Universidade do Minho
Escola de Engenharia

Daniel António Silves Ferreira

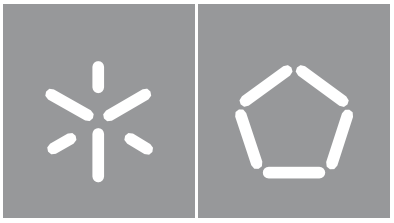
Dispositivo electrónico de monitorização para oceano profundo baseado em imagem

março de 2022

Dispositivo electrónico de monitorização para oceano profundo baseado em imagem

Daniel Ferreira

UMinho | 2022



Universidade do Minho
Escola de Engenharia

Daniel António Silves Ferreira

Dispositivo electrónico de monitorização para oceano profundo baseado em imagem

Dissertação de Mestrado
Engenharia Eletrónica Industrial e Computadores
Instrumentação e Microssistemas Eletrónicos

Trabalho efetuado sob a orientação do
Professor Doutor Luís Miguel Valente Gonçalves
Professor Doutor Sérgio Adriano Fernandes Lopes

março de 2022

Acknowledgements

É com grande satisfação que chego ao final de uma etapa tão importante como a conclusão do Mestrado em Engenharia Electrónica Industrial e Computadores, jornada marcada por grande comprometimento, disciplina, esforço, dedicação e alegrias, mas também desafios, imprevistos, dificuldades, cansaço e inquietações.

Portanto, em primeiro lugar, gostaria de agradecer a Deus, que esteve presente em todos os momentos da caminhada, e foi o refúgio bem presente em qualquer situação. A Ele seja dada toda a honra e toda glória.

Também gostaria de expressar o meu profundo agradecimento aos professores e orientadores, Doutor Luís Miguel Valente Gonçalves e Doutor Sérgio Adriano Fernandes Lopes, pelo apoio incansável durante o desenvolvimento do trabalho, sem os quais seria impossível a realização do mesmo.

Para o meu pai, Daniel Andrade Silves Ferreira e a minha mãe, Edite Maria Leitão Mendes Ferreira, não existirão palavras capazes de demonstrar o quanto fizeram por mim. Deixo o meu eterno agradecimento pela pessoa que me ajudaram a ser e pelo suporte, carinho e amor imensurável que tiveram e sempre demonstraram.

Aos meus irmãos Débora e David, aos meus avós, João e Edite Mendes, à minha tia Joseana e à minha avó Laurinda Ferreira, que são o meu orgulho e alegria, também deixo uma palavra de apreço, amor e gratidão.

Aos tios e tias, primos e primas e à família em geral, que comigo sempre celebraram cada conquista minha, fica o meu reconhecimento e a minha gratidão.

Aos amigos, que sempre me ajudaram a ser melhor em todos os aspectos da vida, com ensinamentos, apoio e uma amizade sincera, manifesto o meu profundo agradecimento.

À Igreja do Nazareno, à comunidade da Igreja dos Irmãos, à Igreja Batista, ao Grupo Bíblico Universitário, ao Clube de Leitura, à Comissão de Residentes de Azurém, a minha gratidão por me terem permitido ser parte sua e participar, aprender e ensinar, e também desenvolver capacidades de liderança, de logística e organização, gestão de tempo e de pessoas e outras tantas qualidades inestimáveis.

A todos os professores, colegas, trabalhadores e funcionários da Universidade do Minho que de alguma forma me ajudaram e mostraram-se valiosos, um muito obrigado.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Resumo

A monitorização das águas profundas tem sido cada vez mais um tópico de extrema relevância para a sustentabilidade do planeta e a sua conservação e várias organizações mundiais têm-se preocupado com sua monitorização e proteção. Sendo a superfície terrestre composta por mais 70% de água, pouco sabemos sobre sua enorme biodiversidade e os efeitos provocados pela atividade humana.

Todo o ecossistema biológico marinho tem grande repercussão na vida do planeta, sendo de extrema importância à existência humana e se a sua preservação for ameaçada as consequências poderão ser catastróficas. Dada a esta enorme importância, é cada vez maior a motivação para monitorizar, conhecer, prever e proteger estes ambientes. As águas profundas (*deep waters*) são o maior dos ecossistemas conhecidos pelo homem e largamente o menos explorado, devido às difíceis condições de acesso.

O desenvolvimento dos sensores de imagem e as suas altas resoluções, assim como a rápida ascensão da capacidade de tratamentos destas mesmas imagens permitem novas formas de efectuar essa monitorização. Para tal, o recurso a imagens e o seu processamento têm tido um papel fulcral na forma como projectamos e adquirimos informação sobre esse ecossistema. Com a tecnologia USB disponível já é possível e fiável a construção de sistemas mais robustos, baratos e eficientes que seja compatível com a evolução do mercado de câmaras de vídeo, sendo expectável que futuros sistemas baseados nesta tecnologia evoluam de igual maneira.

Para que seja possível monitorizar de forma sustentada o *deep-sea* e recolher a maior quantidade de informação, a monitorização baseada em imagens é a técnica pretendida. Para o encapsulamento e *housing* do sistema desenvolvido foram consideradas esferas de vidro Vitrovex. Deste modo, o sistema baseado na família de microcontroladores Cortex-M7 (STM32) foi usado como plataforma de processamento. A aquisição de imagens a partir de uma câmara de vídeo com a interface USB foi feita por intermédio do protocolo UVC. Foi feita a implementação do Software da classe de dispositivos UVC para a plataforma de desenvolvimento, conseguindo um baixo consumo de corrente. Foram implementados comandos de iluminação a partir de LED e o armazenamento da captura de imagem em memória externa. Foram ainda feitas capturas de imagens com resoluções de 320x240, as quais foram armazenadas em cartões de memória microSD com um sistema de ficheiros. O consumo médio de funcionamento obtido foi de 225μA.

palavras-chave: águas profundas, oceanos, câmaras de vídeo, sensores, sustentabilidade, visão por computador, USB, UVC, Arm Cortex-M7

Abstract

Deepwater monitoring has increasingly been a topic of extreme exclusion to the sustainability of the planet and its conservation, and several world organizations have been concerned about its oversight and protection. As the earth's surface is made up of more than 70% of water, we know little about this enormous biodiversity and the effects caused by human activity.

The entire marine biological ecosystem has great repercussions on the life of the planet Earth, being of extreme importance to human existence, and threatening its preservation can lead to catastrophic consequences. Given this enormous motivation, the importance of monitoring, knowing, predicting and protecting these environments is growing. Deep Waters is the largest ecosystem known to man, and largely the least explored, due to the various and difficult conditions of access and exploitation.

The use of images and their processing has played a key role in the way we project and acquire information about these ecosystems. The development of image sensors and their high resolution, as well as the rapid rise in the processing capacity of the capturing devices, reveal new ways to carry out this monitoring. Also, with the emerging technologies it is increasingly possible and reliable to build more robust systems, cheap and efficient.

In order to be able to sustainably monitor *deep-sea* and collect the greatest amount of information, image-based monitoring is the desired technique. For the encapsulation and *housing* of the developed system, Vitrovex glass spheres were considered. Thus, the system based on the Cortex-M7 microcontroller family (STM32) was used as a processing platform. The acquisition of images from a video camera with the USB interface was performed using the UVC protocol. The implementation of the UVC device class Software for the development platform was made, achieving a low current consumption. Lighting commands from LED and the storage of the image capture in external memory were also implemented. Images were captured with resolutions of 320x240, which were stored on microSD memory cards with a file system. The average operating consumption obtained was 225μA.

keywords: deep sea, ocean, cameras, sensors, sustainability, computer vision, USB, UVC, Arm Cortex-M7

Conteúdo

Lista de Figuras

Lista de Tabelas

List of Abbreviations

API	Application Programming Interface.
AUV	Autonomous Underwater Vehicles.
BRUVS	Baited Remoted Underwater Video System.
CDC	Communications Device Class.
CPU	Central Processing Unit.
CRC	Cyclic Redundancy Check.
CSI	Camera Serial Interface.
DMA	Direct Memory Access.
DS	Datasheet.
EOV	Essential Ocean Variables.
FAT	File Allocation Table.
FMC	Flexible Memory Controller.
FS	Full Speed.
GOOS	Global Ocean Observing System.
GPIO	General Purpose Input Output.
HAL	Hardware Abstraction Layer (Software Library).
HD	High Definition.
HID	Human Interface Device.
HS	High Speed.
IDE	Integrated Development Environment.
IFSOO	Integrated Framework for Sustained Ocean Observing.
LED	Light Emitting Diode.
LS	Low Speed.

MCU	Microcontroller Unit.
MJPEG	Motion-Joint Photographic Experts Group.
MMC	MultiMediaCard.
MP	Mega Pixels.
MPA	Marine Protected Areas.
OEM	Original Equipment Manufacturer.
PC	Personal Computer.
PCC	Program Consumption Calculator.
PHY	Physical Layer.
PID	Product ID.
RAM	Random Access Memory.
RM	Reference Manual.
ROV	Remote Operated Vehicles.
RTC	Real Time Clock.
SD	Secure Digital.
SDIO	Secure Digital Input Output.
SDMMC	Secure Digital and MultiMediaCard.
SOF	Start of Frame.
SRAM	Static Random Access Memory.
UAC	USB Audio Class.
UART	Universal Asynchronous Receiver-Transmitter.
UAV	Unmanned Aerial Vehicle.
ULPI	UTMI + Low Pin Interface.
UM	User Manual.
UML	Unified Modeling Language.
UNESCO	United Nations Educational, Scientific and Cultural Organization.
USB	Universal Serial Bus.
UVC	USB Video Class.

VC	Video Control.
VS	Video Streamming.

Capítulo 1: Introdução

1.1 Motivação

Os oceanos compõem cerca de 70% da superfície terrestre e são de vital importância para a regularização do clima e das condições meteorológicas no planeta, o equilíbrio da temperatura, a biodiversidade de fauna e flora e os ciclos hidrológicos. Os oceanos têm ainda uma grande influência no papel social, político e económico, manifestada em actividades comerciais e industriais como a pesca, transporte, energia, turismo, mineração e outras.

O oceano profundo (*deep sea*) é caracterizado por ser o maior *habitat* do planeta, mas ainda com uma taxa de exploração bastante baixa. Este ecossistema consiste na zona oceânica que oscila entre o fim da plataforma continental (aproximadamente 200 m) até aos 10 900m, o ponto mais profundo conhecido [10].

No entanto, a importância desse ecossistema é enorme para a vida humana e a sua preservação e para a geração de vários recursos entre os quais a potencial mineração de outros recursos tais como depósitos maciços de sulfato no fundo do mar (cobre, zinco, prata, ouro, manganésio, cobalto, níquel) entre outros variados minérios[11].

A degradação dos oceanos, a superexploração da pesca, o declínio da biodiversidade, os recifes de coral ameaçados, a acidificação dos oceanos levantam preocupações legítimas, o que nos leva a questionar o estado das águas profundas [1].

Torna-se importante quantificar o impacto humano nesse ecossistema, à medida que o fenómeno de ecotoxicidade causado por indústrias petrolíferas e de mineração, pescas, exploração de hidrocarbonetos microplásticos, desperdícios e despejos assolam os oceanos. Um exemplo de contaminação do oceano profundo pode ser o caso do acidente de *Deepwater Horizon* no golfo do México, em que houve derrame de óleo a 1500 m de profundidade [12].

Não obstante a grande relevância e importância da monitorização, as condições encontradas no oceano profundo são bastante adversas, sendo que 88% da área oceânica seja para profundidades maiores a 1000 m. Essas condições envolvem pressões elevadas, temperaturas baixas, impossibilidade de transmissão de dados, corrosividade, inexistência de iluminação e dificuldades de isolamento eléctrico [13].

A monitorização biológica do oceano profundo requer variáveis físicas tais como a temperatura, a

salinidade, o pH, entre outras. A monitorização biológica exige sistemas mais complexos, dada a variedade biológica dos oceanos (fitoplâncton, zooplâncton, larvas, bactérias, peixes, algas, etc). Para esta monitorização biológica *in-situ*, o recurso a imagens é a técnica mais adequada, dada a diversidade de informação que é possível recolher ao longo de um intervalo de tempo mais amplo [14].

Os sistemas comerciais de monitorização com recurso a imagens tendem a ser caros, de grandes dimensões e pouco adaptados às necessidades de aquisição particulares.

A tecnologia permitiu que o desenvolvimento de sistemas por visão de computador e de processamento de imagens tivessem avanços importantes tais como maiores resoluções de imagens ou frequências de oscilador mais rápidas, tornando-os assim cada vez mais adequadas para um grande número de aplicações [7] [5].

No caso deste trabalho em particular, este tipo de sistemas pode fazer a colecta de dados sem que danifique o meio em que se faça a aquisição. Os progressos nos designs, nas câmaras de vídeo digitais com cada vez maiores resoluções, técnicas de armazenamento de imagens mais optimizadas, assim como encapsulamentos - *housing* - para pressões elevadas, e o baixo consumo dos sistemas permitem recolher dados durante períodos de tempo mais alargados.

1.2 Objectivos

O projecto desta dissertação tem como objectivo principal o desenvolvimento de um sistema de monitorização de águas profundas, com base na recolha de imagens a uma taxa de amostragem fixa configurável. Pretende-se que tenha uma alta autonomia, assim como um sistema embebido fiável e de baixo custo em comparação com as soluções de mercado existentes. Para tal propões-se a utilizar uma câmara de vídeo OEM de alta resolução, com um protocolo de comunicação standard da indústria. A capacidade de submersão será feita através de esferas de Vitrovex, capazes de suportar as condições adversas das águas profundas. Portanto, é também necessário desenvolver o sistema de aquisição de imagens com reduzidas dimensões, de modo a que se possa usar as esferas Vitrovex.

O sistema de landing não será o foco do desenvolvimento, sendo que se assume que haverá uma equipa multidisciplinar para esta tarefa.

- Revisão da Literatura relevante e soluções de mercado
- Escolha da plataforma de processamento
- Implementação do Host para comunicação com Device USB

- Implementação da classe de dispositivos USB Video Class (UVC)
- Implementação do Armazenamento de dados com um sistema de ficheiros
- Implementação da Aplicação de Monitorização com a captura de imagens
- Testes funcionais e não-funcionais do protótipo do sistema de aquisição
- Testar o sistema em ambiente de laboratórios
- Quantificar a energia consumida

1.3 Estrutura

A presente dissertação está organizada em seis capítulos distintos.

O capítulo 1 tem uma breve introdução geral ao tema do trabalho. Dele também constam a relevância da questão e a motivação deste projecto bem como os objectivos a alcançar e a estrutura da dissertação.

O capítulo 2 aborda de forma sucinta a Revisão da Literatura relevante para a compreensão do problema, a metodologia da monitorização e dos métodos tradicionais de monitorização subaquática, e, por fim, a monitorização com recurso a imagens.

No capítulo 3 são apresentadas as especificações do sistema. Primeiramente, é referida a plataforma de processamento na qual será desenvolvida o projecto, sendo que a STM32 é o caso de análise e os modos de gestão de energia apresentados. Posteriormente, é feita uma breve revisão das interfaces de câmaras de vídeo usuais em aplicações similares. É então descrita de forma sucinta o funcionamento da interface USB com conceitos relevantes para o restante do trabalho. Depois, é feita a análise das capacidades da câmara de vídeo e apresentada a ligação de hardware. Aborda-se ainda neste capítulo, o armazenamento de dados em cartão SD, sendo para tal especificados o sistema de ficheiros e o periférico SDMMC. Finalmente, é apresentada a arquitectura do sistema.

O capítulo 4 descreve a implementação do sistema de acordo com os pressupostos e requisitos determinados. Primeiramente, é abordada a classe de dispositivos UVC, com a abordagem escolhida e a implementação da mesma. Posteriormente, são apresentados os módulos e o protótipo do sistema. Por fim, é feita a implementação da aplicação de captura de imagens.

No capítulo 5 são apresentados os testes e resultados do protótipo desenvolvido, começando pelos resultados funcionais do sistema, seguidos dos não funcionais.

Por último, o capítulo 6 contém as conclusões assim como as possíveis perspectivas de trabalho futuro.

Capítulo 2: Estado da Arte

Neste capítulo, é feita a revisão da literatura dos conceitos científicos relevantes para o desenvolvimento do tema desta dissertação. Um dos grandes objectivos mundiais é o da sustentabilidade e da proteção dos oceanos, tais como definidos pelas Nações Unidas em "Década das Nações Unidas da Ciência dos Oceanos para o Desenvolvimento Sustentável (2021-2030)" e pelo Desenvolvimento Sustentável das Nações Unidas segundo o objectivo 14, em que é estipulada estabelecer Áreas Marinhas Protegidas - Marine Protected Areas (MPA). Para tal, é necessário conhecer os comportamentos e as dinâmicas marítimas subsequentes, o que requer a monitorização desse enorme ecossistema [15].

Neste âmbito, no contexto da monitorização biológica em mar profundo, são apresentados os métodos tradicionais de monitorização subaquática e a monitorização com recurso a imagens.

No primeiro subcapítulo, são revistas as variáveis oceânicas essenciais e considerações sobre o oceano profundo. No segundo subcapítulo, são apresentados variados meios de monitorização disponíveis e as suas diferenças. Por fim, no terceiro subcapítulo, são apresentadas as câmaras de vídeo Original Equipment Manufacturer (OEM) e é definido o conceito de *underwater-housing* e monitorizações *in-situ*.

2.1 Monitorização

A monitorização biológica subaquática é feita em diferentes tipos de ambientes tais como em rios, praias fluviais, praias costeiras ou mar profundo. Esta monitorização tem aspectos e finalidades diferentes no que diz respeito à informação recolhida e, mediante o caso da aplicação, uma técnica ou outra poderá ser a mais adequada. De ressaltar, que existem diferenças específicas em cada um destes ambientes, que deverão ser levadas em conta na concepção e escolha da tecnologia. Portanto, a monitorização biológica de um rio, será diferente da monitorização do mar, e esta da do oceano profundo, por exemplo. Mas, de forma geral, a monitorização biológica subaquática, defronta-se com condições adversas. Em termos de acessos, por serem muitas vezes inacessíveis sem meios adaptados, pontos como a iluminação e a impermeabilidade são de primeira ordem. A robustez, a pressão e a temperatura a altas profundidades exigem que os equipamentos e técnicas usados sejam apropriadas. E ainda, em termos de limitação de tecnologia, as condições inerentes do meio limitam o tipo de tecnologia passível de ser utilizada. No entanto, é muitas vezes necessário que se conheça e se estude esse ecossistema, de modo a avaliar o impacto antropogénico e assim criar medidas de proteção e preservação.

O relatório do grupo de trabalho Integrated Framework for Sustained Ocean Observing (IFSOO) [1] relembra-nos da importância dos oceanos na biosfera terrestre. O seu contributo alimentar e de recursos, a sua influência directa sobre a economia, a segurança e as formas alternativas de gerar energia são de vital importância para a humanidade. Ainda em [1], vemos que negligenciar o compromisso de preservação dos oceanos, com a degradação dos habitats costeiros, a poluição, o recuo das camadas de gelo e o aumento do nível do mar ameaçam mais de 40% da população mundial, e é urgente um método sistémico para a colecta de informações oceânicas em escalas local, nacional, regional e global.

É, portanto, fundamental que se garanta a sustentabilidade desses ecossistemas, posto que a sua importância é conhecida para o funcionamento do ecossistema global, tal como apresentada na figura 2.1.

2.1.1 Variáveis Oceânicas Essenciais

O progama Global Ocean Observing System (GOOS) executado pela Comissão Intergovernamental Oceanográfica da United Nations Educational, Scientific and Cultural Organization (UNESCO) definiu algumas variáveis oceânicas consideradas essenciais, para a observação dos oceanos. Com o objectivo de fomentar políticas para gestão e mitigação dos efeitos das mudanças globais dos oceanos, estas variáveis foram usadas para a orientação das decisões, como por exemplo, fornecer oportunidades aos países

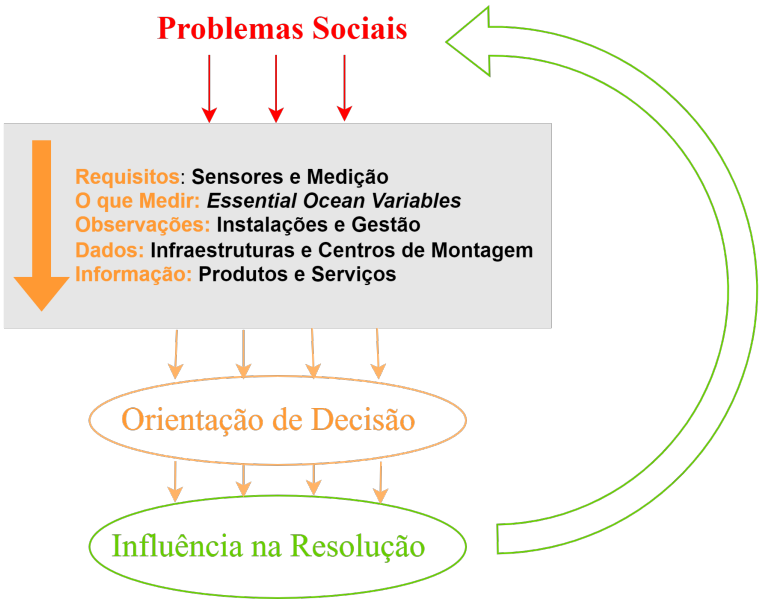


Figura 2.1: Framework do Grupo de Trabalho IFSOO
adaptado de [1]

em desenvolvimento de modo a identificarem as suas necessidades de participação e de apoio global, melhorando em geral as economias e o bem-estar da sociedade em todo o mundo [16].

Entre essas variáveis pode-se apontar, como exemplos de EOV, a temperatura oceânica, o carbono oceânico, a biomassa de zooplâncton, a biomassa de fitoplâncton, a salinidade, o pH, os nutrientes, entre muitos outros. A partir dessas métricas, será possível um estudo da condição de um ecossistema.

2.1.2 Oceano Profundo

O oceano profundo é considerado a partir de profundidades maiores do que 2000 m, sendo a maior profundidade conhecida com mais de 10 984 metros. É o maior habitat no planeta e é muito pouco explorado, constituindo a maior parte da área dos oceanos [17].

As características no oceano profundo como temperaturas extremamente mais baixas [1, 4°C], pressões atmosféricas mais elevadas [1,100 atm] e luminosidade inexistente fazem com que os acessos sejam ainda mais restritos e difíceis.

Como é possível observar na figura 2.2, as áreas científicas de monitorização assim como os impactos e benefícios dos dados obtidos são abrangentes.

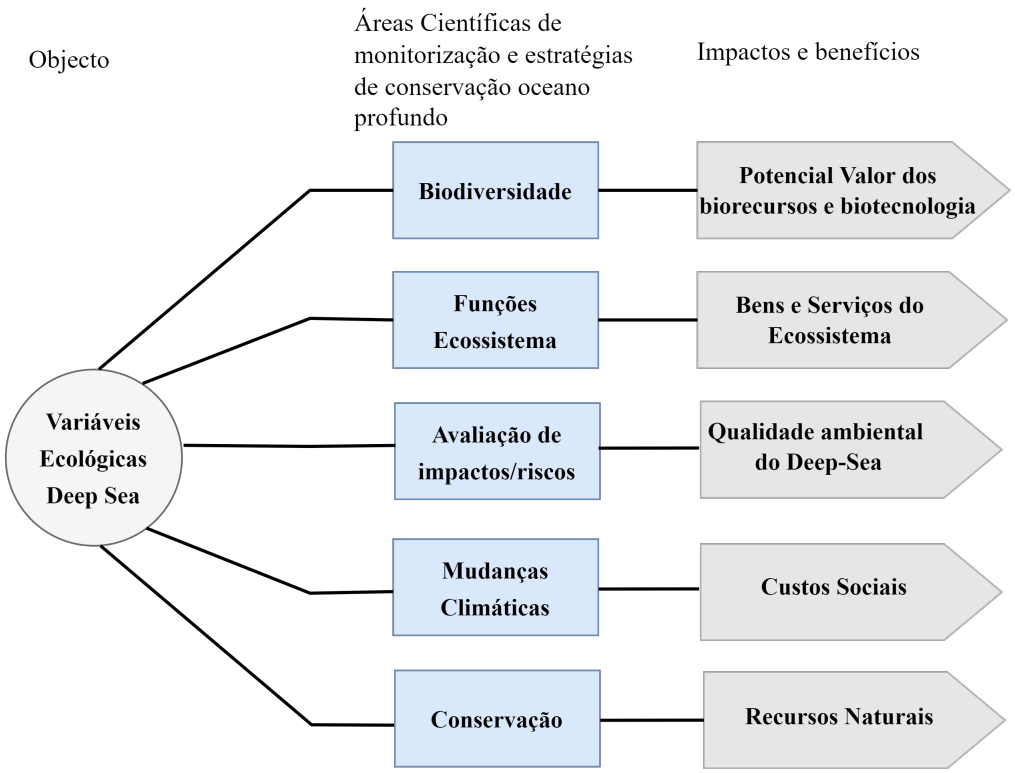


Figura 2.2: Estratégia de monitorização oceano profundo
adaptado de [2]

2.2 Métodos Tradicionais de Monitorização Subaquática

Os métodos e técnicas de monitorização subaquática poderão ter várias métricas para a sua classificação. O artigo [2] classifica os sistemas de monitorização em: Sistemas móveis, Sistemas fixos e experimentais *in situ*, Observatórios, Tecnologia transportada em Animais e Análises laboratoriais. Apenas no contexto das análises laboratoriais, não é predominantemente feita qualquer tipo de mapeamento ou monitorização à base de imagens recolhidas. Estes sistemas são equipados de meios humanos e tecnológicos necessários para que seja possível obter o maior número de dados possíveis. Cada um poderá ter uso diversificado para se obter conhecimento específico sobre os valores bióticos ou abióticos, da biologia e da biodiversidade, do funcionamento do ecossistema, e dos serviços ecológicos. E consoante o interesse de estudo, um dos métodos poderá ser mais adequado para a informação específica que se poderá obter.

Em [2] é apresentado um diagrama conceptual que interliga as diversas áreas de estudo, a tecnologia envolvida e o conhecimento adquirido e paralelamente eo interesse de estudo, um dos métodos poderá ser mais adequadom [18], que resume as actuais e futuras plataformas de medição oceânica, assim

como os sensores usados e as maiores limitações de cada plataforma. Segundo estas classificações, um sistema de monitorização *in-situ* consegue monitorizar as interações biológicas, a reprodução, taxas de crescimento, as respostas biológicas, os fatores bióticos e abióticos, o mapeamento do habitat, entre outras. As principais limitações deste tipo de sistema prendem-se com o facto da localização dos mesmos, devendo portanto ser colocadas em lugares de interesses estratégicos tais como as Marine Protected Areas (MPA).

A monitorização biológica com recurso a imagens, através de sensores de imagem ou câmaras, em formatos de vídeo ou imagem, tem uma vantagem muito proeminente em relação a outras formas mais tradicionais de medição. O número muito elevado de informações captadas por sistemas com base nestas tecnologias e em paralelo com a grande evolução da capacidade de processamento digital de sinal fazem com que este seja um método preferencial, no que diz respeito à monitorização marítima. Ao longo dos anos a monitorização biológica com recurso a imagens tem tido importantes evoluções. Desde a primeira fotografia subaquática e da filmagem feitas nos finais do século XIX e inícios do século XX, várias evoluções tecnológicas assinalaram marcos importantes na monitorização subaquática a partir de imagens [3].

Os veículos submarinos operados remotamente (ROV) foram comercializados pela primeira vez no final década de 80 e têm desempenhado papéis importantes em contexto de exploração e de monitorização. São veículos ligados por cabo a um operador que os dirige de forma remota. Os ROV têm também acompanhado a evolução das tecnologias de imagem, tendo cada vez mais, câmaras com melhores resoluções. Estes veículos são também muitas vezes equipados com sensores de condutividade, de temperatura, de profundidade, acústicos e ainda outros.

As Autonomous Underwater Vehicles são veículos subaquáticos autónomos. Os AUVs são muito similares aos ROVs vistos anteriormente, com a importante diferença de não serem operados, mas sim automáticos. No entanto, possuem importantes vantagens relativamente ao ROV dado que não são restringidos por cabos, têm uma maior área de actuação. Mas, por outro lado, têm limitações pelo que são precisos cuidados adicionais.

Unmanned Aerial Vehicle (UAV) são aeronaves pilotadas remotamente, capazes de sobrevoar grandes áreas de observação, mas apenas capazes da monitorização da superfície marinha. As condições de visibilidade e a legislação acerca das alturas permitidas também podem ser vistas como uma das limitações da monitorização baseadas neste método.

O Sistema de Vídeo Subaquático Remoto com Isca (BRUVS), é um sistema estático, ao qual é atraída

Figura 2.3: H300V do Grupo ECA

Figura 2.4: L3Harris Technologies Iver3

uma espécie marinha e são obtidas imagens dessa espécie. O sistema é usado para se estudar os comportamentos e a abundância da fauna marítima e também como uma ferramenta de exploração.

O *Remote Underwater Lander* ou Módulo de pouso remoto subaquático é considerado como um instrumento de medição estático e fixo, no qual o sistema é posicionado *in-situ* para um determinado propósito de monitorização. É muito usado nas Marine Protected Areas (MPA) - Áreas Marinhas Protegidas, que normalmente têm um ecossistema muito variado, pelo que a qualidade das imagens torna-se assim extremamente relevante.

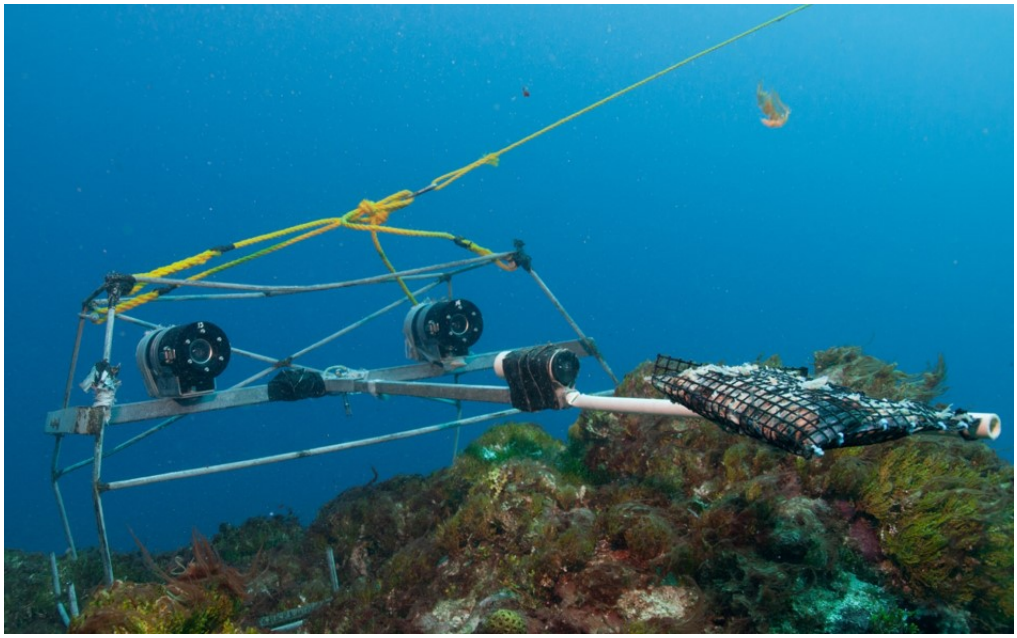


Figura 2.5: Benthic BRUVs

2.3 Monitorização com recurso a Imagens

O impacto antropogénico nos oceanos é normalmente difícil de quantificar, medindo apenas alguns parâmetros físicos e biológicos. Para uma melhor compreensão dos impactos no ecossistema como um todo, é muitas vezes necessária uma grande quantidade de informação, o que normalmente, traduz-se, em sistemas mais complexos. Dito isto, os métodos de monitorização com recurso a imagens, juntamente com um pós-processamento digital robusto são capazes de fornecer informações fidedignas de variáveis que se pretendam medir. As variáveis poderão ser físicas assim como biológicas e permitem assim qualificar da melhor forma, os ecossistemas subaquáticos. A grande evolução da capacidade de processamento de imagens tem tornado possível que se aplique métodos cada vez mais complexos e céleres na descodificação dos dados armazenados por aplicações de vídeo e/ou imagens. A evolução da tecnologia tem feito cada vez mais com que sistemas baseados em imagem sejam escolhas preferenciais devido às melhorias na qualidade e resolução, aos longos períodos de aquisição, ao decrescimento gradual do custo e ao aumento da capacidade de armazenamento de dados [19].

