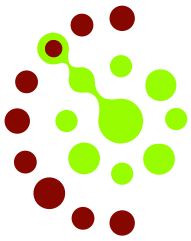


IPBeja

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Monitorização da Qualidade da Água

Raul Manuel Sousa Carvalho

Dezembro de 2021

INSTITUTO POLITÉCNICO DE BEJA

Escola Superior de Tecnologia e Gestão

Mestrado em Internet das Coisas

Monitorização da Qualidade da Água

Raul Manuel Sousa Carvalho

Orientado por :

Doutor José Jasnau Caeiro (orientador)

Doutor João Carlos da Silva Martins (coorientador)

Dissertação de Mestrado

2021

Agradecimentos

Em primeiro lugar um agradecimento aos meus orientadores, Doutor José Jasnau Caeiro (orientador) e Doutor João Carlos da Silva Martins (coorientador), pela partilha de conhecimento e experiência, e pela confiança que em mim depositaram.

Um agradecimento particular ao Doutor João Miguel da Silva Paiva Fatela dos Santos pela colaboração e disponibilidade demonstrada sempre que precisei.

Um agradecimento a todos os professores do Mestrado em Internet das Coisas, anos lectivos 2019/2020 e 2020/2021, pela motivação e pelo conhecimento que partilharam.

Uma agradecimento ao Doutor João Miguel da Silva Paiva Fatela dos Santos, Doutor João Carlos da Silva Martins, Doutor João Filipe Fragoso dos Santos, Doutora Patrícia Alexandra Dias Brito Palma e Doutor José Jasnau Caeiro, pela amabilidade e confiança patenteada no auspício da minha «estreia» na escrita e apresentação de artigos em conferências.

Um agradecimento especial ao Professor Doutor José Jasnau Caeiro pela disponibilidade, paciência, acompanhamento, sugestões de trabalho e principalmente por me ter enriquecido académica, cultural e pessoalmente.

Uma nota para os manos/as e sobrinhos/as, pela disponibilidade, mesmo em pequenas audições reforçaram sempre a minha confiança.

Dedico este trabalho aos meus filhos, Rita e Filipe e esposa Elsa, pelo carinho, apoio, motivação e tranquilidade que transmitiram durante este percurso. Nunca saberão o quão importantes e decisivos foram. Aos meus pais que não acompanharam este percurso, mas acredito que seria para eles um motivo de felicidade.

Todos tiveram quinhão neste percurso.

A todos, o meu Muito Obrigado!

Resumo

Monitorização da Qualidade da Água

O motor de desenvolvimento e crescimento económico, para o presente milénio, é definido pela Organização das Nações Unidas como sendo o acesso à água potável e o combate às alterações climáticas.

Neste contexto, a importância dos sistemas de informação sobre os recursos hídricos e a sua gestão não pode ser ignorada no processo de desenvolvimento socioeconómico das nações e na preservação do meio ambiente mundial.

É apresentado um sistema, baseado no paradigma arquitetónico da Internet das Coisas (IoT), concebido para a recolha e processamento de dados meteorológicos e de qualidade das águas superficiais. Este documento tenta fornecer uma visão geral sobre sistemas de monitorização da qualidade da água, com principal ênfase na arquitetura de *software*.

A arquitetura de *hardware* e *software*, de código aberto, é um aspeto importante da concepção deste sistema. Integra microcontroladores de baixo consumo energético e computadores de placa única para adquirir e processar os dados dos parâmetros medidos pelos sensores; comunicações de longo alcance utilizando o protocolo *LoRaWAN*, permitindo que o sistema seja implantado em áreas remotas e extensas; a aproximação de *software* baseada na tecnologia de *contentores* para uma realização fácil, escalável e controlável, reduzindo assim a complexidade da reprodução da configuração experimental em sistemas de investigação. É descrita a implementação de um sistema de gestão de base de dados geoespacial utilizando *software* livre e de código aberto.

As medições em «tempo real», de parâmetros como a temperatura da água e do ar, condutividade elétrica, pH, precipitação e pressão atmosférica, entre outros, poderão ser correlacionados com outras medições, fornecendo informação para prever, prevenir e agir na qualidade da água.

O sistema permite a visualização e análise dos dados em «tempo real», de vários corpos de água em diferentes períodos.

Palavras-chave: *Qualidade da água, Internet das coisas, Tecnologia de Contentores, Arquitetura de Software, Código Aberto.*

Abstract

Monitorização da Qualidade da Água

The engine of development and economic growth for the present millennium is defined by the United Nations as being access to clean drinking water, sanitation, combat climate change and its impacts.

In this context, the importance of information systems on water resources and their management cannot be ignored in the process of socio-economic development of nations and in the preservation of the global environment.

A system based on the Internet of Things (IoT) architectural paradigm, designed for the collection and processing of meteorological and surface water quality data, is presented. This paper attempts to provide an overview on water quality monitoring systems, with emphasis on the software architecture.

The open source hardware and software architecture is an important aspect of the design of this system. It integrates low power consumption microcontrollers and single board computers to acquire and process the data from the parameters measured by the sensors; long range communications using the LoRaWAN protocol, allowing the system to be deployed in large and remote areas; the software approach based on container technology for easy, scalable and controllable realization, thus reducing the complexity of reproducing the experimental setup in research systems. The implementation of a geospatial database management system using free and open source software is also described.

The "real time"measurements of parameters such as water and air temperature, electrical conductivity, pH, precipitation and atmospheric pressure, among others, can be correlated with other measurements, providing information to predict, prevent and act on water quality.

The system allows the visualization and analysis of "real-time"data from various water bodies in different periods.

Keywords: *Water Quality, Internet of Things, IoT, Container Technology, Software Architecture, Open Source.*

Índice

Índice de Figuras

Índice de Tabelas

Índice de Listagens

Abreviaturas e Siglas

3GPP	*3rd Generation Partnership Project*
6LoWPAN	*IPv6 over Low power Wireless Personal Area Network*
ABP	Ativação por Personalização
APA	Agência Portuguesa do Ambiente
API	Aplicação para Interface de Programação
ARM	*Advanced RISC Machine*
BLE	*Bluetooth Low Power*
BNC	*Conetor-Bayonet Neill–Concelman*
CoAP	Protocolo de Aplicação Restrita
CSS	*Cascading Style Sheets*
CSS	*ChirpSpread Spectrum*
DCCP	*Datagram Congestion Control Protocol*
DO	Oxigénio Dissolvido
EEPROM	Memória de Leitura Programável Apagável Eletricamente
EUI	*Extended Unique Identifier*
FPGA	*Field Programmable Gate Array*
FTP	Protocolo de Transferência de Ficheiros
gRPC	*Open Source Remote Procedure Call*
GSM	Sistema Global para Comunicações Móveis
HDL	Linguagem de Descrição de Hardware
HTML	Linguagem de Marcação de HiperTexto
HTTP	Protocolo de Transferência de Hipertexto
I2C	*Inter-Integrated Circuit*
IDE	Ambiente de Desenvolvimento Integrado
IoT	Internet das Coisas
IP	*Internet Protocol*
IPBeja	Instituto Politécnico de Beja
JSON	*JavaScript Object Notation*
LPWAN	Rede de Longa Distância e de Baixo Consumo
LTE	*Long Term Evolution technology*

M2M	*Machine to Machine*
MAC	*Media Access Control*
MCU	*Microcontroller Unit*
MIdC	Mestrado em Internet das Coisas
MIPS	Milhões de Instruções por Segundo
MIT	Instituto de Tecnologia de Massachusetts
ML	*Machine Learning*
Modbus	*Serial Communications Protocol*
MQTT	*Message Queuing Telemetry Transport*
MSN	Nós de Sensores Móveis
MySQL	*Structured Query Language*
NB-IOT	*Narrow-Band*
NonSQL	Não SQL ou Não Relacional
NTU	Unidade de Turbidez Nefelométrica
ORP	Potential de Redução de Oxidação
OSI	Organização Internacional de Normalização
OTAA	Ativação Através do Ar
pH	Potencial de Hidrogénio
PWM	Modulação por Largura de Pulso
QoS	Qualidade do Serviço
REST	*Representational State Transfer*
RFID	Identificação por Radiofrequência
RSSF	Redes Sensores Sem Fios
RTC	*Real Time Clock*
RTP	*Real-time Transport Protocol*
SBC	Computador de Placa Única
SCADA	Sistemas de Supervisão e Aquisição de Dados
SCTP	*Stream Control Transmission Protocol*
SDRAM	Memória de Acesso Aleatório Dinâmica Síncrona
SGBD	Sistema de Gestão de Base de Dados
SI	Sistema Internacional de Unidades
SMS	Serviço de Mensagens Curtas
SQL	*Structured Query Language*
SSH	*Secure Socket Shell*
SSL	*Secure Socket Layer*
SST	Sólidos Suspensos Totais
SWQMS	*Surface Water Quality Monitoring Systems*
TCP	*Transmission Control Protocol*
TDS	Sólidos Dissolvidos Totais

TLS	*Transport Layer Security*
UART	*Universal Asynchronous Receiver Transmitter*
UDP	*User Datagram Protocol*
UIT	União Internacional de Telecomunicações
UWB	*Ultra Wide Band*
VHDL	VHSIC *Hardware Description Language*
VMA	Valor máximo admissível
VMR	Valor máximo recomendado
WHO	Organização Mundial da Saúde
WiFi	*Wireless Fidelity*
WiMax	Interoperabilidade Mundial para Acesso de Micro-Ondas
WLAN	*Wireless Local Area Network*
WMAN	*Wireless Metropolitan Area Network*
WMS	*Water Monitoring Systems*
WPAN	*Wireless Personal Area Network*
WQMS	*Water Quality Monitoring Systems*
WSN	*Wireless Sensor Network*
ZigBee	Comunicações Sem Fios segundo o padrão IEEE 802.15.4

Capítulo 1

Introdução

O tema desta dissertação resulta na realização dum protótipo de sistema de *software* que exibe em contínuo os dados de qualidade de água duma plataforma. A realização desta dissertação de mestrado faz parte dos trabalhos do projeto de investigação AquaQ2, financiado pelo programa H2020, região do Alentejo. Neste capítulo é realizada uma introdução ao trabalho que se propõe desenvolver.

Em primeiro lugar é apresentada a motivação que levou à sua realização. É discutido o seu posicionamento na situação atual do mundo tecnológico, bem como a forma como pode influenciar hábitos e potencialmente mudar a forma, como o Mundo encara a escassez da água, fator fortemente ligado aos problemas ambientais em geral e ao aquecimento global em particular. Em segundo e terceiro lugar são apresentados objetivos que se pretendem atingir com este trabalho e as contribuições que o mesmo traz aos *Sistemas de Monitorização de Qualidade de Águas Superficiais* (SWQMS), respetivamente. De seguida é apresentada a metodologia adoptada e finalmente, é apresentada a estrutura deste documento [86].

A qualidade da água é um dos temas predominantes na sustentabilidade da vida humana e do seu desenvolvimento, enquadrando-se na luta premente de combate às alterações climáticas [128].

A água é um bem essencial à vida, para consumo humano e animal, para a produção de alimentos, na agricultura ou aquacultura, e até nas atividades de lazer. O desenvolvimento e rápido crescimento da população mundial, Figura 1.1, contribui para a deterioração do ambiente, com forte impacto na qualidade e quantidade de água disponível. Os sistemas de gestão da água são infraestruturas críticas, a sua avaliação, a constante monitorização e gestão, são tarefas cruciais.

Portugal tem escassos e pequenos lagos naturais, pelo que a maior parte da água doce disponível se encontra na precipitação, na que escoa superficialmente através dos rios, na que se infiltra (aquíferos) e na que está armazenada nas albufeiras [16]. A escassez de água tem sido um dos principais constrangimentos ao desenvolvimento da região do Alentejo, limitando a modernização da agricultura e a sustentabilidade do abastecimento público

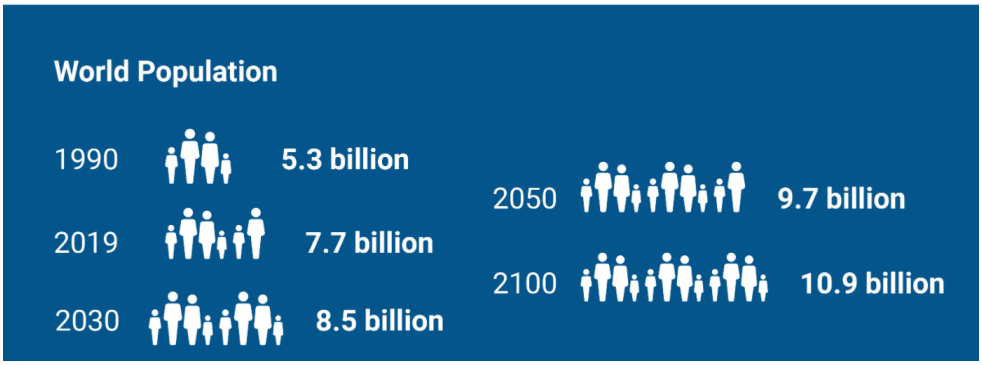


Figura 1.1: [Aumento da população mundial. As Nações Unidas prevêem, ao logo das próximas décadas, um grande aumento da população mundial. Em 2100 prevê-se que se atinja mais do dobro da população existente em 1990.

Fonte: *United Nations (2019) World population prospects* [139]

de água. Esta região é caraterizada por um grande défice de precipitação e uma elevada intensidade de irrigação agrícola. Os recursos hídricos tornam-se assim uma oportunidade de investigação, uma vez que os resultados podem conduzir a uma gestão mais sustentável dos recursos disponíveis[33].

Ao longo da maior parte da história humana, a avaliação da qualidade da água foi baseada na percepção sensorial e na observação dos efeitos que a água tinha sobre os organismos vivos. No final do século XX, o sistema tradicional de monitorização da qualidade da água consistia em recolher amostras de água de rios ou lagos, transportá-las para os laboratórios, geralmente distantes dos locais de recolha, realizando aí, a análise da qualidade da água. As principais desvantagens deste sistema prendiam-se com a impossibilidade de monitorizar em «tempo real» e com o facto dos resultados obtidos ficarem confinados a uma pequena área geográfica.

A *Internet das Coisas* (*IoT*) assenta no pressuposto de que qualquer coisa (objeto) pode ter uma identidade na Internet e pode ser monitorizada (analisada) e servir como base para se atuar sobre o sistema. Dentro das problemáticas atuais que podem beneficiar deste ecossistema de tecnologias, está a monitorização e análise da qualidade da água, o recurso do séc. XXI. Insere-se num tema mais abrangente que é a monitorização ambiental no combate às alterações climáticas, claramente inserido em 2 dos 17 *Objetivos de Desenvolvimento Sustentável*, objetivos adotados por todos os estados membros das Nações Unidas em 2015 [1]: 6ª meta, *«Achieve access to water and sanitation for all»* e 13ª meta, *«Take urgent action to combat climate change and its impacts»*.

Atualmente, a evolução da *IoT* tem permitido um aumento significativo nas medições

automáticas de características e parâmetros, físicos ou químicos, da água e da atmosfera, como a temperatura, condutividade, pH, precipitação, velocidade e direção do vento, etc., muitas vezes complementada por amostragens para análises laboratoriais *offline*.

Os avanços tecnológicos em sensores, microcontroladores (MCU), computadores de placa única (SBC) e tecnologias de comunicação, com uma queda significativa no consumo e custo de energia, tornaram possível satisfazer a crescente necessidade de monitorização da qualidade da água utilizando sistemas com arquiteturas baseadas em *IoT*, numa abordagem que tem sido objeto de estudo ao longo dos últimos anos [104],[7],[107]. Os sistemas baseados em *IoT*, em particular aqueles desenvolvidos com *hardware* e *software* de código aberto, têm custos mais baixos, tornando-os mais adequados à replicação e escalabilidade.

A monitorização automática *in-situ* permite um aumento significativo da quantidade de dados disponíveis em «tempo real», que podem ser combinados com dados de outras fontes, tais como detecção remota e análise laboratorial com base numa amostragem mais precisa. Reduz substancialmente os custos de mão-de-obra envolvidos nas amostragens, que naturalmente obrigam à deslocação aos locais em análise e a serviços de manutenção nesses locais. Por outro lado, a monitorização contínua de alguns dos parâmetros físicos e químicos da água permite substituir a amostragem tradicional por tempos pré-determinados, por uma amostragem desencadeada pela evolução dos parâmetros monitorizados.

Partindo de uma arquitetura de *hardware* que recolhe em continuo, um conjunto selecionado de parâmetros físicos e químicos da água, e da atmosfera, foi estabelecido como objetivo geral desta dissertação, o desenvolvimento de uma arquitetura de *software*, para a realização de um protótipo baseado em *IoT*, para monitorizar a qualidade da água superficial de rios e lagos em regiões distantes dos centros de monitorização, ou de difícil acesso [96].

As principais contribuições, a considerar em resultado da execução dos objetivos propostos são as seguintes:

- A arquitetura de *software* implementada. Adequada para a leitura, transmissão, processamento, armazenamento e visualização dos dados, permitindo a gestão e visualização em «tempo real». As comunicações baseiam-se na tecnologia *LoRa*, protocolo *LoRaWAN* e são suportadas pela pilha de *software ChirpStack* [1]. Todos os componentes da infraestrutura de *software* são implementados utilizando tecnologia de contentores, reduzindo a complexidade da gestão, implantação e portabilidade do *software*. A infraestrutura implementada facilita a replicação e adaptação da arquitetura de *hardware* e *software* a outros sensores e ao incremento de módulos locais de aquisição de dados;

[1] Servidores e serviços assentes na tecnologia de contentores.

- Armazenamento numa base de dados relacional os dados adquiridos sobre a qualidade da água e da atmosfera. É adicionada informação georreferenciada com a localização e caraterização das estações (sistema de informação geográfica (*GIS*);
- Aplicação que armazena os dados, transmitidos através da rede LoRaWAN;
- Aplicação que permite a visualização e análise a informação adquirida, em «tempo real», em qualquer dispositivo com acesso à Internet;
- Visualização adicional utilizando *Node-RED*.

Devido às características do projeto, de que esta dissertação é parte, foi importante definir uma metodologia coerente, com a finalidade de manter e desenvolver um estudo sustentado. O primeiro passo foi a realização de uma revisão/estado da arte, com a finalidade de estudar e aprofundar o conhecimento sobre o tema e sobre os sistemas que possam contribuir para o desenvolvimento do mesmo.

Esta dissertação tem como contributo os artigos, *«A Smart IoT System for Water Monitoring and Analysis»* apresentado no dia 15/09/2021, na conferência *EAI GOODTECHS 2021 - 7th EAI International Conference on Smart Objects and Technologies for Social Good* [106] e *«Hydrometereological Monitoring IoT System - The Software Architecture»* [10] apresentado no dia 5/11/2021, na conferência *4TH IFIP International Internet of Things (IOT) Conference*. O primeiro será publicado nos *proceedings of the 7th EAI International Conference on Smart Objects and Technologies for Social Good, GOODTECHS 2021*, o segundo será publicado em 2022, numa edição especial da revista Springer Nature Computer Science (SNCS).

O levantamento bibliográfico e apontamentos teóricos foram compilados com o intuito de os adaptar ao desenvolvimento de um protótipo II. Além disso, para complementar o estudo bibliográfico, foi realizada uma etapa de estudos empíricos que ancoram e comprovam, no plano da experiência ou da observação aquilo que se pretende conceptualmente. Ambos os estudos, bibliográfico e empírico, contribuíram para a elaboração do protótipo que responde aos objetivos idealizados.

Esta dissertação apresenta uma arquitetura para um sistema baseado em *IoT*, para monitorizar a qualidade da água superficial de rios e lagos.

Além da introdução, esta dissertação é composta por mais 4 capítulos. No capítulo 2 é realizado o enquadramento do tema monitorização da qualidade da água, nomeadamente a legislação aplicável, as propriedades físicas ou químicas e as principais características destes sistemas, o que contribui para a apresentação de uma solução e definição de alguns requisitos do sistema.

No capítulo 3 é apresentado um estudo sobre o estado da arte da monitorização da qualidade da água e são apresentados trabalhos semelhantes. Nomeadamente, no que concerne aos parâmetros de qualidade da água, sensores, microcontroladores, transmissão de dados, tecnologias e protocolos de comunicação, e armazenamento de dados. São apresentados os resultados sobre os trabalhos publicados sobre este tipo de sistema, nos últimos anos.

No capítulo 4 é apresentada e descrita a arquitetura do sistema, com ênfase na arquitetura de *software*. Inicia-se com a descrição de um diagrama geral, de monitorização de qualidade da água baseado em camadas. É apresentada uma arquitetura geral para um sistema de aquisição e análise de dados de qualidade da água, baseado em IoT. Partindo de uma arquitetura de *hardware* que adquire um conjunto de parâmetros físicos e químicos da água e da atmosfera é apresentada a arquitetura de *software* adequada para a leitura, transmissão, processamento e visualização dos dados.

No capítulo 5 é apresentada a realização do sistema que dá suporte à arquitetura de software. É apresentada: a implementação da *tecnologia de contentores Docker*; a configuração e utilização do conjunto de serviços *Chirpstack* para implementar redes *LoRaWAN*; a implementação de um *SGBD* com capacidade para gerir informação geográfica, para gerir e armazenar, os dados gerados pelos módulos de aquisição de dados; a utilização do ambiente de programação *low-code open-source Node-RED* e da linguagem de programação Python para o desenvolvimento das aplicações de armazenamento e visualização de dados; a utilização do *microframework Flask* para o desenvolvimento da aplicação de visualização de dados.

No capítulo 6 são apresentadas as conclusões gerais do trabalho realizado, nomeadamente a utilidade e aplicabilidade do estudo realizado e a sua relevância para a água como um bem social. Conclui-se com sugestões para trabalho futuro de modo a expandir e melhorar o presente projeto.

Capítulo 2

Monitorização da Qualidade da Água

O processo atual de monitorização de qualidade das águas superficiais, de uma forma geral, costuma obrigar à aquisição manual de amostras de água em vários pontos/locais, dos corpos de água que terão de ser transportadas para laboratórios, para realizar testes, com o objetivo de medir a qualidade da água. Este procedimento é ineficiente e pode tornar-se impreciso porque os parâmetros da qualidade da água podem sofrer alterações em curtos espaços de tempo. É, também dispendioso e demorado, não fornece os resultados em «tempo real» e não promove respostas pró-ativas à possível contaminação ou escassez da água.

Neste capítulo, no contexto da presente dissertação é estudada a informação usada para a implementação de um SWQMS (*Surface Water Quality Monitoring System*).

Na implementação de sistemas de monitorização da qualidade em águas superficiais, a seleção dos locais de monitorização, dos parâmetros de qualidade da água a monitorizar e das ferramentas utilizadas para analisar e manipular a informação gerada, assumem particular importância.

A sustentabilidade económica e financeira, desta tipologia de projeto obriga a uma criteriosa escolha das propriedades físicas ou químicas da água a monitorizar e da frequência das medições.

A Diretiva 2000/60/CE, no anexo II carateriza os tipos de águas de superfície [34] e o Decreto-Lei n.º 236/98 [24] estabelece parâmetros, normas, critérios e objetivos de qualidade da água com a finalidade de proteger o meio aquático e melhorar a qualidade das água em função do seu uso principal.

2.1 Legislação Aplicável à Qualidade da Água

Em Portugal, a Lei da Água (LA - Lei n.º 58/2005, de 29 de dezembro) e o Decreto-Lei nº 77/2006, de 30 de Março, transpuseram para a ordem jurídica nacional a Diretiva Quadro da Água (DQA - diretiva n.º 2000/60/CE, do Parlamento Europeu e do Conselho, de 23 de outubro), alterada e republicada pelo Decreto-Lei n.º 130/2012, de 22 de junho, onde

se estabelece que os Estados-Membros deverão proteger, melhorar e recuperar os corpos de água superficiais e subterrâneas com o objetivo ambiental de alcançar um **bom estado** das águas (Artigo 4º, DQA) através da aplicação dos programas de medidas especificados nos planos de gestão das regiões hidrográficas (PGRH), competência cometida à Agência Portuguesa do Ambiente [4]. Estes planos estabelecem como objetivos para as águas superficiais [4]:

- Evitar a deterioração do estado das massas de água;
- Proteger, melhorar e recuperar todos os corpos de água com o objetivo de alcançar o bom estado das águas – bom estado químico e ecológico;
- Proteger e melhorar todos os corpos de água fortemente modificados e artificiais com o objetivo de alcançar um bom potencial ecológico e o bom estado químico;
- Reduzir gradualmente a poluição e eliminar as emissões, as descargas e as perdas de substâncias perigosas.

Águas Subterrâneas [4]:

- Evitar ou limitar as descargas de poluentes nos corpos de água e evitar a deterioração do seu estado;
- Manter e alcançar o bom estado das águas - químico e quantitativo, garantindo o equilíbrio entre captações e recargas;
- Inverter qualquer tendência significativa persistente para aumentar a concentração de poluentes.

A qualidade da água descreve as características químicas, físicas e biológicas da água, em relação à qualidade natural, efeitos nos humanos e uso pretendido. A Diretiva 2000/60/CE, no anexo II, carateriza os tipos de águas de superfície [34] e o Decreto-Lei n.º 236/98 [24] estabelece parâmetros, normas, critérios e objetivos de qualidade da água com a finalidade de proteger o meio aquático e melhorar a qualidade das águas em função dos seus usos principais, conforme se pode observar no resumo apresentado na Tabela 2.1, onde são definidos os parâmetros (propriedades/unidades de medida), os valores máximos recomendados (VMR) e os valores máximos admitidos (VMA) que regulam a qualidade da água em Portugal.

2.2 Propriedades Físicas ou Químicas da Água

A água pura é uma substância química cujas moléculas são formadas por dois átomos de hidrogénio (H) ligados a um átomo de oxigénio (O), sendo H_2O a sua fórmula química.

		DL n.º 236/98								
		Rega		Objetivos ambientais	Produção para consumo humano					
		Anexo XVI [1]		Anexo XXI [2]	A1 do Anexo I [3]		A2 do Anexo I [3]		A2 do Anexo I [3]	
Parâmetros		VMR	VMA	VMA	VMR	VMA	VMR	VMA	VMR	VMA
pH	Escala de Sorensen	6,5-8,4	4,5-9,0	5,0-9,0	6,5-8,5		5,5-9,0		5,5-9,0	
Temperatura	°C	-	-	30	22	25	-	-	-	-
Condutividade	μS/cm	1000	-	-	1000	-	1000	-	1000	-
Oxigénio dissolvido	% de saturação	-	-	50*	70*	-	50	-	30	-
Sólidos suspensos totais	mg/L	60	-	-	25	-	-	-	-	-
Cloretos	mg/L	70	-	250	200	-	200	-	200	-
Nitratos	mg/L	50	-	-	25	50	-	50	-	50
Sulfatos	mg/L	575	-	250	150	250	150	250	150	250
Azoto amoniacal	mg/L	-	-	1	0,05	-	1	1,5	2	4

*refere-se a um valor mínimo recomendado e/ou admissível.
[1] – Qualidade das águas destinadas à rega
[2] – Objetivos ambientais de qualidade mínima das águas superficiais
[3] – Qualidade das águas doces superficiais destinadas à produção de água para consumo

Tabela 2.1: Parâmetros de aferição da qualidade da água referidos no Decreto-Lei n.º 236/98.

A água é singular, sendo a única substância encontrada nos três estados (sólido, líquido e gasoso) às temperaturas normalmente encontradas no nosso planeta. As principais propriedades das águas de superfície, segundo a Confederação Nacional da Agricultura (CNA) [16] são a temperatura, o pH, a condutividade elétrica, o oxigénio dissolvido, o potencial de redução de oxidação, a turbidez, os sólidos totais dissolvidos, o dióxido de carbono, o sulfato, a amónia, o nitrato, a nitrito e o carbono.

A redução de custos envolvidos na análise, oferecida pela utilização de sensores *in situ*, tem sido um fator motivador para melhorar a monitorização da qualidade das águas superficiais. A utilização destes sensores permite a recolha e medição das propriedades físicas ou químicas da água, de forma contínua ou intermitente, em «tempo real» ou quase real. A elevada densidade de medições em períodos relativamente curtos podem ser críticas porque a qualidade da água pode sofrer alterações muito rápidas. A correlação entre algumas propriedades físicas ou químicas da água, por exemplo, a temperatura, a condutividade e a turbidez, podem fornecer informação sobre outras propriedades mais difíceis e dispendiosas de analisar [137]. A Tabela 2.2 adaptada de [137] propõe uma lista de grandezas disponíveis e confiáveis, adequadas para a incorporação em sistemas automatizados de monitorização ambiental e da qualidade da água.

Considerando que quantos mais parâmetros de qualidade da água forem selecionados para monitorização, maior será o risco de diluir alguns fatores principais/dominantes da qualidade da água [23] e maior será o custo e a quantidade de trabalho para os sistemas a implementar.

Considerando que a seleção de parâmetros a monitorizar deverá ser baseada nos obje-

Medições automáticas	Eletrónicas	Óticas
Temperatura	x	
Condutividade / salinidade	x	
Clorofila		x
Ciano bacteriana		x
pH	x	
DO	x	x
Turbidez		x
Sólidos em suspensão		x
DOC		x
TOC		x
Nitrato	x	x
Nitrito	x	x
COD		x
Nível de água	x	
Hidrocarbonetos		x
Velocidade / direção da água	x	

Tabela 2.2: Parâmetros ambientais monitorizáveis automaticamente [137]. Propõe uma lista de variáveis adequadas à incorporação em sistemas automatizados de monitorização da qualidade da água.

tivos do projeto [126] e os parâmetros mais críticos que definem a qualidade da água são a temperatura, o oxigénio dissolvido e o potencial de hidrogénio (pH) [25].

Foram selecionados para monitorização as seguintes propriedades físicas ou químicas: a temperatura, o pH, a condutividade elétrica, o oxigénio dissolvido e o potencial de redução de oxidação.

Temperatura

Unidade de medida do SI: [°C]. É fortemente influenciada pela temperatura do ar, pela latitude, altitude, estação do ano, hora do dia, circulação do ar, nebulosidade e fluxo e profundidade do corpo de água. medição. Afeta a *vida biológica* na água, bem como o número de reações químicas, a *solubilidade*, a *condutividade* e a *toxicidade da água*. Por sua vez, a temperatura afeta os processos físicos, químicos e biológicos em corpos d'água e, portanto, a concentração de muitas variáveis. À medida que a temperatura da água aumenta, as *reações químicas* aumentam, tal como a *evaporação* e *volatilização* de substâncias da água. O aumento da temperatura também diminui a *solubilidade dos gases* na água: como o oxigénio, dióxido de carbono, nitrogénio, metano e outros.

A monitorização e controlo, da temperatura da água, nas diversas massas de água é essencial em estudos de auto purificação das mesmas para a vida aquática, para fins de refrigeração na indústria, para o consumo humano e rega agrícola [137].

pH

Unidade de medida do SI: [Escala de Sorensen]. O valor de pH é uma importante característica da água porque é capaz de afetar os organismos aquáticos, sendo ainda um indicador potencial do aumento da poluição ou de alteração de qualquer outro fator ambiental. O pH (potencial de hidrogénio) é uma medida de como ácida/básica a água.

O valor de pH varia entre 0 e 14, sendo 7 o valor de pH neutro (escala Sorensen). Valores de pH inferiores a 7 indicam acidez, enquanto um pH superior a 7 indica uma base. Uma vez que o pH pode ser afetado pela presença de produtos químicos na água, o pH é um indicador importante de água que está a mudar quimicamente. O pH é reportado em «unidades logarítmicas«. Cada unidade representa uma mudança de 10 vezes na acidez/basicidade da água. A água com um pH de cinco é dez vezes mais ácida que a água com um pH de seis [137].

Condutividade Elétrica

Unidade de medida do SI: [μS/cm]. A água pura (sem outras substâncias) é uma excelente isoladora, não conduzindo a eletricidade, deixa de ser um excelente isolante à medida que começa a dissolver as substâncias com as quais contata, já que mesmo uma pequena quantidade de *iões* numa solução de água torna-a capaz de conduzir eletricidade.

A água é reconhecida como o solvente universal. Quanto maior a quantidade de *iões* dissolvidos na água, maior é a sua condutividade. O facto de a água na natureza, nomeadamente nos rios e albufeiras, conduzir a corrente elétrica é, por exemplo muito vantajoso para as pessoas que necessitam de monitorizar as comunidades piscícolas. Caso contrário, a amostragem da fauna piscícola que é muitas vezes realizada através de um método que utiliza eletricidade para capturar os peixes (pesca elétrica), seria totalmente ineficaz [16]. A condutividade além de ser um indicador aproximado do *conteúdo mineral* quando outros métodos não podem ser facilmente utilizados, pode ser medida para determinar a *poluição* existente numa determinada zona, por exemplo em torno de uma descarga de um efluente, ou a extensão da influência das águas de escoamento.

Por exemplo, a condutividade elétrica permite determinar a *salinidade* (dS/m) ou o valor dos *Sólidos Dissolvidos Totais* (SDT - mg/l). Estas medições contínuas são particularmente úteis em rios e águas subterrâneas para a gestão das variações temporais nos *sólidos totais dissolvidos* e *íons* principais [137].

Os valores de condutividade elétrica da água são utilizados como indicativos da qualidade da água, com sua representação pelo Sistema Internacional em unidades miliSiemens por cm (mS/cm) ou micro Siemens por cm (μS/cm). A água da superfície deve em geral deve apresentar uma condutividade entre 500 e 1000 μS/cm. A água do mar tem cerca de 55 mS/cm.

Oxigénio Dissolvido

Unidade de medida do SI: [% de saturação]. O oxigénio dissolvido corresponde à concentração de oxigénio na água, essencial para todas as formas de vida aquática. Assim, este parâmetro é normalmente medido para avaliar a «saúde» das massas de água (rios, lagos, albufeiras). O oxigénio dissolvido nas massas de água (rios, lagos e albufeiras) é crucial para os organismos e as criaturas que nelas vivem.

Quando a quantidade de oxigénio dissolvido desce abaixo dos níveis normais, a qualidade da água é prejudicada e os seres vivos presentes, nomeadamente os peixes, podem morrer [137].

Potencial de Redução de Oxidação

Unidade de medida do SI: [mV]. O potencial de redução de oxidação (ORP) mede a capacidade de uma massa de água se purificar ou eliminar resíduos. Quando o ORP é alto há muito oxigénio presente na água. Em geral, quanto maior o valor de ORP, mais saudável é a massa de água. O POR diminui com a profundidade.

O ORP é tratado de forma separada do oxigénio dissolvido porque ORP pode fornecer aos investigadores informações adicionais sobre a qualidade da água e o grau de poluição [94].

No Apêndice I são apresentadas as características de um conjunto mais vasto de parâmetros físico ou químicos da qualidade da água.

2.3 Sistemas de Monitorização da Qualidade de Águas Superficiais

Os *SWQMS*[1], envolvem a medição de vários parâmetros, de qualidade da água, em corpos de águas superficiais, com o objetivo de garantir a avaliação e monitorização dessa qualidade em «tempo real» permitindo a deteção de incidentes de contaminação e ameaças à qualidade da água.
No projeto de um sistema desta natureza é importante conseguir estimar a sua sustentabilidade, económica ou financeira, isto é, o grau em que os benefícios resultantes das informações geradas, justificam, o custo e o nível de esforço, necessário para sua implementação e colocação em operação.

Os benefícios são amplamente determinados pelos objetivos que suportam o SWQMS, fazendo com que seja muito importante, a identificação e caraterização de ameaças à qualidade da água, a seleção de locais de monitorização, a selecção dos parâmetros de qualidade

[1] *Surface Water Quality Monitoring System).*

da água a monitorizar e as ferramentas utilizadas, para analisar e manipular a informação gerada.

Um local de monitorização no corpo de água onde é realizada a colheita de água, para análise ou medição. Estes locais devem ser selecionados, considerando a possível necessidade de ter que iniciar um processo de tratamento (por exemplo a adição de produtos químicos de pré-tratamento) ou uma ação de resposta (por exemplo, fechando ou abrindo uma comporta). Os sistemas dedicados à deteção de incidentes de contaminação, de uma forma geral requerem a adição, de vários locais de monitorização e a seleção de parâmetros, capazes de detetar, uma gama mais ampla de mudanças na qualidade da água, em conjunto com métodos de análise de dados, mais sofisticados. Para monitorizar ameaças à qualidade da água a longo prazo. Poderá ser necessário, adicionar ainda mais locais.

A seleção de parâmetros a monitorizar, deverá ser baseada nos objetivos do projeto, e nos resultados de avaliação dos riscos que ameaçam o corpo de água. As variações sazonais nas condições do corpo de água selecionado, como a temperatura, a precipitação e o fluxo [126]., são exemplo.

As ferramentas ou sistemas que recebem, processam, analisam, armazenam e apresentam os dados gerados pela estação de monitorização, podem incluir ferramentas de análise de dados que geram alertas e notificações quando são detetadas anomalias na qualidade da água.

Os CIO's [2] de 50 grandes empresas de serviços públicos nos Estados Unidos, estimaram que apenas 10 a 15 por cento das informações recolhidas e armazenadas pelas suas organizações são avaliadas adequadamente [126].

A análise, automatizada e tratamento correto dos dados, pode ajudar a resolver a utilização deficiente dos dados recolhidos. Antes da utilização dos dados recolhidos na estação de monitorização, estes devem ser validados, removendo ou corrigindo os erros óbvios, estabelecendo e compreendendo a variabilidade normal de cada parâmetro.

A monitorização manual da qualidade da água em locais remotos é muito dispendiosa e pode apresentar dificuldades à sua implementação, o que torna um desafio, a monitorização contínua e em «tempo real». É importante considerar os seguintes aspetos:

- auto-suficiência energética;
- comunicação de dados em «tempo real»;
- condições difíceis de acesso aos equipamentos;

[2] Chief Information Officer. Responsável pelas tecnologias da informação (TI) de uma empresa.

- operações de manutenção;
- degradação dos equipamentos.

Considerando as necessidades energéticas, para a recolha e transmissão de dados, confiar numa bateria sem capacidade de recarga, pode levar, a elevadas necessidades de manutenção. Os dispositivos, devem ser auto-suficientes em energia.

As operações de manutenção dos equipamentos de aquisição de dados são difíceis porque os dispositivos estão geralmente expostos a condições climáticas difíceis e continuamente num ambiente aquático, o que obriga a que a manutenção seja geralmente realizada a partir dum barco. É necessário adequar os dispositivos, a estas difíceis condições. Os corpos de água são frequentemente utilizados para fins recreativos, os dispositivos e equipamentos poderão estar em contato direto com as populações, o que pode levar a interferências e à sua degradação. As necessidades de manutenção devem ser minimizadas. Para reduzir a frequência das operações de manutenção, os diferentes sensores/sondas devem estar operacionais, sem qualquer limpeza manual, durante vários meses. É um desafio maior para os sensores ópticos, por exemplo os de medição de UV e turbidez.

Os locais escolhidos para monitorização da qualidade da águas encontram-se tipicamente em áreas remotas, onde as redes de comunicação não existem, situação que coloca entraves às comunicações de dados em «tempo real».

2.4 Conclusão

A recolha manual de amostras de água para aferir a sua qualidade, com recurso a laboratórios, não fornece resultados em tempo útil. A qualidade e a quantidade da água pode variar muito rapidamente, e os testes laboratoriais, necessários para caraterizar a qualidade da água, consomem muito tempo e são dispendiosas.

A monitorização da qualidade da água baseada em sensores *in situ*, oferece vantagens: a operação automática, a fácil instalação e o baixo custo, e nomeadamente, as resultantes da monitorização contínua e em «tempo real» que permitem detetar as variações da qualidade da água em períodos muito curtos. Contudo, estes sistemas exigem a disponibilidade de recursos de processamento computacional, memória, alcance das comunicações e energia necessária ao seu funcionamento. Se a arquitetura do sistema e as escolhas «chave» (locais de medição, parâmetros, comunicações, energia, ferramentas de tratamento de dados, etc), não forem tratadas adequadamente, estas exigências podem contribuir para resultados menos satisfatórios.

A seleção das propriedades físicas ou químicas da água, a monitorizar e a frequência das medições, deve garantir a sustentabilidade económica e financeira dos projetos e o respeito pelas leis e normas que definem os intervalos admissíveis, para os vários parâmetros de qualidade da água.

A recolha manual de amostras, para aferir a qualidade da água com recurso a laboratórios poderá continuar a existir, mas com muito menor frequência. Poderá ser utilizada para aferir a qualidade das medições realizadas, pelos sensores *in situ*.

Capítulo 3

Sistemas de Monitorização da Qualidade da Água

Neste capítulo são abordados trabalhos, tecnologias e soluções que podem ser utilizados como fonte de inspiração para a solução a propor. Antes de mais, é necessário entender o modo de funcionamento e evidenciar o conhecimento necessário para atuar em sistemas de monitorização da qualidade da água. É conveniente conhecer os mais importantes parâmetros físico ou químicos que influenciam a qualidade da água e a forma de realizar a sua avaliação. As tecnologias de transmissão, armazenamento, tratamento e visualização dos dados, assumem um papel essencial, na utilização dos dados adquiridos.

A monitorização da qualidade da água obriga à aquisição de dados num conjunto de diferentes locais ao longo da superfície da água, em diferentes profundidades e em intervalos regulares de tempo [12, 31].

A *IoT* está a desempenhar um papel importante na monitorização, em tempo real, da qualidade e quantidade da água. Um sistema *IoT* compreende tanto a monitorização de uma única *coisa* ou a interconexão de milhões de *coisas*, com a capacidade de fornecer serviços e aplicações complexas [70]. Vários trabalhos de investigação têm sido realizados recentemente para desenvolver sistemas inteligentes de identificação e monitorização de parâmetros de qualidade da água. Da análise de algumas publicações sobre o tema [23, 80, 107] destacam-se três subsistemas principais: o subsistema de aquisição de dados, o subsistema de transmissão de dados e o subsistema de gestão de dados [49].

Alguns dos desafios operacionais incluem o consumo de energia, a segurança e a interoperabilidade. Em geral, os três subsistemas estão interligados da seguinte forma: os sensores são responsáveis pela aquisição de informação do ambiente em que estão instalados; a recolha de dados dos sensores é realizada através de *MCUs* e/ou *SBCs*, ligados através de diferentes protocolos como *UART*, I^2C ou *SPI*. Uma vez recolhidos, os dados são transmitidos utilizando tecnologias de comunicação, por exemplo *WiFi*, *Bluetooth*, *ZigBee*, *LoRa*, *NBIoT*, *3G/4G/5G*, etc.

Os dados recolhidos são processados, limpos, modelados, analisados e armazenados num *SGBD*. Estes são combinados para fornecer informação ao utilizador final, tornando esta componente um valor que é potenciado pela *IoT*; o objetivo é fornecer, informação da forma mais simples e transparente, através de múltiplas plataformas: *tablets*, *smartphones* e *desktops*.

São analisadas soluções apresentadas por alguns autores que apresentaram este tipo de sistemas para conhecer as opções mais comuns e a sua evolução dentro destes três subsistemas. Os critérios de seleção e pesquisa de literatura são descritos no Apêndice II. O objetivo é apresentar conhecimento e ideias de como um sistema deste tipo pode ser implementado. Esta análise contribui para a proposta discutida na dissertação.

3.1 Parâmetros de Qualidade da Água

Não existem métodos ou padrões internacionais para a seleção de parâmetros da qualidade da água, a incluir obrigatoriamente nos sistemas de monitorização da qualidade da água. Quantos mais parâmetros de qualidade da água forem selecionados para monitorização, maior será o custo e a carga de trabalho para os implementar e corre-se o risco de diluir alguns factores principais/dominantes da qualidade da água [23].

Na Tabela 3.1, os sensores utilizados para a monitorização da qualidade água, referenciados com maior frequência (85%) no conjunto das 31 propostas analisadas no âmbito desta dissertação, são: temperatura, pH, turbidez, condutividade, oxigénio dissolvido, fluxo e nível da água. Conforme publicado por Dong *et al.* em 2015 [23] e pela Comissão Europeia na «*Review of sensors to monitor water quality*» [120] os 10 principais parâmetros monitorizados *online* desde a água bruta até à água utilizada para consumo humano, com base na percentagem de monitorização para cada parâmetro, em diferentes países do Mundo, Estados Unidos da América, Bélgica, Países Baixos, Reino Unido e Austrália, é fortemente coincidente com as incidências evidenciadas na Tabela 3.1.

3.2 Sensores

O estudo apresentado por Jiping Jiang *et al.* em 2020 [50], divide o histórico de estudos sobre WQMS[1] em três fases semelhantes ao crescimento de uma planta: I. Germinação (antes de 2005), II. Mudas (2005–2013) e III. Crescimento (após 2013). E constata que a fase de crescimento coincide com a observação de sensores *in situ*, que se tornou mais popular com a disseminação de sistemas *IoT*.

[1] Sistemas de monitorização de qualidade da água.

