet, o objetivo é treinar a função residual $F(x)$, tendo por base a diferença entre o valor de entrada e o de saída, isto é, $F(x) = H(x) - x$, melhorando a precisão da rede [88]. Com esta transição da identidade x, a complexidade computacional é mantida e o Problema da Dissipação do Gradiente é resolvido, permitindo às camadas mais profundas manterem o gradiente, isto é, a performance de cada camada não é perdida ao longo da rede. Desta forma, a

utilização da ResNet melhora significativamente a performance de redes neuronais com mais camadas, comparativamente com as redes neuronais rasas com o mesmo número de camadas, que não possuem transições de identidade.

Existem 2 tipos de blocos utilizados em ResNet, dependendo se as dimensões da entrada e as de saída são iguais [90], sendo estes:

- **Bloco de identidade** que consiste no bloco *stantard* usado na rede ResNet, como descrito acima e representado pela Figura 3.2. Neste caso, este é aplicável nas situações em que as dimensões da entrada x são iguais às de $F(x)$;
- **Bloco de convolução** é utilizado quando as dimensões da entrada x são diferentes das de $F(x)$, sendo para isto necessário existir uma camada de convolução no processo de transição da identidade x.

3.2 *Tracker* NvDCF

O *tracker NVIDIA-adapted Discriminative Correlation Filter* (NvDCF) consiste numa biblioteca de baixo nível, desenvolvida pela NVIDIA, que suporta *multis-treaming*, assim como o *tracking* de múltiplos objetos, tendo por base a informação retirada de imagens no formato NV12 ou *Red-Green-Blue-Alpha* (RGBA) [91]. Este *tracker* baseia-se tanto num *Discriminative Correlation Filter* (DCF), para *tracking* visual de objetos, como num algoritmo de associação de dados, tendo por base informação visual e contextual. O *NVIDIA-adapted Discriminative Correlation Filter* (NvDCF) consiste numa solução que permite, com recurso ao GPU, a aceleração de operações de *tracking*.

Usufruindo das potencialidades do GPU, é feita a alteração da escala das imagens e conversão do formato de cores, para além do processo de *tracking*. Como resultado, o *tracker* retorna as coordenadas, o nível de confiança e o ID dos objetos *tracked*.

Inicialmente, o NvDCF aloca memória com base no número máximo definido de objetos *tracked* e o número de *streams*. Adicionalmente, este aplica a Média Móvel Exponencial (MME), atribuindo maior peso aos dados das imagens mais recentes, permitindo uma reação mais rápida à alteração da posição dos objetos [92]. Para otimização do filtro *Discriminative Correlation Filter* (DCF), define-se o desvio padrão da distribuição Gaussiana.

O NvDCF define, em torno da posição do objeto, uma região com dimensões suficientes para assegurar que será detetado nesta região na seguinte imagem. Posteriormente, esta região é recortada e dimensionada para a escala definida de extração de *features*.

Por forma a tornar o *tracker* mais robusto, são aplicadas técnicas para evitar falsos positivos representadas pela Figura 3.3. Quando um novo objeto é detetado, é temporariamente criada uma instância *tracker* no modo *Tentative*. Caso seja

detetado nas imagens seguintes, passa para o modo *Active*; caso o objeto não seja detetado nas imagens seguintes, passa para o modo *Terminated*. No modo *Active*, o objeto transita para o modo *Inactive*, quando deixa de corresponder a qualquer *bounding box* de um novo objeto ou quando a confiança do *tracker* é baixa. Já nos modos *Inactive* e *Active*, caso um objeto deixe de ser *tracked*, por um longo período de tempo, ou até que a sua *bounding box* fique parcialmente oculta, o objeto passa para o modo *Terminated*.

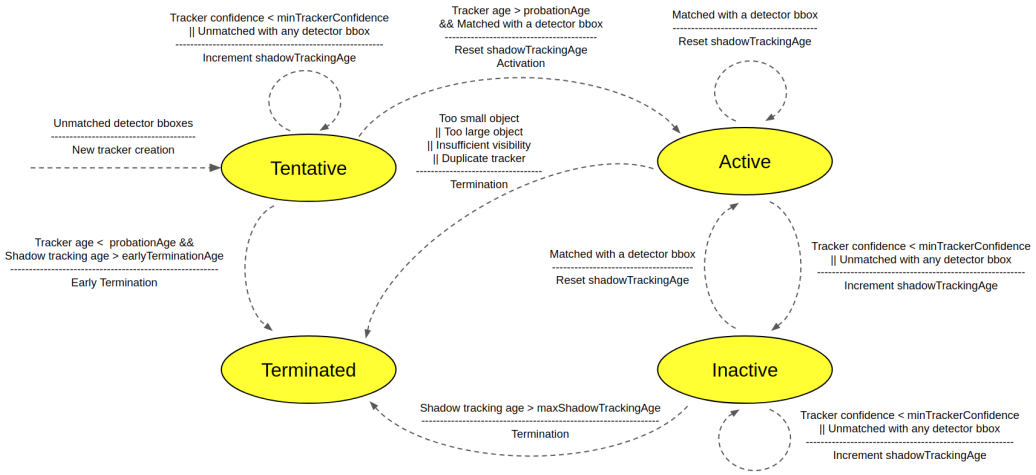


Figura 3.3: Máquina de estados e respetivas transições de estado do *tracker* NvDCF [91].

O NvDCF apresenta 2 tipos de estimadores de estado, sendo que um se baseia no filtro de *Kalman*, e o outro no *MovingAvgEstimator* (MAE). O estimador MAE apresenta 7 estados, $\{x, y, a, h, dx, dy, dh\}$, sendo x e y as coordenadas do canto superior esquerdo da *bounding box* do objeto; a e h o *aspect ratio* e a altura da *bounding box*, respetivamente, e os estados dx, dy e dh correspondentes aos valores de velocidade.

3.3 *Queue*

Uma *queue*, em Ciências da Computação, consiste numa fila constituída por uma série de entidades organizadas de forma sequencial, começando pela parte frontal desta (*front*) até à posterior (*rear*) [93]. A *queue* pode ser manipulada fazendo a adição de outras entidades, *enqueue*, ou a remoção das existentes, *dequeue*. A forma como a adição e a remoção de entidades é realizada na *queue* segue uma ordem particular, denominada por *First In, First Out* (FIFO), sendo que as entidades adicionadas à *queue* à mais tempo são as primeiras a ser retiradas [94]. Na Figura 3.4 é apresentada uma *queue* de tamanho N, composta por quatro entidades, sendo

demonstrado o processo de adição de uma entidade X para a quinta posição da *queue*, e, simultaneamente, a retirada da primeira entidade desta.

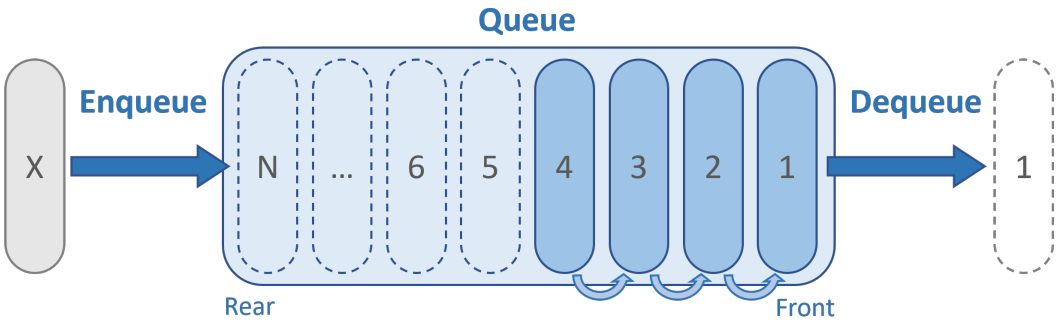


Figura 3.4: Representação da estrutura de uma *queue* e o seu funcionamento.

Embora a estrutura e funcionamento sejam bastante semelhantes, este conceito é por vezes confundido com o termo *stack*, sendo que nesta são retiradas as entidades que mais recentemente foram adicionadas à fila, consistindo na metodologia denominada por *Last In, First Out* (LIFO) [94].

Dependendo do caso de aplicação de *queues*, existem vários tipos com estruturas e funcionamentos ligeiramente distintos, como por exemplo [95]: *queue* circular; *queue* prioritária e *queue double-ended*.

A utilização de *queues* destina-se a casos onde os dados disponibilizados por uma fonte de informação não têm necessariamente de ser processados no momento, ou não é possível que sejam processados no instante pelo recetor [93]. Assim, com a utilização da *queue*, garante-se maior eficiência de funcionamento por parte do recetor, garantindo-se que não é perdida informação, visto ser armazenada temporariamente na *queue*.

Desta forma, a sua aplicação é útil em cenários como [96]: testes de simulação de comunicações; agendamento de tarefas; gestão de recursos partilhados; *buffer* de teclados ou gestão de congestionamento da rede.

3.4 Multiprocessamento

O conceito de multiprocessamento (*multiprocessing*) consiste numa técnica em que o sistema suporta mais do que um processo em simultâneo. As tarefas existentes são repartidas em pequenas rotinas que funcionam de forma independente, sendo o Sistema Operativo (SO) quem faz a sua alocação às várias unidades de processamento, melhorando a performance global do sistema. Assim sendo, é de salientar que um sistema de multiprocessamento pode ser:

- **Multiprocessador**, isto é, um computador com mais do que um CPU;

- **Processador *multicore***, isto é, um computador com apenas um CPU, sendo este constituído por duas ou mais unidades de processamento independentes designadas *cores*.

Por vezes, o termo multiprocessamento é confundido com o conceito *multithreading*. Neste segundo caso, um sistema executa múltiplas *threads* de um processo em simultâneo. Cada *thread* é parcialmente executada e, de seguida, executada parte de uma outra *thread*, criando-se assim um pequeno *delay* ao longo da execução de cada tarefa. Na Figura 3.5 é demonstrada uma experiência que compara o comportamento de 4 tarefas utilizando as diferentes técnicas: *multithreading* e *multiprocessamento*. Com isto, verifica-se que com o *multithreading* existe uma execução alternada entre as várias tarefas, enquanto que no multiprocessamento estas não são só continuamente executadas, mas mais rapidamente. [97].

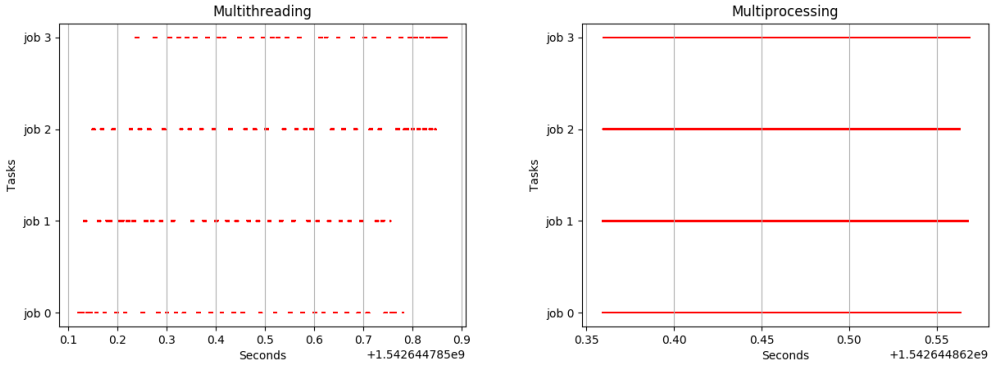


Figura 3.5: Comparação da execução de quatro tarefas com as técnicas *multithreading* e de multiprocessamento [97].

A utilização de multiprocessamento destina-se a casos de utilização intensiva do CPU em que não existem interações com um utilizador ou outros periféricos [98]. Aspetos como o paralelismo, a ausência de latência e o aumento da velocidade de execução de cada tarefa são os pontos fortes desta técnica, sendo também um sistema mais fiável [99]. Por outro lado, esta técnica implica um custo associado aos vários CPU ou a um CPU com múltiplos *cores*. A gestão dos processos implica um SO mais complexo assim como maior memória.

3.5 GPU

Um *Graphics Processing Unit* (GPU) é um processador composto por centenas de pequenos *cores* especializados em tarefas específicas [100], em que o processamento é realizado em paralelo. O funcionamento conjunto destes *cores* apresentam uma elevada performance quando o processamento de uma tarefa específica é paralelizado,

visto que irão ser realizadas biliões de operações aritméticas, de forma repetitiva, acelerando a computação gráfica [100, 101].

Já o *Central Process Unit* (CPU) consiste num processador com bastantes menos *cores*, para além de uma arquitetura distinta com diferentes finalidades quando comparadas com as do GPU. O CPU destina-se a uma ampla variedade de aplicações onde a latência e o desempenho dos núcleos são relevantes. Os seus poucos *cores* têm como foco executar o mais rápido possível tarefas singulares genéricas [100].

Hoje em dia, a utilização do GPU tornou-se mais frequente porque se tornou mais flexível e programável, aumentando as suas capacidades e casos de aplicação [102]. Atualmente, o recurso à execução em GPU foi adotado em diversos campos [103]:

- **Processamento 3D** onde realiza cálculos 2D e 3D, assim como uma renderização de gráficos 3D com um desempenho bastante melhor que um CPU;
- **Análise de dados** com um processamento aritmético de elevada precisão e uma complexa programação matemática;
- **Escalonamento CPU/GPU** de carga intensa de processamento por parte do CPU para o GPU para libertar os recursos do CPU e conseguir um desempenho consistente;
- **Eficiência energética** dado que o GPU necessita de menos energia para realizar uma mesma tarefa que o CPU. Assim, economizam-se os custos energéticos e, indiretamente, consiste numa prática mais amiga do ambiente;
- **Inteligência Artificial** porque o treino de modelos que exige é mais eficiente quando é processado pelo GPU, não só para a procura de padrões, como para otimização do tempo de inferência;
- **Codificação de vídeo e *streaming*** dado que o GPU deteta mais eficientemente alterações do formato do vídeo de eventos de *streaming*. Tratarem-se de tarefas que exige uma elevada carga de recursos que só o GPU tem capacidade de processar no tempo útil.

Apesar de todas as capacidades do GPU perante o CPU, a combinação das potencialidades de ambos oferece ótimas condições para aplicações de DL e IA [100].

3.6 MQTT

Como referido na Seção 2.7.1, o MQTT é um protocolo do tipo *publish/subscribe* indicado para estabelecer uma conexão remota entre dispositivos com uma largura de faixa de rede mínima e utilização de poucas linhas de código para estabelecimento da comunicação [104].

Esta arquitetura é constituída por [105]:

- ***Broker* MQTT**, que é o servidor que gere o protocolo, responsável por manter a informação e subscrição dos seus clientes; receber todas as mensagens dos publicadores e remetê-las aos respetivos subscritores; verificar a autenticidade dos clientes que se conectam;
- **Clientes MQTT**, que são os dispositivos que se conectam ao *broker*. Podem ser publicadores, subscritores ou ambos. Um cliente MQTT é qualquer dispositivo que executa uma biblioteca MQTT e se conecta a um *broker* MQTT através rede recorrendo à *stack* TCP/IP;
- **Tópicos**, que correspondem aos nomes dos diretórios que permitem ao *broker* filtrar as mensagens para cada cliente. São *strings* no formato UCS *Transformation Format 8* (UTF-8), que contêm um caminho hierarquicamente estruturado e delimitado por barras ('/'). Um exemplo de tópico com quatro níveis é o seguinte: "casa/cozinha/forno/temperatura".

A Figura 3.6 apresenta a estrutura desta arquitetura. É apresentado o exemplo de um cliente publicador, sensor de temperatura, que publica mensagens com o valor de temperatura para o tópico *temperature* e os dois clientes subscritores deste tópico (um telemóvel e um servidor de *backend*) que recebem a mensagem de temperatura enviada pelo *broker*.

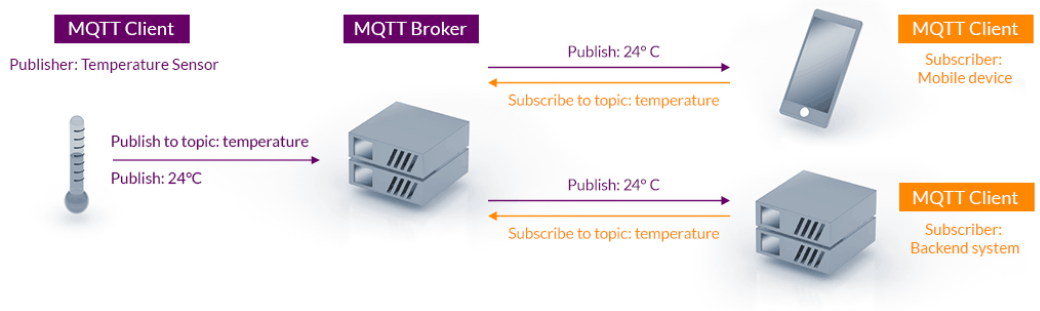


Figura 3.6: Arquitetura *publish/subscribe* do protocolo MQTT [104].

O padrão *publish/subscribe* desacopla o publicador do subscritor, ou seja, os clientes não necessitam de conhecer o IP e porta dos outros clientes ligados ao *broker*, mas sim apenas do *broker* [105]. Adicionalmente, os vários clientes não têm a necessidade de estar em funcionamento nem de interromper o seu funcionamento no momento da publicação ou receção.

A lógica típica de funcionamento do protocolo MQTT consiste num cliente publicar uma mensagem num tópico específico e os interessados subscreverem o mesmo tópico para receber a mensagem publicada. Neste processo de troca de mensagens,

o *broker* utiliza a lista de tópicos e clientes subscritores para tratar do reencaminhamento das mensagens apenas para os clientes subscritos. Caso, num dado momento, algum dos interessados não esteja ligado ao *broker* devido a algum problema de conexão, o *broker* é capaz de guardar as mensagens e enviá-las assim que este se reconecte.

Sempre que um cliente pretende ligar-se ao *broker*, tem de enviar uma mensagem de conexão, composta por:

- **ID do cliente** para identificar cada cliente junto do *broker*;
- **Nome de utilizador** e **palavra-passe** para autenticação e autorização do cliente;
- **Informação de *Last Will and Testament* (LwT)** com a mensagem a publicar em caso de desconexão do cliente;
- ***Keep alive*** com o intervalo de tempo máximo que o cliente e o *broker* podem permanecer sem trocar mensagens, antes do *broker* dar o cliente como desconectado.

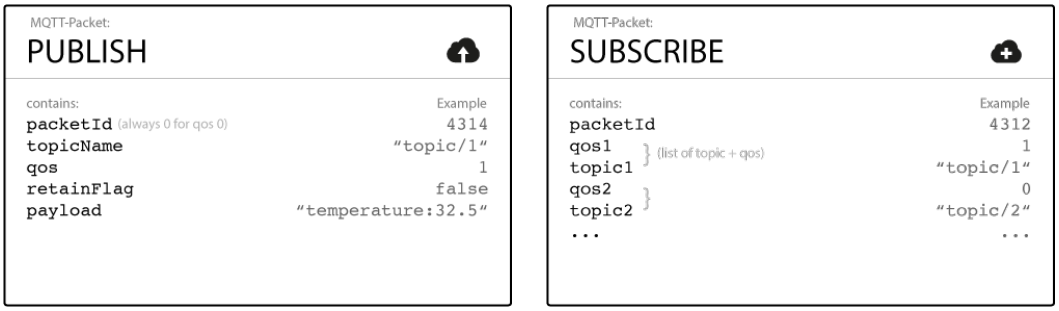
Um cliente MQTT apenas consegue publicar mensagens depois de estar conectado ao *broker*. Cada mensagem deve seguir uma estrutura (conforme representada na Figura 3.7a), tendo como parâmetros:

- **ID da mensagem** para identificar numericamente as mensagens durante a comunicação entre cliente e *broker*;
- Nome do **tópico** para que o *broker* encaminhe a mensagem apenas para os clientes subscritores do tópico;
- O **QoS**, que pode tomar os valores 0, 1 ou 2, para definir a forma com a mensagem publicada deve ser recebida pelos clientes (0 - no máximo uma vez; 1 - pelo menos uma vez; 2 - exatamente uma vez);
- ***Flag retained*** que define se a última mensagem publicada no tópico é guardada pelo *broker*. Sendo *retained*, cada novo cliente que subscreva este tópico recebe esta mensagem *retained* imediatamente após a subscrição;
- ***Payload*** dos dados a transmitir.

Para que o *broker* saiba que mensagens um dado cliente tem interesse em receber, o cliente tem de enviar uma mensagem de subscrição após a conexão. Esta mensagem segue a estrutura apresentada na Figura 3.7b e tem como parâmetros:

- **ID da mensagem** para identificar numericamente as mensagens;

- **QoS** relativo ao tópico a subscrever;
- Nome do **tópico** a subscrever.



(a) Mensagem de publicação (b) Mensagem de subscrição

Figura 3.7: Estrutura de mensagens MQTT para publicação e subscrição [105].

Como confirmação da subscrição, o *broker* envia ao cliente uma mensagem de *acknowledge*, reportando o sucesso ou insucesso da subscrição do tópico. Existe igualmente uma mensagem de dessubscrição que permite apagar subscrições do cliente, sendo a sua estrutura de mensagem semelhante à da subscrição.

Como resumo, a Figura 3.8 ilustra de forma sequencial as várias etapas da comunicação entre um cliente MQTT e o *broker* MQTT:

- Enviar a mensagem de conexão ao *broker* e aguardar por um *acknowledge* de conexão estabelecida;
- Subscrever um conjunto de tópicos e aguardar pelo *acknowledge* das subscrições;
- Publicar uma mensagem num tópico com um QoS (no caso da Figura 3.8 o QoS é igual a 2, sendo necessária a troca de 4 mensagens);
- Realizar um *ping* periódico ao *broker* para garantir a manutenção da conexão durante o período de *keep alive*;
- Dessubscrição dos tópicos e aguardar por um *acknowledge* de dessubscrições;
- Desconexão do cliente ao *broker*.

Para assegurar uma comunicação segura e encriptada entre clientes e o *broker*, o MQTT pode funcionar para o protocolo de segurança SSL/TLS, embora isto aumente o *overhead* da comunicação, ou seja, o pacotes TCP trocados são de maior dimensão [107].

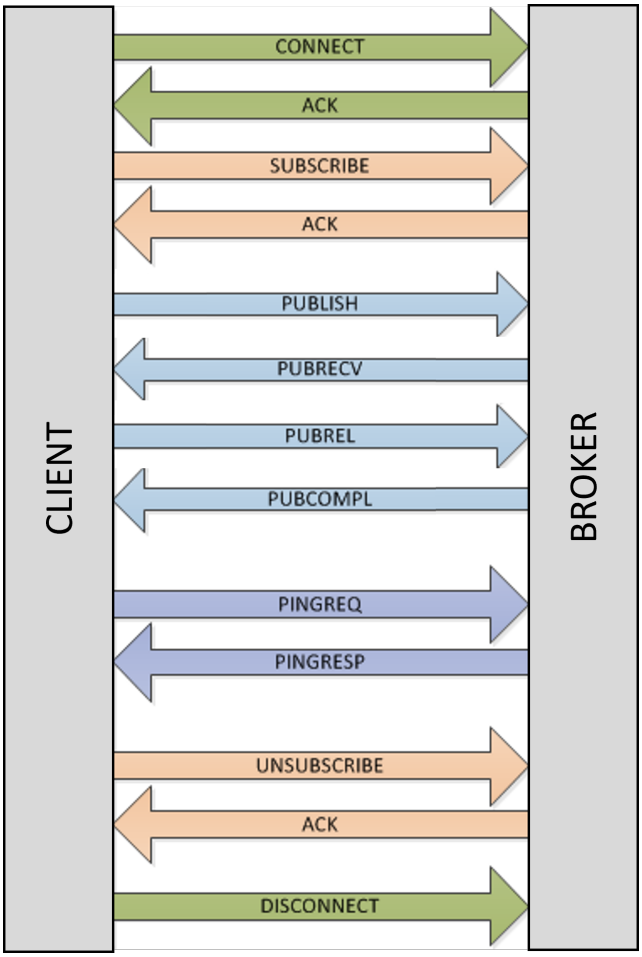


Figura 3.8: Diagrama ilustrativo das várias etapas numa comunicação MQTT (adaptado de [106]).

3.7 SSL/TLS

O *Secure Sockets Layer* (SSL) e o seu sucessor, o *Transport Layer Security* (TLS) são protocolos de segurança criptográfica para comunicação em rede. Fornecem a integridade dos dados e a privacidade da comunicação entre ambas as partes, isto é, asseguram que nenhum terceiro lê ou adultera os dados trocados entre o cliente e servidor [108].

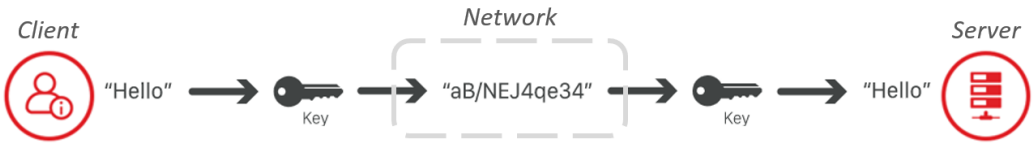
Existem três componentes principais que o protocolo *Transport Layer Security* (TLS) utiliza [109]:

- **Encriptação** - oculta de terceiros a informação transferida;
- **Certificação** - garante que as entidades com as quais se trocam mensagens são quem realmente afirmam ser;
- **Integridade** - verifica se a informação não foi falsificada ou adulterada.

3.7.1 Tipos de Encriptação

O estabelecimento de uma sessão SSL/TLS usa 2 tipos de encriptação:

- **Encriptação Simétrica** - processo em que a mesma chave, *session key*, é utilizada tanto para encriptar como para desencriptar informação. Na Figura 3.9a, o cliente usa a chave para encriptar a mensagem "Hello" e o servidor desencripta-a usando a mesma chave;
- **Encriptação Assimétrica**, também referida como criptografia com chave pública, utiliza pares de chaves: chave pública e chave privada. Uma entidade pode partilhar a sua chave pública com qualquer uma outra, enquanto que a sua chave privada apenas deve ser conhecida pela própria entidade. Qualquer informação encriptada por uma chave pública apenas pode ser desencriptada pelo seu par, isto é, pela chave privada correspondente. Na Figura 3.9b o cliente usa a chave pública do servidor para encriptar a mensagem "Hello" e, após o envio desta, apenas o servidor é capaz de a desencriptar viso que só ele possui a chave para, isto é, a sua chave privada.



(a) Encriptação Simétrica

(b) Encriptação Assimétrica

Figura 3.9: Diferenciação entre o processo de (a) encriptação e desencriptação simétrica com uso de *session keys* e o de (b) encriptação assimétrica com recurso às chaves pública e privada (adaptado de [110]).

3.7.2 TLS *Handshake*

O estabelecimento de uma conexão segura SSL/TLS consiste num processo que envolve várias etapas. A etapa mais importante é a do TLS *Handshake*, onde o servidor e o cliente trocam informação para determinar as propriedades da conexão [111, 112]. Segue-se a descrição das várias fases do TLS *Handshake*, tendo por base a Figura 3.10:

- **Etapa 1:** O cliente começa por enviar para o servidor uma mensagem *Client Hello* que contém: versão do TLS suportada; número aleatório para gerar a futura chave de encriptação simétrica; lista de algoritmos de criptografia suportados;
- **Etapa 2:** O servidor responde com uma mensagem *Server Hello* com: o seu certificado SSL/TLS assinado, que prova a sua identidade ao cliente e contém a sua chave pública; algoritmo de criptografia escolhido; número aleatório para gerar a futura chave de encriptação simétrica;
- **Etapa 3:** O cliente fornece o seu certificado SSL/TLS assinado caso o servidor exija a certificação;
- **Etapa 4:** O cliente cria a chave *pre-master* (*string* de bytes aleatórios) e envia-a para o servidor, encriptando-a previamente com a chave pública que recebeu na Etapa 2 (aqui é usada encriptação assimétrica);
- **Etapa 5:** O servidor desencripta a chave *pre-master* utilizando a sua própria chave privada (aqui é usada encriptação assimétrica);
- **Etapa 6:** Esta chave *pre-master*, juntamente com os 2 números aleatórios trocados inicialmente (Etapas 1 e 2), são usados pelo cliente e pelo servidor para calcular, de forma independente, a *session key*. Esta será utilizada para encriptar simetricamente a comunicação entre ambos daqui por diante;
- **Etapa 7:** A última mensagem do TLS *Handshake* por parte do cliente é enviada, sendo esta a primeira mensagem encriptada com a *session key*;
- **Etapa 8:** A última mensagem do TLS *Handshake* por parte do servidor é enviada, também ela encriptada com a *session key*. O processo de *Handshake* termina, estando estabelecido um canal de comunicação encriptada simetricamente entre o cliente e o servidor.

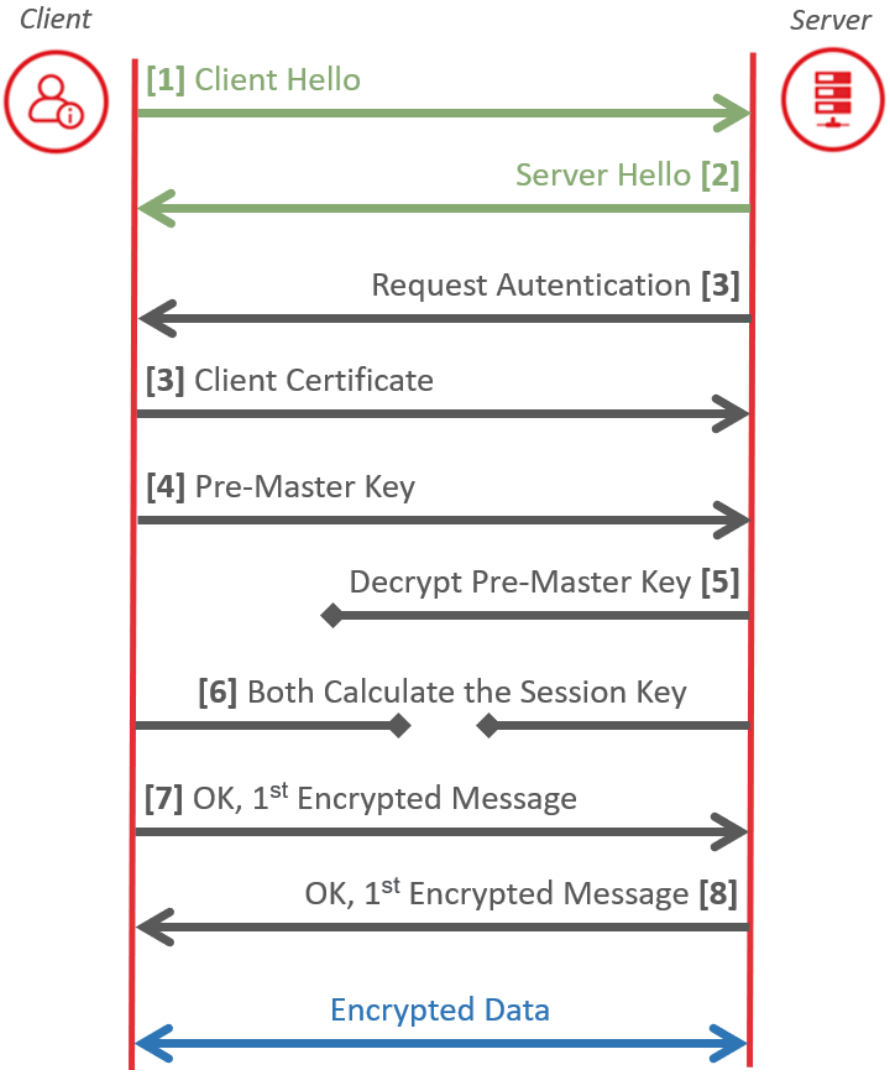


Figura 3.10: Etapas do processo TLS *Handshake* [111, 113].

3.7.3 Certificados Digitais

Na Seção 3.7.2, foi referido o conceito de certificado como sendo a prova de idoneidade do servidor perante o cliente, ou o meio de autenticação do cliente perante o servidor. Efetivamente, um certificado digital é um ficheiro assinado por uma Entidade Certificadora (CA) que contém, entre outras informações, uma chave pública.

De seguida, apresentam-se os passos necessários para a criação de um par chave pública - chave privada, estando a primeira contida num certificado devidamente assinado [114]:

- Criar uma chave privada (normalmente guardada num ficheiro denominado "client_key.pem");
- Usar a chave privada para gerar um Pedido de Assinatura de Certificado (normalmente guardado num ficheiro denominado "client_csr.pem"), especificando

o nome do domínio, assim como outros detalhes da organização;

- Enviar o Pedido de Assinatura de Certificado para a CA, que o irá assinar utilizando a sua chave privada;
- Receber da Entidade Certificadora o certificado assinado (normalmente guardada num ficheiro denominado por "client_cert.pem"), bem como o certificado que contém a chave pública da CA (normalmente guardada num ficheiro denominado por "ca_cert.pem"), que permite comprovar a autenticidade do certificado do cliente.

O procedimento descrito foi apresentado para um certificado de cliente, mas é exatamente análogo para um certificado de servidor.

Neste capítulo efetuou-se uma abordagem de conceitos relativos a técnicas, tecnologias e protocolos a utilizar nos desenvolvimento do sistema a desenvolver. No próximo capítulo é apresentada a proposta de um sistema de deteção e contagem, envolvendo tanto a componente de *Hardware* como a de *Software*.

Capítulo 4

Sistema de Deteção e Contagem

Neste capítulo é apresentada uma proposta tanto da arquitetura de *Hardware* como de *Software* relativa ao sistema a desenvolver. Para tal, foram tidos em consideração os requisitos pré-definidos (Seção 1.1), assim como as características dos produtos e novas tecnologias existentes no mercado (Capítulo 2).