A alta resolução das câmaras actuais permite com que haja uma grande preferência por este método, já que assim é possível não só fazer capturas de organismos grandes, mas também de pequenos organismos. Podemos ver, por exemplo em [20], em que o objecto de estudo é o zooplâncton, que é capaz de monitorizar organismos de dimensões inferiores a 1mm. Um outro exemplo, [21], o objecto do estudo são peixes e, neste caso, os organismos monitorizados têm dimensões relativamente maio-

Figura 2.6: Imagem captada em monitorização subaquática
Extraída de [3]

res. Outra grande vantagem são os actuais tamanhos reduzidos destes sistemas, e a possibilidade de ter *underwater-housing*, também de reduzidas dimensões e com a capacidade de suportar as condições adversas, tais como pressão e temperatura em águas profundas. No entanto, ainda, persistem algumas limitações nestes sistemas. Como se pode ver em [3], a tecnologia, os efeitos do dispositivo, a visibilidade, os atractivos e a pluma olfactiva, a análise de vídeo e/ou foto, a aquisição e o design experimental, são ainda limitações e desvantagens associadas a este tipo de técnicas.

2.3.1 Câmaras de vídeo Original Equipment Manufacturer (OEM)

As câmaras de vídeo OEM são componentes fabricadas para serem incorporadas em um determinado equipamento ou para um determinado desenvolvimento tecnológico. Visto que possuem normalmente uma interface de fácil integração e com protocolos standard, torna-se mais conveniente utilizar esses equipamentos para o desenvolvimento de novos produtos e técnicas. A facilidade de integração, a abrangência do mercado e a qualidade dessas câmaras são algumas das principais vantagens. Estas câmaras são amplamente usadas em ambientes como o automobilístico, o industrial, a saúde e os cuidados especializados, o marítimo, entre vários outros [19].

As principais características das câmaras de vídeo existentes no mercado prendem-se com o tipo de interface, o sensor de imagem, a resolução e a *frame rate*. Os tipos de interface mais habituais para aplicações do tipo embebido são Camera Serial Interface e a USB.

2.3.2 Underwater-housing

O underwater-housing ou cápsula é todo o sistema físico e mecânico responsável para suportar as condições adversas que são encontradas em oceano profundo. As pressões elevadas a altas profundidades, as condições de iluminação escassas, o ambiente em grande medida desconhecido, as correntes marítimas e as temperaturas muito baixas são algumas das condições a se ter em conta, quanto ao dimensionamento da estrutura [22].

(a) Underwater-Housing em Vidro I

(b) Underwater-Housing em Vidro II

Figura 2.7: Underwater-Housing em Vidro

Na figura 2.7a podemos observar que todo o sistema de aquisição está presente dentro do invólucro de vidro que serve como *underwater-housing*. As especificações do modelo Vitrovex da *Nautilus Marine Service* [23] garante profundidades de até 12 000m, sendo assim adequadas para aplicações em oceano profundo.

2.3.3 Sistemas de Monitorização in situ

Nesta subsecção faz-se a revisão da literatura, de forma mais específica, de cinco artigos sobre a construção de um sistema de monitorização subaquática *in situ* com tecnologias diferentes. O foco será primariamente o sistema de aquisição de imagens, as características e vantagens de cada.

DEEPi

Em [4] foi construído um sistema de monitorização *deep-sea* com o *single-board computer* Raspberry Pi zero baseado no processador Arm Broadcom BCM2835. A câmara utilizada foi o módulo Raspberry module V2, com resolução de 8 MP 1080p30 e suporte para *Still Images*. Uma biblioteca foi desenvolvida em Python em *open-source* customizada para a configuração da transmissão de streaming do sistema. Com o sistema operativo de desenvolvimento Linux, o sistema tem a possibilidade de conectar-se a redes wifi, conferindo-lhe assim maiores potencialidades. Testes Laboratoriais determinaram que a capacidade de mergulho do sistema seria até aos 5500 m e foram realizadas aquisições a uma profundidade máxima de 1096 m.

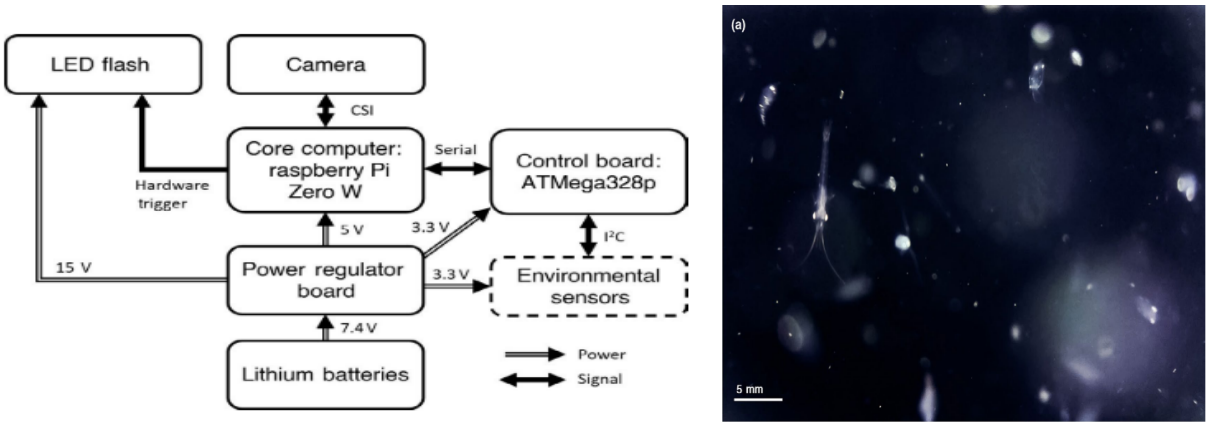
(a) Raspberry pi zero e módulo câmera (b) Captura imagem

Figura 2.8: Sistema Monitorização DEEPi
Imagens retiradas de [4]

O sistema de *housing* foi construído a partir de uma impressora 3D e é um dos mais compactos sistemas de câmara de vídeo HD *deep-sea*. O sistema de iluminação foi baseada em dois LEDs e as baterias usadas foram de Li-ion, conseguindo uma autonomia de 6 horas de vídeo contínuas, com um consumo médio no intervalo de 0.22 a 0.4 A. O custo do sistema foi inferior a 100$, excluindo as baterias de Li-ion. O protótipo do sistema e uma captura são apresentadas na figura 2.8.

IPAx

A aplicação descrita em [5] não é propriamente para *deep-sea*, mas é também um sistema subaquático para aquisição de imagens baseado igualmente no Raspberry Pi zero, modelo W, juntamente com uma placa de controlo da atmega328p. A aplicação faz a medição de níveis de plâncton, especificamente zooplâncton, a partir de imagens recolhidas.



(a) Diagrama do Sistema (b) Captura imagem

Figura 2.9: Sistema Monitorização
Imagens retiradas de [5]

A profundidade alcançada foi de 100 m e a aquisição com resolução de 1640x1232 a 30 fps. A câmara de vídeo utilizada foi um módulo compatível com a Raspberry. A autonomia conseguida foram 8 horas de vídeo com baterias de lítio de 51.8 Wh, e o armazenamento de dos em cartão de memória SD. O custo do sistema foi inferior a 450$.

Omni-Cam

Em [6] é implementado um sistema de monitorização para *deep-sea*, baseado em seis câmaras de vídeo *Allied Vision Technologies* modelo GX1910 GigE, controladas por sete PCs embutidos, via EPIA-P720. As resoluções das câmaras são de 1920x1080 e 60 fps. A interface dos *single-board computer* EPIA com as câmaras é feita com o protocolo gigabit Ethernet.

A autonomia conseguida por este sistema foi de 4 horas, sendo que a voltagem das baterias é controlada e monitorizada por uma unidade de gestão de energia.

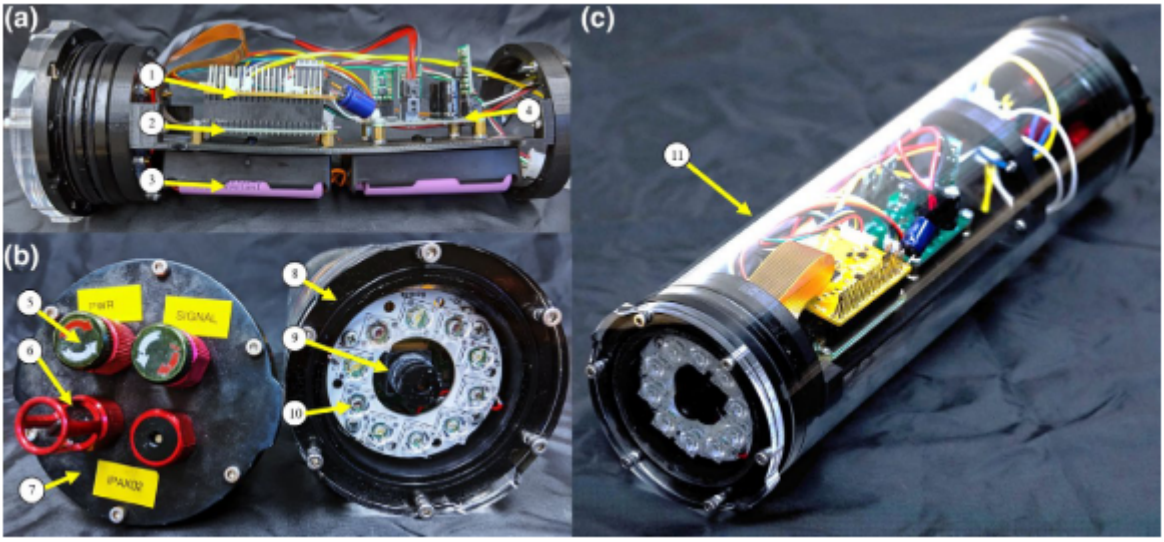
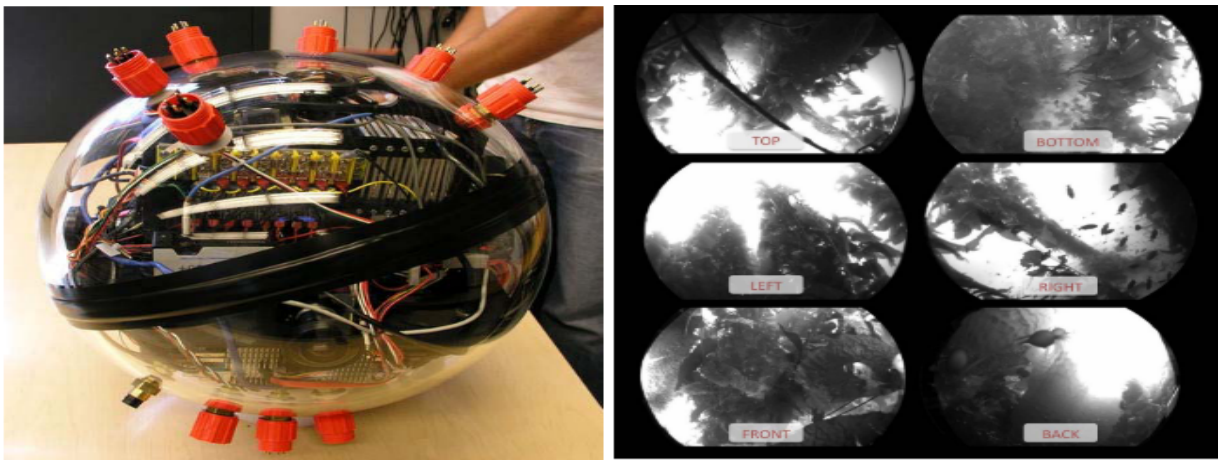


Figura 2.10: Sistema de Monitorização Ipax
retirado de [5]



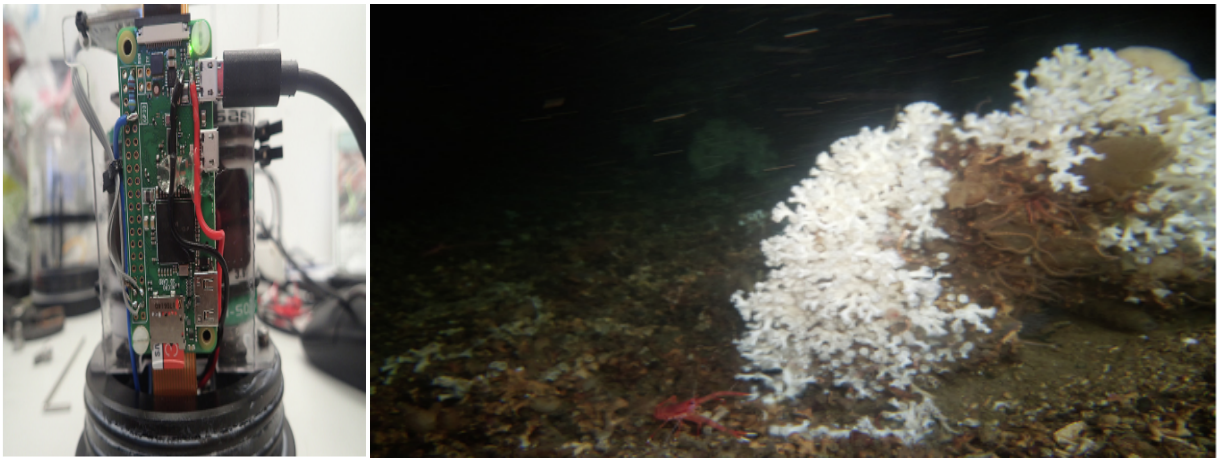
(a) Sistema (b) Capturas imagens

Figura 2.11: Sistema Monitorização
Imagens retiradas de [6]

PlasPI

Em [7], à semelhança do DeepI e do IPAX é também implementado com uma Raspberry Pi zero W e com o módulo câmara V2, este com uma resolução de 3280x2464. A comunicação entre o *single-board computer* e o módulo da câmara de vídeo é feita com o protocolo CSI e o sistema operativo é o Raspbian. O armazenamento de energia é feito com baterias de lítio.

A profundidade máxima deste sistema é de 150 m e o custo total foi inferior a 200 €, com o *underwater housing* em plástico.



(a) Sistema (b) Capturas imagens

Figura 2.12: Sistema Monitorização
Imagens retiradas de [7]

Sony2021

Já o artigo [9] faz uma abordagem diferente. Apesar de também utilizar o *single-board computer* Raspberry Pi, utiliza o modelo A+ e a captura de imagens é feita com uma máquina fotográfica profissional da Sony, modelo A7S II E-mount com sensor de frame. A resolução especificada para esta aplicação foi de uma captura de 4240x2380. É também usada uma Witty Pi 2 como RTC e placa de gestão de energia.

A comunicação é feita através da interface USB com auxílio da biblioteca gphoto2, um conjunto de funções desenvolvidas para variadas câmaras de vídeo de modo a serem utilizadas em diversos sistemas operativos. Assim a biblioteca é responsável por toda a comunicação USB, tais como aceder a câmara, configurações e aquisição de imagens. O consumo de energia foi estipulado em 0.42 A em funcionamento, com 0.02 Wh em média e 0.28 Wh por cada ciclo de captura/sleep. Foram ainda escolhidas baterias alcalinas de 720 Wh (48 células-d) que deram para fazer ensaios de 16 horas/dia durante 14 dias.

Uma das grandes desvantagens deste método é o custo, visto que o preço da câmara é extremamente superior aos outros vistos anteriormente.

A tabela 2.1 apresenta os cinco principais trabalhos citados nesta subsecção, sumarizando as mais relevantes.

É possível observar a predominância do uso de *single-board computer*, com especial destaque ao Raspberry Pi, que aparece em quatro dos cinco sistemas observados. A preferência por plataformas que

Artigo	µprocessador	DeepSea	Consumo	Câmara	Sensor Imagem	Resolução	Baterias	RAM	Interface
DeepI	Raspberry Pi Zero	Sim	0.22-0.4A	Raspberry Camera Module V2	Sony IMX219PQ	8MP 3280x2464 1080p30	Li-ion	512 MB	CSI
IPAX	Raspberry Pi zero W Atmega328p	Não	–	Raspberry Compatible Camera	Sony IMX219	2MP 1640x1232 30fps	Lithium 51.8Wh	512 MB	CSI
Omni-Cam	VIA EPIA-P720 embedded PC	Sim	–	GX1910 GigE	ON Semi KAI-02150	2.1MP 1920x1080 60fps	Li-ion	2 GB	gigabit Ethernet
PlasPi	Raspberry Pi Zero	Não	–	Raspberry Camera Module V2	Sony IMX219	8MP 3280x2464 1080p30	Lithium	512 MB	CSI
Sony2021*	Raspberry Pi A+	Não	0.42A	Sony a7S E-mount	CMOS Exmor full-frame	12.2MP 4240x2380	48 D-cell Alcaline 720Wh	512 MB	USB

Tabela 2.1: Comparação das características Diz respeito a [9]

suportam um sistema operativo é a grande norma nestes tipo de sistema, com periféricos dedicados para a receção de dados do tipo de vídeo. Para a implementação do sistema proposto para a plataforma STM32 F767, o software é desenvolvido *bare-metal* para alta performance. Entretanto, o armazenamento de dados, através de uma memória externa SD, a temporização de capturas, com um módulo RTC e a interface com uma câmara de vídeo por USB, são finalidades em comum.

Capítulo 3: Especificações de Sistema

Este capítulo tem como secções os pressupostos pelos quais será desenvolvido o sistema. As especificações e os requisitos serão delineados e todo o desenvolvimento da aplicação será baseado nestas secções. Na primeira secção aborda-se a plataforma de processamento. Na segunda secção, são apresentadas as interfaces de comunicação das câmaras de vídeo mais usuais para o tipo de aplicação em causa. Na terceira secção, são abordada a interface USB e a classe de dispositivos de vídeo, com as particularidades relevantes para a implementação. Na quarta secção, é apresentada a câmara de vídeo. Na secção cinco, é abordado o armazenamento de dados do sistema. Por fim, na secção seis, apresenta-se a arquitectura do sistema de aquisição de imagens.

3.1 Plataforma de Processamento

O sistema tem como requisitos funcionais a aquisição e armazenamento de imagens com um elevado desempenho, baixo consumo e elevada autonomia. Para tal a plataforma de processamento terá de estabelecer a taxa de amostragem desejada e o armazenamento destas imagens. Um microcontrolador que possa ter as opções de grande fiabilidade e de Low-Power torna-se assim necessário. Portanto a principal função do microcontrolador é a de fazer a comunicação com uma câmara OEM.

3.1.1 STM32

Pela particularidade de aquisição do tipo de dados pelo MCU, é necessário garantir a fiabilidade e eficiência das operações efectuadas. A familiaridade e os conhecimentos sobre a plataforma de desenvolvimento são o critério fundamental da escolha do microcontrolador STM32F767. Trata-se de um processador de 32-bit de elevado desempenho com um núcleo M7. Como características deste microcontrolador, temos a sua frequência de cristal de 216 MHz.

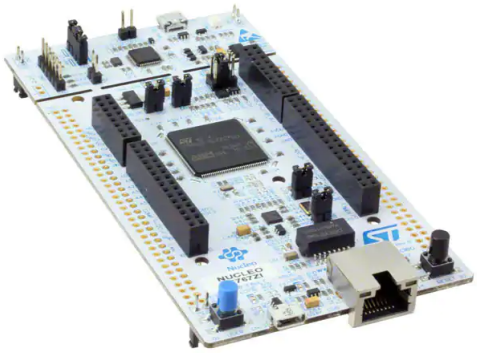


Figura 3.1: NUCLEO-F767ZI

A arquitectura do microcontrolador é baseada no ARM Cortex-M7 32-bit com FPU, com memórias Flash de 2 Mbytes e SRAM de 512 Kbytes. A placa de desenvolvimento STM32 Nucleo-144 - NUCLEO-F767ZI - foi usada como plataforma de desenvolvimento do sistema.

3.1.2 Modo de baixo consumo

Dada a aplicação em causa, o nosso sistema só terá que estar a funcionar durante um breve período de tempo, de modo a fazer a captura da imagem. Com um período de aquisição estimado em algumas horas, é expectável que seja possível optimizar o consumo de energia para uma autonomia muito elevada. De facto, o consumo de energia deverá ser primariamente do sistema da câmara de vídeo, nomeadamente a interface USB, o protocolo UVC e a câmara de vídeo OEM, o sistema de armazenamento de dados e o sistema de iluminação. Para além do pouco tempo estimado de funcionamento do sistema, são implementadas soluções de standby e de modos de consumo *low-power*. A tabela 3.1 resume os modos aqui descritos.

Modo	**Entrada**	**Wakeup**	**Clocks Domínio 1.2 V**	**Clocks Domínio VDD**	**Regulador de tensão**
Sleep	WFI	Qualquer Interupção	CPU desligado	_	Ligado
	WFE	Evento Wakeup			
Stop		Qualquer EXTI	Todos desligados	HSI e HSE desligado	Regulador Principal ou Low Power
Standby		Pino WKUP, RTC, IWDG e reset externo NRST			Desligado

Tabela 3.1: Modos Low Power

O MCU da ARM Cortex-M7 possuí dois modos para o processador, o modo "*Run*"e o modo "*Low Power*". No caso específico da STM32F767, existem três principais níveis de consumo para *Low Power*, cada um com características de funcionamento diferentes. Os modos são denominados por *Sleep Mode*, *Stop Mode* e *Standby Mode*. No modo de *Sleep* o clock do CPU é desligado, embora os periféricos continuem a funcionar. O consumo aumenta à medida que hajam mais periféricos a ser usados, com maiores frequências de clock desses periféricos. O modo *Stop* é de mais baixo consumo, tendo igualmente o clock do CPU desligado além de outros clocks do sistema. Os dados dos registos da SRAM são mantidos, assim como o estado das portas. Por fim, o modo *Standby Mode* é o modo de consumo mais baixo dos três, com o regulador de tensão desactivado, assim como todos os clocks do domínio 1.2V. Exceptuando possíveis configurações de registos de SRAM de backup, do RTC e do circuito do standby, o MCU fica totalmente desligado. De acordo com o Reference Manual [24], são sumarizados os níveis máximo de consumo de cada um dos modos na tabela 3.2 a título de exemplo.

Modo	Condições	frequência (MHz)	Típico	unidades
Run	Todos Periféricos Ligados	216	190	mA
	Todos Periféricos Desligados		92	
Sleep	Todos Periféricos Ligados	216	128	mA
	Todos Periféricos Desligados		18	
Stop	Normal Mode	–	0.55	mA
	Under-drive Mode		0.13	
Standby	SRAM, RTC e LSE Desligados	–	5	uA
	SRAM, RTC Ligados. LSE High Drive Mode		9	

Tabela 3.2: Consumo modos de energia

Levando em conta os requisitos do sistema e as particularidades apresentadas de cada modo, foi escolheu-se o modo de *Low Power* que assegura um óptimo compromisso entre a performance e o consumo reduzido de energia.

3.2 Interfaces com Câmaras de Vídeo

MIPI CSI-2

É uma interface bastante usual em sistemas embebidos com visão por computador. Possuí uma largura de banda muito elevada (a maior das três), sendo de elevada performance conseguindo por isso resoluções de imagem significativas. Se o processador for multi-core, permite que carga no CPU não seja muito elevada. É de reduzidas dimensões o que permite a sua integração em sistemas embebidos compactos. Esta interface necessita de *drivers* externos para a comunicação e é de curta distância. O consumo desta interface é considerada low power e também é de médio custo. O software pode ser mais complexo.

Gigabit Ethernet (GigE)

É uma interface industrial com largura de banda elevada, e com velocidades elevadas de transmissão. É de fácil controlo (requer menos esforço no software) e tem capacidades para maiores distâncias, o

que não é relevante para este projecto. O seu custo é considerado alto e, habitualmente tem maiores dimensões do que os dispositivos com interface CSI e USB, sendo assim mais robustos. O consumo de energia para esta interface é o maior das três apresentadas.

USB

Esta é uma interface com uma largura de banda média, de tamanho reduzido e baixo consumo de energia. Tem capacidades para curtas distâncias (máximo de 5 metros). Têm uma grande aceitação por parte dos fabricantes OEM e, portanto, um número significativo de produtos. É de uso fácil em grande parte dos sistemas operativos, com a funcionalidade de *"plug and play"*. No entanto, sobrecarrega o CPU, sendo considerado como carga média.

O CSI e o USB são os mais usados em aplicações com sistemas embebidos [25].

3.3 Universal Serial Bus (USB)

A STM32 têm periféricos USB embutidos para operações Full Speed e High Speed e são normalizados de acordo com a especificação USB 2.0. Em termos de software, a plataforma de processamento STM32 possuí um módulo *middleware* de bibliotecas para USB. Esta biblioteca fornece uma API para o acesso e a comunicação com vários tipos de classes de dispositivos USB.

As velocidades especificadas pela revisão USB 2.0 são três, sendo elas: Low Speed (LS), Full Speed (FS) e High Speed (HS). A tabela 3.3 ilustra essas velocidades, designações e taxas de transmissão de cada tipo. A plataforma de processamento STM32F7 disponibiliza uma API para a interface USB para as três velocidades apresentadas.

Designação	Velocidades	Taxa Transmissão
LS	Low Speed	1.5 Mbps
FS	Full Speed	12 Mbps
HS	High Speed	480 Mbps

Tabela 3.3: Velocidades USB revisão 2.0

Host

Em sistemas embebidos, a função de USB *host*, é realizada pelo microcontrolador, *single board computer* ou microprocessador. O *host* USB é responsável por detectar a conexão e desconexão de um

dispositivo, controlar o estado de actividades, gerir o controlo de fluxo da comunicação e fornecer energia aos dispositivos conectados. A comunicação no barramento é sempre iniciada pelo host, não sendo possível portanto haver comunicações entre dois *devices*, ou não haver um *host* na comunicação. O dispositivo não pode iniciar a comunicação, mas deverá esperar pelo pedido para transferir dados para o *host*. A única excepção é no caso de o dispositivo ser colocado no modo suspenso (estado de baixo consumo) pelo *host*.

Device

Todos os dispositivos têm um endereço único atribuído pelo host no momento da ligação. Existem dois grupos de devices: os dispositivos *bus-powered* e os *self-powered*. Os dispositivos *bus-powered* são alimentados pelo barramento USB e têm um consumo inferior a 100 mA no caso de dispositivos *low-power*, e inferior a 500 mA, no caso de *high power*. Em qualquer um dos casos, é sempre o dispositivo que extrai a corrente do barramento e nunca o contrário.

Conectado	Ligado	Default	Endereço	Configurado	Suspenso	Estado
Não	–	–	–	–	–	Dispositivo não conectado
Sim	Não	–	–	–	–	Dispositivo conectado
Sim	Sim	Não	–	–	–	Dispositivo conectado e ligado
Sim	Sim	Sim	Não	–	–	Dispositivo conectado e ligado. Foi feito reset
Sim	Sim	Sim	Sim	Não	–	Dispositivo conectado e ligado. Foi feito reset e atribuído um endereço único.
Sim	Sim	Sim	Sim	Sim	Não	Dispositivo conectado e ligado. Foi feito reset, atribuído um endereço único, configurado
Sim	Sim	Sim	Sim	Sim	Sim	Dispositivo conectado, Ligado foi feito reset. mínimo de energia do modo suspenso.

Tabela 3.4: Estado do dispositivo no barramento

Na comunicação do barramento com o dispositivo USB este poderá apresentar diversos estados como estão ilustrados na tabela 3.4.

3.3.1 Conceitos

Nesta subsecção são abordados alguns conceitos básicos relacionadas com a interface USB de modo a se compreender a ligação entre o dispositivo e o *host*. Para tal são aqui descritos sucintamente o significado de *pipes* e *endpoints*, o protocolo de barramento com os formatos dos pacotes e os modos de transacção.

Endpoint

Um *endpoint* é uma parte do dispositivo USB que é única e serve como identificação e terminal do fluxo da comunicação entre o *host* e o dispositivo. Cada dispositivo tem um conjunto de *endpoint* independentes e o software pode comunicar unicamente com o dispositivo através de um ou mais *endpoint*. Cada *endpoint* tem um identificador único determinado pelo dispositivo referido como número de *endpoint*.

Cada *endpoint* determina o tipo e as características de transferência obrigatória entre este endpoint e o software do device tais como a frequência de acesso ao barramento, a largura de banda, o número de *endpoint*, o tratamento de erros e o tamanho máximo do pacote que o *endpoint* é capaz de enviar ou receber. O *endpoint* 0 é especial, sendo usado para configurações e inicializações.

Pipes

Um *pipe* USB é definida como uma associação de um *endpoint* do dispositivo e do software no *host*. Estes *pipes* servem para a transmissão de informação entre o *host* e o *endpoint* do device. Existem dois tipos distintos de *pipes* denominados de *stream* - os dados transportados não têm uma estrutura definida e são unidireccionais - e mensagem - os dados transmitidos têm uma estrutura definida e são bidireccionais. Cada *pipe* tem associado o pedido de acesso ao barramento USB, a largura de banda, o tipo de transferência e um *endpoint* associado, com a direcção e máximo tamanho de *payload* de dados. O *pipe* do tipo *stream* suporta tipos de transferências *Bulk*, *Interrupt* e *Isochrounous*, enquanto o *pipe* do tipo mensagem suporta transferências control, sendo sempre iniciadas pelo *host*.

Protocolo do Barramento

As transacções através do barramento iniciam-se pelo controlador de *host*, que envia um pacote USB que descreve o tipo e a direcção da transacção, o endereço USB do dispositivo e o número de *endpoint*. A este tipo de pacote, dá-se o nome de *token*. Após o pacote de *token* poderá haver um estágio de dados, que poderão ser transmitidos ou recebidos. Por fim, também poderá existir o estágio de *status*, que informa sobre o sucesso ou erro da transacção.

Tabela 3.5: Formato do Pacote Token

8 bits	7 bits	4 bits	5 bits
PID	**ADDR**	**ENDP**	**CRC5**

As tabelas 3.5, 3.6 e 3.7 definem a estrutura de cada um dos tipos de pacotes que serão abordados em mais detalhe em 3.3.2 Descriptores.