Parâmetros monitorizados	Nº Pub	%	Ident.	%
T (°C)	29	94%	23	79%
pH (Escala de Sorensen)	28	90%	19	68%
Tu (NTU)	15	48%	9	60%
EC (μS/cm)	12	39%	5	42%
DO (% de sat. O_2 (mg/l))	11	35%	7	64%
WL (m)	8	26%	6	75%
WFl (l/min)	5	16%	4	80%
CO_2 (mg/l)	3	10%	3	100%
NH_3 (mg/l)	3	10%	2	67%
NO_3^- (mg/l)	3	10%	2	67%
ORP (mV)	3	10%	1	33%
TDS (mg/l)	2	6%	1	50%
NO_2^- (mg/l)	2	6%	2	100%
Wp (bar)	1	3%	1	100%
CO (mg/l)	1	3%	1	100%
SO_4^{2-} (mg/l)	1	3%	1	100%

Tabela 3.1: Número de parâmetros monitorizados (Nº Pub) no conjunto das 31 publicações analisadas, evidenciando as publicações que identificam (Ident.) os sensores utilizados. A tabela apresenta os parâmetros físicos e químicos que é comum serem medidos em sistemas inteligentes de qualidade de água, utilizando uma abordagem IoT: temperatura (T), potencial de hidrogénio (pH), turbidez (TU), condutividade elétrica (EC), oxigénio dissolvido (DO), nível de água (WL), fluxo de água (WFl), dióxido de carbono (CO_2), amónia (NH_3), nitratos (NO_3^-), potencial de redução de oxidação (ORP), total de sólidos dissolvidos (TDS), nitritos (NO_2^-), pressão da água (WP), monóxido de carbono (CO) e sulfatos (SO_4^{2-}).

A análise da evolução das soluções no que concerne à utilização de sensores é baseada na informação sintetizada na Figura 3.1. Observar-se uma síntese dos sensores selecionados nos 31 sistemas estudados, na monitorização de cada um dos 16 parâmetros da qualidade da água. Na Tabela 3.1, dos 127 sensores utilizados, não foi possível identificar 40 (31%). Dos 87 (69%) sensores identificados 79% dizem respeito a apenas 5 grandezas (temperatura, pH, turbidez, condutividade, oxigénio dissolvido e nível da água) dos 16 parâmetros analisados.

O sensor de *temperatura* da água apresentado na grande maioria dos sistemas selecionados (34%) é o *DS18B20*. O *SEN0161* é o sensor de *pH* apresentado na grande maioria dos sistemas selecionados (25%).

Na medição da *turbidez* destacam-se os sensores *TSD10* e *SEN0189*, os dois representam 57% das escolhas, os dois sensores apresentam características muito semelhantes.

Na medição do fluxo, o sensor *YF-S201* foi o eleito em 40% das escolhas.

Na análise dos restantes parâmetros, condutividade, oxigénio dissolvido, nível, amónia, nitrato, nitrito, potencial de redução de oxidação e total de sólidos dissolvidos, não se verificou a repetição da escolha de qualquer sensor. O *SEN0219* foi o sensor escolhido por

unanimidade para monitorizar o dióxido de carbono.

No Apêndice III, em complemento à Figura 3.1 são apresentadas de forma mais pormenorizada, as características dos sensores dos parâmetros mais comuns utilizados na medição da qualidade da água.

3.3 Microcontroladores

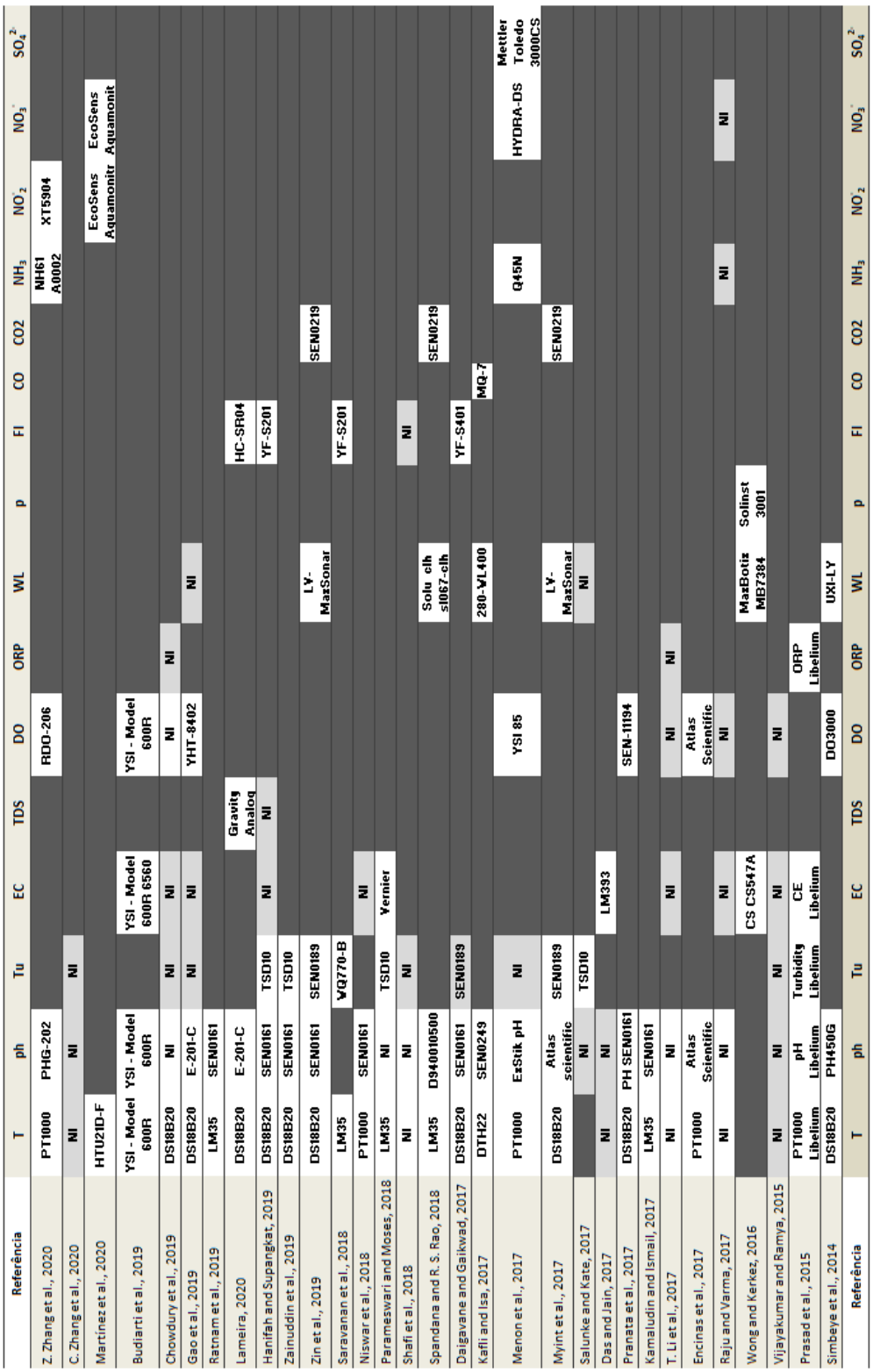
A aquisição de dados dos sensores é realizada através de *MCUs* e/ou *SBCs*, conetados por diferentes protocolos como o *UART* (XII), I^2C (XII) ou *SPI* (XII).

Os *MCUs* utilizados nos sistemas de monitorização da qualidade da água controlam os sensores e a aquisição de dados. Por vezes, também pré-processam os dados e enviam para o subsistema de transmissão. No conjunto dos sistemas analisados verifica-se que os microcontroladores (*MCUs*) mais sugeridos são da família *Atmega*, série *Atmega328*, usado em sistemas Arduino, por exemplo, Tabela 3.2. Os sistemas mais recentes incluem *MCUs ARM Cortex-M de 32 bits*, *MCUs* de baixo custo e baixo consumo de energia da linha *SMT32* e da linha *ESP 8266/32*, com recursos para comunicações sem fios.

MCU/SBC	Nº	Referências
ATmega8A/16L	1	Simbeye *et al.*, 2014
ATmega328	8	Hanifah and Supangkat, 2019; Parameswari and Moses, 2018; Saravanan et al., 2018; Daigavane and Gaikwad, 2017; Encinas *et al.*, 2017; Kamaludin and Ismail, 2017; Pranata *et al.*, 2017; Zainuddin *et al.*, 2019
ATmega2560	3	Chowdury *et al.*, 2019; Niswar *et al.*, 2018; Shafi *et al.*, 2018
ATmega1281	3	Menon *et al.*, 2017; T. Li *et al.*, 2017; Prasad *et al.*, 2015
ESP	2	Lameira, 2020; Spandana and R. S. Rao, 2018
ARM - SMT32	2	Z. Zhang *et al.*, 2020; C. Zhang *et al.*, 2020
ARM - LPC	2	Kafli and Isa, 2017; Das and Jain, 2017
FPGA	3	Zin *et al.*, 2019; Myint *et al.*, 2017; Wong and Kerkez, 2016
GalileoGen2	1	Salunke and Kate, 2017
Raspberry	7	Martínez *et al.*, 2020; Budiarti *et al.*, 2019; Ratnam *et al.*, 2019; Niswar *et al.*, 2018; Raju and Varma, 2017; T. Li *et al.*, 2017; Vijayakumar and Ramya, 2015

Tabela 3.2: Microcontroladores e computadores de placa única propostos nas 31 publicações analisadas. O microcontrolador ATmega328 e o SBC Raspberry, cada um na sua categoria, são os mais escolhidos.

A programação dos *MCUs* da família *ATmega* é compatível com um conjunto vasto de ferramentas de desenvolvimento de programas e sistemas. São normalmente programados em linguagem *C* e *C++*. As linguagens de programação *Assembly* e *JavaScript* também são uma possibilidade, contudo com menos recursos (bibliotecas) e menos documentadas. Existem várias ferramentas de desenvolvimento de *software* para a família *ARM*, disponibi-



Referência	T	ph	Tu	EC	TDS	DO	ORP	WL	p	FI	CO	CO2	NH_3	NO_2	NO_3^-	SO_4^{2-}
Z. Zhang et al., 2020	PT1000	PHG-202				RDO-206							NH61 A0002	XT5904		
C. Zhang et al., 2020	NI	NI	NI													
Martínez et al., 2020	HTU21D-F													EcoSens Aquamonitr	EcoSens Aquamonit	
Budiarti et al., 2019	YSI - Model 600R	YSI - Model 600R		YSI - Model 600R 6560		YSI - Model 600R										
Chowdury et al., 2019	DS18B20	NI	NI	NI		NI	NI									
Gao et al., 2019	DS18B20	E-201-C	NI	NI		YHT-8402		NI								
Ratnam et al., 2019	LM35	SEN0161			Gravity Analog					HC-SR04						
Lameira, 2020	DS18B20	E-201-C			NI					YF-S201						
Hanifah and Supangkat, 2019	DS18B20	SEN0161	TSD10	NI				LY-MaxSonar				SEN0219				
Zainuddin et al., 2019	DS18B20	SEN0161	TSD10													
Zin et al., 2019	DS18B20	SEN0161	SEN0189							YF-S201			Q45N		HYDRA-DS	Mettler Toledo 3000CS
Saravanan et al., 2018	LM35		VQ770-B					Solu cell sf067-clh				SEN0219				
Niswar et al., 2018	PT1000	SEN0161		NI						YF-S401	MQ-7					
Parameswari and Moses, 2018	LM35	NI	TSD10	Vernier				280-WL400								
Shafi et al., 2018	NI	NI	NI							NI						
Spandana and R. S. Rao, 2018	LM35	D940010500										SEN0219				
Daigavane and Gaikwad, 2017	DS18B20	SEN0161	SEN0189							YF-S401						
Kaffi and Isa, 2017	DTH22	SEN0249									MQ-7					
Menon et al., 2017	PT1000	EzStik pH	NI			YSI 85							Q45N		HYDRA-DS	Mettler Toledo 3000CS
Myint et al., 2017	DS18B20	Atlas scientific	SEN0189					LV-MaxSonar				SEN0219				
Salunke and Kate, 2017		NI	TSD10					NI								
Das and Jain, 2017	NI	NI		LM393												
Pranata et al., 2017	DS18B20	PH SEN0161				SEN-11194										
Kamaludin and Ismail, 2017	LM35	SEN0161														
T. Li et al., 2017	NI	NI		NI		NI	NI									
Encinas et al., 2017	PT1000	Atlas Scientific				Atlas Scientific										
Raju and Varma, 2017	NI	NI		NI		NI							NI		NI	
Wong and Kerkez, 2016				CS CS547A				MaxBotix MB7384	Solinst 3001							
Vijayakumar and Ramya, 2015	NI	NI	NI	NI		NI										
Prasad et al., 2015	PT1000 Libelium	pH Libelium	Turbidity Libelium	CE Libelium			ORP Libelium									
Simbeye et al., 2014	DS18B20	PH450G				DO3000		UXI-LY								
Referência	T	ph	Tu	EC	TDS	DO	ORP	WL	p	FI	CO	CO2	NH_3	NO_2	NO_3^-	SO_4^{2-}

Figura 3.1: Sensores utilizados em cada uma das 31 publicações (NI - fabricantes dos sensores não identificados).

Dispositivo	Linha	Modelos	Sist. IoT	Freq.
MCU	ATmega	ATmega8/16	1	43%
		ATmega328	8	
		ATmega2560	2	
		ATmega1281	2	
	ARM32	ESP8266	1	17%
		ESP32	1	
		STM32	1	
		LPC1768	1	
		LPC2148	1	
SBC	Galileo	GalileoGen2	1	20%
	RPi	RPi B+	3	
		RPi 3	2	
MCU+SBC	ATmega+RPi	ATmega2560+RPi 3	1	10%
		ATmega1281+RPi 3	1	
	ARM32+RPi	Kinetis K66 + RPi Zero	1	
FPGA		CycloneV	1	10%
		AlteraNiosII	1	
		PSoC 5LP	1	

Tabela 3.3: Dispositivos utilizados no controlo da aquisição e transmissão de dados de sensores, em sistemas inteligentes de processamento de dados de qualidade da água, utilizando uma abordagem *IoT*: Microcontroladores (MCU), Computadores de placa única (SBC) e *Field Programmable Gate Arrays* (*FPGA*).

lizadas em ambientes de desenvolvimento integrado (IDE). As linguagens de programação utilizadas na programação destes microcontroladores são *C* e *C++*, existem também algumas referências à linguagem *GRPL-µPython*, por exemplo, a placa *SMT32* possibilita a instalação do *firmware MicroPython* [140].

Além das propostas de uso de *MCU's*, existem autores (17%) (Tabelas 3.3, 3.2) que propõem como alternativa computadores de baixo custo, de placa única (*Single Board Computer-SBC*). Estes dispositivos podem executar um sistema operativo e são fáceis de utilizar [51]. São alternativas interessantes para o desenvolvimento de sistemas de monitorização da qualidade da água no âmbito da *IoT*.

Sempre que é sugerida a utilização de computadores de placa única (*SBC*) é a família *Raspberry* a mais escolhida (Tabela 3.3).

Existe um conjunto muito vasto de linguagens de programação para programar o *SBC Raspberry*. Nas publicações selecionadas, só uma propõe a plataforma *Intel Galileo Gen 2* (atualmente descontinuada) como solução para a aquisição, processamento e comunicação de dados. Esta placa é compatível com o ambiente de desenvolvimento de programas Arduíno.

A combinação de placas *MCU* ricas em recursos, de baixo custo e muito baixo consumo

de energia, com computadores de placa única (*SBC's*), proporciona outra arquitetura interessante no desenvolvimento de sistemas de monitorização da qualidade da água. Esta arquitetura é proposta em 10% das publicações utilizadas nesta análise.

Outra classe de sistemas é baseada em *Field Programmable Gate Arrays* (*FPGA*) para processamento e comunicações, esta é apresentada em 10% das publicações. *FPGA* é mais indicado para aplicações em que é exigido processamento paralelo e a altas velocidades [82]). A principal vantagem deste tipo de sistema é a possibilidade de proceder a reconfigurações, porém, para sensores genéricos e protocolos padrão, o custo geralmente alto deste tipo de sistema, e a necessidade de ferramentas e capacidades especiais de programação, não justificam o seu uso no desenvolvimento de sistemas de monitorização da qualidade da água.

Estes sistemas são programados com base em descrição de *hardware*, utilizando linguagens de descrição de *hardware* (*HDL*), por exemplo: *VHSIC*[2] ou *Verilog*[3]. O ambiente *HDL* fornece o resumo da utilização lógica do controlador. A simulação pode ser realizada através de um simulador *Modelsim*[4] [82].

No Apêndice IV, em complemento às Tabelas 3.2 e 3.3 são apresentadas de forma mais pormenorizadas, as características dos sistemas utilizados na recolha e transmissão dos dados adquiridos pelos sensores.

3.4 Transmissão de Dados

Tecnologias de Comunicação

As tecnologias de comunicação utilizadas nestes sistemas incluem geralmente classes de curto e longo alcance, no Apêndice XI encontra-se uma abordagem mais profunda às classes e tipos de tecnologias de comunicação. Alguns sistemas utilizam apenas uma classe, outros incluem ambas, dependendo da sua configuração: sistemas de uma única estação, sistemas locais ou remotos, ou sistemas de múltiplas estações (rede de estações). Todas as configurações podem incluir sistemas com fios, sem fios ou ambos.

As propostas em que recorrem à comunicações com fios são na sua maioria utilizando a arquitetura*Ethernet*, embora existam algumas propostas que apresentam as arquiteturas *RS232*, *RS485* e *SDI-12*.

A tecnologia mais frequente de curto alcance é o *WIFI*. A grande maioria das soluções estudadas utiliza esta tecnologia (Tabela 3.4), nomeadamente como complemento de outras tecnologias. Nas tecnologias de comunicação de longo alcance, a tecnologia de redes móveis (*GSM, 3G, 4G,* etc) é a que apresenta mais propostas. A utilização destas tecnologias,

[2]VHSIC ou VHDL, é uma linguagem de descrição de *hardware*.
[3]Padrão IEEE, é uma linguagem de descrição de equipamentos físicos.
[4]Simulador de linguagens de descrição de *hardware*.

não enquadráveis nas tecnologias de baixo consumo, sugere que a sua utilização se deva principalmente à sua disponibilidade e facilidade de utilização.

A maioria dos sistemas baseia-se numa rede de sensores sem fios. A tecnologia de baixa potência, *ZigBee* e outras variantes da norma *IEEE 802.15* são predominantes para comunicações de curto alcance.

A tecnologia *LoRa*, apesar de apresentar uma baixa taxa de adoção no conjunto de sistemas analisados, cerca de 12%, é cada vez mais utilizada no seio da comunidade *IoT*. É escolhida, em particular para sistemas de monitorização de qualidade da água, devido às suas características de longo alcance e baixo consumo de energia. Comparativamente à tecnologia de redes móveis (*GSM, 3G, 4G,* etc) apresenta como vantagens o facto de ser uma tecnologia de uso livre e de apresentar baixo consumo de energia. Como desvantagem, a necessidade de um investimento inicial superior ao necessário para implementar a tecnologia de redes móveis [3].

Um sistema utilizando a tecnologia LoRa foi proposto por Gao *et al.*, 2019 [39]. Os dados são adquiridos através dos sensores utilizando um *MCU* e são transmitidos usando a tecnologia *LoRa*. A informação resultante é transmitida através de um módulo *GPRS* para um servidor remoto e armazenada numa base de dados, disponibilizando-a para visualização. O desenvolvimento de *software* para levar a informação ao utilizador final foi baseado na estrutura cliente/servidor e utilizou a estrutura de desenvolvimento *LAMP*[5]. Para construir o *front-end* foi utilizado: *HTML5* + *CSS* + *JSP* [6] + *JQUERY*[7]. A linguagem de programação *Java* e a tecnologia de base de dados *MySQL* foram adotadas para contruir o *back-end*.

Tecnologia		Sist. IoT	Freq.
ZigBee	Short-range/LPLAN	9	27%
WiFi	Short-range/LAN	18	55%
Ethernet	Short-range/LAN	12	36%
Cellular Networks	Long-range/WAN	14	42%
LoRa	Long-range/LPWAN	4	12%
Outras	–	4	12%

Tabela 3.4: Tecnologias de comunicação utilizadas em sistemas inteligentes de monitorização da qualidade da água, recentemente propostos nas 31 publicações analisadas e construídos com uma abordagem *IoT*: tecnologia e classificação. Local Area Network (LAN), Wide Area Network (WAN), Low Power (LP).

Nas publicações analisadas no âmbito das *LPWAN* não se verificou qualquer referência à utilização da tecnologia *Sigfox*. Esta é uma tecnologia proprietária que permite imple-

[5] É uma combinação de programas *open-source*. O acrónimo LAMP refere-se as primeiras letras de: Linux, Apache, MariaDB ou MySQL e PHP ou Python.

[6] JavaServer Pages é uma tecnologia que ajuda os programadores a criarem páginas web geradas dinamicamente baseadas em HTML, XML ou outros tipos de documentos.

[7] Biblioteca de funções JavaScript que interage com o HTML.

mentar redes sem fios com baixa ou ultra baixa largura de banda. Possui restrições no respeita à frequência de envio de dados, o tamanho das mensagens pode no máximo utilizar 12 bytes por pulso, o que dificulta a sua utilização em aplicações que necessitem de grandes transmissões de dados [113, 112].

No âmbito das tecnologias de comunicação desenvolvidas pela organização *3GPP*, embora só se encontrem referências à utilização da tecnologias GPRS, 2G, 3G e 4G, estão atualmente disponíveis outras tecnologias que utilizam as mesmas redes. Introduzem melhorias para permitir disponibilizar a tecnologia de rede sem fios de longo alcance utilizando baixas potências (*LPWAN*), satisfazendo as necessidades da *IoT*. São disso exemplo, a *EC-GSM-IoT*[8], *NB-IOT*[9], *eMTC*[10][117]. Estas tecnologias têm como objetivos gerais a redução de custos dos equipamentos e os baixos consumos de energia. Além das taxas de transferência e alcance das comunicações, diferenciam-se pelas frequências utilizadas, suporte ou não para voz, suporte para informação georreferenciada e capacidade móvel.

Atualmente a arquitetura *5G* possui o potencial para complementar as redes existentes (tecnologias móveis, *LoRa, Sigfox*, etc), adicionando capacidade para suportar mais dispositivos, com maior equilíbrio entre velocidade, latência e custo [117]. A tecnologia *5G* tenderá a atingir o nível de desempenho que a *Internet das Coisas* exige.

Em complemento à Tabela 3.4, no Apêndice XI são apresentadas de forma pormenorizada as características das tecnologias de comunicações utilizadas para transmitir dados.

Protocolos de Comunicação

A grande maioria das soluções estudadas utiliza o protocolo de comunicações *HTTP* (Tabela 3.5) em comunicações com o cliente via Internet (pedidos e respostas), em complemento de outros ptotocolos.

Tecnologia	Sist. IoT	Freq.
HTTP	20	65%
MQTT	3	10%
Outros	6	19%

Tabela 3.5: Protocolos de comunicação utilizados em sistemas de monitorização da qualidade da água, recentemente propostos, e construídos com uma abordagem *IoT*. HTTP, MQTT e outros (SOAP, COAP, FTP, SSH, etc).

Alguns autores (10%) também recorrem à utilização do protocolo *MQTT*. É exemplo, o projeto e implementação de um sistema de monitorização da qualidade da água *IoT* para cultivo de caranguejo apresentado por Niswar *et al.* em 2018 [77]. Este consiste em

[8] Padrão baseado no LTE, um melhoramento da tecnologia GSM, utilizado em aplicações M2M ou IoT. [70 Kbit/s, 15Km].

[9] Narrow-Band, padrão baseado no LTE NB1. [50Kbit/s, 11Km].

[10] Enhanced Machine-Type Communication, padrão LTE M1[1Mbit/s].

nós sensores atuando como publicadores *MQTT*, um *SBC* atuando como intermediário e dispositivos clientes móveis como subscritores *MQTT*. Os nós sensores são construídos com *MCUs*, rádios *LoRa* e sensores de qualidade da água. O sistema de monitorização permite o acesso remoto aos níveis de qualidade da água através do servidor *Node-RED*.

Budiarti *et al.* em 2019 [9] apresentou um sistema de monitorização ambiental e de gestão de água, em que os dados transmitidos via *MQTT* são recolhidos por um *software* especifico que os armazena numa base de dados *SQLite*. A aplicação de visualização de dados utiliza um sistema *Web UI* utilizando *PHP* e *HighChart* [11] *Javascript* para a exibição de gráficos.

No âmbito da exibição de dados, Z. Zhang *et al.* [143] em 2020, utilizou o *software Grafana* para visualizar a partir da base de dados, os dados das séries temporais. C. Zhang *et al.*[144] utilizou uma aplicação cliente *web* (aplicação móvel) que lê a informação a partir da base de dados (*SQL Server*) através do protocolo *HTTP*.

No Apêndice XII são apresentadas de forma mais pormenorizada, características dos protocolos utilizados neste tipo de sistema.

3.5 Armazenamento de Dados

Cerca de metade das propostas estudadas apresentam o recurso ao armazenamento de dados na *cloud*, algumas em complemento de sistemas locais de gestão de base de dados. Não apresentam qualquer sistema de gestão de base de dados cerca de 32% das publicações.

O primeiro sistema de monitorização da qualidade água estudado no contexto do tema desta dissertação que apresenta uma solução com recurso ao armazenamento do dados na *cloud* foi proposto por Prasad *et al.* em 2015 [94]. Descreve um sistema de monitorização da qualidade da água do mar, que adquire e processa os sinais dos sensores, e os envia para a Internet através de um módulo *GSM*. Utiliza o protocolo *FTP* para enviar os dados para um servidor remoto (na *cloud*). Outros sistemas que referem o recurso ao armazenamento de dados na *cloud* são apresentados por Vijayakumar and Ramya (2015) [130], Wong and Kerkez (2016) [138], Raju e Varma (2017) [99], Encinas *et al.* (2017) [30], Kamaludin and Ismail (2017) [53], Salunke and Kate (2017) [105], Shafi *et al.* (2018) [111], Hanifah and Supangkat (2019) [42], Chowdury *et al.* (2019) [15] e C. Zhang *et al.* (2020) [143].

Destaca-se na Figura 3.6 que 58% dos sistemas analisados fazem referência à utilização de armazenamento de dados na *cloud* para a gestão de dados e que destes 32% não concretizam como é que o fazem ou que plataforma utilizam. Existe uma grande diversidade de plataformas utilizadas, com um pequeno destaque para a plataforma *ThingSpeak*.

A utilização da plataforma *ThingSpeak* foi proposta por Das and Jain em 2017 [20] e Ratnam *et al.* em 2019 [103]. É uma plataforma *IoT* de código aberto que permite

[11] Highcharts é uma biblioteca de programas para exibição de gráficos escrita em JavaScript.

construir de forma fácil e rápida, protótipos *IoT*. Fornece uma *API HTTP* e *MQTT* para envio de dados do dispositivo *IoT* para armazenar dos dados na *cloud*. Os dados podem ser visualizados e analisados na plataforma *ThingSpeak*. Inclui a ferramenta *MATLAB* para tratamento inteligente dos dados.

Plataforma *cloud*	Sist. IoT	Freq.
ThingSpeak	2	6%
thethings.io	1	3%
SCADA	1	3%
Google Drive	1	3%
IBM Watson	1	3%
Azure	1	3%
IBM Watson	1	3%
Amazon	1	3%
Outros não ident.	6	32%

Tabela 3.6: Plataformas utilizadas para armazenamento na *cloud*. Em cerca de um quinto das publicações (19%) existem referências à utilização, mas não são identificadas as plataformas.

Menon *et al.* (2017) [68] propõem uma solução utilizando a plataforma *Azure*. Esta plataforma fornece soluções para o armazenamento, monitorização remota, manutenção preditiva e mecanismos de *ML - Machine Learning*. Plataformas com valências semelhantes são propostas por Kafli e Isa (2018) [52] com a plataforma *IBM Watson* e Martínez *et al.* [63] (2020) com a plataforma *thethings.iO*.

Parameswari and Moses (2018) [83] propuseram o armazenamento dos dados numa folha de cálculo disponibilizada através do *Google Drive*.

A utilização do sistema *SCADA* é proposto por Saravanan *et al.*(2018) [108]. Utilizam este sistema para a aquisição e supervisão de dados e armazenamento na *cloud*. O *SCAD* é tipicamente uma combinação de elementos de software e de hardware, como controladores lógicos programáveis (PLCs) e unidades terminais remotas (RTUs). Os sistemas *SCADA* são sistemas com muitos recursos que empresas podem utilizar para controlar equipamentos em diferentes locais, e para adquirir e armazenar dados que relacionados com as suas operações. A análise de dados pode ser realizada em «tempo real» a partir de um qualquer navegador da Internet.

Apresentaram **sistemas locais de gestão de base dados** apenas 18% dos autores. Menon *et al.* em 2017 [68] apresentaram um *WQMS* que recorre ao armazenamento na *cloud* para gerir os dados através da plataforma *Azure*, esta plataforma utiliza o *SGBD* relacional comercial, *SQL*. T. Li *et al.* [57], Encinas *et al.* [30] em 2017, Hanifah and Supangkat [42] e Gao *et al.* [39] em 2019, propuseram o *SGBD* de código aberto e relacional, *MySQL*, para estruturar os dados adquiridos do *WQMS*. É uma base de dados cliente-servidor muito escalável entre sistemas operativos e diferentes linguagens de programação.

Budiarti *et al.* em 2019 [9], propuseram uma versão reduzida *MySQL*, o *SQLite*. Este sistema não necessita de um grande poder computacional, é fácil de configurar, indicado para bases de dados temporárias, pequenos volumes de dados e para desenvolvimento rápido. Menos segura do que o *MySQL*, não permite, por exemplo a gestão/utilização de utilizadores. Chowdury *et al.* em 2019 [15] propuseram o único sistema de base de dados *NoSQL*,

SGBD	Sist. IoT	Freq.
MySQL	3	10%
Apache Hbase	1	3%
SQLlite[1]	1	3%
SQL	1	3%

Tabela 3.7: Sistemas locais de gestão de base de dados[1] utilizados para armazenamento de dados. Apenas 18% das publicações estudadas, apresentam um *SGBD* local, 31% não apresentam qualquer sistema, nem local nem na *cloud*.

o *Apache Hbase*. Sistema de código aberto que fornece acesso aleatório e consistência para grandes quantidades de dados.

Embora não se encontrem referências nas publicações estudadas, mas com um grande potencial no âmbito *IoT*, foi testado o *SGBD InfluxDB* em articulação com o servidor *web Grafana* [55]. O *InfluxDB* é um *SGBD open-source*, que fornece armazenamento com componente temporal automática e a aplicação *Grafana* é um servidor de Internet que gera interfaces gráficas para a visualização de dados temporais [55]. Os dados utilizados na visualização através da aplicação *Grafana* podem ser obtidos diretamente através do *SGBD InfluxDB*. Os dados podem ser armazenados na base de dados, por exemplo através de uma aplicação *Python* ou *Node-RED*, ou ainda utilizando a aplicação *Telegraph*. Esta possui compatibilidade plena com o *InfluxDB* (são um produto do mesmo desenvolvedor). O *Telegraf* é um módulo que permite implementar a aquisição de dados de dispositivos, por exemplo via *MQTT*, transferindo os dados para uma base de dados temporais *InfluxDB* [55].

3.6 Conclusão

Apresentou-se aqui o estado da arte dos três subsistemas principais de monitorização automática da qualidade da água. Durante os últimos anos, estes subsistemas têm beneficiado da evolução da *IoT*, que veio facilitar a recolha dados do ambiente em grandes quantidades.

O caminho mais eficiente e económico é a monitorização automática da qualidade da água através da utilização de sensores *in-situ*, contudo não há um programa ou modelo universal para as realizar.

[1] O SQLite não é um SGBD. É uma biblioteca escrita em linguagem de programação C que implementa um mecanismo de base de dados SQL embebida.

Em síntese, é possível notar semelhanças e diferenças nas reflexões dos autores que se debruçam sobre este tema, o laço em comum apresentado é que todos reforçam a necessidade de dar resposta à monitorização da qualidade da água de forma automática e em «tempo real».

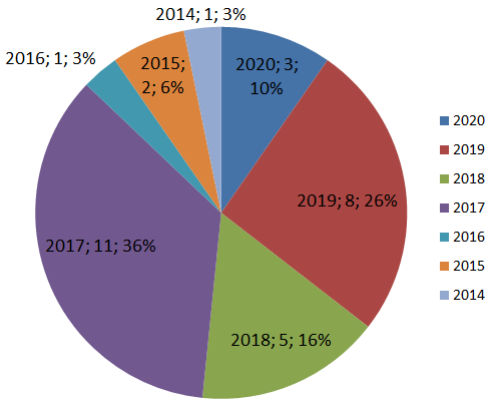


Figura 3.2: Distribuição das 31 publicações analisadas, por ano.

Apesar do foco principal deste artigo estar colocado no estudo da evolução das soluções de monitorização de qualidade da água em águas superficiais, apresenta-se aqui alguma informação que oferece suporte a este estudo:

- quase 90% das publicações referenciadas para o presente estado da arte foram publicadas nos últimos 3 anos, Figura 3.2, o que oferece um bom grau de confiança da atualidade das fontes de informação;

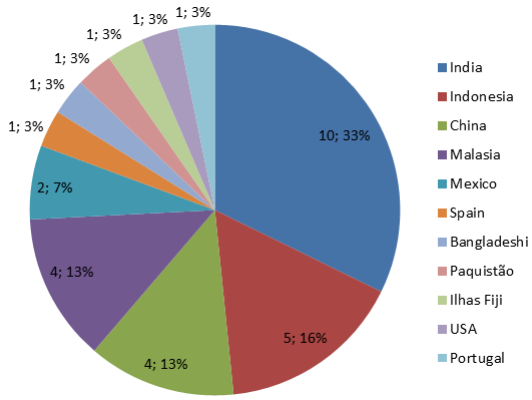


Figura 3.3: Distribuição das 31 publicações analisadas, por país.

- em termos geográficos o presente estudo abrange publicações de 11 países, Figura 3.3, com grande incidência na Índia, país que apresenta um domínio no número de publicações, ao que não deve ser alheia a extrema escassez de água [134]. Uma em cada cinco pessoas sem acesso a água no planeta, vive na Índia;

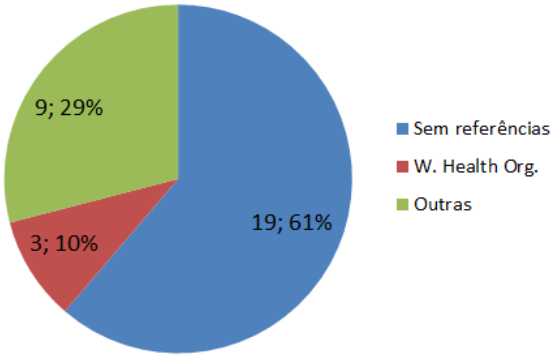


Figura 3.4: Publicações com referências a parâmetros legais da qualidade da água. Mais de metade das publicações, não fazem referência aos valores legais, dos parâmetros da qualidade da água.

- a maioria das publicações utilizadas não demonstra preocupação em validar os dados recolhidos pelos sensores com os parâmetros definidos pelas autoridades de cada país, Tabela 3.4. Daqueles que fazem alguma referência confirma-se a não existência de um padrão no que concerne a essas possíveis referências. Observa-se que ainda assim, que são as referências à organização Mundial *WHO* [134] que prevalecem;

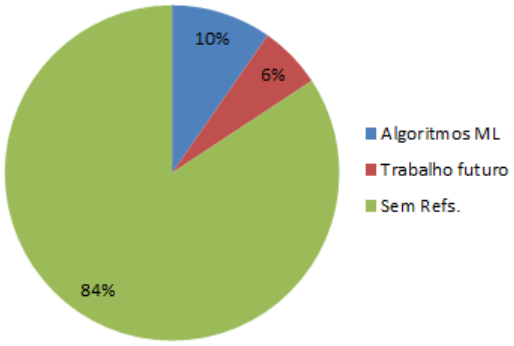


Figura 3.5: Publicações com referências à utilização de *Machine Learning*. Apenas 10% das 31 publicações analisadas faz referência à utilização de técnicas de *Machine Learning*.

- observa-se ainda, que são poucas as referências ao uso de *Machine Learning* para prever e agir sobre a qualidade da água.

  Das propostas consideradas na análise do estado da arte desta dissertação, a primeira a apresentar uma abordagem *ML* foi apresentada por Prasad *et al.* em 2016 [94] onde descreveram um sistema de monitorização da qualidade da água do mar, em que utilizam as diferentes fontes de dados de qualidade da água para construir classificadores que se destinam a realizar análises automatizadas de água em forma de *Análise de Rede Neural*.

  Encinas *et al.* em 2017 [30], referiram também a necessidade de futuramente utilizar

a análise preditiva dos dados recolhidos, utilizando algoritmos de *Machine Learning* (ML), para detetar e prevenir eventos que contribuam para a perda de qualidade da água.

Um protótipo para registar os parâmetros de qualidade da água em «tempo real», de amostras recolhidas de várias fontes foi proposto por Shafi *et al.* em 2018 [111], utiliza a análise preditiva dos dados recolhidos é realizada utilizando algoritmos de *Machine Learning* (*ML*) aplicados ao desafio de classificar e gerir de forma mais eficiente a qualidade da água. As amostras de água são utilizadas para *Machine Learning* utilizando os algoritmos *SVM* (Máquina de Suporte Vetorial), *NN* (Redes Neurais) e *kNN* (K-vizinhos mais próximos) . Técnicas de *ML* também foram propostas por Chowdury *et al.* em 2019 [15]. O sistema propõe a integração de *Big Data Analytics (Apache Spark), Deep Learning (DL)* e Redes Neurais.

Gao *et al.* em 2019 [39] propuseram para realizar a análise e previsão de dados, a utilização do algoritmo *LOF* (*local outlier factor* para filtrar os dados, removendo os dados irregulares. A tendência de alterações para cada parâmetro é previsto usando o algoritmo modelo de árvore.

A utilização de *ML* ainda ocorre com pouca frequência como se pode verificar na Figura 3.5, apenas 10% das propostas apresentam a utilização de algoritmos de *ML* para melhorar os *WQMS* (o primeiro em 2018) e 6% demonstram essa intenção para trabalho futuro.

No Apêndice VI são apresentadas algumas soluções comerciais existentes no mercado, destinadas à monitorização da qualidade da água.

Capítulo 4

Arquitetura do Sistema

Neste capítulo é apresentada e descrita a arquitetura de um sistema que adota um paradigma *IoT*, para monitorizar a qualidade da água, com ênfase nos seus aspetos de *software*.

Na primeira Secção (4.1) é apresentado o diagrama de camadas dum sistema, destinado à monitorização de qualidade da água. São caraterizadas 4 camadas: dispositivos; rede; suporte de serviços e suporte de aplicações. Cada camada é constituída por módulos que oferecem um conjunto coeso de serviços.

Na segunda Secção (4.2) é discutida uma arquitetura *IoT* para monitorização de águas superficiais. São apresentados os vários componentes do sistema, como interagem entre si e com os utilizadores. O sistema pode ser dividido nos seguintes tipos de subsistemas de processamento de dados: aquisição; transmissão e gestão, que inclui o armazenamento e processamento de dados de alto nível.

A arquitetura de sistema de aquisição de dados e de comunicação dos dispositivos contém os módulos de qualidade da água e estação meteorológica, que adquirem e transmitem os dados dos sensores através da tecnologia *LoRa*.

A arquitetura de comunicações *LoRaWAN* é adotada no sistema de transmissão de dados. Assenta em *ChirpStack*, uma pilha de servidores e serviços *LoRaWAN*.

A arquitetura do sistema de gestão de dados inclui a subscrição dos dados do *MQTT Broker*, o processamento, armazenamento, análise e exibição aos utilizadores.

4.1 Modelo de Camadas

A arquitetura de um sistema *IoT* pode ser representada através de diferentes tipos de diagramas [110]. Uma arquitetura de referência da *IoT* amplamente adoptada, organizada em camadas, onde cada camada agrupa módulos que oferecem um conjunto coeso de serviços, é definida pela União Internacional de Telecomunicações (UIT). Consiste em quatro camadas: dispositivos; rede; suporte de serviços e suporte de aplicações [49]. Um sistema desta natureza apresenta diferentes componentes/blocos, que comunicam entre si. Um diagrama baseado em camadas, da arquitetura de um sistema geral de monitorização de

águas superficiais é representado na Figura 4.1. Exemplifica como os dados são recolhidos nos níveis mais baixos e fluem através do sistema até serem apresentados aos utilizadores. No nível mais baixo, da natureza, temos as fontes de dados: águas superficiais e atmosfera.

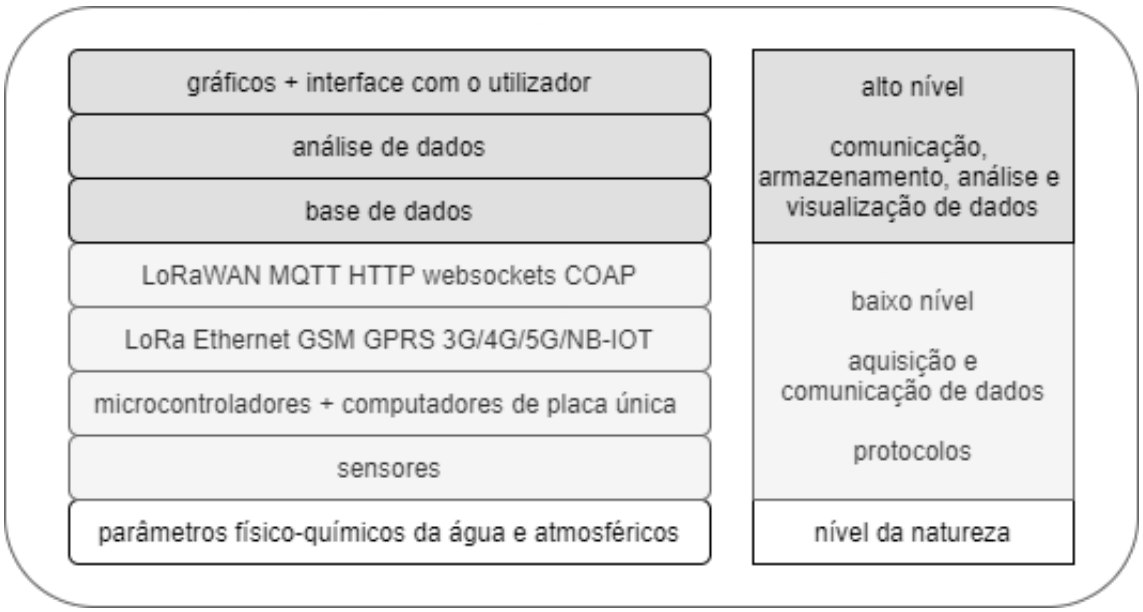


Figura 4.1: Diagrama de camadas de um sistema *IoT* de monitorização de qualidade da água. Cada camada é constituída por módulos que constituiem um conjunto coeso de serviços.

O nível seguinte compreende uma rede de sensores *in situ*, *MCU's*, *SBC's* e, tecnologia e protocolos de transmissão de dados. Os sensores são responsáveis por fornecer medições dos parâmetros físicos e químicos, da água de superfície e da atmosfera circundante. Utilizando módulos de aquisição de dados, *hardware* de comunicações e protocolos de *software*, os *MCU's* e *SBC's* recebem e transmitem periodicamente os dados recolhidos, para a camada de rede responsável pela transmissão de dados para outros dispositivos e servidores de rede.

A camada de rede compreende as tecnologias de comunicação e os protocolos de transmissão de dados. A escolha das tecnologias de comunicações é definida pelas condições existentes no local de aquisição de dados, em particular as condições físicas, a disponibilidade das comunicações, e o tipo de fontes de energia existentes. Como ilustrado na Figura 4.1, a tecnologia de comunicações escolhida pode ser a *LPWAN LoRa* ou outras tecnologias baseadas em *Ethernet*, *GSM*, *GPRS*, *3G*, *4G*, *5G*. Pode ser a combinação de mais do que uma destas tecnologias. Os protocolos de transmissão de dados são responsáveis pelas comunicações com serviço de suporte e com a camada aplicacional. O diagrama apresenta uma série de protocolos de aplicação como *LoRaWAN*, *MQTT*, *HTTP*, *WebSockets* e *CoAP*, normalmente utilizados em sistemas *IoT*.

O nível superior inclui o processamento, análise, armazenamento e a visualização dos dados recolhidos. Interage com os utilizadores através de ecrãs, formulários, gráficos, menus

e relatórios. É a camada mais visível da aplicação e define o seu aspecto.

Os dados podem ser analisados de forma a realizar previsões e sugestões de acção. No domínio da tecnologia de bases de dados, existem dois tipos primários de bases de dados: relacionais e não relacionais. A qualidade da informação é importante para extrair conhecimento de alto nível. Esta depende da informação captada por cada sensor e de vários outros fatores, nomeadamente erros nas medições, precisão e exatidão da aquisição de dados, o ruído ambiental e da conversão digital das medições [61].