4.1 Arquitetura de *Hardware*

Nesta seção é efetuada a descrição da arquitetura de HW do sistema. Inicialmente, é apresentada uma arquitetura de alto nível, seguida de um *trafe-off* de equipamentos que desempenhem as funcionalidades dos módulos que constituem o sistema. Escolhidos os equipamentos, é apresentado o esquema de ligações elétricas. Por fim, é feito um projeto mecânico do sistema, assim como um balanço do custo dos equipamentos do sistema.

4.1.1 Arquitetura de Alto Nível

Com base no estudo do estado da arte (Capítulo 2), reconheceram-se vários tipos de elementos de um sistema para a análise de tráfego em *smart cities* baseado em visão computacional. Identificaram-se os pontos positivos e negativos que, em conjunto com uma análise de módulos de HW atuais, permite propor um conceito de sistema mais refinado.

Desta forma, é representada na Figura 4.1 a proposta de solução, correspondente à arquitetura de alto nível do HW do sistema. Esta é constituída por 12 módulos,

sendo apenas o servidor remoto exterior ao produto. Os restantes módulos estão embebidos no interior de uma caixa de isolamento, sendo estes: sensor de temperatura; cartão de memória; módulo de ventilação; módulo de conversão AC/DC; módulo de respiração; *Light-Emitting Diode* (LED); câmara; cartão *Subscriber Identity Module* (SIM) e módulo IoT.

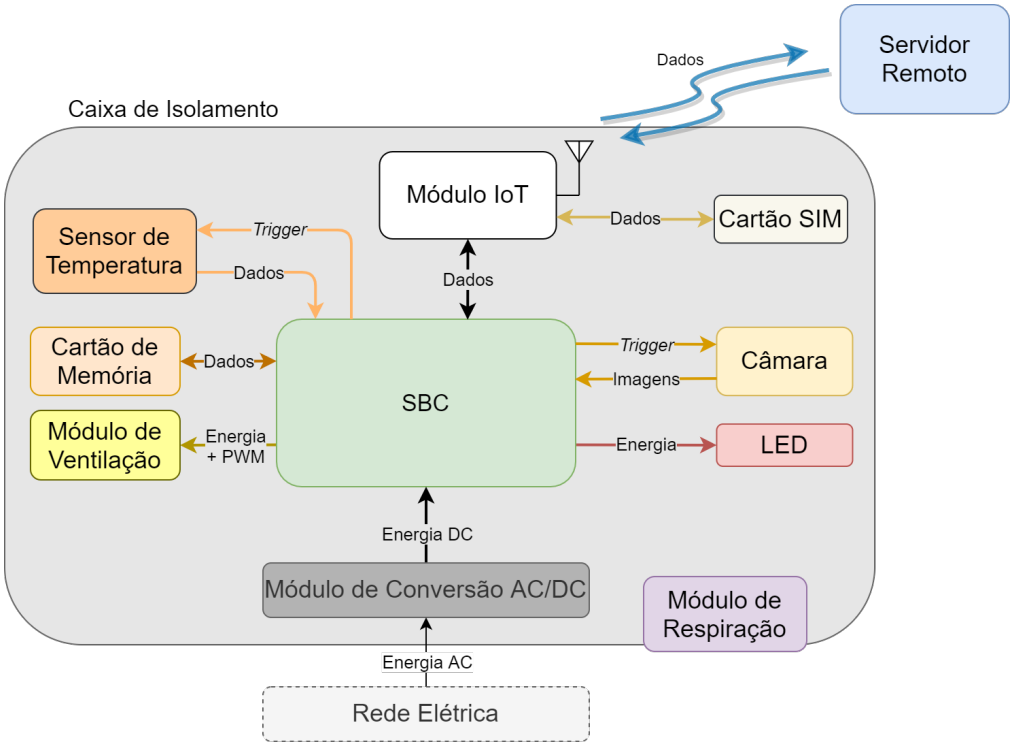


Figura 4.1: Arquitetura de alto nível do HW do sistema.

Quanto ao modo de fixação de todos estes módulos no interior da caixa de isolamento, seguiu-se a norma *Deutsches Institut für Normung* (DIN) utilizada para padronizar o processo de fixação de equipamentos [115]. Trata-se de um padrão importante no setor industrial, garantindo alto nível qualidade na fabricação dos produtos, impactando diretamente a segurança do consumidor final. Apresenta como vantagens a redução de desperdícios, sendo mais económico, e a eliminação das barreiras técnicas graças à sua eficiência e fácil instalação.

Segue-se agora uma breve descrição da função de cada um dos módulos presentes na arquitetura de alto nível do HW do sistema:

- **SBC** - consiste no principal módulo do sistema, sendo responsável por todo o processamento de dados e estabelecimento da comunicação com os restantes módulos periféricos do sistema;
- **Cartão de Memória** - módulo de memória para o armazenamento de dados, assim como do SO, ficheiros de configuração e restantes ferramentas de SW necessárias para o normal funcionamento do sistema;

- **Câmara** - módulo sensorial para a aquisição de imagens de vídeo a processar pelo SBC. Esta aquisição é controlada pelo SBC, através de um sinal de *trigger*, sempre que pretenda receber imagens da câmara;
- **LED** - emissor de luz que notifica o correto funcionamento do sistema, nomeadamente o processo de análise de imagens;
- **Sensor de Temperatura** - sensor que faz a aquisição da temperatura interior da caixa de isolamento com a finalidade de assegurar as condições mínimas de funcionamento do sistema;
- **Módulo IoT** - módulo que estabelece uma comunicação bidirecional entre o sistema local e o servidor remoto. Por um lado, o sistema reporta dados da análise de tráfego para o servidor, por outro lado, o servidor tem a possibilidade de enviar instruções para reconfigurar o funcionamento do sistema;
- **Cartão SIM** - cartão de comunicação que estabelece a ligação do módulo IoT à rede móvel de um operador. Desta forma, é reforçado o cumprimento do RGPD, visto que a transmissão de dados é realizada através de faixas de frequência LTE licenciadas;
- **Caixa de Isolamento** - módulo que garante o isolamento de todo o sistema do meio exterior, assegurando um Índice de Proteção mínimo (IP65) que garanta proteção perante condições meteorológicas diversas;
- **Módulo de Ventilação** - módulo que ajuda no processo de arrefecimento do SBC, juntamente com um dissipador de calor passivo, como forma de prevenção de sobreaquecimentos. O funcionamento deste módulo é regulado segundo um sinal *Pulse Width Modulation* (PWM) para controlar a refrigeração e consumo energético;
- **Módulo de Respiração** - módulo que permite o fluxo de ar entre o interior da caixa de isolamento e o meio exterior, diminuindo a acumulação do calor dissipado pelos módulos. Desta forma, são melhoradas as condições de funcionamento do sistema com níveis de temperatura mais reduzidos, aumentando vida útil dos módulos sem que o Índice de Proteção (IP) do sistema seja comprometido;
- **Módulo de Conversão AC/DC** - módulo responsável por fornecer alimentação a todo o sistema, proveniente da rede elétrica pública em corrente contínua;
- **Servidor Remoto** - módulo remoto que estabelece comunicação com o módulo IoT do sistema. Tem como função a recolha dos dados reportados pelo sistema, assim como o envio de instruções.

4.1.2 Módulos

Abordados os vários módulos do sistema, realizou-se um *trade-off* (Anexo B) face aos equipamentos comerciais candidatos a cada um dos módulos. No final deste processo foram selecionados os que melhor satisfazem os requisitos impostos, nomeadamente o custo. Segue-se a lista dos equipamentos escolhidos.

- SBC

Tendo por base a análise de SBC (Seção 2.3.1), a NVIDIA Jetson Nano *Developer Kit* corresponde à melhor opção [30], visto que o Raspberry Pi 4 Modelo B, embora seja significativamente mais barato, possui um GPU com menos capacidades computacionais. O Google Coral *Dev Board* é ainda recente no mercado e não possui um bom suporte à comunidade nem existem muitos projetos utilizando este SBC. Adicionalmente, a Jetson Nano apresenta outros aspetos que levaram à sua seleção:

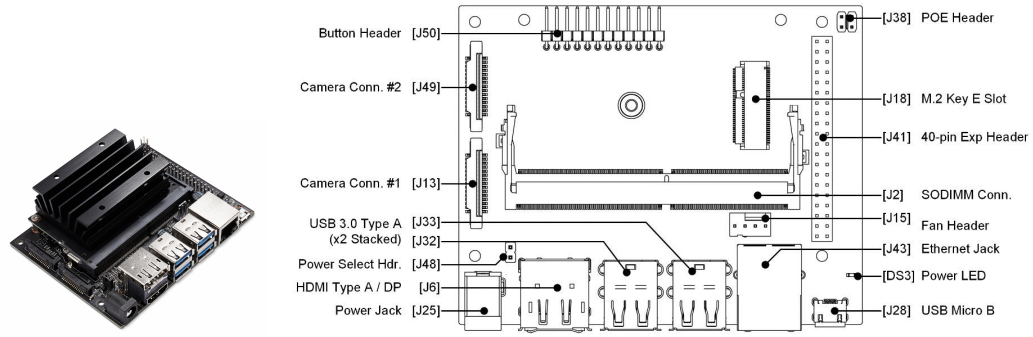
- Melhor relação qualidade-preço;
- Menor tempo de computação para certos modelos de DL [41];
- Mais indicado para utilização de um único modelo de DL, por longos períodos de tempo [44];
- Consumo energético bastante menor em modo *idle* [44];
- SW mais flexível, permitindo o processamento, em simultâneo, de imagens HD provenientes de múltiplos sensores de *streaming*, recorrendo a modelos de DL [37];
- Compatibilidade com uma maior quantidade de modelos de DL [37].

Na Figura 4.2a está representado o produto NVIDIA Jetson Nano *Dev Kit* B01, que integra um dissipador de calor. Na Figura 4.2b são identificados os vários periféricos que constituem esta *board*.

- Câmara

Tendo por base a análise de câmaras (Seção 2.3.2), a Raspberry Pi Camera Module v2 corresponde à melhor opção porque é compatível com a NVIDIA Jetson Nano, ao contrário da Raspberry Pi Camera Module v1 [116]. Desta forma, a Coral Camera é descartada porque o SBC selecionado não foi o SBC da Coral.

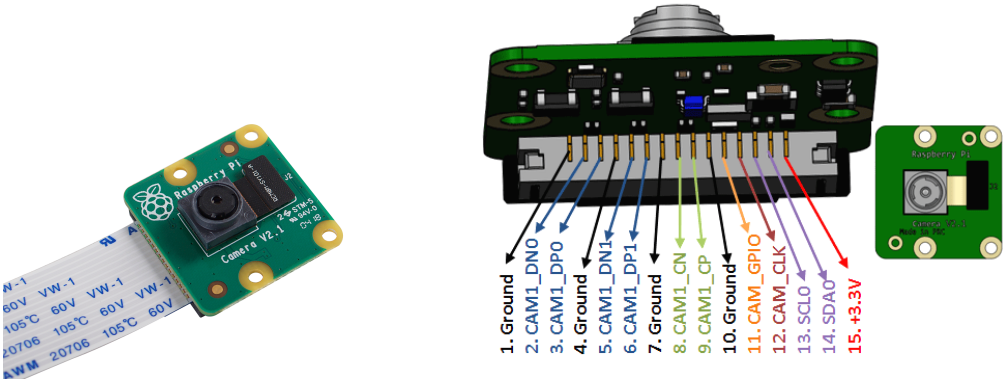
Na Figura 4.3a é apresentado o módulo de câmara, seguido do seu *pinout* na Figura 4.3b. A sua interface de comunicação é do tipo *Mobile Industry Processor Interface* (MIPI) CSI-2.



(a) Representação do produto

(b) Periféricos da *board*

Figura 4.2: NVIDIA Jetson Nano *Development Kit* B01 [30].



(a) Representação do produto [117]

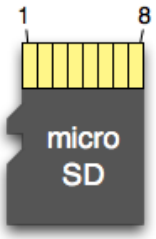
(b) Respetivo *pinout* [118]

Figura 4.3: Módulo de câmara v2 da Raspberry Pi.

- Cartão de Memória

Para a instalação do SO, ferramentas adicionais e SW desenvolvido foi utilizado um cartão Micro-SD UHS-1 com capacidade mínima de 32 GB de memória (Figura 4.4a), conforme recomendado pela NVIDIA para a Jetson Nano. O seu *pinout* encontra-se representado na Figura 4.4b.

(a) Representação do produto [119]



Pin	SD	SPI
1	DAT2	X
2	CD/DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	X

(b) Respetivo *pinout* [120]

Figura 4.4: Cartão de memória Micro-SD UHS-1 32GB.

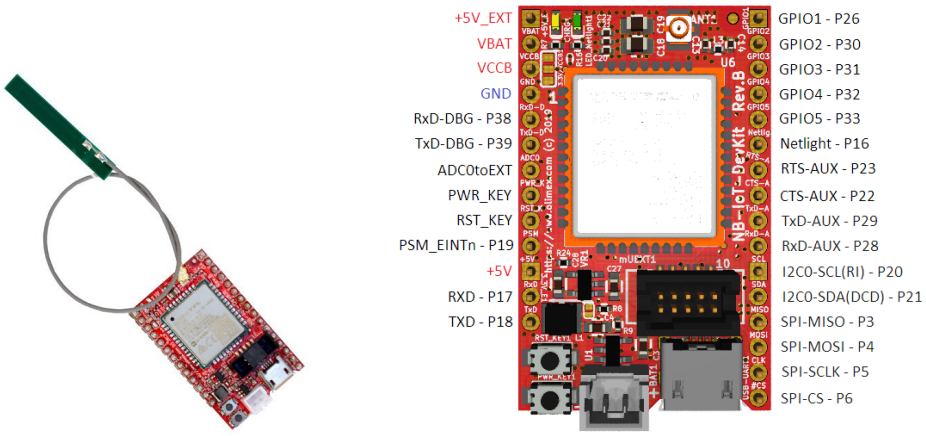
- Módulo IoT

Após um estudo dos módulos de comunicação mais comuns no âmbito de IoT (Seção 2.6), chegou-se à conclusão que a tecnologia NB-IoT é a melhor opção porque funciona com faixas de frequência licenciadas [71], tornando a transmissão de dados mais segura e fiável através da rede de um operador móvel. A sua principal desvantagem é o consumo energético devido ao mecanismo QoS incorporado. Adicionalmente, é o que apresenta menor latência e permite uma grande quantidade de dispositivos ligados e mensagens com maior *payload*.

Optou-se por utilizar o NB-IoT-DevKit. Este *kit* de desenvolvimento consiste numa *board* com um módulo BC66 LTE GSM da Quectel que opera nas faixas GSM (Figura 4.5a). Trata-se de um produto da Olimex com uma antena GSM incluída [121]. Permite transferências a uma velocidade de 25,5 kb/s e consumos de apenas 126 µA em modo de poupança de energia. Segundo a Figura 4.5b, este módulo possui as seguintes interfaces de comunicação: *slot* para cartões Nano-SIM; Micro-USB para fácil implementação; 5 pinos GPIO; *Inter-Integrated Circuit* (I2C); *Serial Peripheral Interface* (SPI) e *Universal Asynchronous Receiver-Transmitter* (UART). Apresenta também um LED de estado e as dimensões da *board* são de 40×25 mm.

- Cartão SIM

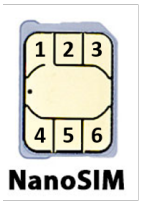
O módulo NB-IoT requer um cartão SIM que suporte o protocolo de comunicação NB-IoT. Apesar da tecnologia GSM funcionar com LTE, necessita de um novo



(a) Representação do produto (b) Respetivo *pinout*

Figura 4.5: *Board* de desenvolvimento NB-IoT-DevKit com um módulo BC66 [121].

protocolo de instalação e do pagamento de uma taxa de licenciamento que, por omissão, não está habilitada. Desta forma adquire-se um cartão SIM específico para aplicações NB-IoT com um *plafond* de 5 MB, que corresponde ao valor máximo disponibilizado pela operadora da rede móvel. Na Figura 4.6a está representado um cartão Nano-SIM, seguido do seu *pinout* na Figura 4.6b.



Pino	Descrição
1	GND
2	VPP
3	I/O
4	Vcc
5	Reset
6	CLK

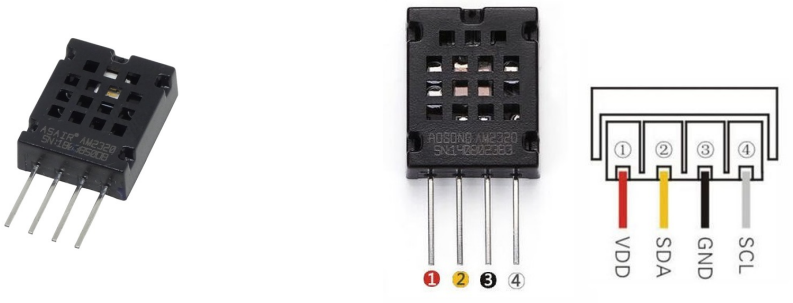
(a) Representação do produto (b) Respetivo *pinout*

Figura 4.6: Cartão Nano-SIM para NB-IoT.

- Sensor de Temperatura

A escolha do sensor de temperatura resultou baseou-se nos seguintes requisitos: fácil implementação; resolução de pelo menos uma casa decimal e de baixo custo. Na Tabela B.1 é feito o levantamento de 3 sensores. O sensor escolhido foi o AM2320 (Figura 4.7a), não só por ser o módulo mais barato, mas também por existir uma biblioteca de *drivers* que facilita a sua integração no sistema.

O AM2320 corresponde a um sensor digital de temperatura e humidade com um protocolo de comunicação I2C e alimentação a 3,3 V ou 5 V. Na Figura 4.7b é indicado o *pinout*.



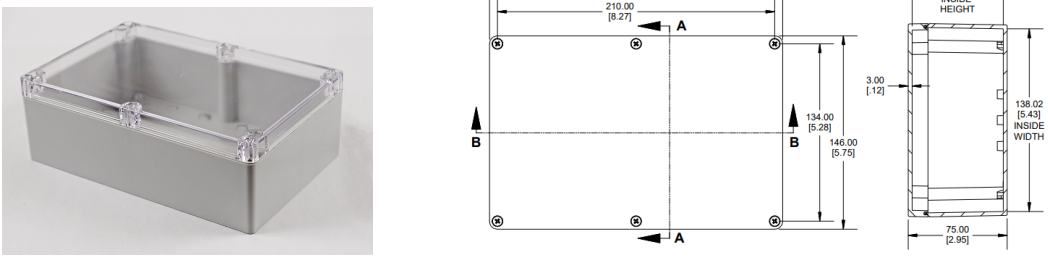
(a) Representação do produto [122] (b) Respetivo *pinout* [123]

Figura 4.7: Sensor AM2320.

- Caixa de Isolamento

A escolha da caixa de isolamento indicada para este sistema foi realizada tendo em conta IP dos produtos de mercado para análise de tráfego (Anexo A), deve ser garantido um IP mínimo correspondente ao IP65; ter uma zona de transparência que permita a aquisição de imagens; baixo custo unitário e, por fim, o material ser indicado para aplicações em ambiente *outdoor*. Desta forma na Tabela B.2 estão listadas 5 opções, tendo sido escolhida a 2ª opção de menor custo, pelo facto de o seu material (Policarbonato) ser indicado para o meio exterior, ao contrário da alternativa mais barata (Plástico ABS). O IP é IP65, sendo a tampa totalmente transparente.

Esta caixa de isolamento (Figura 4.8a), modelo RZ0232C da Hammond Manufacturing [124, 125], tem dimensões exteriores de 222×146×75 mm (Figura 4.8b) e um peso de 0,57 kg.



(a) Representação do produto (b) Respetivas dimensões em milímetros

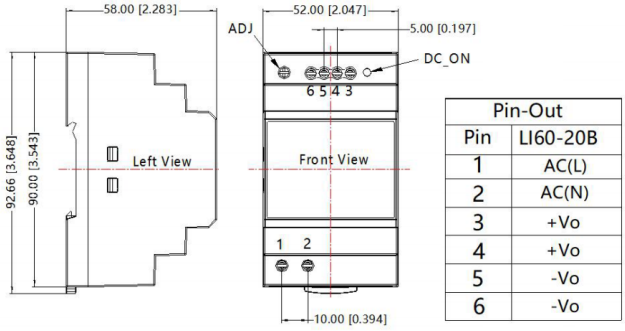
Figura 4.8: Caixa de isolamento RZ0232C [124].

- Conversor AC/DC

A seleção do módulo de alimentação teve como requisitos: saída de alimentação de 5 V *Direct Current* (DC) a 3 A, correspondente ao consumo máximo do SBC escolhido (15 W); seguir a norma DIN; ter dimensões adequadas para incluir na caixa de isolamento escolhida. Assim sendo, na Tabela B.3 é feito o levantamento de 6 conversores. Foi escolhido o de maior potência visto ser a opção com melhor relação preço-potência.

Este conversor AC/DC é o modelo MP-LI60-20B05PR2 da Multicomp Pro (Figura 4.9a) que disponibiliza 32,5 W de potência para 2 saídas 5 V DC e valores de tensão de entrada 85-264 V *Alternating Current* (AC) [126]. As suas dimensões, assim como o *pinout*, são indicados na Figura 4.9b. Trata-se de um produto que segue a norma DIN e com dimensões 35×7,5 mm.

(a) Representação do produto



Pin-Out	
Pin	LI60-20B
1	AC(L)
2	AC(N)
3	+Vo
4	+Vo
5	-Vo
6	-Vo

(b) Respetivas dimensões e *pinout*

Figura 4.9: Conversor AC/DC MP-LI60-20B05PR2 [126].

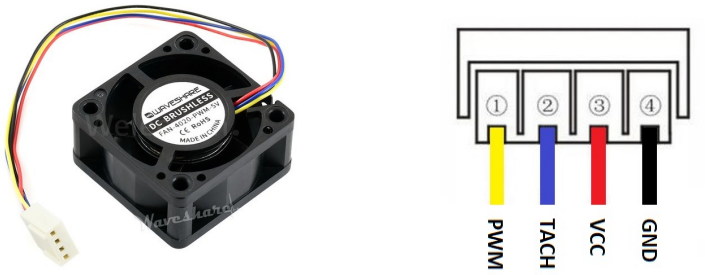
- Módulo de Ventilação

Para seleção do módulo de ventilação, definiram-se como requisitos: dimensões 40×40 mm (comprimento×largura) para instalação dos furos do dissipador de calor do SBC escolhido; possibilidade de controlo por sinal PWM e alimentação a 5 V DC. Na Tabela B.4 são listadas 4 opções, tendo sido escolhida a única que permite o controlo por PWM e que cumpre as dimensões requeridas.

Este módulo de ventilação (Figura 4.10a), Fan-4020-PWM-5V [127] é um produto dedicado da Jetson Nano. Este é alimentado a 5 V DC, possui um controlo por PWM e apresenta dimensões 40×40×20 mm. O seu *pinout* está representado na Figura 4.10b.

- Módulo de Respiração

Na seleção do módulo de respiração, foram tidos em conta: o baixo preço unitário, garantir o Índice de Proteção da caixa de isolamento (IP65) e fácil instalação. A

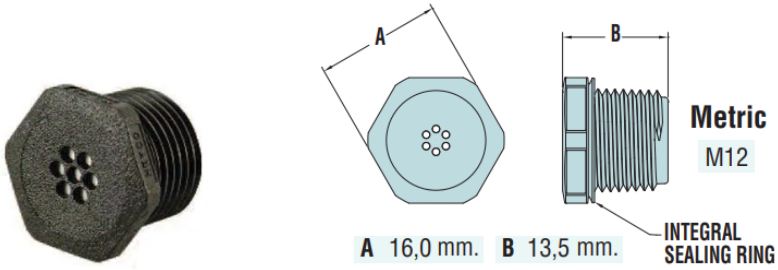


(a) Representação do produto (b) Respetivo *pinout*

Figura 4.10: Módulo de ventilação Fan-4020-PWM-5V [127].

Tabela B.4 lista 7 opções. O produto escolhido foi o de menor custo, visto satisfazer os restantes requisitos.

Este módulo de respiração (Figura 4.11a), modelo 3582VB da Heyco [128], consiste num filtro que garante a vedação do orifício onde é montado, permitindo a passagem de ar para ventilação, assim como a dissipação de calor. Por outro lado, impede a entrada de humidade e partículas, garantindo o IP68. A sua compensação de pressão impede a acumulação de pressão dentro da caixa devido ao fluxo da temperatura ambiente. Por fim, a Figura 4.10b identifica as dimensões deste produto.



(a) Representação do produto (b) Respetivas dimensões

Figura 4.11: Módulo de respiração Heyco 3582VB [128].

4.1.3 Desenvolvimento do Sistema

Escolhidos os equipamentos, procedeu-se ao desenvolvimento do sistema, sendo feita a interligação elétrica, conforme o esquema representado na Figura 4.12. Este esquema assegura não só a alimentação dos vários equipamentos elétricos, como também a comunicação entre os equipamentos.

A Figura 4.12 detalha as seguintes ligações:

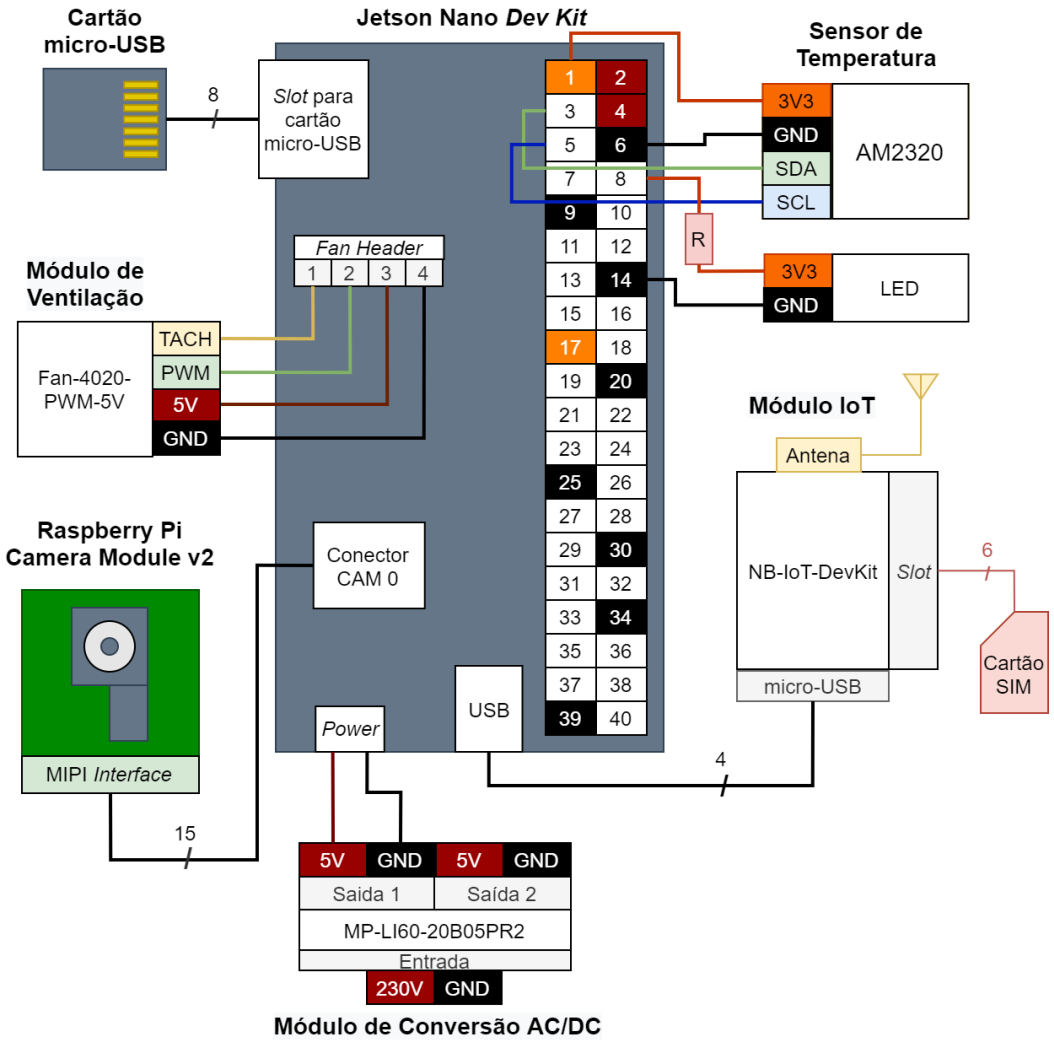


Figura 4.12: Diagrama de ligações dos módulos de HW do sistema.

- O **cartão Micro-SD** estabelece a ligação dos seus 8 pinos ao *slot* da Jetson Nano *Dev Board*, sendo a comunicação realizada segundo protocolo de comunicação SPI;
- O **módulo de ventilação** Fan-4020-PWM-5V conecta-se ao conector J15 da Jetson Nano *Dev Board*, sendo alimentado a 5 V, controlado pelo pino PWM e envia *feedback* da sua rotação pelo pino TACH;
- A **Raspberry Pi Camera Module v2** é ligada ao *slot* CAM0 na Jetson Nano *Dev Board* através de uma fita de 15 pinos com interface de comunicação MIPI CSI-2;
- O **conversor AC/DC**, MP-LI60-20B05PR2, é ligado à rede elétrica que fornece 230 V AC, e disponibiliza 5 V DC no conector *Jack* de *power* da Jetson Nano *Dev Board*;

- O **cartão SIM** estabelece a ligação dos seus 6 pinos com *slot* para cartão Nano-SIM do NB-IoT-DevKit;
- O **módulo IoT** NB-IoT-DevKit estabelece comunicação com a Jetson Nano *Dev Board*, segundo o protocolo de comunicação UART, através de um cabo Micro-USB/USB, e alimenta o NB-IoT-DevKit a 5 V;
- O **sensor de temperatura** AM2320 conecta-se ao *header* de 40 pinos GPIO (1 e 6 para alimentação a 3,3 V e os pinos SDA e SCL da comunicação I2C aos pinos 3 e 5);
- O **LED** é alimentado a 3,3 V através da ligação aos pinos 8 e 14 do *header* de 40 pinos GPIO. Em série, possui uma resistência R de 200 Ω para proteção do LED. Esta resistência foi calculada segundo a Equação 4.1, deduzida do esquema eléctrico representado na Figura 4.13, onde o V_{GPIO} corresponde à tensão do pino (3,3 V); V_{LED} é a tensão de polarização do LED (2,1 V) e I_{LED} a corrente que passa pelo LED (20 mA).

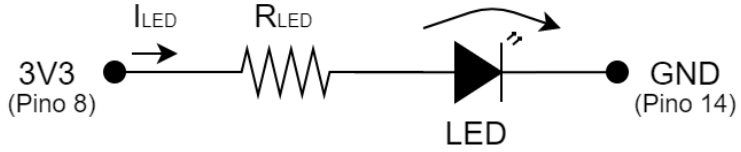


Figura 4.13: Esquema eléctrico de ligação do LED ao *header* de 40 pinos da Jetson Nano *Dev Board*.

$$R_{LED} = \frac{V_{GPIO} - V_{LED}}{I_{LED}} \tag{4.1}$$

Sabendo que o *plafond* do cartão SIM é de 5 MB/mês, é importante calcular-se o limite de mensagens que o sistema pode reportar por minuto, desprezando as mensagens iniciais de configuração do módulo à rede. Tendo em conta que uma mensagem MQTT é constituída por: *header* (2 B); *over head* (5 B, no máximo) e *payload* (19 B, sendo 10 B relativos ao *timestamp*, 3 B para o número de pessoas, 3 B para o número de carros e 3 B para o número de bicicletas), cada mensagem totaliza 26 B. Através da Equação 4.2 conclui-se que o sistema pode reportar até 4,3 mensagens por minuto.

$$Freq_{Max} = \frac{Plafond}{Tempo \times Msg} = \frac{5\ [\frac{\text{MB}}{\text{mes}}]}{44640\ [\frac{\text{min}}{\text{mes}}] \times 26\ [\text{B}]} \approx 4,3\ [\frac{\text{Msg}}{\text{min}}] \tag{4.2}$$

4.1.4 Projeto Mecânico

Visto que se trata do desenvolvimento completo de um sistema, é apresenta-se o projeto mecânico que representa a organização e fixação dos vários equipamentos no

interior da caixa de isolamento, assim como as ligações com o meio exterior, sem comprometer o IP65.

Com vista a estabelecer a ligação elétrica entre a rede elétrica pública e o interior o conversor AC/DC, é necessário uma perfuração na caixa de isolamento. Para tal, fez-se um *trade-off* de bucins (Tabela B.6) e de blocos terminais DIN (Tabela B.7) para fixação das ligações de fase, neutro e terra do cabo de alimentação. Destes *trade-offs*, foram escolhidos os produtos mais baratos de ambos os tipos, visto que satisfazem os requisitos de proteção e capacidade, respetivamente. As Figuras 4.14a e 4.14b representam o bucim e o bloco terminal selecionados.

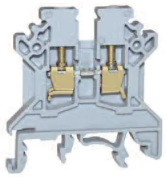
Adoptada a norma DIN, analisaram-se as calhas DIN 35×7,5, tendo sido selecionado o produto de menor custo representado na Figura 4.14c.

Para fixação do *kit* de desenvolvimento Jetson Nano na calha, surgiu a necessidade de desenvolver um suporte com furação compatível com a *board* da Jetson Nano. O modelo 3D deste produto está representado na Figura 4.14d, tendo sido desenvolvido na aplicação *web* Tinkercad da Autodesk e impresso numa impressora 3D.