Tabela 3.6: Formato do Pacote Data

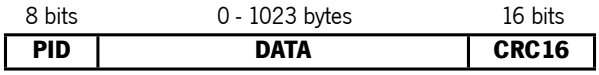
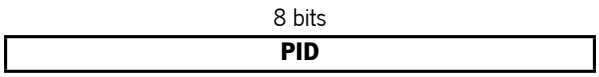


Tabela 3.7: Formato do Pacote Handshake



Tipos de transferências na USB

A especificação USB define o transporte de dados entre o software do *host* e um *endpoint* do *device* através de *pipes*. Existem tipos diferentes de transferências especificadas, que determinam várias características e aspectos específicos da comunicação tais como o formato de dados, a direcção da comunicação, os limites do tamanho de pacotes, os limites do acesso ao barramento e os requisitos de sequência de dados. As transferências são de quatro tipos diferentes, em que cada um tipo é o mais adequado para cada tipo específico de aplicação.

Para o desenvolvimento deste trabalho as transferências do tipo *Control* e *Isochronous* são as relevantes. As transferências de *Control* são transferências não-periódicas iniciadas pelo software do *host* baseadas no envio de pedidos e respostas, usadas para operações como enviar comandos e ler o estado. As transferências *Isochronous* são transferências periódicas, com uma comunicação contínua entre o *host* e o device. É tipicamente usada para aplicações em tempo-real, com os dados encapsulados em relação ao tempo, sendo desta forma *time-critical*.

Packet Identifier

O Product ID (PID) é o campo que identifica o tipo e o *layout* da mensagem USB e é composto por quatro bits do tipo do campo mais quatro bits de CRC. O PID indica o tipo de pacote, o formato do pacote e o tipo de detecção de erro aplicada ao pacote. Tanto o *host* como o dispositivo são obrigados a decodificar completamente todos os campos recebidos de PID, sendo que, em caso de falha, estes pacotes são ignorados. Caso a função receba um PID válido mas que não seja suportado pelo dispositivo, este não envia qualquer resposta ao *host*.

Os PIDs estão divididos em quatro categorias sendo elas: *token*, *data*, *handshake* e especial, sendo que os dois primeiros bits indicam a qual dos grupos se insere cada tipo de PID enviado. A tabela 3.8 resume os tipos do PID da USB.

Tabela 3.8: Tipos de PID

Tipo PID	Nome PID	Valor PID
Token	OUT	0001b
	IN	1001b
	SOF	0101b
	SETUP	1101b
Data	DATA0	0011b
	DATA1	1011b
	DATA2	0111b
	MDATA	1111b
Handshake	ACK	0010b
	NAK	1010b
	STALL	1110b
	NYET	0110b
Special	PRE	1100b
	ERR	1100b
	SPLIT	1000b
	PING	0100b
	Reservado	0000b

3.3.2 Descriptores

Os descriptores são uma estrutura de dados com formatos definidos pelos protocolos USB e pela classe de dispositivos UVC. É através delas que os dispositivos comunicam os seus atributos, interfaces e capacidades. Cada descriptor possui pelo menos, um campo em bytes que contém o tamanho e o tipo do referido descriptor, além de variados outros campos, de acordo com o tipo de descriptor. Para o desenvolvimento deste trabalho, os tipos de descriptores mais importantes são: Informações do dispositivo, Dispositivo, Configuração, Interface e Endpoint.

Descriptor de Informações do Dispositivo

Cada dispositivo USB disponibiliza um descriptor de informações no qual apresenta o nome do produto, a conexão, as configurações, endereço, e as outras informações que o fabricante achar relevante disponibilizar.

Descriptor do Dispositivo

O descriptor do dispositivo apresenta uma informação generalizada que descreve de forma mais minuciosa e menos ambígua um dispositivo USB. Este facto requer que cada dispositivo tenha um e apenas um descriptor de dispositivo. Este descriptor em específico utiliza o *endpoint* 0, do *pipe* predefinido para a transmissão. O valor máximo de tamanho do *endpoint* 0 do dispositivo e outras configurações são descritas. O tipo de classe, o protocolo de comunicação e a especificação USB implementada são algumas outras informações encontradas neste descriptor.

Descriptor de Configuração

O descriptor de configuração descreve informações específicas da configuração do dispositivo. A configuração é dada pelo parâmetro *bConfigurationValue* e o parâmetro *bNumInterfaces* que estabelece o número de interfaces suportadas pela configuração. O valor do parâmetro *wTotalLength* é importante, visto que informa o *host* do número total de bytes de todos os descriptores do dispositivo. Um dispositivo USB poderá ter um ou mais descriptores de configuração, sendo que cada descriptor de configuração por sua vez poderá ter um ou mais descriptores de interface.

Descriptor de Interface

Este descriptor apresenta as interfaces de uma configuração e indica o número da interface, o número de *endpoints* associados e os perfis alternativos. O perfil alternativo é um campo que contém vários perfis e através deles, é possível alternar entre as diferentes configurações. Esse campo é denominado por *bAlternateSetting* e representa as diferentes variantes associadas à mesma interface. A título de exemplo, no caso das câmaras de vídeo, uma interface pode ter diversos perfis alternativos com diferentes resoluções.

Descriptor de *Endpoint*

Cada *endpoint* usado por uma interface específica possuí o seu próprio descriptor de *endpoint*, o qual contém informações relevantes ao *host* para a determinação da largura de banda e a identificação do tipo de transacções, através dos parâmetros *wMaxPacketSize* e *bmAttributes*, respectivamente. Os *endpoints* não podem ser referenciados isoladamente, devendo estar sempre associados à interface, e por conseguinte, como parte do descriptor de configuração.

3.3.3 Classes de dispositivos

A biblioteca da STM32 disponibiliza algumas APIs que implementam e disponibilizam classes de dispositivos de modo a serem desenvolvidas aplicações USB. Entre os exemplos que podem ser citados existem as APIs de Human Interface Device (HID), o Communications Device Class (CDC) e USB Audio Class (UAC).

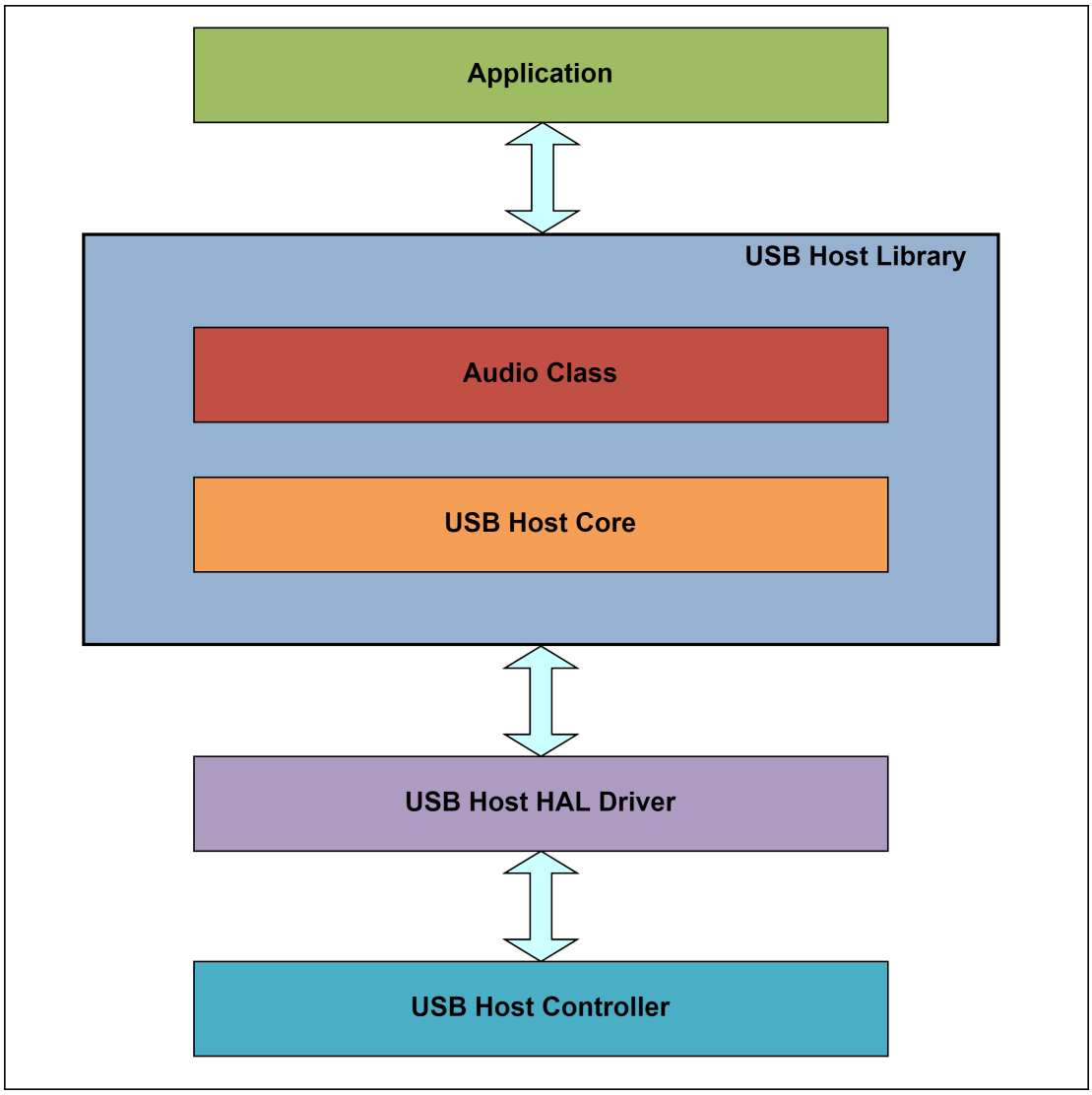


Figura 3.2: Diagrama de camadas classe UAC

A figura 3.2 mostra um exemplo de uma arquitectura de camadas da classe de dispositivos Áudio. Podemos observar o diagrama de camadas em que a classe de dispositivos USB Áudio pertence ao *Middleware*, sobre a camada *USB Host Library* da STM32.

3.4 Câmara de Vídeo

A câmara de vídeo é da ELP, ELP-USB13M02-MFV, a qual possui as especificações apresentadas na tabela 3.9. Trata-se de uma câmara com a interface USB e da classe UVC [26].

Modelo:	ELP-USB4KHDR01-V100
Sensor:	Sony IMX317 (1/2.5")
Resolução Máxima:	3840(H) *2880 (V)
Sensibilidade:	1000mV/Lux-sec 0.65V/lux-sex@550nm
Área de Imagem:	6100μmx4524μm
Formato Comprimido:	MJPEG/YUY2 (YUV)
Resolução de Frames:	MJPEG 3840x21600@30fps / 2592x1944@ 30fps 2048x1536@ 30fps /1600x1200@ 30fps 1920x1080@ 30fps / 1280x1024@30fps 1280x960@30fps /1280x720@ 30fps 1024 x 768@ 30fps / 800 x 600@ 30fps 640x480@ 30fps / 320x240@ 30fps YUV 3840x21600@1fps / 2592x1944@ 1fps 2048x1536@ 1fps / 2592x1944@ 3fps 1600x1200@ 3fps / 1920x1080@ 3fps / 1280x1024@3fps 1280x960@5fps /1280x720@ 5fps 1024 x 768@ 5fps / 800 x 600@ 20fps 640x480@ 30fps / 320x240@ 30fps
Definição de Centro:	1000 LW/PH (centro)
S/N	26dB
Baixa Iluminação	0.2lux
Shutter	shutter eletrónico/ exposição de frame
Parâmetros ajustáveis:	Brightness Contrast Hue Saturation Sharpness Gamma White Balance Exposure Focus
Foco:	2.8-12mm varifocal lens
Distância do Objecto:	8CM-100M
Resolução:	1000LW/PH (Center)
Interface:	USB 2.0 High Speed
Protocolo:	USB Video Class(UVC)
AGC/AEC/AWB/ABF	Auto
Parâmetros ajustáveis:	Brightness Contrast Hue Saturation Sharpness Gamma White Balance Exposure Focus
Parâmetros lentes:	2.8-12mm varifocal lens
Conector:	Support 2P-2.0mm socket
Alimentação:	USB BUS POWER 4P-2.0mm socket
Alimentação:	DC5V
Corrente:	200-250 mA
Tamanho Módulo:	42mm × 42mm
Temperatura (Storage temperature):	-20°C to 70°C
Temperatura (Working temperature):	0°C to 60°C
Cabo USB:	3M
Sistema Operativo:	Win XP/Vista/Win7/Win8 / Linux with UVC(above linux-2.6.26) MAC-OS X 10.4.8 or later/ Android 4.0 or above with UVC

Tabela 3.9: Especificação câmara

Destacam-se aqui as temperaturas de funcionamento e de armazenamento, a baixa iluminação, a alimentação de 5 V, a corrente máxima de consumo de 250 mA, a resolução máxima e outras resoluções disponíveis, a sensibilidade e os parâmetros configuráveis.

3.4.1 Análise das capacidades de Vídeo

Utilizando o programa utilitário USBView e conectando directamente a câmara de vídeo a um PC, faz-se a leitura dos descriptores fornecidos na comunicação entre o PC com o sistema operativo Windows e a câmara de vídeo.

A tabela 3.10 apresenta o descriptor de informação do dispositivo. A partir deste descriptor confirma-se o modo de USB como sendo *High Speed* e o endereço de dispositivo.

English product name:	"HD USB CAMERA"	
ConnectionStatus:		
Current Config Value:	0x01	->Velocidade Barramento: High
Device Address:	0x05	->Tipo de Transferência Interrupt
Open Pipes:	1	= 1 transação por microframe, 0x10 max bytes

Tabela 3.10: Descriptor de Informação do Dispositivo

Na tabela 3.11 temos o descriptor do dispositivo, o qual apresenta o máximo tamanho dos pacotes, o protocolo do dispositivo, a classe do dispositivo e outras informações sobre o fabricante e o produto.

bLength:	0x12	
bDescriptorType:	0x01	
bcdUSB:	0x0200	
bDeviceClass:	0xEF	->Dispositivo de código de função Multi-interface
bDeviceSubClass:	0x02	->Subclasse do dispositivo
bDeviceProtocol:	0x01	->Protocolo do descritor de associação de interface
bMaxPacketSize0:	0x40	= (64) Bytes
idVendor:	0x32E4	= ID Frabricante não registado USB.org
idProduct:	0x0317	
bcdDevice:	0x0333	
iManufacturer:	0x01	
English (United States) "4K USB CAMERA"		
iProduct:	0x04	
"HD USB CAMERA"		
iSerialNumber:	0x03	
English (United States) "01.00.00"		
bNumConfigurations:	0x01	

Tabela 3.11: Descriptor do Dispositivo

O descriptor de configuração contém os campos de tamanho total de configuração, o número de interfaces e a máxima potência da câmara de vídeo. Pode-se notar que, pela máxima potência da câmara

de vídeo, esta é desta forma considerada como *High Power*. Este descriptor é apresentado na tabela 3.12.

bLength:	0x09	
bDescriptorType:	0x02	
bcdUSB:	0x0200	
wTotalLength:	0x073A	(1850 bytes)
bNumInterfaces:	0x04	(4 Interfaces)
bConfigurationValue:	0x01	(Configuração 1)
iConfiguration:	0x00	
bmAttributes:	0x80	
D7: Reserved, set 1:	0x01	
D6: Self Powered:	0x00	
D5: Remote Wakeup:	0x00	
D4..0: Reserved, set 0:	0x00	
MaxPower:	0xFA	(250 mA)

Tabela 3.12: Descriptor de Configuração

O descriptor de interface da tabela 3.13 apresenta os parâmetros pelos quais esta interface é referenciada e acedida. Os campos *bAlternateSetting* e *bNumEndpoints* abordados no capítulo anterior são os parâmetros de identificação de interface.

bLength:	0x09	
bDescriptorType:	0x04	
bInterfaceNumber:	0x00	
bAlternateSetting:	0x00	
bNumEndpoints:	0x01	
bInterfaceClass:	0x0E	->Interface da Classe de Vídeo
bInterfaceSubClass:	0x01	->Subclasse da Interface de Controlo de Vídeo
bInterfaceProtocol:	0x00	
iInterface:	0x04	
English (United States) "HD USB CAMERA"		

Tabela 3.13: Descriptor de interface - Interface Descriptor

O descriptor de enpoint apresentado pela tabela 3.14 trata-se de um endpoint do tipo interrupt, o qual é usado para configurações e transferências de controlo. Este descriptor de enpoint contém o respectivo endereço, o máximo de dados transferidos por cada microframe, o intervalo entre transacções e o tipo de transferência.

bLength:	0x07	
bDescriptorType:	0x05	
bEndpointAddress:	0x87	->Direcção: IN - ID Endpoint: 7
bmAttributes:	0x03	->Tipo de Transferência Interrupt
wMaxPacketSize:	0x0010	= 1 transação por microframe, 0x10 max bytes
bInterval:	0x08	

Tabela 3.14: Exemplo do descriptor de um Endpoint do tipo Isócrono

3.4.2 Ligação de Hardware

Como referido anteriormente, a câmara de vídeo é um dispositivo High Speed. No entanto, a placa de processamento apenas possuí Physical Layer (PHY) para Full Speed (FS), pelo que a integração de um Physical Layer (PHY) High Speed (HS) externo para a ligação com a STM32 é necessária. O módulo USB3300 foi escolhido como o PHY externo, sendo que possui o protocolo UTMI + Low Pin Interface (ULPI) que converte os 4 pinos que a especificação USB determina para a comunicação HS para 12 pinos, que se ligam ao microcontrolador. O diagrama da figura 3.3 mostra-nos o módulo da Waveshare juntamente com a estrutura de ligações deste módulo, no qual um conector USB standard ou mini conecta-se, por um lado, e por outro, o *link* ULPI. Esse *link* disponibiliza vários pinos para conexão com a placa. Esses pinos estão descritos na tabela 3.15.

Tabela 3.15: Sinais da Interface ULPI

Sinal	Direcção	Descrição
CLKOUT	OUT	Clock de saída de referência de 60MHz.Todos os sinais ULPI são actualizados pela transição ascendente de forma síncrona.
DATA[7:0]	I/O	Barramento bidireccional de dados de 8 bits. O direccionamento do barramento é determinado por DIR.
DIR	OUT	Controla a direção do barramento de dados. Quando PHY tem dados a transferir para o Link, DIR é colocado no estado High. Quando não há dados é colocado a Low.
STP	IN	O STP informa ao PHY que foi enviado o último byte de dados no ciclo anterior.
NXT	OUT	Quando o Link está enviando dados para o PHY, NXT indica quando o byte foi aceite pelo PHY.

A ligação entre o módulo de PHY externo e a placa de processamento STM32 é feita como mostra a figura 3.3 e a câmara fica ligada ao PHY externo.

A biblioteca da classe USB disponibilizada pela STM tem o suporte para o uso de PHY externo com as configurações dos sinais descritos pela tabela.

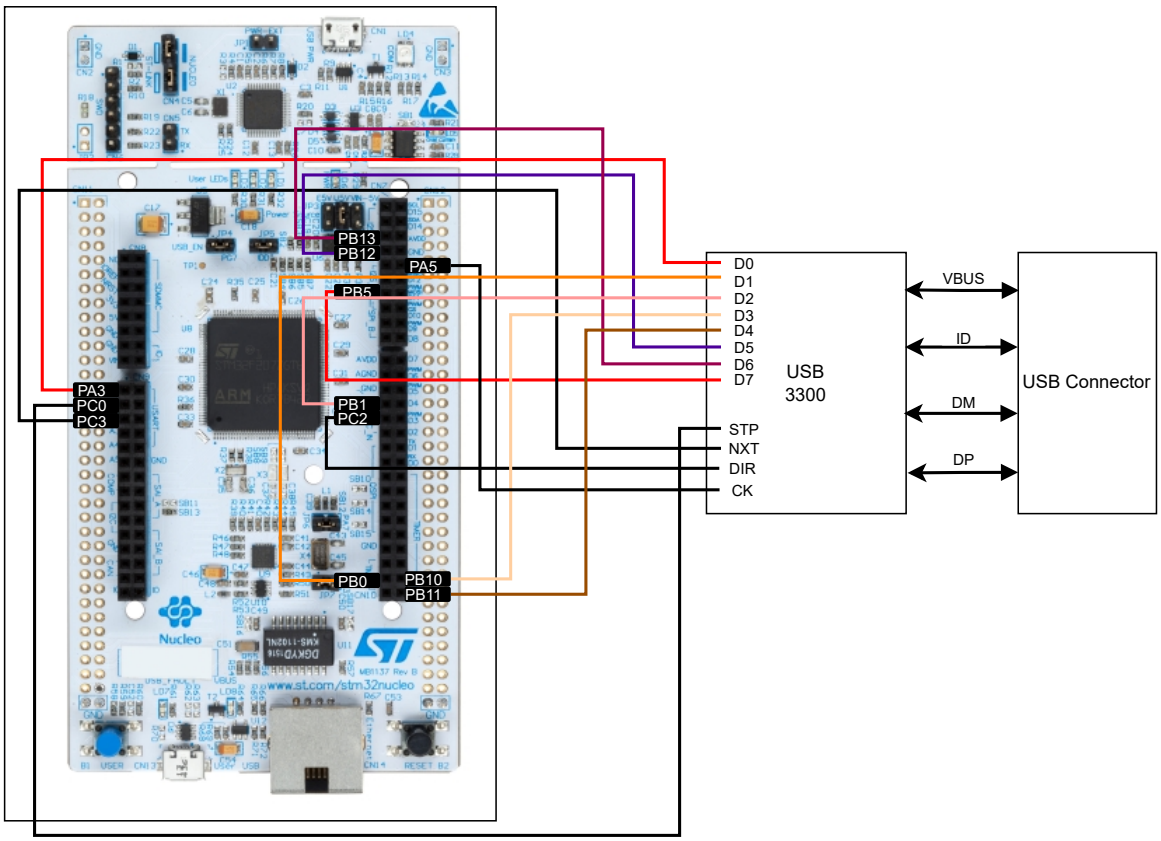


Figura 3.3: Módulo USB3300 [8]

3.5 Cartão SD

O armazenamento de dados é feito com um cartão Secure Digital (SD) de modo a aumentar a capacidade de memória do sistema. Os cartões microSD foram os escolhidos devido à sua reduzida dimensão e comodidade para o levantamento posterior das imagens adquiridas. Os cartões SD permitem que os dados possam ser descarregados para uma unidade de processamento *in-situ* com bastante rapidez e fiabilidade em termos teóricos.

Resolveu-se implementar a comunicação do microSD com o microcontrolador através do periférico da STM32, o SDMMC. Também de modo a acrescentar a multioperacionalidade entre sistemas operativos, o sistema de ficheiros File Allocation Table (FAT) foi escolhido como o sistema de armazenamento de dados.

3.5.1 File Allocation Table (FAT) Sistema de Ficheiros

Sendo a FAT muito utilizada em sistemas embebidos, existe uma API desenvolvida em open-source para a STM32. Apresenta-se o diagrama da biblioteca na figura 3.4, quevai ser aprofundado no capítulo

seguinte.

Tabela 3.16: Tipos e características de Cartões de Memória

Tipo de Cartão	Limite tamanho	Velocidade de Escrita	Tipo FAT
SD	4 GB	0.9-20 MB/s	FAT16
SDHC	32 GB	2-40 MB/s	FAT32
SDXC	2 TB	max 300 MB/s	exFAT

O controlador SDMMC é um barramento presente em algumas famílias do microcontrolador STM32, incluindo a F767, que faz a interface de comunicação com cartões SD e MMC, fornecendo funções de *clock*, comando e transferência de dados. A interface é configurável e com uma fácil conexão, sendo que para tal são necessários poucos pinos de ligação. É compatível com as especificações SD 2.0, SDIO 2.0 e MMC 4.2. Este periférico comporta o suporte para modos de barramento de dados de 1-bit, 4-bit e 8-bit, aumentando assim a taxa de transferência.

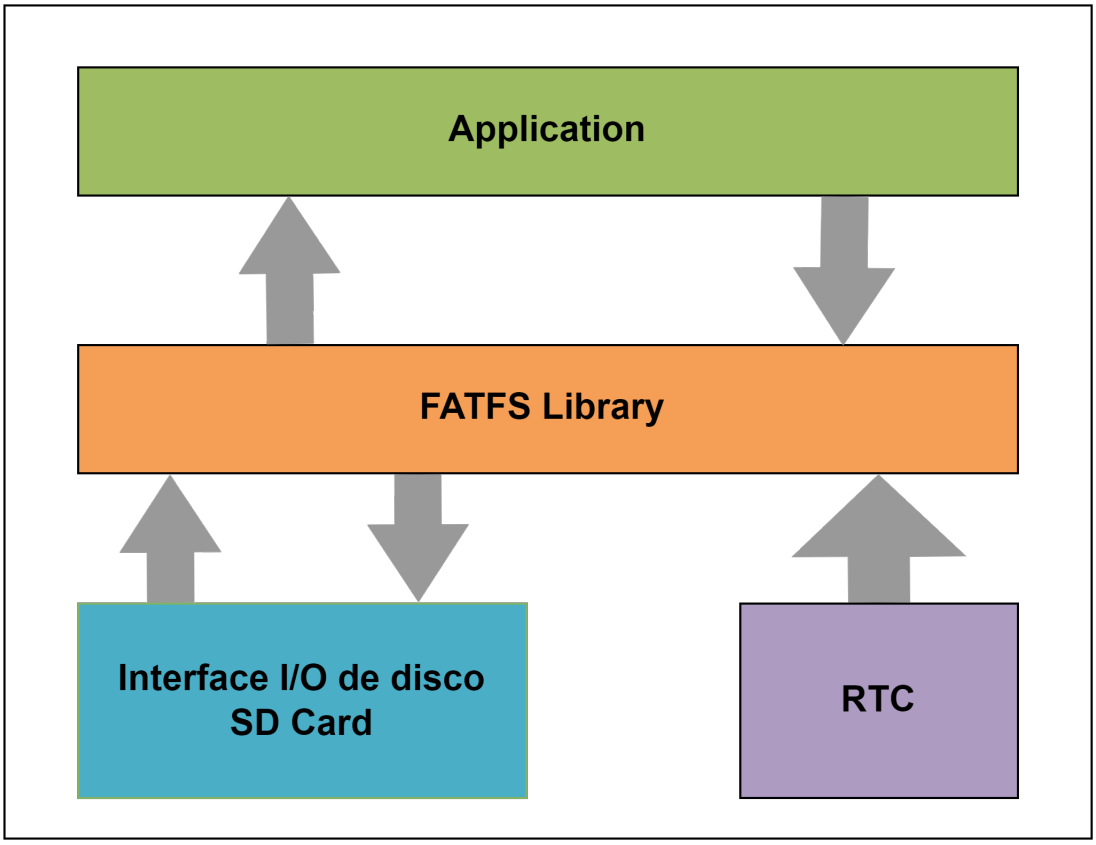
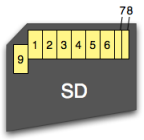


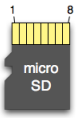
Figura 3.4: Diagrama da biblioteca FATFS

A comunicação no barramento é feita com base em transferências de comandos e dados, com as

operações essenciais mais elementares das transacções seguindo a estrutura de comandos e respostas. As transferências de dados são feitas em blocos de dados ou streams de dados.



Pin	SD	SPI
1	CD/DAT3	CS
2	CMD	DI
3	VSS1	VSS1
4	VDD	VDD
5	CLK	SCLK
6	VSS2	VSS2
7	DAT0	DO
8	DAT1	X
9	DAT2	X



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

Figura 3.5: Configuração dos Pinos SD e microSD

A figura 3.6 exemplifica através de um diagrama de blocos o periférico do SDMMC do MCU STM32. Este consiste em duas partes essenciais. O adaptador SDMMC, que executa todas as funções especificas para o cartão de memória (geração de clock, comandos e transferências de dados), e o barramento APB2, que acede aos registos do adaptador e gera as interrupções e os pedidos de DMA.

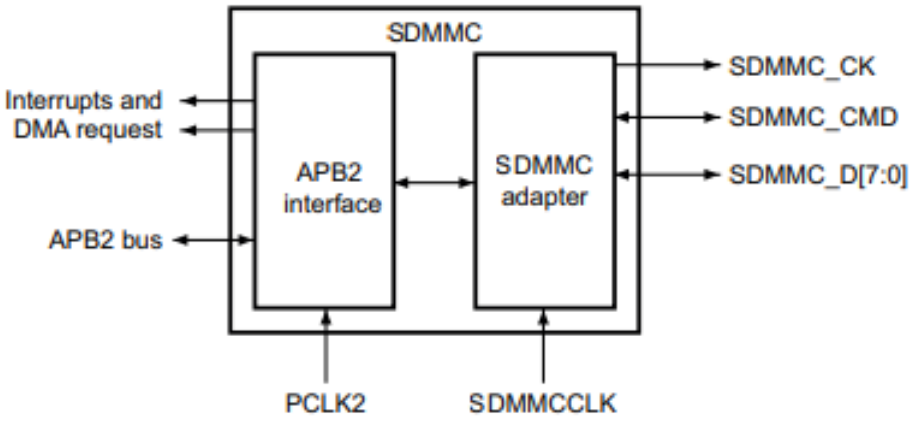


Figura 3.6: Diagrama de blocos do periférico SDMMC

Uma condição imposta pelos documentos e que é importante ressaltar é a seguinte limitação em termos de frequências, dadas pela seguinte equação:

$$Frequency(PCLK_2) > (3 * Width/32) * Frequency(SDMMC_{ck})$$

Isto significa que a escolha da frequência de clock do periférico para o cartão SD (Frequency($SDMMC_{ck}$)) deve respeitar a relação apresentada em cima, cujos os valores da frequência do barramento APB2 (Frequency($PCLK_2$)) e a largura do barramento (width) devem ser cuidadosamente escolhidos. Portanto para a configuração do periférico com um barramento de 4 bits (*width*), a frequência do barramento deve ser mais do que 3/8 da frequência de *clock* do periférico SDMMC.

3.6 Arquitectura

Tendo em conta o objectivo e a motivação discutidos no capítulo 1, o sistema tem de ser capaz de fazer a aquisição de capturas de imagem a uma taxa de amostragem fixa e configurável e guardar as imagens num cartão de memória SD. O sistema tem de ter dimensões reduzidas, de modo a não ultrapassar as dimensões do *underwater-housing* da Vitrovex.