4.2 Arquitetura IoT para Monitorização de Águas Superficiais

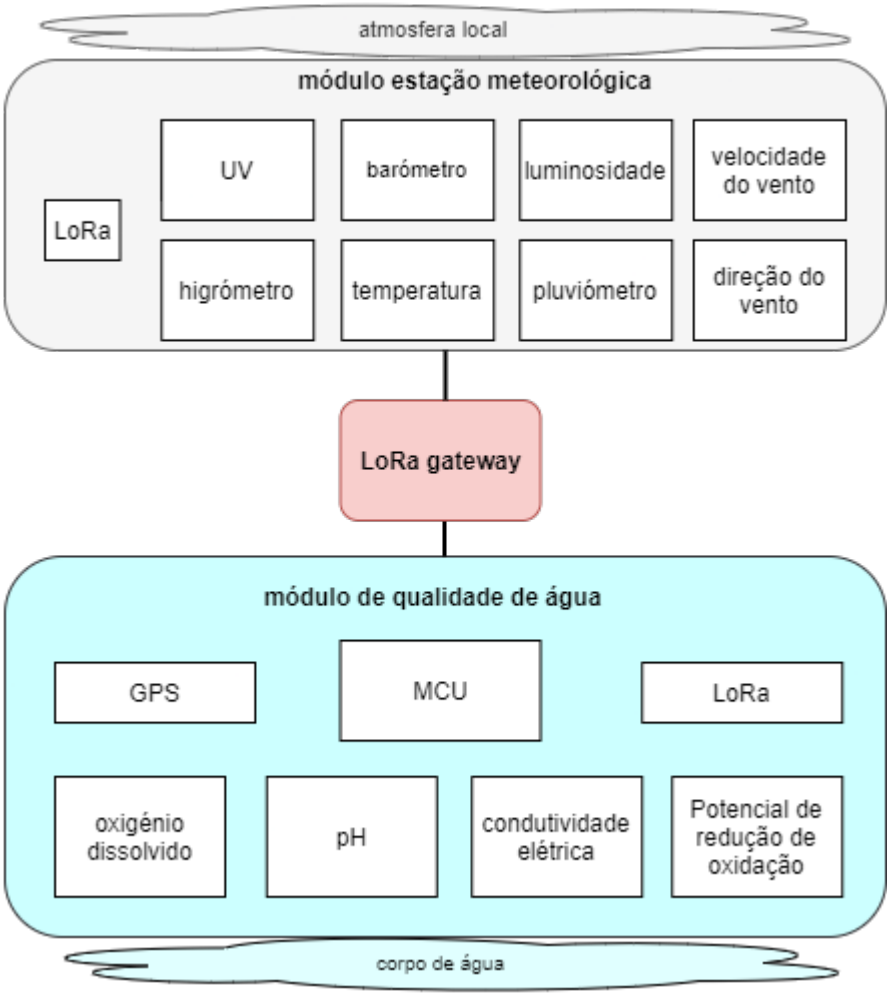


Figura 4.2: Arquitetura de aquisição de dados e de comunicação dos dispositivos. Os módulos de qualidade da água e estação meteorológica, adquirem e transmitem os dados dos sensores através da tecnologia LoRa.

A arquitetura proposta para o sistema de aquisição e análise de dados, de qualidade

da água e meteorológicos, está representada nas Figuras 4.2, 4.4 e 4.5. O sistema pode ser dividido, *grosso modo*, nos seguintes tipos de subsistemas de processamento de dados: aquisição; transmissão e a gestão de dados; que inclui o armazenamento e processamento de alto nível.

O subsistema de aquisição de dados está representado na Figura 4.2. Este inclui o *Módulo de Qualidade da Água* e o *Módulo de Estação Meteorológica*. Ambos os módulos entregam dados à *gateway* através da tecnologia *LoRa*. O *Módulo da Qualidade da Água* faz a aquisição dos parâmetros físicos e químicos de qualidade da água mais comuns, nomeadamente: *pH*; *ORP*; *DO* e *EC*. A *Estação Meteorológica* fornece dados do ambiente/atmosfera circundante, nomeadamente: temperatura; humidade e pressão do ar; precipitação; velocidade e direcção do vento; luminosidade e radiação ultra-violeta(UV).

O sistema de transmissão de dados utiliza os componentes de rede desenvolvidos pelo *LoRa Server Project Chirpstack* (Figura 4.3), um projeto *open-source* que fornece componentes para a implementação de redes *LoRaWAN* [13].

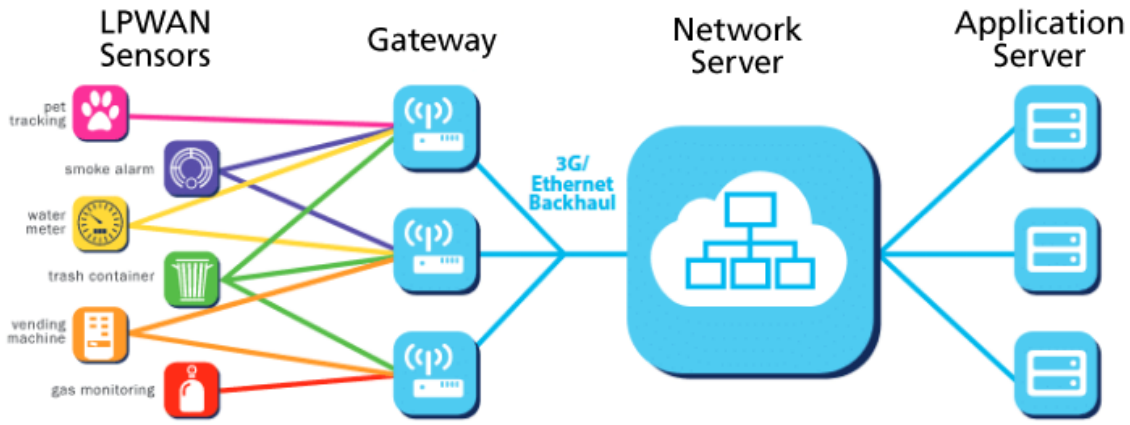


Figura 4.3: Estrutura de rede LoRaWAN - Chirpstack

Fonte: (https://www.chirpstack.io/project/architecture/)

Todos os componentes são cobertos por uma licença *MIT* [72], e podem ser utilizados para fins comerciais. Estes componentes formam uma solução pronta a utilizar, incluindo um interface *web*, *APIs gRPC* [1] e *REST*. Os componentes principais são: *LoRa Gateway Bridge*; *LoRa Network Server*; *LoRa Application Server* e *MQTT Broker*, Figura 4.4. A *gateway LoRa* recebe a informação dos módulos de aquisição de dados (dispositivos *LoRaWAN*), que podem encontrar-se a distâncias que vão desde algumas dezenas de metros até alguns quilómetros.

[1] O gRPC é uma estrutura *opensource* que permite chamadas de procedimentos remotamente.

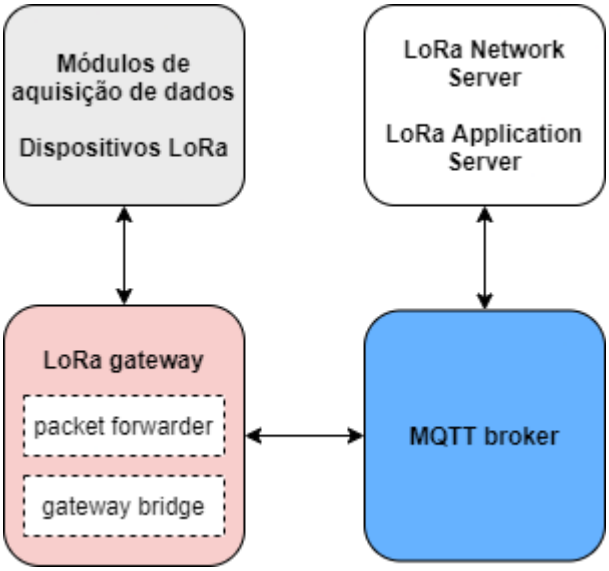


Figura 4.4: Arquitetura de Comunicações *LoRaWAN* - O sistema de transmissão de dados adota o *ChirpStack*, uma pilha de servidores e serviços *LoRaWAN* de código aberto com cinco componentes principais: *LoRa Gateway Bridge*, *LoRa Network Server*, *LoRa Application Server* e *MQTT Broker* [13].

O programa responsável por enviar e receber os dados adquiridos pelo do serviço *LoRa Gateway Bridge* é denominado *Packet Forwarder*. As implementações mais comuns deste programa são o *Semtech UDP Packet Forwarder* e o *Semtech Basic Station Packet Forwarder*. Converte as mensagens dos módulos de aquisição de dados, recebidas pelo chipset de rádio *LoRa*, em mensagens no formato *JSON* para os módulos de aquisição de dados, enviadas pelo *chipset* de rádio *LoRa*. Comunica com os módulos de aquisição de dados através do protocolo *LoRaWAN* e com o *LoRa Gateway Bridge* através do protocolo UDP. O *LoRa Gateway Bridge* converte os dados no protocolo UDP e envia/publica para o *LoRa Network Server* através de uma subscrição num servidor *MQTT*.

O *MQTT Broker* é utilizado como mediador entre o serviço *LoRa Gateway Bridge* e os dois servidores, *LoRa Network Server* e *LoRa Application Server*.

O *LoRa Network Server*:

- encaminha as mensagens recebidas do serviço *LoRa Gateway Bridge* para o *LoRa Application Server* e vice-versa;
- cifra as mensagens da aplicação utilizando o algoritmo AES de 128 bits, com a chave: Network Session Key;
- gere o acesso à rede *LoRaWAN* na segunda camada do modelo OSI (MAC) [2];

[2] A camada de enlace faz o controle de fluxo da transmissão dos dados, detetando e corrigindo erros do

- remove mensagens duplicadas quando mais do que uma *gateway* recebe a mesma (se existir mais do que uma);
- agenda o envio de mensagens para os módulos de aquisição de dados e decide qual a *gateway* melhor posicionada para enviar as mensagens;
- comunica com o *LoRa Gateway Bridge* através do protocolo *MQTT* e com o *LoRa Network Server* através do protocolo *gRPC*;
- armazena os dados de registo dos dispositivos e *gateways* numa base de dados relacional orientada por objetos, *PostgreSQL* e utiliza o *SGBD NoSQL Redis* para armazenar temporariamente os dados de sessão de dispositivo e dados não persistentes, permitindo monitorizar os eventos em «tempo real».

O LoRa Application Server, servidor de aplicações *ChirpStack* é responsável pelo tratamento dos dados adquiridos dos dispositivos *LoRa*, pela gestão e criptografia na aplicação e o controlo de pedidos. Fornece uma interface *Web* e *APIs* para gerir utilizadores, organizações, aplicações, *gateways* e dispositivos (Figura XIII.3). Armazena os dados das configurações de todos os componentes e definições da rede *LoRaWAN* numa base de dados *PostgreSQL*.

O subsistema de gestão de dados representado na Figura 4.5, realiza os serviços de armazenamento de dados, servidor *HTTP*, sistema *GIS*, aplicação de armazenamento e visualização de dados, e *Node-RED*.

O *MQTT Broker* publica os dados dos módulos de aquisição de dados.

As aplicações de armazenamento e processamento de dados são escritas na linguagem de programação *Python* e utilizam as *APIs* disponíveis para *PostgreSQL (psycopg2* [3]*)* e *QGIS*.

A *Data Storing App* desenvolvida em linguagem *Python*, com a estrutura apresentada na Figura 4.6, subscreve todos os tópicos com dados dos sensores, publicados pelo *MQTT broker* e armazena a informação no *SGBD* . Existem numerosas opções comerciais e *open-source* para armazenamento e gestão de dados. Optou-se por utilizar o sistema de gestão de base de dados relacionais *PostgreSQL*, com as extensões geográficas *PostGIS*, e o sistema de informação geográfica QGIS, porque são fornecidas como *software open-source*, integram-se facilmente e têm bom registo de fiabilidade. O *PostGIS* é a ferramenta para dados espaciais do *SGBD PostgreSQL*. No *PostGIS* existe suporte para objetos geográficos, permite realizar consultas de localização na linguagem *SQL* [88].

Pela disponibilidade, rapidez e facilidade de operação, numa primeira fase, para testar, monitorizar e os visualizar dados publicados pelo *MQTT Broker* foi utilizado o servidor

nível físico. Utiliza o MAC Adress para encaminhar os pacotes.

[3] Psycopg é o adaptador *Python* para base de dados PostgreSQL [40]

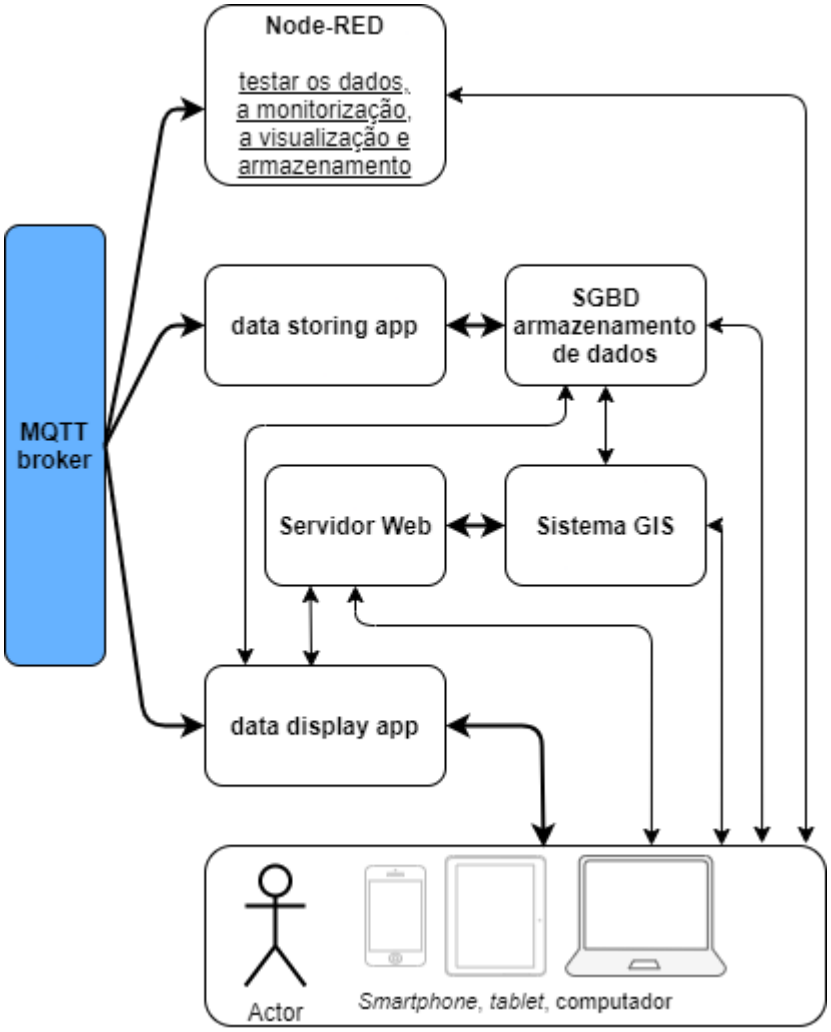


Figura 4.5: Arquitetura de Gestão de Dados. Os dados subscritos no *MQTT Broker* ficam disponíveis para análise, armazenamento e uso geral pelos utilizadores.

open-source Node-RED [78] baseado em *Node.js*. O *Node-RED* é uma ferramenta de desenvolvimento baseada no conceito de fluxos e nós, com uma interface que permite realizar programação com apoio visual. Apresenta um editor de fluxos em ambiente *web*, que tem a capacidade de criar e interpretar funções *JavaScript* (podendo os elementos desenvolvidos ser guardados e partilhados para novas utilizações). O compilador é construído «em cima» da *framework* do *Node.js*, sendo os fluxos gerados e armazenados no formato *JSON*[4].

A interface com o utilizador é disponibilizada pela aplicação *data visualization app*, fornece um meio para os utilizadores visualizarem os resultados das medições. Filtra a informação da base de dados e fornece informação aos utilizadores através de um servi-

[4] JavaScript Object Notation (JSON) é um formato baseado em texto utilizado para representar dados estruturados, com base na sintaxe de um objeto JavaScript. Os arquivos JSON utilizam a extensão .json.

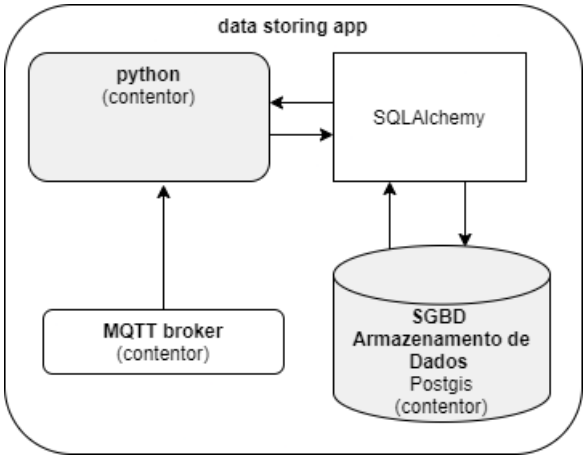


Figura 4.6: Arquitetura da aplicação de armazenamento de dados. Os dados subscritos e recebidos do *MQTT Broker* são armazenados no *SGBD* com uma aplicação escrita na liguagem *Python* utilizando o pacote *SQLAlchemy*

dor *HTTP*, baseado no servidor *proxy NGINX* e no servidor *HTTP Gunicorn* [5] *UWSGI* [6]. Esta aplicação é escrita em linguagem de programação *Python* e adopta o *microframework web Flask*, Figura 4.7. O *microframework web Flask* utiliza uma abordagem padrão *Model-View-Controller* e a *SQLAlchemy* é utilizada como componente *Modelo*. O *View* e o *Controller* já fazem parte do *Flask*.

O projeto e implementação de *software* é baseado na *tecnologia de contentores*, o que permite uma implementação fácil, escalável e portátil, reduzindo a complexidade da gestão, reprodução e replicação de configurações experimentais em sistemas em fase de desenvolvimento.

4.3 Conclusão

A *Internet das Coisas* (*IoT*) tornou possível a aquisição de grandes quantidades de dados do mundo físico, em particular do meio ambiente.

Neste capítulo foi proposta uma arquitetura baseada na *IoT*, projetada para a aquisição, transmissão e tratamento de dados. Esta satisfaz o objetivo proposto de implementação de um sistema de monitorização da qualidade da água.

[5] Gunicorn (*Green Unicorn*) é um servidor HTTP Python UWSGI (*Web Server Gateway Interface*) que encaminha solicitações de um servidor *HTTP* (Apache, NGINX, etc.) para uma aplicação Python *web* ou estrutura de *backend*.

[6] O uWSGI é um servidor e um dos protocolos implementados é o WSGI, utilizado para comunicar com os servidores *web* para balanceamento de carga e, principalmente, para aproveitar os recursos extras que o *HTTP* não pode fornecer. O *NGINX* implementa este protocolo.

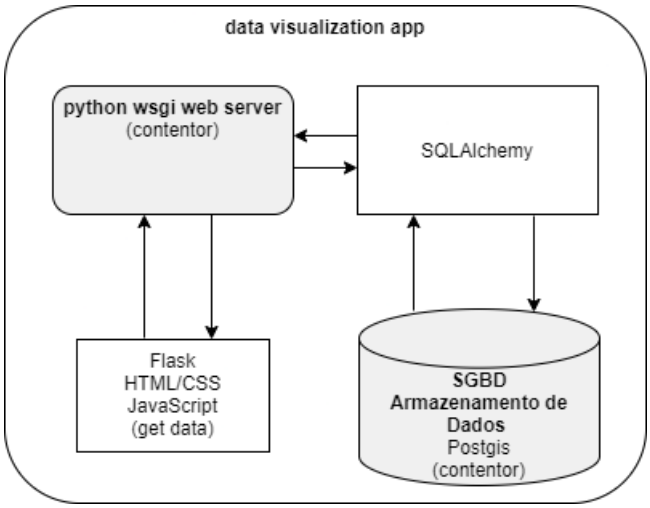


Figura 4.7: Aplicação web Flask (*microframework* do ecossistema *Python*). Processa em «tempo real» os dados armazenados e disponibiliza também em «tempo real» os dados em ambiente da web utilizando JavaScript, CSS e HTML.

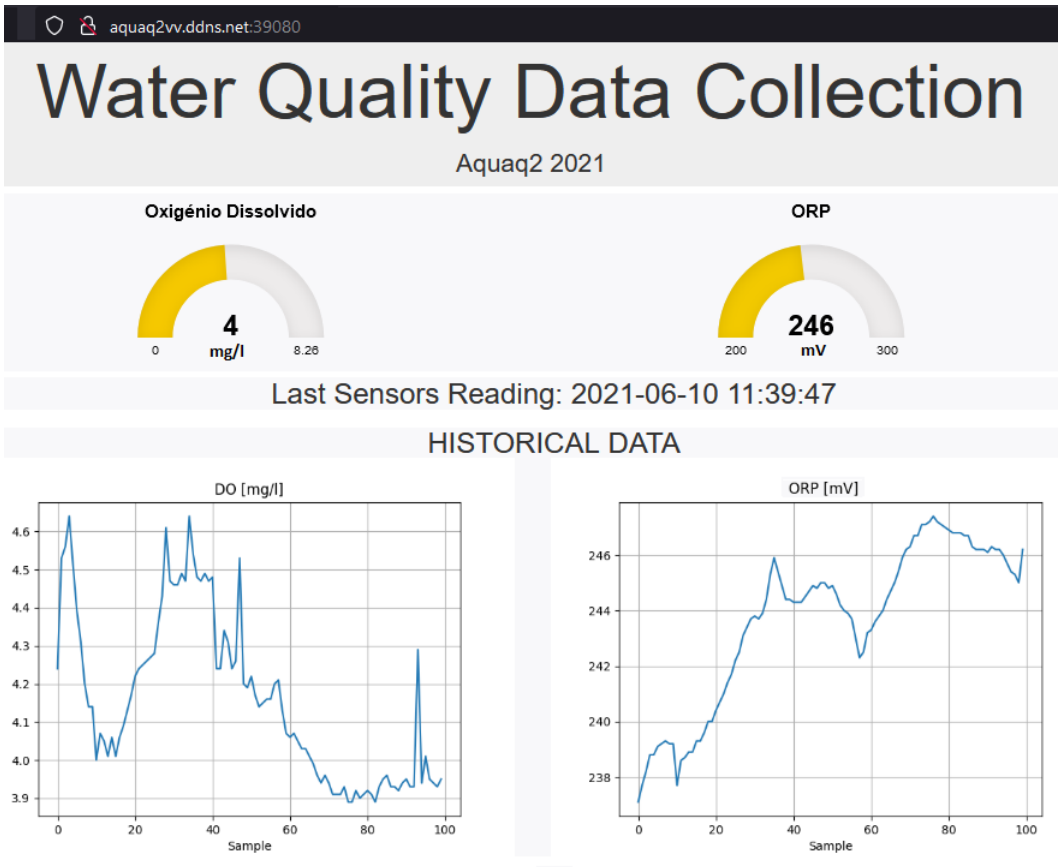


Figura 4.8: Resultado da execução da aplicação de visualização de dados. Exibe leituras em «tempo real» de cada sensor no módulo de monitorização de qualidade da água e o histórico de leituras.

O desenvolvimento de *software* para proceder ao armazenamento dos dados adquiridos através dos módulos de aquisição de dados e para visualização dos mesmos 4.8, permitirá observar os dados históricos e em «tempo real». Esta informação chegará aos utilizadores e decisores em «tempo real» através de qualquer equipamento com acesso à Internet, oferecendo-lhes conhecimento para tomar melhores e oportunas decisões.

Capítulo 5

Realização do Sistema

O protótipo é composto por vários componentes de *hardware* (Apêndice VII) e de *software*. O *software* é escrito para *hardware* dedicado, como é o caso do *MCU*, ou para *hardware* partilhado, como na pilha de servidores e serviços do *LoRa Server Project*(*ChirpStack*).

Neste capítulo, em complemento do que é apresentado na Secção 4.2, discutem-se as realizações de componentes de *software* deste projeto. Nomeadamente:

- *tecnologia de contentores Docker*, para implementar um conjunto de aplicações e serviços, que dão suporte à arquitetura de *software*;
- a configuração e utilização do conjunto de serviços *ChirpStack*, componentes de rede desenvolvidos pelo *LoRa Server Project* para implementar redes *LoRaWAN*;
- a implementação de um SGBD, com capacidade para gerir informação geográfica, para gerir e armazenar, os dados gerados pelos módulos de aquisição de dados;
- a utilização do sistema operativo *Linux Ubuntu Server 20.04 LTS - Distribuição: focal*, dedicado a gerir todos os servidores e serviços através da *tecnologia de contentores Docker*;
- a utilização do ambiente de programação *open-source Node-RED* e da linguagem de programação *Python*, para o desenvolvimento das aplicações de armazenamento e visualização de dados;
- a utilização do *microframework Flask*, para o desenvolvimento da aplicação de visualização de dados.

5.1 Tecnologia de *Contentores*

Um *contentor* no ambiente Linux tem analogia ao conceito de *chroot*[1]. É um processo executável que permite isolar componentes do sistema. Consiste numa *imagem*[2], num ambiente de execução e num conjunto padronizado de instruções, definindo uma forma padrão de distribuição de *software*. O sistema de gestão de *contentores* permite a execução de diversos processos e aplicações separadamente, facilitando a utilização e gestão da infraestrutura.

As tecnologias de *contentores* permitem, empacotar e isolar aplicações, com todo o ambiente de execução, colocando-as num subsistema e permitem que vários *contentore* possam funcionar sobre um único sistema operativo. Torna os servidores e serviços mais eficientes e mais célere a implementação de aplicações. Facilita também, a migração de aplicações de um ambiente (desenvolvimento, teste e produção) para outro sem perder funcionalidades. Reduz a complexidade da gestão, implantação e portabilidade do *software* [29].

Existem várias tecnologias de *contentores* [114] [29], nomeadamente:

- *Linux containers (LXC)*
- *CoreOS*
- *Rocket (rkt)*
- *KVM virtualization*
- *Docker*

Devido à facilidade de utilização, popularidade, documentação disponível e ser especialmente dedicado a *contentores* aplicacionais, foi selecionada a tecnologia de *contentores*, *Docker*, para implementar a arquitetura de *software* definida no Capítulo 4.

Docker

A «Docker Inc.» é a empresa que desenvolveu o «*Docker*», foi fundada como «dotCloud Inc.» em 2010, por Solomon Hykes. Desenvolveu ferramentas, para implmentar e gerir, *contentores Linux*, utilizando as ferramentas *cgroups* e *namespace*, do *Kernel Linux*, reduzindo a complexidade em torno da utilização de *contentores* [8].

O *Docker* é um conjunto de *software open-source*, que constitui um ecossistema baseado numa arquitetura cliente/servidor, Figura 5.1. Fornece um modelo de implementação com base em *imagens*, que facilita a partilha de aplicações ou serviços, incluindo todas as

[1] Processo com recursos do sistema operativo Linux que permite isolar e manter uma distribuição *Linux* secundária instalada dentro da distribuição primária.

[2] É um ficheiro (modelo) que contêm um conjunto de instruções, a partir das quais se pode construir um *contentor*.

dependências em vários ambientes [43]. O cliente *Docker* envia comandos (utilizando API REST ou sockets UNIX) para que o *docker daemon* (servidor) execute as tarefas, por exemplo: construir, obter *imagens* e executar *contentores*. Neste ecossistema, temos [22]:

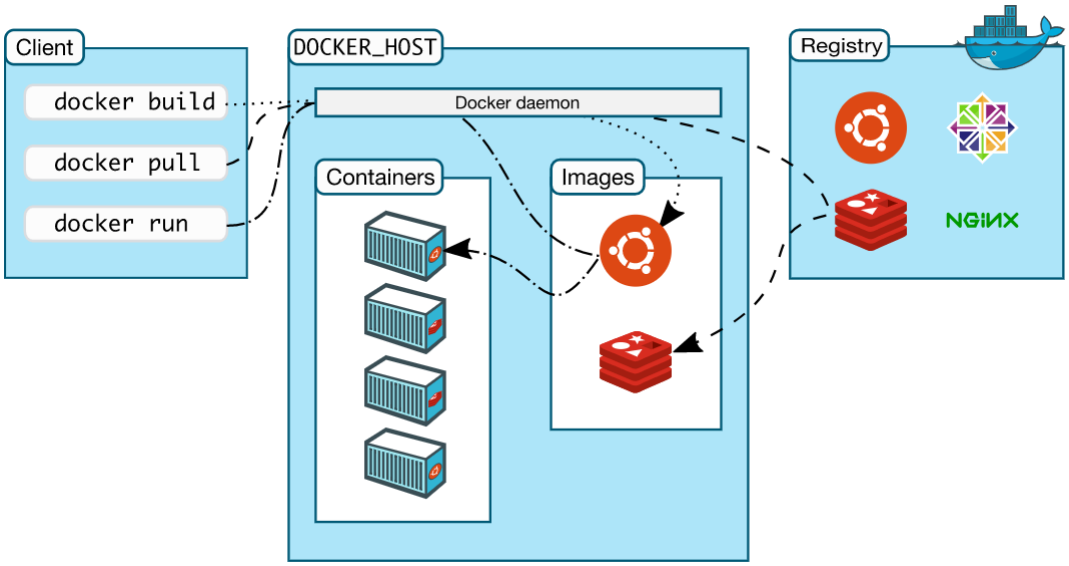


Figura 5.1: Arquitetura Docker - O cliente *Docker* comunica com o processo *daemon Docker (dockerd)*, responsável por construir, executar e distribuir *contentores Docker*. O *Registry* (registo *Docker*) armazena imagens *Docker*. O comando *docker pull* ou *docker run*, extrai as imagens. O comando *docker push*, envia a *imagem* para o registo. Uma *imagem* é um modelo com instruções, para criar um *contentor Docker*. Um *contentor* é uma instância executável de uma *imagem*. Quando um *contentor* é removido, todas as alterações ao seu estado, não armazenadas persistentemente, são perdidas.

Fonte: (https://docs.docker.com/get-started/overview/ - consultado em 20-05-2021)

- o *Docker Engine*, o *software* base da solução;
- o *Docker Compose*, a ferramenta responsável pela definição e execução de múltiplos *contentores* com base no ficheiro *docker-compose.yml*;
- o *Docker Machine*, a ferramenta que possibilita criar e manter ambientes *Docker* em máquinas físicas, virtuais ou, ambientes na *cloud*;
- o *Docker Client*, que possui ferramentas para comunicar com o processo *daemon* [3] *Docker (dockerd)*, responsável por construir, executar e distribuir *contentores Docker*;
- o *Docker Registry*, repositório para armazenamento e distribuição de imagens previamente criadas. Depois das imagens serem «construídas», pode-se realizar o *upload* para o *registo*.

[3] É um programa, que é executado como um processo em segundo plano.

A tecnologia de *contentores Docker* utiliza recursos do *kernel* do *Linux* e das suas características, como o *cgroups*[4] e *namespaces*[5].

Um *contentor Docker* é uma instância executável de uma *imagem*. É um conjunto de *cgroups* e *namespaces*.

A *imagem* utilizada para criar um *contentor* é composta por instruções apenas de leitura, a partir do qual são criados os *contentores* (instâncias de uma *imagem Docker*). Pode conter um sistema operativo ou uma aplicação. As imagens são construídas a partir de uma *imagem* base, utilizando um conjunto de instruções, cada uma, cria uma nova camada e a adiciona à *imagem*. Estas instruções estão armazenadas num arquivo denominado *Dockerfile*. Quando é solicitada a compilação de uma *imagem*, o *Docker Engine* lê esse arquivo, executa as instruções, e retorna uma *imagem* final. A solicitação de criação de uma *imagem* é realizada, via cliente do *Docker*, com o comando «*docker build*» [29]. Todas as instruções, no *Dockerfile* são executadas de forma sequencial incluindo a instalação de pacotes, criação de diretórios e definição de variáveis de ambiente [47].

Todos os servidores e serviços implementados no sistema descrito nesta dissertação são geridos com o *software Docker* através de *Dockerfiles* e ficheiros de configuração *Docker-Compose*.

O *Docker-Compose* é uma ferramenta para definir e executar aplicações *Docker* multi-*contentores*. Utiliza um ficheiro *YAML* [6] para configurar os serviços associados a uma aplicação. Cria e inicia, todos os serviços a partir da sua configuração, executando um único comando.

Os *volumes* são o mecanismo «preferido» para dados persistentes, gerados e utilizados pelos contentores *Docker* [22]. Enquanto as áreas de armazenamento (*bind mounts*) criadas nos *contentores*, dependem da estrutura de diretórios e do sistema operativo da máquina anfitriã, quando se utilizam *volumes*, um novo diretório é criado dentro do diretório de armazenamento do *Docker*, ficando o *Docker*, responsável pela gestão desse conteúdo. Os *volumes* têm várias vantagens sobre os suportes *bind mount*. É mais fácil realizar cópias de segurança de *volumes*, que podem ser geridas utilizando os comandos *Docker CLI* ou o *Docker API*. Podem ser partilhados de forma mais segura entre múltiplos contentores e a

[4] Permite alocar recursos, tais como tempo do CPU, memória, largura de banda de rede ou combinações destes recursos.

[5] Permite isolar processos (redes, etc) de tal forma que não podem «ver» os recursos uns dos outros.

[6] YAML significa *Yet Another Markup Language*.. É um formato de dados legível para humanos usado para serialização de dados, para ler e gravar dados independentemente de uma linguagem de programação específica.

eliminação de um *contentor* não elimina os dados gerados pelo mesmo.

A rede *Docker* é uma abstração, criada para facilitar a comunicação entre *contentores* e os nós externos ao ambiente *Docker*. O *Docker* disponibiliza, por defeito, três redes (Tabela 5.3) [22]. Estas redes, oferecem configurações específicas para gestão do tráfego de dados. A rede *Bridge* é a rede padrão para qualquer *contentor*, a menos que se associe explicitamente a outra rede. Esta rede, disponibiliza ao *contentor* um interface que faz *bridge* com o interface *docker0* da máquina *Docker*. Este interface recebe, automaticamente o próximo endereço disponível na rede IP 172.17.0.0/16. A rede *None*, tem como objetivo isolar o *contentor* das comunicações externas, não recebe qualquer interface para comunicação externa. O único interface de rede IP, será a máquina local. A rede *Host*, tem como objetivo, expor ao *contentor* todas os interfaces existentes na máquina *Docker* e agiliza a entrega dos pacotes, uma vez que não há *bridge* no encaminhamento das mensagens. As redes definidas pelo utilizador são associadas a um determinado *driver* de rede, se o utilizador não criar o seu próprio *driver*, que deve escolher entre os *driver*s disponibilizados pelo *Docker*.

O *Docker*, destaca-se na facilidade de utilização e compatibilidade com vários sistemas operativos. É uma ferramenta que oferece simplicidade, agilidade e estabilidade nos processos de desenvolvimento, permite experimentar rapidamente programas, isoladamente ou em conjunto, sem afetar o sistema operativo ou outros programas que estamos a utilizar.

Pilha de *Contentores* - Portainer

O sistema de gestão de *contentores*, executado também a partir de um *contentor*, assenta no *software Portainer* (Figuras: 5.2 e XIII.1).

O *Portainer* disponibiliza uma das muitas interfaces para *Docker* sendo uma das mais populares, com muitas funcionalidades e desenvolvimento ativo [8]. É uma solução para gestão de recursos, como imagens e *contentores Docker*, redes e volumes. O *Portainer* é gratuito e *open-source* [87]. Através do menu de *contentores* é possível excluir, executar, parar, iniciar e adicionar *contentores*, além de disso, para cada *contentor* é possível, por exemplo, visualizar os *logs* e estatísticas, e aceder à consola de linha de comandos.

Na Tabela 5.1 e na Figura XIII.1, estão representados os três *contentores* que constituem a pilha de *contentores Portainer*. O *contentor* «*portainer_portainer_1*», que contém a aplicação *Portainer*, o «*portainer_db_1*», que contém o sistema de gestão de base de dados *MariaDB*[7], destinado a armazenar os dados gerados pela aplicação *Portainer*, e o «*portainer_nginxproxymanager_1*», que contém o servidor proxy, *Nginx Proxy Manager*.

No ficheiro *docker-compose.yml*, (Apêndice VIII) podem ser consultadas as configurações utilizadas, na criação dos três *contentores* da pilha *Portainer*. São utilizadas as

[7] Versão livre do MySQL

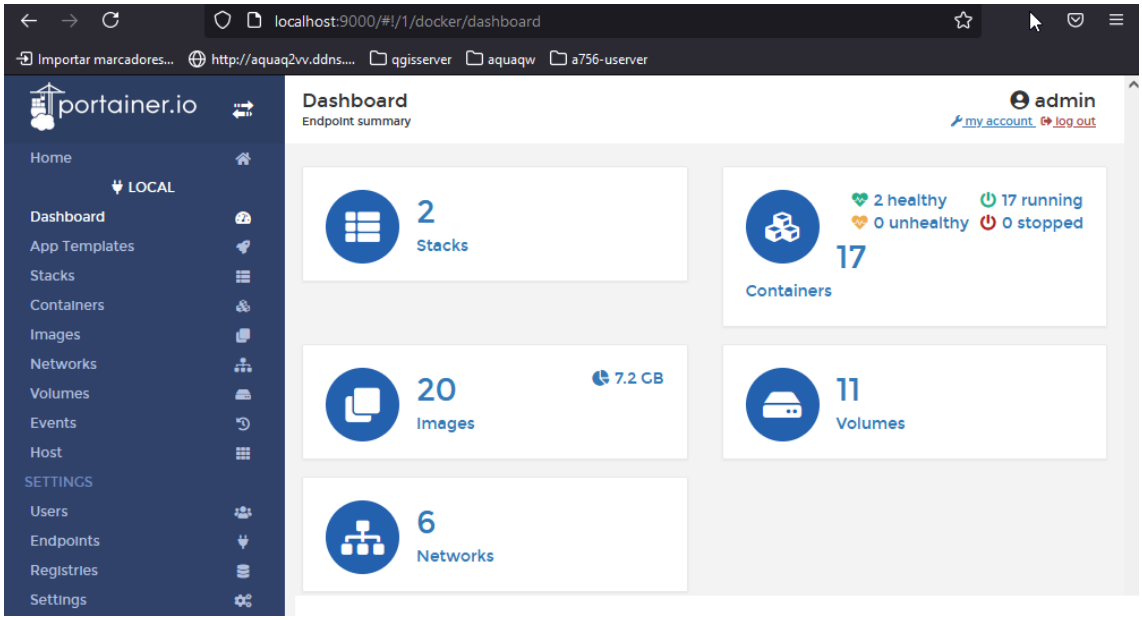


Figura 5.2: Interface *Portainer*. Acedendo a http://ipdoservidor:9000 é oferecida uma visão geral do conteúdo dos componentes da tecnologia de *contentores* utilizada.

Tabela 5.1: Pilha de *contentores* - Portainer. Constituída por 3 *contentores*. Rede portainer_default, IPV4 Subnet - 172.18.0.0/16.

nome	imagem	portos
portainer_db_1	jc21/mariadb-aria:latest	-
portainer_nginxproxymanager_1	jc21/nginx-proxy-manager:latest	33081:81 33443:443 33080:80
portainer_portainer_1	portainer/portainer-ce:latest	9000:9000 8000:8000

últimas versões, do *Nginx Proxy Manager* e do SGBD *MariaDB*.

Pilha de *Contentores* - AquaQ2

Na Tabela 5.2, observar-se os 13 *contentores* e respetivos *portos*, que compõem a pilha de *contentores AquaQ2*. As principais funções de cada um, foram abordadas na Secção 4.2. No ficheiro *docker-compose.yml*, (Apêndice VIII, Secção VIII) pode-se consultar as configurações de criação dos *contentores* da pilha *AquaQ2* e no ficheiro *Dockerfile*, (Apêndice VIII, Secção VIII), pode-se consultar as configurações de criação do *contentor qgis-server*.

Os *contentores* nº 1, 2, 3, 4, 7 e 8, correspondem aos servidores e serviços, necessários para colocar em funcionamento a pilha *Chirpstack*. O *contentor* nº 9, corresponde ao serviço PostGIS, necessário para armazenar os dados produzidos pelos módulos de aquisição de dados. Os *contentores* nº 5 e 10, correspondem aos servidores *HTTP* e *QgisServer*,

Tabela 5.2: Pilha de *contentores* - AquaQ2. Constituída por 13 contentores. Rede aquaq2_default, IPV4 Subnet - 172.19.0.0/16, rede aquaq2_node-red-net, IPV4 Subnet - 172.20.0.0/16.

nº	nome	IP	portos
1	aquaq2_chirpstack-network-server_1	172.19.0.12	-
2	aquaq2_chirpstack-application-server_1	172.19.0.4	38080:8080
3	aquaq2_chirpstack-gateway-bridge_1	172.19.0.6	1700:1700
4	aquaq2_mosquitto_1	172.19.0.3	31883:1883
5	aquaq2_nginx_1	172.19.0.9	32080:80
6	aquaq2_adminer_1	172.19.0.5	35081:8080
7	aquaq2_redis_1	172.19.0.11	-
8	aquaq2_postgresql_1	172.19.0.7	36432:5432
9	aquaq2_postgis_1	172.19.0.13	35432:5432
10	aquaq2_qgis-server_1	172.19.0.8	-
11	aquaq2_node-red_1	172.20.0.2	31880:1880
12	aquaq2_meinheld_1	172.19.0.2	39080:80
13	aquaq2_data_insert	172.17.0.2	-

responsáveis por servir mapas com dados georreferenciados das estações de aquisição de dados. Os *contentores* nº 11, 12 e 13 são responsáveis por facilitarem o armazenamento e visualização de dados.

Os *portos* de Internet da pilha *AquaQ2*, encontram-se na gama 30000-40000, com a exceção do sistema de gestão dos *contentores*. Quando expostos são utilizados *portos* distintos para o interior e exterior dos *contentores*. Na Tabela 5.3 estão descriminadas as redes *Docker* utilizadas pelo conjunto de *contentores*.

As redes *bridge*, *host* e *none* são criadas por defeito. As redes «*portainer_default aquaq2_default*» e «*aquaq2_node-red-net*» estão associadas ao *driver bridge*, existente por defeito no *Docker* e que permite a utilização do *DNS* interno do *Docker*. Este associa automaticamente todos os nomes de *contentores* dessa rede para os respetivos IPs da rede IP correspondente (Tabela 5.2).

Rede	
Nome	Driver
aquaq2_default	bridge
aquaq2_node-red-net	bridge
portainer_default	bridge
bridge	bridge
host	host
none	none

Tabela 5.3: Lista de redes *Docker*. As redes *bridge*, *host* e *none* são criadas por defeito.

Na Tabela 5.4, estão descriminados os *volumes* criados para o conjunto de *contento-*

res. Estes volumes, permitem resolver questões de persistência de dados e garantem a salvaguarda destes, mesmo removendo um *contentor*.

Volumes
aquaq2_redisdata
eclipse-mosquitto
meinheld-app
nginx-html
node-red-data
portainer_portainer_data
postgisdata
postgresqldata
qgis-data

Tabela 5.4: Lista de volumes *Docker*. No Apêndice VIII, Secção VIII e Secção VIII é apresentado o código/comandos utilizados para criação do sistema de ficheiros (pastas permanentes dos volumes) e respetivos volumes.

Servidores/Serviços	**Máquina Local**		**Contentor**
	Docker/Volume caminho: /var/lib/docker/volumes/	**Máquina local caminho: /var/local/aquaq2/**	**caminho**
Mosquitto	eclipse-mosquitto/_data	eclipse-mosquitto/data	/mosquitto/data
Flask	meinheld-app/_data	meinheld/app	/app
NGINX	nginx-html/_data	nginx/html	/usr/share/nginx/html
Node-RED	node-red-data/_data	node-red/data	/data
Postgresql	postgresqldata/_data	postgresql/data	/var/lib/postgresql/data
Postgis	postgisdata/_data	postgis/data	/var/lib/postgresql/data
Redis	aquaq2_redisdata/_data		/data
Qgis	qgis-data/_data	qgis/data	/data
Portainer	portainer_data/_data		/data portainer/portainer-ce
network-server	.../aquaq2/configuration/chirpstack-network-server		/etc/chirpstack-network-server
application-server	.../aquaq2/configuration/chirpstack-application-server		/etc/chirpstack-application-server
gateway-bridge	.../aquaq2/configuration/chirpstack-gateway-bridge		/etc/chirpstack-gateway-bridge
geolocation-server	.../aquaq2/configuration/chirpstack-geolocation-server		/etc/chirpstack-geolocation-server

Tabela 5.5: Sistema de ficheiros e respetivos volumes, na máquina local e nos contentores. Os volumes garantem a salvaguarda dos mesmos, mesmo removendo o respetivo *contentor*.

Instalação e Configuração

As aplicações e serviços são lançados em ambiente *Linux* através da *tecnologia de* contentores Docker.

No Apêndice VIII são apresentados os ficheiros com as configurações de instalação do sistema de controlo de *contentores* e procedimento de instalação da pilha de aplicações e serviços *Docker*[8]

Os ficheiros com a descrição dos procedimentos e configurações podem ser obtido através do repositório *Git «git clone gitolite@sepsi.ipbeja.pt:aquaq2»*. O ficheiro «readme.md»,

[8] A versão inicial destas instruções, foram disponibilizado pelo Professor José Jasnau Caeiro no âmbito do projeto AquaQ2

oferece todas as instruções necessárias à instalação. A sua execução gera a estrutura apresentada na Figura 5.3, respeitando o sistema de ficheiros e respetivos volumes, apresentados na Tabela 5.5.