Como forma de fixação dos restantes módulos (câmara; módulo NB-IoT; sensor de temperatura e LED), criou-se uma *board* para acoplar ao conector de 40 pinos da *board* da Jetson Nano, recorrendo a um *header* de expansão de pinos. Nas Figuras 4.14e e 4.14f está representada a *board* perfurada e o *header* de expansão, respetivamente.

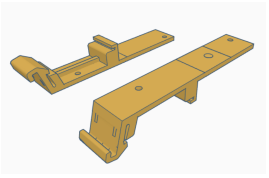
Por fim, analisaram-se diferentes espaçadores (Tabela B.8) e elementos de fixação (Tabela B.9) para fixar a PCB perfurada à *board* da Jetson Nano e da fixação da Jetson Nano ao suporte DIN. Foram escolhidos os elementos de zinco de menor custo para maior durabilidade.

(a) Bucim

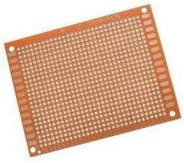


(b) Blocos terminais DIN

(c) Calha DIN



(d) Suporte DIN para Jetson Nano



(e) PCB perfurada

(f) *Header* expansor

(g) Espaçadores

(h) Elementos de fixação

Figura 4.14: Elementos para interface do sistema com a fonte de energia (a-b) e elementos de fixação dos módulos (c-h).

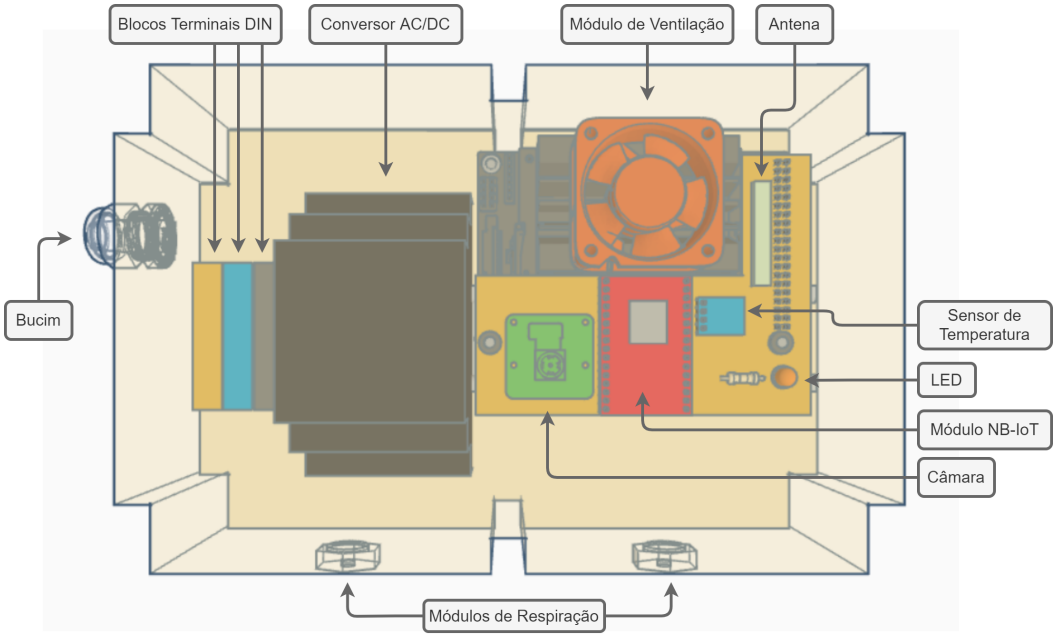
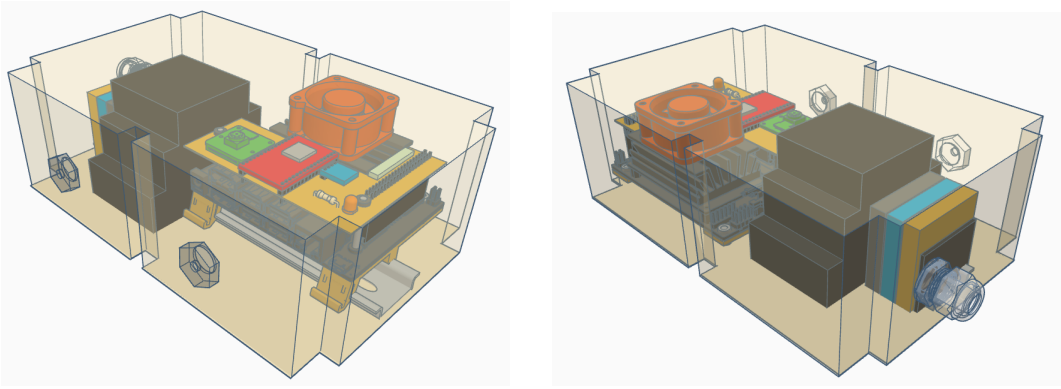


Figura 4.15: Identificação dos principais elementos no interior da caixa de isolamento.

Na Figura 4.15 é feita a identificação dos vários elementos do sistema dispostos no interior da caixa de isolamento. A Figura 4.16 apresenta o modelo tridimensional do produto.



(a) Vista frontal (b) Vista posterior

Figura 4.16: Modelo tridimensional do produto.

4.1.5 Custos do Sistema

Tendo em conta que se pretende desenvolver um sistema de baixo custo, a Tabela 4.1 lista todo o material, incluindo quantidades e preços unitários. Associado a cada material, é indicada a referência do local de compra. De salientar que, nesta tabela, o valor total apresentado não contabiliza:

- Custos de horas de engenharia no desenvolvimento deste produto;
- Imposto sobre Valor Acrescentado (IVA);
- Custos dos portes de envio do material;
- Tarifário mensal do *plafond* do cartão SIM.

Tabela 4.1: Tabela de preços elementos constituintes do produto (sem IVA incluído).

Módulo	Designação do Produto	Unidades	Custo (€)
SBC	NVIDIA Jetson Nano Developer Kit - B01 [129]	1	92,27
Câmara	Raspberry Pi Camera Module v2 [130]	1	16,38
IoT	Olimex NB-IoT DevKit [131]	1	16,22
Cartão SIM	Cartão SIM para NB-IoT da Altice[a]	1	2,50
Memória	MicroSD UHS-1 de 32 GB Kingston [132]	1	4,59
Temperatura	AM2320 [133]	1	2,72
Ventilação	Fan-4020-PWM-5V [134]	1	2,00
Respiração	Heyco 3582VB [135]	2	4,56
Conversor	MP-LI60-20B05PR2 [136]	1	13,72
Caixa	Hammond Manufacturing RZ0232C [137]	1	17,05
LED	LTL2R3KRD-EM [138]	1	0,07
Resistência	603-CFR-12JT-52-220R [139]	1	0,07
Bucim	PELB0265 [140]	1	0,16
Bloco terminal	Altech CTS2.5U-N [141]	3	0,76
Calha DIN	Phoenix Contact 5603400 [142]	1/3[b]	7,92
Suporte do SBC	N/A[c]	2	0,02
PCB perfurada	BusBoard Prototype Systems ST1 [143]	1	2,58
Header expansor	Adafruit 1979 [144]	1	2,03
Espaçadores	Wurth Elektronik 971250154 [145]	3	0,61
Parafusos	M2.510PRSTMCZ100- [146]	4/100[d]	2,06
Porcas	M2.5- HFST-Z100- [147]	10/100[d]	1,36
		Total:	182,41

[a] Elemento adquirido pelo CEiiA em contacto com a operadora móvel.
[b] Cada unidade resulta em 3 elementos deste tipo.
[c] Elemento produzido no CEiiA por uma impressora 3D com filamento em PLA. Este filamento apresenta um preço de mercado de aproximadamente 20 €/kg (sem IVA). Visto a massa de cada unidade ser de 1,1 g, o valor unitário é cerca de 0,02 €.
[d] Cada quantidade contém 100 unidades, desta forma são indicadas as unidades necessárias por produto.

Tendo em conta os preços dos produtos de mercado analisados na Seção 2.1.1, o custo dos componentes é de 182,41 €, valor significativamente baixo para o desenvolvimento de um sistema *low-cost*, recorrendo a tecnologias recentes do mercado. O custo apresentado refere-se ao preço de um protótipo, sendo que este valor baixará significativamente numa produção em escala, sem contabilização dos custos associados às licenças e certificações do produto.

4.2 Arquitetura de *Software*

Nesta seção é efetuada a descrição da arquitetura de SW do sistema. Apresenta-se a estrutura geral do SW seguida de uma descrição detalhada de cada um dos módulos.

4.2.1 Estrutura do *Software*

A Figura 4.17 representa uma *overview* da estrutura de SW e da interligação dos diferentes elementos se interligam entre si. Inicialmente, são criadas *queues* para a entre os elementos. Posteriormente, é feita a criação dos vários elementos, denominados por processos, que funcionam em simultâneo. Estes processos subdividem-se em três categorias: *loggers*, contagem de pessoas e de veículos e comunicação IoT.

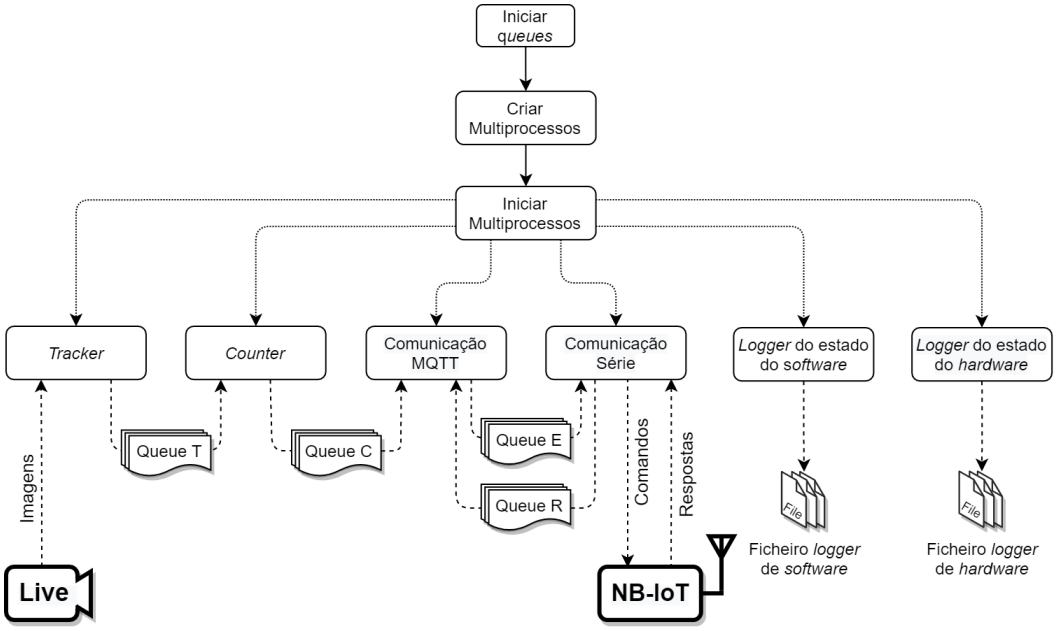


Figura 4.17: *Overview* da relação entre os principais blocos de SW e os meios de comunicação associados.

Segue uma breve descrição da função de cada processo, *queues* e ficheiros associados:

- ***Tracker*** - recolhe, em tempo-real, imagens da câmara em modo *streaming*; faz a deteção, classificação e *tracking* de pessoas e veículos nas imagens. Depois

de processar, a cada imagem, reporta para a *Queue T* os dados de posição dos objetos *tracked*;

- ***Counter*** - analisa, sequencialmente, os dados provenientes da *Queue T* e identifica o cruzamento, ou não, dos objetos *tracked* por uma certa região. Caso algum se cruze, a informação é publicada na *Queue C*;
- **Comunicação MQTT** - responsável por, inicialmente, ligar o sistema ao *broker* MQTT através do módulo NB-IoT. De seguida, envia comandos para o módulo e recebe as respostas através das *Queue E* e *Queue R*, respetivamente. As respostas consistem em mensagens provenientes do operador do servidor remoto enviadas para o *broker* MQTT. Os comandos correspondem a mensagens pré-definidas, nomeadamente relativas a dados de contagem de pessoas e veículos no último período de tempo;
- **Comunicação Série** - estabelece a ponte de ligação entre o processo de comunicação MQTT e o módulo NB-IoT, sendo responsável por enviar, através do protocolo UART (porta série), o conteúdo recebido da *Queue E* para o módulo NB-IoT, assim como por receber a informação adquirida pelo módulo NB-IoT e enviá-la para a *Queue R*;
- ***Logger* do estado do SW** - recolhe informações do estado do SW ao longo do tempo e guarda-as num ficheiro local;
- ***Logger* do estado do HW** - recolhe informações do estado do HW ao longo do tempo e guarda-as num ficheiro local.

Segue-se nas seguintes subseções uma descrição mais detalhada sobre cada um dos 6 processos apresentados.

4.2.2 Processo - *Tracker*

A Figura 4.18 representa um diagrama do funcionamento do processo de *tracking* de objetos. Este começa por configurar um LED que indica se a aquisição de imagens está funcional. Posteriormente, são criados os elementos que constituem o processo, importando dos ficheiros de configuração os parâmetros necessários. De seguida, é criada e iniciada a *pipeline* de *tracking* de objetos.

Uma vez arrancada, a *pipeline* inicia a leitura de imagens provenientes da câmara. Para cada imagem recebida, a *pipeline* deteta e classifica objetos através de algoritmos de DL. Após esta fase, é atribuído um ID a cada novo objeto detetado e, recorrendo ao algoritmo de *tracking*, os objetos são identificados nas consecutivas imagens com o mesmo ID, sendo actualizada a sua posição ao longo do tempo. Ao fim de cada ciclo de leitura de imagens e *tracking* de objetos, é enviada para a *Queue T* a localização destes objetos.

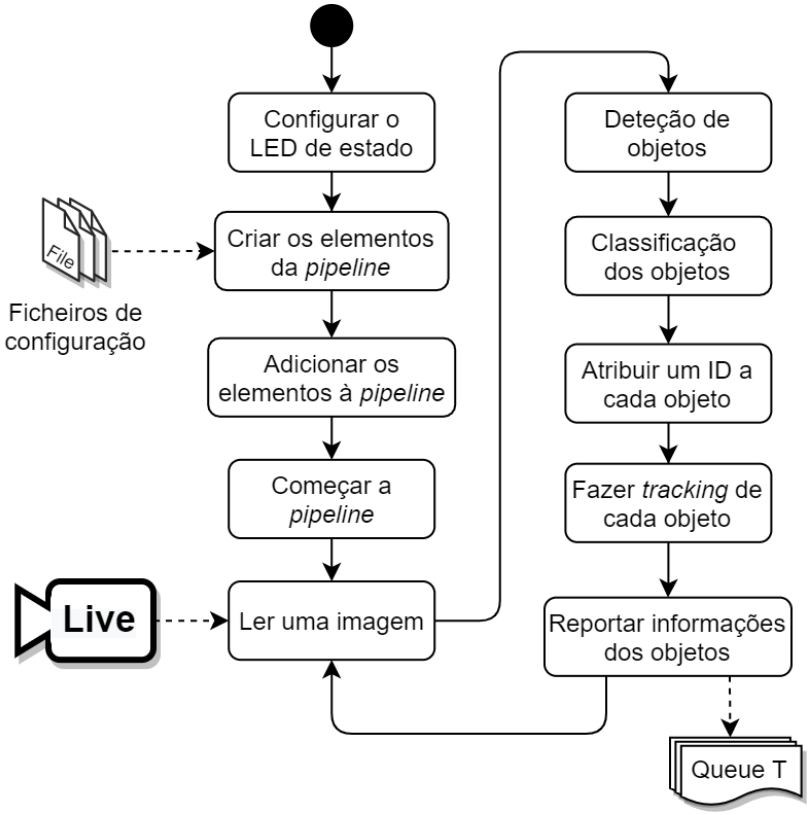


Figura 4.18: Diagrama de funcionamento do processo *Tracker*.

4.2.3 Processo - *Counter*

O processo de contagem de objetos, classificados como pessoas ou veículos é apresentado no diagrama da Figura 4.19. Inicialmente, este inicia as linhas de contagem previamente definidas. Se existem mensagens na *Queue* T, analisada a mensagem mais antiga relativa aos objetos *tracked*. Caso o ID de algum objeto já tenha sido identificado em imagens anteriores, a sua localização é atualizada na lista de ID; caso contrário, o objeto é adicionado à lista. Durante esta constante atualização da lista de ID se, em algum instante, um objeto *tracked* cruza a linha de contagem, é removido da lista de ID e regista a contagem na *Queue* C. Por fim, sempre que a *Queue* T está vazia, removem-se da lista ID os objetos que deixaram de ser *tracked* durante um certo período de tempo.

- Método de Contagem

A contagem dos objetos detetados no processo *Counter* utiliza um método abordado [26] (Seção 2.2), baseado numa linha virtual de contagem. Na Figura 4.20a estão apresentadas 3 *bounding boxes* de objetos com diferentes ID, assim como um segmento de reta perpendicular ao movimento destes objetos (linha de contagem). A Figura 4.20b ilustra a aplicação técnica, sendo $\overline{AB}$ a linha de contagem $LC(x)$; a

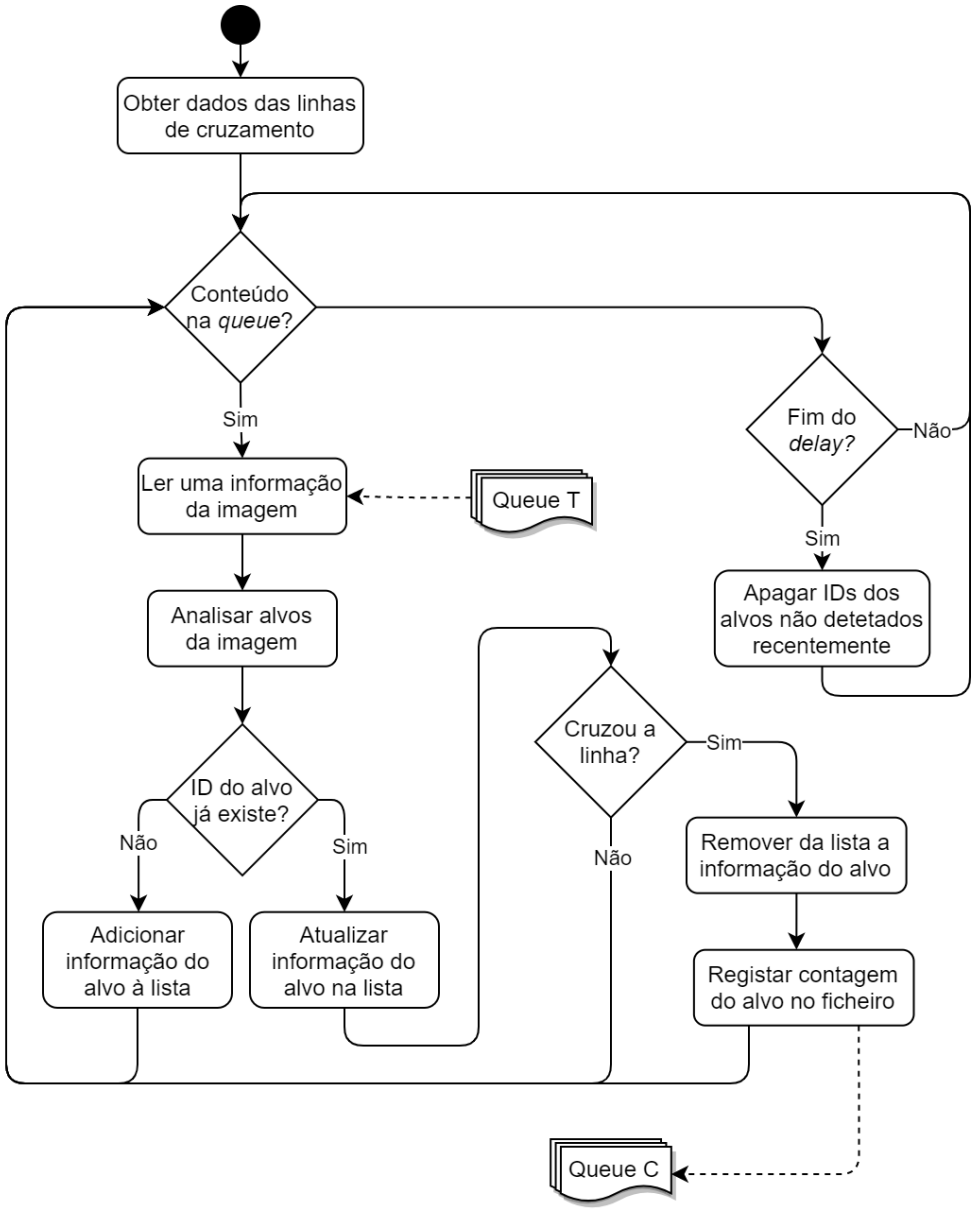
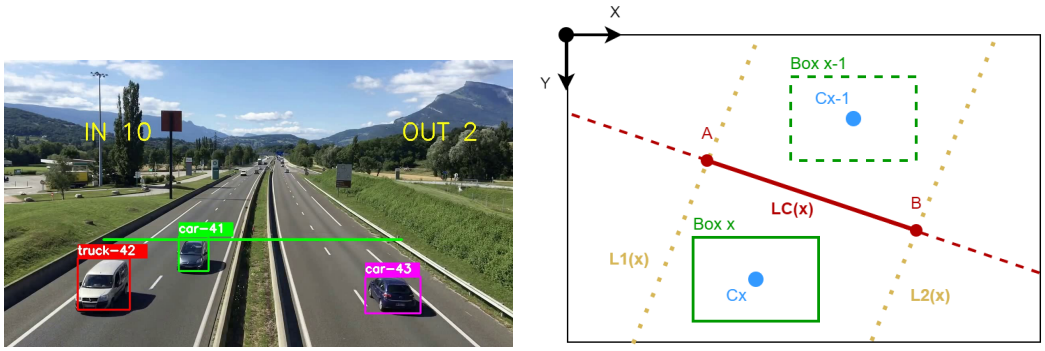


Figura 4.19: Diagrama de funcionamento do processo *Counter*.

zona de interesse compreendida entre as linhas $L1(x)$ e $L2(x)$ e os centróides (C_{x-1} e C_x) de 2 *bounding boxes* consecutivas do mesmo objeto. Nesta ilustração verifica-se que o objeto detetado em 2 instantes cruzou a linha de contagem, visto que a Box_{x-1} está acima da linha e a Box_x abaixo. O Sistema de Equações 4.3 permite determinar a equação da linha de contagem $LC(x)$, tendo como referência os pontos A e B. O Sistema de Equações 4.4 permite determinar as equações das linhas delimitadoras $L1(x)$ e $L2(x)$, tendo por base os mesmos pontos e sabendo que são perpendiculares à linha de contagem.



(a) Exemplo de aplicação do método [148] (b) Ilustração do método

Figura 4.20: Método de contagem de objetos.

$$\begin{cases} LC(x) = m_{LC} \cdot x + b_{LC} \\ m_{LC} = \frac{A(y)-B(y)}{A(x)-B(x)} \\ b_{LC} = A(y) - m_{LC} \cdot A(x) \end{cases} \tag{4.3}$$

$$\begin{cases} L1(x) = m_L \cdot x + b_{L1} \\ L2(x) = m_L \cdot x + b_{L2} \\ m_L = m_{LC}^{-1} \\ b_{L1} = A(y) - m_L \cdot A(x) \\ b_{L2} = B(y) - m_L \cdot B(x) \end{cases} \tag{4.4}$$

Descrito o processo de determinação das 3 linhas, o Sistema de Equações 4.5 permite determinar se o centróide C do objeto está compreendido entre as linhas delimitadoras, assim como a sua posição relativamente à linha de contagem LC. Verificada a transição da linha por centróides consecutivos do mesmo objeto, é feita a sua contagem.

$$\begin{cases} C_y - LC(x) > 0, & C \text{ acima da } LC \\ C_y - LC(x) < 0, & C \text{ abaixo da } LC \\ C_y - LC(x) = 0, & C \text{ sobreposto a } LC \end{cases} \tag{4.5}$$

4.2.4 Processo - Comunicação Série

A Figura 4.21 explica o funcionamento do processo de comunicação série. Inicialmente, é feita a configuração da porta série para comunicar através do protocolo UART com o módulo NB-IoT. De seguida, inicia-se um ciclo que verifica sistematicamente as seguintes condições:

- **Condição 1** - analisar se existe conteúdo na *Queue E* e, em caso afirmativo, identificar o tipo da mensagem. Esta poderá corresponder a um *reset* do módulo NB-IoT, ou a um comando com instruções para o módulo NB-IoT;
- **Condição 2** - verificar se o módulo NB-IoT reportou alguma mensagem para a porta série. Em caso afirmativo, a mensagem é lida e enviada para a *Queue R*.

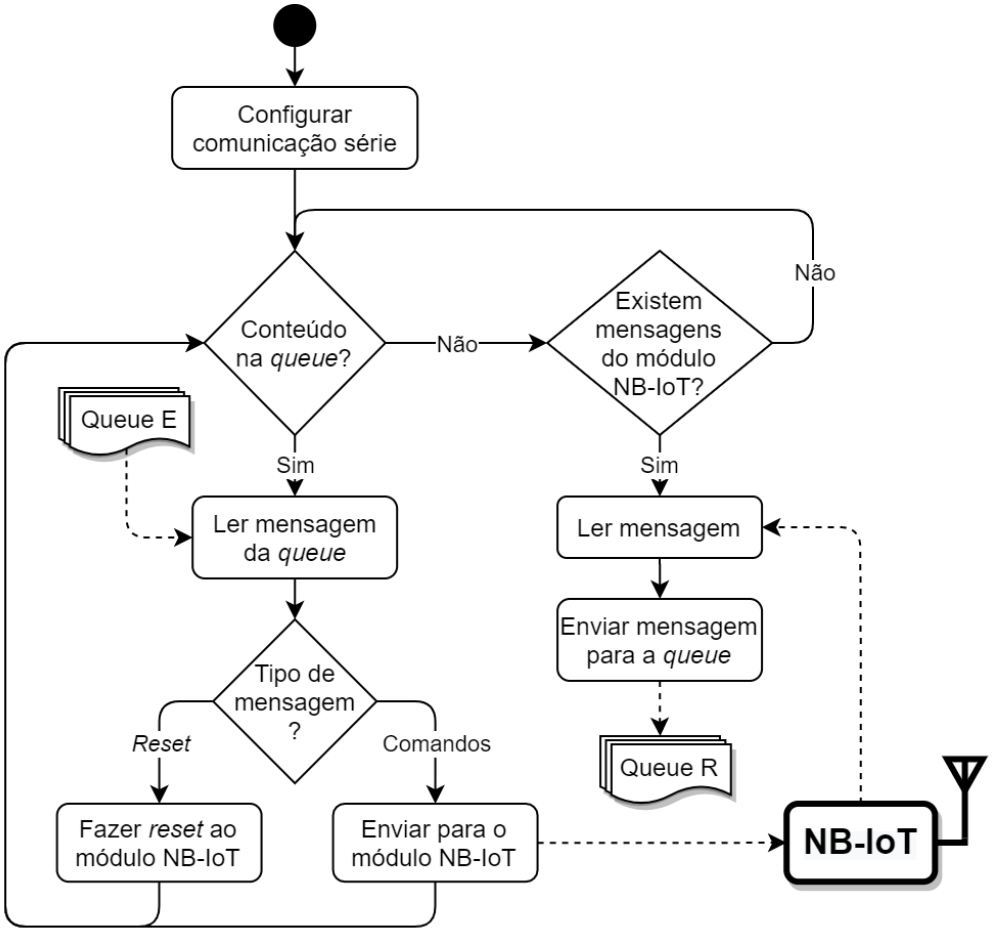


Figura 4.21: Diagrama de funcionamento do processo Comunicação Série.

4.2.5 Processo - Comunicação MQTT

Na Figura 4.22 está representado um diagrama que descreve o processo responsável pela configuração e comunicação do módulo NB-IoT, usando o protocolo de mensagens MQTT. Este processo inicia o módulo NB-IoT, seguindo-se a aquisição de informações relativas ao módulo e ao cartão SIM utilizados. Posteriormente, é feita a conexão do módulo à rede móvel do operador, seguida do estabelecimento da comunicação MQTT encriptada segundo o protocolo de segurança SSL/TLS para com um *broker* MQTT.

Estabelecida a comunicação com o *broker*, tem início um ciclo que, periodicamente, coloca na *Queue E* informação dos objetos contabilizados, tendo por base a análise dos dados provenientes da *Queue C*. Por outro lado, é analisada a *Queue R*, e, caso tenha informação, será lida e as respetivas ações executadas.

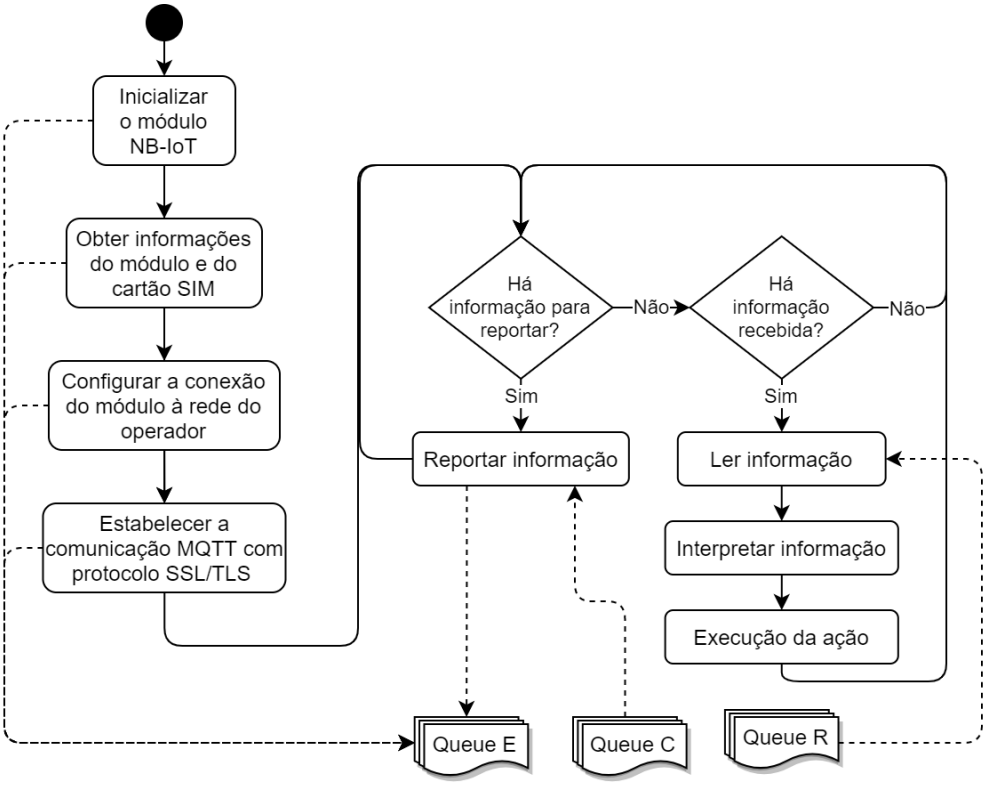


Figura 4.22: Diagrama de funcionamento do processo Comunicação MQTT.

4.2.6 Processo - *Logger* do Estado do *Hardware*

Na Figura 4.23 é apresentado o diagrama do processo de aquisição dos dados sobre o estado do HW. Inicialmente, é definida a periodicidade da leitura, seguida da entrada no ciclo. Este ciclo adquire, periodicamente, dados do sensor de temperatura, indicando a temperatura interior da caixa de isolamento, assim como de um sistema de monitorização que obtém o estado do CPU, GPU e RAM, velocidade da ventoinha e temperatura da *board*. Por fim, estes dados são guardados num ficheiro local.

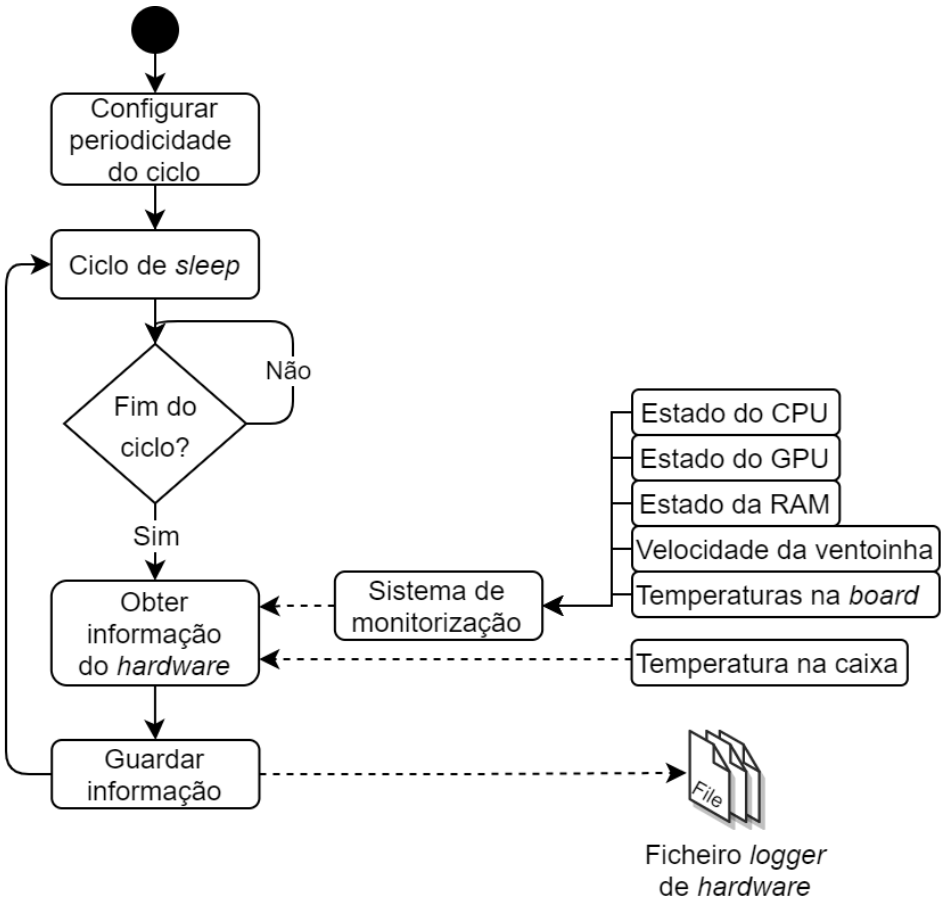


Figura 4.23: Diagrama de funcionamento do processo *Logger* do estado do HW.