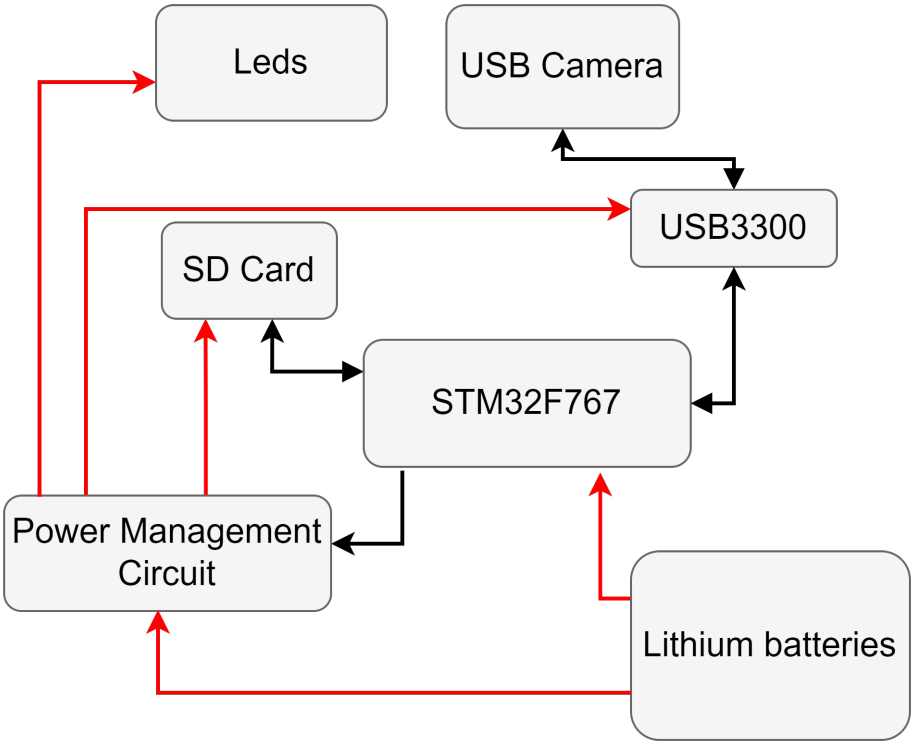


Figura 3.7: Sistema de Captura

Para que se possa fazer a captura de imagens, torna-se necessário implementar sobre a camada da API do USB a classe de dispositivos UVC. Devido às condições de iluminação inexistentes no oceano profundo o bloco de LEDs para a captura é igualmente fundamental. De modo a que a autonomia seja

elevada, o sistema deve ter o consumo de energia minimizado, sempre que não esteja a fazer capturas.

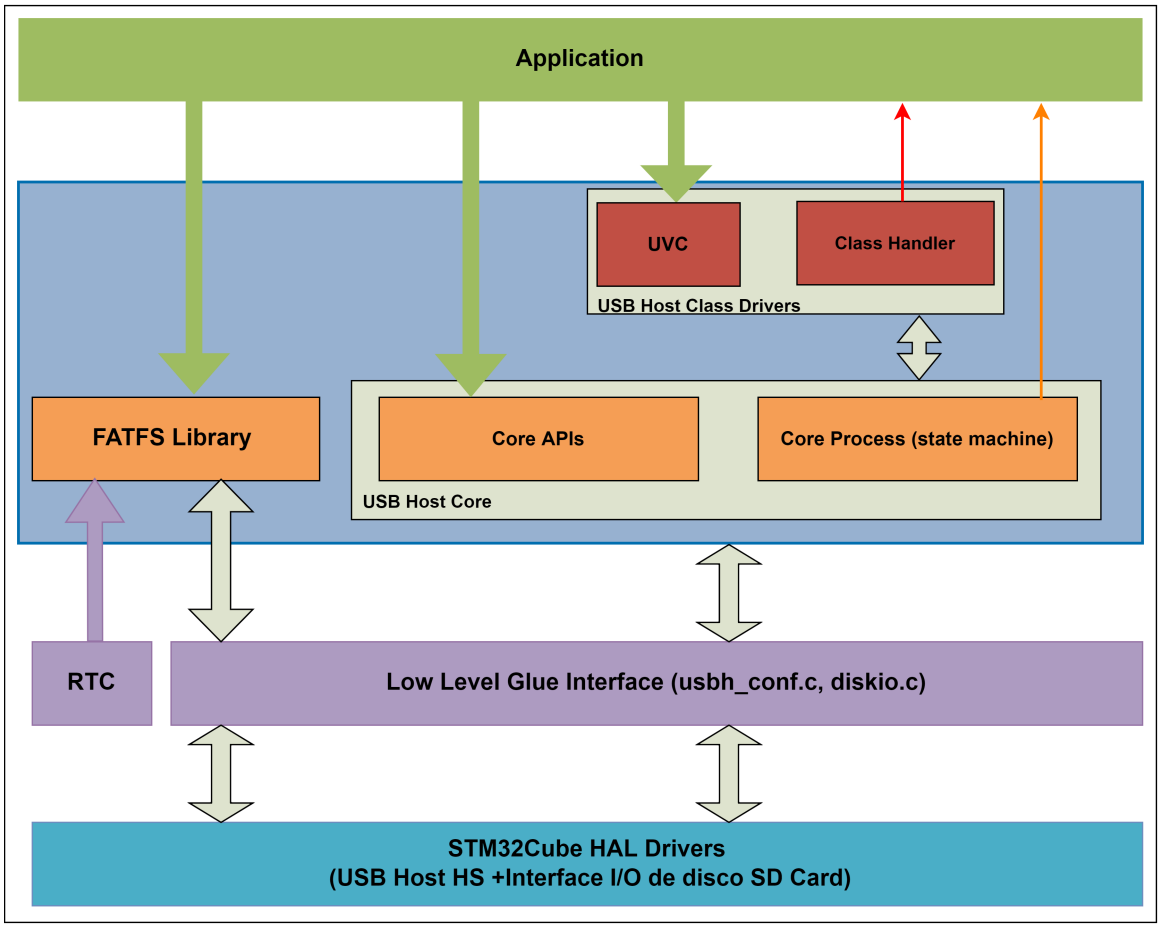


Figura 3.8: Diagrama de Camadas

A figura 3.8 representa o diagrama de camadas completo do desenvolvimento de Software.

O desenvolvimento da aplicação consiste na captura de imagens de uma câmara de vídeo com o protocolo UVC para a plataforma de processamento da STM32, o modelo F767. O sistema ainda deve accionar o comando de iluminação que comporta como uma luz *flash* para a captura de imagens. A imagem deve ser guardada num cartão de memória SD para o armazenamento de dados. A gestão de energia deve ser optimizada de modo a ter um baixo consumo de potência. A comunicação com a câmara de vídeo é feita através da interface Universal Serial Bus, com a implementação desenvolvida do driver da USB Video Class.

A bateria é calculada em função da escolha do tempo de autonomia do sistema, levando em conta os pré-requisitos do consumo de corrente. Por conseguinte a capacidade do armazenamento também pode ser derivada a partir da autonomia do sistema. Sendo a taxa de amostragem fixa, para um determinado tempo de funcionamento, a capacidade de armazenamento pode ser deduzida.

Capítulo 4: Implementação

Neste presente capítulo é aborda-se o desenvolvimento completo do protótipo para o sistema de monitorização de águas profundas. A arquitectura definida para o sistema engloba a placa de processamento, uma PHY externa, uma câmara de vídeo, um sistema de iluminação e uma memória externa SD. O desenvolvimento é feito com o microprocessador da ARM, a STM32. Para tal, é necessário implementar o driver do protocolo UVC - para que se possa fazer a comunicação com o módulo de câmara USB. São abordados alguns conceitos relevantes do protocolo USB e, de seguida, o desenvolvimento do driver e a comunicação com a câmara. O software de desenvolvimento usado como Integrated Development Environment (IDE) foi o Keil juntamente com STM32CubeIDE, tendo como auxiliar de configuração e bibliotecas o MXCUBE. Foi usado o Realterm como terminal UART para depuração na fase de testagem. A aplicação 7YUV foi usada como software de imagens do formato Raw YUV. O *Power Consumption Calculator*(PCC) que éuma ferramenta da aplicação STM32CubeIDE, foi usada para a simulação do consumo de energia do sistema. O HxD e o MiniTOolPartition fotam utilizados para a correcção e averiguação do sistema de ficheiros FAT. O programa Amcap foi utilizado para a configuração dos parâmetros da câmara e o USBView para a visualização da interface entre a câmara de vídeo e o sistema operativo Windows 10.

Foi utilizado o firmware do repositório da STM32 disponibilizado pela Hardware Abstraction Layer (Software Library) (HAL) para a interface USB e em específico a classe de dispositivos UAC como base para a implementação da classe UVC.

Igualmente foi usada o firmware para a FAT desenvolvida em open-source como uma biblioteca por (C)ChaN, 2017. Esta API foi usada para a criação e gestão do sistema de ficheiros da aplicação.

4.1 Abordagem

A classe de dispositivos UAC é definida para dispositivos de manipulação de áudio, voz ou funcionalidades de sons no geral. A STM32 disponibiliza uma API para esta classe de dispositivos, na qual implementa a interface de comunicação para dispositivos deste tipo. Dado a semelhança entre as classes UAC e UVC, primeiro é descrita a implementação da classe UAC e depois analisadas as diferenças e a implementação da UVC. Os documentos UM2195 e *Device Class Definition for Audio Devices* definem a classe e a sua implementação para STM32 [27] [28].

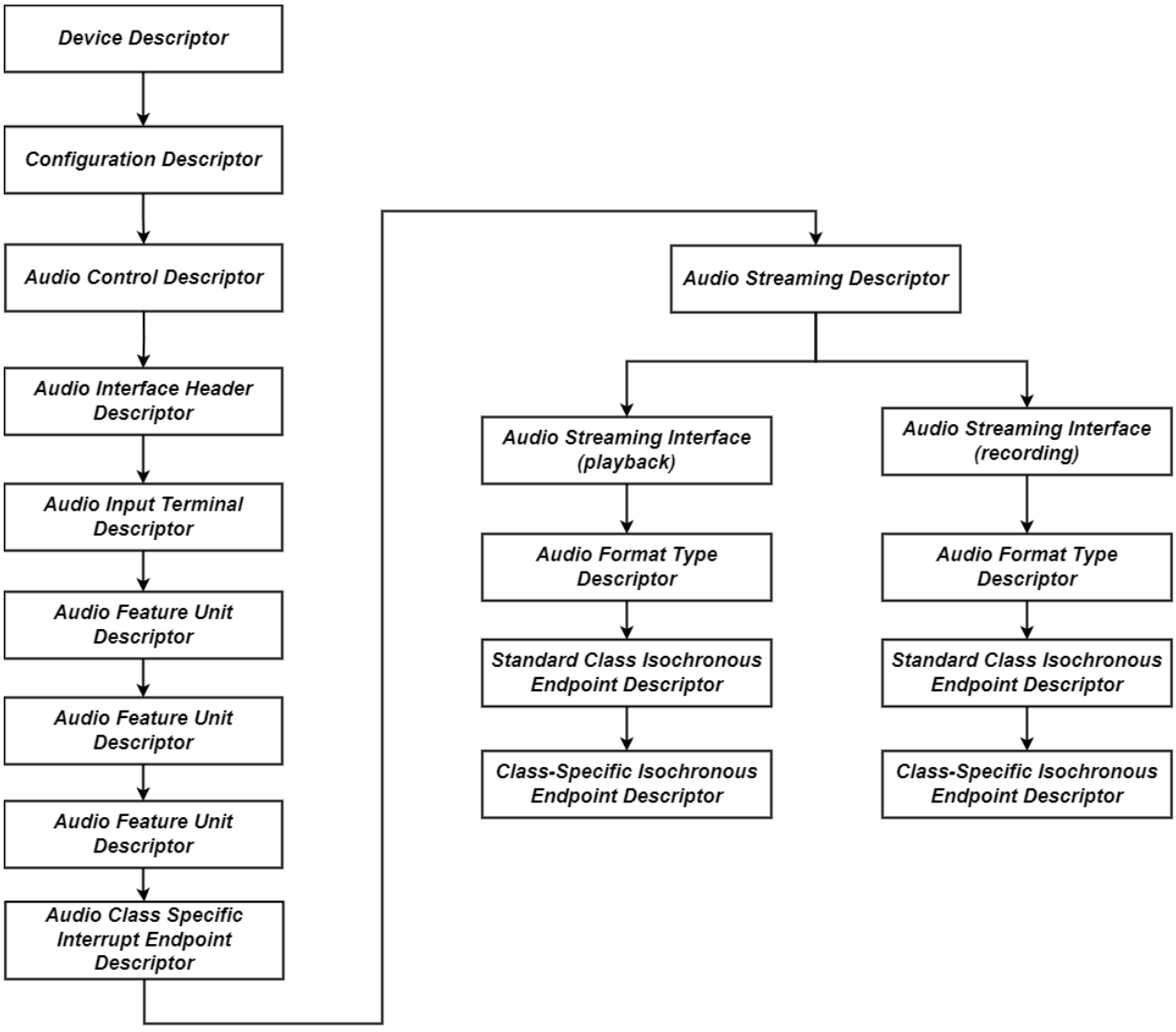


Figura 4.1: Camadas de descriptores da classe de dispositivos áudio

A figura 4.1 apresenta a estrutura hierárquica dos descriptores da classe Áudio. A camada superior trata dos descriptores de dispositivo e de configuração, ao qual todos os dispositivos USB possuem. Na

camada inferior temos os descriptores de interface e os descriptores de terminal de entrada e saída, de funcionalidades e *class-specific*. Esta camada possuí os descriptores de controlo e dá-se pelo nome de *Audio Control*. Seguidamente, existem duas interfaces de *streaming*. Ambas são constituídas por descriptores de interface áudio standard e *class-specific*, assim como o descriptor *class-specific* de formato, de *endpoint* isócrono e um descriptor de *endpoint* isócrono *standard*.

Apresenta-se na figura 4.2 a estrutura hierárquica dos descriptores da classe de dispositivos vídeo, a partir do estudo das capacidades da câmara de vídeo com os descriptores obtidos em 3.4.1 - Análise das capacidades de Vídeo.

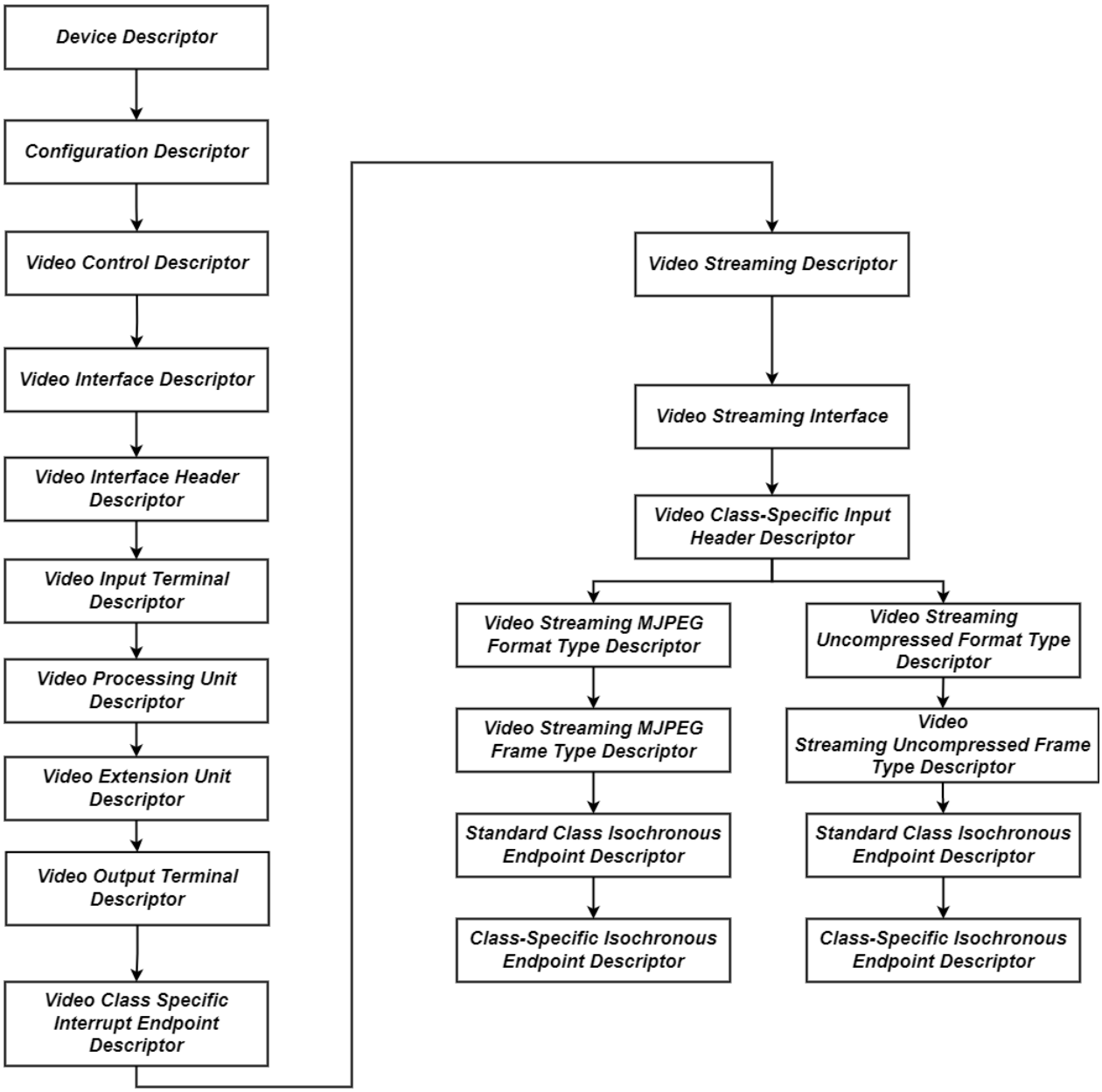


Figura 4.2: Camadas de descriptores da classe de dispositivos vídeo

A hierarquia dos descriptores da classe de dispositivos áudio e vídeo possuem algumas semelhanças observadas pelas duas figuras apresentadas em 4.1 e 4.2. A primeira camada de descriptores das duas classes é exactamente igual em termos estruturais e a segunda retrata a mesma interface de controlo, ainda que hajam diferenças nesta camada. Já a terceira e quarta camadas de descriptores da classe áudio possui descriptores de interface para configurações diferentes, a terceira, com *streaming* em *playback* e portanto através do descriptor de *Output* e a quarta camada de descriptores, com *streaming* e *recording*, e portanto, através do descriptor de *Input*.

Para o caso desta análise da implementação da classe de vídeo, há apenas uma única configuração. Isto deve-se deve ao interesse para a aplicação em específico, na qual só interessa a transmissão do *streaming* de dados da câmara de vídeo tal como visto pelo estudo dos requisitos do sistema. Esta classe apresenta a interface para a implementação da classe de áudio.

A figura 4.3 mostra através de um diagrama UML a definição da classe áudio implementada pela biblioteca da STM32.

O diagrama de interface da classe apresenta as funções e as variáveis de estruturas de dados usados e fornecidos. No entanto, é necessário fazer a análise estrutural e comportamental, para que seja possível implementar a classe UVC, assim como a análise relacional com a API do USB.

A partir da descrição da classe de áudio podemos observar a definição da estrutura de dados "_AUDIO_Class"a qual possui as definições de classe, inicialização da interface, *Class Requests* (Pedidos de classe), Processo Áudio e Start of Frame (SOF).

4.2 Análise UAC

Uma análise às estruturas de dados presentes dentro da "_AUDIO_Process"pode constatar que a semelhança na aquisição dos descriptores, a máquina de estados de configuração e o processo de *streaming* é verificada na camada superior da hierarquia dos descriptores. Portanto, as estruturas que são muito semelhantes ou iguais às pretendidas para a classe de vídeo vão ser aqui descritas de seguida.

A estrutura da figura 4.4 relaciona as características de um *endpoint* e o seu respectivo tamanho, assim como a interface e a configuração da interface. É assim estruturado o *streaming* de entrada, o que condiz com a estrutura pretendida para o *streaming* da classe de vídeo. A natureza similar do tipo de dados de áudio e de vídeo, bem como a especificação USB dessas classes de dispositivos, possibilitam esta análise. E tal como a classe de dispositivos áudio, o tipo de transferências USB para esta implementação

Figura 4.3: Classe Áudio

é a *isochronous* tal como explicado em 3.3 - Universal Serial Bus (USB). A estrutura associa o *streaming* a um pipe e os tempos de acesso, assim como aos diferentes descriptores do dispositivo áudio e vários buffers de controlo e armazenamento, fornecendo assim a interface de *streaming*. Além disso, possuí uma estrutura de controlo para funcionalidades particulares tais como volumes actuais, máximos e mínimos, mudo e resolução.

A estrutura de dados do *endpoint* para vídeo está baseada nesta implementação da interface de *streaming*, com as devidas adaptações para esse tipo de dados.

4.2.1 Estrutura

A figura 4.5 apresenta as estruturas de dados dos descriptores *Header*, *Input Terminal*, *Output Terminal*, *Feature*, *Mixer* e *Selector*. A estrutura de dados "AUDIO_HeaderDescTypeDef"é o descriptor *Header* do Controlo de Áudio. Para a classe de vídeo, o descriptor de *Header* é muito semelhante de

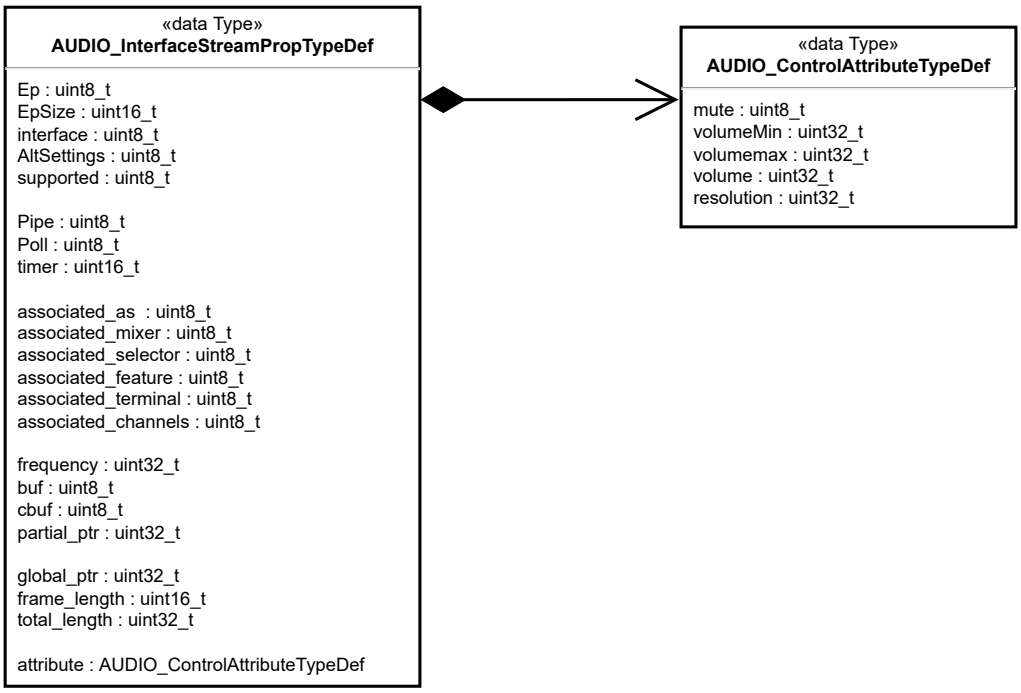


Figura 4.4: Estrutura da interface de streaming

acordo com as especificações USB para áudio e vídeo, à excepção da adição de um campo *dwClock-Frequency* e ao campo "bcdADC"que se chama "bcdUVC". Como referido em 3.3 - Universal Serial Bus (USB), o descriptor de terminal de input existe para a classe de vídeo, mas a estrutura apresentada em "AUDIO_ITDescTypeDef"apresenta algumas diferenças na classe UVC. Este descriptor para UVC possuí ainda os campos de "wObjectiveFocalLengthMin", "wObjectiveFocalLengthMax", "wOcularFocalLength", "bControlSize"e "bmControls". Todos são do tipo de dados byte, excepto os três primeiros que são de dois bytes e o último elemento de três bytes. Da mesma forma, não existem correspondentes para "bNrChannels", "wChannelConfig"e "iChannelNames"da classe UAC na UVC. O descriptor de terminal de saída da classe de áudio é representada por "AUDIO_OTDescTypeDef"e tem correspondência directa com o descriptor de terminal de *Output* da classe de dispositivos de vídeo. Os descriptores de *Feature*, "AUDIO_FeatureDescTypeDef"e de Mixer, "AUDIO_MixerDescTypeDef"não têm correspondente na classe de dispositivos UVC. O descriptor de Selector, "AUDIO_SelectorDescTypeDef"pode ser opcional para a classe de dispositivos UVC.

Através do controlo de Áudio é possível aceder às unidades e terminais do dispositivo tal como visto na figura 4.5. É importante referir que cada dispositivo deverá ter apenas um descriptor de controlo. Resumidamente, o descriptor de controlo "AUDIO_ACDescTypeDef"descreve os caminhos e controlos das unidades apresentadas.

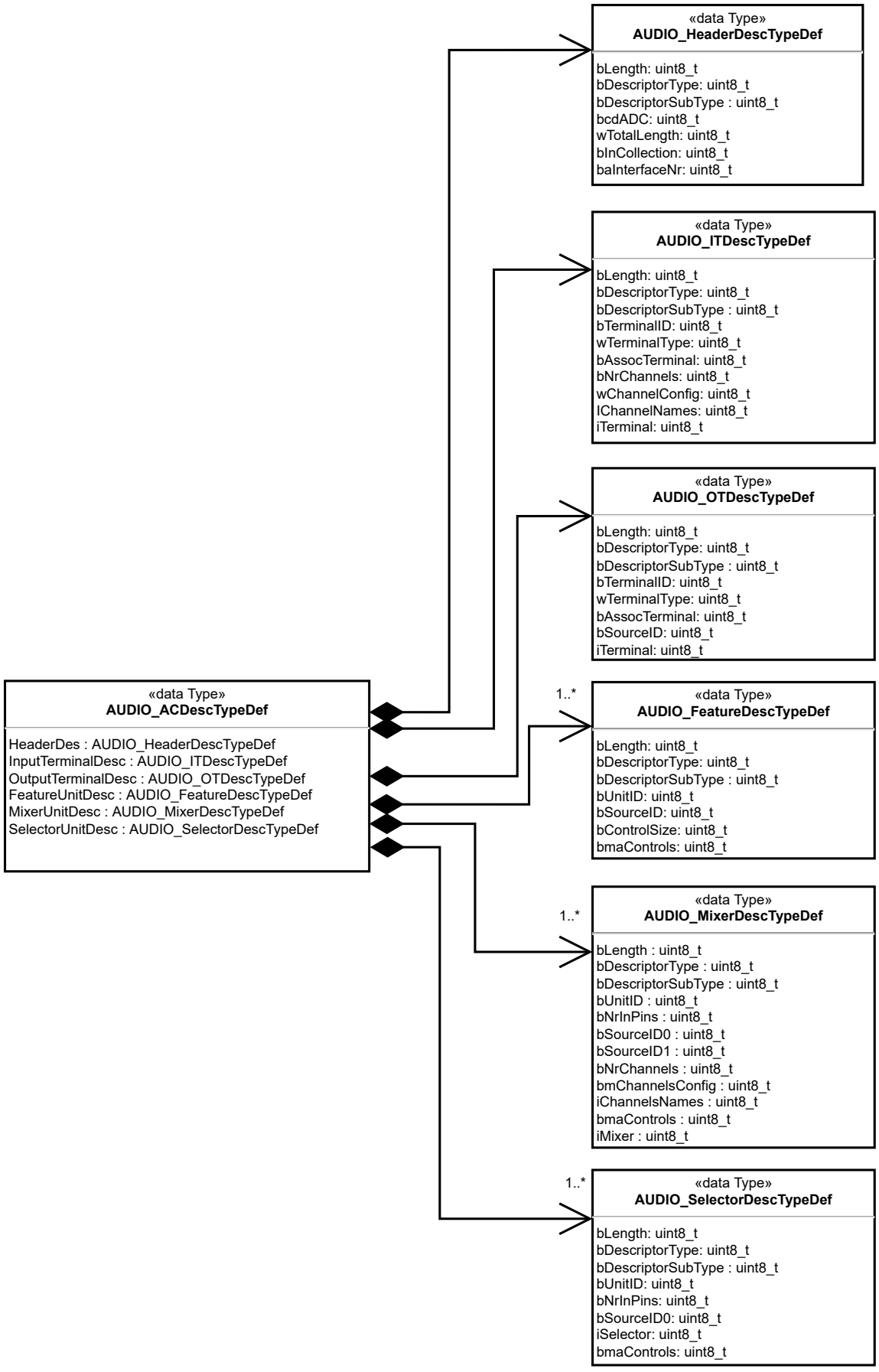


Figura 4.5: Estrutura de dados do Controlo de Áudio

A estrutura apresentada em 4.6 é o descriptor de *streaming* da classe e é representado por "AUDIO_ASDescTypeDef", o qual contém os descriptores de *streaming* de áudio e de formato. O descriptor de interface áudio de *streaming* "AUDIO_ASGeneralDescTypeDef"e o de formato de streaming "AUDIO_ASFormatDesc- TypeDef"contêem as informações tais como o formato, a identificação da conexão de um terminal, o número de canais, as frequências de aquisição e outras informações da transmissão de dados. O formato em concreto depende do dispositivo, mas tem de ser um formato válido, segundo a norma USB.

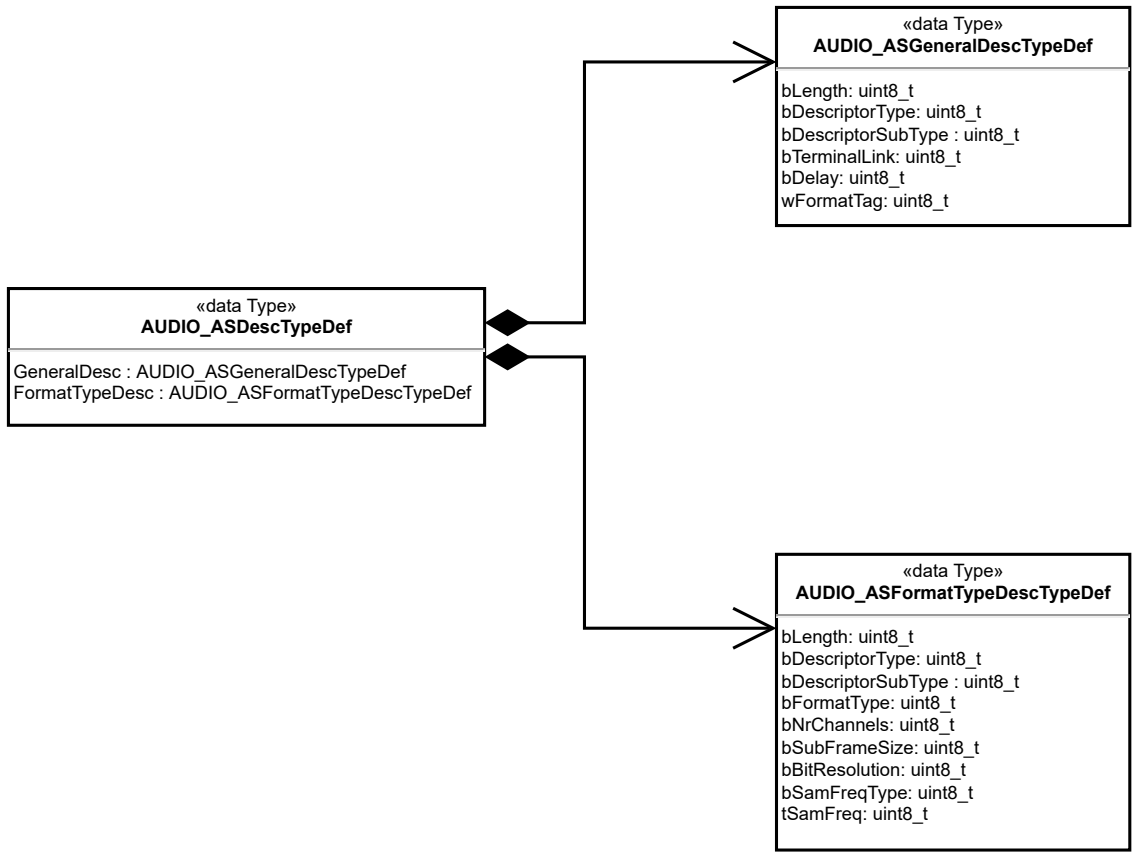


Figura 4.6: Estrutura de dados Streaming de Áudio

À semelhança da classe de áudio, a classe de vídeo também possui o descriptor de *streaming* de Vídeo (VS), o qual apresenta mais campos se comparado com a classe de áudio. Para UVC, deve haver um descritor de formato de *streaming* e outro descriptor de *streaming* de vídeo. O descriptor de formato de *streaming* representa os tipos de formatos suportados pela câmara de vídeo que poderá ser MJPEG, H.264, Descomprimido, entre outros. O descriptor de *streaming* de vídeo representa o tipo de *frame* de *streaming* para cada formato de imagens que a câmara de vídeo suporte. Uma câmara de vídeo que suporte vários formatos, tem para cada formato especificado um ou vários tipos de *frames* diferentes. Um

exemplo pode ser uma câmara de vídeo que suporte o formato MJPEG, podendo ter *streaming* de vídeo para *frames* de resolução 1600x1200, 1280x960 e 1280x720. Apesar de só suportar um formato, tem três *streaming* de vídeo ou tipos de *frames*.