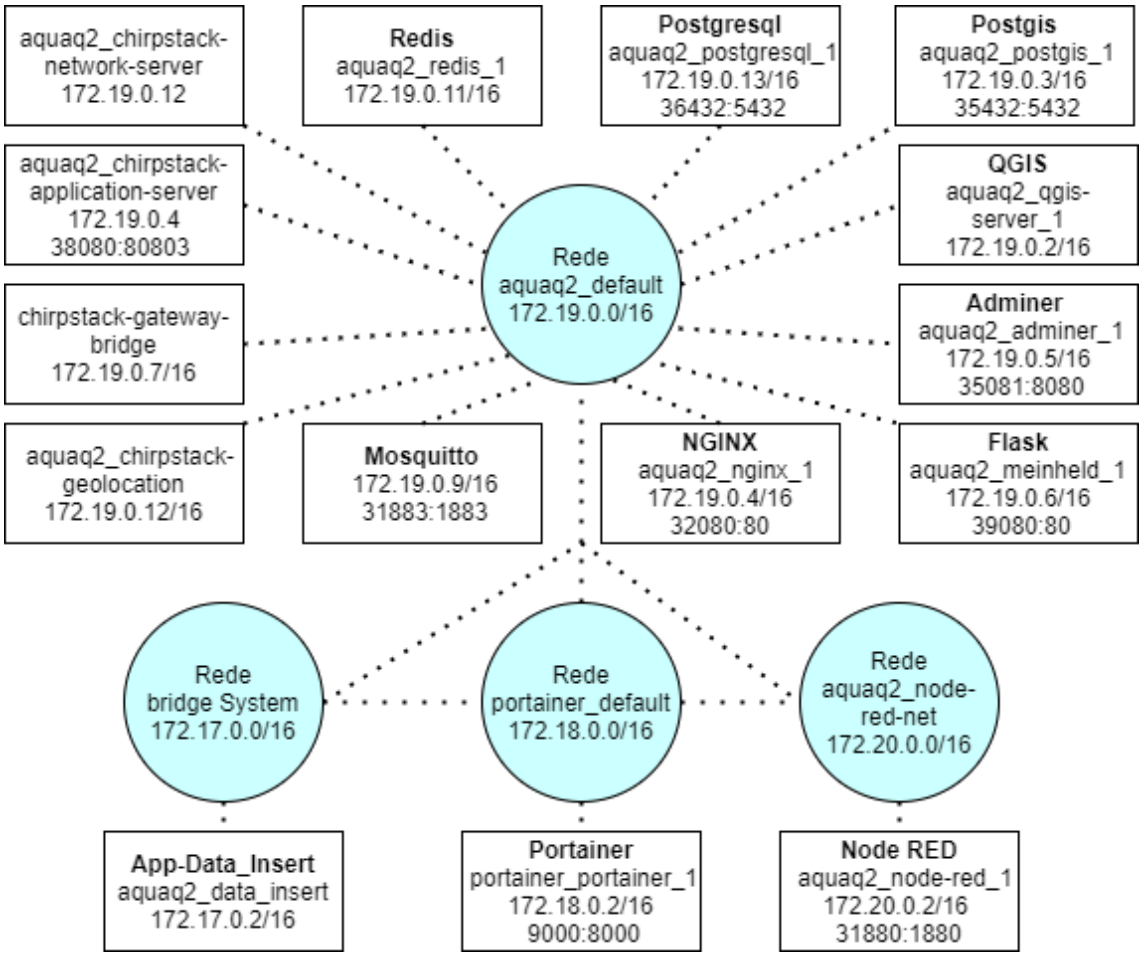


Figura 5.3: Rede de *contentores*. Como resultado da instalação e configuração, do sistema de controlo de *contentores* e procedimento de instalação do *stack* para *Docker*, obtemos um conjunto de *contentores* com as características aqui ilustradas.

5.2 Pilha de Comunicações LoRaWAN

No caso da implementação de redes *LoRaWAN*, existem duas opções *open-source*, o *ChirpStack* e *TTN* (*The Things Network*).

The Things Network (TTN), facilita soluções baseadas numa *cloud* pública e na partilha de *gateways*. Abstrai a implementação do servidor de rede *LoRaWAN* e o mecanismo de transporte de dados. O *ChirpStack* é uma plataforma computacional, que também implementa o *LoRaWAN*, permitindo aos utilizadores criarem o seu próprio servidor de rede. Criada em 2015, a *Lora Alliance*, uma organização sem fins lucrativos, dedica-se a

garantir a interoperabilidade de todos os produtos e tecnologias *LoRaWAN*.

No âmbito da dissertação foi utilizado o sistema *ChirpStack*, por ser uma solução pronta a utilizar, por garantir a utilização de uma rede local e privada, e porque inclui um interface *web* amigável para a gestão de dispositivos e APIs para a integração. A arquitetura modular do *ChirpStack* permite a integração com as infraestruturas existentes. Todos os componentes têm licença MIT [72] e podem ser utilizados para fins comerciais [13].

A arquitetura *LoRaWAN* pode ser dividida em duas estruturas distintas. A primeira é dedicada à camada física do modelo *OSI* utilizando a tecnologia LoRa; a segunda é descrita pelo padrão *open-source LoRaWAN* dedicada à gestão de acesso ao meio, semelhante à camada de enlace ou de ligação do modelo *OSI*, que detalha a estrutura necessária para a gestão de recursos da rede. O *ChirpStack* implementa um sistema de transmissão de dados que utiliza os componentes de rede desenvolvidos pelo *LoRa Server Project* para implementar redes *LoRaWAN* [13], baseadas na tecnologia LoRa (Figura XIII.2).

O diagrama apresentado na Figura XIII.2 é original da empresa *Semtech*, constituído pelos componentes já abordados no Capítulo 4. Esta Secção é um complemento à abordagem e desenvolvimento deste assunto na Subsecção 4.2.

Os dispositivos *LoRaWAN*, ilustrados na Figura 4.2 são dispositivos que podem transmitir os dados através de uma ou mais *gateways* LoRa. Estes dispositivos são o sistema formado pelos sensores e MCU/SBC que possuem capacidade de comunicação através desta tecnologia.

A *Gateway LoRa* (Secção VII.5) executa o sistema operativo *open-source*, baseado em Linux, *LORIX OS*. Esta *gateway permite* escutar vários canais em simultâneo e encaminha os dados recebidos dos dispositivos, pelo *chipset* de rádio *LoRa* através de um link IP/UDP (programa *Packet-forwarder*) para o serviço *LoRa Gateway Bridge*.

O serviço *LoRa Gateway Bridge* está posicionado «entre» o *Packet-forwarder* e o *MQTT Broker*, transforma o formato produzido pelo *Packet Forwarder* num formato de dados utilizado pelos componentes *ChirpStack* (JSON ou Protobuf[9]). As instruções de configuração encontram-se no ficheiro apresentado no Apêndice VIII.

O *MQTT* (Apêndice XII) é o protocolo *Internet* utilizado na fronteira entre o serviço *LoRa Gateway Bridge* e o *LoRa Network Server*. Funciona no modelo assíncrono, cliente/servidor (publicação/subscrição). O *MQTT Broker* publica todos os dados *JSON*, que recebe do *LoRa Gateway Bridge*. É necessário subscrever os respetivos tópicos para receber dados dos dispositivos. Todos os componentes *ChirpStack* fornecem *APIs gRPC* e/ou

[9]ProtoBuf (sigla de Protocol Buffers) é um mecanismo criado e utilizado pela *Google* para serializar dados estruturados. É independente de linguagem ou plataforma.

REST para integração com serviços externos. Por defeito, todos os dados das aplicações são publicados através do *MQTT Broker*. O *MQTT Broker* é executado num *contentor* (Figura 5.3) e implementa a versão 3.1.1 do protocolo *MQTT* [26].

O servidor *MQTT Broker*, por defeito, requer autenticação dos clientes durante o estabelecimento da ligação. Suporta tanto a autenticação baseada no nome de utilizador/-password, como a autenticação baseada no certificado *TLS* e utiliza estes mecanismos na seguinte ordem: certificado e utilizador/palavra-passe. Na autenticação «utilizador/-password», quando um dispositivo necessita autenticar-se com este mecanismo, tem de fornecer um nome de utilizador e palavra-passe. Para ligação ao *MQTT Broker*, este verifica as credenciais fornecidas pelo clientes. A configuração do *MQTT Broker* é descrita num ficheiro apresentado no Apêndice VIII).

Os dados são recebidos pelo *LoRa Network Server* através do *MQTT Broker*. O *LoRa Network Server* é responsável pela gestão da rede. Deteta ativações de dispositivos na rede e é capaz de lidar com solicitações de ingresso quando os dispositivos o solicitam. Os módulos de qualidade da água e a estação meteorológica são registados no *Lora Application Server* como dispositivos pertencentes à rede *LoRaWAN*. Existindo mais do que uma *gateway* a enviar dados, o *LoRa Network Server* elimina os duplicados e encaminha através do protocolo *gRPC* a carga útil para o *LoRa Application Server*. Quando o *LoRa Application Server* envia dados de resposta para um dispositivo, o *LoRa Network Server* manterá esses itens em fila de espera, até ser possível enviá-los. A configuração do *LoRa Network Server* é descrita num ficheiro de configuração (Secção VIII), onde podem ser definidos:

- os parâmetros de ligação à base de dados no *PostgreSQL*;
- os parâmetros de ligação à base de dados *Redis*;
- os parâmetros de ligação ao *MQTT Broker*;
- os tópicos *MQTT* utilizados para comunicar com o *LoRa* Gateway Bridge;
- os parâmetros da rede, por exemplo, o identificador de rede ou os canais de rádio;
- os parâmetros que configuram a comunicação com os dispositivos das várias classes;
- os parâmetros e intervalos de tempo entre comunicações da(s) *gateway(s)*;
- o tipo de estatísticas a recolher e a sua granularidade[10], entre outros.

Algumas destas configurações podem ser realizadas a partir do interface *web* do *LoRa Application Server*.

[10] Grau de detalhe.

O *LoRa Application Server* é o servidor de aplicações *LoRaWAN*, compatível com o *LoRa Network Server*. Fornece um interface *Web* e *APIs*, para gerir utilizadores, organizações, aplicações, *gateways* e dispositivos. Os dados recebidos são encaminhados para as integrações configuradas. As várias *APIs* permitem integrar com outros sistemas, entre as quais, servidores *MQTT*, *gRPC*, *REST* e serviços de *cloud computing* (por ex.: Google Cloud Platform). É possível integrar com sistemas de gestão de bases de dados (por ex.: PostgreSQL). Neste caso é possível poderá armazenar todos os eventos relacionados com os dispositivos numa base de dados PostgreSQL, para que possam ser consultados por outras aplicações ou para que possam ser visualizados utilizando, por exemplo, a aplicação *Grafana* através do SGBD *PostgreSQL*. O *LoRa Application Server* também depende de *software* adicional para funcionar: o sistema de gestão de bases de dados *PostgreSQL* para armazenamento de dados permanentes e *Redis* para armazenamento de dados temporários. Na Secção VIII é descrita a configuração do *LoRa Application Server*.

O interface *web* do *LoRa Application Server* pode ser visto na Figura XIII.3. Neste é possível configurar a(s) *gateway(s)*, dispositivos e aplicações. É possível, também, definir parâmetros dos dispositivos e *gateways*, e a chave identificadora da aplicação, nomeadamente o EUI[11].

Através do interface *web* configuram-se os perfis de dispositivos, em que cada um, define parâmetros de configuração de um tipo de dispositivo no *LoRa Network Server*. A classe do dispositivo (A, B, C), o método de associação à rede (OTAA[12] ou ABP[13]) (Apêndice XI), os intervalos de transmissão e receção (RX1, RX2 *window*), a potência máxima isotrópica radiada equivalente[14] (Maximum EIRP), são exemplo entre outros. Dispositivos que são configurados de forma semelhante podem ser associados a um perfil de dispositivo. Caso seja necessário alterar uma configuração específica, individualmente por dispositivo, pode-se sobrepor as configurações do perfil na página de configuração do dispositivo. À semelhança dos dispositivos é possível configurar *gateways* e perfis de *gateway*. As *gateways* que são configuradas de forma semelhante podem também ser associadas a perfis de *gateway*.

Adicionalmente, o interface *web* exibe estatísticas sobre as *frames* enviadas e recebidas pela(s) *gateway(s)*, as coordenadas e altitude das *gateways*, e as suas localizações no mapa (Figura XIII.4).

A uma instância de *LoRa Application Server* podem-se ligar vários *LoRa Network Servers*. Os parâmetros de ligação com os *LoRa Network Servers*, por exemplo o IP, porto e certificados utilizados para *TLS*, podem ser configurados a partir do interface *web* do *LoRa Application Server*.

[11] Identificador exclusivo atribuído pelo fabricante.
[12] Ativação através do ar.
[13] Ativação por personalização.
[14] Produto da potência fornecida à antena pelo seu ganho em relação a uma antena isotrópica, numa dada direção (ganho isotrópico ou absoluto).

É possível monitorizar, eventos e mensagens, encaminhadas de/para os dispositivos, no interface *web* do *LoRa Application Server* (Figura XIII.5). O registo de eventos pode ser visualizado por dispositivo, assim como o registo de mensagens também por dispositivo e por *gateway*. Todas as mensagens encaminhadas por uma determinada *gateway* podem ser monitorizadas.

As «*End-application*» recebem os dados dos dispositivos através das várias integrações. Neste protótipo integramos com o *MQTT Broker*.

5.3 Sistema de Gestão de Base de Dados

Os sistemas de gestão de base de dados *PostgreSQL* e *Redis* são utilizados na gestão e armazenamento de dados do sistema *LoRa Chirptack*. É utilizado o SGBD *PostgreSQL*, com as extensões *PostGIS*, para gerir e armazenar os dados gerados pelos módulos de aquisição de dados. O *SGBD MariaDB* é utilizado para armazenar os dados gerados pelo gestor de *contentores Portainer*.

PostgreSQL

O *PostgreSQL* é o *SGBD* escolhido para este projeto, com o fim de armazenar as leituras dos módulos de aquisição de dados (subsecção 4.2). É um *SGBD* relacional com mecanismos *object oriented* e *open-source* que utiliza e estende a linguagem *SQL*. É compatível com todos os principais sistemas operativos e com *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade). Possui inúmeros complementos, nomeadamente o extensor de base de dados geoespacial *PostGIS*(subsecção 5.3). O *PostgreSQL* tornou-se o *SGBD* relacional, de código aberto preferido de muitas pessoas e organizações [92]. É possível obter uma instância do *PostgreSQL* de diversas formas, a mais tradicional é a instalação de raíz [91] em alternativa recorrendo à tecnologia de contentores utilizando o *Docker* [89] ou o *Vagrant*[15] [90]. Por último, este *SGBD* pode ser obtido como um serviço através de um fornecedor, por exemplo o *AWS* ou *Azure*. A forma escolhida terá em consideração a sua finalidade.

PostGIS

O complemento do *PostgreSQL* para suporte a sistemas de informação geográfica é constituído pelas extensões *PostGIS* e pelo Servidor *QGIS*. O *PostGIS* é um extensor de base de dados espacial para o *SGBD PostgreSQL*. Adiciona suporte para objetos geográficos, permitindo a execução de consultas de localização na linguagem *SQL* [88]. O sistema *GIS*

[15] *Software open-source* para criar e gerir ambientes de desenvolvimento virtuais.

utiliza os dados dos sensores e os dados de informação geográfica, para melhorar a capacidade de análise de dados. As funcionalidades base do QGIS podem ser estendidas pela adição de *plugins*, os *sites* [88, 98] apresentam uma lista. O serviço *QGIS Cloud* fornece um *plugin* que possibilita a integração do *QGIS* com este serviço. No *site QGIS Cloud* [97], encontra-se descrita de forma detalhada o procedimento de integração.

O ambiente de trabalho da aplicação *QGIS* (Figura IX.3) possibilita a interação com *PostgreSQL* com as extensões *PostGIS*. Para disponibilizar a informação geográfica na *In-*

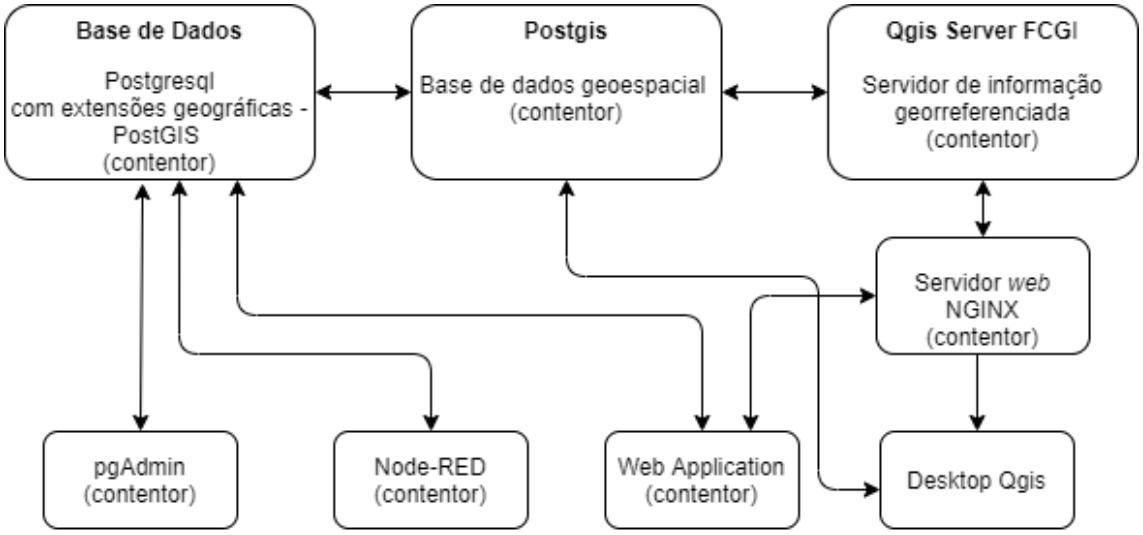


Figura 5.4: Sistema de gestão de bases de dados e de informação geográfica. Os dados são armazenados numa base de dados com extensões geográficas. São acedidos para diferentes fins através de diferentes plataformas e aplicações.

ternet é comum recorrer ao QGIS-Server. O *QGIS-Server* é uma implementação *WMS*[16] 1.3 e *WFS*[17] 1.0.0 de código aberto, que implementa características cartográficas avançadas para a cartografia temática [18]. O servidor *QGIS* é uma aplicação *FastCGI/CGI* (*Common Gateway Interface*) que pode ler dados de *PostGIS* e servir imagens ou dados, de acordo com os padrões do *Open Geospatial Consortium* (OGC[19]). Representa mapas *QGIS* através de um serviço *HTTP*, com suporte para CGI [20] (*Common Gateway Interface*). Neste projeto utilizamos o servidor *HTTP NGINX* com o módulo FastCGI (*Common Fast Gateway Interface*). Acede-se à informação geográfica armazenada no servidor *QGIS* através do servidor *HTTP NGINX* com a aplicação *desktop QGIS* ou com pedidos *HTTP* (Figura 5.4).

[16]Web Map Service - A especificação WMS é uma norma que permite disponibilizar informação geográfica sob a forma de imagens georreferenciadas através da *Internet*, não permite o *download*.

[17]Web Feature Service - é uma especificação de um formato que disponibiliza dados geográficos em formato vetorial, permite o *download*.

[18]É a representação geoespacial de temas diversos. Existem ainda, as categorias específicas de cartografia , topográfica e geográfica.

[19]Comunidade mundial empenhada em melhorar o acesso à informação geoespacial.

[20]É um padrão para o interface de aplicações externas ou *gateways*.

Base de Dados do Sistema LoRa ChirpStack

O *ChirpStack*, nomeadamente o *LoRa Network Server* e o *LoRa Application Server*, de acordo com a Secção 5.2, utilizam o *SGBD PostgreSQL* para armazenamento de dados permanentes (exemplo ilustrado nas Figuras XIII.6 e XIII.7) e o *SGBD Redis* para armazenamento de dados temporários (Figura XIII.8).

O *SGBD Redis* é uma implementação chave-valor. Atribui valores às chaves, para facilitar o acesso e o armazenamento desses valores, que são sempre encontrados através das suas chaves (Figura XIII.9). O *SGBD Redis* mantém os pares de valores-chave em memória, tornando o acesso mais rápido. O *SGBD MariaDB* é utilizado pelo gestor de contentores *Portainer*, para armazenar a informação gerada.

Armazenamento de Dados dos Módulos de Aquisição de Dados

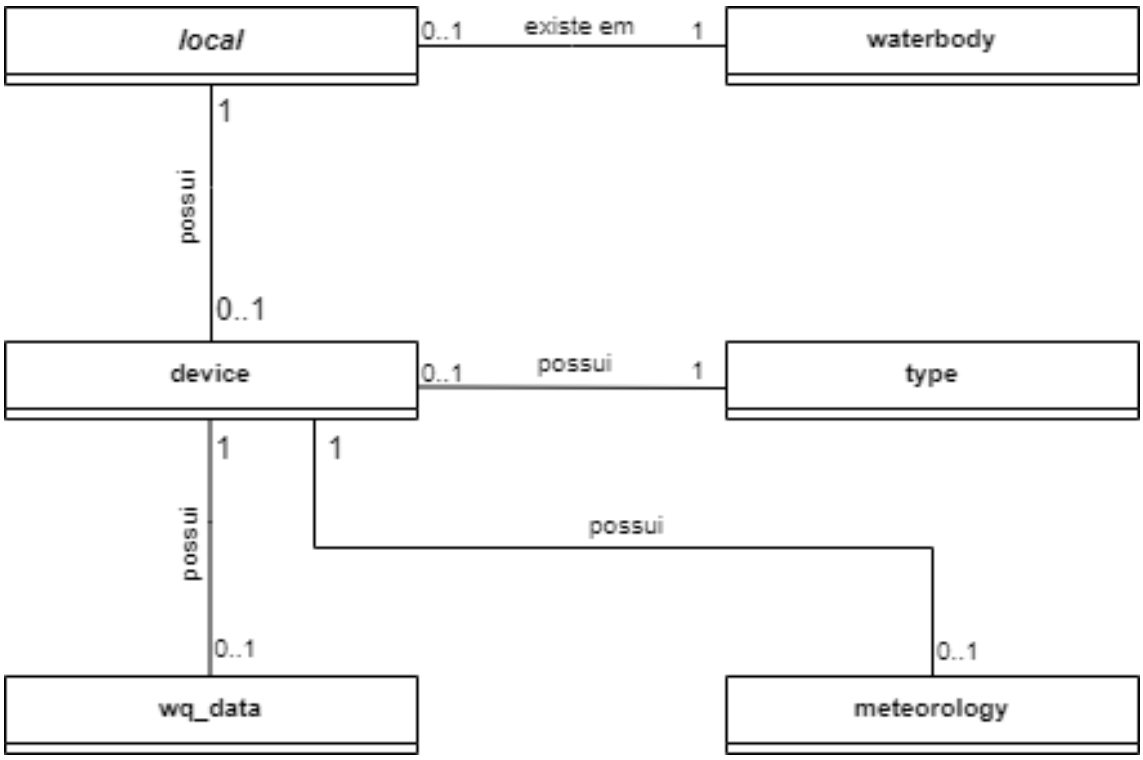


Figura 5.5: Modelo de dados relacional utilizado para armazenar os dados dos módulos de aquisição de dados. A tabela *«waterbody»* armazena a identificação dos corpos de água escolhidos para a aquisição de dados. A tabela *«local»* armazena a localização das estações de aquisição de dados. A tabela *«type»* armazena os tipos de dispositivos. A tabela *«device»* armazena os dispositivos sensores. A tabela *«wq_data»* armazena as medições da qualidade da água, referente ao módulo de qualidade da água. A tabela *«meteorology»* armazena as medições da Estação Meteorológica.

A base de dados *«aquaq2»* está criada e parametrizada, na instância *contentorizada* do *PostGIS*. No Apêndice IX são apresentadas as instruções e código para criar a base de

dados.

Quando é executada a aplicação *Data Storing App* (Secção 5.5), cria-se, em caso de inexistência, o modelo de dados (Figura 5.5). São criadas seis tabelas/relações, conforme Figura IX.1 (Apêndice IX).

A tabela *«waterbody»* armazena a identificação dos corpos de água escolhidos para a aquisição de dados. A Tabela IX.1 ilustra a estrutura da tabela, assim como uma amostra dos dados que esta armazena.

A tabela *«local»* armazena a localização das estações de aquisição de dados, armazena o nome do corpo de água, o nome do local/estação, as coordenadas de localização, e o estado do «local», ativo ou não ativo. O campo que representa as coordenadas de localização é a chave que permite a georeferenciação, das estações de recolha de dados (Figura 5.11). Todo o *«local»* pertence a um corpo de água. Cada *«local»*, possui o *«id»* de um corpo de água. A cada registo desta tabela, corresponde a uma «localização» onde existe ou já existiram dispositivos associados. A Tabela IX.2 ilustra, a estrutura da tabela *«local»*, assim como uma amostra dos dados que esta armazena.

A tabela *«type»* armazena os tipos de dispositivos, por exemplo: módulos de aquisição de dados de qualidade da água ou módulos meteorológicos.

A tabela *«device»*, armazena os dispositivos sensores (Tabela IX.4). A cada dispositivo (*«device»*) é atribuído um «tipo» e um «local» .

A tabela *«wq_ data»* armazena as medições da qualidade da água, referente ao módulo de qualidade da água, é ilustrado um exemplo na Tabela IX.5. Cada registo na tabela *«wq_ data»* é atribuído a um dispositivo de qualidade da água.

A tabela *«meteorology»*, armazena as medições da Estação Meteorológica, exemplo: Tabela IX.6.

Ferramentas para Interagir com a Base de Dados

A ferramenta gráfica *opensource pgAdmin* é utilizada para a gestão do *SGBD PostgreSQL*, com extensões *PostGIS* (Figura 5.4). É escrita na linguagem de programação *Python* e *jQuery*[21], e permite administrar instâncias *PostgreSQL* [84] e executar comandos em *SQL*.

A ferramenta *PgAdmin*, Figura IX.2 disponibiliza um *interface* para a administração do servidor *PostgreSQL* [84].

Pode-se utilizar a ferramenta de linha de comandos *psql* ou a aplicação *QGIS* (Figura IX.3) para interagir com o *SGBD PostegreSQL*.

[21] Biblioteca *JavaScript* que facilita a interação com o HTML simplificando os *scripts* interpretados no navegador cliente.

5.4 Ambiente de Desenvolvimento de Software

É frequente que os protótipos ou sistemas de *Internet das Coisas* tenham que ser desenvolvidos rapidamente.

É utilizada uma plataforma de *backend* e *frontend*, que permite observar e interagir com sistemas. A escolha de ambientes de desenvolvimento, de aplicações e de linguagens de programação para o fazer é muitas vezes ditada por outros constrangimentos: o custo, a disponibilidade, a formação e o investimento prévio, ou mesmo a simples preferência [36].

Sistemas Operativos

O sistema operativo do servidor, dedicado a gerir todos os servidores e serviços, através da *tecnologia de contentores Docker* é baseado na distribuição *Linux Ubuntu Server 20.04 LTS* [22] *- (Focal Fossa)* [35].

O sistema operativo reside numa máquina virtual *KVM* (*Kernel-based Virtual Machine* - infraestrutura de virtualização) [54].

O sistema está acessível a partir de qualquer sistema operativo através dos protocolos *ssh* e *http*, permitindo a realização de testes e visualização de resultados.

O sistema *LORIX One* (Figura VII.5) executa uma versão do sistema operativo *LORIX OS* [122]. Este gere a receção dos dados recebidos dos módulos de aquisição de dados, através do protocolo *LoRaWAN* e a sua transmissão, através um link IP/UDP (programa *Packet-forwarder*) para o serviço *LoRa Gateway Bridge*.

A Linguagem de Programação

A linguagem de programação *open-source*, *Python*, foi adotada para o desenvolvimento das aplicações. Tem suporte de uma grande comunidade *online* e apresenta-se, em diversos fóruns, como a linguagem de programação mais popular da atualidade [123, 124]. O *microframework Flask*[125], de desenvolvimento *web* é pardigmática como exemplo da utilização da linguagem de programação *Python*.

O *SQLAlchemy* é um *framework open source* com licença MIT, de mapeamento objeto-relacional SQL (ORM-*Object Relational Mapper*) que permite o uso programático de bases de dados.

As operações *CRUD* são executadas na base de dados *PostgreSQL* (com extensões *PostGIS*) através do *SQLAlchemy*. Não é o *SQLAlchemy* que faz a ligação à base de dados, constitui uma camada de abstração que mapeia as tabelas em objetos. Utiliza a classe *Dialect*[23], o pacote *psycopg2* como padrão DB-API. É necessário o *driver* de base de

[22] *Long Term Service*

[23] É o sistema que o *SQLAlchemy* utiliza para comunicar com vários tipos de implementações e bases de dados DB-API (API-interface de programação de aplicações)

dados, *psycopg*. Este permite a comunicação entre o *SQLAlchemy* e o SGBD *PostgreSQL*.

O *microframework Flask* foi utilizado para a visualização de dados. Este *microframework* escrito na linguagem de programação *Python*, facilita o desenvolvimento de aplicações *web*, adota o *MVC* (*Model View Controller*) como padrão de *software*.

5.5 Aplicações

A partir da estrutura de *hardware* baseada em *IoT*, que recolhe e transmite um conjunto de parâmetros físicos ou químicos da água, e do ambiente, uma aplicação processa e armazena os dados (Figura 5.10), num *SGBD*, e outra aplicação gera a visualização e análise dos dados em «tempo real» (Figura 5.12).

O módulo de aquisição de dados da qualidade da água usa um *MCU* que executa *software* que permite medir os dados recebidos dos nós sensores e enviar as leituras periodicamente (Figura 5.6).

O código disponibilizado pelos fabricantes dos sensores foi adaptado ao projeto, de forma a concretizar-se o instrumento proposto.

O módulo que realiza a aquisição de dados da estação meteorológica (Figura VII.4) é gerido por um *SBC* que executa *software* com o objetivo de enviar as leituras periodicamente.

A solução *IoT LoRaWAN* foi instalada para realização de testes na barragem da Lage, localizada no Alentejo, perto de Pias, Portugal (Figura 5.11), a aproximadamente 160m de altitude. Os dados são adquiridos e transmitidos como representado na Figura 5.6.

Os módulos de aquisição de dados, comunicam periodicamente as leituras realizadas, através de uma *gateway*. A *gateway* liga-se à *Internet*, através da rede móvel de telecomunicações, para comunicar com os servidores de suporte à rede LoRa.

A rede *LoRaWAN*, através dos servidores e serviços *ChirpStack*, alimenta a camada superior com os dados recebidos dos *end devices*. Esta é constituída pelas aplicações, que armazenam, processam e que permitem visualizar os dados recebidos, através do *MQTT Broker*.

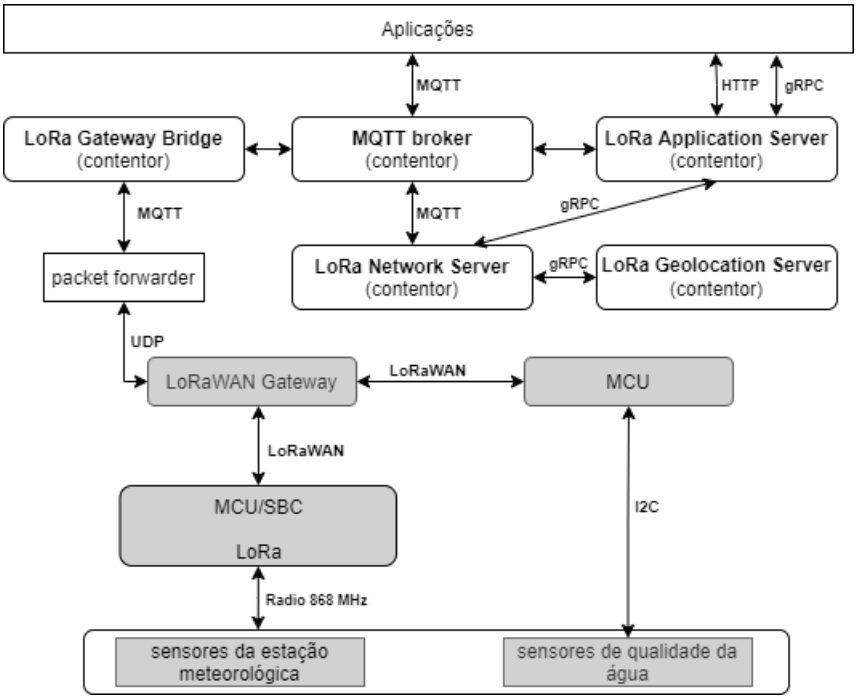


Figura 5.6: Estrutura de comunicação de dados utilizando as tecnologias *LoRa* e *4G*. Serviços e aplicações são executados recorrendo à *tecnologias de contentores*. Os dados recolhidos pelos dispositivos são enviados para o *MQTT Broker*, através da tecnologia *LoRa*. Os dados recebidos pelo *MQTT Broker* através da tecnologia *4G* são publicados pelo *MQTT Broker* e subscritos pelas aplicações para análise, processamento e armazenamento.

Testes à Aquisição de Dados

O servidor *open-source Node-RED* [78] baseado em *Node.js* (um dos *contentores* da pilha *AquaQ2*, Tabela 5.2) foi utilizado para realizar os primeiros testes e verificar se a arquitetura implementada cumpria os requisitos técnicos e funcionais. Estão implementados diversos fluxos de teste, como ilustrado na Figura 5.7, que permite verificar e analisar, os dados recebidos do módulo de aquisição de dados, através do *MQTT Broker*.

A Figura 5.8 representa um conjunto de fluxos que permitem testar a receção de dados da estação meteorológica e mostrá-los graficamente em ambiente *web*, Figura 5.9.

Aplicação de Armazenamento de Dados

Os procedimentos de armazenamento de dados garantem que este é realizado sem a introdução de erros. Garantem também que todas as informações necessárias para identificar a amostra foram armazenadas, nomeadamente: o local de amostra e a identificação do dispositivo.

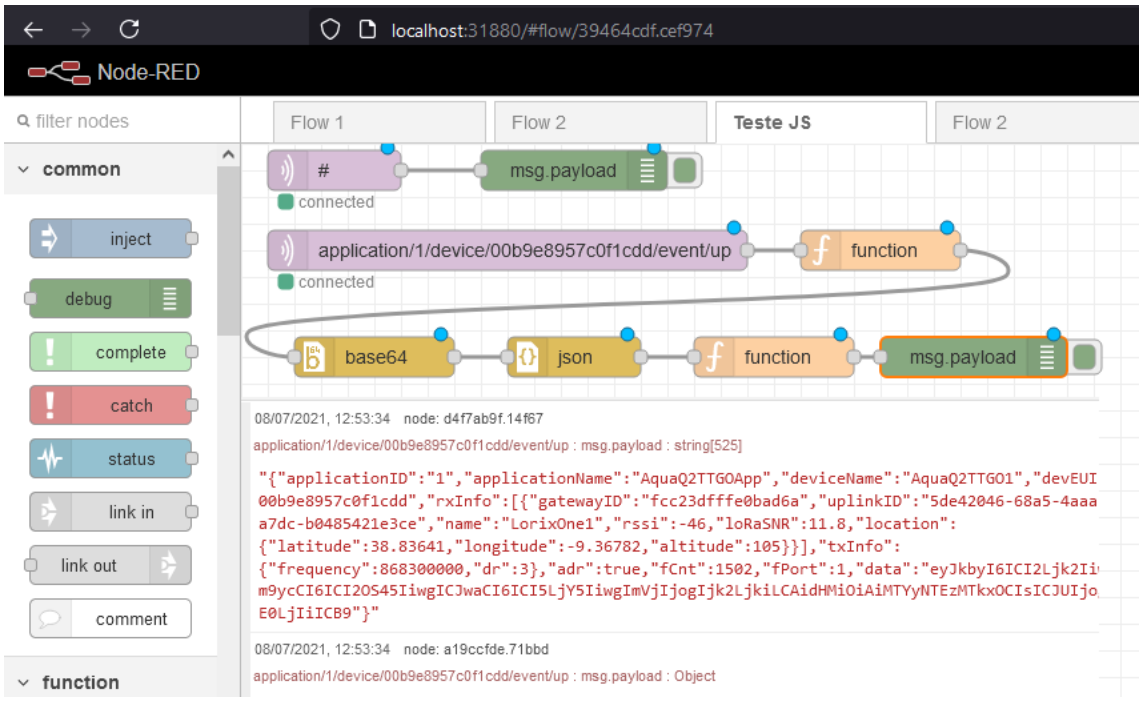


Figura 5.7: Fluxo de receção dos dados do módulo de aquisição de parâmetros de qualidade da água, através do *MQTT Broker*. São apresentadas duas subscrições de dados, a primeira subscreve de todos tópicos que o *MQTT Broker* publica; a segunda subscreve o tópico que publica apenas os dados do módulo de aquisição de dados de qualidade da água.

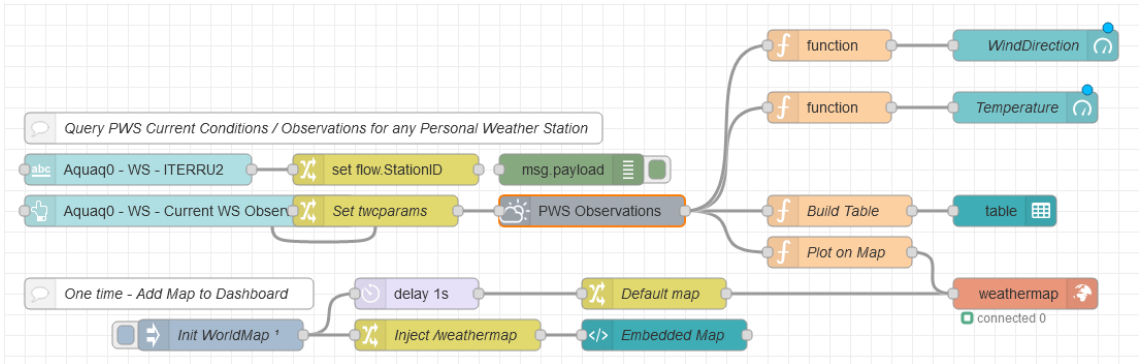


Figura 5.8: Fluxo de receção e visualização de dados, do módulo de aquisição de medições da estação meteorológica, através do *MQTT Broker*. Os fluxos apresentados, permitem receber os dados da estação meteorológica e verificar graficamente a sua leitura.

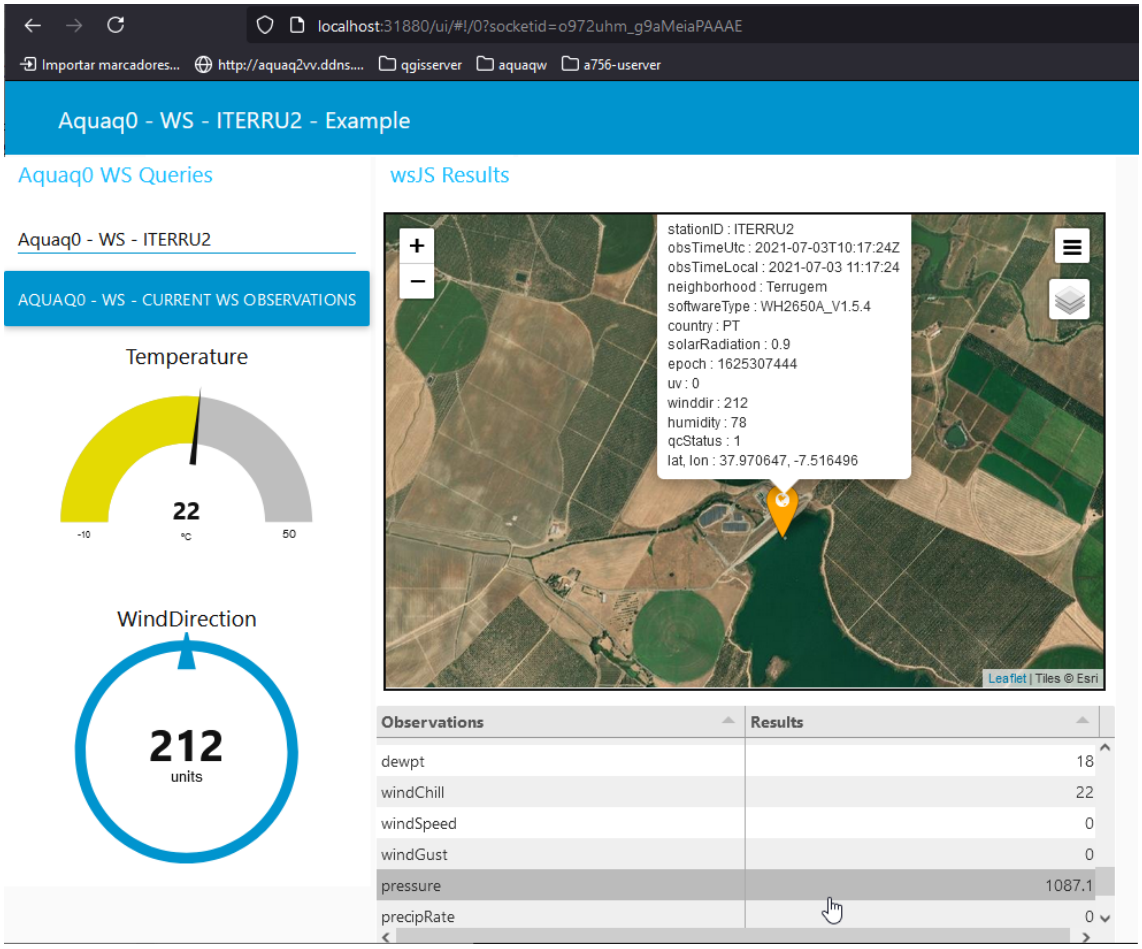


Figura 5.9: Visualização em «tempo real», de dados recebidos da estação meteorológica. Permite visualizar em dois gráficos, as medições da temperatura do ar e da direção do vento. No mapa é possível verificar a localização da estação e um conjunto de parâmetros/medições ambientais.

A aplicação *«Data Storing App»* ilustrada na Figura 4.6, subscreve do *MQTT Broker*, todos os tópicos que «transportam» os dados dos módulos de aquisição de dados, e armazena a informação, através do *SGBD* e de informação geográfica, ilustrado em maior detalhe na Figura 5.4.

A estrutura da *«Data Storing App»* é ilustrada na Figura 5.10. A aplicação é executada no *contentor «aquaq2_data_insert»* (Tabela 5.2), através do programa escrito no ficheiro *Python «main.py»* (Apêndice X). O ficheiro *«aquaq2mqtt.py»* (Apêndice X) é um ficheiro de configuração *Python*, o seu código gera o esquema de base de dados e «passa» as credencias de acesso ao *MQTT Broker* e ao *SGBD*. Evocado a partir do programa escrito no ficheiro *«aquaq2mqtt.py»*, executa o código associado ao ficheiro *«basededados.py»* (*models*) (Apêndice X), que representa as tabelas/coleções da base de dados e declara como os dados são armazenados e estruturados. Evocado também a partir do código do ficheiro *«aquaq2mqtt.py»*, no código associado ao ficheiro *«inserir_dados.py»* (*utils*) (Apêndice X)

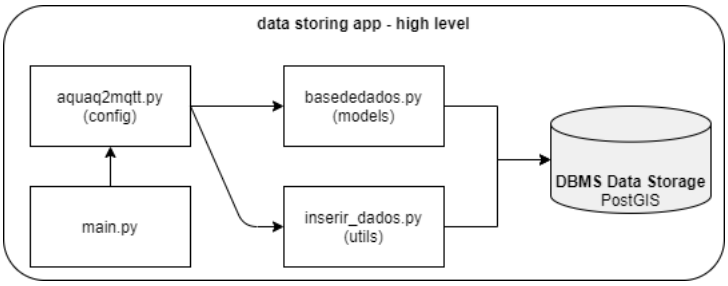


Figura 5.10: Estrutura da *Data Storing APP*.

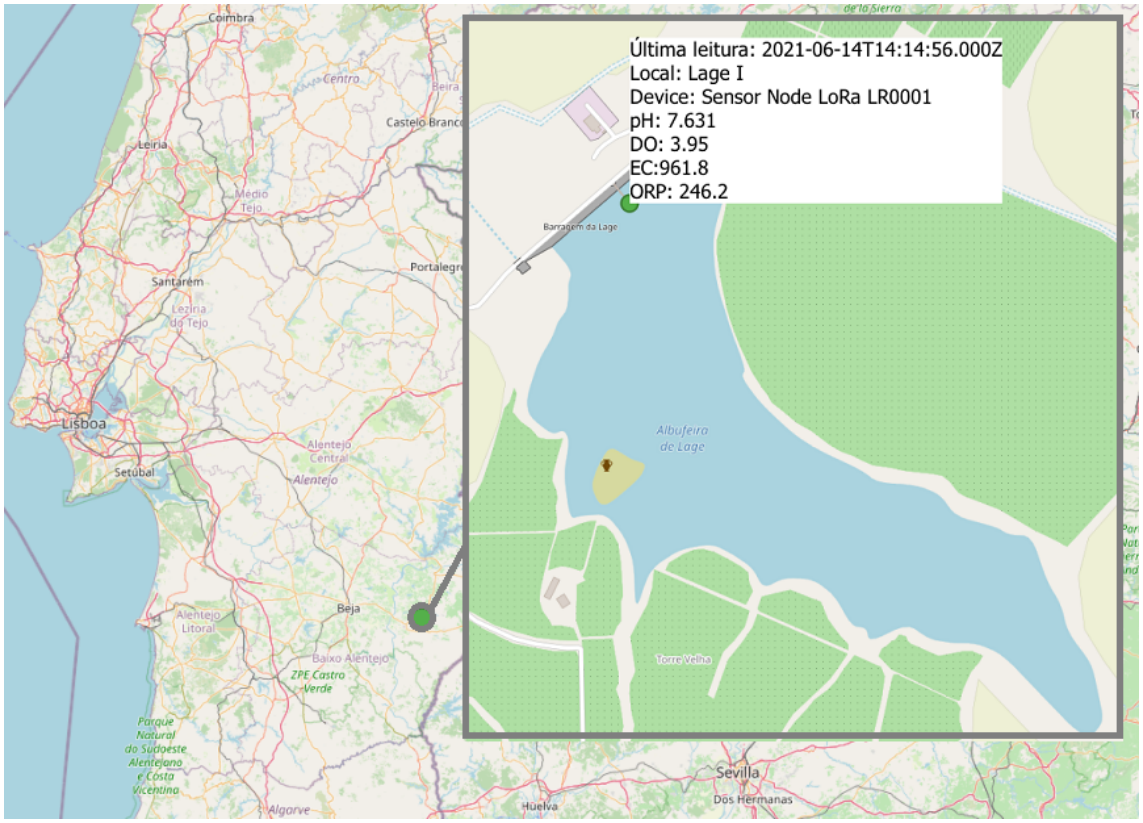


Figura 5.11: Informação do sistema GIS. Fornece dados em «tempo real» da estação de monitorização da qualidade da água localizada na Barragem da Laje (Alentejo, Portugal), incluindo a data da última leitura, o nome do dispositivo, e os parâmetros lidos a partir dos sensores. O campo que representa as coordenadas de localização é a chave que permite a georeferenciação das estações de recolha de dados.