4.2.7 Processo - *Logger* do Estado do *Software*

O *logger* do estado do SW é uma funcionalidade transversal aos restantes processos do sistema. Este é responsável por guardar num ficheiro local todas as informações relativas ao funcionamento do SW. Neste ficheiro serão reportadas mensagens de *debug*, informativas, de alerta e de erros.

Neste capítulo apresentou-se uma proposta de sistema de deteção e contagem, abordando-se a sua componente de *Hardware*, tendo em conta um conjunto de requisitos, como também se apresentou a estrutura e organização de toda a sua arquitetura de *Software*. No próximo capítulo é apresentada a implementação do sistema, fazendo uso de novas ferramentas, assim como a realização de testes de performance do sistema perante diversos cenários.

Capítulo 5

Implementação

Neste capítulo é apresentado o processo integral da implementação do sistema. Inicialmente, são abordados os vários cenários de validação e teste, assim como o primeiro protótipo do sistema. Segue-se a organização dos ficheiros de *Software*, o ambiente em que foi desenvolvido e ferramentas adicionais usadas durante o desenvolvimento. Posteriormente, são adquiridos dados relativos ao *Hardware*, assim como da implementação dos algoritmos de deteção, classificação, *tracking* e contagem de objetos. Por fim, é abordada a implementação da comunicação IoT.

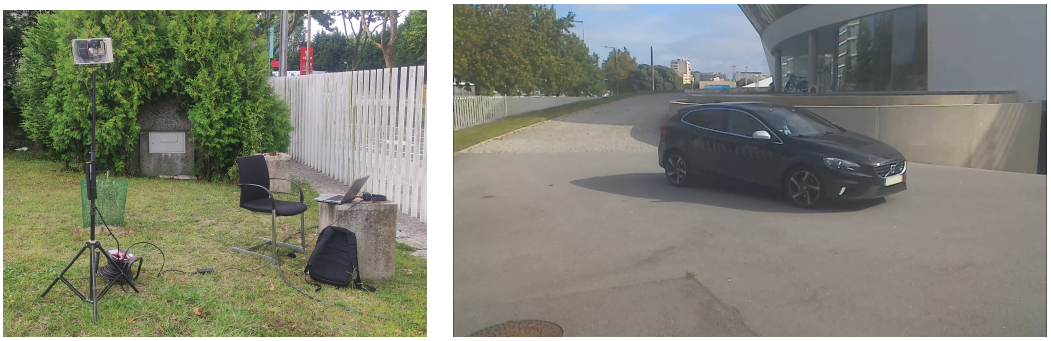
5.1 Cenários de Validação e Teste

Os cenários para validação e teste do produto dividem-se em três fases:

- **Performance do HW** - corresponde a testes de performance do HW para análise de parâmetros como: temperatura interna da caixa de isolamento; consumo energético e níveis de desempenho dos principais módulos. Tratam-se de testes de bancada efetuados numa sala com temperatura controlada na ordem dos 23,5 °C, assegurando que a variação da temperatura exterior não tenha influência sobre os resultados;
- **Aquisição de *datasets* para calibração** - consiste na aquisição de uma gravação de vídeo num cenário de aplicação semelhante, isto é, com circulação de veículos e de pessoas. O objetivo é, posteriormente, proceder à calibração e análise da eficiência do SW de contabilização de objetos em modo *offline*.

Estes *datasets* foram adquiridos em ambiente *outdoor*, no interior do recinto do CEiiA, conforme representado na Figura 5.1a, no período da manhã durante 2 horas de gravação. Para cumprimento do RGPD garantiu-se a não recolha de imagens exteriores ao recinto do CEiiA; divulgou-se com a devida antecedência o período e local da gravação; e, previamente, foi pedido o consentimento aos colaboradores da empresa para a gravação, não só dos próprios, como também dos seus veículos pessoais (Figura 5.1b). Adicionalmente, foi criado um documento relativo à AIPD, de acordo com o artigo 35.° do RGPD, que visa estabelecer e demonstrar conformidade com os requisitos do RGPD [149];

- **Funcionamento em tempo-real** - tem como objetivo testar o funcionamento do sistema num cenário real de contagem de objetos e respetiva partilha desta informação, de forma periódica, para o *broker* MQTT. O local de aplicação é junto à portaria do CEiiA (Figura 5.2), tendo o teste sido realizado por um período de 11 horas, ao longo do dia, incluindo os períodos de maior fluxo de pessoas e veículos na entrada do recinto (Figura 5.2b). A realização deste teste encontra-se também de acordo com o RGPD.



(a) Zona de instalação do sistema (b) Campo de visão do sistema

Figura 5.1: Cenário de aplicação para recolha de *datasets*.

(a) Zona de instalação do sistema (b) Campo de visão do sistema

Figura 5.2: Cenário de teste do funcionamento em tempo-real.

5.2 Protótipo

Na Figura 5.3 encontra-se representado o primeiro protótipo desenvolvido, resultante do projeto mecânico do sistema (Seção 4.1.4). Este protótipo possui um peso total de 0,990 kg, consideravelmente mais leve que os produtos congéneres existentes no mercado (Anexo A).

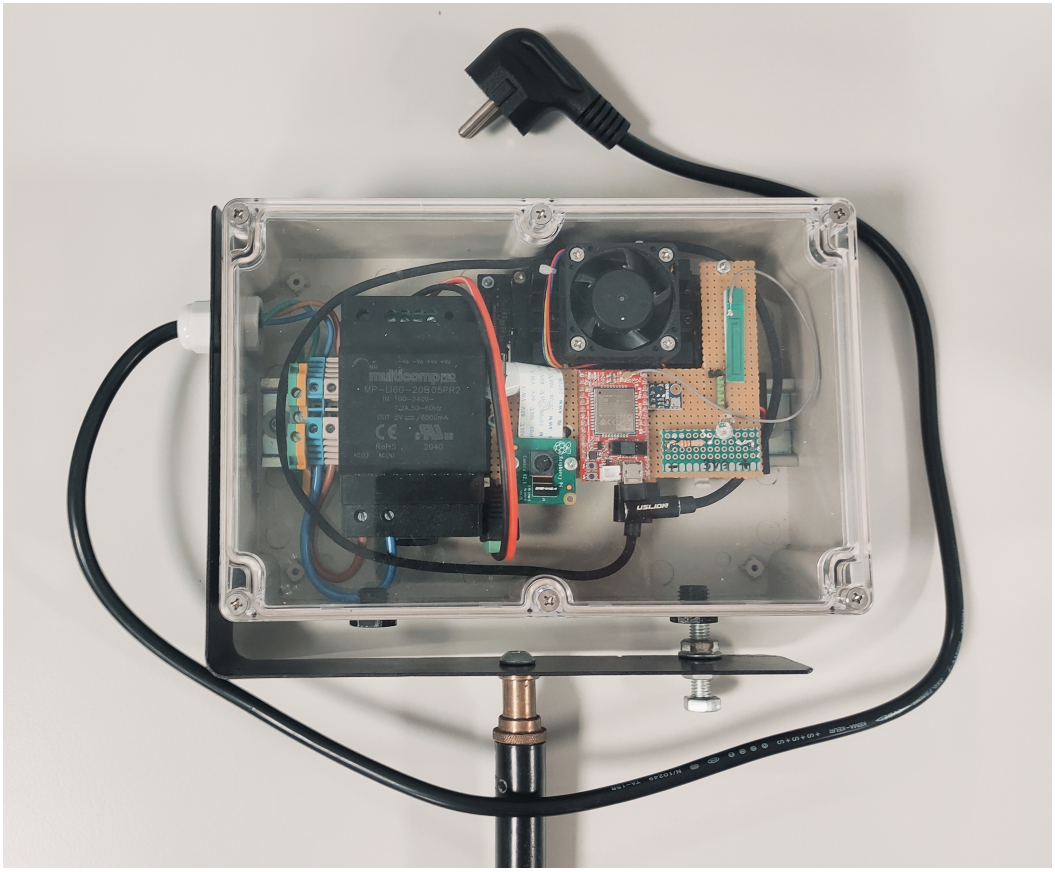


Figura 5.3: Representação do protótipo real do HW do sistema.

Este protótipo destina-se exclusivamente para efeitos de validação e testes tanto da componente mecânica como do funcionamento do SW desenvolvido. Contudo, é de salientar que neste protótipo alguns elementos não correspondem aos projetados:

- **Blocos terminais DIN** - utilizados blocos semelhantes que existiam em *stock* no CEiiA, garantindo as mesmas funcionalidades;
- **Sensor de temperatura** - utilizado o BMP180 existente em *stock* no CEiiA. Trata-se de um sensor de temperatura e pressão que possui o mesmo protocolo de comunicação e a mesma precisão dos valores de temperatura face ao sensor projetado (AM2320);
- **Cartão de memória** - utilizou-se um cartão de maior capacidade para possibilitar o armazenamento de *datasets* de vídeo para validação e teste do SW;

- **Cabo de alimentação monofásico** - permite a alimentação de todo o sistema, embora não esteja incluído no seu projeto;
- **Módulo Wi-Fi** - utilizou-se um adaptador USB *wireless* que permite o acesso remoto à Jetson Nano via *Secure Shell* (SSH) e *Virtual Network Computing* (VNC) através da rede Wi-Fi privada do CEiiA;
- **Mensagem MQTT** - adoptou-se o formato *JavaScript Object Notation* (JSON) para melhor interpretação visual dos dados.

5.3 Organização dos Ficheiros de *Software*

A Figura C.1 representa, em forma de árvore, a organização de todos os ficheiros que fazem parte da estrutura de SW do sistema. A sua organização encontra-se subdividida por 8 diretorias, conforme consta na Figura C.1.

5.4 Ambiente de Desenvolvimento

A Jetson Nano *Dev Kit* funciona com um SDK desenvolvido pela própria NVIDIA para os dispositivos da família Jetson, consistindo numa solução abrangente para o desenvolvimento de aplicações de IA. Este inclui: o *package* de *drivers* Linux mais recente para Jetsons com o SO Linux (Ubuntu 18.04); bibliotecas de aceleração CUDA-X e várias API para DL, visão computacional e aceleração da computação [150]. A versão instalada deste SDK foi a mais recente (JetPack 4.5.1), segundo o guia de instalação disponibilizado pela NVIDIA [151].

O SW desenvolvido para este sistema foi desenvolvido integralmente na linguagem de programação *Python* (versão 3.6), utilizando o editor de código fonte Visual Studio Code. Tanto as *queues* como os vários multiprocessos foram criados com recurso a uma biblioteca *multiprocessing* [152]. A Listagem 5.1 apresenta a criação tanto das *queues* como dos vários processos. Na fase de iniciação dos processos, cada um recebe como argumentos as respetivas *queues*, assim como a periodicidade (em minutos) da aquisição e envio de dados. Adicionalmente, a variável *lines* representa as coordenadas, em píxeis, dos pontos no plano da imagem da câmara que definirão as linhas de contagem.

```
import sys
from multiprocessing import Process, Queue

def main(args):

    # Create Queues
    q_T = Queue()
    q_C = Queue()
    q_R = Queue()
    q_E = Queue()

    # Counting line
    lines = {1: {'xl': 400, 'yl': 330, 'xr': 800, 'yr': 280,
             2: {'xl': 200, 'yl': 450, 'xr': 450, 'yr': 450}

    # Create multi-processes
    p1 = Process(target=t_hw_status, args=(0.5))
    p2 = Process(target=t_mqtt,      args=(q_R, q_E, 0.05))
    p3 = Process(target=t_serial,    args=(q_R, q_E))
    p4 = Process(target=t_tracker,   args=(q_T, q_C, lines))
    p5 = Process(target=t_counter,   args=(q_T, q_C, lines))

    # Start multi-processes
    p1.start() # Hardware Logger
    p2.start() # MQTT Communication
    p3.start() # Serial Communication
    p4.start() # Tracker
    p5.start() # Counter

# Start code
if __name__ == '__main__':
    sys.exit(main(sys.argv))
```

Listagem 5.1: Código do *script* principal do SW do sistema.

Para a configuração do módulo de ventilação, existe um *script* em *Python* que, automaticamente, faz o controlo do módulo segundo sinais PWM [153]. É necessário definir as temperaturas mínimas e máximas, em graus *Celsius*, para o qual o PWM deve ser nulo e máximo, respetivamente, no respetivo ficheiro JSON de configuração. Neste intervalo de temperaturas a relação entre o valor de PWM e o valor da temperatura é linear. Terminada a configuração, este *script* é executado sempre que o SO é iniciado.

Por fim, para a configuração do LED de estado, recorreu-se a uma biblioteca de GPIO da família Jetson, denominada por Jetson.GPIO [154]. Esta biblioteca permite o controlo dos pinos GPIO da Jetson, tendo a configuração deste LED sido baseada num código exemplo, em *Python*, para piscar o LED [155].

5.5 Ferramentas Adicionais

Nesta seção são descritas as ferramentas computacionais utilizadas no desenvolvimento do sistema.

Para a implementação e testes deste sistema, estabeleceu-se o acesso remoto à Jetson Nano com recurso a um computador ligado à mesma rede Wi-Fi, segundo o protocolo de rede criptográfico SSH [156], sendo a sua utilização recomendada para redes não protegidas.

Em certos casos, recorreu-se ao VNC, visto que, comparativamente ao SSH, apresenta uma interface gráfica para a visualização e interação com a Jetson Nano. Para esta comunicação, instalou-se o Vino [157], servidor VNC, na Jetson Nano e o VNC *viewer* da RealVNC no computador. Na Figura 5.4 está representado o acesso remoto à Jetson Nano de um computador com o VNC *viewer*.

Figura 5.4: Acesso remoto à Jetson Nano através de um computador com VNC *viewer*.

A ferramenta FFMPEG é uma *framework* de multimédia para gravação, conversão e *stream* de áudio e vídeo [158]. Nesta implementação, esta ferramenta é utilizada para conversão de vídeos de e para diferentes formatos, assegurando um determinado *frame rate*. Na Listagem 5.2 é apresentado um exemplo de conversão de um ficheiro de vídeo com extensão *.h264* para *.mp4* com um *frame rate* de 25 FPS.

```
1 $ ffmpeg -framerate 25 -i input.h264 -c copy output.mp4
```

Listagem 5.2: Comando para conversão de um vídeo *.h264* para *.mp4* com um *frame rate* de 25 FPS.

Ao longo do processo de implementação foram desenvolvidas ferramentas auxiliares que consistem em *scripts* de *Python*, localizadas na diretoria *tools/* do SW (Anexo C). Segue uma breve descrição de cada uma das ferramentas:

- **"hw_status.py"** - desempenha a mesma função do processo *logger* de estado do HW, embora nesta situação possa ser executado separadamente;
- **"live_video.py"** - realiza um *streaming* de vídeo, recorrendo ao GStreamer [159], que é uma *framework open source* existente no JetPack SDK, com o objetivo de desenvolver aplicações que lidam com *streams* multimédia. Este *script* baseou-se num repositório com exemplos de aplicações para câmaras CSI [160]. A Listagem 5.3 apresenta o comando executado neste *script*;

```
1 $ gst-launch-1.0 nvarguscamerasrc sensor_id=0 ! 'video/x-
    raw(memory:NVMM),width=1280, height=720, framerate
    =25/1, format=NV12' ! nvvidconv flip-method=2 ! 'video/
    x-raw,width=960, height=720' ! nvvidconv !
    nvegltransform ! nveglglessink -e
```

Listagem 5.3: Comando para *stream* de vídeo à resolução 1280×720 a 25 FPS.

- **"record_video.py"** - ferramenta que também recorre ao GStreamer para realizar uma gravação de N minutos com uma determinada resolução suportada pela câmara. A Listagem 5.4 apresenta o comando executado neste *script*;

```
1 $ gst-launch-1.0 nvarguscamerasrc num-buffers=<time> ! '
    video/x-raw(memory:NVMM),width=1280, height=720,
    framerate=25/1, format=NV12' ! nvvidconv flip-method=2
    ! omxh264enc ! h264parse ! 'video/x-h264,stream-format=
    byte-stream,alignment=au' ! filesink location=<
    directory> -e
```

Listagem 5.4: Comando para gravação de vídeo à resolução 1280×720 a 25 FPS.

- **"single_shot.py"** - faz a captura instantânea de uma imagem recorrendo ao GStreamer. A Listagem 5.5 apresenta o comando executado neste *script*.

```
1 $ nvgstcapture-1.0 --automate --capture-auto --image-res=3
     --orientation 0
```

Listagem 5.5: Comando para captura de uma imagem à resolução 1280×720.

No processo de desenvolvimento e implementação da comunicação IoT, segundo o protocolo de comunicação MQTT, utilizou-se o MQTT *Explorer* [161]. Este consiste num cliente MQTT que apresenta de forma estruturada os tópicos MQTT (Figura 5.5). Desta forma, é facilitada a confirmação do correto *flow* da comunicação com o cliente MQTT do sistema. Nesta implementação, esta ferramenta desempenha o papel do servidor remoto simulado.

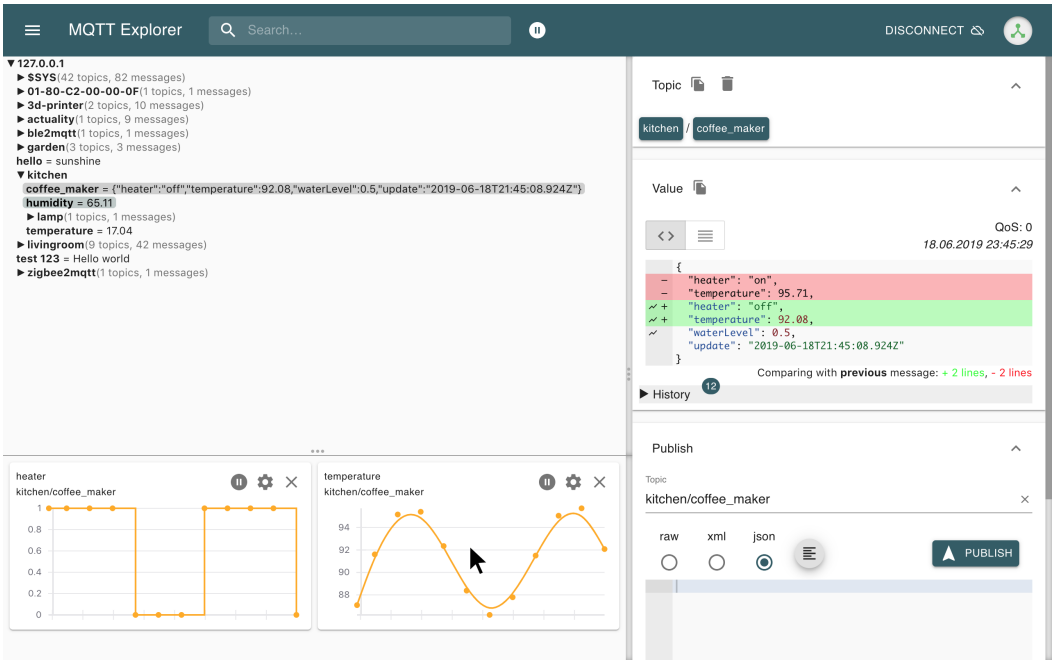


Figura 5.5: Interface visual do MQTT *Explorer* [161].

5.6 Aquisição de Dados de *Hardware*

No processo de aquisição de dados do estado do HW do sistema, nomeadamente de parâmetros relativos à Jetson Nano, utiliza-se o *package*, denominado Jetson Stats [162], que monitoriza e controla os dispositivos do ecossistema NVIDIA Jetson. Na Figura 5.6 é representada, através da janela terminal do SO, a interface visual que indica, em tempo-real, as temperaturas de módulos da *board*, assim como os níveis de utilização dos *cores* do CPU, GPU e RAM. Adicionalmente, é possível controlar certos parâmetros através desta interface.

Por forma a adquirir estes dados sem recurso à interface visual, criou-se um processo baseado num *logger* em *Python* que recolhe periodicamente os vários parâmetros numa trama de mensagens [163]. As modificações feitas permitem alterar dinamicamente a periodicidade de aquisição de dados, assim como a introdução do valor de temperatura recolhido pelo BMP180. O BMP180 foi configurado utilizando uma biblioteca da Adafruit [164].

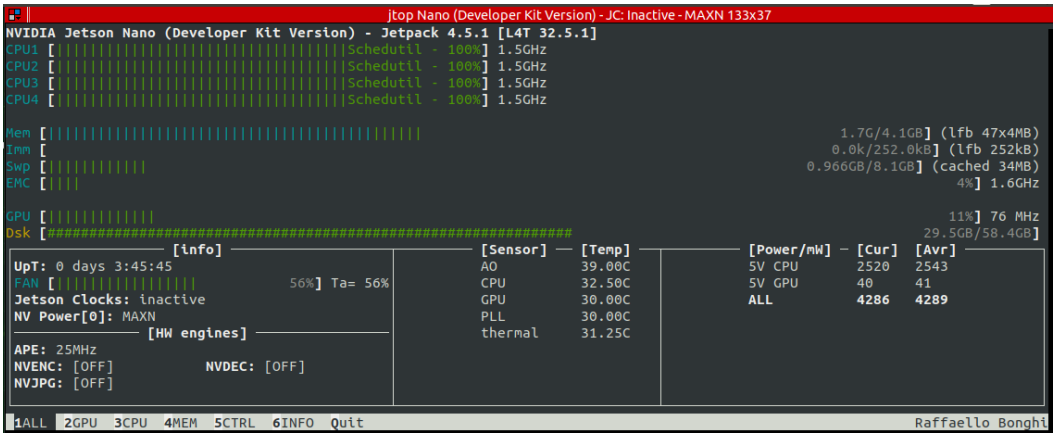


Figura 5.6: Interface visual do *package* Jetson Stats na Jetson Nano.

5.7 Deteção, Classificação e *Tracking*

Para a implementação de um SW baseado em DL para a deteção, classificação e *tracking* de objetos recorreu-se a uma *toolkit* de *streaming* denominada DeepStream [165]. A DeepStream adquire dados de *streaming* provenientes de câmaras USB/CSI ou de ficheiros de vídeo e, posteriormente, utiliza IA e visão computacional caracterizar o meio ambiente. Este SDK suporta o desenvolvimento de aplicações tanto em C/C++ como em *Python*. No caso do desenvolvimento em *Python*, recorre à interface *Python Bindings*. A DeepStream consiste em vários *plugins* de aceleração do HW, nomeadamente para GPU e CPU. Adicionalmente, a DeepStream baseia-se na *framework* GStreamer e em várias bibliotecas da NVIDIA. É o caso de bibliotecas de multimédia, *Compute Unified Device Architecture* (CUDA) e TensorRT, que aceleram a inferência de IA no GPU da NVIDIA. Os *plugins* da DeepStream facilitam o desenvolvimento de *pipelines* de aplicações de vídeo ao abstrair os detalhes da utilização direta destas bibliotecas.

5.7.1 DeepStream Python

O *Python Bindings* permite o desenvolvimento de aplicações de IA com recurso à linguagem de programação *Python* [166]. Desta forma, as *pipelines* da DeepStream podem ser criadas utilizando o Gst-Python, consistindo na *framework* GStreamer.

A aplicação DeepStream *Python* acede a tipos de dados nativos da linguagem de programação C/C++, recorrendo aos *packages PyBindings* e *NumPy* para estabelecer a interface da linguagem *Python* com C/C++, usufruindo do poder computacional das linguagens C/C++. Na Figura 5.7 é apresentada esta interface entre a aplicação e o modelo de IA.

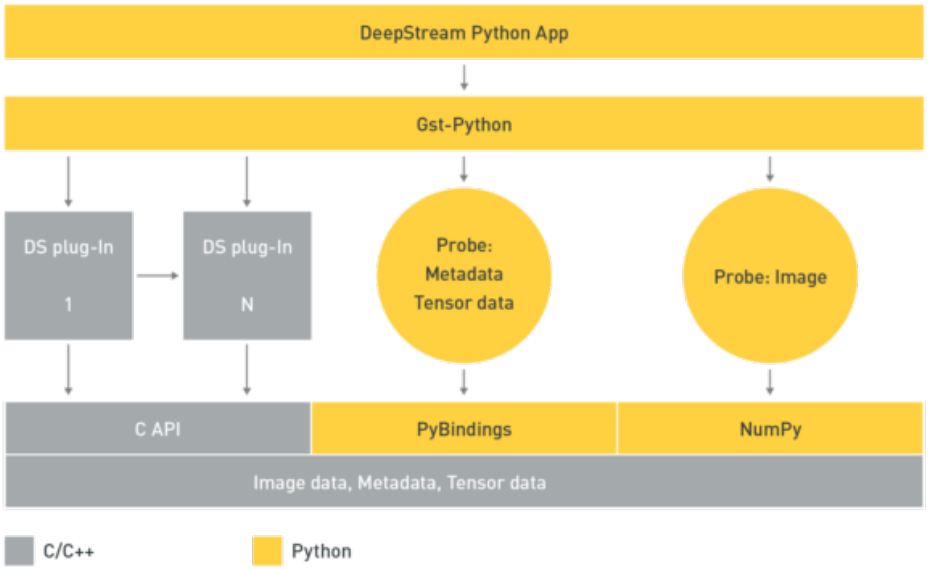


Figura 5.7: Interligação entre a aplicação DeepStream *Python* e a *pipeline* do modelo de IA na linguagem C/C++ [165].

5.7.2 Aplicação do DeepStream

Para demonstração da utilidade da DeepStream, são disponibilizados pela NVIDIA alguns exemplos de aplicações que utilizam os *Python Bindings* [167]. Um destes exemplos, denominado "deepstream-test2", permite através de um ficheiro de vídeo extrair informações sobre 4 classes de objetos: pessoas; carros; veículos de duas rodas e sinais de trânsito. A sua *pipeline* começa por criar uma instância de "Gst-nvinfer"[168], referida como *Primary GPU Inference Engine* (PGIE), que processa cada *frame* de vídeo utilizando o TensorRT, otimizador da NVIDIA para redes neuronais pré-treinadas. Esta instância deteta, em cada *frame*, múltiplas classes de objetos tendo por base a utilização de um modelo de deteção com o *backbone* ResNet-10. A estrutura do *backbone* é composta por 10 camadas de convolução e representada na Figura D.1 com recurso ao visualizador de redes neuronais Netron. Trata-se de um modelo pré-treinado na *framework* de DL Caffe com recurso a *datasets* de múltiplos objetos proprietários da NVIDIA.

Segue-se a criação de uma instância *nvtracker* para o *tracking* dos objetos detetados pelo PGIE. Este possibilita a utilização de um de três algoritmos de *tracking* disponíveis. A Tabela 5.1 descreve estes algoritmos, constatando-se diferenças ao nível da carga computacional, assim como prós e contras. A utilização do *traker* IoU foi descartada por não se enquadrar no *tracking* de objetos com movimentos rápidos. Desta forma, os *trakers Kanade–Lucas–Tomasi* (KLT) e NvDCF são testados em cenários reais para identificar o que melhor se comporta.

Posterior à inferência de *tracking*, seguem-se na *pipeline* três diferentes instâncias

Tabela 5.1: Tabela comparativa entre os diferentes algoritmos de *tracking* existentes na DeepStream SDK [91].

Propriedades	IoU	KLT	NvDCF
Utilização do GPU	Nenhum	Nenhum	Médio
Utilização do CPU	Muito baixo	Alto	Baixo
Prós	Não impõe requisitos computacionais	Bom para cenários simples	Troca de ID menos frequente em oclusão parcial
Contras	Frequentes falhas e trocas na atribuição do ID ao objeto	Elevada utilização do CPU e sensível a objetos com baixa textura	Lentidão devida à complexidade computacional
Casos de uso	Para objetos dispersos e de diferentes dimensões	Para um *background* simples	Múltiplos objetos com oclusão parcial e cenários complexos

"Gst-nvinfer"denominadas *Secondary GPU Inference Engine* (SGIE). Estas instâncias (SGIE1, SGIE2 e SGIE3) têm a função de classificar os veículos detetados pelo PGIE ao nível de cor, marca e tipo, respetivamente. Estas instâncias utilizam o modelo de classificação com o *backbone* ResNet-18 com uma estrutura do *backbone* composta por 18 camadas de convolução (Figura D.2).

Na Figura 5.8 está representada a *pipeline* do exemplo na DeepStream, sendo constituída por 8 grandes etapas. Os blocos representados a verde indicam *plugins* do GStreamer da *pipeline* original do exemplo que podem ser reutilizadas no sistema a desenvolver, sendo necessário fazer a captura de imagens, em tempo-real, provenientes de uma câmara CSI. Para tal, são utilizados o *plugin nvarguscamerasrc* para receção de imagens da câmara; o *nvvideoconvert* para conversão do formato de cores; o *capsfilter* para limitação do formato dos dados da imagem e, por fim, o *fakesink* que consiste numa API de renderização de imagem de vídeo. O funcionamento da câmara ficou configurado para uma resolução 1280×720p a 25 FPS. A especificação deste *frame rate* foi baseada num estudo que determina a influência do *frame rate* no *tracking* de objetos [169]. Este estudo indica que valores acima de 20 FPS não apresentam um significativo aumento da precisão do *tracking* de pedestres e ciclistas. Por outro lado, cada ciclo deste exemplo da DeepStream demora cerca de 35 milissegundos, isto é, 28 FPS. Desta forma, não se justifica um *frame rate* da câmara superior a este valor. Sendo assim, a *pipeline* resultante é capaz de receber imagens provenientes de ficheiros de vídeo com compressão de vídeo H.264 (fase de testes) ou *streaming* de uma câmara CSI ou USB (caso real de aplicação).

Para a configuração destes blocos (PGIE, SGIE e *tracker*), existe um ficheiro de

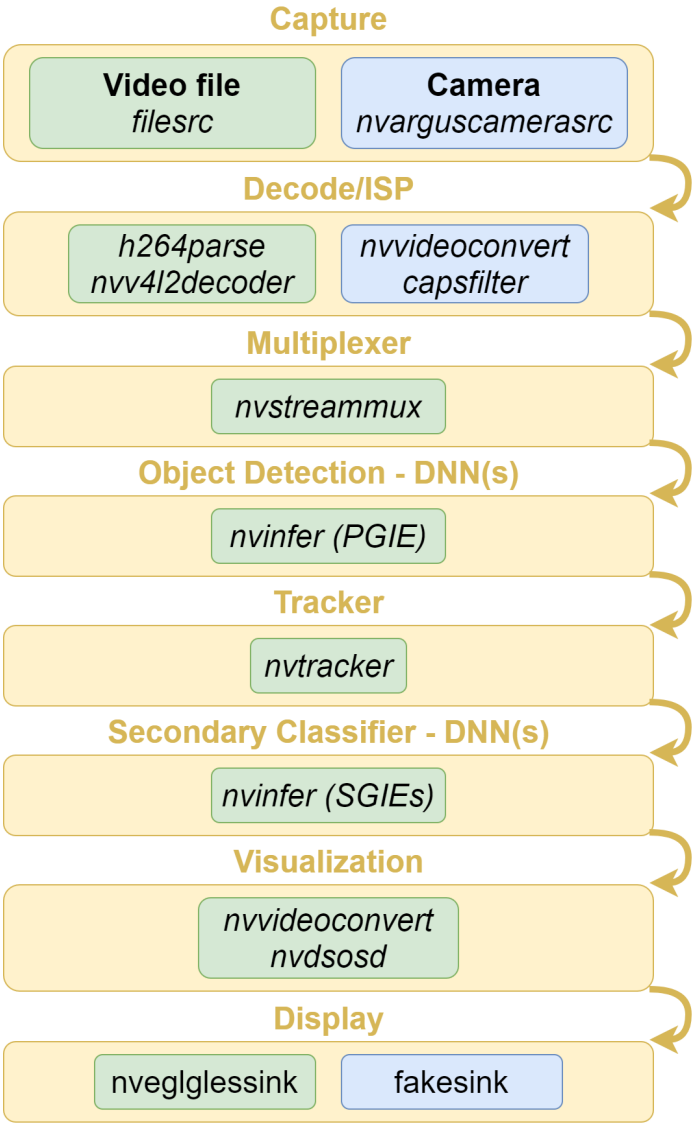


Figura 5.8: *Pipeline* do modelo exemplo da DeepStream (blocos amarelos) com representação dos seus módulos pré-definidos (blocos verdes) assim como os módulos adicionados (blocos azuis) para satisfazer as necessidades do sistema.

configuração correspondente a cada tipo de inferência, contendo parâmetros configuráveis como o tamanho mínimo da *bounding box* para aceitação; o nível de *threshold* de confiança e os pesos para associação de dados de *tracking* [170].