A estrutura de dados 4.7 contém todos os descriptores, tanto de controlo como de *streaming*. Como referido anteriormente, esta estrutura só pode ter uma interface de controlo de áudio e uma ou mais interfaces de streaming. Dado ao estudo dos descriptores UVC e às capacidades da câmara de vídeo feito em 3.4, é expectável que a classe de vídeo tenha igualmente esta estrutura contendo os descriptores de *class-specifics*, com as devidas alterações para os descriptores de controlo de Vídeo (VC) e de *streaming* de Vídeo (VS).

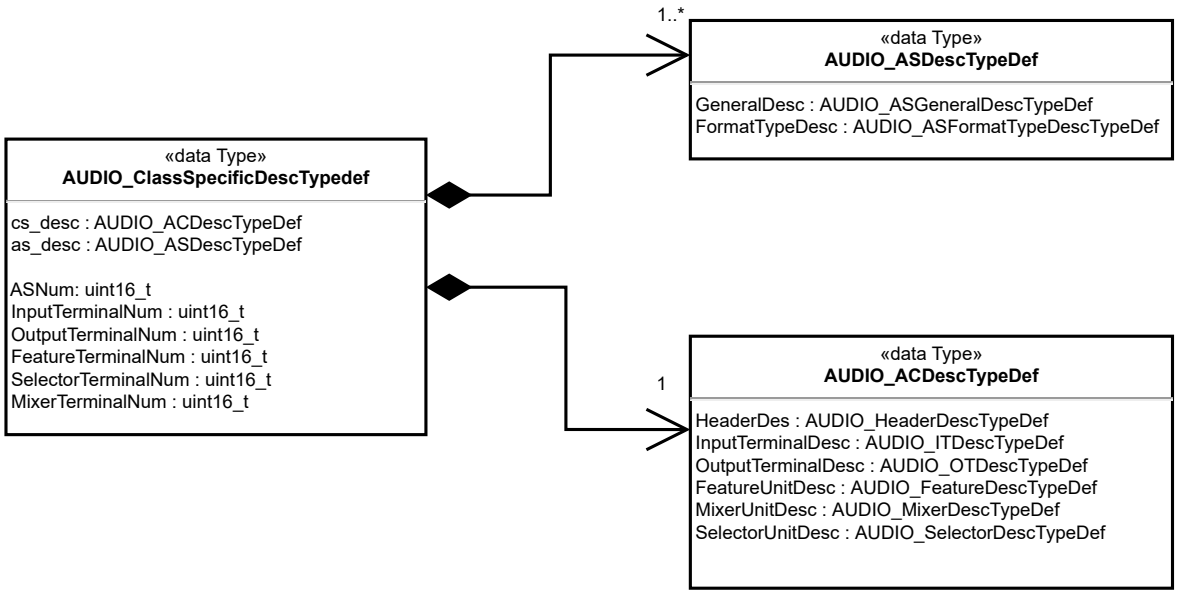


Figura 4.7: Estrutura de dados *class-specific* da Classe Áudio

O diagrama de classes da figura 4.8 mostra as classes em que se pode aceder aos diferentes dados que compõe a estrutura da classe UAC.

A "Audio_HandleTypeDef", presente na figura 4.8, define todo o processo de Áudio tais como as estruturas de dados, os descriptores, as unidades e terminais, as máquinas de estados das implementações, interfaces, e outras funcionalidades do processo áudio. Portanto, é através desta estrutura geral que se tem acesso a todas as variáveis descritas ao longo desta análise.

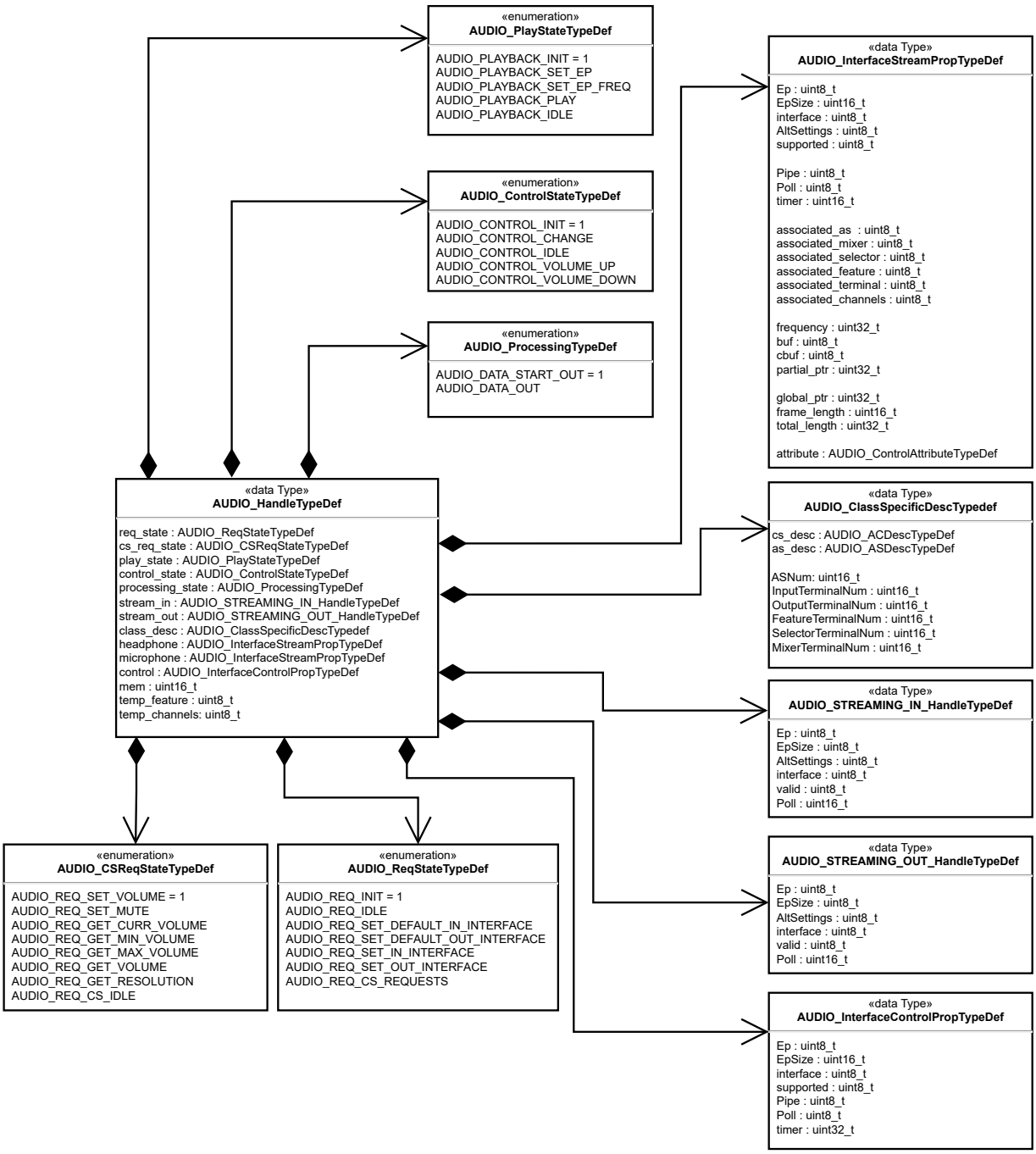


Figura 4.8: Estrutura de Dados da Classe de Áudio

4.2.2 Comportamento

Fazemos agora uma análise comportamental da classe de dispositivos áudio e, sempre que se achar relevante e de igual modo com a análise estrutural, será feita a depuração das semelhanças e diferenças entre as duas classes de estudo. A classe de áudio é inicializada pela função "USBH_AUDIO_InterfaceInit"o qual é apresentado pelo fluxograma da figura 4.9.

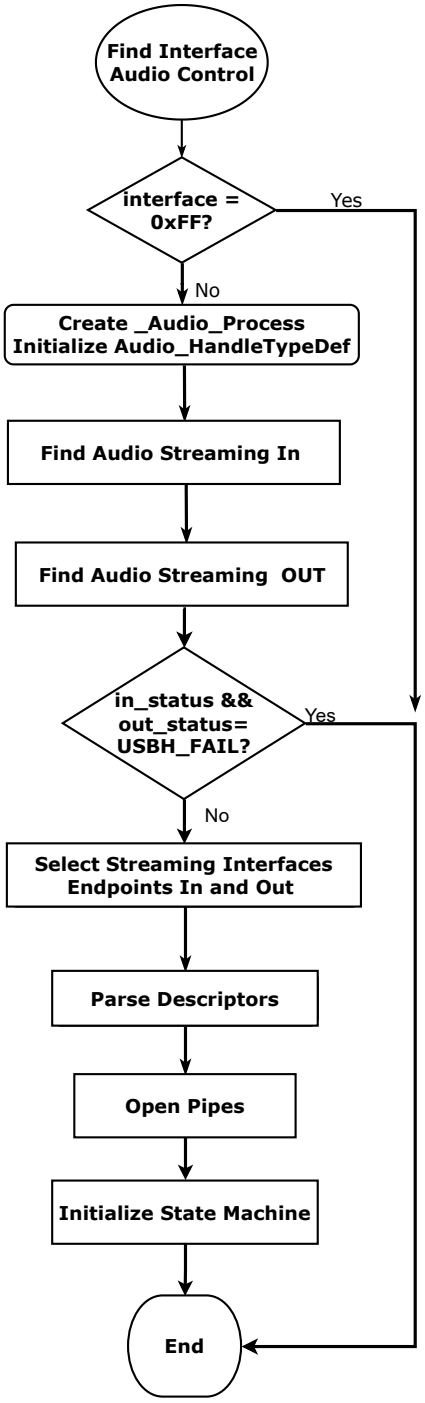


Figura 4.9: Fluxograma Interface classe Áudio

Para UVC será necessário, primeiramente, criar e inicializar um "_Video_Process"assim como "VIDEO_HandleTypeDef". Não havendo a necessidade de *streaming* de saída, visto apenas ser relevante a aquisição de imagens, este vai ser excluído, mantendo apenas interfaces de *streaming* de entrada. A selecção das interfaces será apenas adaptada à classe de vídeo, assim como a função *Parse Descriptors*, sendo que os descriptores como visto na análise anterior, não serão todos iguais.

A função USBH_AUDIO_FindStreamingIN, representada na figura 4.9 por "*Find Audio Streaming In*", possui o seu diagrama de actividade apresentada na figura 4.10.

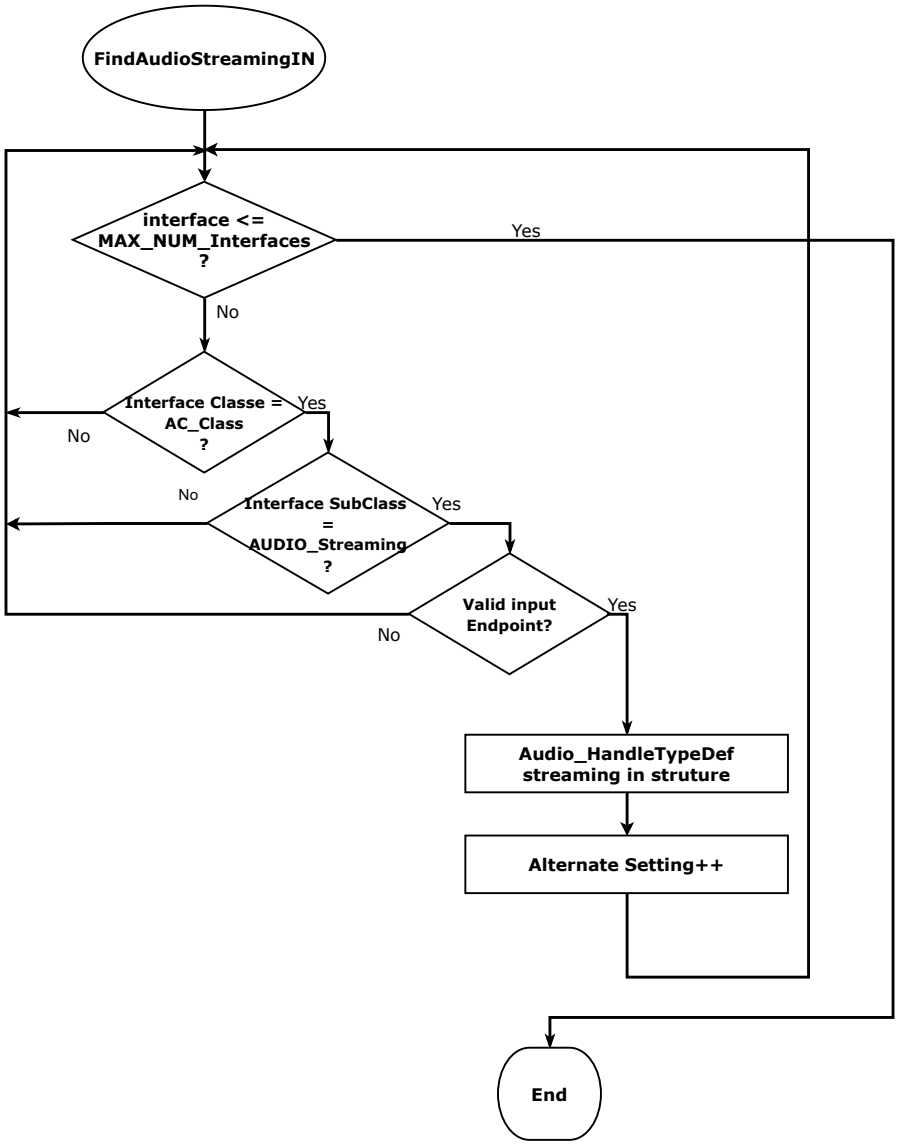


Figura 4.10: Função de Procura para Áudio Streaming de Entrada

Com implementações da estrutura de dados para UVC, e levando em conta a especificação da UVC, é possível adaptar esta implementação para fazer a procura por *streaming* de entrada de vídeos.

O diagrama de actividade da figura 4.11 apresenta os pedidos de classe (*Class Requests*) áudio

implementada pela função "USBH_AUDIO_ClassRequest". Estes pedidos configuram as unidades com interfaces específicas obtidas na inicialização da classe.

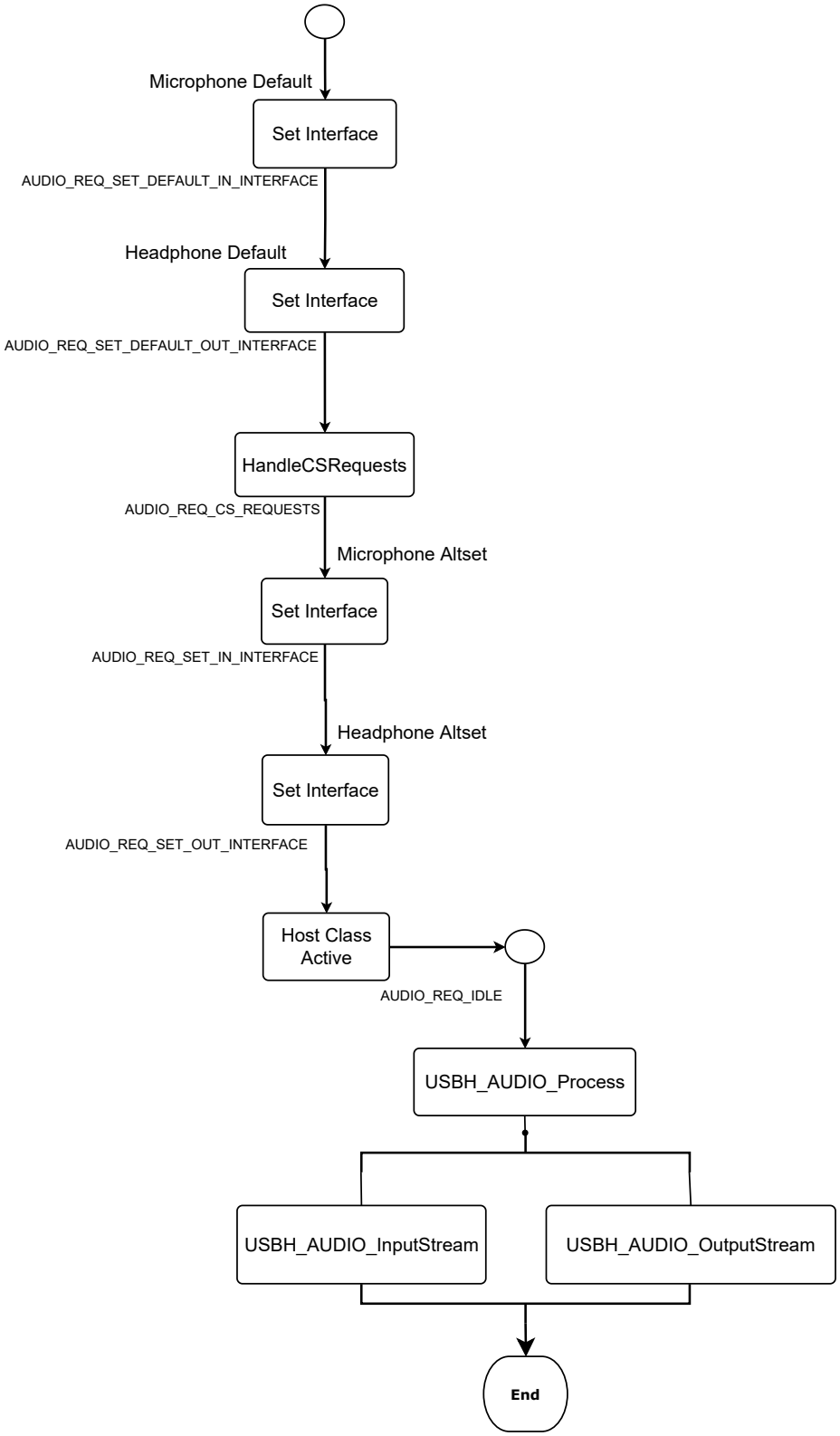


Figura 4.11: Máquina de Estados Pedidos da classe(*Class Request*) Áudio

Inicialmente, as unidades de entrada e saída (*microphone* e *headphone*) são configuradas como genéricas. O estado seguinte, a "HandleCSRequests"implementa por sua vez outra máquina de estado para pedidos específicos de classe *Class Specific Requests*. Deste modo, fica estabelecida a comunicação com o dispositivo através de pedidos *probes* e *commits*, determinando as características do dispositivo áudio.

Determinadas as características do dispositivo, são então configuradas as unidades de entrada e saída com as respectivas interfaces, representadas na figura 4.11 por "Set Interface"correspondente pela função "USBH_SetInterface". É então comunicada à camada USB que a classe está activa.

É inicializada o processo de áudio para recepção através do microfone ou transmissão, para os auscultadores, de stream de dados. A recepção é implementada pela função "USBH_AUDIO_InputStream"e a transmissão pela função "USBH_AUDIO_OutputStream".

Para este projecto apenas interessa a recepção de stream de dados para o caso da UVC, não sendo necessária a transmissão de dados. Para tal, será necessária a "USBH_AUDIO_InputStream"para UVC. No entanto, ao contrário do "USBH_AUDIO_OutputStream", a função "USBH_AUDIO_InputStream"não foi implementada. Portanto, é necessária fazer a implementação desta função para a recepção de stream de dados UVC.

4.3 Implementação USB Video Class (UVC)

Para a implementação da classe de dispositivos UVC, fazemos especificamente a necessária análise estrutural e comportamental para a implementação desta classe de dispositivos de acordo com os requisitos apresentados no capítulo 3 na secção 3 e tendo em consideração a referência da implementação da classe de dispositivos áudio da STM32 e as notas apresentadas na secção 4.2.

A estrutura da interface de Video Control (VC) desta implementação da UVC, possui para além dos descriptores descritos para a classe de áudio os descriptores de extensão (*Extension*) e de processamento (*Processing*).

4.3.1 Estrutura

A estrutura do Video Control é apresentada pela figura 4.13 com todas as unidades desta interface. O Video Control (VC) é constituído por um *Header*, pelos terminais de *input* e *output*, pela unidade de processamento e pela unidade de extensão. Apenas pode haver um *header* do VC, enquanto que os outros descriptores podem ter um ou mais interfaces. O descriptor *header* ("VIDEO_HeaderDescTypeDef") é

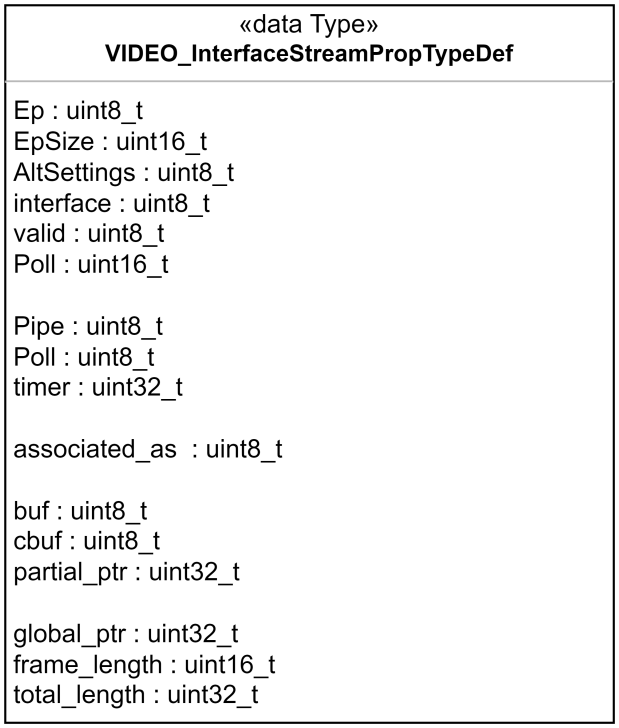


Figura 4.12: Estrutura de dados Interface de Streaming

composto por dados do tipo byte e definido pela estrutura "VIDEO_HeaderDescTypeDef". Este descriptor contém os campos que descrevem não apenas esta interface, mas todo o dispositivo. O campo "bcdUVC"define qual a especificação USB suportada pela câmara de vídeo e o campo "wTotalLenght"define qual o tamanho total de todos os descriptores do dispositivo. O descriptor de terminal de input ("VIDEO_ITDescTypeDef") fornece informações funcionais ao *host*, como o tipo de terminal ("wTerminalType"), valores máximos e mínimos de ampliação ("wObjectiveFocalLengthMax"e "wObjectiveFocalLengthMin"). O descriptor de terminal de output ("VIDEO_OTDescTypeDef") descreve quais os tipos de endpoints pelos quais é difundido o stream de dados ("wTerminalType"). O descriptor de processamento ("VIDEO_ProcessingDescTypeDef") indica os atributos de processamento (como exemplo o "bmControls"que indica certas capacidades de *streaming* da câmera de video) e o seu suporte para normas de vídeos analógicos ("bmVideoStandards"). O descriptor de extensão ("VIDEO_ExtensionDescTypeDef") indica as capacidades complementares adicionadas pelo fabricante.

A estrutura da interface VS é apresentada pela figura 4.14 com todas as unidades desta interface. O Video Streamming (VS) é constituído por um descriptor de *Header*, pelos descriptores de formato e de frame. Deve haver pelo menos um descriptor de formato para todos os formatos suportados, havendo um ou mais desciptores de frame para cada formato. Os formatos e as frames são acedidos por meio desta interface.

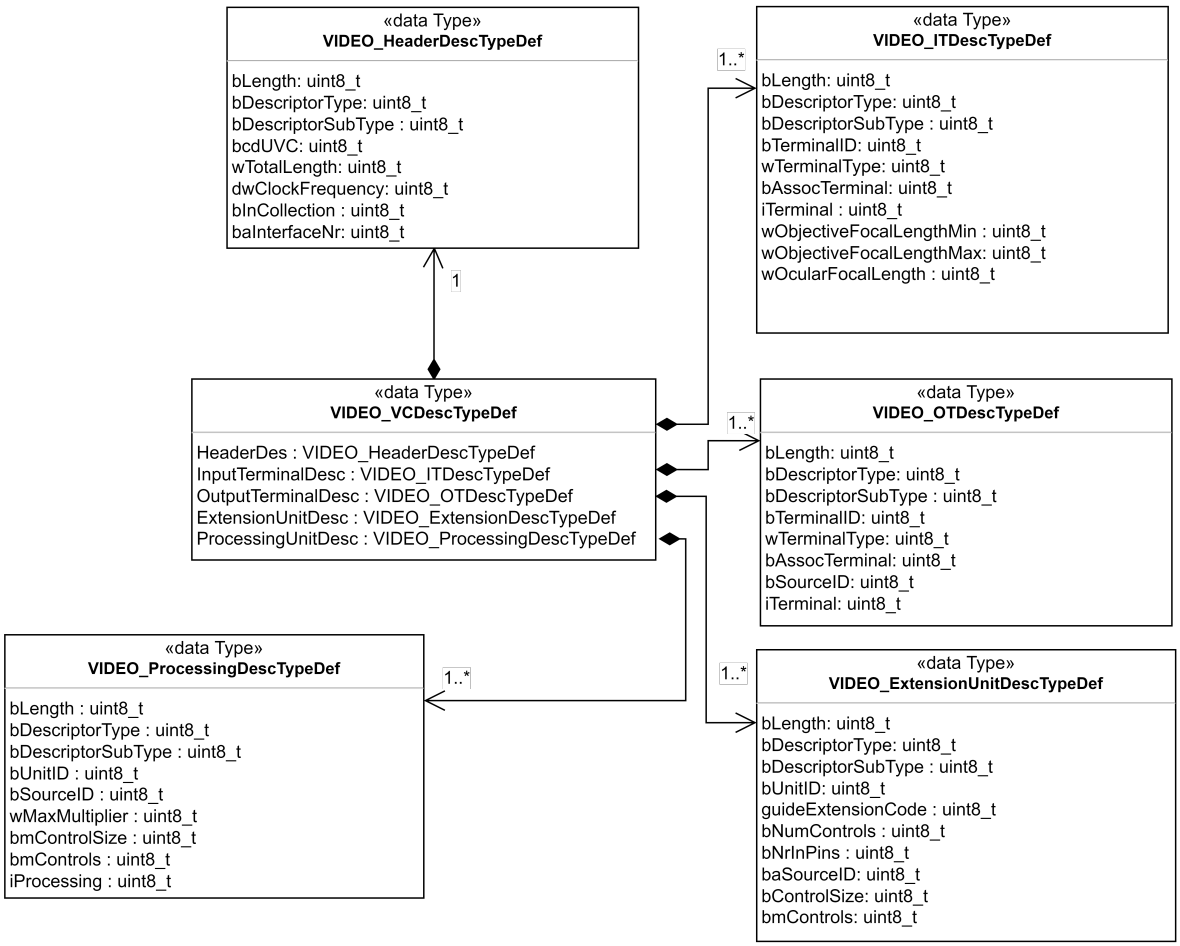


Figura 4.13: Estrutura da interface de controlo de vídeo

O descriptor de *header*, "VIDEO_InHeaderDescTypeDef"oferece as informações do streaming tais como o número de formatos suportados ("bNumFormats"), o endereço de enpoint desta interface ("bEndpointAddress"), o suporte e o método de captura de *still images* ("bTriggerSuport"e "bStillCaptureMethod"). O descriptor "VIDEO_MJPEGFormatDescTypeDef"diz respeito ao formato Motion-Joint Photographic Experts Group (MJPEG) e fornece o número de descriptores de frame deste formato ("bNumFrameDescriptors"), enquanto o descriptor "VIDEO_MJPEGFrameDescTypeDef"fornece a resolução da frame ("wWidth"e "wHeigh"), o máximo tamanho de uma frame ("dwMaxVideoFrameBufferSize"). O descriptor"VIDEO_UncompFormatDescTypeDef"e o "VIDEO_UncompFrameDescTypeDef"são os descriptores de formato descomprimido e da frame descomprimida. Estes fornecem os mesmos campos apresentados para MJPEG.

A estrutura da figura 4.15 apresenta os descriptores de interfaces VC ("VIDEO_VCDescTypeDef") e VS ("VIDEO_VSDescTypeDef") assim como a quantidade de cada tipo de descriptores adquiridos. Por exemplo, existem descriptores que são únicos (para cada dispositivo diferente haverá apenas um descriptor de

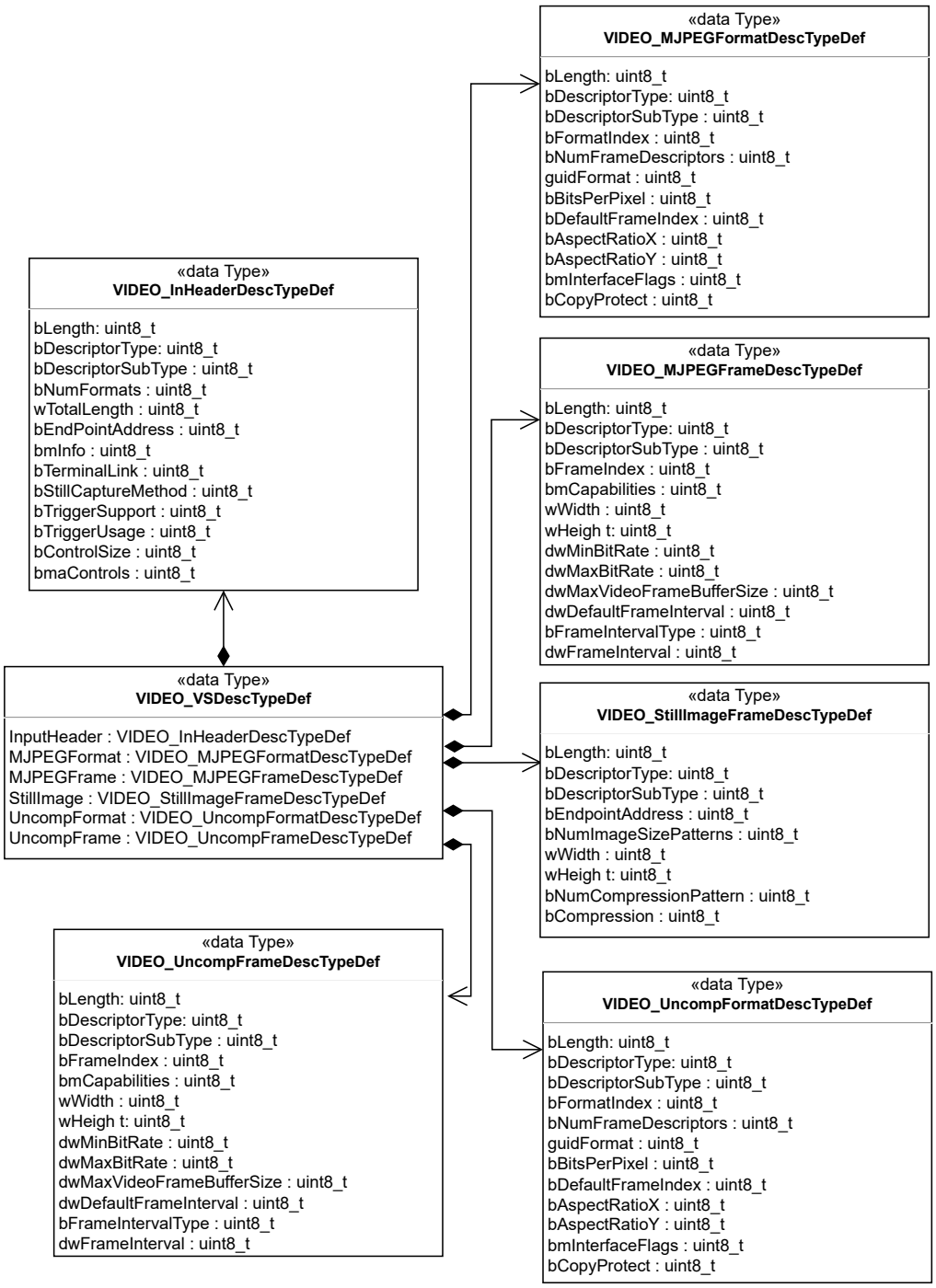


Figura 4.14: Estrutura da interface de streaming de vídeo

VC) e a quantidade de descriptor adquirido é de apenas um. No entanto poderão existir vários descriptores de VS, assim como de Formato e de *Frame*.