é declarada e executada a função, que armazena os dados. A Figura 5.11 ilustra um exemplo, da utilização dos dados armazenados, exibindo-os geograficamente através da aplicação *desktop QGIS*.

Sistema de Monitorização

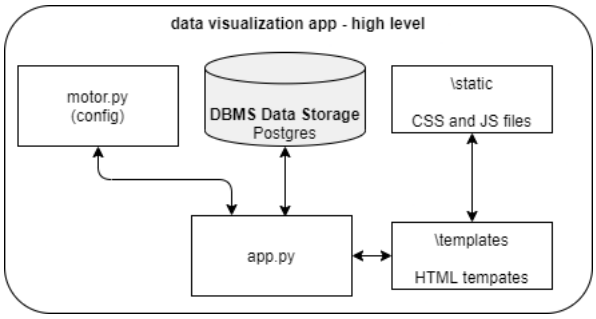


Figura 5.12: Estrutura da aplicação *Data Visualization APP*.

A estrutura da aplicação *«Data Visualization App»* é ilustrada na Figura 5.12. A aplicação *«Data Visualization App»* filtra a informação da base de dados e fornece informação aos utilizadores através de um servidor *HTTP*, ilustrado na Figura 4.7, desenvolvido em *Flask* (*microframework* do ecossistema *Python*). É executada no *contentor «aquaq2_meinheld_1»* (Tabela 5.2), que contém o servidor *HTTP Meinheld WSGI* [66]. Processa em «tempo-real»» os dados armazenados e disponibiliza-os, também em «tempo real» num ambiente *web* [24]

A biblioteca *Matplotlib* [65] é utilizada para desenhar e exibir graficamente o histórico de leituras (Figura 5.15). É uma biblioteca que permite a criação de gráficos estáticos, animados e interativos em *Python*. As leituras são exibidas graficamente através de manómetros (Figuras: 4.8, 5.13, 5.14 e 5.15). Para desenhar os manómetros é utilizado o *plug-in Javascript justgage.js* [21], este possui uma única dependência, a biblioteca de desenho vetorial *raphael.js*[101].

A aplicação é iniciada, através do programa controlador, escrito na linguagem de programação *Python*, escrito no ficheiro *«app.py»* (Apêndice X). Este configura e lança o servidor *Flask*, e define e configura as rotas *HTTP*. Evocado a partir do programa associado ficheiro *«app.py»*, o ficheiro *«motor.py»* (Apêndice X) é o típico ficheiro de configuração *Python*. Garante as credenciais de acesso à base de dados.

A visualização dos dados dos módulos de aquisição de dados em »tempo real» ou o histórico de leituras, permite ter o conhecimento suficiente para antecipar acontecimentos e reagir atempadamente a variações na qualidade da água.

[24] Através das linguagens *Javascript*, *CSS* e *HTML*.

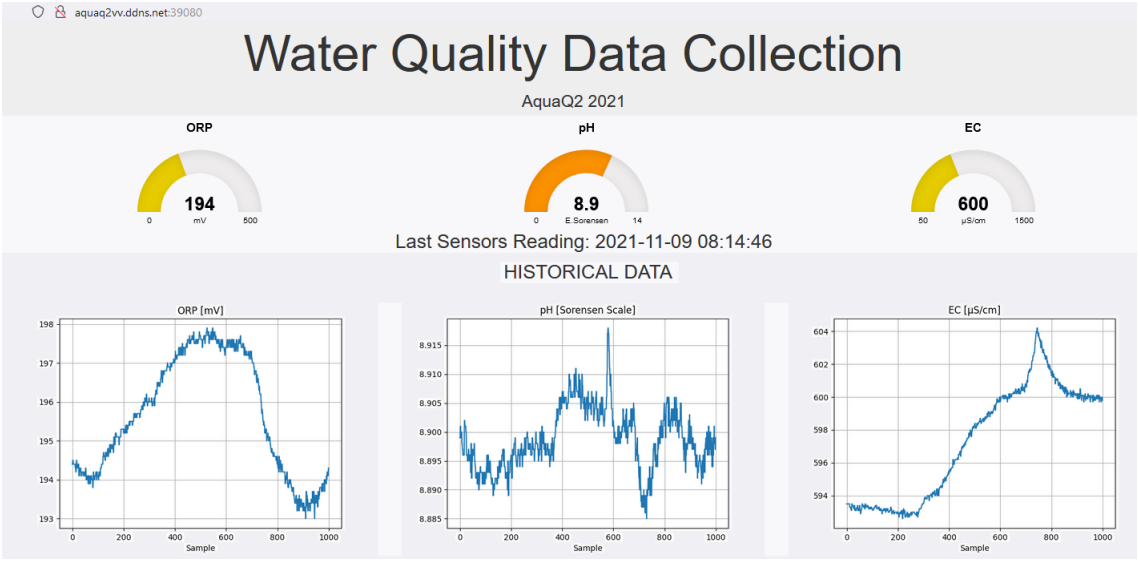


Figura 5.13: Acesso à aplicação de visualização de dados. Exibe leituras em «tempo real» de cada parâmetro do módulo de monitorização de qualidade da água e exibe graficamente o histórico de leituras. Permite monitorizar os parâmetros adquiridos dos módulos de aquisição de dados, oferecendo a possibilidade de interpretar as variações dos diversos parâmetros e atuar em conformidade.

5.6 Conclusão

A adoção de *software open-source* permitiu reduzir custos e experimentar diferentes abordagens para satisfazer os objetivos do *SWQMS*.

A abordagem baseada na *Tecnologia de Contentores* facilita a implementação, manutenção, replicação e escalamento, de aplicações e serviços.

A utilização da tecnologia de *contentores Docker* permite economizar recursos financeiros e computacionais. Implementa uma arquitetura de gestão de *software* que oferece escalabilidade e portabilidade. Destaca-se pela facilidade de utilização e compatibilidade com vários sistemas operativos. Providencia simplicidade, agilidade e estabilidade nos processos de desenvolvimento. Permite experimentar rapidamente programas, isoladamente ou em conjunto, sem afetar o sistema operativo ou outros programas em utilização. A ferramenta *Portainer* um dos muitos interfaces disponíveis para *Docker* facilita a monitorização e gestão dos *contentores*

Os dados recolhidos do ambiente são transmitidos através da *Tecnologia LoRa* e armazenados numa base de dados relacional com extensões geográficas A utilização da informação georreferenciada permite visualizar no mapa os resultados das estações de aquisição de dados.

A tecnologia *LoRa* e a implementação do protocolo *LoRaWAN* através da pilha *ChirpStack* tornam fácil o escalamento e replicação da rede de estações de aquisição de dados.

Através dum ambiente de programação consistente com linguagem de programação

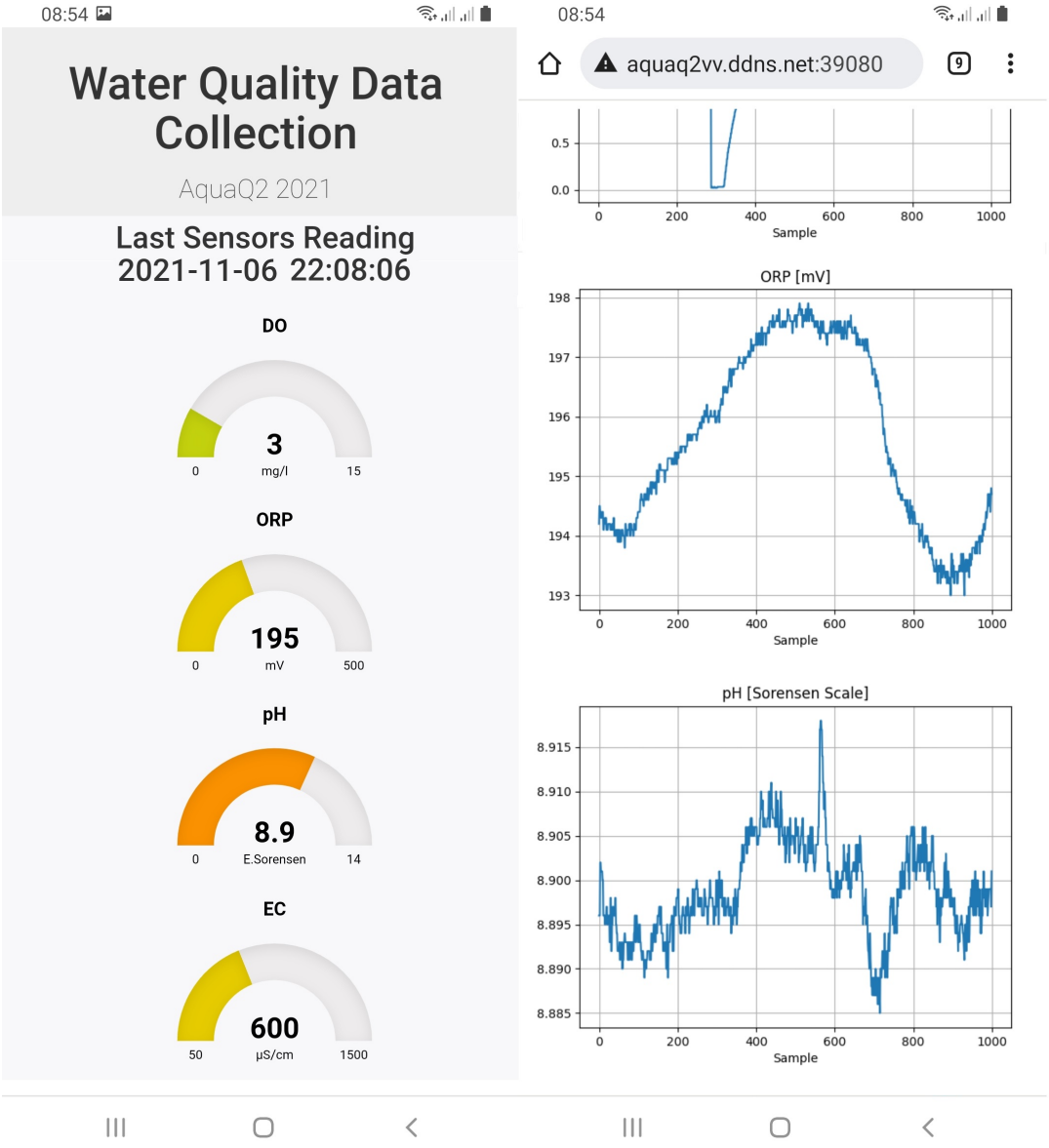


Figura 5.14: Visualização da aplicação *web* num *smartphone*. O *site* é *responsivo*, a visualização é adaptada a qualquer dispositivo com acesso à Internet.

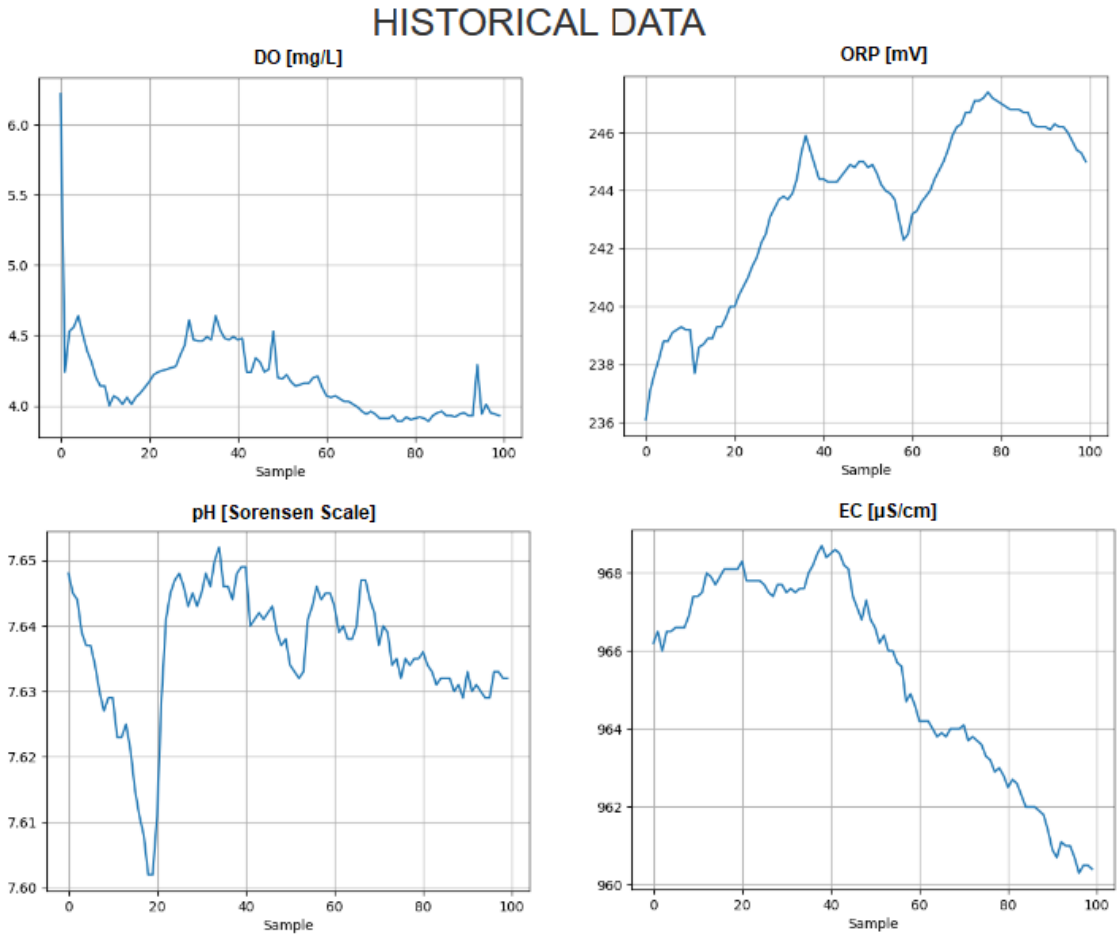


Figura 5.15: Acesso à aplicação de visualização de dados. Exibe graficamente o histórico de leituras, de cada parâmetro dos módulos de aquisição de dados. Permite inserir, o número de amostras, que pretende visualizar.

Python, a informação adquirida é armazenada e disponibilizada em «tempo real», garantindo a fácil monitorização da qualidade da água. A utilização do *Node-RED* fornece um ambiente de testes à aquisição de dados e visualização adicional.

A verificação em «tempo real» da variação dos parâmetros medidos através dos módulos de aquisição de dados, possibilita uma triagem visual e a comparação com valores históricos dos dados obtidos em cada local de amostragem. A detecção de valores anormais deve levar a verificações adicionais. Os utilizadores podem interagir para obter a visualização pretendida. Esta informação chega aos utilizadores e decisores em «tempo real», através de qualquer equipamento com acesso à Internet, oferecendo-lhes conhecimento, para decidir oportunamente. A informação georreferenciada permite aprimorar a análise dos dados adquiridos.

Capítulo 6

Conclusões

Os recursos hídricos, nomeadamente a quantidade e a qualidade da água, sempre tiveram primazia entre os bens sociais. O ritmo acelerado das alterações climáticas, exige a sua utilização criteriosa e torna-se cada vez mais premente. É crucial ter acesso imediato aos dados que a caraterizam de forma fiável, para melhor gerir um recurso tão precioso.

Apesar dos avanços nas tecnologias que permitem realizar a monitorização da qualidade da água, não há um programa ou modelo universal para a realizar. Em síntese, é possível notar semelhanças e diferenças nas reflexões dos autores que se debruçam sobre esse tema, o laço em comum apresentado é que todos tratam de monitorização da qualidade da água de forma automática, através de sensores. Não restam dúvidas que o caminho mais eficiente e económico é a monitorização automática através da utilização de sensores *in-situ*. As tecnologias emergentes precisam de evoluir conjuntamente e tornarem-se economicamente mais acessíveis, sendo desejável um equilíbrio final entre o custo e a facilidade de implementação.

Nos Capítulos anteriores é apresentada a concepção e a implementação de um sistema de monitorização da qualidade da água e do ambiente, em «tempo real», com uma estrutura de *hardware* e *software IoT* que recolhe um conjunto de parâmetros físicos e químicos da água e da atmosfera. O sistema armazena dados num sistema de gestão de base de dados, permitindo a gestão e exibição em tempo real, gerando conhecimento e facilitando decisões de cariz ambiental. Durante os últimos anos, estes subsistemas têm beneficiado da evolução da *IoT*. Podem beneficiar deste tipo de sistema comunidades municipais ou intermunicipais, instituições do Estado na área do ambiente e empresas, na gestão ambiental de rios, albufeiras, barragens, estações de tratamento de águas residuais e canais de rega.

Os aspetos cruciais são: o baixo custo, nomeadamente através da adopção de *software open-source*; serviços fáceis de implementar, manter e escalar, escolhendo uma abordagem tecnológica baseada em contentores, e um ambiente de programação consistente com a linguagem de programação *Python*. Os dados são recolhidos e transmitidos, com o apoio de uma pilha de servidores e serviços *LoraWAN*, armazenados numa base de dados relacional

com extensões geográficas e visualizados com a ajuda de uma estrutura *web*. É também fornecida visualização adicional utilizando o sistema de programação *low-code Node-RED*. É importante salientar o nível de integração que é alcançado pelo sistema de informação geográfica, ligando os dados relativos à água e às condições meteorológicas recolhidos pelo *hardware*, aos dados introduzidos a partir de outras fontes georreferenciadas.

No futuro, o *software* do sistema irá fornecer dados sobre o consumo de energia de todos os dispositivos. Está prevista a integração deste sistema de monitorização com um módulo de lisímetro[1]. Fica em aberto a instalação de outros sensores e/ou atuadores de monitorização remota, por exemplo a velocidade do caudal, o controlo de comportas, a videovigilância, etc., tornando evidente a diversidade de possibilidades que o sistema *IoT* oferece. Devido à natureza crítica dos dados relativos à qualidade da água, julga-se que um sistema de tecnologia *Blockchain* [2] para complementar o armazenamento existente, oferecerá o benefício de proporcionar um mecanismo seguro de gestão e partilha de dados.

A utilização de sistemas modernos de reconhecimento de padrões, a aprendizagem gerada com o apoio de máquinas e *software* de inteligência artificial, em sintonia com técnicas para garantir a fiabilidade da informação irá proporcionar uma visão dinâmica e fácil de seguir, da evolução da qualidade e quantidade de água e das consequências das alterações climáticas e outras actividades relacionadas com a intervenção do Homem, tais como as fontes de poluição. Permitindo prever e prevenir eventos, e monitorizar ameaças à qualidade da água, a longo prazo.

A aplicação do modelo físico *SWAT* (*Soil and Water Assessment Tool*) será considerada para analisar e prever impactos a longo prazo, das medidas de gestão da água e de alterações de usos do solo [119].

Prevê-se que este sistema se torne um instrumento de grande utilidade para a detecção e análise de recursos hídricos com amplas possibilidades científicas.

Nos módulos de aquisição de dados é utilizado *software* adaptado pelo Professor Assistente Dr. João Santos, no âmbito do seu trabalho de doutoramento, a partir de código disponibilizados pelos fabricantes.

Esta dissertação integrou-se com os trabalhos realizados no projeto AquaQ2. Projecto n.º 039494, com as referências ALT20-03-0145-FEDER-039494(SAICT-ALT/39494/2018), financiado pelo Programa Operacional Regional do Alentejo/Portugal 2020/FEDER.

[1] É um dispositivo de medição que mede a quantidade de evapotranspiração, perda de água para a atmosfera por evaporação do solo e por transpiração das plantas.

[2] É o registo/armazenamento de dados descentralizados que podem ser partilhados com segurança.

Bibliografia

[1] UN-2018. *Sustainable Development Goal 6: Synthesis Report 2018 on Water and Sanitation*. UN-Water. New York, USA: United Nations, 2018. 195 pp. ISBN: 978-92-1-101370-2 (citado na página 2).

[2] Antonio Almeida e Jaime González-Arintero. *A Gentle Introduction to IoT Protocols: MQTT, CoAP, HTTP & WebSockets*. Rel. téc. 2017 (citado na página 181).

[3] *Analog and Mixed-Signal Semiconductors*. en. Library Catalog: www.semtech.com. URL: https://www.semtech.com (acedido em 19/04/2020) (citado nas páginas 24, 114, 176).

[4] APA. *Políticas - Água - Planeamento - Planos de Gestão de Região Hidrográfica - 1.º Ciclo*. 2019. URL: https://apambiente.pt/index.php?ref=16%7B%5C&%7Dsubref=7%7B%5C&%7Dsub2ref=9%7B%5C&%7Dsub3ref=834 (acedido em 06/12/2020) (citado na página 8).

[5] *Atlas Scientific - Environmental Robotics*. URL: https://atlas-scientific.com/?gclid=EAIaIQobChMIgvXxz4TW8QIVCLLVCh32ZgAgEAAYASAAEgL0cvD_BwE (acedido em 09/07/2021) (citado nas páginas 123, 124).

[6] *Atlas-Scientific pH Probe*. URL: https://atlas-scientific.com/probes/ph-probe/ (acedido em 20/02/2021) (citado na página 124).

[7] Shajulin Benedict et al. «Real Time Water Quality Analysis Framework using Monitoring and Prediction Mechanisms». Em: *2018 Conference on Information and Communication Technology, CICT 2018* (2018). DOI: 10.1109/INFOCOMTECH.2018.8722381 (citado na página 3).

[8] Sathyajith Bhat. *Practical Docker with Python*. Apress, 2018. ISBN: 9781484237830. DOI: 10.1007/978-1-4842-3784-7 (citado nas páginas 44, 47).

[9] Rizqi Putri Nourma Budiarti et al. «Development of IoT for Automated Water Quality Monitoring System». Em: *Proceedings - 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering, ICOMITEE 2019* 1 (2019), pp. 211–216. DOI: 10.1109/ICOMITEE.2019.8920900 (citado nas páginas 26, 28, 101, 110, 111, 113, 114).

[10] Raul Sousa Carvalho et al. «Hydrometereological Monitoring IoT System - The Software Architecture». Em: *IFIP Advances in Information and Communication Technology*. Ed. por Luis Camarinha-Matos et al. Springer Nature, 2022 (citado na página 4).

[11] Tiago Balieiro Cetrulo, Rui Cunha Marques e Tadeu Fabrício Malheiros. *An analytical review of the efficiency of water and sanitation utilities in developing countries*. Set. de 2019. DOI: 10.1016/j.watres.2019.05.044 (citado na página 93).

[12] Deborah V. Chapman et al. *Water Quality Assessments : A Guide to the Use of Biota, Sediments and Water in Environmental Monitoring*. 2nd Edition. London: London : E & FN Spon, 1996. ISBN: 978-0-419-21600-1 (citado na página 17).

[13] *ChirpStack open-source LoRaWAN - Network Server*. URL: https://www.chirpstack.io/ (acedido em 28/03/2021) (citado nas páginas 36, 37, 52).

[14] M. Cho Zin et al. «Real-time water quality system in internet of things». Em: *IOP Conference Series: Materials Science and Engineering* 495.1 (2019). ISSN: 1757899X. DOI: 10.1088/1757-899X/495/1/012021 (citado nas páginas 103, 111, 113).

[15] Mohammad Salah Uddin Chowdury et al. «IoT based real-time river water quality monitoring system». Em: *Procedia Computer Science* 155 (2019), pp. 161–168. ISSN: 18770509. DOI: 10.1016/j.procs.2019.08.025. URL: https://doi.org/10.1016/j.procs.2019.08.025 (citado nas páginas 26, 28, 31, 97–99, 109, 114).

[16] CNA. *Conselho Nacional da Água*. URL: https://conselhonacionaldaagua.weebly.com/ (acedido em 06/12/2020) (citado nas páginas 1, 9, 11, 87–89).

[17] Rs-Components. «Raspberry Pi Model B». Em: *Raspberrypi.Org* June (2013), p. 1 (citado na página 181).

[18] *Constrained Application Protocol for Internet of Things*. URL: https://www.cse.wustl.edu/%7B~%7Djain/cse574-14/ftp/coap/ (acedido em 13/07/2020) (citado na página 184).

[19] Contimetra. *Analisadores de qualidade de água*. URL: https://www.contimetra.com/PagIA/Produtos.aspx?capitulo=221&catalogo=71#posicionar (acedido em 05/08/2021) (citado na página 118).

[20] Brinda Das e P. C. Jain. «Real-time water quality monitoring system using Internet of Things». Em: *2017 International Conference on Computer, Communications and Electronics, COMPTELIX 2017* (2017), pp. 78–82. DOI: 10.1109/COMPTELIX.2017.8003942 (citado nas páginas 26, 95, 110, 113, 114).

[21] Bojan Djuricic. *justgage - npm*. URL: https://www.npmjs.com/package/justgage (acedido em 16/07/2021) (citado na página 65).

[22] *Docker Documentation*. URL: https://docs.docker.com/ (acedido em 12/07/2021) (citado nas páginas 45–47).

[23] Jianhua Dong et al. «A survey of smart water quality monitoring system». Em: *Environmental Science and Pollution Research* 22.7 (2015), pp. 4893–4906. ISSN: 16147499. DOI: 10.1007/s11356-014-4026-x (citado nas páginas 9, 17, 18, 104).

[24] DRE. *Decreto-Lei n.º 236/98 - Estabelece normas, critérios e objectivos de qualidade com a*. URL: https://dre.pt/web/guest/legislacao-consolidada/-/lc/75031534/indice (acedido em 06/12/2020) (citado nas páginas 7, 8).

[25] Charlotte Dupont, Philippe Cousin e Samuel Dupont. «IoT for Aquaculture 4.0 Smart and easy-to-deploy real-time water monitoring with IoT». Em: *2018 Global Internet of Things Summit (GIoTS)*. 2018, pp. 1–5. DOI: 10.1109/GIOTS.2018.8534581 (citado na página 10).

[26] *Eclipse Mosquitto*. URL: https://mosquitto.org/ (acedido em 15/07/2021) (citado na página 53).

[27] *Ecosens Aquamonitrix*. URL: https://www.ecosensaquamonitrix.eu/ (acedido em 12/12/2020) (citado na página 102).

[28] *EGFheal SDR RTL2832U R828D A300U DAB FM Receiving Frequency 25MHz-1760MHz*. URL: ttps://www.amazon.co.uk/EGFheal-RTL2832U-Receiving-Frequency-25MHz-1760MHz-blue/dp/B08KG34K7B (acedido em 01/07/2021) (citado na página 123).

[29] *Empowering App Development for Developers - Docker*. URL: https://www.docker.com/ (acedido em 26/06/2021) (citado nas páginas 44, 46).

[30] Cesar Encinas et al. «Design and implementation of a distributed IoT system for the monitoring of water quality in aquaculture». Em: *Wireless Telecommunications Symposium* (2017). ISSN: 19345070. DOI: 10.1109/WTS.2017.7943540 (citado nas páginas 26, 27, 30, 97, 98, 109, 113, 114).

[31] EPA, ed. *Parameters of Water Quality: Interpretation and Standards*. Wexford, Ireland: Environmental Protection Agency, Ireland, 2001. 133 pp. (citado na página 17).

[32] *Estação Meteorológica Steinberg Systems SBS-WS-600 WiFi*. URL: https://www.expondo.pt/steinberg-systems-estacao-meteorologica-sem-fio-2-sensores-10030583 (acedido em 10/06/2021) (citado na página 123).

[33] SUWANU Europe. *International water projects in Alentejo, Portugal*. URL: https://suwanu-europe.eu/alentejo-portugal/ (acedido em 26/03/2021) (citado na página 2).

[34] Uniao Europeia. *Diretiva 2000/60/CE*. URL: https://eur-lex.europa.eu/resource.html?uri=cellar:5c835afb-2ec6-4577-bdf8-756d3d694eeb.0009.02/DOC%7B%5C_%7D1%7B%5C&%7Dformat=PDF (acedido em 06/12/2020) (citado nas páginas 7, 8).

[35] *FocalFossa/ReleaseNotes - Ubuntu Wiki*. URL: https://wiki.ubuntu.com/FocalFossa/ReleaseNotes (acedido em 15/10/2021) (citado na página 59).

[36] Python Software Foundation. *Comparing Python to Other Languages - Python.org*. URL: https://www.python.org/doc/essays/comparisons/ (acedido em 13/06/2021) (citado na página 59).

[37] Lou Frenzel. *Oh, No! Not Another IIoT Wireless Technology*. en-us. Library Catalog: www.mwrf.com. Nov. de 2017. URL: https://www.mwrf.com/technologies/systems/article/21848780/oh-no-not-another-iiot-wireless-technology (acedido em 24/04/2020) (citado na página 178).

[38] Vaishnavi V. Daigavane Gaikwad e Dr. M.A. «Water Quality Monitoring System Based on IoT». Em: *ICDCS 2020 - 2020 5th International Conference on Devices, Circuits and Systems* 10.5 (2020), pp. 279–282. DOI: 10.1109/ICDCS48716.2020.243598 (citado nas páginas 96, 109, 114).

[39] Guandong Gao, Ke Xiao e Miaomiao Chen. «An intelligent IoT-based control and traceability system to forecast and maintain water quality in freshwater fish farms». Em: *Computers and Electronics in Agriculture* 166.August (2019), p. 105013. ISSN: 01681699. DOI: 10.1016/j.compag.2019.105013. URL: https://doi.org/10.1016/j.compag.2019.105013 (citado nas páginas 24, 27, 31, 101, 113, 114).

[40] Federico Di Gregorio. *psycopg2 · PyPI*. URL: https://pypi.org/project/psycopg2/ (acedido em 28/06/2021) (citado na página 38).

[41] Dominique D. Guinard e Vlad M. Trifa. *Building the Web of Things*. 1^{nd}. ISBN: 9781617292682. Manning Publications Co., 2016 (citado nas páginas 179, 180, 184).

[42] Hani Purwati Hanifah e Suhono Harso Supangkat. «IoT-based River Water Quality Monitoring Design for Smart Environments in Cimahi City». Em: *Proceedings of the International Conference on Electrical Engineering and Informatics* 2019-July.July (2019), pp. 496–499. ISSN: 21556830. DOI: 10.1109/ICEEI47359.2019.8988883 (citado nas páginas 26, 27, 97, 109, 113).

[43] Red Hat. *O que é Docker?* URL: https://www.redhat.com/pt-br/topics/containers/what-is-docker (acedido em 26/06/2021) (citado na página 45).

[44] *Home*. en-US. Library Catalog: zigbeealliance.org. URL: https://zigbeealliance.org/ (acedido em 19/04/2020) (citado nas páginas 113, 173).

[45] *HTML5 WebSocket - A Quantum Leap in Scalability for the Web*. URL: http://www.websocket.org/quantum.html (acedido em 21/12/2020) (citado na página 183).

[46] *Hypertext Transfer Protocol – Wikipédia, a enciclopédia livre*. URL: https://pt.wikipedia.org/wiki/Hypertext%7B%5C_%7DTransfer%7B%5C_%7DProtocol (acedido em 10/07/2020) (citado na página 184).

[47] Docker Inc. *Best practices for writing Dockerfiles - Docker Documentation*. URL: https://docs.docker.com/develop/develop-images/dockerfile_best-practices/ (acedido em 11/07/2021) (citado na página 46).

[48] *IPv6 over Low power WPAN (6lowpan)* -. URL: https://datatracker.ietf.org/wg/6lowpan/about/ (acedido em 19/04/2020) (citado na página 174).

[49] Telecommunication Standardization Sector of ITU. *ITU-T Recommendation Y.4000/Y.2060 : Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks, Next Generation Networks-Frameworks and Functional Architecture Models, Overview of the Internet of Things*. ITU. 2012. URL: https://www.itu.int/rec/T-REC-Y.2060-201206-I (citado nas páginas 17, 33).

[50] Jiping Jiang et al. «A comprehensive review on the design and optimization of surface water quality monitoring networks». en. Em: *Environmental Modelling & Software* 132 (out. de 2020), p. 104792. ISSN: 13648152. DOI: 10.1016/j.envsoft.2020.104792. URL: https://linkinghub.elsevier.com/retrieve/pii/S1364815220300049 (acedido em 26/12/2020) (citado na página 18).

[51] Steven J. Johnston et al. «Applicability of commodity, low cost, single board computers for Internet of Things devices». Em: *2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016* (2017), pp. 141–146. DOI: 10.1109/WF-IoT.2016.7845414 (citado nas páginas 22, 100).

[52] Nurliyana Kafli e Khalid Isa. «Internet of Things (IoT) for measuring and monitoring sensors data of water surface platform». Em: *2017 IEEE 7th International Conference on Underwater System Technology: Theory and Applications, USYS 2017* 2018-January (2018), pp. 1–6. DOI: 10.1109/USYS.2017.8309441 (citado nas páginas 27, 99, 110, 114).

[53] Kamarul Hafiz Kamaludin. «Water Quality Monitoring Z ith Internet R f Things (IoT)». Em: *2017 IEEE Conference on Systems, Process and Control (ICSPC 2017), 15–17 December 2017, Melaka, Malaysia Water* December (2017), pp. 15–17 (citado nas páginas 26, 97, 98, 109, 114, 115).

[54] *KVM*. URL: https://www.linux-kvm.org/page/Main_Page (acedido em 15/10/2021) (citado na página 59).

[55] Aleksey Kychkin et al. «Architecture of Compressor Equipment Monitoring and Control Cyber-Physical System Based on Influxdata Platform». Em: *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. 2019, pp. 1–5. DOI: 10.1109/ICIEAM.2019.8742963 (citado na página 28).

[56] Manuel Lameira. «Sistema de Monitorização de Canais de Rega». MSc Thesis. Beja: Instituto Politécnico de Beja, 2020 (citado nas páginas 100, 110, 113, 114).

[57] Teng Li et al. «Automated water quality survey and evaluation using an IoT platform with mobile sensor nodes». Em: *Sensors (Switzerland)* 17.8 (2017). ISSN: 14248220. DOI: 10.3390/s17081735 (citado nas páginas 27, 102, 109, 111, 113, 114).

[58] *LILYGO-TTGO LORA32 V2.0 433/868/915Mhz ESP32 LoRa*. URL: http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1319&FId=t3:50003:3 (acedido em 21/02/2021) (citado nas páginas 123, 124).

[59] Predictable Designs LLC. *Comparison of Wireless Technologies*. en-US. https://predictabledesigns.com/. Nov. de 2018. (Acedido em 17/04/2020) (citado nas páginas 173, 176).

[60] Robocore Tecnologia LTDA. *Comparação Entre Protocolos de Comunicação Serial - Tutoriais - RoboCore*. URL: https://www.robocore.net/tutoriais/comparacao-entre-protocolos-de-comunicacao-serial.html (acedido em 21/02/2021) (citado na página 185).

[61] Mohammad Saeid Mahdavinejad et al. «Machine Learning for Internet of Things Data Analysis: A Survey». en. Em: *Digital Communications and Networks* 4.3 (ago. de 2018), pp. 161–175. ISSN: 2352-8648. DOI: 10.1016/j.dcan.2017.10.002 (citado na página 35).

[62] A. Maier, A. Sharp e Y. Vagapov. «Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things». Em: *2017 Internet Technologies and Applications (ITA)*. 2017, pp. 143–148. DOI: 10.1109/ITECHA.2017.8101926 (citado na página 110).

[63] Ramón Martínez et al. «On the use of an IoT integrated system for water quality monitoring and management in wastewater treatment plants». Em: *Water (Switzerland)* 12.4 (2020). ISSN: 20734441. DOI: 10.3390/W12041096 (citado nas páginas 27, 102, 110, 113).

[64] Ismael Rodrigues Martins e José Luís Zem. «Estudo Dos Protocolos De Comunicação Mqtt E Coap Para». Em: (2015), pp. 64–87 (citado nas páginas 181, 183).

[65] *Matplotlib: Python plotting — Matplotlib 3.4.2 documentation*. URL: https://matplotlib.org/ (acedido em 16/07/2021) (citado na página 65).

[66] *Meinheld*. URL: https://meinheld.org/ (acedido em 12/06/2021) (citado na página 65).

[67] Hélio Sousa Mendonça. *SPI e I2C*. URL: https://paginas.fe.up.pt/%7B~%7Dhsm/docencia/comp/spi-e-i2c/ (acedido em 21/02/2021) (citado na página 185).

[68] Gayathri S. Menon, Maneesha Vinodini Ramesh e P. Divya. «A low cost wireless sensor network for water quality monitoring in natural water bodies». Em: *GHTC 2017 - IEEE Global Humanitarian Technology Conference, Proceedings* 2017-January (2017), pp. 1–8. DOI: 10.1109/GHTC.2017.8239341 (citado nas páginas 27, 96, 109, 113, 114).

[69] *MikroTik Routers and Wireless - Products: wAP LTE kit*. URL: https://mikrotik.com/product/wap_lte_kit (acedido em 10/07/2021) (citado na página 123).

[70] Roberto Minerva, Abyi Biru e Domenico Rotondi. *Towards a Definition of the Internet of Things (IoT)*. 2015 (citado na página 17).

[71] *MIOTY – The Wireless IoT Platform*. en. Library Catalog: www.iis.fraunhofer.de. URL: https://www.iis.fraunhofer.de/en/ff/lv/net/telemetrie.html (acedido em 20/04/2020) (citado na página 176).

[72] *MIT License*. URL: https://mit-license.org/ (acedido em 25/06/2021) (citado nas páginas 36, 52).

[73] *MQTT and CoAP: Security and Privacy Issues in IoT and IIoT Communication Protocols - Security News - Trend Micro USA*. URL: https://www.trendmicro.com/vinfo/us/security/news/internet-of-things/mqtt-and-coap-security-and-privacy-issues-in-iot-and-iiot-communication-protocols (acedido em 13/07/2020) (citado na página 183).

[74] Cho Zin Myint, Lenin Gopal e Yan Lin Aung. «WSN-based reconfigurable water quality monitoring system in IoT environment». Em: *ECTI-CON 2017 - 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (2017), pp. 741–744. DOI: 10.1109/ECTICon.2017.8096345 (citado nas páginas 103, 111).

[75] Cho Zin Myint, Lenin Gopal e Yan Lin Aung. «WSN-based reconfigurable water quality monitoring system in IoT environment». Em: *ECTI-CON 2017 - 2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology* (2017), pp. 741–744. DOI: 10.1109/ECTICon.2017.8096345 (citado na página 113).

[76] *Netilion Smart System para águas superficiais* (citado na página 117).

[77] Muhammad Niswar et al. «IoT-based water quality monitoring system for soft-shell crab farming». Em: *Proceedings - 2018 IEEE International Conference on Internet of Things and Intelligence System, IOTAIS 2018* (2019), pp. 6–9. DOI: 10.1109/IOTAIS.2018.8600828 (citado nas páginas 25, 102, 109, 111, 114).

[78] *Node-RED*. URL: https://nodered.org/ (acedido em 11/06/2021) (citado nas páginas 39, 61).

[79] *O TCP/IP*. URL: https://paginas.fe.up.pt/%7B~%7Dmrs01003/TCP%7B%5C_%7DIP.htm (acedido em 13/07/2020) (citado na página 180).

[80] Segun O. Olatinwo e Trudi-H. Joubert. «Enabling Communication Networks for Water Quality Monitoring Applications: A Survey». Em: *IEEE Access* 7 (2019), pp. 100332–100362. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2904945 (citado na página 17).

[81] *Online Multi-parameter Water Quality Analyzer*. URL: https://www.winmoreltd.com/product/621/ (acedido em 07/08/2021) (citado na página 120).

[82] S. Pandit e V. N. Shet. «Review of FPGA based control for switch mode converters». Em: *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. 2017, pp. 1–5. DOI: 10.1109/ICECCT.2017.8117996 (citado nas páginas 23, 111, 112).

[83] M. Parameswari e M. Balasingh Moses. «Online measurement of water quality and reporting system using prominent rule controller based on aqua care-IOT». Em: *Design Automation for Embedded Systems* 22.1-2 (2018), pp. 25–44. ISSN: 15728080. DOI: 10.1007/s10617-017-9187-7 (citado nas páginas 27, 97, 98, 109, 114).

[84] *pgAdmin - PostgreSQL Tools*. URL: https://www.pgadmin.org/ (acedido em 07/12/2020) (citado na página 58).

[85] Evander Pierre. *NFC and Internet of Things*. en-US. Library Catalog: nfc-forum.org. URL: https://nfc-forum.org/what-is-nfc/ (acedido em 19/04/2020) (citado na página 172).

[86] *Policy Brief on Climate Change and Water launched during High-level Political Forum on Sustainable Development - UN-Water*. URL: https://www.unwater.org/policy-brief-on-climate-change-and-water/ (acedido em 18/06/2021) (citado na página 1).

[87] *Portainer - Open Source Container Management GUI for Kubernetes, Docker, Swarm*. URL: https://www.portainer.io/ (acedido em 26/06/2021) (citado na página 47).

[88] *PostGIS — Spatial and Geographic Objects for PostgreSQL*. URL: https://postgis.net/ (acedido em 11/06/2021) (citado nas páginas 38, 55, 56).

[89] *Postgres - Docker Hub*. URL: https://hub.docker.com/%7B%5C_%7D/postgres (acedido em 10/12/2020) (citado na página 55).

[90] *PostgreSQL For Development With Vagrant - PostgreSQL wiki*. URL: https://wiki.postgresql.org/wiki/PostgreSQL%7B%5C_%7DFor%7B%5C_%7DDevelopment%7B%5C_%7DWith%7B%5C_%7DVagrant (acedido em 10/07/2020) (citado na página 55).

[91] *PostgreSQL: Documentation: 12: 1.1. Installation*. URL: https://www.postgresql.org/docs/current/tutorial-install.html (acedido em 10/12/2020) (citado na página 55).

[92] *PostgreSQL: The world's most advanced open source database*. URL: https://www.postgresql.org/ (acedido em 23/12/2020) (citado na página 55).

[93] Alif Akbar Pranata, Jae Min Lee e Dong Seong Kim. «Towards an IoT-based water quality monitoring system with brokerless pub/sub architecture». Em: *IEEE Workshop on Local and Metropolitan Area Networks* 2017-June (2017). ISSN: 19440375. DOI: 10.1109/LANMAN.2017.7972166 (citado nas páginas 97, 109, 113, 114).

[94] A. N. Prasad et al. «Smart water quality monitoring system». Em: *2015 2nd Asia-Pacific World Congress on Computer Science and Engineering, APWC on CSE 2015* (2016). DOI: 10.1109/APWCCSE.2015.7476234 (citado nas páginas 12, 26, 30, 89, 96, 109, 113, 114).

[95] *Professional grade outdoor LoRaWAN^{TM} gateway - LORIX One*. URL: https://www.lorixone.io/en (acedido em 10/07/2021) (citado na página 123).

[96] «Proposta de tema para Dissertação de Mestrado Título da Dissertação Palavras-Chave Tema da Dissertação Local de orientação e realização Proposta de tema para Dissertação de Mestrado Referências». Em: (), pp. 1–3 (citado na página 3).

[97] *QGIS Cloud*. URL: https://qgiscloud.com/en/pages/quickstart (acedido em 10/12/2020) (citado na página 56).

[98] *QGIS Plugins*. URL: https://plugins.qgis.org/ (acedido em 10/12/2020) (citado na página 56).

[99] K. Raghu Sita Rama Raju e G. Harish Kumar Varma. «Knowledge based real time monitoring system for aquaculture Using IoT». Em: *Proceedings - 7th IEEE International Advanced Computing Conference, IACC 2017* (2017), pp. 318–321. DOI: 10.1109/IACC.2017.0075 (citado nas páginas 26, 101, 110, 111, 114).

[100] Maria Gabriela Araujo Ranieri et al. «Como escrever uma dissertação de mestrado». Em: *Research, Society and Development* 9.3 (fev. de 2020), e149932584. ISSN: 2525-3409. DOI: 10.33448/rsd-v9i3.2584. URL: https://rsdjournal.org/index.php/rsd/article/view/2584 (citado na página 93).

[101] Raphaël. *An Intro to Raphaël - Raphaël*. URL: http://raphaeljs.com/ (acedido em 16/07/2021) (citado na página 65).

[102] Bruna Rasmussen. *LAN, WLAN, MAN, WAN, PAN: conheça os principais tipos de redes*. pt-BR. URL: https://canaltech.com.br/infra/lan-wlan-man-wan-pan-conheca-os-principais-tipos-de-redes/ (acedido em 22/04/2020) (citado na página 172).

[103] M. Rohith Ananth Ratnam et al. «Iot based water quality testing system in aquaculture». Em: *International Journal of Engineering and Advanced Technology* 8.6 Special issue (2019), pp. 981–983. ISSN: 22498958. DOI: 10.35940/ijeat.F1187.0886S19 (citado nas páginas 26, 101, 110, 114).

[104] Purushottam Sarda Sadgir e Parag. «Assessment of Multi Parameters of Water Quality in Surface Water Bodies-A Review.ISSN: 2321-9653». Em: 3.October (2015), pp. 331–336. URL: https://www.researchgate.net/publication/282356694 (citado nas páginas 3, 90, 91).

[105] Prashant Salunke e Jui Kate. «Advanced smart sensor interface in internet of things for water quality monitoring». Em: *2017 International Conference on Data Management, Analytics and Innovation, ICDMAI 2017* (2017), pp. 298–302. DOI: 10.1109/ICDMAI.2017.8073529 (citado nas páginas 26, 101, 102, 111, 114).