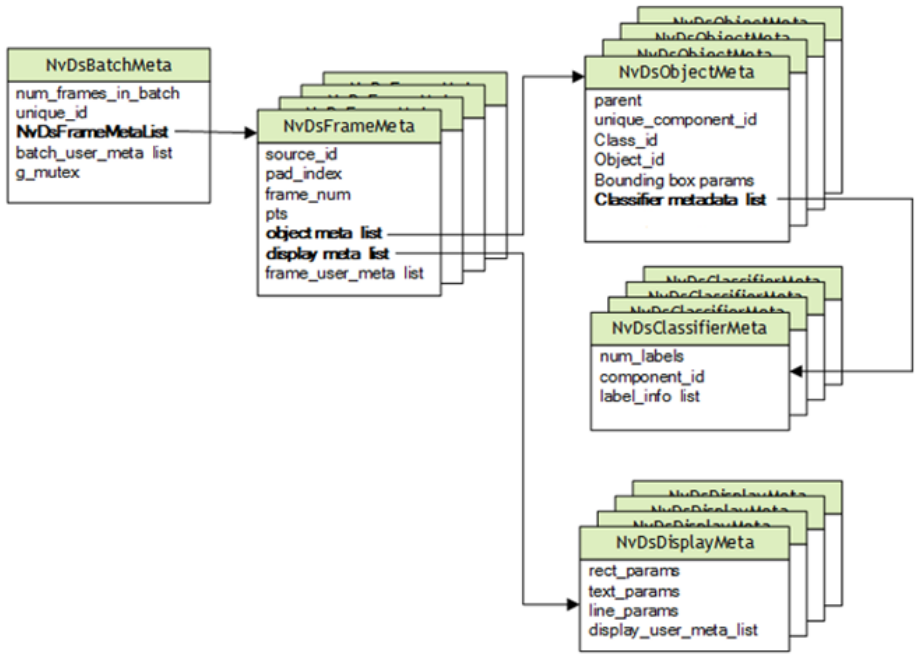


Figura 5.9: Estrutura dos metadados utilizados na DeepStream SDK [171].

Ao longo desta *pipeline*, existem *buffers*, denominados *Gst Buffer*, responsáveis pela transferência interna de dados no GStreamer. A cada um destes estão associados metadados em que a DeepStream SDK anexa ao seu objeto de metadados, *NvDsBatchMeta* [171]. Na Figura 5.9 é apresentada a estrutura *standard* dos metadados, sendo *NvDsBatchMeta* a estrutura base composta por metadados relativos a cada *frame* de vídeo (*NvDsFrameMeta*); objeto identificado (*NvDsObjectMeta*); classificação do objeto (*NvDsClassifierMeta*) e *labels* nas *frames* de vídeo (*NvDsDisplayMeta*).

Visto que a implementação será feita numa Jetson Nano, foi necessário fazer uma recompilação do código base do formato *default* com o nível de precisão INT8 para o FP16, visto que a Jetson Nano apenas suporta os níveis de precisão FP16 e FP32 [172]. Optou-se pelo FP16 visto que garante melhor performance, não se justificando a utilização de FP32 para o nível de complexidade desta aplicação [173].

5.8 Contagem

O processo de contagem dos objetos começa por definir o local de instalação do sistema, seguido da captura instantânea de uma imagem correspondente ao campo

de visão da câmara. Posteriormente, são definidos os limites da linha de contagem, sendo definidas na variável *lines* do *script* principal (Listagem 5.1).

O processo de contagem recebe, de cada *frame* analisada pela *pipeline* da DeepStream, uma estrutura de dados com a informação retirada da *frame*. Na Listagem 5.6 é apresentado um exemplo de estrutura de dados de uma *frame* que contém informação sobre dois objetos detetados.

```
self.frame_msg = {
    "frame_id": 200, # Numero da frame
    "detections": 2, # Numero de objetos detetados
    "timestamp": 1627431000, #  Formato Epoch time
    "targets": # Lista de objetos detetados
    {
        1: # Primeiro objeto detetado
        {
            "id": 32, # Tracking ID
            "label": "car", # Tipo de objeto
            "cross": False, # Cruzamento da linha
            "timestamp": 1627431000 # Instante da detecao
            "posic": # Coordenadas do centroide
            {
                "x": 521, # em pixeis
                "y": 260, # em pixeis
            }
        },

        2: # Segundo objeto detetado
        {
            "id": 49,
            "label": "person",
            "cross": False,
            "timestamp": 1627431000
            "posic":
            {
                "x": 294,
                "y": 400,
            }
        },
    }
}
```

Listagem 5.6: Estrutura de dados exemplo recebida no processo de contagem.

À medida que são recolhidas mensagens resultantes do processamento das *frames*, é atualizada uma lista dinâmica *id_list* que mantém, por um período de tempo predefinido, a informação dos objetos *tracked*. Ao longo do tempo, as coordenadas

do centróide dos objetos *tracked* é atualizada nesta lista e, caso algum objeto deixe de ser *tracked* é automaticamente descartado da lista ao final de um tempo predefinido. A Listagem 5.7 apresenta um exemplo da lista, contendo informações de dois objetos *tracked*, sendo que o elemento "1:" já não se encontra listado por ter sido perdido o seu *tracking*.

```
self.id_list = {

        2: # Elemento 1 da lista
        {
            "id": 32, # Tracking ID
            "label": "car", # Tipo de objeto
            "cross": "Above", # Cruzamento da linha
            "timestamp": 1627431000 # Instante da detecao
            "posic": # Coordenadas do centroide
            {
                "x": 521, # em pixeis
                "y": 260, # em pixeis
            }
        },

        3: # Elemento 2 da lista
        {
            "id": 49,
            "label": "person",
            "cross": "Below",
            "timestamp": 1627431000
            "posic":
            {
                "x": 294,
                "y": 400,
            }
        },
    }
}
```

Listagem 5.7: Exemplo da lista dinâmica de objetos *tracked*.

A cada atualização, é verificado se a posição do centróide se encontra acima/abaixo da linha de contagem e, caso se verifique que cruzou a linha, é enviada para a *queue* de contagem a informação do objeto. A Tabela 5.2 resume a informação enviada: o momento do cruzamento (*timestamp*); o ID da linha de contagem; o ID do objeto e, por fim, o tipo de objeto. Após o envio desta informação, o objeto é eliminado da lista.

Tabela 5.2: Exemplo de uma trama de mensagens resultante da contagem de um objeto.

Timestamp	**ID da linha**	**ID do objeto**	**Tipo de objeto**
1627431000 ,	1 ,	49 ,	"person"

5.9 Comunicação IoT

Quanto à comunicação IoT, a Jetson Nano comunica com o *kit* de desenvolvimento NB-IoT-DevKit recorrendo a comandos *ATtention* (AT) para configuração do módulo, de acordo com o protocolo de comunicação UART. Adicionalmente, estes comandos são usados para ativar a conexão do módulo à rede móvel. Os comandos AT consistem em instruções utilizadas para controlar *modems*. Dado que se trata de um *modem* de GSM/GPRS, os comandos AT utilizados são especificados pela Quectel [174].

A Figura 5.10 representa a comunicação entre a Jetson Nano e o módulo BC66 do *kit* de desenvolvimento NB-IoT, sendo o SW desenvolvido correspondente a uma máquina de estados finita. A máquina de estados desenvolvida pode dividir-se em 5 grandes fases:

- **Obtenção de informações do módulo NB-IoT e do cartão SIM** - nesta fase é definido o *baud rate* para a comunicação UART e desativados os modos *echo* e *sleep*, *Extended Discontinuous Reception* (eDRX) e *Power Saving Mode* (PSM) (configurações ativas por *default*). Segue-se a definição do registo *Unsolicited Result Code* (URC), assim como a aquisição de 3 dados do módulo NB-IoT e 2 do cartão SIM: modelo; versão de SW; *International Mobile Equipment Identity* (IMEI); estado de ligação; *International Mobile Subscriber Identity* (IMSI) e *Integrated Circuit Card Identifier* (ICCID), respetivamente;

- **Configuração da conexão do módulo NB-IoT** - numa primeira fase, analisa se as faixas de frequência existentes são as disponíveis na Europa para NB-IoT: B3 (1800 MHz), B8 (900 MHz) e B20 (800 MHz)) [175]. Segue-se a mesma verificação face ao *Access Point Name* (APN) e ao operador da rede móvel. Caso alguma verificação falhe, o respetivo parâmetro é atualizado com uma configuração pré-definida. Posteriormente, conecta-se o módulo à rede móvel da operadora;

- **Estabelecimento da comunicação MQTT com o protocolo SSL/TLS** - após a ligação à rede móvel, aguarda-se pela receção de um endereço IP dinâmico atribuído pelo operador, seguindo-se a sincronização da data e hora do módulo com recurso a um servidor *Network Time Protocol* (NTP) em Portugal. Segue-se a configuração da comunicação MQTT, assim como do protocolo de

segurança SSL, segundo uma encriptação assimétrica. Terminadas as configurações, estabelece-se a conexão MQTT, seguida da autenticação, encriptação e ligação ao respetivo *broker* MQTT. Estabelecida a comunicação, o módulo subscreve uma lista de tópicos pré-definidos e envia uma mensagem resumo sobre o estado do sistema. Por fim, arranca um ciclo de espera que, periodicamente, envia mensagens com dados da contagem de pessoas, carros/carrinhas e bicicletas/motociclos, respetivamente. No mesmo ciclo, é verificada a receção de mensagens dos tópicos subscritos e são executadas as respetivas ações;

- **Modos de *reset*** - quando ocorrem 2 falhas consecutivas num qualquer estado da máquina de estados, é realizado um *hard reset*, que consiste em reiniciar o módulo NB-IoT através de um *reset* por HW pelo pino *Request to Send* (RTS) da comunicação UART. O *soft reset* utiliza-se para reiniciar o *kit* NB-IoT por SW, isto é, com recurso ao envio de um comando AT, sempre que se pretende reiniciar o módulo após uma reconfiguração;

- **Desconexão da comunicação MQTT** - tem como finalidade terminar a comunicação com o *broker* MQTT de forma temporária ou definitiva, de acordo com um parâmetro enviado pelo servidor remoto. Nesta fase é feita a dessubscrição de todos os tópicos, seguida da desconexão do *broker* MQTT e, por fim, desliga-se o módulo pelo período de tempo estipulado.

Nesta máquina de estados, embora não representado, todos os estados enviam comandos AT para a *queue* de envio (*Queue E*) destinados ao módulo NB-IoT, assim como também recebem as respostas deste módulo através da *queue* de receção (*Queue R*).

5.9.1 Tópicos MQTT

A comunicação segundo o protocolo de mensagens MQTT é organizada tendo em conta uma lista de tópicos. Para organizar a comunicação entre o sistema e o operador, definiram-se 4 tipos de tópicos representados na Figura 5.11:

- **DT** - significa *Data Trigger* e tem a função de enviar um comando que solicita um resposta do tópico do tipo *Data Report* (DR);

- **DR** - significa *Data Report* e tem a função de responder a comandos do tópico do tipo *Data Trigger* (DT) ou reportar os dados para o *broker* com uma determinada periodicidade;

- **FE** - significa *Function Execute* e envia um comando que gera uma ação conhecida no sistema destinatário;

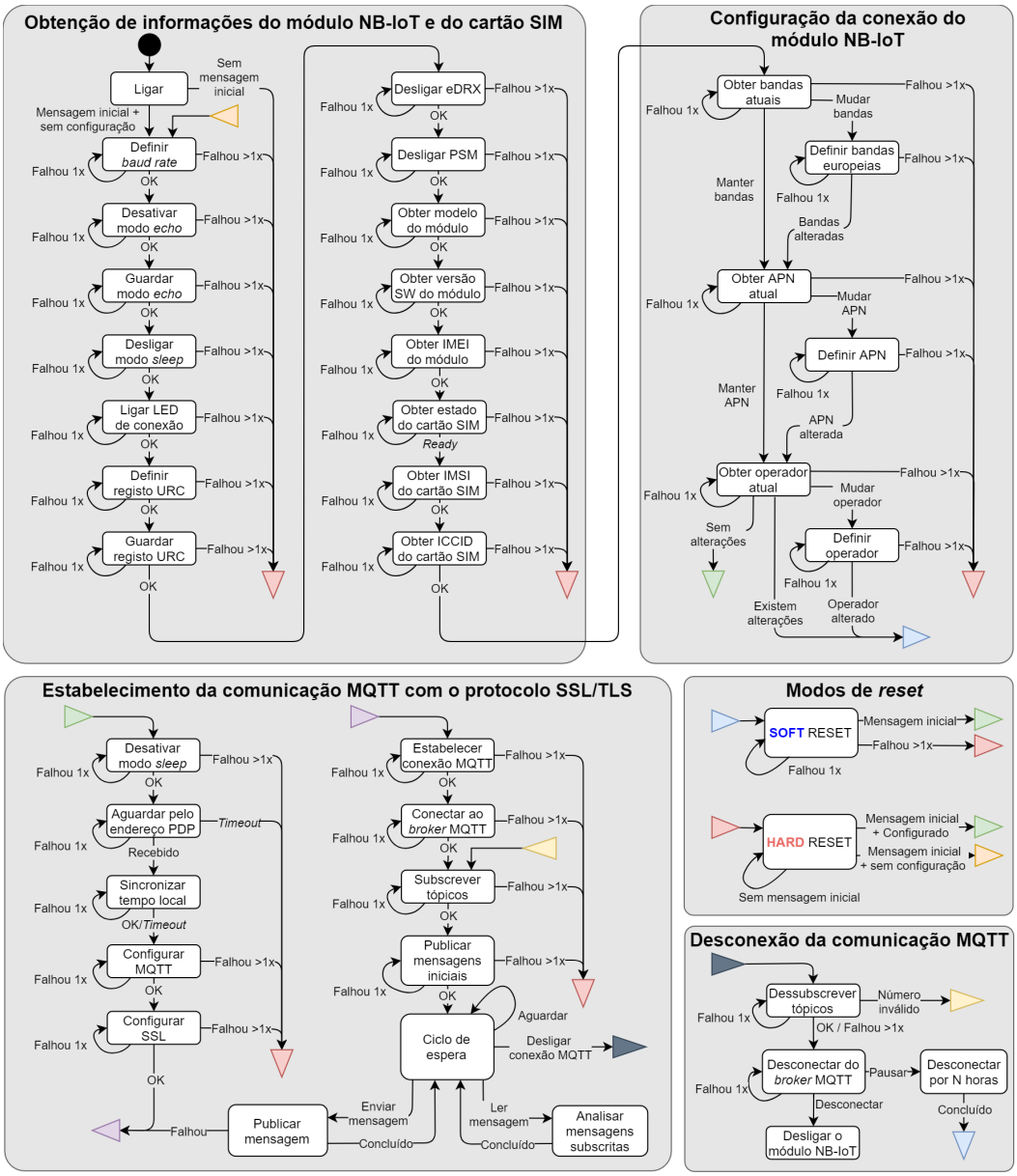


Figura 5.10: Máquina de estados da comunicação IoT.

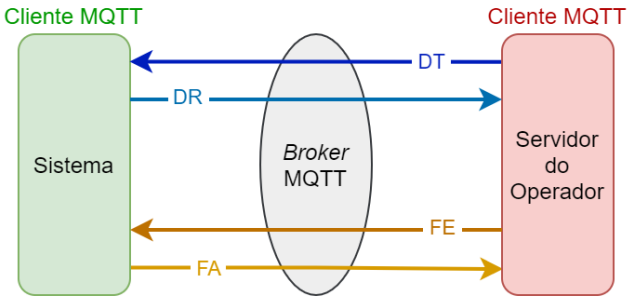


Figura 5.11: Estrutura dos tipos de tópicos MQTT entre o sistema e o operador.

- **FA** - significa *Function Acknowledge* (FA) e responde aos comandos do tópico *Function Execute* (FE), retornando o resultado (sucesso ou insucesso) da execução da ação solicitada.

Desta forma, a rede de tópicos criados seguem uma estrutura com o seguinte formato: *<tipo_dispositivo>/<SN>/<tipo_tópico>/<parâmetro>*, onde:

- **<tipo_dispositivo>** - indica a que tipo de dispositivo corresponde o sistema, neste caso, sendo denominado por "VisionBox", este campo será "vbox";
- **<SN>** - corresponde aos últimos 5 dígitos do *Serial Number* (SN) da Jetson Nano do sistema;
- **<tipo_tópico>** - refere o tipo de tópico: DT, DR, FE ou FA;
- **<parâmetro>** - indica, de forma sucinta, a finalidade do tópico ou o parâmetro a configurar.

Baseados nesta estrutura de tópicos, foram criados um conjunto de tópicos *standard* para o sistema:

- Do operador para o sistema:
  - **vbox/07760/dt/info** - faz o pedido do estado mais recente de certos parâmetros do sistema;
  - **vbox/07760/dt/count** - faz o pedido da contagem de objetos nos últimos N minutos, onde N é um número enviado neste tópico;
  - **vbox/07760/fe/period** - faz a alteração da periodicidade de reporte dos dados de contagem para o servidor, sendo enviado um número N em minutos;
  - **vbox/07760/fe/diconnect** - solicita ao sistema uma desconexão temporária ao *broker* MQTT, sendo enviado o número N de minutos de desconexão ou um tipo de conteúdo diferente para solicitar a desconexão definitiva ao *broker*.
- Do sistema para o servidor:
  - **vbox/07760/dr/info** - reporta o estado mais recente de certos parâmetros do sistema quando se liga ao *broker* MQTT através de uma mensagem do tipo *retained*;
  - **vbox/07760/dr/count** - reporta periodicamente a contagem dos objetos, contendo a data e hora de aquisição; período de contagem e o número de pessoas, carros/carrinhas e bicicletas/motociclos contabilizadas neste período. Adicionalmente, envia o valor da temperatura interna do sistema;

- **vbox/07760/dr/LwT** - reporta o estado de conexão do sistema ao *broker* MQTT, reportando *Online/Offline* através de uma mensagem *retained*;
- **vbox/07760/fa/period** - reporta um OK/NOK caso a alteração da periodicidade tenha sido realizada com sucesso ou insucesso, respetivamente;
- **vbox/07760/fa/diconnect** - reporta um OK/NOK caso a desconexão ao *broker* MQTT tenha sido realizada com sucesso ou insucesso, respetivamente.

No Anexo E, a Figura E.1 representa a interface visual do MQTT *Explorer* ligado ao *broker* MQTT do sistema de contagem. Apresenta a árvore de tópicos associada ao dispositivo *vbox*, indicando que o sistema se encontra *Online* e contendo informações relativas ao sistema, no tópico *info/*, assim como o tópico *count/* que, periodicamente, recebe um reporte de dados relativos à contagem, assim como da temperatura interna do sistema. Adicionalmente, é visível o envio de comandos através dos tópicos *dt/* e *fe/*, e as respetivas respostas por parte do sistema nos tópicos *dr/* e *fa/* correspondentes. Como se constata na Figura E.1, é visível o envio de um pedido de desconexão pelo período de 1 minuto, sendo este realizado com sucesso. Esta desconexão comprova-se com recurso ao gráfico a amarelo, verificando-se um intervalo de tempo equivalente de 1 minuto em que não houve qualquer mensagem recebida.

Neste capítulo apresentaram-se as técnicas e as ferramentas utilizadas nos diferentes componentes do sistema, como também se descreveram os diferentes cenários de testes para avaliação do protótipo desenvolvido. No próximo capítulo são apresentados os resultados da performance do protótipo tanto a nível de *Hardware* como de *Software*.

Capítulo 6

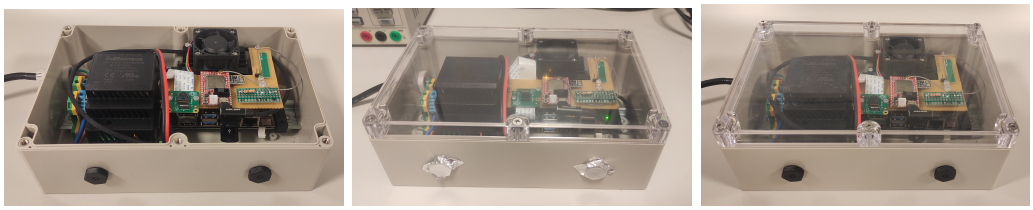
Resultados

Neste capítulo são apresentados os resultados dos testes realizados ao sistema desenvolvido. Três diferentes testes são apresentados: (1) análise da performance do *Hardware* da Jetson Nano *Dev Kit*, sob diferentes condições de isolamento do meio envolvente; (2) análise da qualidade de contagem, com diferentes algoritmos de *tracking* e, por fim, (3) validação da totalidade do sistema num cenário real de aplicação.

6.1 Testes de *Hardware*

Como forma de validação do comportamento mecânico e elétrico do sistema, foi realizado o primeiro cenário de testes, com 3 tipos de isolamento do sistema:

- **Aberto** - o interior da caixa encontra-se sem isolamento, isto é, os módulos do sistema estão diretamente em contacto com o meio exterior (Figura 6.1a);
- **Fechado** - o interior da caixa encontra-se totalmente isolado, isto é, os módulos do sistema não apresentam qualquer contacto com o meio exterior que permita fluxo de ar entre os dois meios (Figura 6.1b);
- **Parcial** - o interior da caixa encontra-se parcialmente isolado, isto é, os módulos do sistema apresentam parcial isolamento com o meio exterior, existindo apenas um fluxo limitado de ar através dos módulos de respiração (Figura 6.1c).



(a) Sem isolamento (b) Isolamento total (c) Isolamento parcial

Figura 6.1: Cenários de isolamento do sistema para com o meio exterior.

Os dados a analisar correspondem ao conteúdo reportado para o ficheiro de *logger* de HW. O registo de dados foi realizado com uma periodicidade de 500 ms ao longo de 2 horas de contínuo funcionamento, sem qualquer deteção de objetos neste período.

As Figuras 6.2 a 6.6 representam gráficos que reportam para cada tipo de isolamento, o comportamento dos parâmetros adquiridos ao longo das 2 horas. Estes parâmetros correspondem ao nível de utilização do CPU, GPU, RAM, sistema de ventilação e temperatura interior da caixa de isolamento.

Segue-se uma análise individual dos dados obtidos relativamente a cada parâmetro medido. A Tabela 6.1 resume esta análise com a listagem dos valores médios destes parâmetros para cada tipo de isolamento.

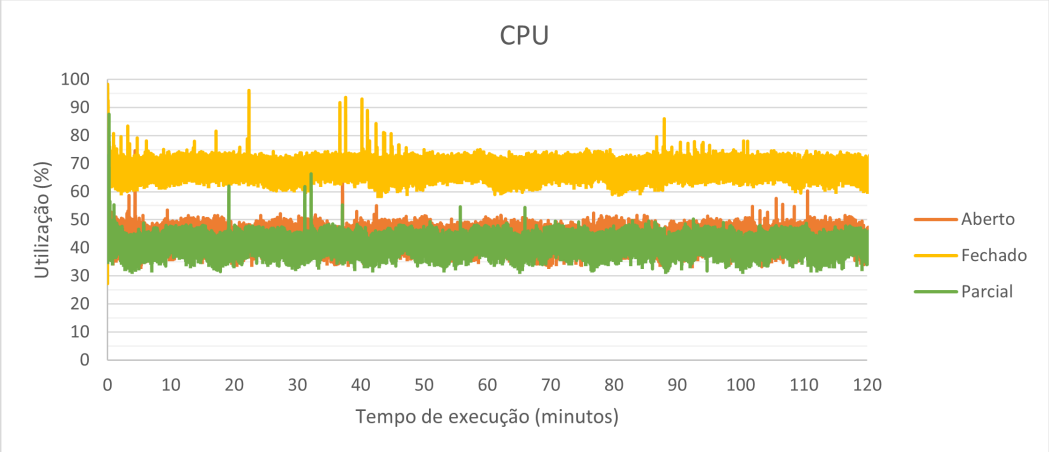


Figura 6.2: Representação da utilização média do CPU.

Na Figura 6.2 verifica-se que o modo *Fechado* apresenta maior utilização do CPU ($69,2\%$), enquanto que o modo *Parcial* está relativamente semelhante ao do modo *Aberto*. Com isto, consegue-se demonstrar o impacto positivo da presença dos módulos de respiração no isolamento *Parcial*. O maior consumo no modo *Fechado* justifica-se pela diminuição da frequência de operação para controlo do aumento da temperatura. Esta temperatura é medida por um sensor localizado na zona de alimentação da Jetson Nano *Dev Board*, denominada *always on thermal zone* [176].

Segundo documentação da NVIDIA, quando a temperatura nesta zona ultrapassa os 50 °C, os *clocks* reduzem automaticamente a frequência de operação como medida de segurança. Com esta redução, cada *core* do CPU acabará por ter de compensar esta redução de frequência com uma maior percentagem de utilização.

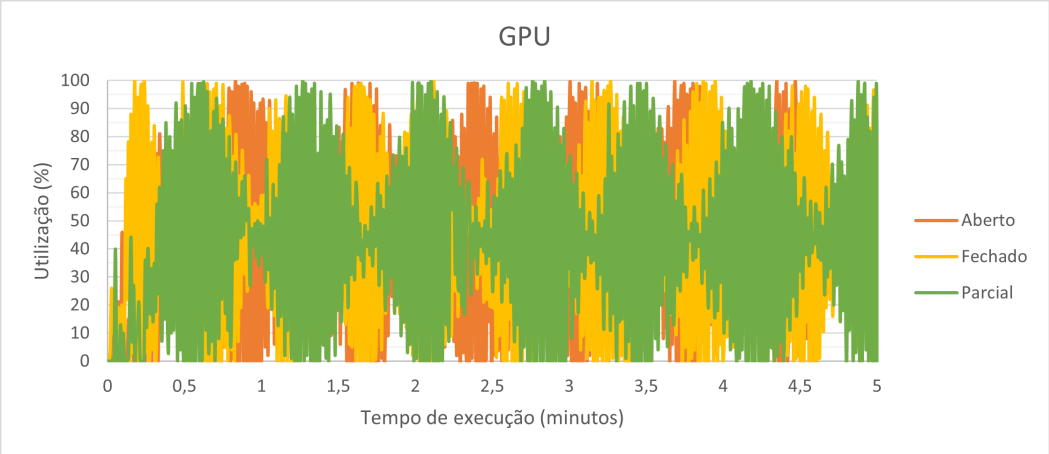


Figura 6.3: Representação da utilização do GPU.

Relativamente à utilização do GPU, verifica-se pela Figura 6.3 que a sua utilização é bastante semelhante nos 3 tipos de isolamento e, segundo a Tabela 6.1, os valores médios de utilização do GPU são praticamente iguais, na ordem dos 45%. Desta forma, conclui-se que o GPU não apresenta variações significativas ($< 1\%$) com o nível de isolamento.

Sobre o nível de utilização de RAM, o valor médio é semelhante com os diferentes isolamentos (66%), apresentando pequenas flutuações até 2%, conforme indicado na Tabela 6.1. Assim, verifica-se que também não há propriamente uma relação direta entre os diferentes níveis de isolamento e a utilização de RAM.

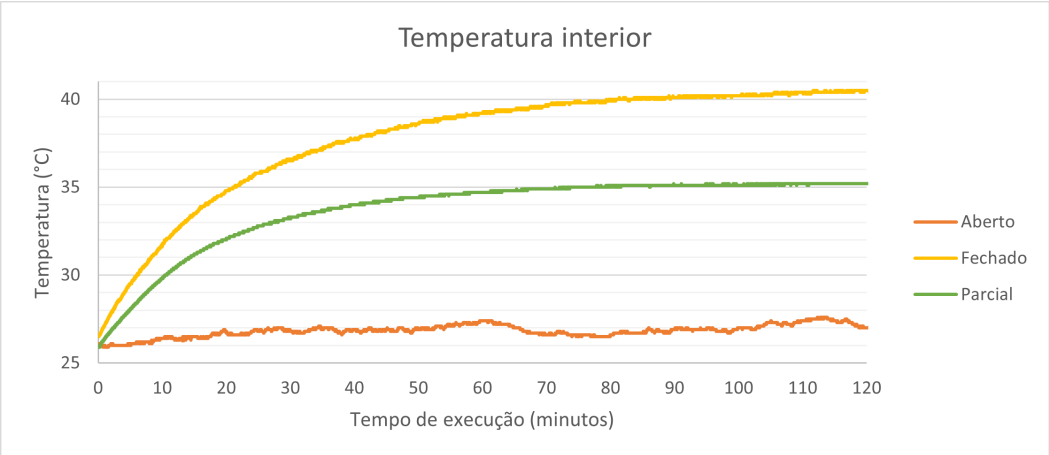


Figura 6.4: Representação da evolução da temperatura interior da caixa.

Quanto ao comportamento da temperatura interior da caixa de isolamento (Figura 6.4), verifica-se que no modo *Aberto* é mantida uma temperatura praticamente constante (27 °C), tendo em conta que a temperatura ambiente ronda os 23,5 °C. Quanto ao modo *Parcial*, a temperatura estabiliza nos 35 °C. Já no modo *Fechado* a sua temperatura tem um crescimento maior e mais rápido, estabilizando apenas nos 40,5 °C.

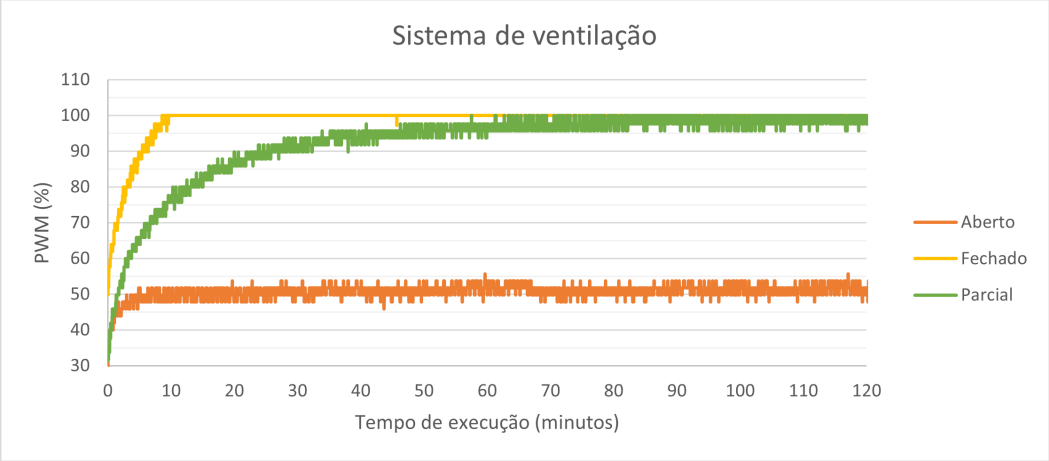


Figura 6.5: Representação da utilização do sistema de ventilação.

Relativamente ao sistema de ventilação, a Figura 6.5 demonstra que, em modo *Aberto*, o nível de PWM (medida da taxa de utilização do sistema de ventilação) é praticamente constante, na ordem dos 50%, uma vez que a temperatura interior é relativamente constante. Já nos outros modos, é visível em ambos uma utilização máxima do sistema de ventilação. No modo *Parcial* a utilização máxima é atingida após 1 hora de utilização, enquanto que no modo *Fechado* é em apenas 10 minutos. Com base nesta análise, verifica-se que o nível de utilização do sistema de ventilação está diretamente relacionado com a evolução da temperatura interna.

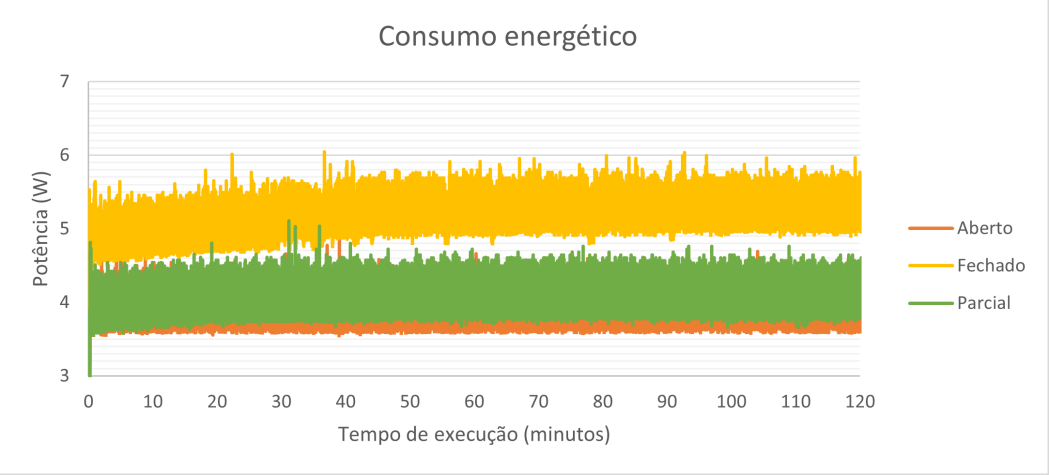


Figura 6.6: Representação do consumo energético.

Por fim, o consumo energético representado na Figura 6.6 demonstra que em modo *Fechado* (5,17 W) é cerca de 33% superior face ao modo *Aberto* (3,91 W). Já no modo *Parcial* (4,05 W), os níveis de consumo aproximam-se mais dos consumos do modo *Aberto*, com apenas um acréscimo de $3,6\%$ de acordo com a Tabela 6.1.

Adicionalmente, para efeitos de níveis de referência de consumos e níveis de utilização, adquiriram-se também informações do comportamento do sistema em *Stand By* no modo *Aberto*, correspondendo ao desempenho do sistema apenas com o processo *Logger* do Estado do *Hardware*. Desta forma, estes valores correspondem ao consumo e utilização base do sistema, isto é, quando todos os restantes processos estão desativados.

Tabela 6.1: Níveis médios de utilização de componentes do sistema e consumos médios de energia.