A estrutura "VIDEO_HandleTypeDef"é usada para todos os controlos da classe de dispositivos de vídeo UVC, o qual contém os descriptores *class-specific*, o controlo da interface, a interface de streaming e

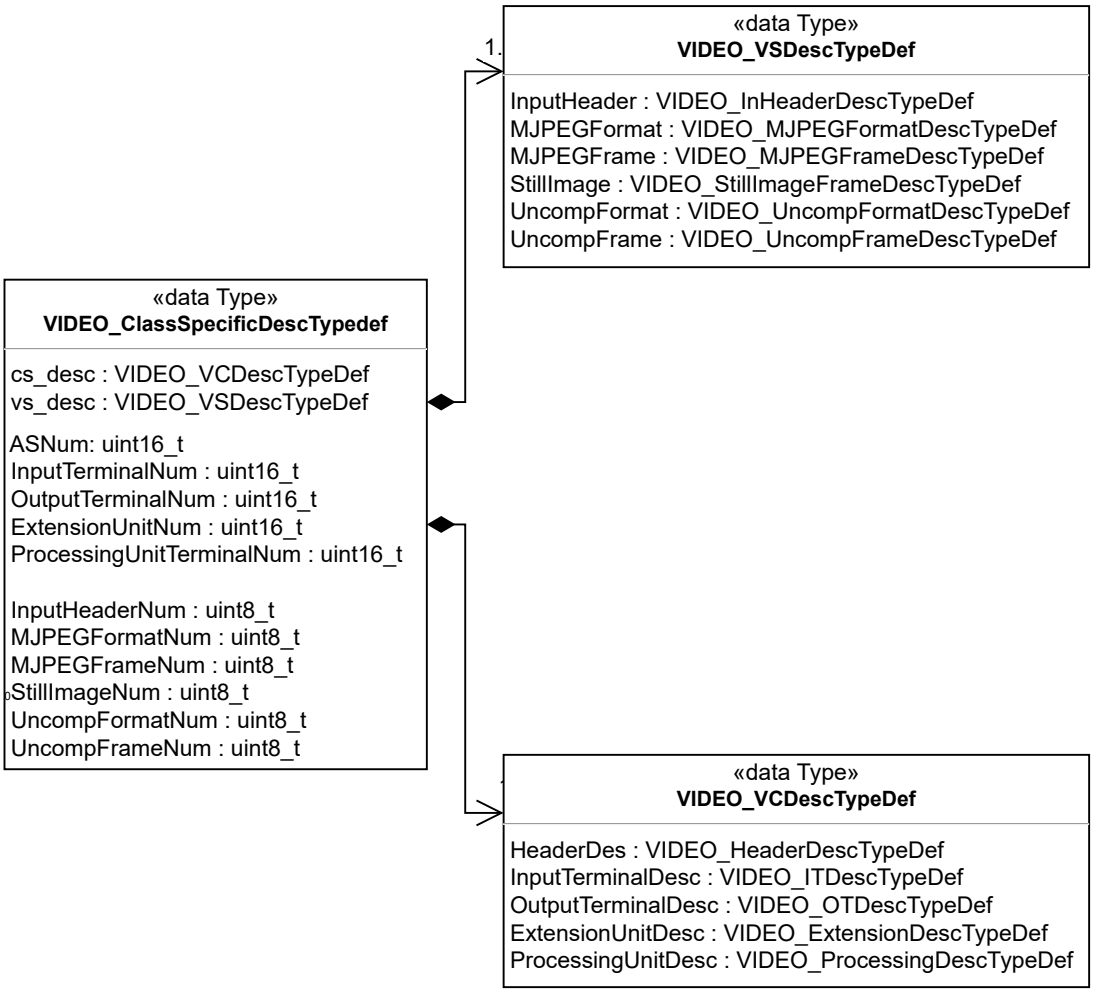


Figura 4.15: Estrutura *Class-Specific* da classe de Vídeo

as estruturas de controlo de estado das máquinas de estados da classe.

A figura 4.16 apresenta o diagrama de classes da USB Video Class (UVC) para a STM32 desenvolvido com base no exemplo da API da classe de dispositivos áudio. Este diagrama de classes contém a classe de dispositivos UVC com todos os descriptores de Controlo e de Streaming apresentados ao longo desta exposição (figuras 4.13 e 4.14), o endpoint de interface (figura 4.12), o streaming de entrada e as estruturas de estado ("VIDEO_ControlStateTypeDef", "VIDEO_StreamStateTypeDef", "VIDEO_ReqStateTypeDef"e "VIDEO_StateTypeDef").

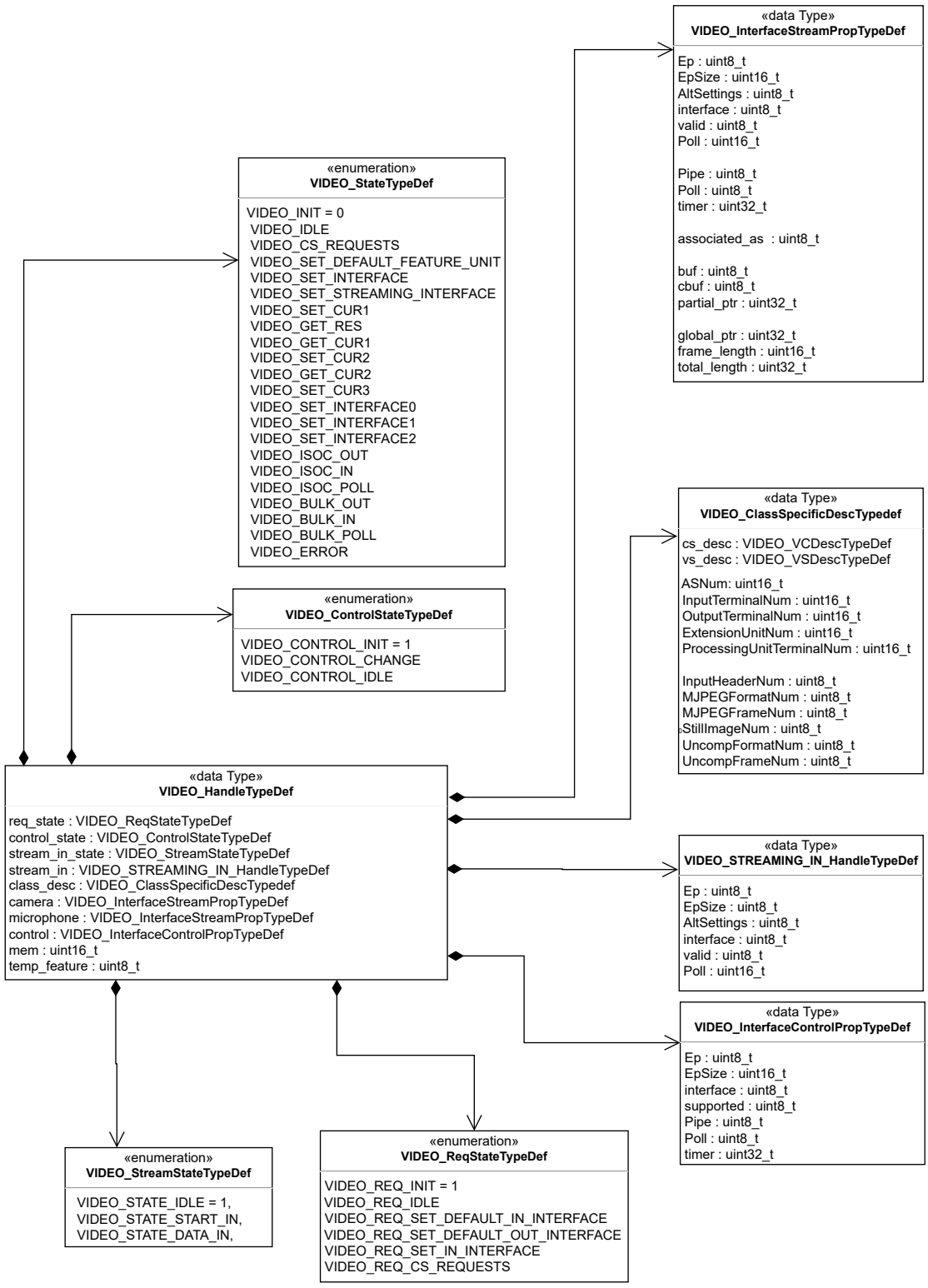


Figura 4.16: Estrutura de Dados da Classe de Vídeo

4.3.2 Comportamento

A aplicação a ser desenvolvida tem as funcionalidades especificadas na secção 3.1, não havendo interesse na implementação de todas as possíveis funcionalidades e potencialidades da classe de dispositivos UVC.

Primeiramente, a implementação da inicialização da classe de dispositivos vídeo "USBH_VIDEO_InterfaceInit".

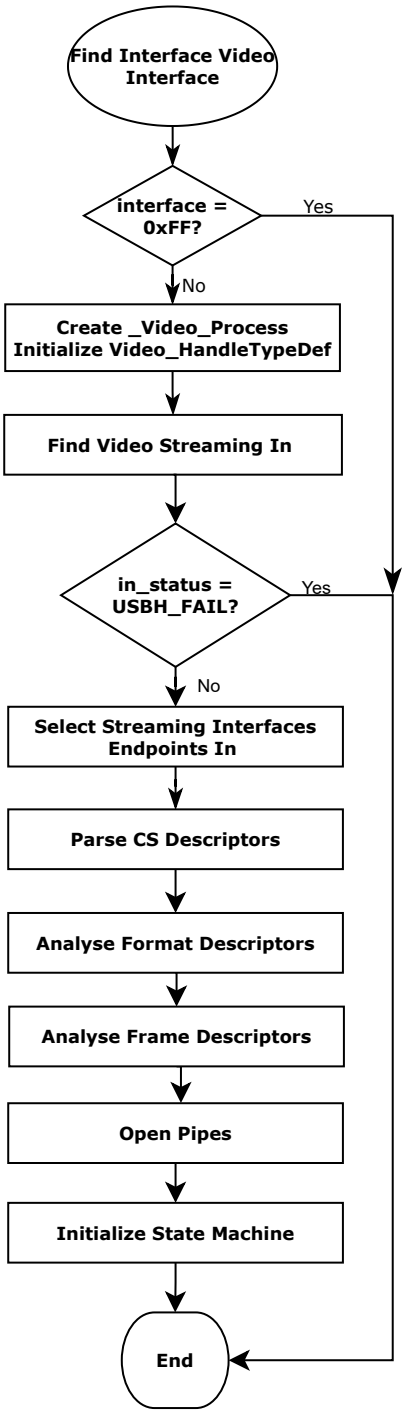


Figura 4.17: Função de Início Interface de Vídeo

A função "USBH_VIDEO_InterfaceInit", representada pelo fluxograma da figura 4.17, é muito semelhante ao visto anteriormente na secção 4.2, para a classe de dispositivos Áudio (figura 4.9). Os blocos "Analyse Format Descriptors"e "Analyse Frame Descriptors"são implementadas para a verificação dos formatos e frames pretendidos pela aplicação e a sua disponibilidade no dispositivo.

A função de procura de streams de entrada *Find Video Streaming In* é análoga à implementação da classe de dispositivos áudio apresentada na figura 4.10.

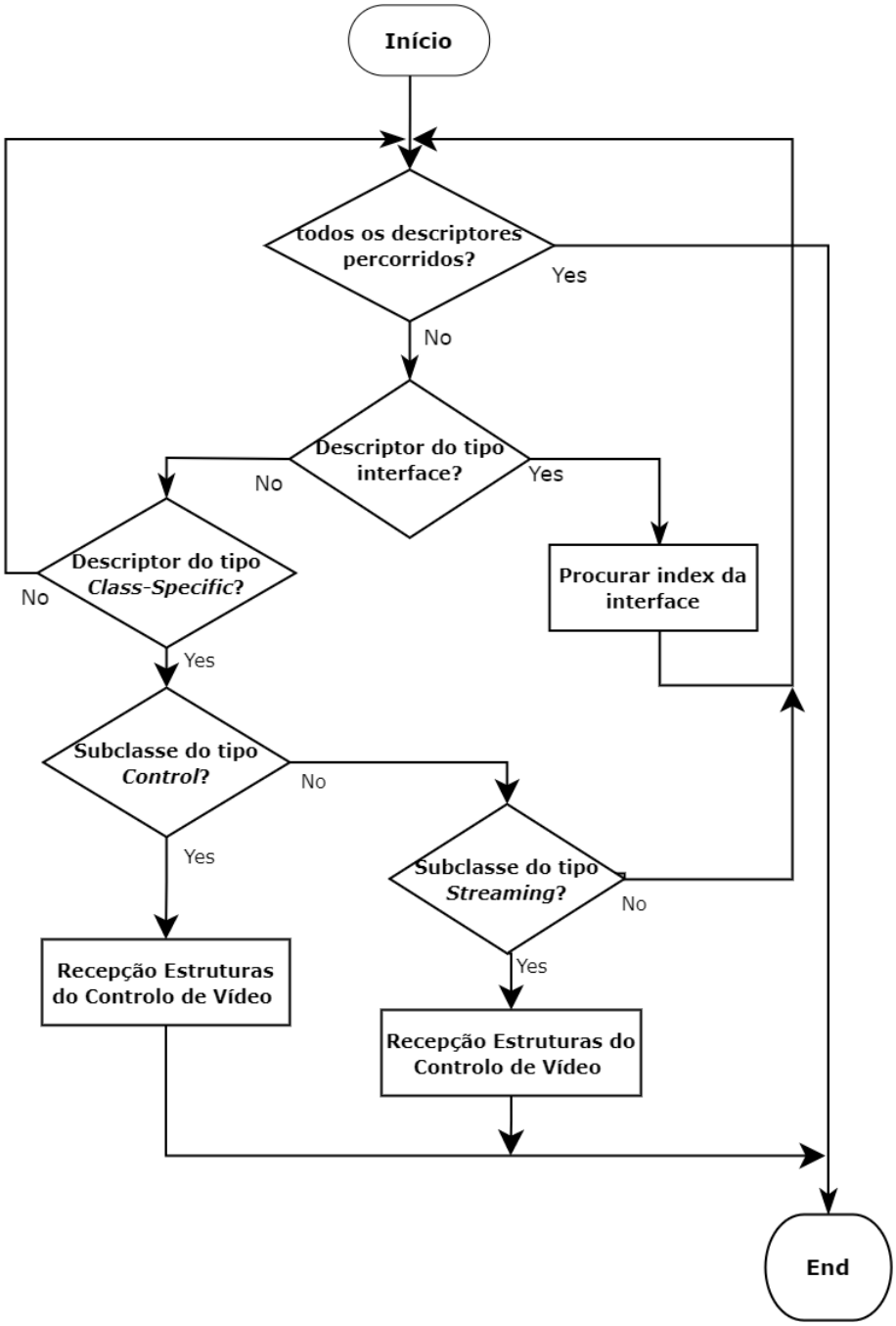


Figura 4.18: Análise Descriptores *Class-Specific*

A função de "Parse Descriptors"faz a decomposição do descriptor global armazenado pela API para descriptores específicos da classe de dispositivos. A função percorre todos os descriptores preenchendo as estruturas de dados *Class-Specific*. Também armazena os índices das interfaces encontradas. O fluxograma da figura 4.18 representa este processo. A especificação USB para dispositivos de vídeo determina quais as sub-classes de *videostreaming* e de *videocontrol* da interface da classe.

A necessidade da análise de formato e de frame deve-se ao facto das capacidades de formato e das frames requeridas pelo host serem suportadas pelo dispositivo. Caso o formato seja suportado pelo dispositivo UVC a variável "bFormatIndex"é actualizada para o valor do descriptor de formato de vídeo.

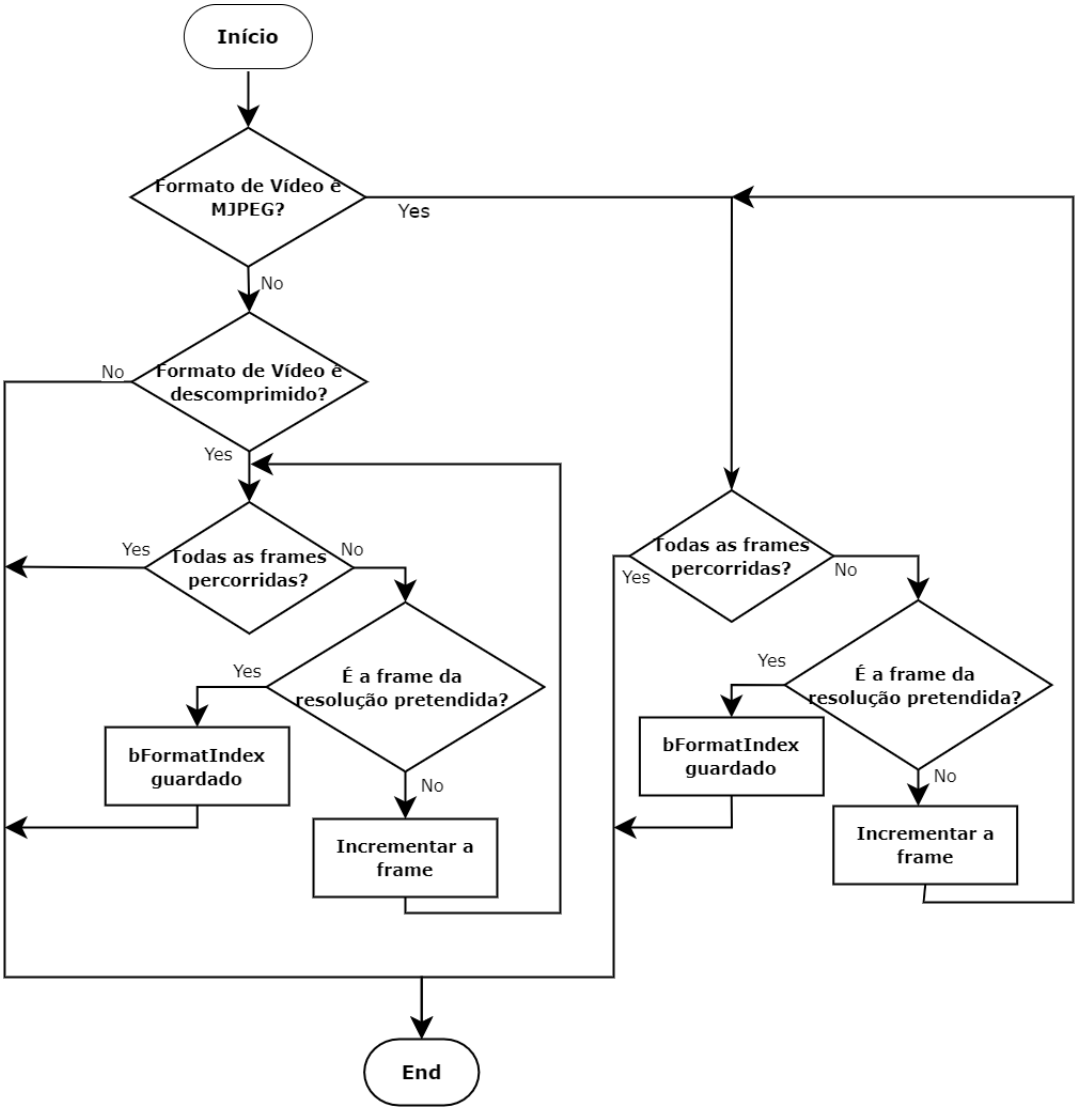


Figura 4.19: Fluxograma da análise dos descriptores de formato

Caso a frame seja suportada pelo dispositivo UVC a variável "bFrameIndex"é actualizada com o valor do descriptor de frame de vídeo recebido (MJPEG ou descomprimido). Caso o formato ou a frame não seja suportado a comunicação com o dispositivo é invalidada. A figura 4.19 representa o fluxograma da função "USBH_VIDEO_AnalyseFormatDescriptors"e a figura 4.20 representa o fluxograma da função "USBH_VIDEO_AnalyseFrameDescriptors".

Os pedidos de classe são implementados de forma diferente da UAC, em grande parte simplificando a implementação de acordo com os requisitos para a nossa classe UVC. Sendo usada apenas um terminal de entrada é apenas configurado o respectivo terminal de entrada. De igual modo o processo de captura necessita apenas da recepção de stream de entrada, representado pela função "USBH_VIDEO_InputStream"observado na figura 4.21. Este processo consiste na recepção periódica de pacotes de streaming, através de pedidos de recepção de dados.

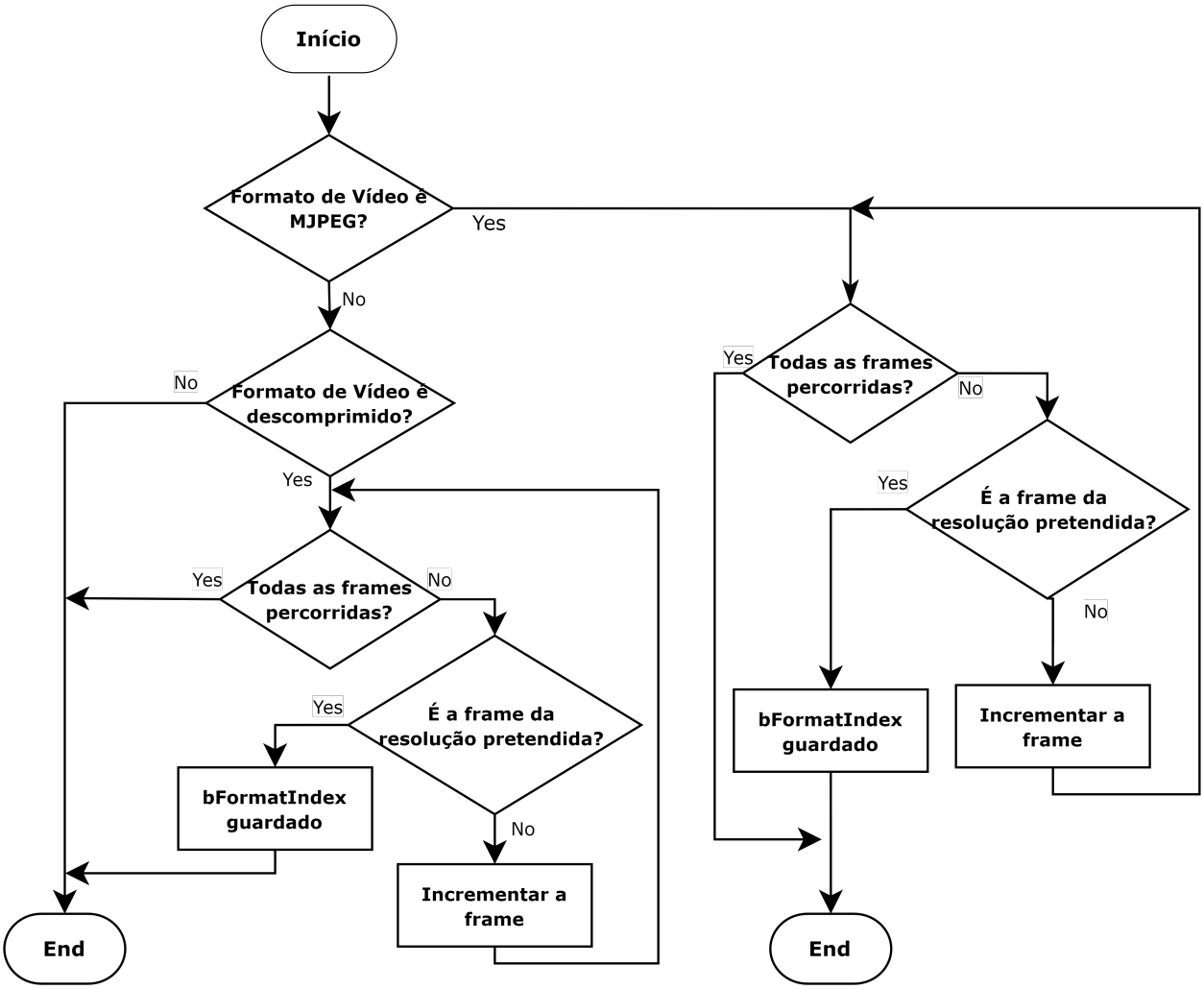


Figura 4.20: Fluxograma da análise dos descriptores de frame

Como é visto na classe de dispositivos áudio e apesar de se referenciar o "USBH_AUDIO_InputStream", não existe a implementação desta função nesta API. No entanto, a API da camada do USB oferece funções tais como a "USBH_IsocReceiveData", "USBH_LL_GetURBState"e "USBH_LL_GetLastXferSize"que permitem fazer a recepção e monitorização dos dados transferidos no barramento. Ainda, a função "USBH_IsocReceiveData"emite *tokens* do tipo IN e é responsável pelo pedido de envio de novos pacotes de dados. A implementação desta função para vídeo é mostrada na figura 4.22.

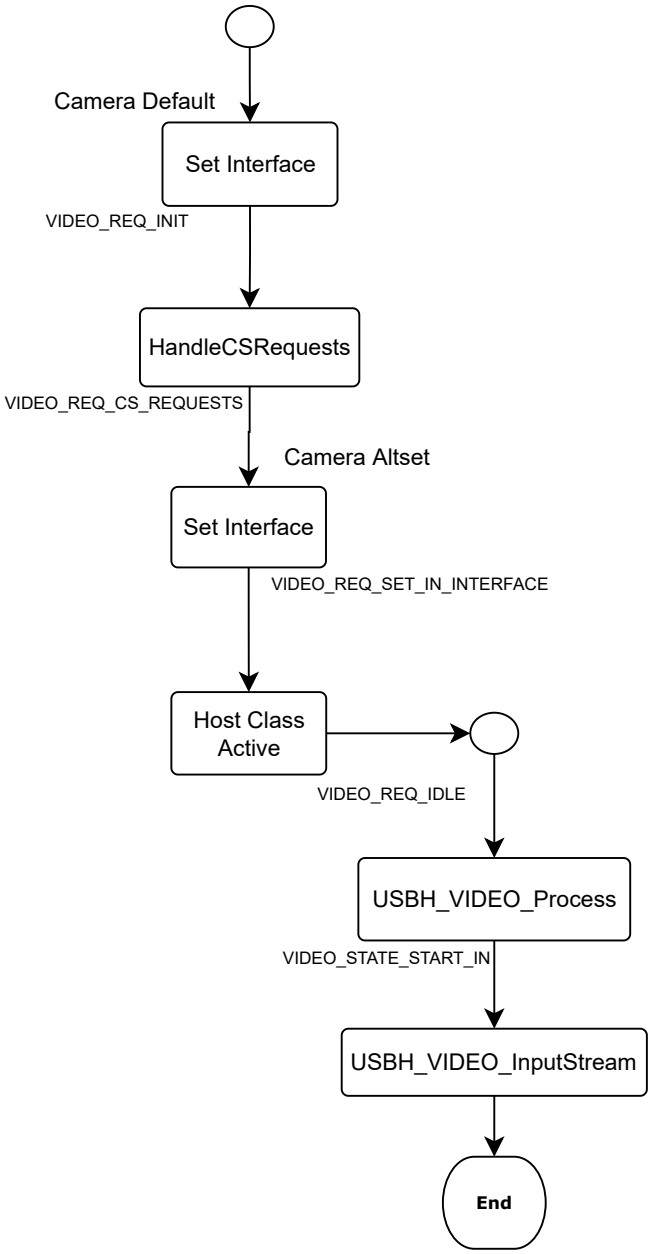


Figura 4.21: Pedidos de Classe - *Class Requests* Vídeo

A máquina de estados da classe de pedidos (*State Machine*) é representado na figura 4.21 com as

configurações de interface (*Set Interface*) e pedidos *Class-Specific*. Após a configuração da interface para os parâmetros correctos é inicializada a STM32 como Host com a classe de vídeo activa (*Host Class Active*). A partir deste instante é possível iniciar o processo de captura de vídeo ("USBH_VIDEO_Process") com a recepção do stream de dados ("USBH_VIDEO_InputStream").

A figura 4.22 apresenta o diagrama de sequências da função "USBH_VIDEO_InputStream"no qual é possível observar as chamadas aos módulos implementados pela classe USB *Host*.