[106] João Miguel Santos et al. «A Smart IoT System for Water Monitoring and Analysis». Em: *Smart Objects and Technologies for Social Good*. Ed. por Ivan Miguel Pires et al. Springer International Publishing, 2021. ISBN: 978-3-030-91420-2. URL: https://link.springer.com/book/9783030914202 (citado na página 4).

[107] João Miguel Santos et al. «Physical and chemical water quality parameters sensing IoT systems for improving water productivity». Em: *Water Productivity Journal* 1.2 (2020), pp. 33–46 (citado nas páginas 3, 17).

[108] K. Saravanan et al. «Real-time water quality monitoring using Internet of Things in SCADA». Em: *Environmental Monitoring and Assessment* 190.9 (2018). ISSN: 15732959. DOI: 10.1007/s10661-018-6914-x (citado nas páginas 27, 96, 97, 109, 113, 114).

[109] *Securing MQTT Systems - MQTT Security Fundamentals*. URL: https://www.hivemq.com/blog/mqtt-security-fundamentals-payload-encryption/ (acedido em 13/07/2020) (citado na página 181).

[110] Dimitrios Serpanos e Marilyn Wolf. *Internet-of-Things (IoT) Systems: Architectures, Algorithms, Methodologies*. Springer International Publishing, 2018. ISBN: 978-3-319-69714-7 (citado na página 33).

[111] Uferah Shafi et al. «Surface Water Pollution Detection using Internet of Things». Em: *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT and IoT, HONET-ICT 2018* (2018), pp. 92–96. DOI: 10.1109/HONET.2018.8551341 (citado nas páginas 26, 31, 97, 98, 109, 114).

[112] *Sigfox*. en. Page Version ID: 944702646. Mar. de 2020. URL: https://en.wikipedia.org/w/index.php?title=Sigfox&oldid=944702646 (acedido em 20/04/2020) (citado nas páginas 25, 176).

[113] *Sigfox - The Global Communications Service Provider for the Internet of Things (IoT)*. en. Library Catalog: www.sigfox.com. URL: https://www.sigfox.com/en (acedido em 20/04/2020) (citado nas páginas 25, 176).

[114] Vitor Silva, Marite Kirikova e Gundars Alksnis. «Containers for Virtualization: An Overview». Em: *Applied Computer Systems* 23 (mai. de 2018), pp. 21–27. DOI: 10.2478/acss-2018-0003 (citado na página 44).

[115] Daudi S. Simbeye, Jimin Zhao e Shifeng Yang. «Design and deployment of wireless sensor networks for aquaculture monitoring and control based on virtual instruments». Em: *Computers and Electronics in Agriculture* 102 (2014), pp. 31–42. ISSN: 01681699. DOI: 10.1016/j.compag.2014.01.004. URL: http://dx.doi.org/10.1016/j.compag.2014.01.004 (citado nas páginas 95, 109, 113).

[116] K. Spandana e V. R.Seshagiri Rao. «Internet of Things (Iot) Based smart water quality monitoring system». Em: *International Journal of Engineering and Technology(UAE)* 7.3 (2018), pp. 259–262. ISSN: 2227524X. DOI: 10.14419/ijet.v7i3.6.14985 (citado nas páginas 100, 110, 113, 114).

[117] *Standards for the IoT*. URL: https://www.3gpp.org/ (acedido em 25/04/2020) (citado nas páginas 25, 114, 177).

[118] *SWARM Buoys-Real-Time Water Quality Monitoring in Reservoirs*. URL: https://www.veolia.com/anz/sites/g/files/dvc2011/files/document/2017/10/VW_SWARM_Brochure_FINAL.pdf (acedido em 07/08/2021) (citado na página 119).

[119] *SWAT - Soil & Water Assessment Tool*. URL: https://swat.tamu.edu/ (acedido em 21/10/2021) (citado na página 70).

[120] Deliverable D Task. *Review of sensors to monitor water quality ERNCIP thematic area Chemical & Biological Risks in the Water Sector*. December. 2013. ISBN: 9789279346187. DOI: 10.2788/35499 (citado nas páginas 18, 104).

[121] *Teach, Learn, and Make with Raspberry Pi*. URL: https://www.raspberrypi.org/ (acedido em 10/07/2021) (citado na página 123).

[122] *The new LORIX OS release candidate is ready for testing! - LORIX One*. URL: https://www.lorixone.io/en/news/lorix-os (acedido em 16/10/2021) (citado na página 59).

[123] TIOBE. *TIOBE - The Software Quality Company*. URL: https://www.tiobe.com/tiobe-index/ (acedido em 16/10/2021) (citado na página 59).

[124] *Top Programming Languages 2021 - IEEE Spectrum*. URL: https://spectrum.ieee.org/top-programming-languages-2021 (acedido em 16/10/2021) (citado na página 59).

[125] *Top Programming Languages 2021 - IEEE Spectrum*. URL: https://spectrum.ieee.org/top-programming-languages-2021 (acedido em 16/10/2021) (citado na página 59).

[126] U.S EPA. «Online source water quality monitoring for water quality surveillance and response systems». Em: *U.S.Environmental Protection Agency* September (2016), p. 114. URL: https://www.epa.gov/sites/production/files/2016-09/documents/online%7B%5C_%7Dsource%7B%5C_%7Dwater%7B%5C_%7Dmonitoring%7B%5C_%7Dguidance.pdf (citado nas páginas 10, 13).

[127] *UE-Start-up activities for Advanced Signalling and Automation Systems*. Fev. de 2019. URL: https://ec.europa.eu/ (acedido em 19/04/2020) (citado na página 177).

[128] UN. *Sustainable Development Goal 6 Synthesis Report 2018 on Water and Sanitation*. 2018, p. 199. ISBN: 978-92-1-101370-2 (citado na página 1).

[129] *UWB – O que é a rede Ultra Wide Band?* pt-br. Library Catalog: www.novida.com.br. URL: https://www.novida.com.br/blog/uwb/ (acedido em 19/04/2020) (citado na página 175).

[130] N. Vijayakumar e R. Ramya. «The real time monitoring of water quality in IoT environment». Em: *ICIIECS 2015 - 2015 IEEE International Conference on Innovations in Information, Embedded and Communication Systems* (2015). DOI: 10.1109/ICIIECS.2015.7193080 (citado nas páginas 26, 100, 110, 114).

[131] *Water Management IoT Technology Solution - Libelium*. URL: https://www.libelium.com/iot-solutions/smart-water/ (acedido em 07/08/2021) (citado na página 121).

[132] *What is REST*. URL: https://restfulapi.net/ (acedido em 13/07/2020) (citado na página 185).

[133] *What is RFID?* en-US. Library Catalog: www.epc-rfid.info. URL: https://www.epc-rfid.info/rfid (acedido em 19/04/2020) (citado na página 173).

[134] WHO. *WHO - World Health Organization*. URL: https://www.who.int/en/ (acedido em 08/12/2020) (citado nas páginas 29, 30).

[135] *Wi-Fi*. pt. Page Version ID: 57769858. Mar. de 2020. URL: https://pt.wikipedia.org/w/index.php?title=Wi-Fi&oldid=57769858 (acedido em 19/04/2020) (citado nas páginas 114, 175).

[136] *WiMAX Forum | AeroMACS, WiGRID, and WiMAX Advanced Technologies*. URL: http://wimaxforum.org/ (acedido em 21/04/2020) (citado na página 178).

[137] WMO. *Planning of Water Quality Monitoring Systems*. 3. 2013, p. 128. ISBN: 978926311 1135 (citado nas páginas 9–12, 87–90).

[138] Brandon P. Wong e Branko Kerkez. «Real-time environmental sensor data: An application to water quality using web services». Em: *Environmental Modelling and Software* 84 (2016), pp. 505–517. ISSN: 13648152. DOI: 10.1016/j.envsoft.2016.07.020. URL: http://dx.doi.org/10.1016/j.envsoft.2016.07.020 (citado nas páginas 26, 103, 111, 113, 115).

[139] *World Population Prospects - Population Division - United Nations*. URL: https://population.un.org/wpp/Publications/ (acedido em 18/06/2021) (citado na página 2).

[140] Joseph Yiu e Ian Johnson. «The Many Ways of Programming an ARM Cortex -M Microcontroller». Em: *ARM White Papers* (2013), pp. 1–19 (citado na página 22).

[141] *Z-Wave Mesh Network Wireless Solutions - Smart Home - Silicon Labs*. URL: https://www.silabs.com/wireless/z-wave (acedido em 19/04/2020) (citado na página 174).

[142] Zaryanti Zainuddin, Riswan Idris e Asmawaty Azis. «Water Quality Monitoring System for Vannamae Shrimp Cultivation Based on Wireless Sensor Network In Taipa, Mappakasunggu District, Takalar». Em: 165.ICMEMe 2018 (2019), pp. 89–92. DOI: 10.2991/icmeme-18.2019.20 (citado nas páginas 97, 109, 113, 114).

[143] Chengcheng Zhang, Jian Wu e Jiancheng Liu. «Water quality monitoring system based on Internet of Things». Em: *Proceedings - 2020 3rd International Conference on Electron Device and Mechanical Engineering, ICEDME 2020* (2020), pp. 727–730. DOI: 10.1109/ICEDME50972.2020.00171 (citado nas páginas 26, 99, 110, 114).

[144] Zewen Zhang et al. «Development of remote monitoring system for aquaculture water quality based on Internet of Things». Em: *IOP Conference Series: Materials Science and Engineering* 768.5 (2020), pp. 6–11. ISSN: 1757899X. DOI: 10.1088/1757-899X/768/5/052033 (citado nas páginas 26, 99, 100, 110, 114).

Apêndices

Apêndice I

Propriedades Físicas ou Químicas da Água

Este apêndice é um complemento à Seção 2.2 «Propriedades Físicas ou Químicas da Água», apresentada no Capitulo 2.

A Confederação Nacional da Agricultura (CNA) [16] indica como principais propriedades das águas de superfície: a *temperatura*, o *pH*, a *condutividade* elétrica, o *oxigénio dissolvido*, o *potencial de redução de oxidação*, a *turbidez*, os *sólidos totais dissolvidos*, o *dióxido de carbono*, o *sulfato*, a *amónia*, o *nitrato*, a *nitrito* e o *carbono*. As principais caraterísticas destes parâmetros/propriedades das águas de superfície são apresentados nas secções seguintes.

Temperatura

Unidade de medida do SI: [°C]. É fortemente influenciada pela *temperatura* do ar, pela latitude, altitude, estação do ano, hora do dia, circulação do ar, nebulosidade e fluxo e profundidade do corpo de água. medição. Afeta a *vida biológica* na água, bem como o número de reações químicas, *solubilidade*, *condutividade* e *toxicidade da água*. Por sua vez, a *temperatura* afeta os processos físicos, químicos e biológicos em corpos d'água e, portanto, a concentração de muitas variáveis. Conforme a *temperatura* da água aumenta, a taxa de *reações químicas* geralmente aumentam, juntamente com a *evaporação* e *volatilização* de substâncias da água. O aumento da *temperatura* também diminui a *solubilidade dos gases* na água, como o oxigénio, *dióxido de carbono*, nitrogénio, metano e outros.

A monitorização e controlo da *temperatura* água nas diversas massas de água é essencial em estudos de auto purificação das mesmas, para a vida aquática, para fins de refrigeração na indústria, para o consumo humano e rega agrícola [137].

pH

Unidade de medida do SI: [Escala de Sorensen]. O valor de pH é uma importante característica da água, já que é capaz de afetar os organismos aquáticos, sendo ainda um indicador potencial do aumento da poluição ou de alteração de qualquer outro fator ambiental. O pH (potencial de hidrogénio) é uma medida de como ácida/básica a água.

O valor de pH varia entre 0 e 14, sendo 7 o valor de pH neutro (escala Sorensen). Valores de pH inferiores a 7 indicam acidez, enquanto um pH superior a 7 indica uma base. Uma vez que o pH pode ser afetado pela presença de produtos químicos na água, o pH é um indicador importante de água que está a mudar quimicamente.

O pH é reportado em «unidades logarítmicas». Cada unidade representa uma mudança de 10 vezes na acidez/basicidade da água. A água com um pH de cinco é dez vezes mais ácida que a água com um pH de seis [137].

Condutividade Elétrica

Unidade de medida do SI: [μS/cm]. A água pura (sem outras substâncias) é uma excelente isoladora, não conduzindo a eletricidade, deixa de ser um excelente isolante à medida que começa a dissolver as substâncias com as quais contacta, já que mesmo uma pequena quantidade de *iões* numa solução de água torna-a capaz de conduzir eletricidade. A água é reconhecida como o solvente universal. Quanto maior a quantidade de *iões* dissolvidos na água, maior é a sua *condutividade*.

O facto de a água na natureza, nomeadamente nos rios e albufeiras, conduzir a corrente elétrica é, por exemplo, muito vantajoso para as pessoas que necessitam de monitorizar as comunidades piscícolas. Caso contrário, a amostragem da fauna piscícola, que é muitas vezes realizada através de um método que utiliza eletricidade para capturar os peixes (pesca elétrica), seria totalmente ineficaz [16].

A *condutividade*, além de ser um indicador aproximado do *conteúdo mineral* quando outros métodos não podem ser facilmente utilizados, pode ser medida para determinar a *poluição* existente numa determinada zona, por exemplo em torno de uma descarga de um efluente, ou a extensão da influência das águas de escoamento. Por exemplo a *condutividade* elétrica permite determinar a *salinidade* (dS/m) ou o valor dos *Sólidos Dissolvidos Totais* (SDT - mg/l).

As medições contínuas são particularmente úteis em rios e águas subterrâneas para a gestão das variações temporais nos *sólidos totais dissolvidos* e *íons* principais [137]. Os valores de *condutividade* elétrica da água são utilizados como indicativos da qualidade da água, com sua representação pelo Sistema Internacional em unidades miliSiemens por cm (mS/cm) ou microsiemens por cm (μS/cm). A água da superfície deve em geral deve apresentar uma *condutividade* entre 500 e 1000 μS/cm. A água do mar tem cerca de 55 mS/cm.

Oxigénio Dissolvido

Unidade de medida do SI: [(% de saturação)]. O *oxigénio dissolvido* na água é usado por todas as formas de vida aquática. Assim, este parâmetro é normalmente medido para avaliar a «saúde» das massas de água (rios, lagos, albufeiras). O *oxigénio dissolvido* nas massas de água (rios, lagos e albufeiras) é crucial para os organismos e as criaturas que nelas vivem. Quando a quantidade de *oxigénio dissolvido* desce abaixo dos níveis normais em massas de água, a qualidade da água é prejudicada e os seres vivos presentes, nomeadamente os peixes, podem mesmo morrer [137].

Potencial de Redução de Oxidação

Unidade de medida do SI: [mV]. Ele é indicado em mV. O *potencial de redução de oxidação* (ORP) mede a capacidade de uma massa de água se purificar ou eliminar resíduos. Quando o ORP é alto, há muito oxigénio presente na água, em geral, quanto maior o valor ORP, mais saudável é a massa de água. O ORP diminui com a profundidade. O ORP é tratado de forma separada do *oxigénio dissolvido* porque POR pode fornecer aos investigadores informações adicionais sobre a qualidade da água e o grau de poluição [94].

Turbidez

Unidade de medida do SI: [NTU] (Unidade Nefelométrica de Turbidez). A *turbidez* é uma propriedade física dos fluidos que se traduz na redução da sua *transparência* devido à presença de materiais em suspensão que interferem com a passagem da luz através do fluido. É medida em unidades NTU, que significa Unidade de Turbidez Nefelométrica.

O valor máximo permitido para água tratada é de 1 NTU na saída das estações de tratamento de água e 5 NTU em qualquer ponto da rede de distribuição. Concentrações elevadas de partículas na água aumentam a *turbidez* ou turvação, afetando a penetração da luz e a produtividade, os valores de recreio e a qualidade dos habitats para os organismos aquáticos. Podem também fazer com que as albufeiras e os rios fiquem assoreados (cheios de sedimentos) mais rapidamente, reduzindo e alterando os habitats existentes para as plantas e animais aquáticos.

A turvação excessiva na água potável é esteticamente desagradável, e pode também representar um problema de saúde, já que as partículas fornecem locais de ligação para poluentes, em especial os metais e bactérias (por esta razão, as leituras de turvação podem ser utilizadas como um indicador do potencial de poluição de uma massa de água) [16].

Sólidos totais dissolvidos

Unidade de medida do SI: [mg/l]. São os resíduos existentes na água que são filtráveis. O termo «sólidos» é amplamente utilizado para a maioria dos compostos que estão presentes em águas naturais e permanecem no estado sólido após a evaporação. O tipo e a concentração de resíduos e matéria em suspensão controlam a *turbidez* e a transparência da água. A matéria suspensa consiste em lodo, argila, partículas finas de matéria orgânica e inorgânica, compostos orgânicos solúveis, plâncton e outros organismos microscópicos. Essas partículas variam em tamanho de aproximadamente 10 nm de diâmetro a 0,1 mm de diâmetro, é geralmente aceite que a matéria suspensa é a fração que não vai passar através de um filtro de diâmetro de poro de 0,45 μm [137].

Dióxido de Carbono

Unidade de medida do SI: [mg/l]. O *dióxido de carbono* (DO), CO_2, é outro gás geralmente avaliado em sistemas de monitorização da qualidade da água. Existe um certo equilíbrio entre as concentrações de O_2 e CO_2, as alterações destes gases em corpos de água estão significativamente ligados a reações biológicas. O CO_2 é altamente solúvel em água e o CO_2 atmosférico é absorvido no interface ar-água. O CO_2 dissolvido na água natural faz parte de um equilíbrio envolvendo íons bicarbonato e carbonato. As concentrações destas formas são dependentes em certa medida do *pH*. Em águas não poluídas, o *pH* é controlado principalmente por o equilíbrio entre o *dióxido de carbono*, íons carbonato e bicarbonato, bem como outros compostos naturais como os ácidos húmico e fúlvico (pH < 2) [137].

Sulfato

Unidade de medida: [mg/l]. O *sulfato* encontra-se na água natural em concentrações que variam de alguns a vários milhares de miligramas por litro. O *sulfato* é o íon SO_4^-2, um dos mais abundantes íons na natureza. Surge nas águas subterrâneas através da dissolução de solos e rochas. Nas águas superficiais, ocorre através das descargas de esgotos domésticos e efluentes industriais. Em águas tratadas o *sulfato* é proveniente do emprego de coagulantes [104].

Amónia

Unidade de medida: [mg/l]. A *amónia* (NH_3) é um dos poluentes mais importantes no meio aquático devido a sua natureza altamente tóxica e sua omnipresença nos sistemas de águas superficiais. É descarregado em grandes quantidades pela industria e por águas residuais agrícolas. A *amónia* presente na água provém, geralmente, de processos de degradação de materiais residuais, de origem vegetal ou animal [104].

Nitrato

Unidade de medida do SI: [mg/l]. O *nitrato* (NO_3^-) é o nutriente essencial para organismos autótrofos fotossintéticos e é geralmente encontrado nas águas superficiais. Em baixas concentrações, o *nitrato* é um problema ambiental menos sério, no entanto, quando as concentrações de *nitrato* se tornam excessivas e outros nutrientes estão presentes, a eutrofização e a proliferação de algas associadas pode se tornar um problema. As principais fontes de *nitrato* na água são dejetos humanos e animais, efluentes industriais e o uso de fertilizantes e produtos químicos [104].

Nitrito

Unidade de medida: [mg/l]. O *nitrito* (NO_2^-) é extremamente tóxico para a vida aquática, no entanto, geralmente está presente apenas em pequenas quantidades na maioria sistemas naturais de água doce porque é rapidamente oxidado a *nitrato*. O processo de conversão é afetado por vários fatores, incluindo *pH*, *temperatura* e *oxigénio dissolvido*. Se o *pH* da solução aumentar naturalmente ou pela adição de uma base, a concentração de NH_3 também aumenta [104].

Apêndice II

Pesquisa Bibliográfica

Foi realizada uma pesquisa bibliográfica (até ao inicio de dezembro de 2020) sobre estudos baseados na Monitorização da Qualidade da Água (WQMN's) fazendo uso da IoT (Internet das coisas), com o objetivo de adquirir e aprofundar conhecimento sobre o tema .

A revisão de literatura, também chamada de fundamentação teórica ou revisão bibliográfica, independente de nomenclaturas, deve apresentar revisões críticas da literatura sobre o tema estudado, usando conhecimentos produzidos no passado como base para análise e discussão [100]. O revisor deve apoiar-se em estudos anteriores e utilizá-los como arcabouço teórico para seu próprio trabalho [100].

A revisão foi baseada principalmente em pesquisas publicadas em revistas internacionais ou conferências [11]. Foi realizada pesquisa de literatura na biblioteca do conhecimento *Online* (b-on), *Google Scholar*, ACM *Digital Library* e na plataforma *Web of Science* utilizando combinações diferentes das seguintes cadeias de carateres: "*surface water quality monitoring*", "*sensoring water quality*" e "*water quality iot*". A língua utilizada na pesquisa foi o inglês e português. A distribuição temporal das publicações sobre o tema pesquisado está enquadrado no período entre 2014 a 2020. Em seguida foi realizada a verificação manual de cada um dos documentos alvo, iniciou-se a leitura dos objetivos com o propósito de verificar a proposta do trabalho apresentada, uma vez que nela se resume a sua ideia central. De seguida, foi realizada uma leitura dos resultados da pesquisa.

Foram selecionados 31 trabalhos considerados mais relevantes e que tiravam partido da Internet das Coisas, da utilização de microcontroladores e/ou microcomputadores e sensores, aplicados à monitorização da qualidade da água em águas superficiais. Na seleção, sempre que possível foi tido em conta a origem geográfica do estudo no sentido de diversificar a origem das abordagens.

Apêndice III

Evolução das Soluções - Sensores

Em complemento ao conteúdo apresentado no Capítulo 3, Seção 3.2, apresentam-se detalhes e caraterísticas, dos sensores utilizados com maior frequência nas publicações analisadas.

Um dos primeiros sistemas de monitorização da qualidade da água a tirar partido de uma abordagem *IoT* foi apresentado por Simbeye et al. [115] . O sistema consiste numa rede de sensores sem fios (RSSF) para monitorização da qualidade da água em aquicultura, com nós sensores construídos em torno de um microcontrolador (*Microcontroller unit* - MCU). Os nós sensores comunicam entre si e com um *gateway* um computador através do protocolo ZigBee. Os dados são enviados para um servidor através de um módulo GSM. Os dados recolhidos são analisados e processados ficando disponíveis para utilizadores dos sistema. Na Tabela III.1 podemos verificar as parâmetros monitorizados e respetivos sensores utilizados. Das e Jain [20] fazem uma abordagem semelhante, onde os autores propõem um

Temperatura	pH	Oxigénio dissolvido	Nível da água
DS18B20	PH450G	DO3000	UXI-LY

Tabela III.1: Parâmetros e sensores apresentados em [115]

sistema de baixo custo para monitorização da qualidade da água em «tempo real», para rastrear o nível de poluição nos rios e gerar alertas de acordo com a informação gerada. Na Tabela III.2 podemos verificar as parâmetros monitorizados e respetivos sensores utilizados. Os dados adquiridos dos sensores utilizados são processados por um microcontrolador

Temperatura	pH	Condutividade elétrica
Não identificado	NI*	LM393

* Não identificado

Tabela III.2: Parâmetros e sensores apresentados em [20]

que os transmite utilizando Zigbee entre nós e WiFi ou GSM para enviar dados para um servidor.

Outra abordagem semelhante é seguida por Menon et al. [68] que apresentam uma proposta de baixo custo para o monitorização da qualidade da água em corpos de água naturais, utilizando uma rede ZigBee de unidades Libelium *Waspmote*. Estas unidades são baseadas em microcontroladores e admitem uma vasta gama de sistemas de transmissão de dados: LoraWan, 4G, WiFi e Zigbee. Na Tabela III.3 podemos verificar os parâmetros monitorizados e respetivos sensores utilizados. Prasad et al. [94] descrevem um sistema

Temperatura	pH	Turbidez	Oxigénio dissolvido	Amónia	Anião Nitrato	Anião Sulfato
PT1000	ExStik pH (pH 100)	NI*	YSI 85	Q45N	HYDRA DS Nitrate	Mettler Toledo 3000CS
* Não identificado						

Tabela III.3: Parâmetros e sensores apresentados em [68]

inteligente de monitorização da qualidade da água do mar, descrito e construído também em torno de um módulo *Waspmote*. Os sensores avaliam vários parâmetros físicos ou químicos da água e são controlados pelo módulo *Waspmote* que adquire e processa os sinais dos sensores e os envia para a Internet, através de um módulo GSM e utiliza o protocolo FTP para enviar os dados para um servidor remoto. Na Tabela III.4 podemos verificar os parâmetros monitorizados e respetivos sensores utilizados.

Temperatura	pH	Turbidez	Condutividade elétrica	Potencial de redução de oxidação
PT1000 Libelium	pH Libelium	Turbidity Libelium	Conductivity Libelium	Oxidation-reduction potential Libelium

Tabela III.4: Parâmetros e sensores apresentados em [94]

Daigavane and Gaikwad [38] apresentam um sistema *IoT* de baixo custo para monitorização em «tempo real» da qualidade da água com base em placas Arduino. Vários sensores que medem parâmetros físicos ou químicos da água, são controlados pelo MCU (Atmega328) que se encontra ligado a um servidor *web* através da tecnologia *WiFi*. Os dados são processados pelo sistema e enviados através da Internet. Na tabela III.5 podemos verificar os parâmetros monitorizados e respetivos sensores utilizados. Saravanan

Temperatura	pH	Turbidez	Fluxo
DS18B20	SEN0161	SEN0189	YF-S401

Tabela III.5: Parâmetros e sensores apresentados em [38]

et al. [108] apresentam um sistema para ser utilizado na monitorização de sistemas de

distribuição de água que faz uso do mesmo tipo de placas micro controladoras de baixo custo. Os sensores apresentados na Tabela III.6 monitorizam um conjunto de parâmetros e são controlados por uma placa Arduino. Utiliza um módulo GSM para enviar os dados para armazenamento num servidor *web*. A análise de dados pode ser realizada em «tempo real» a partir de um qualquer navegador da internet. Outros sistemas que fazem uso das mesmas placas de baixo custo são também apresentados por Chowdury et al.[15], Encinas et al. [30]; Hanifah e Supangkat [42], Kamaludin e Ismail [53], Parameswari e Moses [83], Pranata et al. [93], Shafi et al. [111] e Zainuddin et al. [142]. As publicações de Zainuddin et al. [142] e de Hanifah e Supangkat [42] propõem a monitorização dos mesmos parâmetros e sensores, conforme Tabela III.7.

Temperatura	Turbidez	Fl
LM35	WQ770-B	YF-S201

Tabela III.6: Parâmetros e sensores apresentados em [108]

Temperatura	pH	Turbidez	Condutividade	Total Sólidos Dissolvidos	Fluxo
DS18B20	SEN0161	TSD10	NI*	NI*	YF-S201

* Não identificado

Tabela III.7: Parâmetros e sensores apresentados em [42] e [142]

Pranata et al.[93] propõem um sistema *IoT* para monitoração da qualidade da água utilizando uma arquitetura de publicador/subscritor (protocolo MQTT). O sistema é composto por uma rede de nós de retransmissão (publicadores), os sensores apresentados na Tabela III.8 são controlados por um Arduino e comunicam utilizando o protocolo *Zigbee* com um nó *gateway* (subscritor). Os dados também são enviados para um servidor através do protocolo *Zigbee*. Numa plataforma distinta de monitorização da qualidade da água,

Temperatura	pH	Oxigénio Dissolvido
DS18B20	PH SEN0161	SEN-11194

Tabela III.8: Parâmetros e sensores apresentados em [93]

Kamaludin e Ismail [53] apresentaram a implementação de um sistema *IoT* que incorpora identificação por rádio frequência (RFID), uma plataforma de rede de sensores sem fios (RSSF) e o protocolo de Internet (IP). A rede de sensores é controlada por um Arduino e comunica através do protocolo de rede DigiMesh. Na Tabela III.9 podemos verificar os parâmetros monitorizados e respetivos sensores utilizados.

Outro sistema baseado num módulo Arduino é apresentado por Encinas et al. [30]. O sistema é composto por três sensores apresentados na Tabela III.10, os dados recolhidos

Temperatura	pH
LM35	SEN0161

Tabela III.9: Parâmetros e sensores utilizados por [53]

através destes são transmitidos para o módulo Arduino (coordenador de comunicação) através do protocolo *Zigbee*. Por meio de comunicação série os dados são enviados para um módulo transmissor *Xbee* que os faz chegar a um computador através de um módulo receptor Xbee. Os dados são geridos pelo sistema de gestão de base de dados *MySQL* e enviados fazendo uso do protocolo SOAP para a nuvem.

Temperatura	pH	Oxigénio Dissolvido
PT1000	Atlas Scientific	Atlas Scientific

Tabela III.10: Parâmetros e sensores utilizados em [30]

Aspetos críticos relativamente à segurança dos dados de um sistema de *IoT*, de monitorização da qualidade da água são abordados por Parameswari e Moses [83]. Muitos sistemas *IoT* que gerem informação sobre qualidade da água são projetados sem considerar quaisquer aspetos de segurança, originando vulnerabilidades de confidencialidade. Nesta publicação é proposto um processo seguro e moderno, para um sistema de monitoração da qualidade da água, assente numa rede de sensores sem fios (RSSF). É utilizada uma placa Arduino com WiFi, para receber os dados dos sensores listados na Tabela III.11.

Um protótipo para registar os parâmetros de qualidade da água em «tempo real», de

Temperatura	pH	Turbidez	Condutividade
LM35	NI*	TSD10	Vernier Condutivity

* Não identificado

Tabela III.11: Parâmetros e sensores utilizados em [83]

amostras recolhidas de várias fontes, é utilizado por Shafi et al. [111] com o mesmo tipo de placa. Um controlador Arduino adquire sinais de pH, turbidez, temperatura e de fluxo, e envia os dados para a nuvem através de WiFi. Os dados processados podem ser monitorizados remotamente e o fluxo de água pode ser controlado utilizando a solução proposta. A análise preditiva dos dados recolhidos é realizada utilizando algoritmos de *Machine Learning* (ML) aplicados ao desafio de classificar a qualidade da água. Os sensores utilizados nesta publicação não estão identificados.

Chowdury et al. [15], também propõem técnicas de ML. O sistema consiste numa rede de sensores sem (RSSF) com nós controlados por uma placa Arduino, baseada no ATmega2560 MCU. Cada nó inclui um microcontrolador para processamento, um sistema

de comunicações entre nós e vários sensores listados na Tabela III.12. Os dados recolhidos são enviados para um servidor e podem ser visualizados num ecrã. O sistema propõe a integração de *Big Data Analytics (Apache Spark) e Deep Learning (DL)*. Técnicas de *Deep Learning* utilizam geralmente modelos de rede neural convolucional (CNN).

Temperatura	pH	Turbidez	Condutividade	Oxigénio dissolvido	Potencial de redução de oxidação
DS18B20	NI*	NI*	NI*	NI*	NI*

* Não identificado

Tabela III.12: Parâmetros e sensores apresentados em [15]

Um sistema desenvolvido em torno do microcontrolador de baixo consumo energético, ARM LPC1768 é apresentado por Kafli e Isa [52]. Os sensores listados na Tabela III.13 e o sistema de posicionamento global (GPS), monitorizam o ambiente em corpos de água. Os dados recolhidos são enviados para a nuvem através da internet (WiFi) e são analisados utilizando técnicias de *Machine Learning* por um servidor na nuvem (plataforma IBM Watson *IoT*). Um microcontrolador mais rico em recursos, o ARM da família SMT32 é utilizado

Temperatura	pH	Nível	Oxido de Carbono
DHT22	SEN0249	280-WL400	MQ-7

Tabela III.13: Parâmetros e sensores apresentados em [52]

em dois sistemas apresentados por Z. Zhang et al. [144], 2020 e por C. Zhang et al.[143], o primeiro é um sistema de monitorização da qualidade da água que recolhe e armazena valores de alguns parâmetros de qualidade da água utilizada em aquicultura, conforme Tabela III.14. São emitidos alertas em «tempo real» quando os dados estão fora dos parâmetros de segurança. Tira partido do poder de processamento do chip STM32F767 para processar simultaneamente os distintos parâmetros recolhidos. Os dados pré-processados são armazenados num servidor local que pode ser acedido remotamente; o segundo é uma proposta para um sistema de monitorização da qualidade da água realizado em torno de um chip STM32F103. Os dados dos sensores de temperatura, pH e turbidez são adquiridos e pré-processado pelo ARM MCU e transmitidos através de WiFi (chip ESP8266) para um servidor na nuvem para análise e consulta, fornecendo para além da monitorização remota em «tempo real», também estatísticas, alarmes e consulta *online* para terminais móveis. Os sensores utilizados nesta publicação não estão identificados.

Entre as principais razões para a rápida proliferação de sistemas *IoT* está a disponibilidade de *hardware* a baixo custo, a eficiência energética e o leque vasto de funcionalidades. A família de microprocessadores do fabricante *Espressif Systems* apresentam um baixo custo

Temperatura	pH	Oxigénio dissolvido	Amónia	Nitrito
PT1000	PHG-202	RDO-206	NH61-A0002	XT5904

Tabela III.14: Parâmetros e sensores apresentados em [144]

e baixo consumo de energia, oferecendo uma escolha interessante para o desenvolvimento deste tipo de sistemas de monitorização da qualidade e quantidade de água.

Spandana e R. S. Rao [116] desenvolveram um sistema *IoT* de monitorização da qualidade da água, construído em torno de um microprocessador ESP8266, que monitoriza os parâmetros e utiliza os sensores listados na Tabela III.15, enviando os dados pré-processados através de WiFi para um servidor.

Temperatura	pH	Nível	Dióxido de Carbono
LM35	D940010500	Solu clh=sl067-clh	SEN0219

Tabela III.15: Parâmetros e sensores apresentados em [116]

Outro sistema construído em torno de um microcontrolador ESP, neste caso para a monitorização da qualidade da água num canal de rega é proposto por Lameira [56]. Sistema baseado no ESP32 de baixo custo, que integra de origem comunicações Bluetooth e Wi-Fi. No MCU ESP32 podem ser integradas placas LoRa e/ou comunicações GSM e de posicionamento GPS para estender as capacidades de comunicação. São monitorizados os parâmetros e utilizados os sensores que constam na Tabela III.16. Os dados são comunicados entre os nós através de LoRa e são enviados pelo nó *gateway* para a Internet usando GSM.

Temperatura	pH	Total Sólidos Dissolvidos	Fluxo
DS18B20	E-201-C (PH-4502C)	Gravity Analog TDS	HC-SR04

Tabela III.16: Parâmetros e sensores apresentados em [56]

Um equipamento muito presente nesta área são os computadores de baixo custo, de placa única (*Single Board Computer*-SBC). Estes dispositivos estão capacitados para executar um sistema operativo e são fáceis de utilizar [51], fornecendo alternativas interessantes para o desenvolvimento de sistemas de monitorização da qualidade da água no âmbito da *IoT*.

Um sistema de monitorização de qualidade de água de uso geral desenvolvido em torno de um SBC é apresentado por Vijayakumar e Ramya [130]. É composto por vários sensores que medem parâmetros físicos ou químicos da água, incluindo temperatura, pH, turbidez, condutividade elétrica e oxigénio dissolvido, controlados por um Raspberry Pi (RPiB +).

O SBC adquire e processa os sinais dos sensores e envia os dados processados para a Internet. Os sensores utilizados nesta publicação não estão identificados.

Raju e Varma [99] também utilizaram um Raspberry Pi (RPi3) para adquirir e processar os sinais de temperatura, pH, oxigénio dissolvido, condutividade elétrica, amónia, nitrato e carbonato. Os dados processados são enviados por WiFi para um servidor local conectado à nuvem. Os sensores utilizados nesta publicação também não se encontram identificados.

Uma abordagem semelhante é seguida por Budiarti et al. [9] num sistema de monitorização de gestão ambiental e de água, utilizando sensores padrão de monitorização da qualidade da água que integram a sonda multiparâmetro YSI - Modelo 600R (temperatura, oxigénio dissolvido, condutividade, salinidade, condutância específica, pH e sólidos dissolvidos totais). A sonda está conectada a um Raspberry Pi via RS232 e o SBC é conectado por WiFi a um router/modem que conecta o sistema à Internet. Ratnam et al. [103] apresentam um protótipo para um sistema de monitorização e controlo da qualidade da água, dedicado à aquicultura, é desenvolvido em torno de um Raspberry Pi. São monitorizados os parâmetros e utilizados os sensores apresentados na TabelaIII.17, os sinais destes são adquiridos e controlados pelo SBC e por um sistema de bombas, também controlado pelo SBC, que sempre que exista necessidade evacua as águas residuais e reabastece com água doce. Os dados são enviados para a Internet através de WiFi.

Temperatura	pH
LM35	SEN0161

Tabela III.17: Parâmetros e sensores apresentados em [103]

Gao et al. [39] propõem uma outra solução no domínio da aquicultura que visa ajudar os piscicultores a controlar e gerir de forma inteligente a qualidade da água. Os dados recolhidos são transmitidos usando a tecnologia de LoRa e submetidos para processamento inteligente num microcontrolador. A informação resultante é transmitida para um servidor remoto através de um módulo GPRS e armazenada numa base de dados, ficando disponível para consulta. Na Tabela III.18 podemos verificar os parâmetros monitorizados e respetivos sensores utilizados.

Temperatura	pH	Turbidez	Condutividade	Total Sólido Dissolvidos	Potencial de redução de oxidação
DS18B20	E-201-C	NI*	NI*	YHT-8402	NI*

* Não identificado

Tabela III.18: Parâmetros e sensores apresentados em [39]

Outra proposta que utiliza um SBC é descrita por Salunke e Kate [105]. A plataforma sugerida é a Intel Galileo Gen 2 (atualmente descontinuada) com os sensores apresentados

na Tabela III.19.

pH	Turbidez	Nível
NI*	TSD10	NI*

* Não identificado

Tabela III.19: Parâmetros e sensores apresentados em [105].

A combinação de placas MCU ricas em recursos, de baixo custo e muito baixo consumo energético, com computadores de placa única (SBC's) proporciona outra arquitetura interessante para o desenvolvimento de sistemas de monitorização da qualidade da água: uma rede de sensores sem fios (RSSF) com nós de deteção baseados em MCU's/SBC's, que controlam os sensores, adquirem os seus sinais, executam pré-processamento de dados e comunicam com sistemas agregadores baseados em SBC. Estes processam e analisam posteriormente os dados e enviam os resultados para um servidor remoto na nuvem para novo tratamento/processamento.

Uma plataforma *IoT* com múltiplos nós de sensores móveis (MSN) para a avaliação da qualidade espaço-temporal de águas superficiais é apresentada por T. Li et al. [57]. Cada MSN é composto por cinco sensores (temperatura, pH, oxigénio dissolvido, condutividade elétrica e potencial de redução de oxidação) controlada por um MCU (ATmega1281 + RPi3) e montado num veiculo operado remotamente. Os nós transmitem os dados recolhidos através de WiFi ou Zigbee para uma estação base (PC) que está conectada por GSM, 3G ou 4G à Internet. Os sensores utilizados nesta publicação não estão identificados.

O projeto e implementação de um sistema *IoT* de monitorização da qualidade da água para a criação de caranguejo, com o objetivo de auxiliar o agricultor para manter níveis aceitáveis de qualidade de água, é descrito pot Niswar et al. [77]. O sistema MQTT, consiste em nós sensor que atuam como publicadores, um SBC (Raspberry Pi 3) atuando como *broker* e dispositivos móveis do cliente como subscritores. Os nós sensores são construídos com MCU (ATmega2560), por interface de comunicações LoRa e sensores de qualidade da água apresentados na Tabela III.20. O sistema de monitorização permite o acesso remoto aos níveis de qualidade da água através de nós UI (Node-Red Dashboard).

Temperatura	pH	Condutividade
PT1000	SEN0161	NI*

* Não identificado

Tabela III.20: Parâmetros e sensores apresentados em [77]

O sistema proposto por Martínez et al. [63] consiste numa rede de nós de sensores sem fios (RSSF) implementada com uma combinação de ARM MCUs (Kinetis K66) com um SBC (RPiZero) que controlam um sistema portátil, EcoSens Aquamonitrix [27] , de cromatografia de íons que permite detecção direta *in situ* de nitrito e nitrato em águas

naturais, com uma boa relação custo-benefício. O artigo apresenta a integração desta rede de sensores sem fios num sistema *IoT* completo que fornece mecanismos preventivos e analíticos de dados para apoiar tomando uma decisão.

Outra classe de sistemas é baseada em *Field Programmable Gate Arrays* (FPGA) para processamento e comunicações. A principal vantagem deste tipo de sistema é a possibilidade de proceder a reconfigurações, porém, para sensores genéricos e protocolos padrão, o custo geralmente mais alto deste tipo de sistema e a necessidade de ferramentas e capacidades especiais de programação não justificam seu uso no desenvolvimento de sistemas de monitorização da qualidade da água.

Condutividade	Nível	Pressão
Campbell Scientific CS547A	MaxBotix MB7384	Solinst 3001 Levelogger® Edge

Tabela III.21: Parâmetros e sensores apresentados em [138]

O sistema apresentado por Wong e Kerkez [138] é programável em C e o FPGA (*Field Programmable Gate Arrays*) apresenta um ARM-Cortex M3 de ultra-baixa potência e um módulo de comunicações móveis para a conectividade com a Internet, além de diversos ADC e amplificadores. O sistema possui, conforme Tabela III.21, uma sonda ultrassónica de profundidade e um sensor de pressão para medir o nível de água e acionar um amostrador industrial de qualidade da água também com base na leitura de um sensor de condutividade.

O sistema descrito por Myint et al. [74] utiliza os sensores apresentados na Tabela III.22. O protocolo Zigbee é implementado no núcleo FPGA para adicionar capacidade de comunicação nesta tecnologia. O sistema apresentado por Zin et al. [14] é semelhante ao

Temperatura	pH	Condutividade	Nível	Dióxido de Carbono
DS18B20	Atlas scientific pH (EZO pH Circuit)	SEN0189	LV-MaxSonar-EZ1	SEN0219

Tabela III.22: Parâmetros e sensores apresentados em [74]

anterior em termos de sensores III.23 e comunicações.

Temperatura	pH	Turbidez	Nível	Dióxido de Carbono
DS18B20	SEN0161	SEN0189	LV-MAXSONAR-EZ1	SEN0219

Tabela III.23: Parâmetros e sensores apresentados em [14]

Embora não haja universalmente reconhecidos e aceites, métodos ou padrões internacionais para a seleção de parâmetros da qualidade da água a incluir obrigatoriamente, quantos mais parâmetros de qualidade da água forem selecionados para monitorização,

Parâmetros monitorizados	*Nº Pub*	*%*	*Ident.*	*%*	*N Ident.*	*%*
T	29	94%	23	79%	6	21%
pH	28	90%	19	68%	9	32%
Tu	15	48%	9	60%	6	40%
EC	12	39%	5	42%	7	58%
DO	11	35%	7	64%	4	36%
WL	8	26%	6	75%	2	25%
Fl	5	16%	4	80%	1	20%
CO2	3	10%	3	100%	0	0%
NH3	3	10%	2	67%	1	33%
NO3-	3	10%	2	67%	1	33%
ORP	3	10%	1	33%	2	67%
TDS	2	6%	1	50%	1	50%
NO-2	2	6%	2	100%	0	0%
p	1	3%	1	100%	0	0%
CO	1	3%	1	100%	0	0%
SO42-	1	3%	1	100%	0	0%

Tabela III.24: Parâmetros monitorizados nas publicações selecionadas

maior será o custo e a carga de trabalho para os sistemas a implementar e ao mesmo tempo corre-se o risco de diluir alguns fatores principais/dominantes da qualidade da água [23].

Conforme podemos verificar na Tabela III.24 e no artigo [23], os parâmetros selecionados para monitorização da qualidade água, referenciados com maior frequência no conjunto das 31 publicações utilizadas neste artigo, incluem: temperatura, pH, turbidez, condutividade, oxigénio dissolvido, fluxo e nível da água. Podemos ainda aferir na publicação da Comissão Europeia «*Review of sensors to monitor water quality*» [120] que os 10 principais parâmetros monitorizados *online* desde água bruta até água utilizada para consumo humano, com base na percentagem de monitorização para cada parâmetro, em diferentes países do Mundo, Estados Unidos da América, Bélgica, Países Baixos, Reino Unido e Austrália, é fortemente coincidente com as incidências evidenciadas na Tabela III.24 e com a proposta de variáveis disponíveis e confiáveis, adequadas para incorporação em sistemas automatizados de monitorização ambiental e da qualidade da água, Tabela 2.2.

Na Figura 3.1 podemos observar uma síntese dos sensores selecionados nas 31 publicações, para a monitorização de cada um dos 16 parâmetros da qualidade da água. Dos 127 sensores utilizados, não foi possível identificar 40 (31%). Dos 87 (69%) sensores identificados 79% dizem respeito a apenas 5 (temperatura, pH, turbidez, condutividade, oxigénio dissolvido e nível da água) dos 16 parâmetros analisados. O sensor de temperatura da água apresentado na grande maioria das publicações selecionadas (34%), é eleito o DS18B20. O SEN0161 é o sensor de pH apresentado na grande maioria das publicações selecionadas

(25%). Os sensores TSD10 e SEN0189 destacam-se nas escolhas efetuadas para medir a turbidez. Os dois representam 57% das escolhas e apresentam caraterísticas muito semelhantes. O SEN0219 foi o sensor escolhido por unanimidade para a monitorização do dióxido de carbono.

O sensor de fluxo YF-S201 foi o eleito em 40% das escolhas.