Propriedades	Aberto	Fechado	Parcial	*Stand by*
CPU (%)	44,42±3,48	69,02±3,19	42,31±3,43	6,41±4,47
GPU (%)	44,76±32,68	44,49±32,68	45,27±32,82	2,10±9,30
RAM (%)	66,70±1,05	67,26±0,71	64,95±1,01	37,26±0,17
Consumo (W)	3,91±0,22	5,17±0,25	4,05±0,23	1,80±0,24

Com base na análise dos resultados deste cenário de testes de performance do HW, verifica-se que o sistema está longe de atingir as suas capacidades máximas, nomeadamente a Jetson Nano. Desta forma, torna-se seguro afirmar que a aplicação de um sistema de respiração trouxe vantagens significativas, tanto na diminuição do consumo de energia em cerca de 22%, como na diminuição em 14% da temperatura de estabilização, levando, indiretamente, ao aumento da vida útil do sistema. Assim, este sistema considera-se viável para longos períodos de constante funcionamento. Comparando com os consumos de energia dos produtos de mercado (Tabela A.1), este sistema apresenta, em testes de bancada, um consumo médio na ordem dos 4,05 W, que é inferior aos dos atuais produtos de mercado.

6.2 Testes de *Software*

Para analisar o desempenho do SW desenvolvido, fez-se a recolha de um *dataset* em vídeo no recinto do CEiiA, que inclui carros, carrinhas, motociclos, bicicletas e pedestres. Adquirido o vídeo, extraíram-se os intervalos de tempo onde não existiam quaisquer objetos, reduzindo assim a duração do *dataset* de 2 horas para cerca de 5 minutos.

Antes da realização de testes ao SW de contagem, definiu-se a posição da linha de contagem definida por dois pontos no plano da imagem, tendo resultado na linha a preto representada na Figura 6.7.

Na Figura 6.8 estão representados os vários filtros aplicados sob o *dataset* "Diurno (Original)"(Figura 6.8a). Foram aplicados 3 filtros de desfoque (Figuras 6.8b-6.8e)

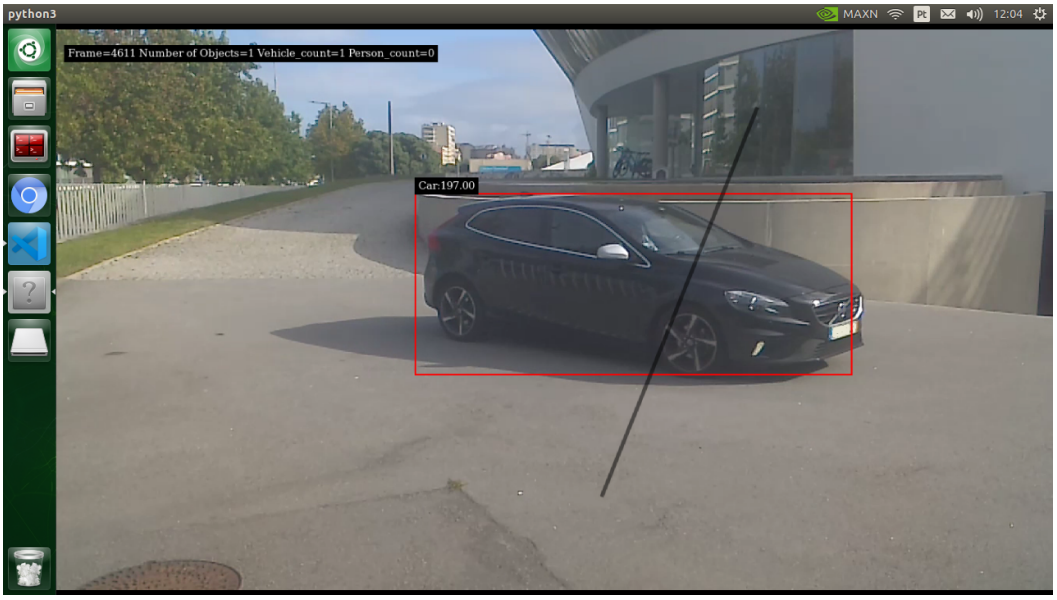
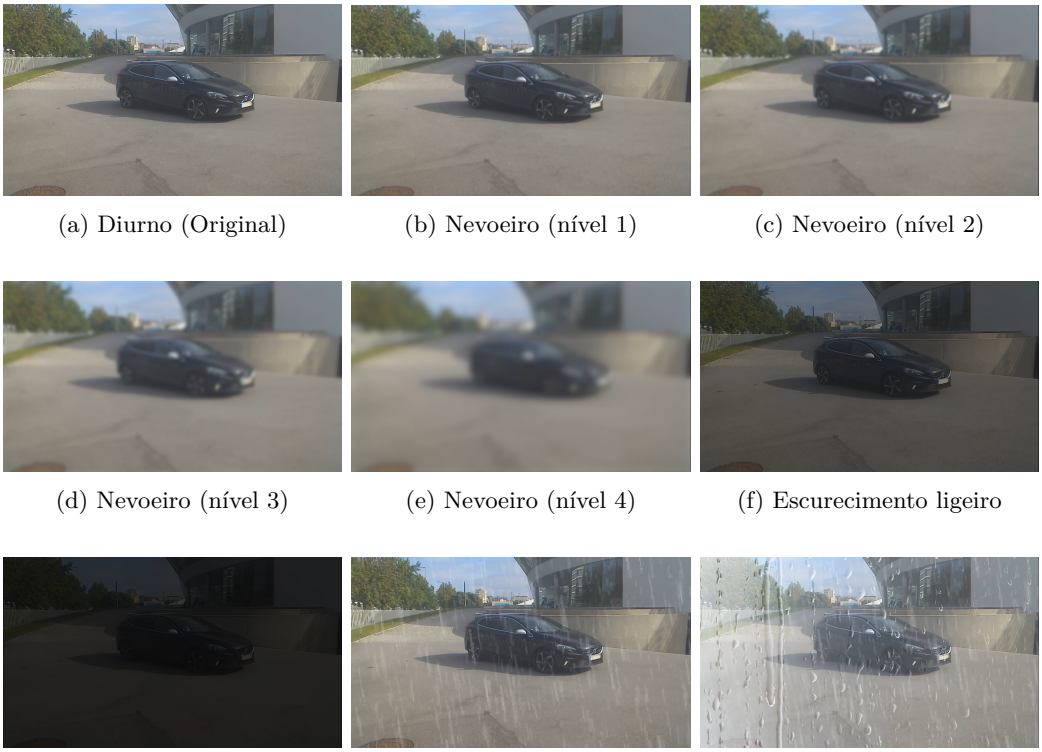


Figura 6.7: Representação da linha de contagem (linha preta) definida para este *dataset* assim como a deteção de um objeto (*bounding box* vermelha) classificado como um carro (*label* junto à *bounding box*).

para simulação de nevoeiro, 2 filtros de diferentes níveis de escurecimento (Figuras 6.8f e 6.8g) para simulação da noite e, por fim, um filtro simples de pluviosidade (Figura 6.8h) e com gotas de água no vidro (Figura 6.8i) para simulação de cenários de precipitação. Desta forma, através deste aumento de dados (*data augmentation*), obtiveram-se 8 novos *datasets*, tendo-se recorrido ao editor de vídeos Adobe Premier Pro e a filtros disponibilizados pela comunidade.

Mantidas as configurações do SW quanto à deteção e classificação dos objetos, procede-se à execução com os diferentes *datasets*, quantificando os objetos classificados e contabilizados através da linha de contagem. Adicionalmente, são contabilizados os parâmetros *True Positive* (TP), *True Negative* (TN), *False Positive* (FP) e *False Negative* (FN) das habituais métricas de classificação. Para cada *dataset*, aplicam-se individualmente os 2 algoritmos de *tracking*. O *tracker* KLT exige maior utilização do CPU, enquanto que o NvDCF faz maior utilização do GPU (de acordo com a Tabela 5.1). A Tabela 6.2 contém os resultados obtidos com a utilização do *tracker* KLT e a Tabela 6.3 os do NvDCF. Os valores de *ground truth* apresentados em ambas as tabelas correspondem à contagem manual de todos os objetos que cruzam a linha de contagem.

Com recurso aos parâmetros TP, FP e FN das Tabelas 6.2 e 6.3, são calculados os valores de *precision* e *recall* segundo as Equações 6.1 e 6.2, respetivamente, para os diferentes *datasets*. Desta forma, a Figura 6.9 apresenta um gráfico do tipo *precision-recall*, onde estão representadas as interpolações de cada um dos algoritmos



(a) Diurno (Original) (b) Nevoeiro (nível 1) (c) Nevoeiro (nível 2)

(d) Nevoeiro (nível 3) (e) Nevoeiro (nível 4) (f) Escurecimento ligeiro

(g) Escurecimento elevado (h) Pluviosidade ligeira (i) Pluviosidade elevada

Figura 6.8: *Dataset* original (a) e *datasets* com filtros de nevoeiro (b-e), escurecimento (f,g) e precipitação (h,i).

Tabela 6.2: Contagem dos objetos em cada *dataset* com o *tracker* KLT.

Dataset	Carros/Carrinhas	Motociclos/Bicicletas	Condutores*	Pedestres	TP	TN	FP	FN
Ground Truth	33	14	14	3	64	0	0	0
Diurno (Original)	30	13	4	3	50	0	1	13
Nevoeiro (nível 1)	32	13	8	3	56	0	1	7
Nevoeiro (nível 2)	33	13	5	3	54	0	0	10
Nevoeiro (nível 3)	6	2	0	2	10	0	0	54
Nevoeiro (nível 4)	0	0	0	0	0	0	1	64
Escurecimento ligeiro	33	13	6	3	55	0	0	9
Escurecimento elevado	13	11	6	3	33	0	1	30
Pluviosidade ligeira	23	13	5	3	44	0	1	20
Pluviosidade elevada	13	6	2	1	22	0	1	42

* referente apenas a condutores de motociclos e bicicletas.

Tabela 6.3: Contagem dos objetos em cada *dataset* com o *tracker* NvDCF.

Dataset	Carros/Carrinhas	Motociclos/Bicicletas	Condutores*	Pedestres	TP	TN	FP	FN
Ground Truth	33	14	14	3	64	0	0	0
Diurno (Original)	32	14	8	3	57	0	2	7
Nevoeiro (nível 1)	34	13	11	3	61	0	4	3
Nevoeiro (nível 2)	34	13	7	3	57	0	3	7
Nevoeiro (nível 3)	12	3	0	2	17	0	1	44
Nevoeiro (nível 4)	0	0	0	0	0	0	0	64
Escurecimento ligeiro	33	12	8	3	56	0	2	8
Escurecimento elevado	18	13	6	3	40	0	7	24
Pluviosidade ligeira	29	12	8	3	52	0	6	10
Pluviosidade elevada	22	10	5	3	40	0	2	23

* referente apenas a condutores de motociclos e bicicletas.

de *tracking*, interpolações resultantes da interligação entre os 9 pontos de *precision-recall* resultantes de cada *dataset*. Neste gráfico, o eixo vertical (*precision*) indica a precisão do SW, enquanto que o eixo horizontal (*recall*) indica a sensibilidade do mesmo.

Com base na Figura 6.9, verifica-se que os valores de *precision* de ambos os algoritmos de *tracking* apresentam valores com elevada precisão, isto é, bastante próximos do valor unitário, embora o *tracker* KLT seja o que apresenta melhor precisão. Por outro lado, os valores de *recall* são bastante dispersos em ambos os *trackers*, sendo que o *tracker* NvDCF tem melhores resultados, isto é, valores mais próximos do valor unitário.

$$Precision = \frac{TP}{TP + FN}. \tag{6.1}$$

$$Recall = \frac{TP}{TP + FP}. \tag{6.2}$$

Na Figura 6.10 são apresentados, sob a forma de gráfico de barras, os valores de *precision* e *recall* de cada algoritmo de *tracking* perante os vários *datasets*. O pior resultado verifica-se no *dataset* com maior nível de nevoeiro, não havendo nenhum objeto contabilizado por ambos os *trackers*. Nos *datasets* "Diurno"; "Nevoeiro (nível

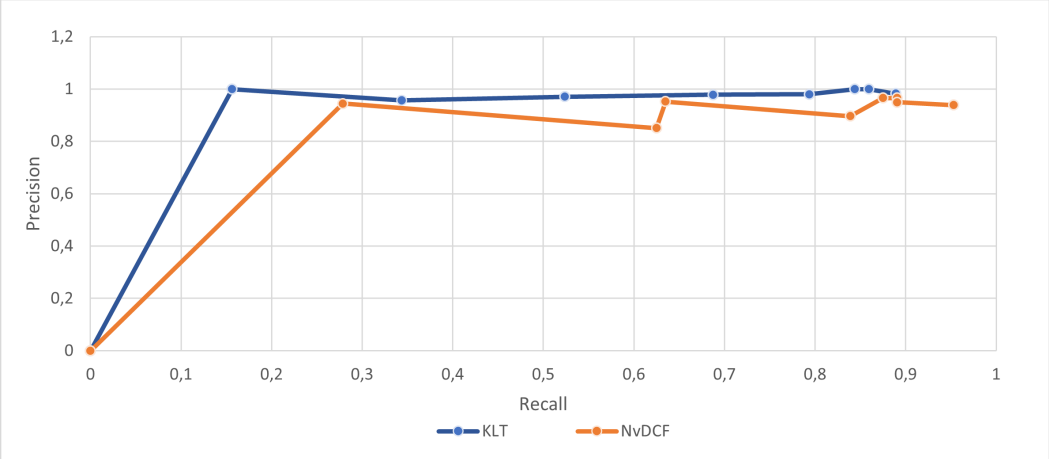


Figura 6.9: *Precision-Recall* de cada *tracker* resultante dos *datasets* analisados.

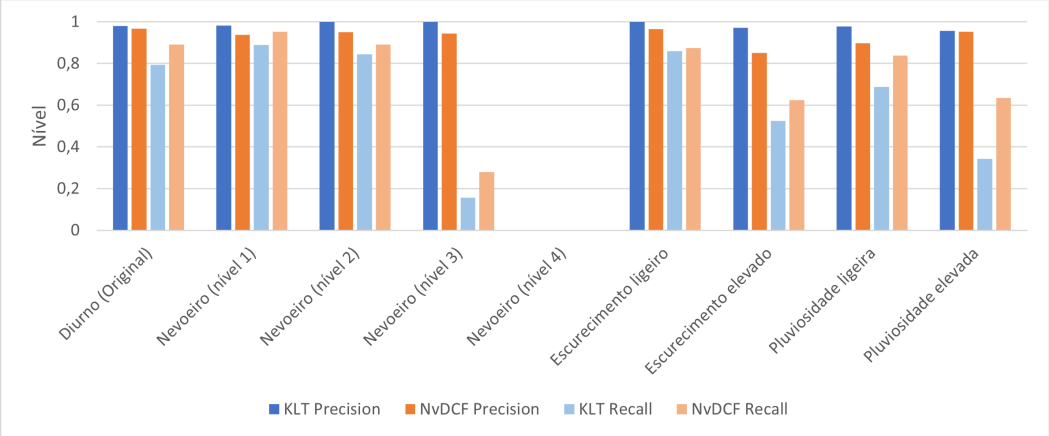


Figura 6.10: Gráfico de barras comparativo entre os valores de *precision* e *recall* de cada *tracker* para os diferentes *datasets*.

1)"; "Nevoeiro (nível 2)" e "Escurecimento ligeiro" são onde ambos os *trackers* apresentam todos os valores de *precision* e *recall* superiores a 0,8. Com isto, verifica-se que a contagem é idêntica em condições atmosféricas ligeiras. Quanto aos cenários mais adversos, como nos *datasets* "Nevoeiro (nível 3)"; "Escurecimento elevado"; "Pluviosidade ligeira" e "Pluviosidade elevada", a *precisão* mantém-se, mas o *recall* de ambos os *trackers* decresce significativamente (até −80%). Nestes cenários mais adversos, é o *tracker* NvDCF que apresenta um menor decréscimo.

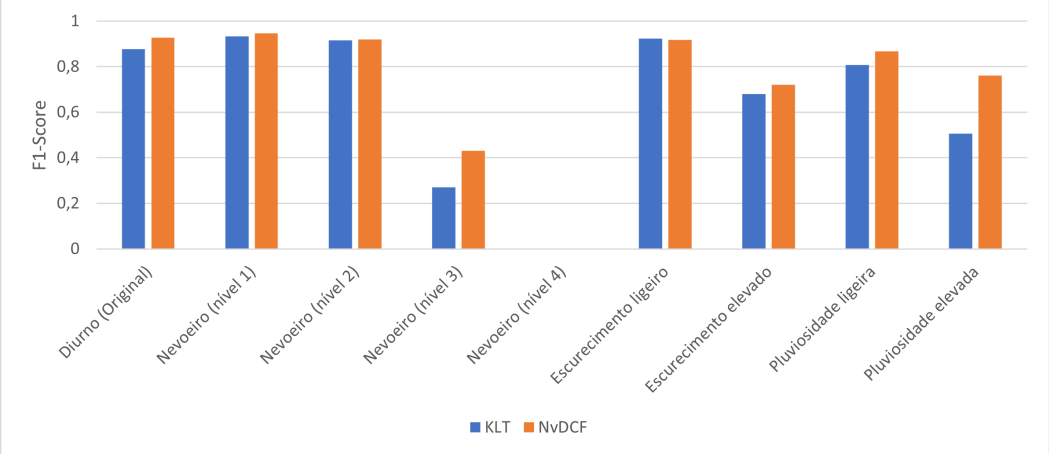


Figura 6.11: Gráfico da relação entre o F1-Score de cada *tracker* e os diferentes *datasets*.

Tabela 6.4: Comparação dos valores do F1-Score resultantes de cada *tracker* para os diferentes *datasets*.

Datasets	KLT	NvDCF
Diurno (Original)	0,88	0,93 (+5,7%)
Nevoeiro (nível 1)	0,93	0,95 (+2,2%)
Nevoeiro (nível 2)	0,91	0,92 (+1,1%)
Nevoeiro (nível 3)	0,27	0,43 (+59,3%)
Nevoeiro (nível 4)	0,00	0,00 (0,0%)
Escurecimento ligeiro	0,92	0,92 (0,0%)
Escurecimento elevado	0,68	0,72 (+5,9%)
Pluviosidade ligeira	0,81	0,87 (+7,4%)
Pluviosidade elevada	0,51	0,76 (+49,0%)

Analisando o gráfico da Figura 6.11 e a Tabela 6.4, sendo o parâmetro F1-Score calculado segundo a Equação 6.3, verifica-se que o NvDCF apresenta melhores resultados comparativamente ao KLT, principalmente em condições atmosféricas mais adversas (nível 3 de nevoeiro e pluviosidade elevada) com resultados de F1-Score 50% melhores, aproximadamente.

$$F1\_Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}. \tag{6.3}$$

Relativamente ao processo de contagem de objetos, conclui-se que:

- A eficácia da contagem é relativamente penalizada pela dificuldade da deteção contínua dos condutores de motociclos e bicicletas. Nas Tabelas 6.2 e 6.3 assinalável para os vários *datasets*, a discrepância entre os objetos contabilizados deste tipo e o valor *ground truth*;
- No *dataset* "Nevoeiro (nível 3)", os carros são corretamente detetados quando é visível a sua parte frontal ou traseira. Por outro lado, quando vistos de perfil a deteção falha significativamente, prejudicando o seu *tracking* junto à linha de contagem;
- Os motociclos e bicicletas são dificilmente reconhecidos pelo algoritmo de deteção perante uma vista frontal ou traseira destes. Neste caso, o seu *tracking* só é conseguido quando estes são avistados de perfil. Tendo em conta a posição da linha de contagem no *dataset* (Figura 6.7), o *tracking* deste tipo de objetos é favorecido;
- O nível de confiança mínimo usado no algoritmo de deteção e classificação dos objetos foi de 20%, dado que demonstrou melhores resultados para o ponto de vista do *dataset* recolhido;
- O *tracker* NvDCF demonstra maior dificuldade na associação de deteções duplicadas de um objeto, resultando numa contagem em excesso face ao *tracker* KLT.

Visto que num cenário real de aplicação as zonas de circulação de pedestres e de veículos não coincidem, o problema da contagem pode subdividir-se em duas linhas de contagem: uma sobre a zona pedonal e outra sobre a faixa de rodagem. Desta forma, deixam de ser contabilizadas as pessoas que conduzem bicicletas ou motociclos.

Quanto ao nível de utilização do CPU e GPU entre os *trackers* (Tabela 5.1), procedeu-se a uma análise comparativa da performance da Jetson Nano com cada um (KLT e NvDCF). Criou-se um *dataset* com uma duração de 0,54 horas, correspondendo a 48345 *frames* a 25 FPS. Este consiste na sequência de múltiplas cópias do *dataset* "Diurno (Original)". Neste teste apenas foram iniciados os processos de *tracking* e de *logging* de informação.

A Tabela 6.5 apresenta o resumo dos resultados relativamente à performance da Jetson Nano com a utilização de cada um dos *trackers*. Segundo esta, verifica-se que, para o processamento do *dataset*, o KLT consumiu 7,37 W durante um período de 0,18 horas, enquanto que o NvDCF consumiu apenas 5,74 W (−22%), embora durante 0,42 horas (+133%). Desta forma, verifica-se que, com o *tracker* NvDCF, a Jetson Nano tem 82% maior consumo energético comparado com o uso do KLT.

Tabela 6.5: Performance da Jetson Nano perante o processo de inferência de um *dataset* com recurso a diferentes *trackers*: KLT e NvDCF.

Parâmetros	KLT	NvDCF
Duração (horas)	0,18	0,42 (+133%)
Consumo (W)	7,37±0,81	5,74±0,48 (−22%)
Frame Rate (FPS)	75,66	31,80 (−58%)
CPU (%)	40,43±5,80	24,83±4,05 (+43%)
GPU (%)	96,29±14,67	86,38±12,43 (+109%)
RAM (%)	59,72±2,71	65,98±2,32 (+158%)
Ventilação (PWM)	76,50±6,80	63,50±3,30 (+94%)

Quanto ao desempenho, com o KLT conseguiu-se um *frame rate* médio de 75,66 FPS, enquanto que com o NvDCF foi de apenas 31,80 FPS (−58%). Relativamente ao nível de utilização dos restantes parâmetros (CPU, GPU, RAM e ventilação), apesar de, durante o processamento com o *tracker* NvDCF, os níveis médios sejam menores que os do KLT, na realidade é o NvDCF que acaba por exigir maior poder computacional para processar o mesmo *dataset*. Assim sendo, há uma utilização significativamente maior por parte do NvDCF, verificando-se aumentos de 43% do CPU; 109% do GPU; 158% de RAM e 94% de ventilação. Com estes dados, comprova-se que o NvDCF exige significativamente mais poder computacional que o KLT.

Tento em conta que o caso de aplicação deste sistema de contagem exigirá um funcionamento por longos períodos de tempo com um *frame rate* de 25 FPS, verifica-se que, independentemente do algoritmo de *tracking*, o *frame rate* do processo de *tracking* é superior ao da aquisição de imagens. Desta forma, assegura-se que nenhuma *frame* deixará de ser processada, não havendo acumulação de imagens na *queue* entre a aquisição de imagens da câmara e o início do processo de *tracking*.

Tendo em conta que o NvDCF apresenta maior consumo energético (+82%) e consegue uma melhor relação *precision-recall*, apresentando valores de F1-Score até 50% melhores, o algoritmo de *tracking* utilizado será o NvDCF.

6.3 Testes em Cenário Real

Como teste final, realizou-se o terceiro cenário, tratando-se de um caso de aplicação real. Este, em tempo-real, faz a contagem de pessoas e veículos, por um longo período de tempo (11 horas), utilizando o *tracker* NvDCF, assim como um reporte periódico de dados para o *broker* MQTT. Com isto, é utilizado o sistema em isolamento no modo *Parcial* para a validação da comunicação com o módulo NB-IoT e do SW de contagem através de imagens recolhidas diretamente pela câmara.

Na Figura 6.12 é apresentado um gráfico relativo ao comportamento da temperatura interna da caixa (linha verde), tendo como referência a temperatura ambiente exterior (linha castanha), assim como a curva referente à temperatura interna obtido em bancada com o mesmo tipo de isolamento (linha laranja). Conclui-se que, em ambiente *outdoor*, o sistema acaba por não conseguir uma estabilização da temperatura devido à variação da radiação solar que incide sobre a caixa ao longo do dia. Desta forma, os valores de temperatura interna da caixa acabam por tender para os 35 °C na fase final do teste devido à baixa incidência ao final do dia, caso contrário, a temperatura ronda os 40 °C para uma temperatura ambiente na ordem dos 25 °C obtidos durante o dia.

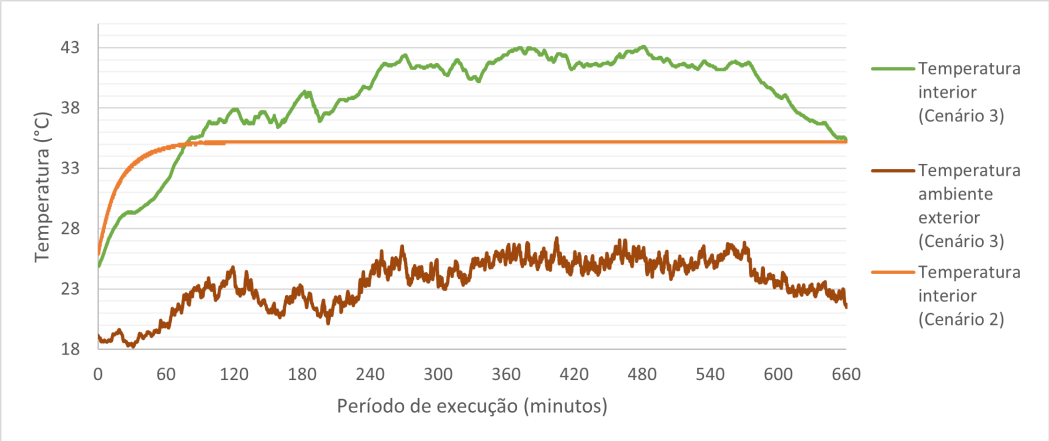


Figura 6.12: Gráfico da variação da temperatura interna da caixa (linha verde) em comparação com a temperatura exterior (linha castanha) e a temperatura a que estabilizou o modo *Parcial* em testes de bancada (linha laranja).

Relativamente à performance do sistema em modo *Parcial*, esta é resumida na Tabela 6.6. Tendo como termo de comparação os resultados dos testes de bancada para o mesmo modo de isolamento (Tabela 6.1), verifica-se que as percentagens de utilização do CPU, GPU e RAM se mantêm relativamente iguais em ambos os meios. Em contrapartida, a média de consumo energético é $6,9\%$ superior no meio exterior, justificando-se com o significativo aumento da temperatura interna atingida (máximo de 43 °C) comparativamente à atingida nos testes de bancada (máximo de 35 °C). Segundo [177], o aumento da temperatura envolvente provoca a diminuição da eficiência energética do sistema, sendo que a resistência do próprio material diminui perante o aumento desta temperatura, acabando por o sistema necessitar de um maior consumo energético para manter a sua performance.

No sentido de validar o funcionamento da comunicação do módulo NB-IoT com o *broker* MQTT, estabeleceu-se a conexão do cliente MQTT com recurso à ferramenta MQTT Explorer que assume o papel de operador do servidor remoto, subscrevendo-se os tópicos onde o sistema reporta periodicamente informação. Com base na Figura

Tabela 6.6: Comparação da performance do sistema em modo *Parcial* no meio interior e no meio exterior.

Parâmetros	Meio Interior	Meio Exterior
CPU (%)	42,31±3,43	43,15±4,71 (+2,0%)
GPU (%)	45,27±32,82	46,39±32,24 (+2,5%)
RAM (%)	64,95±1,01	67,99±1,34 (+4,7%)
Consumo (W)	4,05±0,23	4,33±0,63 (+6,9%)

F.1, do Anexo F, que apresentada a interface visual desta ferramenta, confirma-se a ligação do sistema ao *broker* MQTT através do tópico *LwT/* (estado *Online*), seguida da informação do sistema no tópico *info/* e, por fim, o tópico *count/* que recebe mensagens a cada 12 segundos. O conteúdo mais recente deste tópico é representado sob a forma de 6 gráficos temporais (representados também na Figura F.1) que quantificam, respetivamente: veículos, v; pessoas, p; bicicletas/motociclos, b; temperatura interna do sistema, tp; tensão V de alimentação e corrente C da *board* da Jetson Nano, sendo estes últimos 3 parâmetros temporários para efeitos de testes do sistema.

Na Figura 6.13 é apresentado o gráfico com todos os dados reportados, ao longo do dia, pelo sistema para o *broker* MQTT. Neste gráfico é representada a distribuição de um total de 481 contagens, verificando-se maior fluxo de pessoas nos períodos do início da manhã, almoço e fim de tarde.

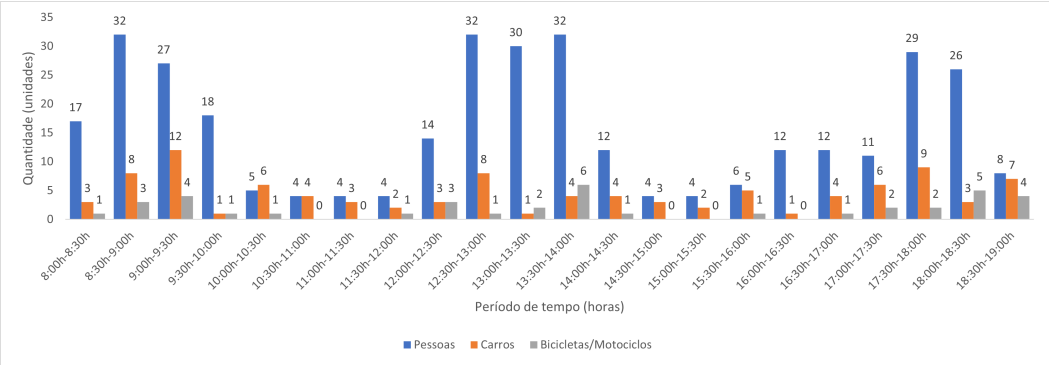


Figura 6.13: Gráfico representativo do fluxo tanto de pessoas, carros, bicicletas e motociclos ao longo do dia na entrada do recinto do CEiiA.

A Figura 6.14 apresenta a evolução temporal desde a inicialização do SO até ao instante onde o sistema estabelece a ligação ao *broker* MQTT e publica a primeira mensagem MQTT. O tempo total de iniciação é aproximadamente 60 segundos, sendo o arranque do SO mais demorado (32,00 segundos), seguida da ligação ao *broker* MQTT (26,58 segundos). Os restantes processos acabam por ser iniciados ao mesmo tempo que o processo de ligação ao *broker* MQTT. Após este período de iniciação, o sistema encontra-se em funcionamento.

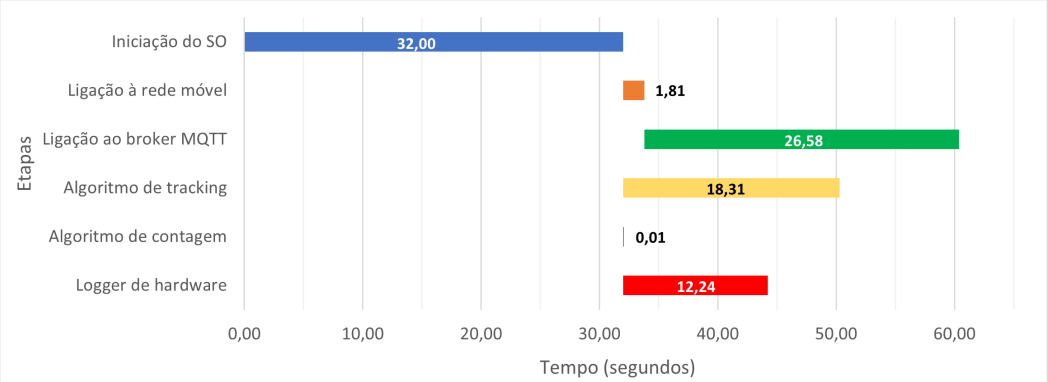


Figura 6.14: Gráfico temporal dos tempos de inicialização das principais fases do SW.

Neste capítulo apresentaram-se os resultados da performance do protótipo ao nível do *Hardware* e *Software*. No próximo capítulo são apresentadas as conclusões quanto aos resultados obtidos, assim como propostas dos próximos avanços neste sistema.

Capítulo 7

Conclusões e Trabalho Futuro

No presente trabalho é apresentada uma proposta de um sistema *low-cost* para deteção e contagem de pessoas e veículos, em tempo-real, e seguindo a tipologia *edge-computing* para o contexto de *smart cities*. Projetou-se a sua arquitetura de *Hardware* resultante de um *trade-off* de módulos necessários no qual se construiu um protótipo para fins de validação do funcionamento. Este sistema encontra-se equipado com: uma *board* NVIDIA Jetson Nano como unidade central de processamento; uma câmara modular da Raspberry Pi com aquisição de vídeo $720p$ a 25 FPS e por um módulo NB-IoT para comunicação com um servidor externo do operador. Quanto à componente de *Software*, foi implementado o conceito de multiprocessamento e exploradas as potencialidades do GPU através de ferramentas da NVIDIA. A deteção e classificação de pessoas e veículos recorrem a redes CNN pré-treinadas baseadas no modelo ResNet. O processo de *tracking* é realizado através do *tracker* NvDCF da NVIDIA e a contabilização tem por base o cruzamento de uma linha virtual de contagem.