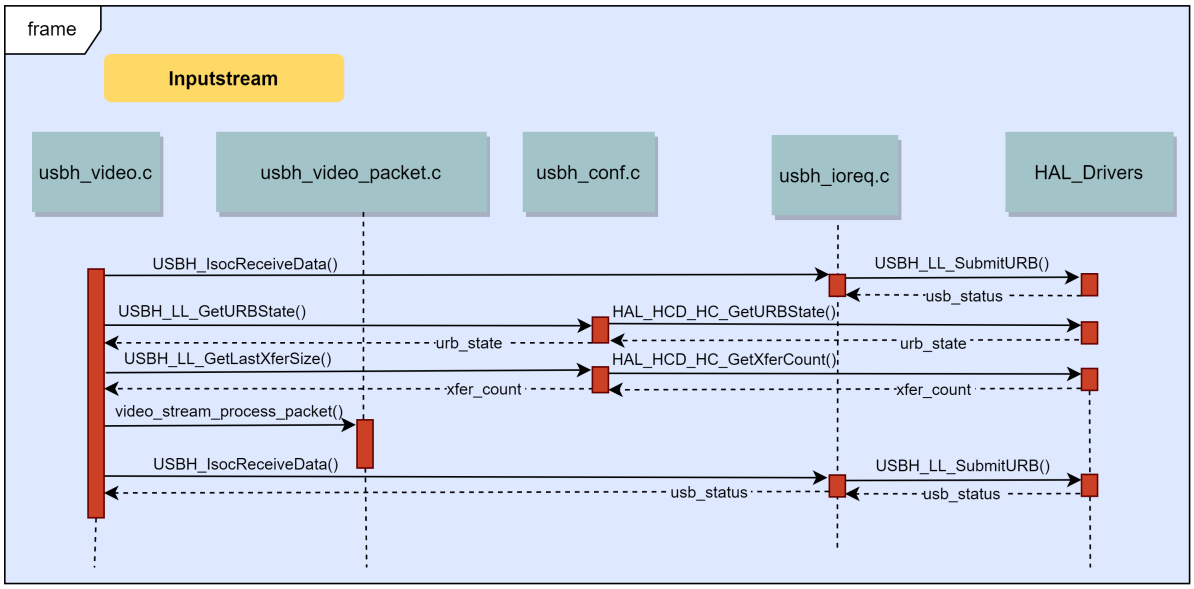


Figura 4.22: Diagrama de Sequências da função *Inputstream*

A implementação da classe UVC é apresentada no diagrama de classes UML da figura 4.23 na qual optou-se por dividir a classe em três sub-classes diferentes. A classe principal define todas as ocorrências dos processos da classe, assim como as funcionalidades oferecidas, a classe de descriptor implementa as funções de análise dos descriptores e das estruturas da classe e de análise de frame e de formato, e, por fim, a classe de pacote que implementa o processo de captura, a actualização do estado da captura e o armazenamento dos pacotes da captura em um buffer.

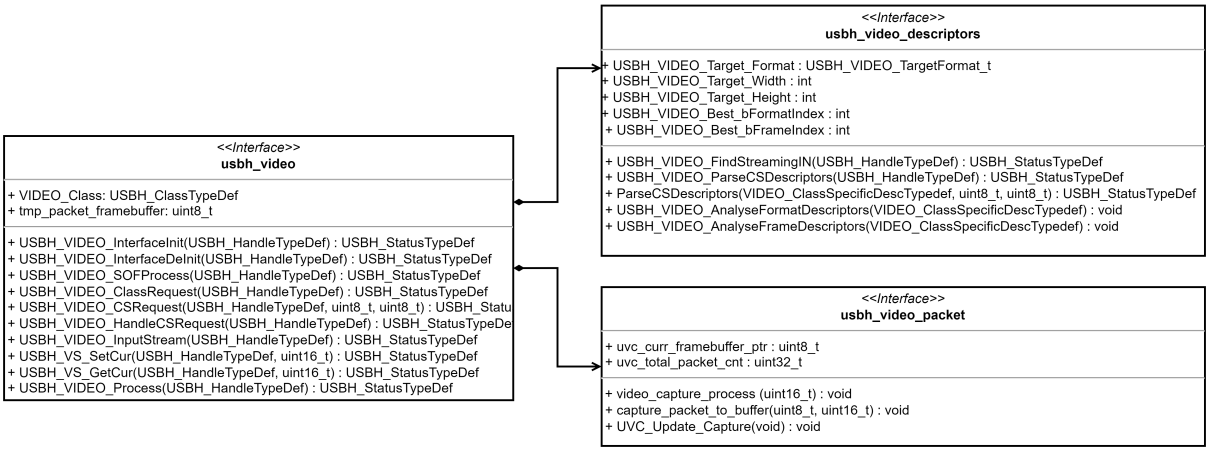


Figura 4.23: Classe Vídeo

4.4 Relógio e Gestão de Energia

O periférico RTC é um contador de tempo real com suporte para calendário com segundos, minutos, horas, dia da semana, data, mês e ano. É fornecido a este clock uma frequência de 32 kHz e é feita a determinação dos prescaler para a frequência pretendida. Também possui um factor de correcção e de calibração. Este periférico determina a frequência de amostragem que pode ser configurável e é o periférico que dá o sinal de *Wakeup* do modo *low power* para a gestão de energia.

O registo de backup usado para guardar o estado do RTC é o RTC_BKP_DR1 do banco de registos deste periférico, presente na memória de backup da SRAM. Os modos de configuração do prescaler determinam o intervalo de tempo do *auto-wakeup* de 1 s a 36 horas, caso a frequência do periférico for 1 Hz.

$$f_{CK\_SPRE} = \frac{f_{RTCCLK}}{(PREDIV\_S + 1) + (PREDIV\_A + 1)}$$

Para além do RTC, foi configurado um timer *general purpose* para as chamadas do processo de UVC, a cada microframe, para a captura dos pacotes de dados em HS.

Optou-se pelo modo *standby*, que é o modo de menor consumo. O modo de *wakeup* escolhido foi através de um evento ou interrupção gerada pelo RTC no modo de *auto-wakeup*. Este modo oferece uma grande variedade de possíveis tempos de aquisição.

O intuito para a aplicação em causa é passar o mínimo de tempo indispensável para a captura das

imagens e assim ter um tempo programado para *auto-wakeup* elevado. Ou seja, mais tempo em modo de *standby* e menos no modo de *run*, portanto menor o consumo total do sistema e maior a autonomia. Segundo da tabela A.1 fornecida pela DS11532, o consumo máximo à temperatura ambiente de 25 °C, com o RTC activo, é sempre inferior a um máximo de 8 μ A. Para optimizar ao máximo esta gestão de energia do sistema, alguns cuidados são tidos em consideração. Desligar os clocks de periféricos sempre que não sejam necessários e a configuração dos GPIO não utilizados no modo analógico são algumas medidas que reduzem o consumo energético do sistema.

4.5 Memória e Armazenamento de Dados

Como referido a interface com o cartão SD foi feita através do periférico SDMMC com o protoloco SDIO. Foi ainda usada um *socket* para o SD da Waveshare, o qual possui resistências de pull-up nas linhas de comunicação, com valores de 10 kΩ. A figura 4.24 apresenta este *socket*, com as entradas para cartões SD e microSD e o esquema de ligações com a placa da STM32.

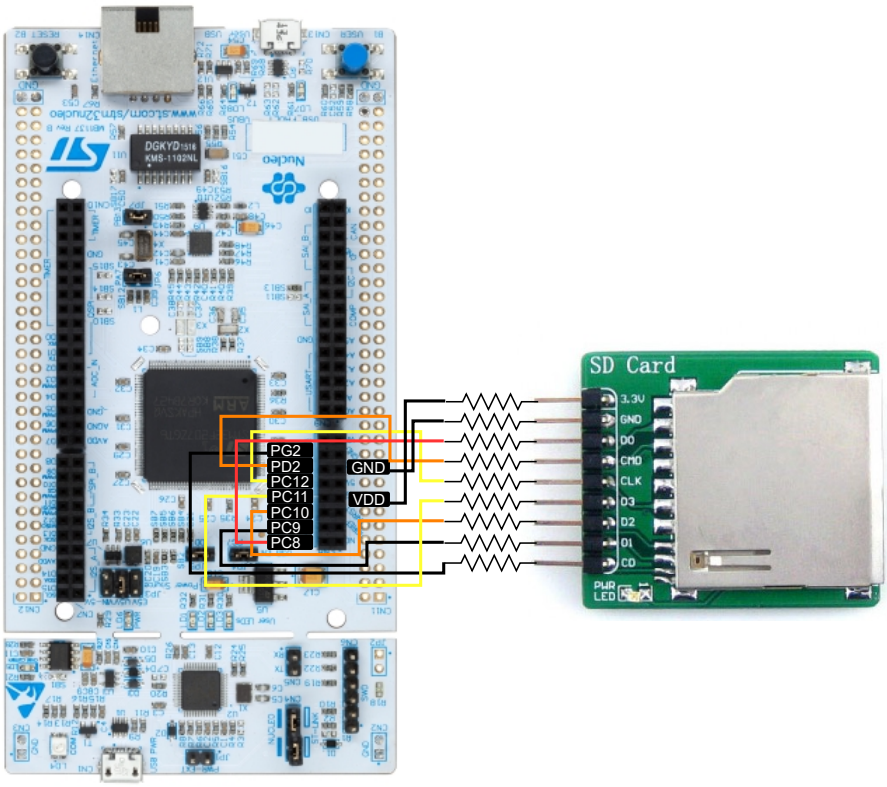


Figura 4.24: Conexão socket SD à STM32

O sistema de ficheiros é o FAT16. O armazenamento das imagens é feita em directórios diferentes, que são organizados consoante o instante da captura. Portanto, as imagens que forem recolhidas no mesmo dia são guardadas no mesmo directório e o nome dos ficheiros é baseado nas horas e minutos em que foram adquiridas, facilitando a posterior análise dos dados capturados. A figura 4.25 exemplifica a descrição.

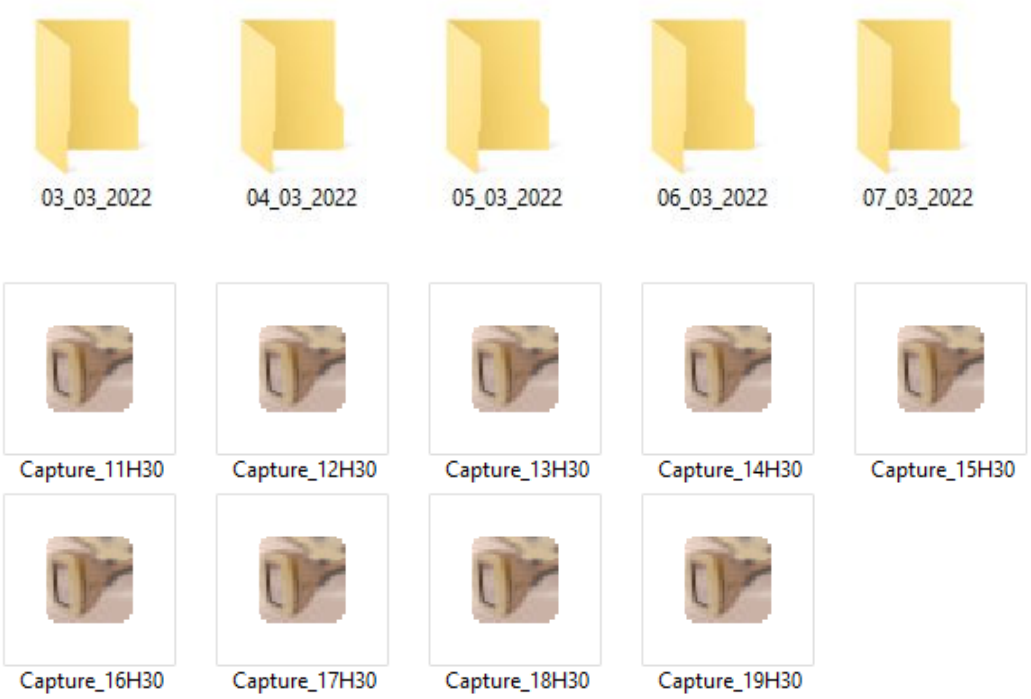


Figura 4.25: Sistema de Ficheiros e Directórios em SD

O DMA é usado para acelerar as transferências sem a intervenção do CPU. É possível assim efectuar transferências em HS de dados entre periféricos e a memória. Este periférico é usado para a comunicação com o periférico SDMMC. Sendo a comunicação bidireccional, são configurados dois *streams* de dados de um canal do DMA2 e os dados são transmitidos em transferências do modo *burst* com tamanho de quatro *words*.

4.6 Protótipo de Hardware e comando de iluminação

Devido às condições inerentes às águas profundas e à natureza do sistema proposto, é necessário que se tenham condições óptimas para a aquisição das imagens. De modo a implementar uma luz *flash*, foi escolhido um módulo de 20 LEDs e uma alimentação de 4.5 V. O consumo de corrente medido foi de 320 mA. A luz de *flash* foi ligada durante a captura da imagem, por um curto período voltando logo de seguida a ser desligada.

A figura 4.26 exemplifica a electrónica para este circuito, simplificando o número de LEDs utilizados. Os LEDs são accionados pelo comando de uma GPIO (PE.0) da plataforma de desenvolvimento.

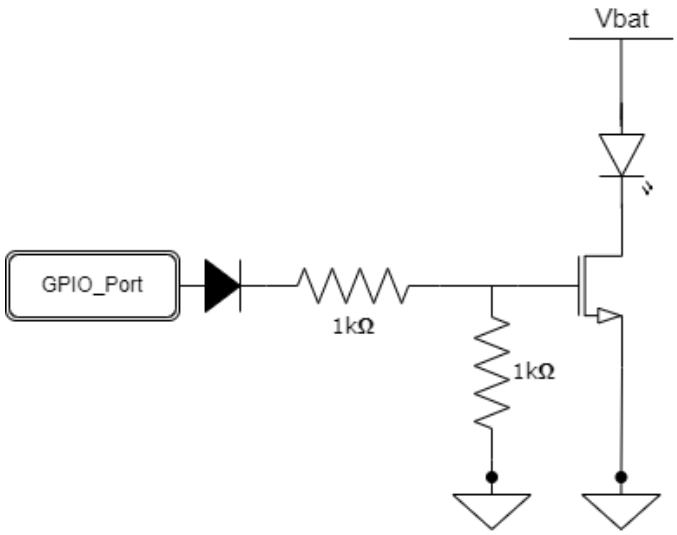


Figura 4.26: Comando da iluminação de LEDs

A figura 4.27 apresenta o protótipo desenvolvido com todos os módulos e partes constituintes nas quais destacam-se: 1 - Câmara de vídeo, 2 - Módulo LED, 3 - Placa de Processamento STM32, 4 - *Socket* SD, 5 - Circuito de Comando, 6 - PHY externo e 7 - Adaptador USB.

As ligações elétricas entre os variados módulos são todas soldadas, com a excepção à conexão da câmara de vídeo ao PHY externo.

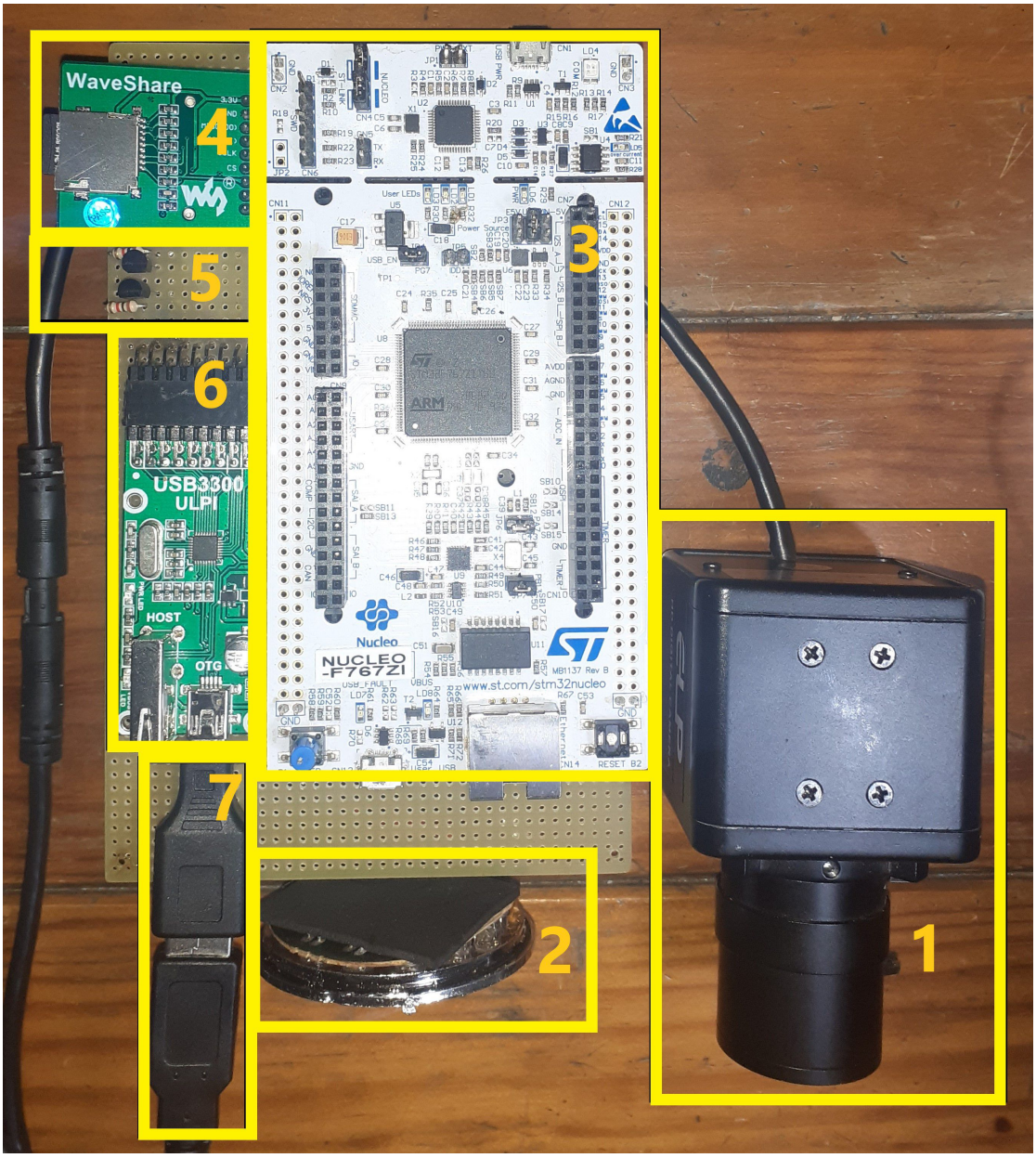


Figura 4.27: Protótipo do sistema

4.7 Aplicação de Software

Nesta secção é feita a descrição do software da aplicação. O formato de imagens escolhido para a aquisição foi o YUV descomprimido, devido às vantagens de eficiência e da qualidade de imagens dos formatos descomprimidos em relação aos formatos comprimidos. Estas vantagens são derivadas ao facto de que nos formatos descomprimidos não haver perda de dados recolhidos pela câmara de vídeo.

Um dos aspectos importantes para a implementação da comunicação USB e do protocolo UVC é a habilitação do cache de dados e de instrução do CPU de modo a aumentar as velocidades da execução do

programa. No entanto, como visto no subcapítulo anterior, o sistema também faz o uso do periférico DMA. A API da FATFS da STM32, que implementa as escritas na memória externa do módulo SD, faz a gestão necessária de modo a evitar conflitos do periférico SDMMC com a memória cache activada, invalidando a memória cache para determinadas operações.

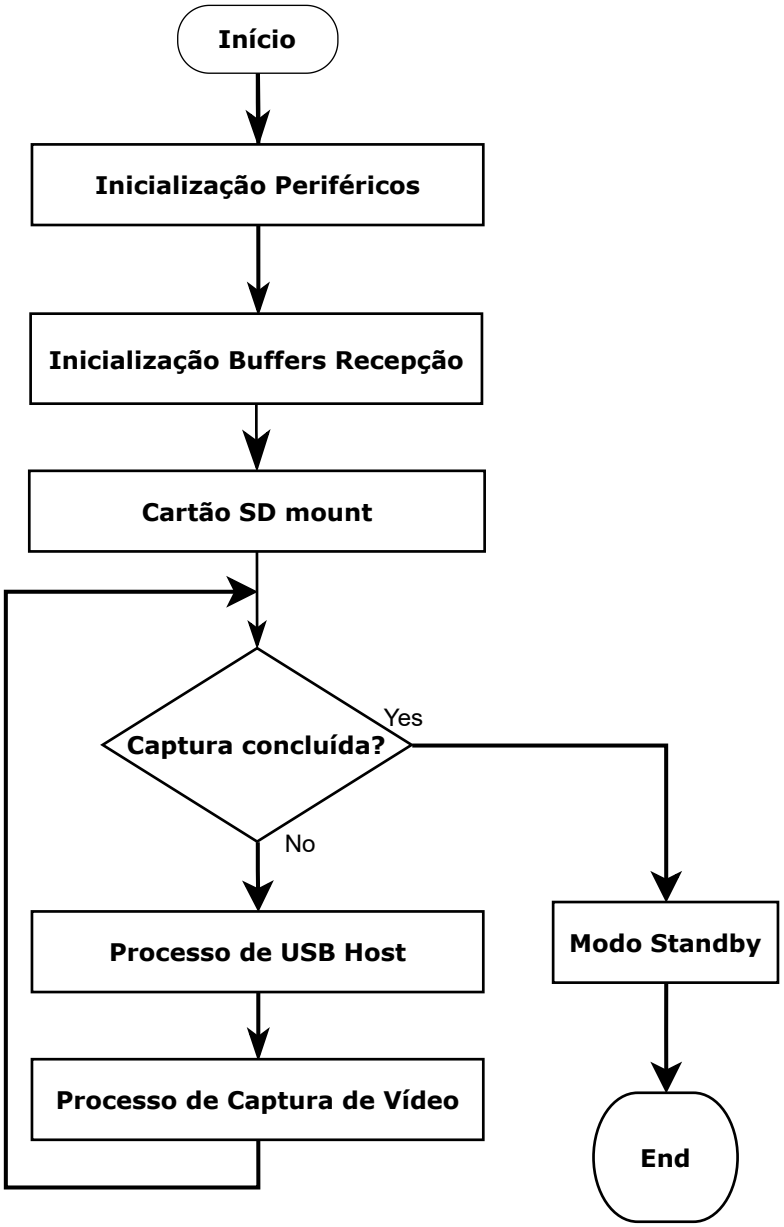


Figura 4.28: Fluxograma da Aplicação

O fluxograma da figura 4.28 descreve a camada da aplicação. No arranque, são inicializados os periféricos descritos nas secções 4.4, 4.5 e 4.6 e outros.

A inicialização dos periféricos consiste na configuração dos módulos de GPIO, do DMA, do timer 3 e RTC, do SDMMC e do FATFS e também na inicialização da interface USB. A inicialização dos *buffers* de recepção é feita por intermédio da alocação do espaço de memória necessário para a recepção de dados. Após a inicialização do periférico SDMMC e FATFS é emparelhado a plataforma de processamento e o cartão SD.

O processo de host USB implementado pela camada de *middleware* USB possui a sequência como está demonstrada no diagrama da figura 4.29.

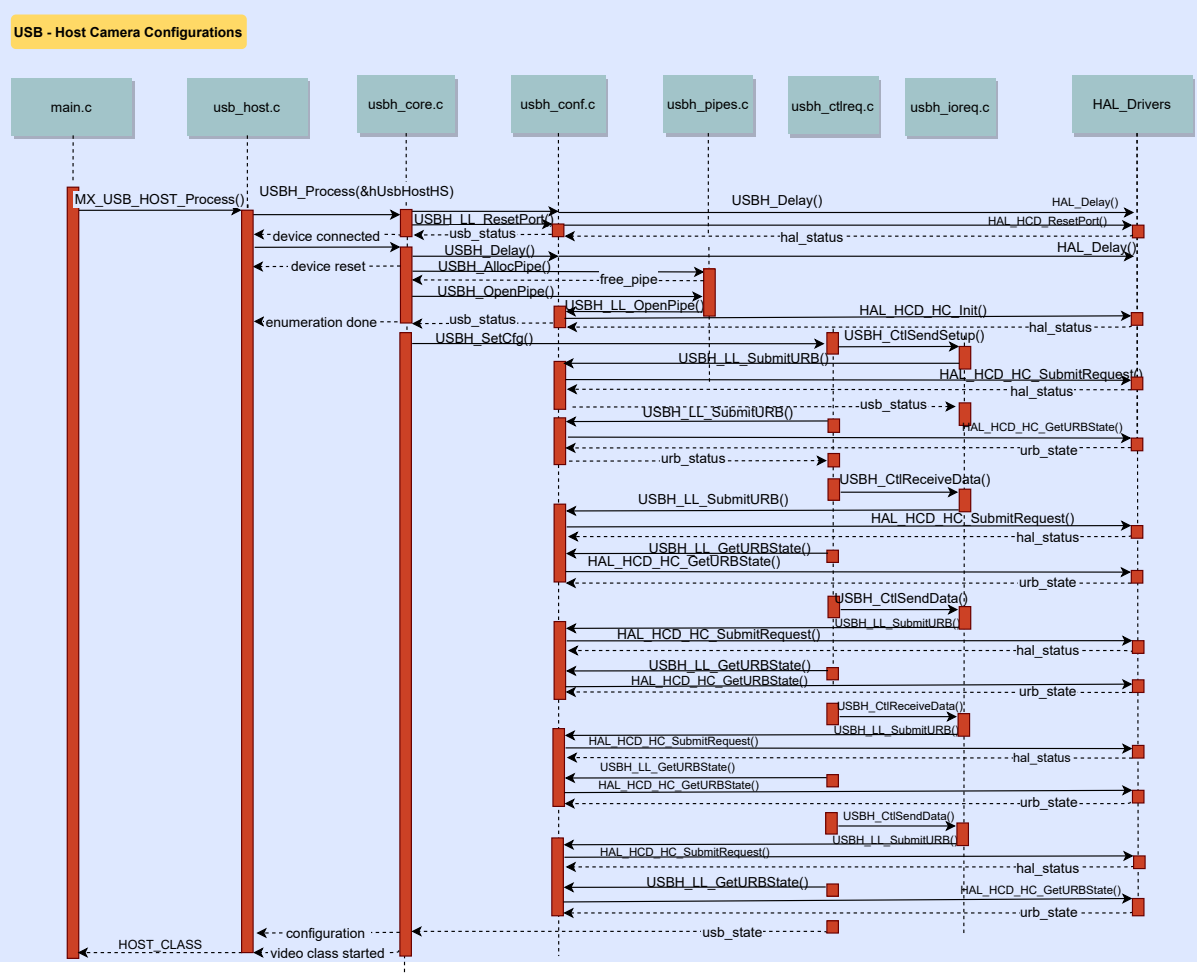


Figura 4.29: Diagrama de Sequência do processo USB

O diagrama de sequência apresentado elucida todo o comportamento da classe USB, com a inicialização dos periféricos, a conexão, o reset e a enumeração do dispositivo. Depois da correcta comunicação, é inicializada a classe de dispositivos de vídeo e o microcontrolador é associado como o host da comuni-

cação ("HOST_CLASS").

Após a completa inicialização do dispositivo e as configurações do *middleware* do processo da USB, é executada a máquina de estados da aplicação até que seja completada a captura de uma frame. Caso a captura tenha sido concluída e a máquina de estados completada, o sistema entra em modo de standby.

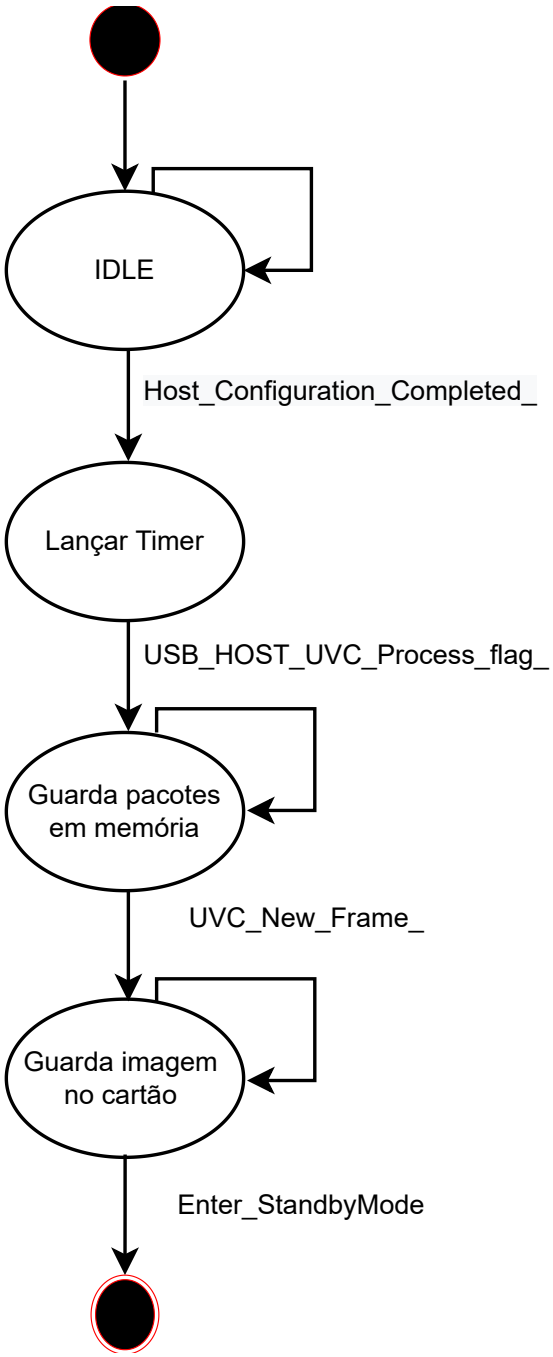


Figura 4.30: Diagrama de Actividades do Ciclo Principal de execução da Aplicação

As figuras 4.30 e 4.31 descrevem o ciclo principal do software da aplicação. Na figura 4.30, apresenta-se o diagrama de estados e o seu funcionamento lógico. A figura 4.31 descreve o ciclo temporal das actividades. O estado inicial do host é *Idle*, no qual o processo USB apresentado na figura 4.29 é chamado. A transição para o estado seguinte, é dado por meio da *flag* "Host_Configuration_Completed". Após a configuração do host como activo é avançado para o estado de espoletar o timer o qual é feita uma única vez e a *flag* de captura é activada. O estado seguinte é a recepção dos pacotes de dados em memória, a qual é repetida inúmeras vezes, até que todos os pacotes da frame sejam adquiridos. Por fim, o estado "Guarda imagem no cartão"consiste no armazenamento da imagem completa no cartão de memória e, completada esta tarefa entra-se em modo de *sleep*.