Para análise dos restantes parâmetros, condutividade, oxigénio dissolvido, nível, amónia, nitrato, nitrito, potencial de redução de oxidação e total de sólidos dissolvidos, não se verificou a repetição da escolha de qualquer sensor.

No Seção seguinte III, em complemento e simplificação do conteúdo da imagem 3.1 podemos visualizar todos os parâmetros e respetivos sensores agrupados por função.

Sensores utilizados agrupados por função

Em complemento e simplificação do conteúdo da imagem 3.1 onde podemos visualizar todos os parâmetros e respetivos sensores, aqui podemos ter uma vista dos mesmos agrupados por função.

Sensores (T) utilizados	Nº Ref.	%
DS18B20	10	34%
LM35	5	17%
PT1000	5	17%
DTH22	1	3%
HTU21D-F	1	3%
YSI - Model 600R	1	3%
Não Identificados	6	21%

Tabela III.25: Sensores de Temperatura utilizados

Sensores (pH) utilizados	Nº Pub	%
SEN0161	7	25%
E-201-C	2	7%
AtlasScientific pH Sensor	2	7%
D940010500 (garden)	1	4%
ExStik pH (pH 100)	1	4%
PH SEN0161	1	4%
pH sensor Libelium	1	4%
PH450G	1	4%
PHG-202	1	4%
SEN0249	1	4%
YSI - Model 600R	1	4%
Não Identificados	9	32%

Tabela III.26: Sensores de pH utilizados

Sensores (Tu) utilizados	Nº Pub	%
TSD10	4	27%
SEN0189	3	20%
Turbidity sensor Libelium	1	7%
WQ770-B	1	7%
Não Identificados	6	40%

Tabela III.27: Sensores de Turbidez utilizados

Sensores (EC) utilizados	Nº Pub	%
YSI - Model 600R	1	8%
Vernier Condutivity	1	8%
LM393 Moisture Sensor	1	8%
Campbell Scientific CS547A	1	8%
Conductivity Sensor Libelium	1	8%
Não Identificados	7	58%

Tabela III.28: Sensores de Condutividade Elétrica utilizados

Sensores (DO) utilizados	Nº Pub	%
DO3000	1	9%
RDO-206	1	9%
YHT-8402	1	9%
YSI - Model 600R	1	9%
Atlas Scientific Dissolved Oxygen	1	9%
SEN-11194	1	9%
YSI Model 85	1	9%
Não Identificados	4	36%

Tabela III.29: Sensores de Oxigénio Dissolvido utilizados

Sensores (WL) utilizados	Nº Pub	%
LV-MAXSONAR-EZ1	2	25%
Solu clh=sl067-clh	1	13%
Transmissor de nível de pressão UXI-LY	1	13%
280-WL400	1	13%
MaxBotix MB7384	1	13%
Não Identificados	2	25%

Tabela III.30: Sensores de Nível utilizados

Sensores (FI) utilizados	Nº Pub	%
YF-S201	2	40%
HC-SR04	1	20%
YF-S401	1	20%
Não Identificados	1	20%

Tabela III.31: Sensores de Fluxo utilizados

No conjunto das publicações revistas, podemos ainda encontrar:

- 3 Sensores de CO_2 utilizados – 3 SEN0219;
- 3 Sensores de NH_3 utilizados – 1 Q45N, 1 NH61-A0002 e 1 não identificado;
- 3 Sensores de NO_3^- utilizados – 1 EcoSens Aquamonitrix, 1 HYDRA-DS Nitrate Analyzer e 1 não identificado;
- 3 Sensores de ORP utilizados – 1 Oxidation-reduction potential sensor e 2 não identificados;
- 2 Sensores de TDS utilizados – 1 Gravity Analog TDSsensor e 1 não identificado;
- 2 Sensores de NO_2^- utilizados – 1 XT5904TDS e 1 EcoSens Aquamonitrix;
- 1 Sensores de Pressão (p) utilizado – 1 Solinst 3001 Levelogger® Edge;
- 1 Sensores de Monóxido de Carbono – 1 MQ-7;
- 1 Sensores de Sulfato SO_4^{2-} – 1 Mettler Toledo 3000CS.

Apêndice IV

Evolução das Soluções - Micro-controladores

Em complemento ao conteúdo apresentado no Capítulo 3, Seção 3.3, apresentam-se mais alguns detalhes e caraterísticas, dos MCU e SBC, utilizados com maior frequência nas publicações analisadas.

Conforme referido no Apêndice III, Seção III, das propostas estudadas no contexto do tema desta dissertação, um dos primeiros sistemas de monitorização da qualidade da água a tirar partido de uma abordagem *IoT* foi apresentado por Simbeye et al. em 2014 [115]. Este sistema apresentou um conjunto de nós sensores construídos em torno de um de uma placa Arduíno Uno equipada com um microcontrolador ATmega8A/16L. Pranata et al. [93], Kamaludin e Ismail [53], Encinas et al. [30] e Daigavane and Gaikwad [38], em 2017, Parameswari e Moses [83] e Saravanan et al. [108], em 2018, Hanifah e Supangkat [42] e Zainuddin et al. [142], em 2019, propuseram a placa Arduíno Mega com o microcontrolador ATmega328 para gerir sistemas diversos de monitorização da qualidade da água. Este microcontrolador oferece melhorias em relação ao ATmega8A/16L, nomeadamente, quadruplica o tamanho do programa e duplica o tamanho das memórias SDRAM e EEPROM, e das portas PWM.
Prasad et al. [94] em 2016, T. Li et al. [57] e Menon et al. [68] em 2017, propuseram a utilização da placa Waspmote equipada com o microcontrolador Atmega1281. Este microcontrolador oferece melhorias em relação ao ATmega328, nomeadamente 32 pinos para 54, memória flash dos 32 Kb para 128 Kb, EEPROM de 1Kb para 4 Kb, RAM de 1Kb para 8 Kb. Outra placa Arduíno, esta com o microcontrolador ATmega2560 foi proposta por Niswar et al. [77] e Shafi et al. em 2018 [111], e por Chowdury et al. em 2019 [15], este microcontrolador oferece melhorias em relação ao ATmega1281, nomeadamente 54 pinos para 100, memória flash dos 128 Kb para 256 Kb e o número de canais ADC de 8 para 16.

Um microcontrolador mais poderoso em recursos, o ARM da família SMT32, foi utili-

zado em dois sistemas apresentados em 2020, por Z. Zhang et al. [144] e por C. Zhang et al. [143]. A extensa família de microcontroladores SMT32, de uma forma geral, superam as caraterísticas do ATmega2560, nomeadamente o oscilador interno do STM32 que é de 32MHz enquanto o do ATmega2560 é de 16MHz, a memória flash do STM32 é de 512KB, e a do ATmega2560 é de 256KB.

Dois sistemas desenvolvidos em torno de microcontroladores de baixo consumo energético, o primeiro, o ARM LPC1768, foi apresentado por Kafli e Isa em 2018 [52] e o segundo, o ARM LPC 2148 foi apresentado por Das and Jain (2017) [20]. Ambos da família ARM, mas com processadores distintos. O LPC1768 (*Ultra-low power RTC*) baseado no processador ARM Cortex-M3, apresenta frequências superiores a 120 MHz e o LPC2148 (*Low power RTC*) baseado no processador ARM7 TDMI, apresentando frequências na ordem dos 60 MHz.

Spandana e R. S. Rao em 2018 [116] e Lameira em 2019 [56], propõem microcontroladores de baixo custo e baixo consumo de energia, da família de microprocessadores do fabricante *Espressif Systems*. O primeiro apresentou um sistema de monitorização da qualidade da água, construído em torno de um microprocessador ESP8266; o segundo apresentou um sistema construído em torno de um microcontrolador ESP32. O ESP8266 já está equipado com WIFI embutido no próprio chip e com um preço muito acessível. Com o *know-how* tecnológico adquirido e com a garantia do sucesso do ESP8266, a Espressif lançou em 2016, o ESP32, outro Microcontrolador, ao qual adicionou Bluetooth ao WIFI que já disponibilizava anteriormente. O excelente desempenho do microcontrolador é alcançado devido à sua estrutura *dual core*, um microprocessador Tensilica Xtensa LX6 de arquitetura Harvard [62].

O primeiro sistema de monitorização da qualidade água, estudado no contexto do tema desta dissertação, que utiliza um SBC é proposto por Vijayakumar e Ramya (2015) [130], utiliza um Raspberry Pi (RPi 1 B +) para adquirir os dados dos sensores, para processar, comunicar e armazenar os dados. Ratnam et al. em 2019 [103], propõem o mesmo SBC. O sistema proposto por Martínez et al. (2020) [63], utiliza um Raspberry Pi Zero (RPiZero), combinado com a placa Teensy 3.6 ARM MCUs (Kinetis K66). O RPI Zero pouco se diferencia do RPi 1 B+, partilham o mesmo processador e memória. As principais diferenças são no tipo e número de portas, por exemplo, o RPi 1 B+ possui 2 portas USB (RPi Zero, 1 micro USB), 1 porta rede Ethernet (RPi Zero, não possui). Raju e Varma em 2017 [99] e Budiarti et al. em 2019 [9], apresentaram soluções utilizando um Raspberry 3 (RPi 3). Este SBC possui um poder computacional maior, assente num processador *Quadcore* ARM Cortex-A53 (RPi 1 B+, *Single core* ARM Cortex-A7), com o dobro da memória, 1 GB , e disponibiliza rede sem fios, Blueetooth e BLE.

Salunke e Kate em 2017 [105], utilizaram a plataforma Intel Galileo Gen 2 (atualmente descontinuada) como solução para aquisição, processamento e comunicação de dados. Esta placa é compatível com o ambiente de desenvolvimento de programas com o Arduíno. Oferece caraterísticas muito inferiores ao RPi 2 B. Principais caraterísticas deste SBC: processador Intel Pentium 400 MHz (RPi 2 B, ARM Cortex 900 MHz), cache 16 KB (RPi 2 B, 512 KB), 512 KB RAM (RPi 2 B, 1 GB).

A combinação de placas MCU ricas em recursos, de baixo custo e muito baixo consumo energético, com computadores de placa única (SBC's) proporciona outra arquitetura interessante para o desenvolvimento de sistemas de monitorização da qualidade da água: uma rede de sensores sem fios (RSSF) com nós de deteção baseados em MCU's/SBC's, que controlam os sensores, adquirem os seus sinais, executam pré-processamento de dados e comunicam com sistemas agregadores baseados em SBC. Estes processam e analisam posteriormente os dados e enviam os resultados para um servidor remoto na nuvem para novo tratamento/processamento. Tal como Raju e Varma em 2017 [99] e Budiarti et al. em 2019 [9], também T. Li et al. em 2017 [57] e Niswar et al. [77] em 2019, fazem uso do SBC Raspberry 3, mas ainda assim utilizam o microcontrolador Atmega1281 e Atmega2560 respetivamente, para fazer a aquisição dos dados sensores, libertando os Raspberry 3 para o processamento e comunicação dos dados.

Outra classe de sistemas é baseada em *Field Programmable Gate Arrays* (FPGA) para processamento e comunicações. FPGA é mais indicado para aplicações em que se exija processamento paralelo e altas velocidades de processamento, pode ser utilizado em controlar a saída de conversores estáticos empregando a tecnologia de comutação, na qual um PWM é gerado para controlar a atuação do comutador, para se atingir uma tensão desejada visando diminuir as variações de saída [82]). A principal vantagem deste tipo de sistema é a possibilidade de proceder a reconfigurações, porém, para sensores genéricos e protocolos padrão, o custo geralmente mais alto deste tipo de sistema e a necessidade de ferramentas e capacidades especiais de programação não justificam seu uso no desenvolvimento de sistemas de monitorização da qualidade da água. O sistema apresentado por por Wong e Kerkez em 2016 [138], para a monitorização da qualidade da água, utiliza uma plataforma FPGA (*Field Programmable Gate Arrays*) Cypress PSoC5LP, que utiliza um processador ARM-Cortex M3 de ultra-baixa potência.

Outra placa FPGA, Altera DE1-SoC, é utilizada por Myint et al. (2017) [74]. Usa um processador Arm Cortex-A9 32 bits Dual-Core (1GB DDR/SDRAM 64).

Zin et al. em 2019 [14], também apresentaram uma placa FPGA, com o processador Nios II (RISC 32 bits/32 MBytes DDR SDRAM/1 MByte síncrona SRAM/16 MBytes Intel P30/P33 flash).

Os sistemas FPGA são programados com base em descrição de *hardware*, utilizando uma linguagem de descrição de *hardware* (HDL), estas são: VHSIC *hardware description language*; (VHDL) e Verilog. O ambiente HDL fornece o resumo da utilização lógica do controlador. A simulação pode ser feita num Simulador Modelsim [82].

Apêndice V

Evolução das Soluções - Transmissão de Dados

Em complemento ao conteúdo apresentado no Capítulo 3, Seção 3.4, apresentam-se mais alguns detalhes e caraterísticas, das tecnologias de transmissão de dados, utilizadas nas publicações analisadas.

Na abordagem *IoT* apresentado por Simbeye et al. em 2014 [115], a primeira no âmbito do estudo desta tese de dissertação, os nós sensores comunicam entre si e com um *gateway* conetado a um computador através do protocolo ZigBee. Os dados são enviados para um servidor através de um módulo GSM.

Em 29% dos sistemas estudados, T. Li et al. [57], Pranata et al. [93], Encinas et al. [30], Das and Jain [20], Myint et al. [75] e Menon et al. [68], em 2017, Zainuddin et al. [142] e Zin et al. [14], em 2019, também utilizaram a tecnologia ZigBee para transmitir os dados referentes à qualidade da água a partir de um MCU. Esta tecnolologia enquadra-se nas redes de curto alcance (WPAN). Padrão de comunicação IEEE 802.15.4. Principais caraterísticas de uso: distâncias de cerca de 300m em linha de vista e cerca de 75-100m entre obstáculos, opera nas frequências 915MHz/2.4 GHz, taxa de transferências entre 20 e 250 Kbps, consumindo muito pouca energia [44].

Tal como Simbeye et al. em 2014 [115], prefazendo 45% dos sistemas estudados, também Prasad et al. em 2015 [94], Wong and Kerkez em 2016 [138], T. Li et al. [57], Das and Jain [20] e Menon et al. [68], em 2017, Spandana and R. S. Rao [116], Saravanan et al. [108], em 2018, Budiarti et al. [9], Gao et al. [39], Hanifah and Supangkat [42], Zainuddin et al. [142] e Zin et al. [14], em 2019, Lameira [56] e Martínez et al. em 2020 [63], utilizaram a tecnologia GSM para transmitir os dados a longa distância.

A tecnologia GSM, padronizado pelo 3GPP (3rd Generation Partnership Project), enquadra-se nas redes de longo alcance (WAN), permite taxas de transferência de cerca de

64 Kbps, opera na frequência dos 30 KHz/200 KHz e oferece uma cobertura de cerca de 30 Km [117].

Foram apresentadas soluções por Menon et al. em 2017 [68], Niswar et al. em 2018 [77], Gao et al. em 2019 [39] e Lameira em 2020 [56], que fazem uso da tecnologia LoRa. Os dois primeiros utilizam a tecnologia para transmitir os dados para servidores, os últimos dois utilizam a mesma tecnologia apenas para comunicar entre nós sensores, elegendo a tecnologia GSM para a transmissão dos dados a longa distância.

Lora (abreviatura de Long-Range) é uma tecnologia de radio frequência que permite a comunicação sem fios a longas distâncias, enquadra-se nas redes de longo alcance (Low Power Wireless Area Network – LPWAN). Principais caraterísticas de uso: baixo consumo energético, alcance de distâncias superiores a 10 Km, implementado sobre a norma IEE802.15.4, taxas de transferência que variam entre os 0.3 e 50Kbps e opera na frequência 868 MHz (Europa). Comparativamente à tecnologia GSM apresenta como vantagens o facto se ser uma tecnologia de uso livre e de apresentar baixíssimo consumo energético. Como desvantagem, a necessidade de um investimento inicial superior ao investimento necessário para implementar a tecnologia GSM [3].

A grande maioria das soluções estudadas utiliza a tecnologia WiFi, nomeadamente como complemento de outras tecnologias, quer para estabelecer ligação a servidor *web*, quer para enviar dados para a nuvem. Esta tecnologia foi utilizada por Prasad et al. [94] e Vijayakumar and Ramya [130], em 2015, Raju and Varma [99], Encinas et al. [30], T. Li et al. [57], Kamaludin and Ismail [53], Salunke and Kate [105], Das and Jain [20], Pranata et al. [93] e Menon et al. [68], Daigavane and Gaikwad [38], Kafli and Isa [52], em 2017, Saravanan et al. [108], Niswar et al. [77], Parameswari and Moses [83], Shafi et al. [111] e Spandana and R. S. Rao [116], em 2018, Zainuddin et al. [142], Ratnam et al. [103] e Chowdury et al. [15], em 2019, e C. Zhang et al. [143] em 2020.

A tecnologia WiFi é uma tecnologia de comunicação sem fios, que utiliza ondas de rádio (RF) para permitir a comunicação entre dispositivos. Esta tecnologia é normalmente utilizada para interconectar routers da Internet com outros dispositivos. Enquadra-se nas redes de locais sem fios (Wireless Lan Area Network – WLAN) e assenta nos padrões IEEE 802.11. Principais caraterísticas de uso: pode utilizar as frequência rádio de 2,4 GHz e 5 GHz, taxas de transferência entre 65 Mbps a 450 Mbps (802.11n), alcance de cerca de 100m, consumo alto de energia [135].

Encontramos ainda a utilização de outras tipos comunicações, com fios, Budiarti et al., 2019 [9] utilizaram o protocolo comunicação série RS232 para ligar a sonda YSI600R, Z. Zhang et al. em 2020 [144] conecta os sensores ao MCU através do protocolo série RS485,

Wong and Kerkez em 2016 [138] utilizaram o interface digital serial assíncrona SDI-12 para conectar o sensor de pressão. Kamaludin and Ismail em 2017 [53] utilizaram a tecnologia DigiMesh para a comunicação entre sensores.

Nas tecnologias de comunicações sugeridas nos sistemas utilizados destacam-se, conforme Figura V.1, as tecnologias que não são de baixo consumo, WiFi/Ethernet e GSM, o que sugere que a sua utilização se deva principalmente à sua disponibilidade e facilidade de utilização.

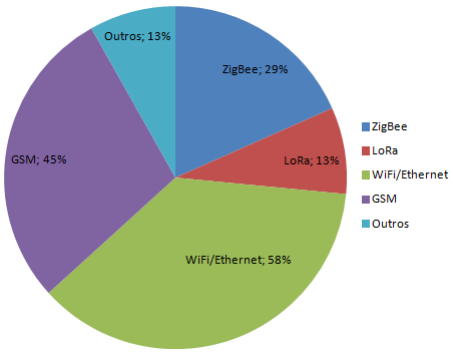


Figura V.1: Tecnologias e Protocolos de comunicação (*hardware*)

Apêndice VI

Soluções Comerciais

Atualmente existem muitas soluções comerciais baseadas em RSSF para monitorização remota de diferentes parâmetros de qualidade da água. Identificam-se, neste contexto, soluções mais complexas e flexíveis e outras mais simples. Estas soluções, de modo geral, consistem em elementos de hardware específico para o sensoriamento, processamento e transmissão dos dados com recurso ao módulos de comunicações de diferentes tecnologias.

Soluções comerciais de monitorização da qualidade da água:

Netilion Smart System

O *Netilion Smart System* é uma solução da empresa *Endress+Hauser*, para monitorizar parâmetros de qualidade de águas superficiais, Figura VI.1. Permite medir constantemente a qualidade da água em rios e lagos ou qualquer outro tipo de água de superfície. Consiste em dois componentes: uma |*Box*, um pacote individual contendo o equipamento de medição para instalar nos pontos de medição relevantes e uma *Smart Systems App*, uma aplicação para *smartphone* com a qual se pode consultar os valores das medições em «tempo real». Disponibiliza também uma aplicação *web*, *Netilion Value*, para acesso via computador. Este sistema permite medir os seguintes parâmetros: Temperatura, pH, Oxigénio dissolvido e Condutividade [76].

BlueBox System

O *BlueBox* é uma solução da empresa *GO Systemelektronik GmbH*, composta por módulos que permitem dar resposta a diferentes necessidades ao longo do tempo, Figura VI.2. Permite visualizar *online* e armazenar remotamente os valores das medições. Possibilidade de ligação a mais de 200 sensores e atuadores. Utiliza o espectrómetro ISA *In-situ*, que é uma ferramenta para a detecção em linha de parâmetros de qualidade da água. As aplicações são várias nas quais se incluem o controlo de qualidade de água potável, gestão do saneamento, controlo de parâmetros de qualidade no tratamento de águas residuais, bem como

Figura VI.1: Netilion Smart System

Fonte: (https://netilion.endress.com/pt/smart-systems/surface-water)

a monitorização ambiental. Ao mesmo conversor *BlueBox* podem ligar-se outros sensores, permitindo a medição de um a grande variedade de parâmetros [19]. Para armazenamento remoto, os dados são transmitidos através da tecnologia *Ethernet*.

SWARM Buoys

A bóia *SWARM* é um sistema de monitorização da qualidade da água em «tempo real», desenvolvido pela empresa *Veolia*, Figura VI.3. Para dar resposta a esta necessidade de monitorização e controlo do estado das águas superficiais, a solução consiste em bóias equipadas com soluções de medição, processamento e comunicação da informação recolhida em «tempo real», de simples operacionalização e manutenção e passíveis de serem utilizadas em diversos locais e necessidades, o que a torna também uma opção economicamente atrativa. As bóias podem ser instaladas diretamente e facilmente em qualquer massa de água

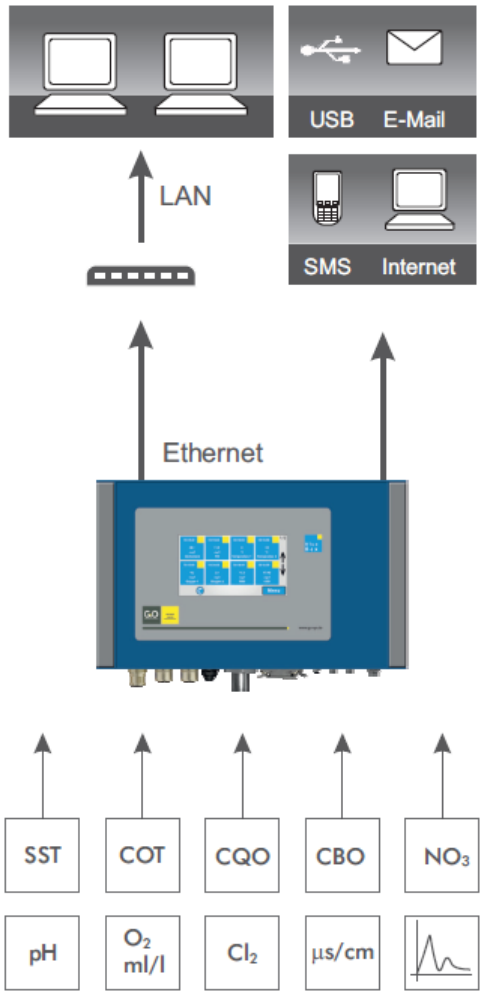


Figura VI.2: BlueBox System

Fonte: (https://www.contimetra.com/Conteudos_F/IA/capitulos/folhetos/go/01_go.pdf)

superficial ou curso de água. A solução permite medir vários parâmetros essenciais para a monitorização da qualidade dos recursos hídricos: condutividade, temperatura, velocidade, profundidade, oxigénio dissolvido, turvação e carga orgânica. Equipadas com um sistema GPRS ou de comunicação rádio, as bóias *SWARM* são auto-suficientes energeticamente graças a uma combinação da energia solar e eólica. Possui sondas de multi-parâmetros, pode medir condutividade, temperatura, fluxo, profundidade, oxigénio dissolvido, turbidez, matéria orgânica e pH. Em simultâneo, pode operar até 6 sensores de 6 tipos diferentes. Os sensores podem ser trocados dependendo do uso da bóia, e a mesma sonda pode ser usada várias vezes na mesma bóia para duplicação de medição, e também em várias profundidades [118].

Figura VI.3: SWARM Buoys

Fonte: (https://www.veolia.com/anz/swarmbuoy)

WM-1020

O sistema de monitorização da qualidade da água multiparâmetros, *WM-1020*, é um produto desenvolvido pela empresa *Yantai Winmore*, Figura VI.4. Permite monitorizar parâmetros de qualidade da água, tais como pH, oxigénio dissolvido, condutividade, salinidade, turbidez, TSS, COD, etc., e é amplamente utilizado em aquacultura, monitorização da qualidade da água para consumo humano, monitorização da qualidade de águas superficiais, monitorização do ambiente marinho e outras indústrias. Possui uma saída digital e analógica isolada de quatro canais: o instrumento fornece interface de comunicação RS485, e utiliza o protocolo de comunicação MODBUS. Possui um ecrã LCD táctil onde exibe os vários parâmetros. Pode-se adicionar até 199 sensores digitais de qualidade da água, e a medição de parâmetros personalizados de qualidade não aquosa está disponível (por exemplo: pressão, nível de líquido, fluxo, etc., os sensores têm de suportar o protocolo de comunicação 485). Em caso de necessidade, pode-se integrar no sistema um router GPS/GPRS, para a transmissão remota de dados [81].

SmartWater Libelium

Desenvolvido e comercializado pela empresa *Libelium*, o *Waspmote Smart Water* é um produto destinado à monitorização da qualidade da água em rios, lagos, piscinas, mar, entre outros, Figura VI.5. Equipado com vários sensores que medem vários parâmetros

Figura VI.4: WM-1020

Fonte: (https://www.winmoreltd.com/product/621/)

de qualidade da água, o *Waspmote Smart Water* é uma plataforma de monitorização da qualidade de água que permite apresentar nós autónomos, que transmitem remotamente os dados adquiridos. Esta plataforma utiliza o microcontrolador *Waspmote* produzido pela *Libelium*, que foi especificamente desenhado para baixo consumo de energia. Waspmote possui várias formas de comunicação e placas de sensores, podendo ser usado em várias aplicações. Os parâmetros de qualidade da água que podem ser medidos, incluem: pH, oxigénio dissolvido (DO), potencial de redução de oxidação (ORP), condutividade (salinidade), turbidez, temperatura, etc. Em termos de conetividade, o *Waspmote* pode utilizar as comunicações rádio: 3G, 4G, 802.15.4, Zigbee, 868 MHz, 900 MHz, WiFi, 4G, Sigfox e LoRaWAN, para enviar informações para a nuvem ou servidores remotos [131].

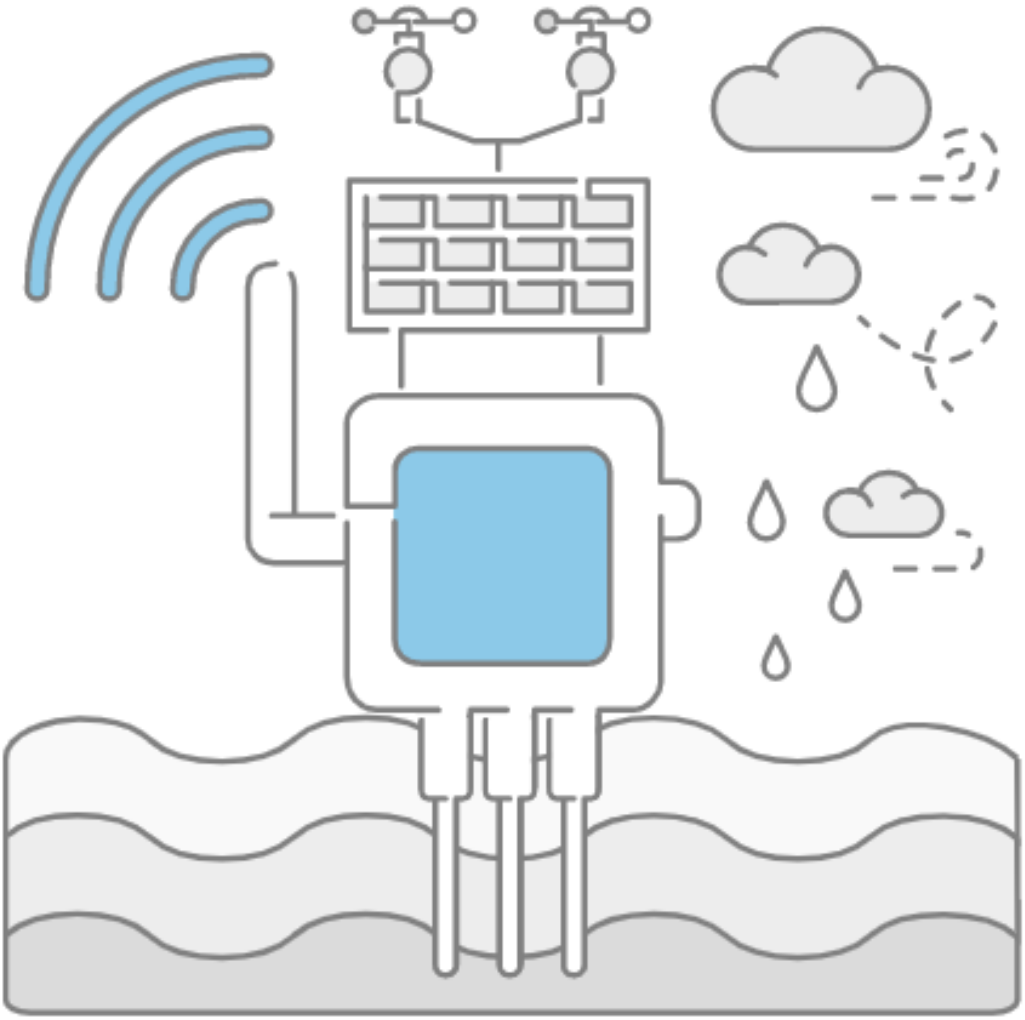


Figura VI.5: WM-1020

Fonte: (http://www.libelium.com/

Apêndice VII

Hardware

A Tabela VII.1 lista todos os equipamentos e dispositivos utilizados para satisfazer a arquitetura apresentada no Capítulo 4.

Equipamentos e Dispositivos
Sonda de pH #ENV-40-pH [5]
Sonda de ORP #ENV-40-ORP [5]
Sonda de ORP #ENV-40-ORP [5]
Sonda de DOX #ENV-40-DOX [5]
Sonda de EC #ENV-40-EC-K1.0 [5]
Sonda de Temperatura PT-1000 [5]
Modulo EZO pH ENV-40 [5]
Modulo EZO ORP ENV-40 [5]
Modulo EZO DOX ENV-40 [5]
Modulo EZO EC ENV-40 [5]
Modulo EZO RDT Temperature Circuit [5]
Estação Meteorológica Steinberg Systems SBS-WS-600 WiFi [32]
Dispositivo RTL-SDR [28]
MCU LILYGO TTGO T-Beam LoRa [58]
SBC Raspberry Pi [121]
Cartão de memória
LORIX One [95]
Router Mikrotik wAP LTE [69]
Cartão 4G
Baterias
Servidor

Tabela VII.1: Equipamentos e dispositivos utilizados na sensorização, transmissão e gestão de dados do sistema de monitorização de qualidade da água.

O *Módulo de Qualidade da Água* é um dos *«end devices»* do sistema de monitorização de qualidade da água (Figura 4.2). É constituído por um *MCU LILYGO TTGO T-Beam LoRa* equipado com rádio *LoRa*, um *kit* de sensores (sondas e módulos de interface

entre as sondas e o *MCU*) analógicos da empresa Atlas Scientific, que foram instalados em locais preparados para o efeito. Faz a aquisição dos parâmetros físicos e químicos mais comuns, de qualidade da água, nomeadamente: *Temperatura*, *pH*, *ORP*, *DO* e *EC*. Conforme é apresentado na Secção 3.2 as sondas deste fabricantesão as que de uma forma geral apresentam melhores caraterísticas. Destaca-se a capacidade para manter a

Figura VII.1: Exemplo de uma sonda do kit utilizado (Sonda pH Probe ENV-40-pH)

Fonte:https://atlas-scientific.com/files/pH_probe.pdf [6]

calibração no mínimo até um ano, em alguns casos até 10 anos, e poderem ser utilizados em profundidades, de uma forma geral superiores a 70 metros [5].

Também da empresa Atlas Scientific, os conetores BNC (*Bayonet Neill–Concelman*) e placas de circuito EZOs (exemplo de um diagrama de ligação: Figura VII.2) são utilizados para conectar as sondas. Estas placas recebem os sinais dos sensores e através de circuitos integrados convertem o sinal para a unidade correspondente de cada sensor. As placas de circuitos EZOs permitem fazer calibrações das sondas e armazenam informações de compensação.

Ambos os módulos permitem a utilização dos protocolos série UART e I^2C. O subsistema de recolha de dados de qualidade da água comunica os dados ao *MCU* através do protocolo de comunicação em série, i_2C.

O protocolo LoRaWAN foi utilizado para recolher os dados das sondas e transmitir periodicamente as leituras efetuadas no corpo de água, para a *gateway LORIX One*. Foram analisadas diversas plataformas computacionais de desenvolvimento, baseadas em diferentes microcontroladores. Optou-se pela utilização da placa de desenvolvimento *MCU LILYGO TTGO T-Beam LoRa* [58] (Figura [58]) baseada no micro-controlador ESP32. Disponibiliza comunicações WiFi e Bluetooth. Dispõe ainda de um chip SX1276 com para comunicação LoRa 868MHz de alta sensibilidade ($>$ 148dBm) e 20dBm de potência de saída, permitindo comunicações de longa distância (até 2Km).

O outro *«end device»* do sistema de monitorização de qualidade da água é o conjunto formado por um *SBC*, rádio *LoRa* e pelos sensores da estação meteorológica (Figura 4.2).

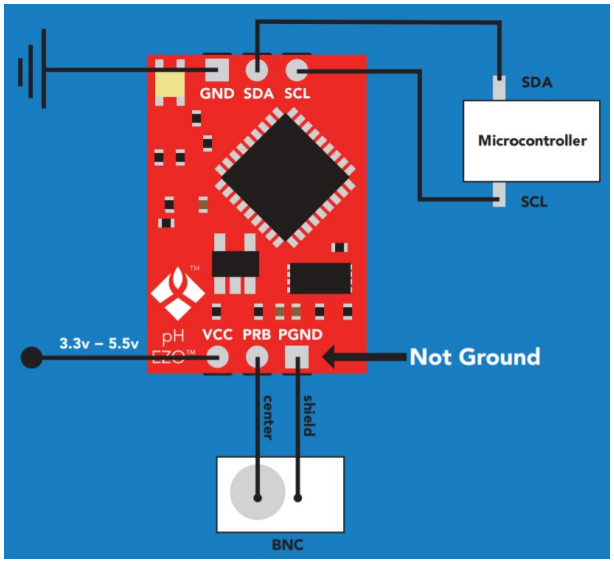


Figura VII.2: Diagrama de ligação I2C - EZO pH Circuit

Fonte:https://atlas-scientific.com/files/ezo-ph-wiringdiagram.pdf

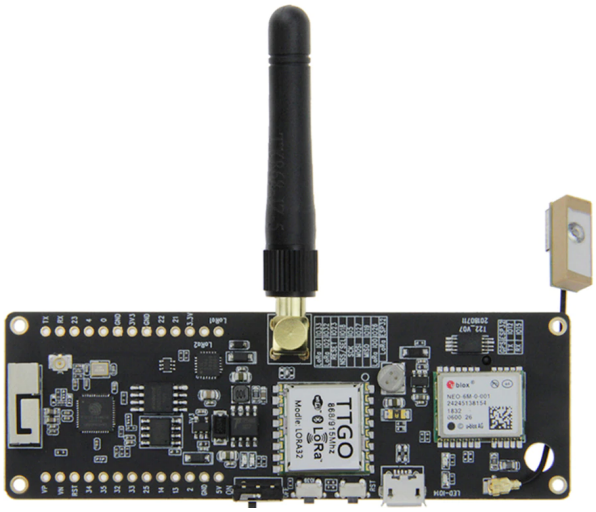


Figura VII.3: Microprocessador LILYGO TTGO T-Beam LoRa

Fonte:http://www.lilygo.cn/

A *Estação Meteorológica* (Figura VII.4) fornece dados do ambiente/atmosfera circundante, nomeadamente: temperatura, humidade e pressão do ar, precipitação, velocidade e direcção do vento, luminosidade e radiação ultra-violeta(UV). Transmite dados na frequência a 868MHz. Esta transmissão é realizada por uma unidade RF, que é interceptada e descodificada por um dispositivo RTL-SDR e *software* instalado num SBC equipado com um LoRa Hat de 868MHz, os dados recebidos no SBC são transmitidos via tecnologia LoRa para a Gateway LoRa, Figura 5.6. Ambos os módulos de aquisição de dados entregam dados

Figura VII.4: Estação Meteorológica - Steinberg Systems SBS-WS-600 WiFi

à *gateway* através da tecnologia *LoRa* que os transmite dados através da rede *LoRaWAN* utilizando os componentes da pilha do *ChirpStack Network Server*.

A *gateway* utilizada, a *Lorix One*, apresenta as seguinte especificações (Tabela VII.2):

O sistema operativo que utiliza ...

Figura VII.5: Gateway

Tabela VII.2: Especificações de hardware da gateway

CPU	ARM® Cortex-A5 600MHz
RAM	128MBytes DDR2 200MHZ
Armazenamento	512MBytes flash e Slot SD-CARD
Conectividade	Rádio LoRa RF versão 863-870 MHz
Frequências LoRa suportadas	868 MHz (EU), 915 MHz (US), 923 MHz (AS)
Ethernet	10/100 Mbps com POE
USB	Sim
Tensão elétrica	PoE 24V passive
Temperatura tolerada em funcionamento	-30°C a +55°C
Humidade relativa tolerada	20% a 90%

Apêndice VIII

Instalação do Software da Pilha «LoRaWAN Network Server»

Este Apêndice inclui as instruções necessárias para a instalação do *software* da pilha de servidores de rede ChirpStack.

Copiar o repositório *AquaQ2*, executar o comando: *git clone gitolite@sepsi.ipbeja.pt:aquaq2*

Procedimento de Instalação do *stack* para *docker

Seguir as instruções contidas no ficheiro *readme.md*, instruções: 1, 2 e 3.

```
1  # ficheiro: readme.md
2
3  # O meu repositório AquaQ2
4  # Procedimento de Instalação do *stack* para *docker*
5  ## Numa Máquina Ubuntu
6
7  O procedimento tem de garantir que o *Docker* não é na versão *snap*.
8    1. remover o componente que dá problemas
9     **sudo snap remove docker**
10   2. instalar o gestor de pacotes visual
11    **sudo apt install aptitude**
12   3. instalar docker-compose e docker.io a partir do aptitude ou por
       outro meio
13    **sudo apt install docker.io**
14    **sudo apt install docker-compose**
15   4. mudar para a diretoria do *portainer*
16    **cd aquaq2/docker/portainer**
17   5. lançar o *portainer*
18    **docker-compose up -d**
19   6. ir para aquaq2/docker/aquaq2/qgis-server
```

```
 7. criar a imagem do *docker* designada por *qgis-server*
  **./b.sh**
 8. mudar para a pasta **aqua2/docker/aquaq2**
 9. construir o sistema de máquinas incluindo *Flask*
  **docker-compose build**
 10. criar as pastas permanentes dos volumes
  **sudo ./criarsistemaficheiros.sh**
 11. criar os volumes do *docker*
  **./criarvolumes.sh**
 12. lançar todos os *contentores*
  **docker-compose up -d**
```

Instalação da Aplicação Portainer

Seguir as instruções contidas no ficheiro *readme.md*, instruções: 4 e 5.

```
# ficheiro: docker-compose.yml

version: "3.7"
services:
  # sistema de controlo de contentores
  portainer:
    image: portainer/portainer-ce:latest
    restart: always
    ports:
      - 8000:8000
      - 9000:9000
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - portainer_data:/data portainer/portainer-ce
  nginxproxymanager:
    image: 'jc21/nginx-proxy-manager:latest'
    restart: always
    ports:
      # Public HTTP Port:
      - '33080:80'
      # Public HTTPS Port:
      - '33443:443'
      # Admin Web Port:
      - '33081:81'
    environment:
      # These are the settings to access your db
      DB_MYSQL_HOST: "db"
      DB_MYSQL_PORT: 3306
      DB_MYSQL_USER: "npm"
      DB_MYSQL_PASSWORD: "npm"
```

```
31        DB_MYSQL_NAME: "npm"
32        # If you would rather use Sqlite uncomment this
33        # and remove all DB_MYSQL_* lines above
34        # DB_SQLITE_FILE: "/data/database.sqlite"
35        # Uncomment this if IPv6 is not enabled on your host
36        # DISABLE_IPV6: 'true'
37      volumes:
38        - ./data:/data
39        - ./letsencrypt:/etc/letsencrypt
40      depends_on:
41        - db
42    db:
43      image: 'jc21/mariadb-aria:latest'
44      restart: always
45      environment:
46        MYSQL_ROOT_PASSWORD: 'npm'
47        MYSQL_DATABASE: 'npm'
48        MYSQL_USER: 'npm'
49        MYSQL_PASSWORD: 'npm'
50      volumes:
51        - ./data/mysql:/var/lib/mysql
52
53  volumes:
54    portainer_data:
```

Criar imagem qgis-server

Seguir as instruções contidas no ficheiro *readme.md*, instruções: 6 e 7.

```
1 # ficheiro: b.sh
2
3 #!/usr/bin/env sh
4 docker build -f Dockerfile -t qgis-server ./
```

```
1 # ficheiro: $Dockerfile$
2 FROM debian:buster-slim
3
4 ENV LANG=en_EN.UTF-8
5
6 RUN apt-get update \
7     && apt-get install --no-install-recommends --no-install-suggests
      --allow-unauthenticated -y \
8         gnupg \
9         ca-certificates \
```

```
10          wget \
11          locales \
12      && localedef -i en_US -f UTF-8 en_US.UTF-8 \
13      # Add the current key for package downloading - As the key changes
         every year at least
14      # Please refer to QGIS install documentation and replace it with
        the latest one
15      && wget -O - https://qgis.org/downloads/qgis-2020.gpg.key | gpg --
        import \
16      && gpg --export --armor F7E06F06199EF2F2 | apt-key add - \
17      && echo "deb http://qgis.org/debian buster main" >> /etc/apt/
        sources.list.d/qgis.list \
18      && apt-get update \
19      && apt-get install --no-install-recommends --no-install-suggests
        --allow-unauthenticated -y \
20          qgis-server \
21          spawn-fcgi \
22          xauth \
23          xvfb \
24      && apt-get remove --purge -y \
25          gnupg \
26          wget \
27      && rm -rf /var/lib/apt/lists/*
28
29 RUN useradd -m qgis
30
31 ENV TINI_VERSION v0.17.0
32 ADD https://github.com/krallin/tini/releases/download/${TINI_VERSION}/
       tini /tini
33 RUN chmod +x /tini
34
35 ENV QGIS_PREFIX_PATH /usr
36 ENV QGIS_SERVER_LOG_STDERR 1
37 ENV QGIS_SERVER_LOG_LEVEL 2
38
39 COPY cmd.sh /home/qgis/cmd.sh
40 RUN chmod -R 777 /home/qgis/cmd.sh
41 RUN chown qgis:qgis /home/qgis/cmd.sh
42
43 USER qgis
44 WORKDIR /home/qgis
45
46 ENTRYPOINT ["/tini", "--"]
47
48 CMD ["/home/qgis/cmd.sh"]
```

```
1 # ficheiro $cmd$
2 #!/bin/bash
3 [[ $DEBUG == "1" ]] && env
```

```
exec /usr/bin/xvfb-run --auto-servernum --server-num=1 /usr/bin/spawn-fcgi -p 5555 -n -d /home/qgis -- /usr/lib/cgi-bin/qgis_mapserv.fcgi
```