Quanto aos resultados obtidos, nos testes de *Hardware* em bancada mostram a diminuição de 15% da temperatura interna com recurso a dois módulos de respiração, provocando a diminuição da utilização do CPU em 39% e do consumo energético em 22% para 4,05 W, sem colocar em causa o IP do sistema (IP65). Nos testes de *Software* simularam-se cenários com diferentes condições atmosféricas num *dataset* obtido no recinto do CEiiA. Verificaram-se dificuldades de deteção em cenários com níveis mais elevados de nevoeiro, escurecimento e pluviosidade. Quanto à utilização

dos *trackers* NvDCF e KLT, os resultados *precision-recall* são relativamente melhores com o NvDCF, embora o consumo energético seja ligeiramente inferior com o KLT. Perante um funcionamento do produto por um longo período de tempo (11 h), este contabilizou 481 objetos (343 pessoas, 99 carros/carrinhas e 39 bicicletas/-motociclos), assim como estabeleceu a comunicação com o módulo NB-IoT segundo o protocolo MQTT (3300 mensagens) com encriptação SSL/TLS, sem a indicação de problemas. Devido à longa exposição solar do sistema, a temperatura interna atingiu os 43 °C, o que despoletou um aumento do consumo energético médio de 7% para 4,33 W, sem afetar o seu correto funcionamento. Em contrapartida, a distinção entre carros e carrinhas assim como entre bicicletas e motociclos não foi conseguida visto que o algoritmo de deteção apenas faz a distinção das classes de pessoas, carros e bicicletas. Por fim, é de salientar que todos os testes realizados no meio tiveram em consideração o cumprimento do Regulamento Geral sobre a Proteção de Dados.

Resultante do desenvolvimento desta proposta de sistema, é de salientar o cumprimento da maioria do requisitos inicialmente estabelecidos:

- Validou-se a implementação de um produto *low-cost* com um valor material de 182,41 €;
- Verificou-se que o sistema apresenta consumos energéticos médios significativamente mais baixos (4,33 W) comparativamente aos produtos existentes no mercado (acima de 6 W);
- O produto consiste num sistema embebido do tipo *edge computing* (centralizado), com um formato compacto (222×146×75 mm) e adequado para instalação em ambiente *outdoor* com o IP65. Adicionalmente, é bastante mais leve (0,990 kg) que os produtos existentes no mercado (acima de 2,2 kg);
- Validou-se a contagem de pessoas em tempo-real, embora o *recall* apresente valores significativamente diferentes perante diferentes condições atmosféricas;
- Validou-se também a contagem de veículos em tempo-real, embora apresente dificuldades em condições atmosféricas mais adversas. Por outro lado, não foi conseguida a distinção entre carros e carrinhas, assim como entre bicicletas e motociclos;
- Visto o processamento de imagem ser em tempo-real, sem guardar permanentemente nem periodicamente imagens ou qualquer outro tipo de conteúdo que comprometa a privacidade das pessoas, assim como os dados de contagem também não possuírem quaisquer dados pessoais, pode-se garantir que o sistema cumpre com o RGPD, visto que se trata de um produto com finalidade comercial;

- Validou-se, por um longo período de tempo (11 horas), o reporte periódico de dados de contagem com uma periodicidade de 12 segundos através de um módulo NB-IoT. Este encontra-se ligado, através da rede móvel de uma operadora, a um *broker* MQTT segundo o protocolo de comunicação MQTT e encriptação desta segundo o protocolo SSL/TLS;
- O sistema, por meio do módulo NB-IoT, subscreve tópicos MQTT específicos que o permitem receber instruções provenientes do servidor remoto, como por exemplo a desativação temporária da conexão ao *broker* MQTT e a alteração da periodicidade de reporte das mensagens de contagem.

Para trabalho futuro, existem algumas funcionalidades tidas em conta ao longo do desenvolvimento desta dissertação e que não foram implementadas devido a constrangimentos de tempo e necessidade de fechar o desenvolvimento do primeiro protótipo do sistema. Assim sendo, segue-se a listagem dos próximos tópicos que se perspetivam desenvolver:

- Retreinar a rede CNN do algoritmo de deteção por forma a adicionar as classes de autocarros, carrinhas e motociclos. Desta forma conseguir-se-á fazer a validação da distinção destes perante as classes de carros e bicicletas;
- Adicionar ao sistema a capacidade de reconhecer o tipo de condição atmosférica do meio por forma a facilitar o processo de deteção de objetos;
- Preparação do produto para o Processo de Certificação do Produto segundo as directrizes na Comunidade Europeia para analisar a conformidade deste segundo os requisitos aplicáveis;
- Realizar testes quanto à performance do sistema relativamente à contagem de veículos que circulem na via pública a velocidades superiores às testadas até então;
- Dinamizar o processo de contagem na qual permita a adição de múltiplas linhas de contagem;
- Estimar a velocidade dos veículos no momento do cruzamento da linha de contagem;
- Adicionar tópicos MQTT que permitam, no momento da instalação do produto, a confirmação do correto posicionamento da câmara, seguida da configuração das linhas de contagem;
- Analisar a nova versão do SDK TensorRT 8.0 que apresenta melhor otimização da inferência de algoritmos *Deep Learning* [178];

- Desenvolvimento de uma PCB em substituição da placa perfurada que desempenha a função de interligação dos vários sensores à *board* de periféricos da Jetson Nano;
- Desenvolver um sistema de fixação do sistema nas infraestruturas existentes, como postes de iluminação e de sinalização semafórica;
- Implementar outros módulos de câmaras para analisar a influência da qualidade de imagem na eficácia do processo de deteção, assim como expor ao exterior a lente da câmara;
- Redimensionar o módulo de conversão AC/DC, visto que os consumos energéticos foram abaixo do previsto, optando pela solução mais barata, segundo o *trade-off* apresentado na Tabela B.3, diminuindo em 6,36 € o valor material do produto;
- Introdução de LED Infravermelhos para analisar possíveis melhorias do processo de deteção em condições atmosféricas com reduzida luminosidade;
- Desenvolver uma plataforma de interface para o operador do serviror, semelhante às abordadas na Seção 2.1.2.

Referências

[1] Dallmeier, "Panomera® cameras." Disponível em https://www.dallmeier.com/technology/panomera-cameras, 2019. (Último acesso em 15/10/2020). [Citado na página 6]

[2] Dallmeier, "Df5450hd-dn/ir." Disponível em https://www.dallmeier.com/fileadmin/user_upload/Downloads/Download_Centre/01_Documents/02_Camera/02_IR_Camera/DF5450HD-DN_IR/01_Specification/ds_DF5450HD_DN_IR_Ultraline_en.pdf, 2020. (Último acesso em 05/11/2020). [Citado nas páginas 6 e 7]

[3] Econolite, "New! autoscope® vision." Disponível em https://www.econolite.com/products/detection/autoscope-vision/, 2018. (Último acesso em 05/11/2020). [Citado nas páginas 6 e 7]

[4] Miovision, "Miovision trafficlink." Disponível em https://miovision.com/trafficlink/, 2020. (Último acesso em 05/11/2020). [Citado nas páginas 6 e 8]

[5] Retail Sensing, "Iot-vt: Clever solutions to urban problems." Disponível em https://www.retailsensing.com/smart-city-specs-web.pdf, 2020. (Último acesso em 05/11/2020). [Citado nas páginas 6 e 8]

[6] Teledyne FLIR, "Vehicle presence and data collection sensor flir traficam x-stream2." Disponível em https://www.flir.com/products/flir-traficam-x-stream2/?model=10-7150, 2021. (Último acesso em 10/5/2021). [Citado nas páginas 6 e 9]

[7] Econolite, "Sensors for traffic detection." Disponível em http://www.econolite.com/wp-content/uploads/sites/9/2017/06/detection-vision-datasheet.pdf, 2019. (Último acesso em 05/02/2021). [Citado na página 8]

[8] Miovision, "Introduction to video detection." Disponível em https://video.miovision.com/watch/aq58bhjcJc55rWiTENmVAv, 2020. (Último acesso em 10/11/2020). [Citado nas páginas 8 e 9]

[9] S. Ludwig, "Pro's and cons for ip vs. analog video surveillance." Disponível em https://www.securitymagazine.com/articles/

88854-pros-and-cons-for-ip-vs-analog-video-surveillance, 2018. (Último acesso em 07/11/2020). [Citado na página 9]

[10] Retail Sensing, "Vehicle traffic counting using cctv or ip-cameras." Disponível em https://www.retailsensing.com/vehicle-traffic-counting.html, 2020. (Último acesso em 05/11/2020). [Citado na página 9]

[11] A. Galletta *et al.*, "Mesmart-pro: Advanced processing at the edge for smart urban monitoring and reconfigurable services," *Journal of Sensor and Actuator Networks*, vol. 9, 2020. [Citado nas páginas 9 e 14]

[12] GoodVision, "Everything you need for transport data analytics in one place." Disponível em https://goodvisionlive.com/, 2020. (Último acesso em 15/11/2020). [Citado nas páginas 11, 12 e 13]

[13] Swisstraffic, "Product my.swisstraffic." Disponível em https://www.swisstraffic.ch/en/content/my_swisstraffic/my_swisstraffic/index.html, 2020. (Último acesso em 15/11/2020). [Citado nas páginas 11, 12 e 13]

[14] UNISEM IoT Division, "Unitraffic." Disponível em https://unisemiot.com/en/unitraffic/traffic-monitoring-system/, 2020. (Último acesso em 15/11/2020). [Citado nas páginas 11, 12 e 13]

[15] A. Santos *et al.*, "Counting vehicle with high-precision in brazilian roads using yolov3 and deep sort," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, (Las Vegas, USA), pp. 69–76, Nov. 2020. [Citado nas páginas 13 e 15]

[16] W. A. Okaishi *et al.*, "Real-time traffic light control system based on background updating and edge detection," in *2019 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, (Fez, Morocco), pp. 1–5, Apr. 2019. [Citado na página 14]

[17] K. Yang *et al.*, "Video-based traffic flow monitoring algorithm for single phase position at an intersection," in *2019 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*, (Guangzhou, China), pp. 101–105, Dec. 2019. [Citado na página 14]

[18] H. Yang and S. Qu, "Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition," *IET Intelligent Transport Systems*, vol. 12, 2017. [Citado na página 14]

[19] J. Barthélemy *et al.*, "Edge-computing video analytics for real-time traffic monitoring in a smart city," *Sensors*, vol. 19, no. 9, 2019. [Citado na página 14]

[20] D. Dinh *et al.*, "Towards ai-based traffic counting system with edge computing," *Journal of Advanced Transportation*, vol. 2021, 2021. [Citado na página 15]

[21] H. Zhang *et al.*, "Object tracking for a smart city using iot and edge computing," *Sensors*, vol. 19, no. 9, 2019. [Citado na página 15]

[22] K. U. Sreekumar *et al.*, "Real-time traffic pattern collection and analysis model for intelligent traffic intersection," in *2018 IEEE International Conference on Edge Computing (EDGE)*, (San Francisco, USA), pp. 140–143, July 2018. [Citado na página 15]

[23] C. Chen *et al.*, "An edge traffic flow detection scheme based on deep learning in an intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1840–1852, 2021. [Citado na página 16]

[24] S. Yang *et al.*, "Cosmos smart intersection: Edge compute and communications for bird's eye object tracking," in *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, (Austin, USA), pp. 1–7, Mar. 2020. [Citado na página 16]

[25] A. Marchetti, "Vehicles detection and tracking using convolutional neural networks on edge-computing platforms," Master's thesis, Polytechnic University of Turin, Turim, 2019. [Citado na página 16]

[26] Y. Yang *et al.*, "A method of pedestrians counting based on deep learning," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*, (Xiamen, China), pp. 2010–2013, Oct. 2019. [Citado nas páginas 17 e 68]

[27] AI City Challenges, "Ai city challenge." Disponível em `https://www.aicitychallenge.org/`, 2021. (Último acesso em 08/08/2021). [Citado na página 17]

[28] M. Naphade *et al.*, "The 5th AI city challenge," *CoRR*, vol. abs/2104.12233, 2021. [Citado na página 17]

[29] Raspberry Pi, "Raspberry pi 4." Disponível em `https://www.raspberrypi.org/products/raspberry-pi-4-model-b/`, 2020. (Último acesso em 14/13/2020). [Citado nas páginas 18 e 20]

[30] NVIDIA, "Jetson nano developer kit." Disponível em `https://developer.nvidia.com/embedded/jetson-nano-developer-kit`, 2020. (Último acesso em 14/12/2020). [Citado nas páginas 18, 20, 54 e 55]

[31] Coral, "Dev board." Disponível em `https://coral.ai/products/dev-board/`, 2020. (Último acesso em 15/12/2020). [Citado nas páginas 18, 19 e 20]

[32] R. Zwetsloot, "Raspberry pi 4 specs and benchmarks." Disponível em `https://magpi.raspberrypi.org/articles/raspberry-pi-4-specs-benchmarks`, 2019. (Último acesso em 14/01/2021). [Citado na página 18]

[33] G. Halfacree, "Benchmarking the raspberry pi 4." Disponível em `https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b`, 2019. (Último acesso em 14/01/2021). [Citado na página 18]

[34] NVIDIA, "Community." Disponível em `https://developer.nvidia.com/embedded/community`, 2021. (Último acesso em 18/01/2021). [Citado na página 19]

[35] F. Serzhenko, "Benchmark comparison for jetson nano, tx2, xavier nx and agx." Disponível em `https://www.fastcompression.com/blog/jetson-benchmark-comparison.htm`, 2021. (Último acesso em 15/01/2021). [Citado na página 19]

[36] S. Weiss, "Nvidia's new jetson xavier nx compared to jetson tx2 and jetson nano." Disponível em `https://3dvisionlabs.com/2020/05/22/jetson-xavier-nx-compared-to-jetson-tx2-and-jetson-nano/`, 2020. (Último acesso em 15/01/2021). [Citado na página 19]

[37] D. Franklin, "Jetson nano brings ai computing to everyone." Disponível em `https://developer.nvidia.com/blog/jetson-nano-ai-computing/`, 2019. (Último acesso em 15/01/2021). [Citado nas páginas 19, 20 e 54]

[38] J. Jo *et al.*, "Benchmarking gpu-accelerated edge devices," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, (Busan, Korea (South)), pp. 117–120, Apr. 2020. [Citado na página 19]

[39] S. Ullah and D. Kim, "Benchmarking jetson platform for 3d point-cloud and hyper-spectral image classification," in *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 477–482, Feb. 2020. [Citado na página 19]

[40] T. Peng *et al.*, "Evaluating the power efficiency of visual slam on embedded gpu systems," in *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, (Dayton, OH, USA), pp. 117–121, Apr. 2019. [Citado na página 19]

[41] R. K. Gupta and S. D. Senturia, "Deepedgebench: Benchmarking deep neural networks on edge devices," in *9th IEEE International Conference on Cloud Engineering (IC2E)*, Aug. 2021. [Citado nas páginas 19 e 54]

[42] Coral, "Edge tpu performance benchmarks." Disponível em `https://coral.ai/docs/edgetpu/benchmarks/`, 2021. (Último acesso em 24/5/2021). [Citado na página 19]

[43] S. Cass, "Nvidia makes it easy to embed ai: The jetson nano packs a lot of machine-learning power into diy projects - [hands on]," *IEEE Spectrum*, vol. 57, no. 7, pp. 14–16, 2020. [Citado na página 19]

[44] M. Antonini *et al.*, "Resource characterisation of personal-scale sensing models on edge accelerators," in *Proceedings of the First International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, (New York, USA), p. 49–55, Nov. 2019. [Citado nas páginas 20 e 54]

[45] Supertek, "The ultimate guideline to customize a camera module." Disponível em `https://www.supertekmodule.com/customized-camera-module/`, 2021. (Último acesso em 12/01/2021). [Citado na página 21]

[46] IoTEDU, "Raspberry pi camera and its variety." Disponível em `https://iot4beginners.com/raspberry-pi-camera-and-its-variety/`, 2020. (Último acesso em 12/01/2021). [Citado na página 21]

[47] Raspberry Pi, "Camera module 2." Disponível em `https://www.raspberrypi.org/products/camera-module-v2/`, 2021. (Último acesso em 12/01/2021). [Citado nas páginas 21 e 22]

[48] Coral, "Camera." Disponível em `https://coral.ai/products/camera/`, 2021. (Último acesso em 12/01/2021). [Citado nas páginas 21 e 22]

[49] Raspberry Pi, "Camera." Disponível em `https://www.raspberrypi.org/documentation/accessories/camera.html`, 2021. (Último acesso em 12/01/2021). [Citado na página 22]

[50] N. O. Mahony *et al.*, "Deep learning vs. traditional computer vision," *CoRR*, vol. abs/1910.13796, 2019. [Citado na página 21]

[51] A. Zenadji, "Decoding the dichotomy: Traditional image processing versus deep learning," tech. rep., HCL Technologies, June 2020. [Citado na página 22]

[52] J. Cao *et al.*, "Mobile augmented reality: User interfaces, frameworks, and intelligence," *CoRR*, vol. abs/2106.08710, 2021. [Citado na página 22]

[53] M. Carranza-García *et al.*, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sensing*, vol. 1, no. 1, 2021. [Citado na página 22]

[54] Intel, "The difference between deep learning training and inference." Disponível em `https://www.intel.com/content/www/us/en/artificial-intelligence/posts/deep-learning-training-and-inference.html`, 2021. (Último acesso em 12/06/2021). [Citado na página 22]

[55] Papers With Code, "One-stage object detection models." Disponível em `https://paperswithcode.com/methods/category/one-stage-object-detection-models`, 2021. (Último acesso em 15/03/2021). [Citado na página 22]

[56] S. Pal *et al.*, "Deep learning in multi-object detection and tracking: state of the art," *Applied Intelligence*, vol. 51, 2021. [Citado nas páginas 23 e 24]

[57] Papers With Code, "Yolov1." Disponível em `https://paperswithcode.com/method/yolov1`, 2021. (Último acesso em 15/03/2021). [Citado na página 23]

[58] Papers With Code, "Yolov2." Disponível em `https://paperswithcode.com/method/yolov2`, 2021. (Último acesso em 15/03/2021). [Citado na página 23]

[59] H. Ampadu, "Yolov3 and yolov4 in object detection." Disponível em `https://ai-pool.com/a/s/yolov3-and-yolov4-in-object-detection`, 2021. (Último acesso em 15/05/2021). [Citado na página 23]

[60] G. Boesch, "Object detection in 2021: The definitive guide." Disponível em `https://viso.ai/deep-learning/object-detection/`, 2021. (Último acesso em 15/08/2021). [Citado na página 24]

[61] J. Jordan, "An overview of object detection: one-stage methods." Disponível em `https://www.jeremyjordan.me/object-detection-one-stage/`, 2018. (Último acesso em 15/03/2021). [Citado na página 24]

[62] G. Ciaparrone *et al.*, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020. [Citado na página 24]

[63] S. Chen *et al.*, "Deep learning for multiple object tracking: A survey," *IET Computer Vision*, vol. 13, pp. 355–368, 2019. [Citado na página 24]

[64] J. H. Yoon *et al.*, "Online multi-object tracking via structural constraint event aggregation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, (Las Vegas, USA), pp. 1392–1400, June 2016. [Citado na página 24]

[65] V. Mandal *et al.*, "Object detection and tracking algorithms for vehicle counting: A comparative analysis," *CoRR*, vol. abs/2007.16198, 2020. [Citado nas páginas 25 e 26]

[66] A. Nyström, "Evaluation of multiple object tracking in surveillance video," Master's thesis, Linköping University, Linköping, 2019. [Citado na página 25]

[67] R. Kanjee, "Deepsort — deep learning applied to object tracking." Disponível em `https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104`, 2020. (Último acesso em 15/04/2021). [Citado nas páginas 25 e 26]

[68] B. Georgievski, "Object detection and tracking in 2020." Disponível em `https://blog.netcetera.com/object-detection-and-tracking-in-2020-f10fb6ff9af3`, 2020. (Último acesso em 15/04/2021). [Citado na página 25]

[69] A. Sachan, "Zero to hero: A quick guide to object tracking: Mdnet, goturn, rolo." Disponível em `https://cv-tricks.com/object-tracking/quick-guide-mdnet-goturn-rolo/`, 2021. (Último acesso em 15/04/2021). [Citado na página 25]

[70] F. J. Ansari *et al.*, "Comparison and study of pedestrian tracking using deep sort and state of the art detectors," *Elementary Education Online*, vol. 20, no. 5, pp. 7848–7859, 2021. [Citado na página 25]

[71] K. Mekki *et al.*, "A comparative study of lpwan technologies for large-scale iot deployment," in *ICT Express*, pp. 1–7, Mar. 2019. [Citado nas páginas 26, 28, 29 e 56]

[72] Embien, "Introduction to lora technology." Disponível em `https://www.embien.com/blog/introduction-to-lora-technology/`, 2019. (Último acesso em 10/2/2021). [Citado na página 27]

[73] The Things Network, "What are lora and lorawan?." Disponível em `https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/`, 2021. (Último acesso em 10/2/2021). [Citado na página 27]

[74] L. Alliance, "Rp2-1.0.1 lorawan® regional parameters." Disponível em `https://lora-alliance.org/resource_hub/rp2-101-lorawan-regional-parameters-2/`, 2021. (Último acesso em 10/2/2021). [Citado na página 28]

[75] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *2017 IEEE International Systems Engineering Symposium (ISSE)*, (Vienna, Austria), pp. 1–7, Oct. 2017. [Citado nas páginas 29, 30 e 31]

[76] T. Salman and R. Jain, "A survey of protocols and standards for internet of things," in *Advanced Computing and Communications*, Feb. 2019. [Citado nas páginas 29, 30 e 31]

[77] S. Jaikar and R. Kamatchi, "A survey of messaging protocols for iot systems," in *International Journal of Advanced in Management, Technology and Engineering Sciences*, (Pune, India), pp. 510–514, Feb. 2018. [Citado nas páginas 29, 30, 31 e 32]

[78] J. Dizdarević *et al.*, "A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration," in *Association for Computing Machinery*, (New York, USA), pp. 1–30, Aug. 2018. [Citado nas páginas 29 e 31]

[79] A. D. Pathak and J. V. Tembhurne, "Internet of things: A survey on iot protocols," in *Proceedings of 3rd International Conference on Internet of Things and Connected Technologies (ICIoTCT)*, (Jaipur, India), pp. 483–487, Mar. 2018. [Citado na página 29]

[80] RingCentral, "Quality of service (qos)." Disponível em `https://www.ringcentral.co.uk/gb/en/blog/definitions/quality-of-service-qos/`, 2021. (Último acesso em 10/2/2021). [Citado na página 32]

[81] Intersoft Consulting, "General data protection regulation gdpr." Disponível em `https://gdpr-info.eu/`, 2021. (Último acesso em 15/03/2021). [Citado na página 32]

[82] Comissão Europeia, "Para que serve o regulamento geral sobre a proteção de dados (rgpd)?." Disponível em `https://ec.europa.eu/info/law/law-topic/data-protection/reform/what-does-general-data-protection-regulation-gdpr-govern_pt`, 2021. (Último acesso em 15/03/2021). [Citado na página 32]

[83] SIC Notícias, "Bruxelas vai escolher operadora em portugal para analisar mobilidade durante pandemia de covid-19." Disponível em `https://sicnoticias.pt/especiais/coronavirus/2020-03-31-Bruxelas-vai-escolher-operadora-em-Portugal-para-analisar-mobilidade-d` 2020. (Último acesso em 18/03/2021). [Citado na página 33]

[84] Comissão Europeia, *Orientações respeitantes a aplicações móveis de apoio à luta contra a pandemia de COVID-19 na perspetiva da proteção de dados (2020/C 124 I/01)*, 2020. [Citado na página 33]

[85] J. Brownlee, "How do convolutional layers work in deep learning neural networks?." Disponível em `https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/`, 2020. (Último acesso em 17/8/2021). [Citado na página 35]

[86] M. Mishra, "Convolutional neural networks, explained." Disponível em `https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939`, 2020. (Último acesso em 17/8/2021). [Citado na página 35]

[87] L. Alzubaidi *et al.*, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions." Disponível em `https://doi.org/10.1186/s40537-021-00444-8`, 2021. (Último acesso em 17/8/2021). [Citado nas páginas 35, 36 e 37]

[88] A. Sachan, "Detailed guide to understand and implement resnets." Disponível em `https://cv-tricks.com/keras/understand-implement-resnets/`, 2021. (Último acesso em 17/8/2021). [Citado na página 37]

[89] H. Mujtaba, "Introduction to resnet or residual network." Disponível em `https://www.mygreatlearning.com/blog/resnet/#`, 2020. (Último acesso em 17/8/2021). [Citado na página 37]

[90] R. Nandepu, "Understanding and implementation of residual networks(resnets)." Disponível em `https://medium.com/analytics-vidhya/understanding-and-implementation-of-residual-networks-resnets-b80f9a507b9c`, 2019. (Último acesso em 17/8/2021). [Citado na página 38]

[91] NVIDIA, "Gst-nvtracker." Disponível em `https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_plugin_gst-nvtracker.html#`, 2021. (Último acesso em 17/6/2021). [Citado nas páginas 38, 39 e 85]

[92] FxPro, "Média móvel exponencial (mme) (exponential moving average (ema))." Disponível em `https://pt.fxpro.com/help-section/traders-glossary/exponential-moving-average-ema`, 2021. (Último acesso em 17/8/2021). [Citado na página 38]

[93] Everything Computer Science, "Stacks and queues." Disponível em `https://everythingcomputerscience.com/discrete_mathematics/Stacks_and_Queues.html`, 2021. (Último acesso em 17/3/2021). [Citado nas páginas 39 e 40]

[94] GeeksforGeeks, "Queue | set 1 (introduction and array implementation)." Disponível em `https://www.geeksforgeeks.org/`

queue-set-1introduction-and-array-implementation/, 2021. (Último acesso em 17/8/2021). [Citado nas páginas 39 e 40]

[95] GeeksforGeeks, "Types of queues." Disponível em https://www.baeldung.com/cs/types-of-queues, 2020. (Último acesso em 17/8/2021). [Citado na página 40]

[96] AfterAcademy, "Stack and queue." Disponível em https://afteracademy.com/tech-interview/ds-algo-concepts/stack-and-queue, 2021. (Último acesso em 17/8/2021). [Citado na página 40]

[97] A. Baatout, "Multithreading vs multiprocessing in python." Disponível em https://medium.com/contentsquare-engineering-blog/multithreading-vs-multiprocessing-in-python-ece023ad55a, 2018. (Último acesso em 17/3/2021). [Citado na página 41]

[98] S. Ghosh, "Multiprocessing vs. threading in python: What every data scientist needs to know." Disponível em https://blog.floydhub.com/multiprocessing-vs-threading-in-python-what-every-data-scientist-needs-to-know/, 2019. (Último acesso em 17/3/2021). [Citado na página 41]

[99] K. Castro, "Multiprocessor systems." Disponível em https://www.tutorialspoint.com/Multiprocessor-Systems, 2018. (Último acesso em 17/3/2021). [Citado na página 41]

[100] Intel, "Cpu vs. gpu: Making the most of both." Disponível em https://www.intel.com/content/www/us/en/products/docs/processors/cpu-vs-gpu.html, 2021. (Último acesso em 17/8/2021). [Citado nas páginas 41 e 42]

[101] Intel, "Introduction to gpus." Disponível em https://nyu-cds.github.io/python-gpu/01-introduction/, 2017. (Último acesso em 17/8/2021). [Citado na página 42]

[102] Intel, "What is a gpu?." Disponível em https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html, 2021. (Último acesso em 17/8/2021). [Citado na página 42]

[103] THG Hosting, "Gpu server processing performance and use cases." Disponível em https://thghosting.com/blog/gpu-server-processing-performance-and-use-cases/, 2020. (Último acesso em 17/8/2021). [Citado na página 42]

[104] MQTT, "Mqtt: The standard for iot messaging." Disponível em https://mqtt.org/, 2021. (Último acesso em 17/8/2021). [Citado nas páginas 42 e 43]

[105] HiveMQ, "Mqtt essentials." Disponível em `https://www.hivemq.com/mqtt-essentials/`, 2021. (Último acesso em 17/8/2021). [Citado nas páginas 43 e 45]

[106] "Messaging protocol for iot (part-1) – mqtt." Disponível em `https://8bitwork.com/2019/04/13/messaging-protocol-for-iot-mqtt-coap-amqp-xmpp-and-dds-1/`, 2019. (Último acesso em 17/8/2021). [Citado na página 46]

[107] M. Yuan, "Getting to know mqtt." Disponível em `https://developer.ibm.com/articles/iot-mqtt-why-good-for-iot/`, 2020. (Último acesso em 17/8/2021). [Citado na página 45]

[108] A. Prodromou, "Tls security 1: What is ssl/tls." Disponível em `https://www.acunetix.com/blog/articles/tls-security-what-is-tls-ssl-part-1/`, 2019. (Último acesso em 17/8/2021). [Citado na página 46]

[109] Cloudflare, "What is tls (transport layer security)?." Disponível em `https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/`, 2021. (Último acesso em 17/8/2021). [Citado na página 46]

[110] Cloudflare, "How does ssl work? | ssl certificates and tls." Disponível em `https://www.cloudflare.com/en-gb/learning/ssl/how-does-ssl-work/`, 2021. (Último acesso em 17/8/2021). [Citado na página 47]

[111] A. Prodromou, "Tls security 5: Establishing a tls connection." Disponível em `https://www.acunetix.com/blog/articles/establishing-tls-ssl-connection-part-5/`, 2019. (Último acesso em 17/8/2021). [Citado nas páginas 47 e 49]

[112] Cloudflare, "What happens in a tls handshake? | ssl handshake." Disponível em `https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/`, 2021. (Último acesso em 17/8/2021). [Citado na página 47]

[113] "How does ssl handshake work?." Disponível em `http://fakecineaste.blogspot.com/2021_02_21_archive.html`, 2021. (Último acesso em 17/8/2021). [Citado na página 49]

[114] J. Nguyen, "Sign server and client certificates." Disponível em `https://jamielinux.com/docs/openssl-certificate-authority/sign-server-and-client-certificates.html`, 2015. (Último acesso em 17/8/2021). [Citado na página 49]

[115] "Norma din: o que é e para que serve?." Disponível em https://tubonasa.com.br/noticias/a-norma-din, 2019. (Último acesso em 10/2/2021). [Citado na página 52]

[116] JetsonHacks, "Jetson nano + raspberry pi camera." Disponível em https://www.jetsonhacks.com/2019/04/02/jetson-nano-raspberry-pi-camera/, 2019. (Último acesso em 15/2/2021). [Citado na página 54]

[117] Seeed Studio Bazaar, "Raspberry pi camera module v2." Disponível em https://www.seeedstudio.com/Raspberry-Pi-Camera-Module-V2.html, 2020. (Último acesso em 15/12/2020). [Citado na página 55]

[118] Componentes101, "Pi camera module – 5mp." Disponível em https://components101.com/misc/pi-camera-module, 2018. (Último acesso em 15/2/2021). [Citado na página 55]

[119] Mauser, "Cartão de memória microsdhc 32gb classe 10 uhs-i + adaptador sd canvas select - kingston sdcs2/32gb." Disponível em https://mauser.pt/catalog/product_info.php?cPath=641_2548_1574&products_id=047-2895, 2020. (Último acesso em 15/12/2020). [Citado na página 56]

[120] Electronic Circuits Design, "Sd microsd card pinout." Disponível em https://www.electroniccircuitsdesign.com/pinout/sd-microsd-card-pinout.html, 2020. (Último acesso em 15/01/2021). [Citado na página 56]

[121] Olimex, "Nb-iot-devkit." Disponível em https://www.olimex.com/Products/IoT/NB-IoT/NB-IoT-DevKit/open-source-hardware, 2020. (Último acesso em 10/2/2021). [Citado nas páginas 56 e 57]

[122] Mouser, "Adafruit 3721." Disponível em https://pt.mouser.com/ProductDetail/Adafruit/3721/?qs=sGAEpiMZZMsvmS05OVPORkos2oZHucfR6rC1oQG%2FG13W9Ptj7sAfLw%3D%3D, 2021. (Último acesso em 10/2/2021). [Citado na página 58]

[123] Arduino Learning, "Am2320 temperature and humidity sensor and arduino example." Disponível em http://arduinolearning.com/code/am2320-temperature-humidity-sensor-arduino-example.php, 2021. (Último acesso em 10/2/2021). [Citado na página 58]

[124] Mouser, "Rz0232c." Disponível em https://pt.mouser.com/ProductDetail/Hammond-Manufacturing/RZ0232C?qs=%2Fha2pyFaduhPz3HOSbex%2Fdlszm5m7RMe7%2Fca2FGSEsvBLBHhRXkcBA%3D%3D, 2020. (Último acesso em 10/2/2021). [Citado na página 58]

[125] H. Manufacturing, "Part number: Rz0232c." Disponível em `https://www.hammfg.com/part/RZ0232C`, 2020. (Último acesso em 10/2/2021). [Citado na página 58]

[126] Farnell, "Mp-li60-20b05pr2." Disponível em `https://pt.farnell.com/multicomp-pro/mp-li60-20b05pr2/power-supply-ac-dc-5v-6-5a/dp/3556382`, 2021. (Último acesso em 10/2/2021). [Citado na página 59]

[127] Waveshare Electronics, "Dedicated cooling fan for jetson nano, pwm adjustment." Disponível em `https://www.waveshare.com/fan-4020-pwm-5v.htm`, 2021. (Último acesso em 10/2/2021). [Citado nas páginas 59 e 60]

[128] Mouser, "523-vent-ps1ncro8001." Disponível em `https://pt.mouser.com/ProductDetail/Heyco/3582VB?qs=kWcaGGHSxRq2X%252BcJlWWlEg%3D%3D`, 2021. (Último acesso em 10/2/2021). [Citado na página 60]