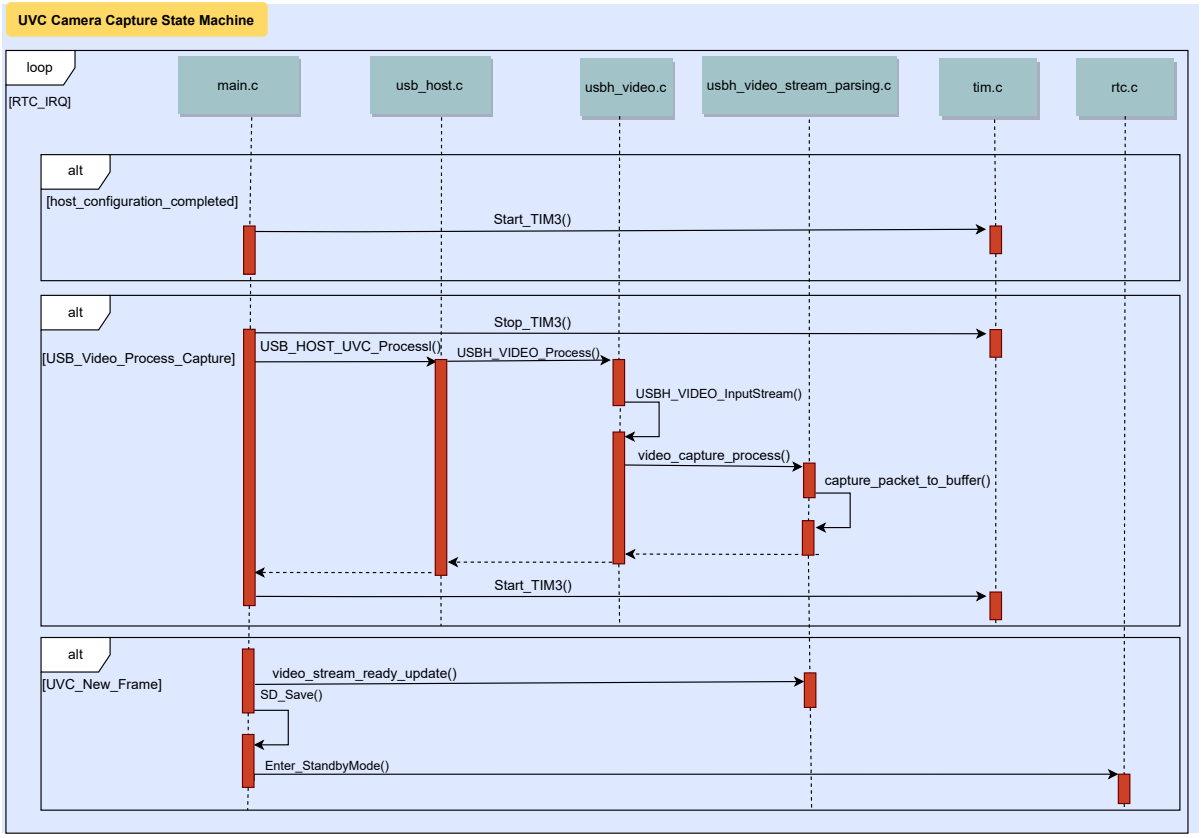


Figura 4.31: Diagrama de Sequências Ciclo Principal de execução da Aplicação

Capítulo 5: Resultados

Este capítulo de resultados foi dividido em dois subcapítulos, sendo que o primeiro subcapítulo foca nos resultados funcionais do sistema e o segundo subcapítulo apresenta os resultados não funcionais do sistema. Na primeira secção, é aborda-se a aquisição de imagens e a temporização do sistema. Na segunda, são abordados o consumo de energia e o dimensionamento da bateria. Para os testes funcionais, foi usada a comunicação através da porta série tendo como terminal o software RealTerm. Para os resultados não funcionais, foram feitas medições com um amperímetro e foi, igualmente, utilizado o software Program Consumption Calculator (PCC) para os dimensionamentos.

5.1 Resultados Funcionais

5.1.1 Temporização

A primeira fase dos testes funcionais foi feita para a validação da frequência de aquisição, a qual foi configurada as chamadas do RTC para tempos de aquisição de uma hora. Este RTC desperta a plataforma de processamento do modo de gestão de energia *standby* para efectuar a captura da imagem.

De modo a confirmar a operação do microcontrolador, recorreu-se à porta série para a depuração dos resultados da temporização e apresenta-se o instante da captura realizada, com a hora, minutos, segundos e subsegundos. Após a apresentação, confirma-se o término da aquisição e o sistema entra em modo de *sleep*. Então, estabeleceu-se um teste com uma duração de cinco horas, no qual é possível observar, pelo terminal apresentado na figura 5.1, o comportamento do sistema e o exacto instante da captura antes de entrar no modo de gestão de energia. Foram então efectuadas seis capturas, as quais foram armazenadas na memória externa, de acordo com o sistema de gestão de ficheiros pretendido. É possível observar pequenas mudanças nos subsegundos da aquisição sendo a diferença máxima de 0,0156 ms, notando que o *clock* do *RTC* é calibrado ao longo das várias aquisições. Os subsegundos apresentados, na tonalidade de cinza, são dados em milisegundos pela seguinte fórmula:

$$miliseconds = \frac{f_{RTCCLK}}{(PREDIV\_S + 1)} subseconds$$

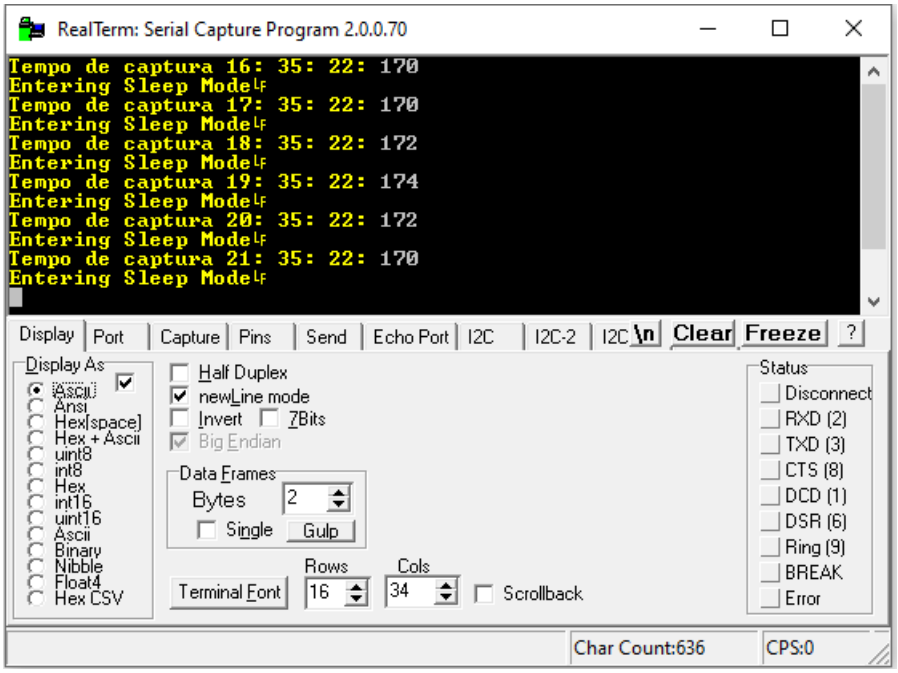


Figura 5.1: Instante de capturas e temporização

5.1.2 Aquisição de Imagens

A primeira fase de testes e de resultados para a aquisição de imagens consiste em adquirir os descriptores correctos da câmara de vídeo USB e a inicialização da classe de dispositivos de vídeo. A figura 5.2 apresenta a selecção do tamanho do endpoint, e os descriptores de frame e de formato encontrados na câmara de vídeo.

Confirmou-se através da análise dos descriptores obtidos na ligação da câmara de vídeo e de um PC e da comunicação da câmara de vídeo e do sistema de aquisição, e assim foram confirmadas todas as resoluções suportadas pelo dispositivo USB. Por fim é reconhecida a classe de dispositivos e inicializada, tornando o MCU como um host de dispositivos de vídeo UVC.

A aquisição de imagens foi feita para formatos do tipo descomprimido e com resoluções de 320 x 240 pixels, com o máximo de bitRate de 4.608 MB/s, e um máximo de tamanho de 153 600 bytes. Para o armazenamento da imagem foi usada um cartão microSD de 4 GB de memória. De modo a confirmar que a escrita no cartão de memória era feita correctamente, fez-se paralelamente o envio de dados pela porta série.

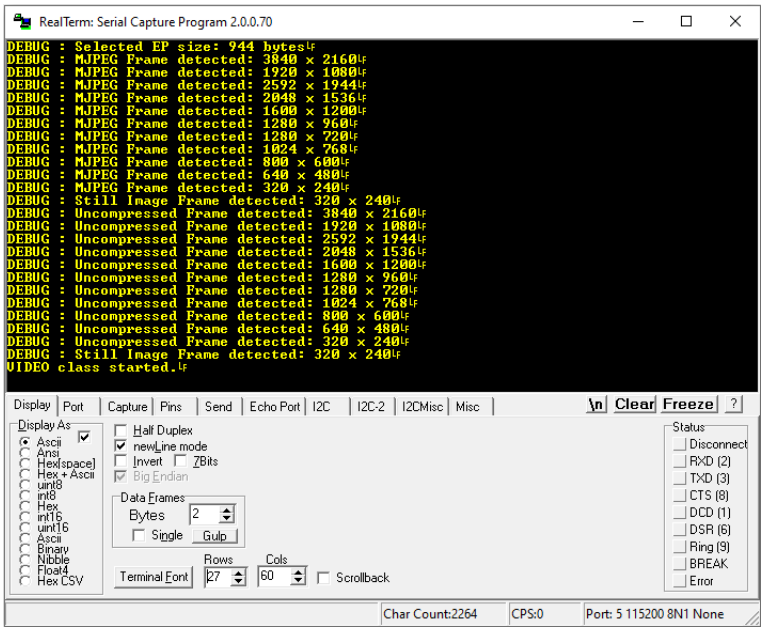


Figura 5.2: Conexão do dispositivo ao sistema

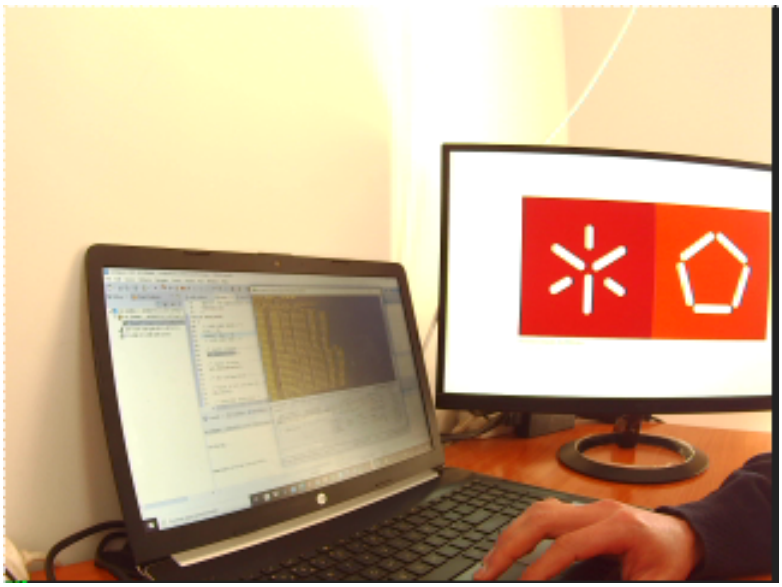


Figura 5.3: Imagem capturada

5.2 Resultados Não Funcionais

5.2.1 Consumo de Energia

O consumo de energia foi medida para os casos de operação da aquisição e durante o modo de standby. A tabela 5.1 apresenta os valores medidos e valores teóricos máximos das partes constituintes.

Módulos	Consumo Operação	Consumo Standby
STM32	26.3 mA	8.7 μA
Iluminação	340 mA	0 mA
Câmara + USB3300	220 mA	0 mA

Tabela 5.1: Consumo dos Módulos

Em modo de standby, o comando de iluminação e a câmara de vídeo, a interface do cartão SD, assim como o módulo PHY externo, são completamente desligados, não existindo consumo de corrente nesse modo. De notar de igual forma que o valor especificado pela câmara mais o módulo USB3300 trata do valor máximo retratado pelos descriptores. O valor prático para a aplicação revela-se bastante inferior a este máximo.

O consumo total para o sistema foi estimado entre valores medidos na plataforma de processamento para os modos de "Run", e em modo de gestão de energia, assim como para o módulo de iluminação composto por 20 LEDs. O valor do hardware de ligação da câmara de vídeo foi apontado como o máximo valor especificado para os cálculos. O consumo de sistema determinado com os módulos de iluminação, a câmara de vídeo, a interface de memória externa e todos os periféricos usados pelo sistema activos, em modo de "Run", foi de 370.1 mA. Para o modo de gestão de energia *standby*, apenas existe o consumo por parte da plataforma de processamento, no qual a gestão de consumo, a partir de comandos da GPIO e do circuito de gestão de energia é desligada todos os demais módulos externos.

Para efeitos da gestão do consumo de energia, foi medido o tempo de duração da execução da aplicação em modo *Run*, entre as configurações dos periféricos, a conexão e enumeração do dispositivo, a captura da imagem e o armazenamento na memória externa o qual teve um valor máximo de 2 segundos.

5.2.2 Dimensionamento da Bateria

Para o dimensionamento da bateria, foram usados os cálculos teóricos, os valores medidos e a plataforma de simulação do Program Consumption Calculator (PCC).Para tal determinou-se a frequência de aquisição de 1 hora Como mostra o gráfico da figura 5.4, houve um pico de corrente de 384.59 mA durante 2 s, e 8.9 μA durante 3598 s. A corrente média simulada foi de 225 μA. De notar que é necessário que a alimentação tenha os 5 V necessário à alimentação da interface USB.

Com por exemplo, com uma bateria de Lítio de 1000 mAh seria possível atingir até 187.68 horas de autonomia. Com uma *self discharge* típica de 8% ao primeiro mês, e de 2% nos meses subsequentes,

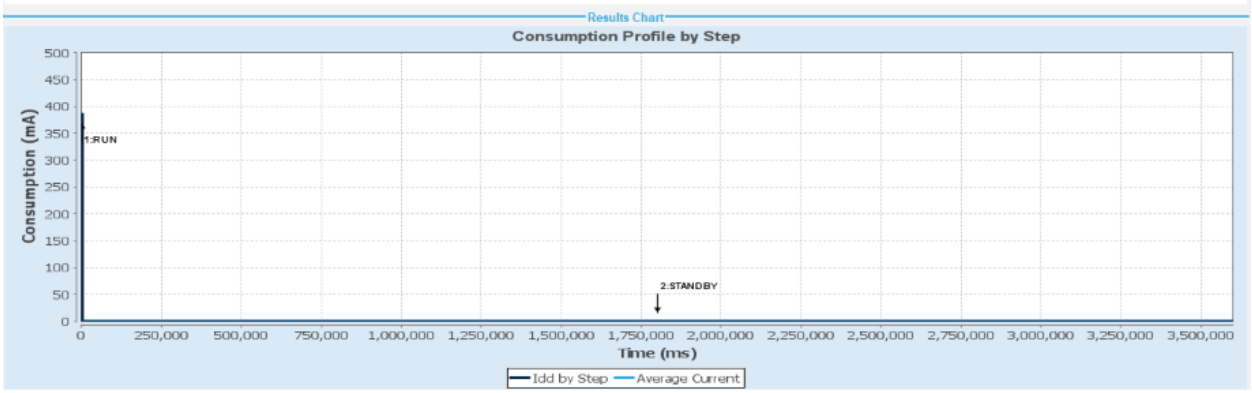
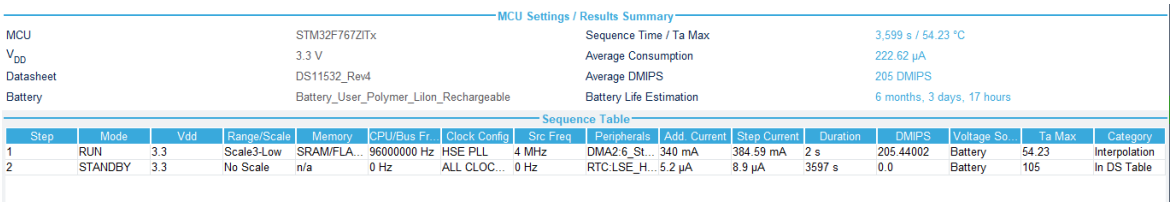


Figura 5.4: Gráfico do consumo do sistema durante um ciclo

isto equivaleria a pelo menos seis meses de autonomia, para uma aquisição de hora em hora. Para tal, o armazenamento em memória teria de ser de pelo menos 650 kB para resoluções de 320x240. Este valor de bateria é confirmado por cálculos feitos para a duração em horas da bateria pretendida. A figura 5.5 apresenta a estimação da vida útil da bateria de lítio recarregável que foi escolhida.



MCU Settings / Results Summary

MCU	STM32F767ZITx	Sequence Time / Ta Max	3.599 s / 54.23 °C
V_{DD}	3.3 V	Average Consumption	222.62 µA
Datasheet	DS11532_Rev4	Average DMIPS	205 DMIPS
Battery	Battery_User_Polymer_LiIon_Rechargeable	Battery Life Estimation	6 months, 3 days, 17 hours

Sequence Table

Step	Mode	Vdd	Range/Scale	Memory	CPU/Bus Fr...	Clock Config	Src Freq	Peripherals	Add. Current	Step Current	Duration	DMIPS	Voltage So...	Ta Max	Category
1	RUN	3.3	Scale3-Low	SRAM/FLA...	96000000 Hz	HSE PLL	4 MHz	DMA2:6_St...	340 mA	384.59 mA	2 s	205.44002	Battery	54.23	Interpolation
2	STANDBY	3.3	No Scale	n/a	0 Hz	ALL CLOC...	0 Hz	RTC:LSE_H...	5.2 µA	8.9 µA	3597 s	0.0	Battery	105	In DS Table

Figura 5.5: Gráfico da estimação da bateria

É possível obter maiores autonomias para períodos de capturas mais longos ou, de outro modo, que se aumente a capacidade das baterias a serem utilizadas pela aplicação.

Capítulo 6: Conclusão

De acordo com os objectivos propostos, foi desenvolvida a implementação de um dispositivo electrónico de monitorização para oceano profundo, baseado em imagens e capaz de recolher imagens a um tempo de amostragem configurável. De modo a ser posteriormente acondicionada em esferas de vidro resistentes que servem de *underwater-housing*, foi desenvolvido um sistema de reduzidas dimensões e de baixo consumo de energia. Para tal, fez um levantamento do estado da arte da importância da monitorização dos oceanos, das métricas pelas quais são classificadas e mensuradas os impactos antropogénicos e do caso específico do oceano profundo. De seguida foram averiguadas as técnicas tradicionais e comerciais de monitorização subaquática e posteriormente, foi feito um estudo da monitorização com recurso a imagem. Finalmente procedeu-se ao estudo das interfaces para câmaras de vídeo e foi determinada a interface de comunicação USB.

Para a interface USB, a comunicação com as câmaras de vídeo é feita por intermédio do protocolo UVC. Para tal, o desenvolvimento de um driver com o UVC foi implementado sobre a camada de comunicação USB disponibilizada em *open-source* pela plataforma de desenvolvimento utilizada, a STM32.

6.1 Conclusões

A possibilidade de uma monitorização sustentada do oceano profundo com soluções de baixo custo abre perspetivas em direção à sustentabilidade dos oceanos. A monitorização do oceano profundo é conseguida captando o maior número informações e dados que podem ser conseguidos através de imagens.

A existência da implementação da interface USB, com os drivers e controladores necessários à comunicação bidireccional, permitiu o desenvolvimento da classe de dispositivos de vídeo UVC para a plataforma de desenvolvimento da STM32F7.

A API da classe de vídeo foi implementada para essa família de microcontroladores sobre a interface USB. Foram realizados testes a esta classe provando os resultados a sua boa performance do software.

A uniformização desta classe de dispositivos torna possível a implementação de sistemas e aplicações de vídeo e/ou imagem com um reduzido tempo de desenvolvimento. As outras vantagens da utilização desta API é o relativo baixo custo de implementação e o seu baixo consumo de energia.

Fez-se uma análise estrutural e comportamental detalhada da classe de dispositivos áudio devido a

sua semelhança com os requisitos da classe de vídeo e adaptou-se a classe de vídeo com uma implementação estrutural e comportamental com base nessa análise. Posteriormente foram revistos os módulos necessários para a implementação da aplicação real. A camada da aplicação foi finalmente desenvolvida como um sistema de monitorização de águas profundas, integrando o protocolo UVC para a comunicação e os módulos e periféricos necessários à camada da aplicação. Os testes efectuados provaram ser possível a aquisição de imagens a frequências de amostragem fixas e com o armazenamento em memória externa dos dados adquiridos. Como resultados não funcionais, foram apresentados o baixo consumo de energia do sistema e o dimensionamento da bateria.

Para que se possa utilizar frames de maiores resoluções seria necessário a adição de memória externa devido ao tamanho reduzido da SRAM presente na placa F767. Esta limitação seria facilmente ultrapassada com o uso do periférico de FMC tornando possível adicionar mais memória mediante os requisitos impostos pelo caso específico de aquisição.

6.2 Trabalho Futuro

Como próximos passos para um trabalho futuro, aponta-se melhoria do sistema de forma a conseguir a aquisição de imagens com a resolução máxima da câmara de vídeo de 3840x2160, sendo para tal necessário expandir a memória RAM. O desenvolvimento de um circuito impresso do hardware completo tornaria o sistema mais robusto, compacto e eficiente. Pelo facto de se tratar de um sistema HS, é importante que se garanta a compatibilidade electromagnética, a qual será possível com a proximidade inerente oferecida pelas pistas de pequenas dimensões do circuito impresso. A redução dos custos por unidade e um sistema optimizado em termos de dimensão seriam outras vantagens.

Uma implementação da UVC com todas as funcionalidades apresentadas pela especificação da UVC, tal como a configuração dos parâmetros da câmara, permitiria ao sistema a adaptação dinâmica às condições do meio de captação.

Finalmente, ainda seria importante desenvolver um sistema de recuperação de falhas, para que havendo alguma anomalia, ela possa ser facilmente recuperada.

A validação com um maior número de câmaras de vídeo com a interface USB e protocolo UVC seria a progressão natural deste desenvolvimento o que levaria à implementação de outras funções desta classe de dispositivos USB.

Bibliografia

[1] E. Lindstrom, J. Gunn, A. Fischer, A. Mccurdy, L. K. Glover, K. Alverson, B. Berx, P. Burkill, F. Chavez, D. Checkley, C. Clark, V. Fabry, J. Hall, Y. Masumoto, D. Meldrum, M. Meredith, P. Monteiro, J. Mulbert, S. Pouliquen, C. Richter, S. Song, M. Tanner, R. Koopman, D. Cripe, M. Visbeck, and S. Wilson, "A framework for ocean observing prepared for the task team for an integrated framework for sustained ocean observing (ifsoo)," 2012.

[2] R. Danovaro, E. Fanelli, J. Aguzzi, D. Billett, L. Carugati, C. Corinaldesi, A. Dell'Anno, K. Gjerde, A. J. Jamieson, S. Kark, C. McClain, L. Levin, N. Levin, E. Ramirez-Llodra, H. Ruhl, C. R. Smith, P. V. Snelgrove, L. Thomsen, C. L. Van Dover, and M. Yasuhara, "Ecological variables for developing a global deep-ocean monitoring and conservation strategy," *Nature Ecology and Evolution*, vol. 4, no. 2, pp. 181–192, 2020.

[3] A. W. Bicknell, B. J. Godley, E. V. Sheehan, S. C. Votier, and M. J. Witt, "Camera technology for monitoring marine biodiversity and human impact," *Frontiers in Ecology and the Environment*, vol. 14, no. 8, pp. 424–432, 2016.

[4] B. T. Phillips, S. Licht, K. S. Haiat, J. Bonney, J. Allder, N. Chaloux, R. Shomberg, and T. J. Noyes, "Deepi: A miniaturized, robust, and economical camera and computer system for deep-sea exploration: A miniaturized deep-sea camera system," *Deep-Sea Research Part I: Oceanographic Research Papers*, vol. 153, p. 103136, 11 2019.

[5] P. Lertvilai, "The In situ Plankton Assemblage eXplorer (IPAX): An inexpensive underwater imaging system for zooplankton study," *Methods in Ecology and Evolution*, vol. 11, pp. 1042–1048, sep 2020.

[6] J. S. Jaffe, F. Simonet, and B. Laxton, "Omni-Cam: An autonomous free-descent omni-directional camera for measuring radiance in marine environments," *OCEANS 2011 IEEE - Spain*, pp. 11–14, 2011.

[7] A. Purser, U. Hoge, J. Lemburg, Y. Bodur, E. Schiller, J. Ludszuweit, J. Greinert, and F. Wenzhöfer, "PlasPI marine cameras: Open-source, affordable camera systems for time series marine studies," *HardwareX*, vol. 7, p. e00102, apr 2020.

[8] "Usb3300 datasheet."

[9] C. M. Pagniello, J. Butler, A. Rosen, A. Sherwood, P. L. Roberts, P. E. Parnell, J. S. Jaffe, and A. Širović, "An optical imaging system for capturing images in low-light aquatic habitats using only ambient light," *Oceanography*, vol. 34, pp. 71–77, sep 2021.

[10] G. Li, X. Chen, F. Zhou, Y. Liang, Y. Xiao, X. Cao, Z. Zhang, M. Zhang, B. Wu, S. Yin, Y. Xu, H. Fan, Z. Chen, W. Song, W. Yang, B. Pan, J. Hou, W. Zou, S. He, X. Yang, G. Mao, Z. Jia, H. Zhou, T. Li, S. Qu, Z. Xu, Z. Huang, Y. Luo, T. Xie, J. Gu, S. Zhu, and W. Yang, "Self-powered soft robot in the Mariana Trench," *Nature*, vol. 591, pp. 66–71, mar 2021.

[11] R. S. Santos, K. M. Gjerde, M. Yucel, D. Santillo, K. A. Miller, K. F. Thompson, and P. Johnston, "An Overview of Seabed Mining Including the Current State of Development, Environmental Impacts, and Knowledge Gaps," *Frontiers in Marine Science | www.frontiersin.org*, vol. 1, p. 418, 2018.

[12] M. Dussauze, K. Pichavant-Rafini, M. Belhomme, S. L. Floch, P. Lemaire, and M. Theron, "Deep-sea versus shallow conditions: a comparative ecobarotoxicological study," 2020.

[13] J. Aguzzi, D. Chatzievangelou, S. Marini, E. Fanelli, R. Danovaro, S. Flögel, N. Lebris, F. Juanes, F. C. D. Leo, J. D. Rio, L. Thomsen, C. Costa, G. Riccobene, C. Tamburini, D. Lefevre, C. Gojak, P. M. Poulain, P. Favali, A. Griffa, A. Purser, D. Cline, D. Edgington, J. Navarro, S. Stefanni, S. D'Hondt, I. G. Priede, R. Rountree, and J. B. Company, "New high-tech flexible networks for the monitoring of deep-sea ecosystems," *Environmental Science and Technology*, vol. 53, pp. 6616–6631, 6 2019.

[14] J. Aguzzi, N. Iveša, M. Gelli, C. Costa, A. Gavrilovic, N. Cukrov, M. Cukrov, N. Cukrov, D. Omanovic, M. Štifanić, S. Marini, M. Piria, E. Azzurro, E. Fanelli, and R. Danovaro, "Ecological video monitoring of Marine Protected Areas by underwater cabled surveillance cameras," *Marine Policy*, vol. 119, sep 2020.

[15] U. Nations, "Decade of ocean science."

[16] P. Miloslavich, N. J. Bax, S. E. Simmons, E. Klein, W. Appeltans, O. Aburto-Oropeza, M. Andersen Garcia, S. D. Batten, L. Benedetti-Cecchi, D. M. Checkley, S. Chiba, J. E. Duffy, D. C. Dunn, A. Fischer, J. Gunn, R. Kudela, F. Marsac, F. E. Muller-Karger, D. Obura, and Y. J. Shin, "Essential ocean variables for global sustained observations of biodiversity and ecosystem changes," *Global Change Biology*, vol. 24, no. 6, pp. 2416–2433, 2018.

[17] J. V. Gardner, A. A. Armstrong, B. R. Calder, and J. Beaudoin, "So, how deep is the mariana trench?," *Marine Geodesy*, vol. 37, pp. 1–13, 2014.

[18] A. Basset, N. Simboura, M. Obst, v. J. der Kooij, N. Eem, T. P. Bean, N. Greenwood, R. Beckett, L. Biermann, J. P. Bignell, J. L. Brant, G. H. Copp, M. J. Devlin, S. Dye, S. W. Feist, L. Fernand, D. Foden, K. Hyder, C. M. Jenkins, J. van der Kooij, S. Kröger, S. Kupschus, C. Leech, K. S. Leonard, C. P. Lynam, B. P. Lyons, T. Maes, E. E. Manuel Nicolaus, S. J. Malcolm, P. McIlwaine, N. D. Merchant, L. Paltriguera, D. J. Pearce, S. G. Pitois, P. D. Stebbing, B. Townhill, S. Ware, O. Williams, and D. Righton, "A Review of the Tools Used for Marine Monitoring in the UK: Combining Historic and Contemporary Methods with Modeling and Socioeconomics to Fulfill Legislative Needs and Scientific Ambitions," *Frontiers in Marine Science | www.frontiersin.org*, vol. 1, p. 263, 2017.

[19] J. S. Jaffe, "Underwater Optical Imaging: The Past, the Present, and the Prospects," *IEEE Journal of Oceanic Engineering*, vol. 40, pp. 683–700, jul 2015.

[20] J. Taucher, P. Stange, M. Algueró-Muñiz, L. T. Bach, A. Nauendorf, R. Kolzenburg, J. Büdenbender, and U. Riebesell, "In situ camera observations reveal major role of zooplankton in modulating marine snow formation during an upwelling-induced plankton bloom," *Progress in Oceanography*, vol. 164, pp. 75–88, may 2018.

[21] M. Broadhurst, S. Barr, and C. D. L. Orme, "In-situ ecological interactions with a deployed tidal energy device; an observational pilot study," *Ocean and Coastal Management*, vol. 99, pp. 31–38, oct 2014.

[22] K. Breddermann, P. Drescher, C. Polzin, H. Seitz, and M. Paschen, "Printed pressure housings for underwater applications," *Ocean Engineering*, vol. 113, pp. 57–63, 2 2016.

[23] by Nautilus Marine Service VITROVEX, "Simple • reliable • affordable proven deep sea housings · subject to change without notice · vitrovex® instrumentation housings protection for deep ocean explorations to 12,000 meters,"

[24] S. Arm, "Reference manual," no. March, 2018.

[25] M. Williamson, "Developments in machine vision camera interfaces title," 2020.

[26] " ELP-USB4KHDR01-MFV(5-50mm)."

[27] "Um2195 user manual usb device audio streaming expansion package for stm32cube," 2019.

[28] O. Tirosh, A. Lansing, C. Todd, D. Laboratories, R. Zimmermann, and L. G. Knapen, "Scope of this release revision history revision date filename author description," 1998.

Apêndice A: Appendix

Symbol	Parameter	Conditions	Typ(1)			Max(2)			Unit
			T_A = 25 °C			T_A = 25 °C	T_A = 85 °C	T_A = 105 °C	
			V_{DD} = 1.7 V	V_{DD}= 2.4 V	V_{DD} = 3.3 V	V_{DD} = 3.3 V			
I_{DD_STBY}	Supply current in Standby mode	Backup SRAM OFF, RTC and LSE OFF	1.1	1.9	2.4	5(3)	18(3)	38(3)	µA
		Backup SRAM ON, RTC and LSE OFF	1.9	2.7	3.2	6(3)	23(3)	48(3)	
		Backup SRAM OFF, RTC ON and LSE in low drive mode	1.7	2.7	3.5	7	26	55	
		Backup SRAM OFF, RTC ON and LSE in medium low drive mode	1.7	2.7	3.5	7	26	56	
		Backup SRAM OFF, RTC ON and LSE in medium high drive mode	1.8	2.8	3.6	8	28	57	
		Backup SRAM OFF, RTC ON and LSE in high drive mode	1.9	2.9	3.7	8	28	59	
		Backup SRAM ON, RTC ON and LSE in low drive mode	2.4	3.4	4.3	8	31	65	
		Backup SRAM ON, RTC ON and LSE in Medium low drive mode	2.4	3.5	4.3	8	31	65	
		Backup SRAM ON, RTC ON and LSE in Medium high drive mode	2.6	3.7	4.5	8	33	68	
		Backup SRAM ON, RTC ON and LSE in High drive mode	2.6	3.7	4.5	9	33	68	

Figura A.1: Tabela Valores Máximos de Consumo Standby Mode