[129] Mauser, "Kit de microcomputador jetson nano quad-core 1.43ghz lpddr4 4gb." Disponível em `https://mauser.pt/catalog/product_info.php?products_id=047-2957`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[130] Mauser, "Módulo de câmara v2 oficial raspberry pi - 8mp - raspberry pi." Disponível em `https://mauser.pt/catalog/product_info.php?cPath=1667_2620_2956&products_id=096-4061`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[131] Olimex, "Nb-iot-devkit." Disponível em `https://www.olimex.com/Products/IoT/NB-IoT/NB-IoT-DevKit/open-source-hardware`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[132] Mauser, "Cartão de memória microsdhc 32gb classe 10 uhs-i + adaptador sd canvas select - kingston sdcs2/32gb." Disponível em `https://mauser.pt/catalog/product_info.php?cPath=641_2548_1574&products_id=047-2895`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[133] Mauser, "Adafruit 3721." Disponível em `https://pt.mouser.com/ProductDetail/Adafruit/3721/?qs=sGAEpiMZZMsvmS05OVPORkos2oZHucfR6rC1oQG%2FG13W9Ptj7sAfLw%3D%3D`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[134] Waveshare Electronics, "Dedicated cooling fan for jetson nano, pwm adjustment." Disponível em `https://www.waveshare.com/fan-4020-pwm-5v.htm`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[135] Mouser, "3582vb." Disponível em https://pt.mouser.com/ProductDetail/Heyco/3582VB?qs=kWcaGGHSxRq2X%252BcJlWWlEg%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[136] Farnell, "Mp-li60-20b05pr2." Disponível em https://pt.farnell.com/multicomp-pro/mp-li60-20b05pr2/power-supply-ac-dc-5v-6-5a/dp/3556382, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[137] Mouser, "Rz0232c." Disponível em https://pt.mouser.com/ProductDetail/Hammond-Manufacturing/RZ0232C/?qs=%2Fha2pyFaduhPz3HOSbex%2Fdlszm5m7RMe7%2Fca2FGSEsvBLBHhRXkcBA%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[138] Mouser, "Ltl2r3krd-em." Disponível em https://pt.mouser.com/ProductDetail/Lite-On/LTL2R3KRD-EM?qs=xb8aMrBSZRL5MG5wqCOHaQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[139] Mouser, "Cfr-12jt-52-220r." Disponível em https://pt.mouser.com/ProductDetail/Yageo/CFR-12JT-52-220R?qs=sGAEpiMZZMtlubZbdhIBIAw2MPiywMVp3fPO03r5Oes%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[140] Farnell, "Pelb0265." Disponível em https://pt.farnell.com/pro-elec/pelb0265/cable-gland-pa-nbr-4mm-8mm-grey/dp/3295865?ost=pelb0265, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[141] Mouser, "Cts2.5u-n." Disponível em https://pt.mouser.com/ProductDetail/Altech/CTS25U-N/?qs=a2ojXfwvUshmFL25fyTMjQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[142] Mouser, "5603400." Disponível em https://pt.mouser.com/ProductDetail/Phoenix-Contact/5603400/?qs=DAHzmYU9cu3YeQzkVE%2FSWQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[143] Mouser, "St1." Disponível em https://pt.mouser.com/ProductDetail/BusBoard-Prototype-Systems/ST1?qs=NGErKr1RxMAXV5lcv8OdyQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[144] Mouser, "1979." Disponível em https://pt.mouser.com/ProductDetail/Adafruit/1979/?qs=GURawfaeGuAYMxpXA2w8sQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[145] Mouser, "971250154." Disponível em https://pt.mouser.com/ProductDetail/Wurth-Elektronik/971250154/?qs=wr8lucFkNMVlvtjP0smSuQ%3D%3D, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[146] Farnell, "M2.510prstmcz100-." Disponível em `https://pt.farnell.com/tr-fastenings/m2-510prstmcz100/pan-head-pozidriv-screw-steel/dp/2770695`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[147] Farnell, "M2.5- hfst-z100-." Disponível em `https://pt.farnell.com/tr-fastenings/m2-5-hfst-z100/nut-full-steel-bzp-m2-5-pk100/dp/1419446`, 2021. (Último acesso em 05/3/2021). [Citado na página 65]

[148] K. Kocento, "Simple vehicle counting using opencv yolo3 python 3.5 and sort." Disponível em `https://www.youtube.com/watch?v=MHVNzIfsLVU`, 2019. (Último acesso em 05/3/2021). [Citado na página 70]

[149] "Orientações relativas à avaliação de impacto sobre a proteção de dados (aipd) e que determinam se o tratamento é «suscetível de resultar num elevado risco» para efeitos do regulamento (ue) 2016/679." Disponível em `https://www.cnpd.pt/media/f0ide5i0/aipd_wp248rev-01_pt.pdf`, 2017. (Último acesso em 05/3/2021). [Citado na página 76]

[150] NVIDIA, "Jetpack sdk." Disponível em `https://developer.nvidia.com/embedded/jetpack`, 2021. (Último acesso em 03/6/2021). [Citado na página 78]

[151] NVIDIA, "Getting started with jetson nano developer kit." Disponível em `https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit`, 2021. (Último acesso em 03/6/2021). [Citado na página 78]

[152] Python Software Foundation, "17.2. multiprocessing — process-based parallelism." Disponível em `https://docs.python.org/3.6/library/multiprocessing.html`, 2021. (Último acesso em 03/6/2021). [Citado na página 78]

[153] M. Uecker, "jetson-fan-ctl." Disponível em `https://github.com/Pyrestone/jetson-fan-ctl`, 2021. (Último acesso em 03/6/2021). [Citado na página 79]

[154] NVIDIA, "Jetson.gpio 2.0.17." Disponível em `https://pypi.org/project/Jetson.GPIO/`, 2021. (Último acesso em 03/6/2021). [Citado na página 79]

[155] A. Sears-Collins, "How to blink an led using nvidia jetson nano." Disponível em `https://automaticaddison.com/how-to-blink-an-led-using-nvidia-jetson-nano/`, 2021. (Último acesso em 03/6/2021). [Citado na página 79]

[156] O. Erichsen, "Getting started with the jetson nano." Disponível em `https://medium.com/@heldenkombinat/`

getting-started-with-the-jetson-nano-37af65a07aab#f208, 2019. (Último acesso em 17/4/2021). [Citado na página 80]

[157] NVIDIA, "Setting up vnc." Disponível em https://developer.nvidia.com/embedded/learn/tutorials/vnc-setup, 2021. (Último acesso em 17/4/2021). [Citado na página 80]

[158] FFMPEG, "Ffmpeg." Disponível em http://ffmpeg.org/, 2021. (Último acesso em 17/4/2021). [Citado na página 80]

[159] GStreamer, "gstreamer open source multimedia framework." Disponível em https://gstreamer.freedesktop.org/, 2021. (Último acesso em 17/4/2021). [Citado na página 81]

[160] JetsonHacksNano, "Csi-camera." Disponível em https://github.com/JetsonHacksNano/CSI-Camera, 2020. (Último acesso em 17/4/2021). [Citado na página 81]

[161] T. Nordquist, "Mqtt explorer." Disponível em http://mqtt-explorer.com/, 2021. (Último acesso em 17/4/2021). [Citado na página 82]

[162] R. Bonghi, "jetson-stats 3.1.0." Disponível em https://pypi.org/project/jetson-stats/, 2021. (Último acesso em 10/5/2021). [Citado na página 82]

[163] R. Bonghi, "jtop_logger.py." Disponível em https://github.com/rbonghi/jetson_stats/blob/master/examples/jtop_logger.py, 2021. (Último acesso em 10/5/2021). [Citado na página 82]

[164] K. Townsend, "Using the adafruit bmp python library (updated)." Disponível em https://learn.adafruit.com/using-the-bmp085-with-raspberry-pi/using-the-adafruit-bmp-python-library, 2021. (Último acesso em 03/6/2021). [Citado na página 82]

[165] NVIDIA, "Welcome to the deepstream documentation." Disponível em https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_Overview.html, 2021. (Último acesso em 17/6/2021). [Citado nas páginas 83 e 84]

[166] Z. Liu *et al.*, "Building intelligent video analytics apps using nvidia deepstream 5.0 (updated for ga)." Disponível em https://developer.nvidia.com/blog/building-iva-apps-using-deepstream-5-0-updated-for-ga/, 2020. (Último acesso em 17/6/2021). [Citado na página 83]

[167] NVIDIA, "Deepstream python apps." Disponível em `https://github.com/NVIDIA-AI-IOT/deepstream_python_apps`, 2021. (Último acesso em 17/6/2021). [Citado na página 84]

[168] NVIDIA, "Gst-nvinfer." Disponível em `https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_plugin_gst-nvinfer.html`, 2021. (Último acesso em 17/6/2021). [Citado na página 84]

[169] A. Mohan *et al.*, "Determining the necessary frame rate of video data for object tracking under accuracy constraints," in *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, (Miami, FL, USA), pp. 368–371, June 2018. [Citado na página 85]

[170] NVIDIA, "deepstream-test2." Disponível em `https://github.com/NVIDIA-AI-IOT/deepstream_python_apps/tree/master/apps/deepstream-test2`, 2021. (Último acesso em 17/6/2021). [Citado na página 87]

[171] NVIDIA, "Metadata in the deepstream sdk." Disponível em `https://docs.nvidia.com/metropolis/deepstream/dev-guide/text/DS_plugin_metadata.html`, 2021. (Último acesso em 17/6/2021). [Citado na página 87]

[172] TechPowerUp, "Nvidia jetson nano gpu." Disponível em `https://www.techpowerup.com/gpu-specs/jetson-nano-gpu.c3643`, 2021. (Último acesso em 17/6/2021). [Citado na página 87]

[173] J. Greene, "Understanding the basics of aixprt precision settings." Disponível em `https://www.principledtechnologies.com/benchmarkxprt/blog/2019/09/05/understanding-the-basics-of-aixprt-precision-settings/`, 2019. (Último acesso em 17/6/2021). [Citado na página 87]

[174] Quectel, "Lte bc66 nb-iot." Disponível em `https://www.quectel.com/product/lte-bc66-nb-iot/`, 2020. (Último acesso em 05/3/2021). [Citado na página 90]

[175] I. Creators, "Roaming network info." Disponível em `https://docs.iotcreators.com/docs/roaming-mobile-iot-networks-in-europe#nb-iot-in-europe`, 2021. (Último acesso em 20/9/2021). [Citado na página 90]

[176] NVIDIA, "Power management for jetson nano and jetson tx1 devices." Disponível em `https://docs.nvidia.com/jetson/l4t/index.html#page/Tegra%20Linux%20Driver%20Package%20Development%20Guide/`

power_management_nano.html#wwpID0E0HH0HA, 2021. (Último acesso em 17/6/2021). [Citado na página 96]

[177] A. Golda and A. Kos, "Temperature influence on power consumption and time delay," in *Euromicro Symposium on Digital System Design, 2003. Proceedings.*, (Belek-Antalya, Turkey), pp. 378 – 382, Sept. 2003. [Citado na página 107]

[178] D. O'Shea, "Nvidia releases tensor rt 8, with better performance, accuracy." Disponível em https://www.fierceelectronics.com/electronics/nvidia-releases-tensor-rt-8-better-perform-accuracy, 2021. (Último acesso em 12/9/2021). [Citado na página 113]

Anexo A

Características dos Produtos no Mercado

Este Anexo contém uma tabela que faz o levantamento das características gerais dos produtos no mercado abordados na Seção 2.1.1.

Tabela A.1: Características gerais dos produtos no mercado.

Propriedades	Panomera S-Series	Panomera W-Series	DF5450HD-DN/IR	Autoscope Vision	Miovision TrafficLink	IoT-VT	TrafiCam x-stream2
Resolução (MP)	190 MPe	95 MPe	12 MP (4K)	8 MP (HD)	9 MP (4K)	HD	VGA
FoV (°)	30/52	360/90	110/80	59,4/33,4	182/176	N/A	N/A
Frame Rate (FPS)	30	30	30	N/A	15	N/A	25
Alcance (m)	322	34	30	183	N/A	N/A	75
Câmaras (unidades)	8	8	1	1	1	1	1
Índice de Proteção	IP69	IP69	IP66	IP67	IP65	IP65	IP67
Consumo (W)	100 (+60*)	80 (+60*)	12,95	16	100	9	6-9
Peso (kg)	17,5	10	2,2	4,20**	40	4/7,5	N/A
Processamento	Local	Local	Local	Local	Local	Remoto	Local
Observações	Grande alcance	Visão ampla	Visão noturna	Curto FoV	Método *stop bar detection*	Baixo custo	Baixo consumo

* Consumo máximo do módulo de aquecimento aquando ligado.
** Apenas o peso do módulo de aquisição de imagem.

Anexo B

Trade-Off de Elementos para o Sistema

Este Anexo contém um conjunto de tabelas, sendo que cada uma apresenta um conjunto de elementos resultantes de um *trade-off* para desempenharem as funções de cada um dos módulos do sistema, incluindo elementos de fixação, tendo em conta um conjunto de requisitos impostos. Esta seleção está representada na Seção 4.1.2.

Tabela B.1: *Trade-off* de sensores de temperatura.

Vendedor	**Sensor**	**Gama** (°C)	**Resolução** (°C)	**Interface**	**Custo** (€)
Mouser	AM2320	0 - 99,9	0,1	I2C	2,72
Mouser	AHT20	0 - 100	0,01	I2C	3,10
Botnroll	BMP180	0 - 65	0,1	I2C	7,66

Tabela B.2: *Trade-off* de caixas de isolamento.

Vendedor	**Referência**	**IP**	**Dimensões*** (mm)	**Material**	**Custo** (€)
Mouser	RZ0353C	IP65	197×103×69	Plástico ABS	13,32
Mouser	RZ0232C	IP65	197×103×69	Policarbonato	17,05
Mouser	1554WA2GYCL	IP66	152×133×84	Policarbonato	27,51
Mouser	RP1465C	IP65	194×139×78	Plástico ABS	21,50
Mouser	105908	IP66	224×128×111	Poliestireno	26,17

* Indicados os valores das dimensões interiores da caixa de isolamento, ao invés das exteriores, visto garantir-se o espaço suficiente para a instalação dos restantes módulos no seu interior. Valores apresentados com aproximação à escala do milímetro.

Tabela B.3: *Trade-off* de conversores AC/DC com fixação 35/7,5, segundo a norma DIN.

Vendedor	**Referência**	**Potência** (W)	**Dimensões** (mm)	**Saídas**	**Custo** (€)
Mouser	709-HDR30-5	15	35×90×54,5	1	12,50
Mouser	709-RS25-5	25	28×92×51	1	7,36
Farnell	3556378	15	35×92,66×58	1	10,97
Farnell	3556382	32,5	52×92,66×58	2	13,72
Mouser	436-PBW30F-5	30	N/A	2	44,01
Mouser	436-PBW30F-5-N	30	N/A	2	47,41

Tabela B.4: *Trade-off* de módulos de ventilação.

Vendedor	**Referência**	**PWM**	**Dimensões** (mm)	**Consumo** (W)	**Custo** (€)
Waveshare	16681	Sim	40×40×20	N/A	2,00
Waveshare	16990	Não	40×40×10	N/A	1,00
Mouser	4020V-050165	Não	40×40×20	0,38	4,67
Mauser	048-0217	Não	40×40×20	0,54	6,69

Tabela B.5: *Trade-off* de módulos de respiração.

Vendedor	**Referência**	**IP**	**Dimensões** (mm)	**Custo** (€)
Mouser	546-SDA412	IP45	66×66×30,5	8,99
Farnell	1870858	IP66	60×60×37	15,07
Farnell	2811594	IP55	65,6×65,6×30,5	9,47
Farnell	2890730	IP68	17×17×17,5	9,99
Farnell	1870859	IP66	60×60×49,5	11,09
Mouser	PS1NCRO8001	IP68	17×17×15,8	2,56
Mouser	836-3582VB	IP68	16×16×13,5	2,28

Tabela B.6: *Trade-off* de bucins.

Vendedor	**Referência**	**IP**	**Custo (€)**
Mouser	IPG-2229-G	IP66	0,48
Mouser	1424475	IP66	0,45
Farnell	3295865	IP68	0,16
Farnell	3383255	IP68	0,21

Tabela B.7: *Trade-off* de blocos terminais DIN com fixação 35/7,5, segundo a norma DIN.

Vendedor	**Referência**	**Dimensões** (mm)	**Capacidade**	**Conectores**	**Custo (€)**
Mouser	ODL2.5A	5×62×63	300 V/ 25 A	2 + 2	1,70
Mouser	ODL2.5(I.S)	5×62×63	300 V/ 25 A	4	1,88
Mouser	3044076	5,2×47,7×46,9	1 kV/ 24 A	2	0,81
Mouser	CTS2.5U-N	5×43×45	600 V/ 25 A	2	0,76

Tabela B.8: *Trade-off* de espaçadores com comprimento de 25 milímetros.

Vendedor	**Referência**	**Furação**	**Custo* (€)**
Essentracomponents	1221358	M2.5 10 mm	0,0258
Essentracomponents	1221233	M2.5 10 mm	0,0455
Fabory	11398.025.025	M2.5 7 mm	0,0241
Mouser	971250154	M2.5 3 mm	0,61
Digi-Key	AE10791-ND	M2.5 6 mm	0,25
Digi-Key	10449-ND	M2.5 7 mm	0,46

* Custo de cada elemento de um *pack* com inúmeras unidades.

Tabela B.9: *Trade-off* de elementos de fixação.

Vendedor	**Referência**	**Tipo**	**Dimensões**	**Custo* (€)**
Farnell	2770695	Parafuso	M2.5, 10 mm	0,02
Farnell	2494523	Parafuso	M2.5, 10 mm	0,07
Farnell	2494547	Parafuso	M2.5, 10 mm	0,042
Farnell	1419446	Porca	M2.5	0,0133
Farnell	1420787	Porca	M2.5	0,0277
Farnell	1420780	Porca	M2.5	0,0998

* Custo de cada elemento de um *pack* com inúmeras unidades.

Anexo C

Estrutura de Pastas do *Software*

Este Anexo uma figura ilustrativa da estrutura de pastas do *Software* desenvolvido, seguida de uma breve descrição do conteúdo de cada uma das pastas. Este tema é introduzido pela Seção 5.3.

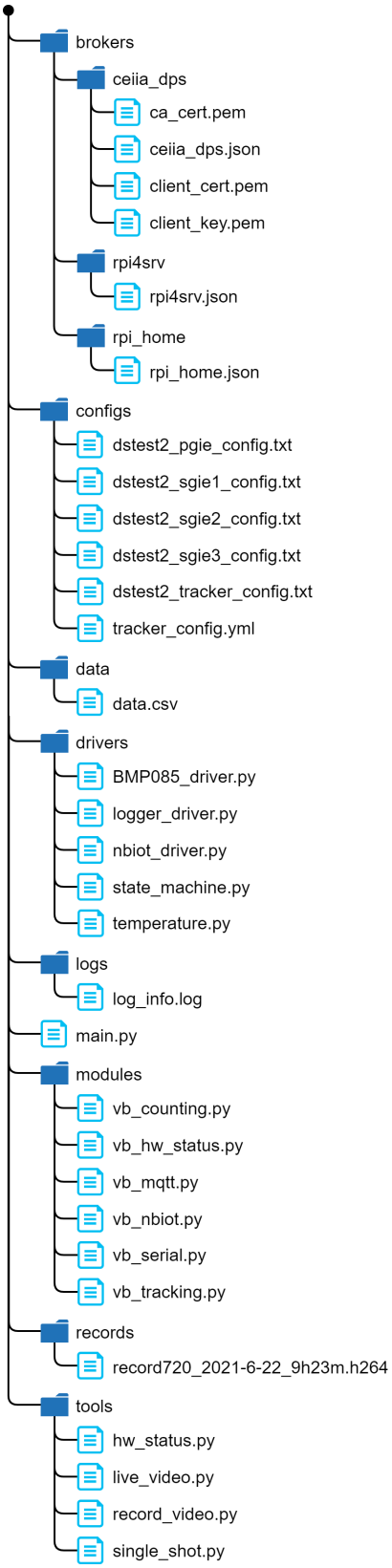


Figura C.1: Estrutura de pastas e ficheiros de SW.

Descrição do conteúdo de cada pasta:

- **"brokers"** - contém os parâmetros para conexão a cada um dos *brokers* de MQTT possíveis: endereço IP; número da porta; nome de utilizador; palavra-passe e, por fim, os certificados de autenticação para os *brokers*; chaves de encriptação, entre outros;
- **"configs"** - contém os ficheiros para configuração de parâmetros tanto do algoritmo de deteção e classificação, como do algoritmo de *tracking* de objetos. Consoante o cenário de aplicação, estes parâmetros têm necessidade de ser reajustados para garantir a maior eficácia destes algoritmos;
- **"data"** - contém um conjunto de ficheiros que, temporariamente, fazem registo da contagem de pessoas e veículos segundo uma ordem cronológica;
- **"drivers"** - contém ficheiros de *drives*: uma biblioteca personalizada para comunicação com o sensor de temperatura; configuração do *logger*; comandos AT para a comunicação com o módulo NB-IoT e a configuração da máquina de estados para a comunicação MQTT;
- **"logs"** - contém os ficheiros com as mensagens reportadas pelos *logger* de HW e SW do sistema;
- **"modules"** - contém 6 ficheiros de código que contém o SW relativo a funcionamento de cada processo do sistema;
- **"records"** - contém imagens e vídeos, temporariamente adquiridos, para efeitos de calibração da posição câmara no terreno, como também para testes de calibração dos algoritmos;
- **"tools"** - contém um conjunto de ferramentas utilizadas em fases de testes e calibração do sistema. As funcionalidades presentes são, respetivamente: aquisição periódica do estado do HW; realização de uma *stream* de vídeo; gravação de vídeo e captura instantânea de uma imagem;
- **"main.py"** - ficheiro principal que é executado para iniciar os vários processos do sistema.

Anexo D

Redes Neuronais ResNet

Este Anexo contém 2 figuras ilustrativas da estrutura das redes neuronais ResNet utilizadas para o processo de deteção, ResNet-10, e no de classificação, ResNet-18, de objetos, respetivamente. A sua abordagem é feita na Seção 5.7.2.

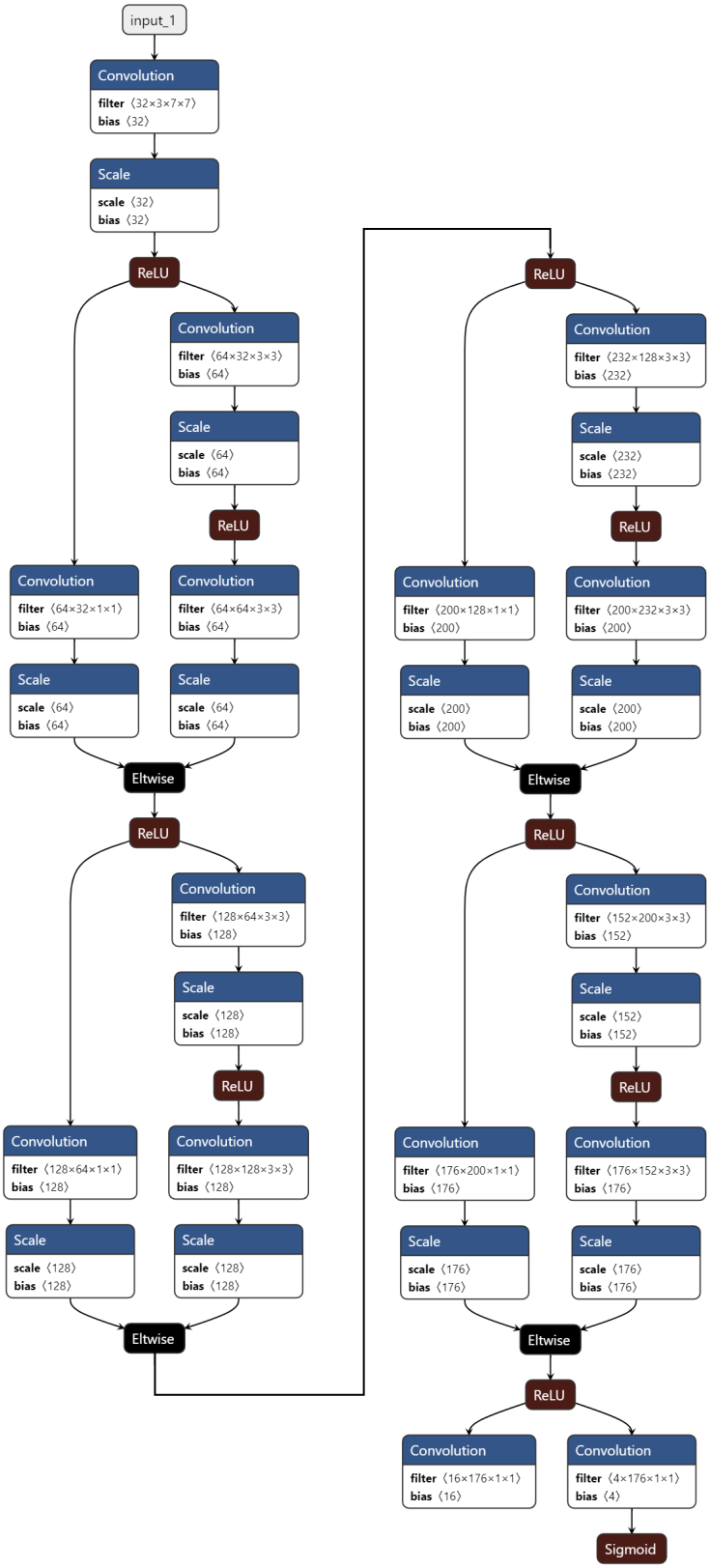


Figura D.1: Estrutura da rede neuronal ResNet-10 utilizada para a deteção de objetos.

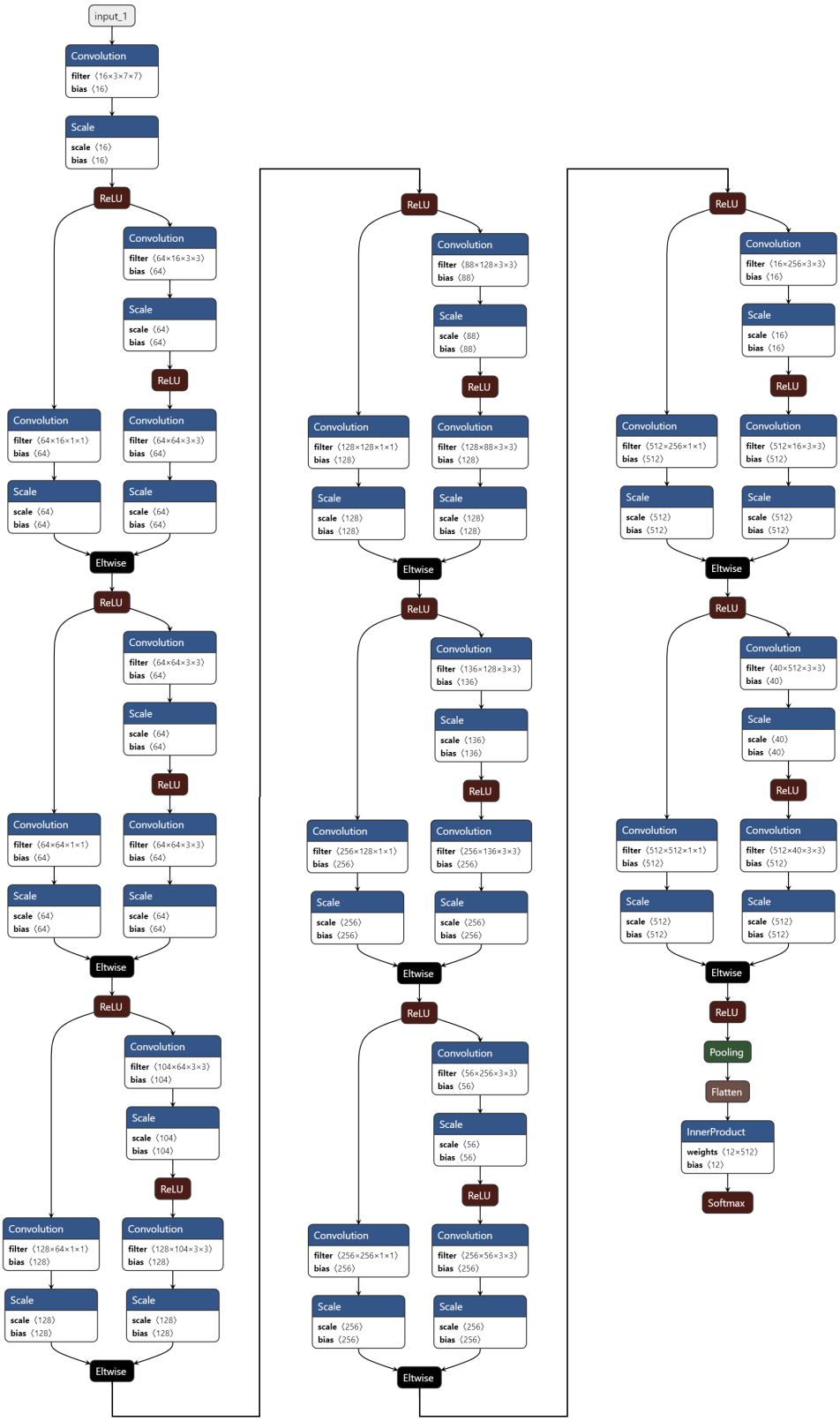


Figura D.2: Estrutura da rede neuronal ResNet-18 utilizada para a classificação de objetos.

Anexo E

Demonstração da Comunicação MQTT entre Servidor e Sistema

Este Anexo contém uma figura que faz a demonstração da comunicação MQTT entre o MQTT *Explorer* (servidor remoto) e o sistema, fazendo uso de toda a lista de tópicos criados para tal. A introdução desta demonstração é feita no final da Seção 5.9.1.

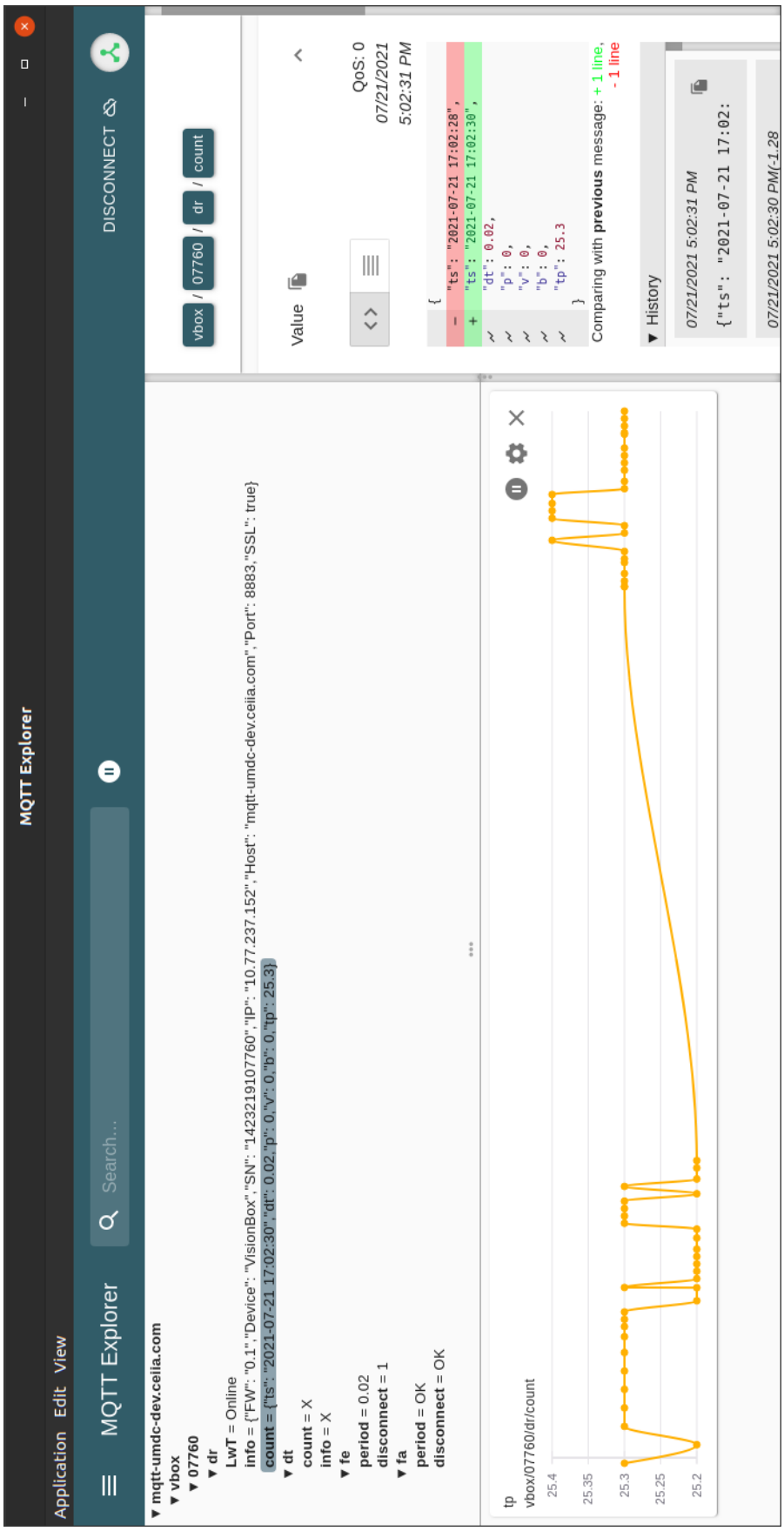


Figura E.1: Demonstração das mensagens trocadas entre o MQTT *Explorer* (servidor remoto) e o sistema.

Anexo F

Demonstração de Dados Reportados pelo Sistema

Este Anexo contém uma figura que faz a demonstração dos dados das últimas 500 mensagens reportadas pelo sistema e recebidas pelo cliente MQTT Explorer da comunicação MQTT entre o MQTT *Explorer* (servidor remoto) e o sistema, sendo estes 6 parâmetros que variam ao longo do tempo, como representado nos gráficos a amarelo na Figura F.1 . A introdução desta demonstração é feita no final da Seção 6.3.

Figura F.1: Representação no MQTT Explorer do conteúdo dos tópicos MQTT para o qual o sistema publica mensagens periodicamente.