Construir o sistema de máquinas

Seguir as instruções contidas no ficheiro *readme.md*, instruções: 8 e 9.

```
# ficheiro: docker-compose.yml
# Aquaq2
#
# stack para o sistema AquaQ2
#
# os portos deste sistema encontram-se na gama 30000-40000
# com a exceção do sistema de gestão dos contentores
#   * portainer
#       - 8000:8000
#       - 9000:9000
#   * postgis (postgresql)
#       - 35432:5432
#   * adminer
#       - 35081:8080
#   * eclipse-mosquitto
#       - 31883:1883
#   * node-red
#       - 31880:1880
#   * nginx
#       - 32080:8080
#   * flask
#       - 39080:80
# a pasta de dados do sistema encontra-se em
# /var/local/aquaq2
#
version: "3.7"

services:
  # chirpstack
  chirpstack-network-server:
    image: chirpstack/chirpstack-network-server:3
    restart: always
    volumes:
      - ./configuration/chirpstack-network-server:/etc/chirpstack-network-server

  chirpstack-application-server:
    image: chirpstack/chirpstack-application-server:3
```

```
38      restart: always
39      ports:
40        - 38080:8080
41      volumes:
42        - ./configuration/chirpstack-application-server:/etc/chirpstack-
        application-server
43
44    chirpstack-gateway-bridge:
45      image: chirpstack/chirpstack-gateway-bridge:3
46      restart: always
47      ports:
48        - 1700:1700/udp
49      volumes:
50        - ./configuration/chirpstack-gateway-bridge:/etc/chirpstack-
        gateway-bridge
51
52    chirpstack-geolocation-server:
53      image: chirpstack/chirpstack-geolocation-server:3
54      restart: always
55      volumes:
56        - ./configuration/chirpstack-geolocation-server:/etc/chirpstack-
        geolocation-server
57
58    postgresql:
59      image: postgres:9.6-alpine
60      restart: always
61      ports:
62        - 36432:5432
63      environment:
64        - POSTGRES_PASSWORD=aquaq2barba2021
65      volumes:
66        - ./configuration/postgresql/initdb:/docker-entrypoint-initdb.d
67        - postgresqldata:/var/lib/postgresql/data
68
69    # base de dados postgres com postgis
70    postgis:
71      image: postgis/postgis
72      restart: always
73      environment:
74        - POSTGRES_PASSWORD=aquaq2barba2021
75      ports:
76        - 35432:5432
77      volumes:
78        - postgisdata:/var/lib/postgresql/data
79
80    # administração web de bases de dados
81    adminer:
82      image: adminer
83      restart: always
```

```
84      ports:
85        - 35081:8080
86
87    redis:
88      image: redis:5-alpine
89      restart: always
90      volumes:
91        - redisdata:/data
92
93    # servidor MQTT
94    mosquitto:
95      image: eclipse-mosquitto:2
96      restart: always
97      ports:
98        - 31883:1883
99      volumes:
100       - ./configuration/eclipse-mosquitto/mosquitto.conf:/mosquitto/
        config/mosquitto.conf
101       - eclipse-mosquitto:/mosquitto/data
102
103   # servidor Node RED
104   node-red:
105     image: nodered/node-red:latest
106     restart: always
107     environment:
108       - TZ=Europe/Lisbon
109     ports:
110       - 31880:1880
111     networks:
112       - node-red-net
113     volumes:
114       - node-red-data:/data
115
116   # flask
117   # https://github.com/tiangolo/meinheld-gunicorn-flask-docker
118   meinheld:
119     build: ./configuration/meinheld-gunicorn-flask
120     restart: always
121     ports:
122       - 39080:80
123     volumes:
124       - meinheld-app:/app
125
126   # QGIS (servidor de mapas)
127   qgis-server:
128     # Should use version with utf-8 locale support:
129     image: qgis-server:latest
130     restart: always
131     volumes:
```

```
132       - qgis-data:/data:ro
133     environment:
134       - LANG=pt_PT.UTF-8
135       - QGIS_PROJECT_FILE=/data/osm.qgs
136       - QGIS_SERVER_LOG_LEVEL=0  # INFO (log all requests)
137       - DEBUG=1                  # display env before spawning QGIS
        Server
138
139   # web server com nginx
140   nginx:
141     image: nginx
142     restart: always
143     volumes:
144       - ./configuration/qgis/nginx.conf:/etc/nginx/conf.d/default.conf
        :ro
145       - nginx-html:/usr/share/nginx/html
146     ports:
147       - 32080:80
148     environment:
149       - NGINX_HOST=fauno.informatik.pt
150       - NGINX_PORT=80
151     depends_on:
152       - qgis-server
153
154   # aplicação escrita em Python para armazenamento dos dados
155   # na base de dados
156   #copiadados:
157   #  image: copiadados
158   #  restart: always
159
160 volumes:
161   postgresqldata:
162     external: true
163   postgisdata:
164     external: true
165   redisdata:
166   eclipse-mosquitto:
167     external: true
168   node-red-data:
169     external: true
170   nginx-html:
171     external: true
172   meinheld-app:
173     external: true
174   qgis-data:
175     external: true
176
177 networks:
178   node-red-net:
```

Instruções de configuração

As informações de configuração de servidores e aplicações encontram-se na pasta:
aqua2/docker/aquaq2/configuration
Estes ficheiros são utilizados pelo *docker-compose* VIII.

chirpstack-application-server

Ficheiro: *chirpstack-application-server.toml*

```
# See https://www.chirpstack.io/application-server/install/config/ for a full
# configuration example and documentation.

[postgresql]
dsn="postgres://chirpstack_as:chirpstack_as@postgresql/chirpstack_as?sslmode=disable"

[redis]
url="redis://redis:6379"

[application_server.integration.mqtt]
server="tcp://mosquitto:1883"
username="sensor"
password="sensor$$$2020"

[application_server.api]
public_host="chirpstack-application-server:8001"

[application_server.external_api]
bind="0.0.0.0:8080"
jwt_secret="verysecret"
```

chirpstack-gateway-bridge

Ficheiro: *chirpstack-gateway-bridge.toml*

```
# See https://www.chirpstack.io/gateway-bridge/install/config/ for a full
# configuration example and documentation.

[integration.mqtt.auth.generic]
servers=["tcp://mosquitto:1883"]
```

```
username="sensor"
password="sensor$$$2020"
```

chirpstack-geolocation-server

Ficheiro: *chirpstack-geolocation-server.toml*

```
# See https://www.chirpstack.io/geolocation-server/install/config/ for a full
# configuration example and documentation.

[geo_server.backend]
name="collos"

[geo_server.backend.collos]
# Collos subscription key.
#
# This key can be retrieved after creating a Collos account at:
# http://preview.collos.org/
subscription_key=""
```

chirpstack-network-server

Ficheiro: *chirpstack-network-server.toml*

```
# See https://www.chirpstack.io/network-server/install/config/ for a full
# configuration example and documentation.
#
# This file is for the EU868 band. See the examples/ folder for more
# configuration examples.

[postgresql]
dsn="postgres://chirpstack_ns:chirpstack_ns@postgresql/chirpstack_ns?sslmode=disable"

[redis]
url="redis://redis:6379"

[network_server]
net_id="000000"

[network_server.band]
name="EU868"

[network_server.network_settings]
```

```
20
21    [[network_server.network_settings.extra_channels]]
22    frequency=867100000
23    min_dr=0
24    max_dr=5
25
26    [[network_server.network_settings.extra_channels]]
27    frequency=867300000
28    min_dr=0
29    max_dr=5
30
31    [[network_server.network_settings.extra_channels]]
32    frequency=867500000
33    min_dr=0
34    max_dr=5
35
36    [[network_server.network_settings.extra_channels]]
37    frequency=867700000
38    min_dr=0
39    max_dr=5
40
41    [[network_server.network_settings.extra_channels]]
42    frequency=867900000
43    min_dr=0
44    max_dr=5
45
46 [network_server.gateway.backend.mqtt]
47 server="tcp://mosquitto:1883"
48 username="sensor"
49 password="sensor$$$2020"
50
51 [join_server.default]
52 server="http://chirpstack-application-server:8003"
53
54 [geolocation_server]
55 server="chirpstack-geolocation-server:8005"
```

eclipse-mosquitto

Ficheiro: *mosquitto.conf*

```
1 listener 1883
2 # os utilizadores tem de ter password
3 allow_anonymous false
4 password_file /mosquitto/data/passwd
5 require_certificate false
```

meinheld-gunicorn-flask

Ficheiro: *Dockerfile*

```
FROM tiangolo/meinheld-gunicorn-flask:python3.7

COPY ./app /app
```

Ficheiro: *./app/main.py*

```
FROM tiangolo/meinheld-gunicorn-flask:python3.7

COPY ./app /app
```

nginx

Ficheiro: *nginx.conf*

```
server {
  listen 80;
  server_name _;
  location / {
    root  /usr/share/nginx/html;
    index index.html index.htm;
  }
  location /qgis-server {
    proxy_buffers 16 16k;
    proxy_buffer_size 16k;
    gzip off;
    include fastcgi_params;
    fastcgi_pass qgis-server:5555;
  }
}
```

postgresql/initdb

Ficheiro: *001-init-chirpstack_ns.sh*

```
#!/bin/bash
set -e

psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
    create role chirpstack_ns with login password 'chirpstack_ns';
    create database chirpstack_ns with owner chirpstack_ns;
EOSQL
```

Ficheiro: *002-init-chirpstack_as.sh*

```
#!/bin/bash
set -e
```

```
3
4 psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
5     create role chirpstack_as with login password 'chirpstack_as';
6     create database chirpstack_as with owner chirpstack_as;
7 EOSQL
```

Ficheiro: *003-chirpstack_as_trgm.sh*

```
1 #!/bin/bash
2 set -e
3
4 psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" --dbname="
      chirpstack_as" <<-EOSQL
5     create extension pg_trgm;
6 EOSQL
```

Ficheiro: *004-chirpstack_as_hstore.sh*

```
1 #!/bin/bash
2 set -e
3
4 psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" --dbname="
      chirpstack_as" <<-EOSQL
5     create extension hstore;
6 EOSQL
```

qgis

Ficheiro: *nginx.conf*

```
1  server {
2    listen 80;
3    server_name _;
4    location / {
5      root   /usr/share/nginx/html;
6      index index.html index.htm;
7    }
8    location /qgis-server {
9      proxy_buffers 16 16k;
10     proxy_buffer_size 16k;
11     gzip off;
12     include fastcgi_params;
13     fastcgi_pass qgis-server:5555;
14   }
15 }
```

Ficheiros: *./data/"imagens.qgs"*

```
1 imagens de teste .qgs
```

html

Ficheiro: *index.html*

```
<h1>Hello World</h1>
<h2>Here I am</h2>
```

Ficheiro: *./app/__init__.py*

```
from flask import Flask
app = Flask(__name__)
from app import views
```

Ficheiro: *./app/main.py*

```
from app import app
```

Ficheiro: *./app/requirements.txt*

```
Flask==1.1.2
```

Ficheiro: *./app/uwsgi.ini*

```
[uwsgi]
module = main
callable = app
master = true
```

Ficheiro: *./app/views.py*

```
from app import app

@app.route('/')
def home():
    return "hello world!"
```

Imagem e Programa de Armazenamento de Dados

As informações de configuração de servidores e aplicações encontram-se na pasta:
aqua2/docker/aquaq2/copia-dados
Estes ficheiros são utilizados pelo *Dockerfile*, listado em baixo.

Seguir as instruções contidas no ficheiro *readme.md*, instruções: 8 e 9.

Ficheiro: *Dockerfile*

```
FROM python:3

ENV LANGUAGE=pt_PT.UTF-8
ENV LC_ALL=pt_PT.UTF-8
ENV LANG=pt_PT.UTF-8

```

```
WORKDIR /usr/src/app

RUN apt-get update \
    && apt-get install --no-install-recommends --no-install-suggests
    --allow-unauthenticated -y \
    apt-utils\
    gnupg \
    ca-certificates \
    wget \
    locales \
    && localedef -i pt_PT -f UTF-8 pt_PT.UTF-8 \
    && apt-get update

COPY requirements.txt ./

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD [ "python", "./main.py" ]
```

Ficheiro: *requirements.txt*

```
GeoAlchemy2==0.8.4
greenlet==1.0.0
paho-mqtt==1.5.1
psycopg2==2.8.6
SQLAlchemy==1.4.1
```

Criar as Pastas Permanentes dos Volumes

Seguir as instruções contidas no ficheiro *readme.md*, instrução: 10.

Ficheiro: *criarsistemaficheiros.sh*

```
#!/usr/bin/env sh
# criarsistemaficheiros.sh
# Aquaq2
#
# cria as pastas para os volumes do stack aquaq2 para Docker
#
sudo mkdir /var/local/aquaq2

########## postgis ##########
sudo mkdir /var/local/aquaq2/postgis/
sudo mkdir /var/local/aquaq2/postgis/data

########## postgresql para chirpstack ##########
sudo mkdir /var/local/aquaq2/postgresql
```

```
15 sudo mkdir /var/local/aquaq2/postgresql/data
16
17 # ########## Mosquitto ##########
18 sudo mkdir /var/local/aquaq2/eclipse-mosquitto
19 sudo mkdir /var/local/aquaq2/eclipse-mosquitto/data
20
21 # cópia da password de acesso ao MQTT
22 sudo cp ./configuration/eclipse-mosquitto/passwd /var/local/aquaq2/
       eclipse-mosquitto/data/passwd
23
24 # ########## Node RED ##########
25 sudo mkdir /var/local/aquaq2/node-red
26 sudo mkdir /var/local/aquaq2/node-red/data
27 sudo chmod ugo+rw /var/local/aquaq2/node-red/data
28
29 # ########## NGINX ##########
30 sudo mkdir /var/local/aquaq2/nginx
31 sudo mkdir /var/local/aquaq2/nginx/html
32
33 # ########## QGIS ##########
34 sudo mkdir /var/local/aquaq2/qgis
35 sudo mkdir /var/local/aquaq2/qgis/data
36 sudo cp ./configuration/qgis/data/* /var/local/aquaq2/qgis/data
```

Criar os Volumes do Docker

Seguir as instruções contidas no ficheiro *readme.md*, instrução: 11.

Ficheiro: *criarvolumes.sh*

```
1  #!/usr/bin/env sh
2  # criarvolumes.sh
3  # José Jasnau Caeiro
4  #
5  # cria os volumes do stack aquaq2 para Docker
6  #
7  docker volume create --driver local \
8         --opt type=btrfs \
9         --opt device=/var/local/aquaq2/postgis/data \
10        --opt o=bind postgisdata
11 docker volume create --driver local \
12        --opt type=btrfs \
13        --opt device=/var/local/aquaq2/postgresql/data \
14        --opt o=bind postgresqldata
15 docker volume create --driver local \
16        --opt type=btrfs \
17        --opt device=/var/local/aquaq2/eclipse-mosquitto/data \
18        --opt o=bind eclipse-mosquitto
```

```
19 docker volume create --driver local \
20        --opt type=btrfs \
21        --opt device=/var/local/aquaq2/node-red/data \
22        --opt o=bind node-red-data
23 docker volume create --driver local \
24        --opt type=btrfs \
25        --opt device=/var/local/aquaq2/nginx/html \
26        --opt o=bind nginx-html
27 docker volume create --driver local \
28        --opt type=btrfs \
29        --opt device=/var/local/aquaq2/meinheld/app \
30        --opt o=bind meinheld-app
31 docker volume create --driver local \
32        --opt type=btrfs \
33        --opt device=/var/local/aquaq2/qgis/data \
34        --opt o=bind qgis-data
```

Lançar todos os Contentores

Seguir as instruções contidas no ficheiro *readme.md*, instrução: 12.
12. lançar todos os *contentores*
executar o comando:
> docker-compose up -d

Apêndice IX

Base de Dados e Ferramentas de Gestão de SGBD

Este Apêndice é um complemento à Secção 5.3.

Criar Base de Dados

Para criar a base de dados, *PostGIS*, utilizada para armazenar a informação recolhida do módulos de qualidade da água e da estação meteorológica, são realizadas as seguintes instruções:

1. entrar na shell do contentor onde reside o PostGIS: *docker exec -it <nome_ do_ contentor> /bin/bash*;
2. No servidor (contentor) *PostGIS*, entrar no sistema: *psql -U <utilizador>*;
3. criar a base de dados: *create database <nome_ da_ base_ de_ dados with owner <utilizador_ propriétario_ da_ bd>*;
4. estabelecer ligação à base de dados: `\c` *<base_ de_ dados> <utilizador>*;
    - *CREATE EXTENSION POSTGIS*;
    - *CREATE EXTENSION postgis_ topology*;
    - *CREATE SCHEMA postgis*;

Criar Tabelas e Relações

Ilustramos aqui as tabelas e relações, que compõem a base de dados *aquaq2*.

A tabela *waterbody* armazena o nome dos corpos de água escolhidos para a aquisição de dados.

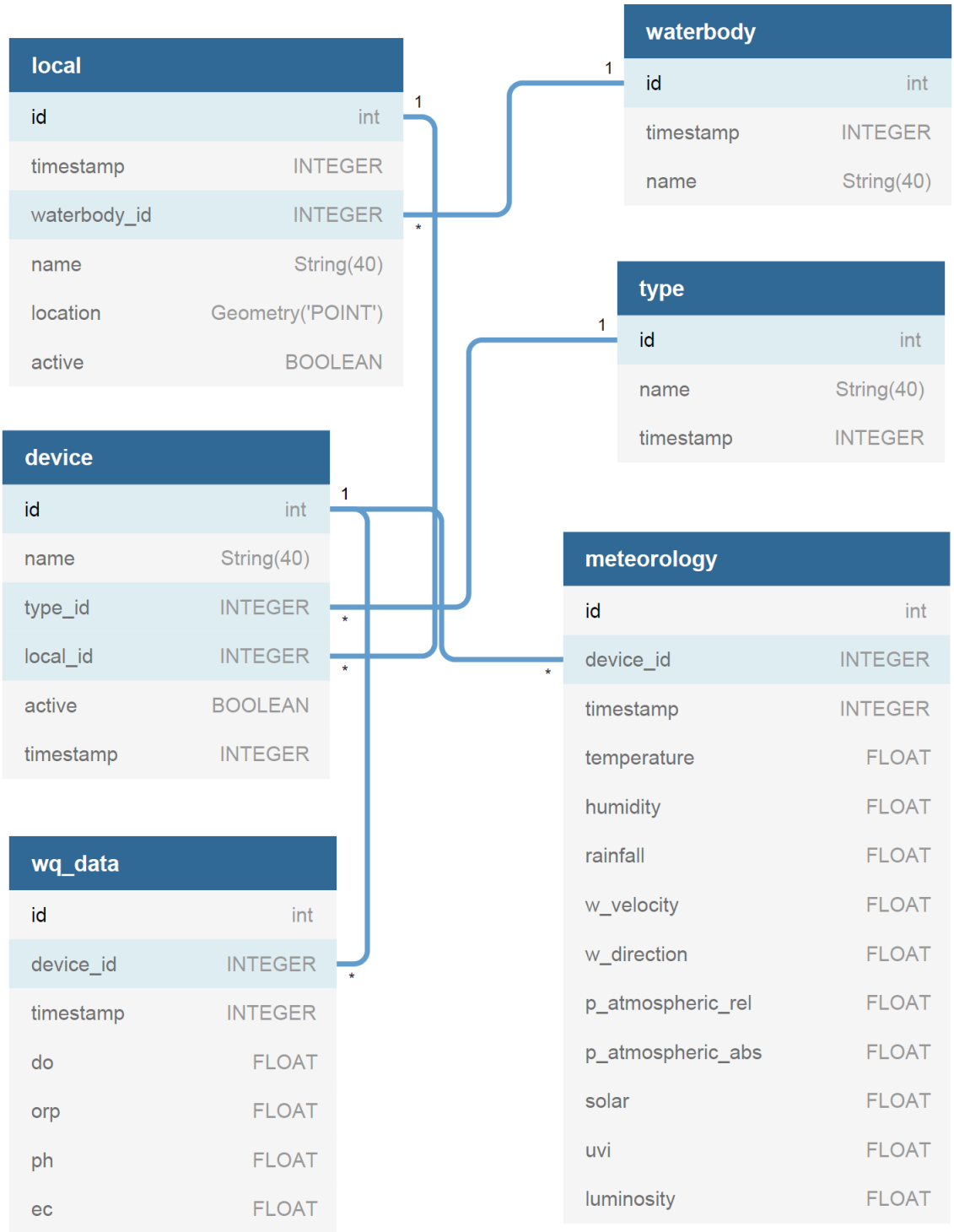


Figura IX.1: São criadas seis tabelas/relações. Quando é executada a aplicação *Data Storing App* (Secção 5.5), se ainda não foi criado o modelo de dados, cria-o (Figura IX.4).

Tabela IX.1: Exemplo de dados da tabela *waterbody*

id	timestamp	name
4	1619793573	Barragem da Laje
1	1619437140	Barragem de Alqueva

A tabela *local* armazena a localização das estações de aquisição de dados. Cada local de aquisição de dados, pertencente a um corpo de água.

Tabela IX.2: Exemplo de dados da tabela *local*

id	timestamp	waterbody_id	name
3	1619793782	4	Lage I
2	1619437998	1	Mourão I

A tabela *type* armazena tipos de dispositivos, tais como módulos de aquisição de dados de qualidade da água ou módulos meteorológicos.

Tabela IX.3: Exemplo de dados da tabela *type*

id	name	timestamp
2	Meteorology	1619444095
1	Water Quality Physical	1621884754

A tabela *devices*, contém informação referente a dispositivos que possuem funções diversas e que estão associados a um «local».

Tabela IX.4: Exemplo de dados da tabela *device*

id	name	type_id	loca_id	active
2	Sensor Node Meteo MT0001	2	3	true
1	Sensor Node LoRa LR0001	1	3	true

A tabela *wq_ data* armazena as medições da qualidade da água, referente ao módulo de qualidade da água. Cada registo de dados, na tabela *wq_ data*, é atribuído a um dispositivo de qualidade da água.

Tabela IX.5: Data from Water Quality Module - Table wq_data

id	device_id	timestamp	do	orp	ph	ec
499	1	1622931070	5	249	7.5	963
498	1	1621884754	6	249	7.7	962

A tabela *meteorology* armazena as medições da Estação Meteorológica, exemplo: Tabela IX.6.

Tabela IX.6: Data from Weather Station Module - Table meteorology

id	device_id	timestamp	temperature	humidity	rainfall	...
50	2	1620057566	23	65	0.0	...
49	2	1620057552	23	65	0.0	...

Ferramentas PostgreSQL e PostGIS

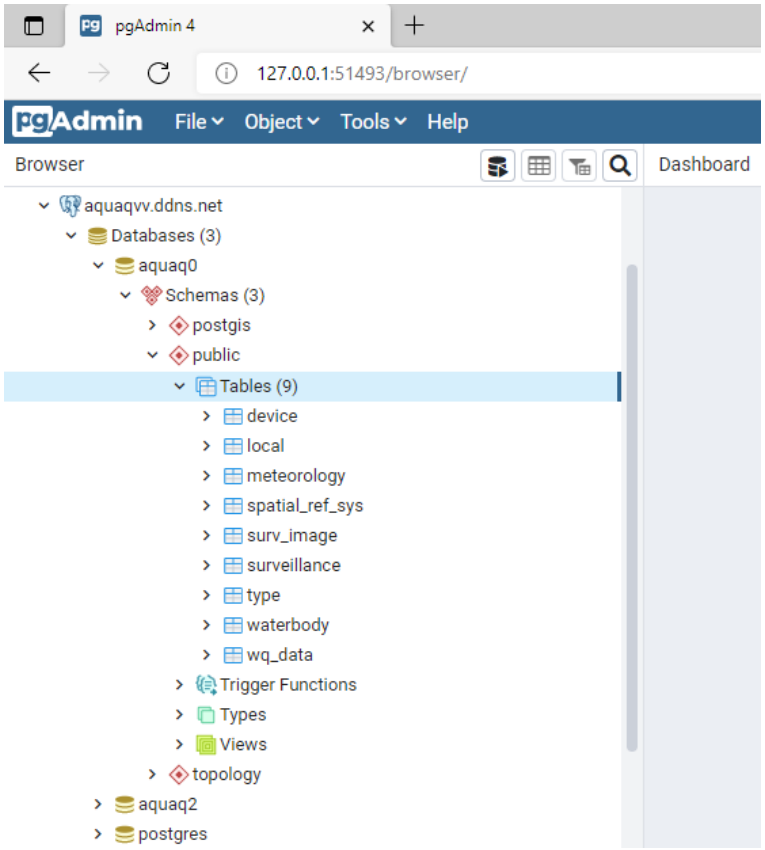


Figura IX.2: pgAdmin - permite interagir com os *SGBD* PostgreSQL/PostGIS.

Armazenamento de Informação Gerada Dados dos Módulos de Aquisição de Dados

A base de dados *«aquaq2»*, foi criada e parametrizada, na instância contentorizada do *PostGIS*. No Apêndice IX são apresentados os comandos e código, para criar a base de

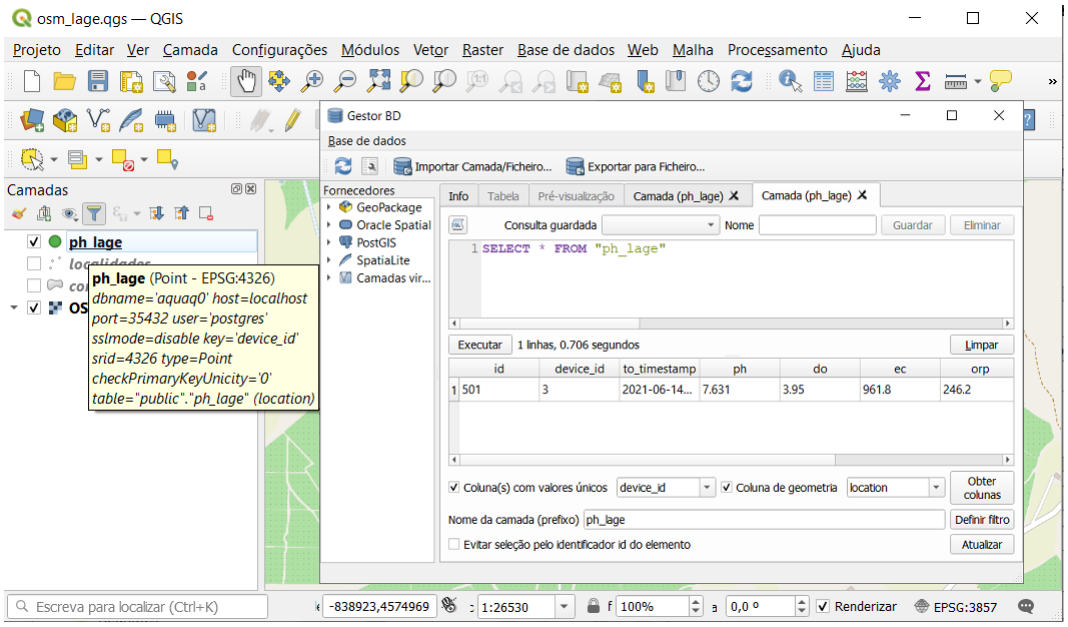


Figura IX.3: Ambiente de trabalho *QGIS* possibilita a interação com a base de dados *PostGIS*.

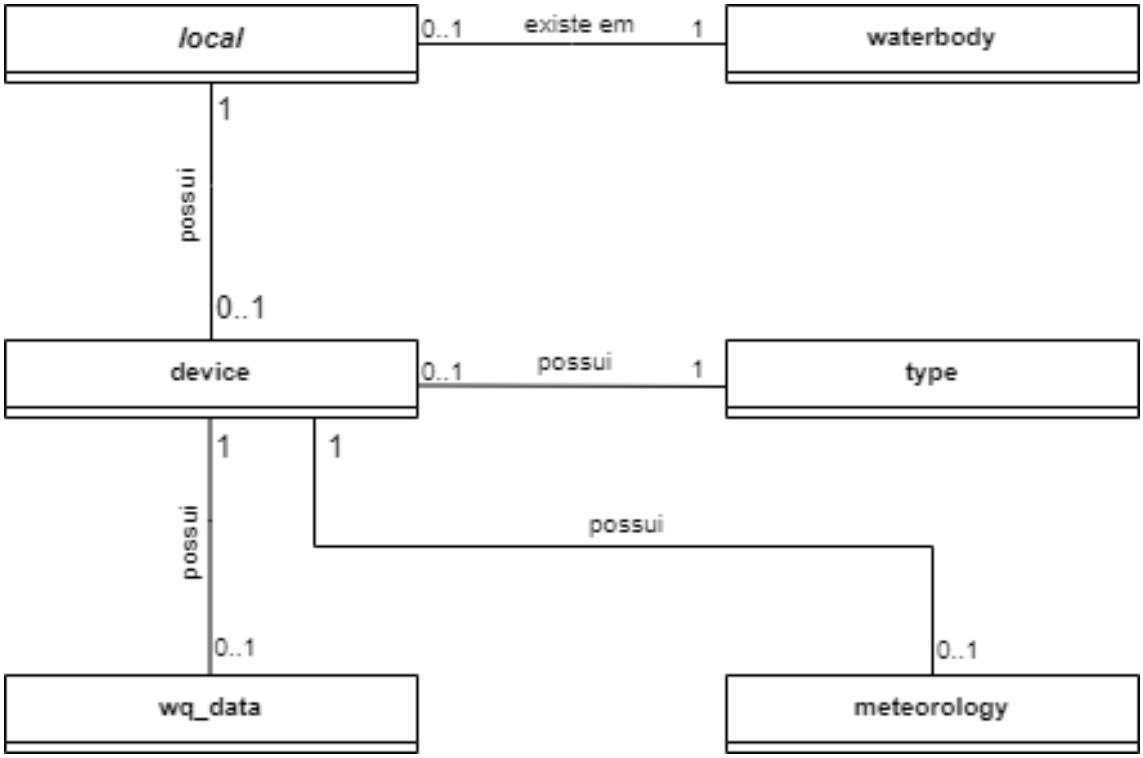


Figura IX.4: Modelo de dados relacional, utilizado para armazenar os dados dos módulos de aquisição de dados.

dados.

Quando é executada a aplicação *Data Storing App* (Secção 5.5), se ainda não foi criado o modelo de dados, cria-o (Figura IX.4). São criadas seis tabelas/relações, conforme Figura IX.1 (Apêndice IX).

A tabela *«waterbody»* armazena a identificação dos corpos de água escolhidos para a aquisição de dados. A Tabela IX.1 ilustra a estrutura da tabela, assim como uma amostra dos dados que esta armazena.

A tabela *«local»* armazena a localização das estações de aquisição de dados, armazena o nome do corpo de água, o nome do local/estação, as coordenadas de localização, e o estado do «local», ativo ou não ativo. O campo que representa as coordenadas de localização é a chave que permite a georeferenciação, das estações de recolha de dados (Figura 5.11). Todo o *«local»* pertence a um corpo de água. Cada *«local»*, possui o *«id»* de um corpo de água. A cada registo desta tabela, corresponde a uma «localização» onde existe ou já existiram dispositivos associados. A Tabela IX.2 ilustra, a estrutura da tabela *«local»*, assim como uma amostra dos dados que esta armazena.

A tabela *«type»* armazena os tipos de dispositivos, por exemplo: módulos de aquisição de dados de qualidade da água ou módulos meteorológicos.

A tabela *«device»* armazena os dispositivos sensores (Tabela IX.4). A cada dispositivo (*«device»*) é atribuído um «tipo» e um «local» .

A tabela *«wq_ data»* armazena as medições da qualidade da água, referente ao módulo de qualidade da água, é ilustrado um exemplo na Tabela IX.5. Cada registo na tabela *«wq_ data»* é atribuído a um dispositivo de qualidade da água.

A tabela *«meteorology»* armazena as medições da Estação Meteorológica, exemplo: Tabela IX.6.

Apêndice X

Código

Neste último anexo, estão incluídas as partes mais relevantes do código Python que foram feitas neste projeto para o correto funcionamento do sistema.

Data Storing App

main.py

```
1  # ----------------------------
2  # I.P. Beja AquaQ2 2021
3  # Aluno n. 756
4  # ----------------------------
5  from sqlalchemy import create_engine, Column, Integer, String,
       Sequence, Float,PrimaryKeyConstraint, ForeignKey
6  from sqlalchemy.ext.declarative import declarative_base
7  from sqlalchemy.orm import sessionmaker, relationship, backref
8  from sqlalchemy.sql import *
9  import os
10 from geoalchemy2 import Geometry
11 from motor import gerar_motor
12 from basededados import *
13
14 #from basedados import copia_dados
15 from aquaq2mqtt import mqtt_aquaq2
16
17 if __name__ == "__main__":
18     print(" AquaQ2 - Data Storing App - 2021.11.22")
19     mqtt_aquaq2()
20     pass
```

Listagem X.1: main.py

aquaq2mqtt.py

```
1  # ----------------------------
2  # I.P. Beja AquaQ2 2021
3  # Aluno n. 756
4  # ----------------------------
5  """
6  AquaQ2 - ligação MQTT para a recolha de dados
7  """
8  import paho.mqtt.client as mqtt
9  import json
10 from motor import gerar_motor
11 from inserir_dados import inserir_dados
12 from basededados import *
13
14 DISPOSITIVO_station_1          = 1
15
16 motor = gerar_motor()
17 # Gera o esquema da base de dados
18 Base.metadata.create_all(motor)
19
20 # The callback for when the client receives a CONNACK response from
       the server.
21 def on_connect_st1_sensores(client, userdata, flags, rc):
22     client.subscribe("application/1/device/00b9e8957c0f1cdd/event/up")
23     pass
24
25 # Quando uma mensagem é subscrita do servidor, envia os dados para
       armazenamento
26 def on_message_st1_sensores(client, userdata, msg):
27     """
28     recebe a mensagem e armazena na base de dados
29     """
30     payload = msg.payload
31     print(msg.topic+" "+str(msg.payload))
32     dados = json.loads(payload)
33     inserir_dados(dados, motor, DISPOSITIVO_station_1)
34     pass
35
36 def mqtt_aquaq2():
37     client_st1_sensores = mqtt.Client()
38     print('client_st1_sensores')
39     client_st1_sensores.on_connect = on_connect_st1_sensores
40     client_st1_sensores.on_message = on_message_st1_sensores
41     username = 'sensor'
42     password = 'sensor$$$2020'
43     server = "aquaq2vv.ddns.net"
44     client_st1_sensores.username_pw_set(username=username, password=
       password)
45     client_st1_sensores.connect(server, 1883, 60)
```

```
46
47      client_st1_sensores.loop_start()
48      while True:
49          pass
50      pass
```

Listagem X.2: aquaq2mqtt.py

basededados.py

```
1  # ----------------------------
2  # I.P. Beja AquaQ2 2021
3  # Aluno n. 756
4  # ----------------------------
5  From sqlalchemy import create_engine, text, MetaData,\
6      Column, Integer, String, Float, Boolean, ForeignKey, select
7  from sqlalchemy.orm import registry, declarative_base, \
8      relationship, Session, backref
9  from geoalchemy2 import Geometry
10 from sqlalchemy.ext.declarative import declarative_base
11
12 Base = declarative_base()
13
14
15 class Waterbody(Base):
16      """
17      classe que regista corpos de água
18      """
19      __tablename__ = 'waterbody'
20
21      id            = Column(Integer, primary_key=True)
22      timestamp         = Column(Integer)
23      name          = Column(String(40))
24
25      locais = relationship("Local")
26
27      def __repr__(self):
28       return "<User(timestamp='%s', name='%s', location='%s', active='%
        s')>" % (self.timestamp, self.name, self.location, self.active)
29      pass
30
31
32 class Local(Base):
33      """
34      classe que mapeia as tabelas dos locais referidos
35      com a sua designação e respetivas coordenadas GPS
36      """
37      __tablename__ = 'local'
```

```
38
39      id    = Column(Integer, primary_key=True)
40      timestamp        = Column(Integer)
41      waterbody_id = Column(Integer, ForeignKey('waterbody.id'))
42      name         = Column(String(40))
43      location         = Column(Geometry('Point'))
44      active       = Column(Boolean)
45
46      devices = relationship("Device")
47
48      def __repr__(self):
49       return "<User(timestamp='%s', name='%s', location='%s', active='%
        s')>" % (self.timestamp, self.name, self.location, self.active)
50      pass
51
52
53  class Type(Base):
54      """
55      classe que representa os tipos de dispositivos
56      """
57
58      __tablename__ = 'type'
59
60      id           = Column(Integer, primary_key=True)
61      name         = Column(String(40))
62      timestamp        = Column(Integer)
63
64      devices = relationship("Device")
65
66      def __repr__(self):
67       return "<User(name='%s',timestamp='%s')>" % (self.name, self.
        timestamp)
68      pass
69
70
71  class Device(Base):
72      """
73      classe que representa os dispositivos
74      """
75
76      __tablename__ = 'device'
77
78      id   = Column(Integer, primary_key=True)
79      name = Column(String(40))
80      type_id = Column(Integer, ForeignKey('type.id'))
81      local_id = Column(Integer, ForeignKey('local.id'))
82      active     = Column(Boolean)
83      timestamp = Column(Integer)
84
```

```
85
86      devices_data = relationship("WQ_Data")
87      devices_data = relationship("Meteorology")
88      devices_data = relationship("Surveillance")
89      devices_data = relationship("Surv_Image")
90
91      def __repr__(self):
92       return "<User(name='%s', type_id='%s', local_id='%s')>" % (self.
        nome, self.type_id, self.local_id)
93      pass
94
95
96  class WQ_Data(Base):
97      """
98      informação a representar na base de dados
99      """
100     __tablename__ = 'wq_data'
101
102     id = Column(Integer, primary_key=True)
103     device_id    = Column(Integer, ForeignKey('device.id'))
104     timestamp        = Column(Integer)
105     do    = Column(Float)
106     orp   = Column(Float)
107     ph    = Column(Float)
108     ec    = Column(Float)
109     temperature = Column(Float)
110     field01       = Column(Float)
111     field02       = Column(Float)
112     field03       = Column(Float)
113     def __repr__(self):
114         return "<User(device_id='%s, timestamp='%s', 'do='%s, 'orp='%s
        ', ph='%s', ec='%s', field01='%s', field02='%s', field03='%s' )>" %
         (self.device_id, self.timestamp, self.do, self.orp, self.ph, self.
        ec, self.temperature , self.field01 , self.field02 , self.field03)
115     pass
116
117 class Meteorology(Base):
118     """
119     informação a representar na base de dados
120     """
121     __tablename__ = 'meteorology'
122
123     id = Column(Integer, primary_key=True)
124     device_id    = Column(Integer, ForeignKey('device.id'))
125     timestamp        = Column(Integer)
126     temperature = Column(Float)
127     humidity   = Column(Float)
128     rainfall    = Column(Float)
129     w_velocity   = Column(Float)
```

```
130     w_direction = Column(Float)
131     p_atmospheric_rel  = Column(Float)
132     p_atmospheric_abs  = Column(Float)
133     solar    = Column(Float)
134     uvi    = Column(Float)
135     luminosity  = Column(Float)
136     field01       = Column(Float)
137     field02       = Column(Float)
138     field03       = Column(Float)
139     def __repr__(self):
140         return "<User(device_id='%s', timestamp='%s', 'temperature='%s,
         'humidity='%s', rainfall='%s', w_velocity='%s', w_direction='%s',
        p_atmospheric_rel='%s', p_atmospheric_abs='%s', solar='%s',uvi='%s
        ', luminosity='%s', field01='%s',field02='%s', field03='%s' )>" % (
        self.device_id, self.timestamp, self.temperature, self.humidity,
        self.rainfall, self.w_velocity, self.w_direction, self.
        p_atmospheric_rel, self.p_atmospheric_abs, self.solar, self.uvi,
        self.luminosity, self.field01, self.field02, self.field03)
141     pass
142
143 class Surveillance(Base):
144     """
145     informação a representar na bafrom basededados import *se de dados
146     """
147     __tablename__ = 'surveillance'
148
149     id = Column(Integer, primary_key=True)
150     device_id    = Column(Integer, ForeignKey('device.id'))
151     timestamp        = Column(Integer)
152     temperature = Column(Float)
153     humidity  = Column(Float)
154     presence     = Column(Boolean)
155     gas  = Column(Float)
156     field01       = Column(Float)
157     field02       = Column(Float)
158     field03       = Column(Float)
159     def __repr__(self):
160         return "<User(device_id='%s', timestamp='%s', 'temperature='%s,
         'humidity='%s', presence='%s', gas='%s', field01='%s',field02='%s
        ', field03='%s' )>" % (self.device_id, self.timestamp, self.
        temperature, self.humidity, self.presence, self.gas, self.field01,
        self.field02, self.field03)
161     pass
162
163
164 class Surv_Image(Base):
165     """
166     informação a representar na base de dados
167     """
```

```
168     __tablename__ = 'surv_image'
169
170     id = Column(Integer, primary_key=True)
171     device_id    = Column(Integer, ForeignKey('device.id'))
172     timestamp        = Column(Integer)
173     photo  = Column(String(40))       #bytea
174     video  = Column(String(40))  #bytea
175     field01       = Column(Float)
176     field02       = Column(Float)
177     field03       = Column(Float)
178     def __repr__(self):
179         return "<User(device_id='%s, timestamp='%s', 'photo='%s, '
        video='%s', field01='%s',field02='%s', field03='%s' )>" % (self.
        device_id, self.timestamp, self.photo, self.video, self.field01,
        self.field02, self.field03)
180     pass
```

Listagem X.3: basededados.py

inserir_dados.py

```
1  # ------------------------------
2  # I.P. Beja AquaQ2 2021
3  # Aluno n. 756
4  # ------------------------------
5  from motor import gerar_motor
6  from sqlalchemy.orm import Session
7  import json
8  import pybase64
9  from basededados import WQ_Data
10 import io
11 import time
12
13 i = 0
14 a_ts = [0] * 5
15 # Verificar medições duplicadas. No mesmo TS
16 # -- para rever ---
17 def conta(d_ts):
18   global i
19   global a_ts
20   i = i + 1;
21   a_ts[i]=d_ts
22   print(a_ts)
23   i = i + 1
24   if ( i >=2 ) :
25    if (a_ts[0] != a_ts[1]):
26     i=0
27     a_ts[0]= a_ts[1]
```

```
28      return True
29     else :
30      i=0
31      a_ts[0]= a_ts[1]
32    else:
33     i=0
34     a_ts[0]= a_ts[1]
35     return False
36    pass
37
38
39  def inserir_dados(dados, motor, DISPOSITIVO_station_1):
40   """
41   dados do sensor a serem guardadados na base de dados
42   """
43   with Session(motor) as session:
44    if (dados["data"] != None ) :
45     dados_sensores = json.loads(pybase64.standard_b64decode(dados["data
       "]))
46     if (conta(int(dados_sensores["ts"])) == True) :
47      id = 'default'
48      dox__ = float(dados_sensores["do"])
49      orp__ = float(dados_sensores["orp"])
50      ph__ = float(dados_sensores["ph"])
51      ec__ = float(dados_sensores["ec"])
52      ts__ = int(dados_sensores["ts"])
53      temperature__ = float(dados_sensores["T"])
54      field01__ = 0;
55      field02__ = 0;
56      field03__ = 0;
57      dispositivo_id_ = int(dados["applicationID"])
58      dispositivo_data = WQ_Data(device_id=dispositivo_id_,timestamp=
       ts__,do=dox__,orp=orp__,ph=ph__,ec=ec__,temperature=temperature__,
       field01=field01__,field02=field02__,field03=field03__)
59      session.add(dispositivo_data)
60      session.commit()
61    pass
62   pass
```

Listagem X.4: inserir_dados.py

Data Visualization App

main.py

```
1 ##!/usr/bin/env python
2 # ----------------------------
```

```
3 # I.P. Beja AquaQ2 2021
4 # Aluno n. 756
5 #  main.py
6 #  Adapted from https://github.com/Mjrovai/RPI-Flask-SQLite/tree/
      master/dhtWebHist_v2
7 # ------------------------
8  '''
9  AquaQ2 WEb Server - Monitorig Water Quality - Uses Gage and Graph
       plot
10 '''
11 from motor import gerar_motor
12 from datetime import datetime, time
13 from matplotlib.backends.backend_agg import FigureCanvasAgg as
       FigureCanvas
14 from matplotlib.figure import Figure
15 import io
16 from flask import Flask, render_template, send_file, make_response,
       request
17
18 app = Flask(__name__)
19 curs = gerar_motor()
20
21 # Retrieve LAST data from database - table wqdata - Device 1
22 def getLastData():
23  for row in curs.execute("SELECT * FROM wq_data ORDER BY timestamp
       DESC LIMIT 1"):
24   time = str(datetime.fromtimestamp(int(row[2])))
25   do = row[3]
26   orp = row[4]
27   ph  = row[5]
28   ec  = row[6]
29  return time, do, orp, ph, ec
30
31
32 # Retrieve LAST data from database - table wqdata - Device 1
33 def getLastReading():
34  for row in curs.execute("SELECT * FROM wq_data ORDER BY timestamp
       DESC LIMIT 1"):
35   time = str(datetime.fromtimestamp(int(row[2])))
36  return time
37
38
39
40 # Retrieve LAST data from database - table meteorology - Device 2
41 def getLastData_d2():
42  for row in curs.execute("SELECT * FROM meteorology ORDER BY timestamp
       DESC LIMIT 1"):
43   time = str(datetime.fromtimestamp(int(row[2])))
44   temperature = row[3]
```

```
45    humidity = row[4]
46    rainfall  = row[5]
47   return time, temperature, humidity, rainfall
48
49 # Get 'x' samples of historical data
50 def getHistData (numSamples):
51   result = curs.execute("SELECT * FROM wq_data ORDER BY timestamp DESC
       LIMIT "+str(numSamples))
52   data = result.fetchall()
53   dates = []
54   dos = []
55   orps = []
56   phs  = []
57   ecs  = []
58   for row in reversed(data):
59    time_ = str(datetime.fromtimestamp(int(row[2])))
60    dates.append(time_)
61    dos.append(row[3])
62    orps.append(row[4])
63    phs.append(row[5])
64    ecs.append(row[6])
65    dos, orps, phs, ecs = testeData(dos, orps, phs, ecs)
66   return dates, dos, orps, phs, ecs
67
68
69 # Get 'x' samples of historical data - DO
70 def getHistData_do (numSamples):
71   result = curs.execute("SELECT * FROM wq_data ORDER BY timestamp DESC
       LIMIT "+str(numSamples))
72   data = result.fetchall()
73   dates = []
74   dos = []
75   for row in reversed(data):
76    time_ = str(datetime.fromtimestamp(int(row[2])))
77    dates.append(time_)
78    dos.append(row[3])
79    dos = testeData_do(dos)
80   return dates, dos
81
82 # Get 'x' samples of historical data - ORP
83 def getHistData_orp (numSamples):
84   result = curs.execute("SELECT * FROM wq_data ORDER BY timestamp DESC
       LIMIT "+str(numSamples))
85   data = result.fetchall()
86   dates = []
87   orps = []
88   for row in reversed(data):
89    time_ = str(datetime.fromtimestamp(int(row[2])))
90    dates.append(time_)
```

```
91      orps.append(row[4])
92    return dates, orps
93
94  # Get 'x' samples of historical data - pH
95  def getHistData_ph (numSamples):
96    result = curs.execute("SELECT * FROM wq_data ORDER BY timestamp DESC
        LIMIT "+str(numSamples))
97    data = result.fetchall()
98    dates = []
99    phs = []
100   for row in reversed(data):
101     time_ = str(datetime.fromtimestamp(int(row[2])))
102     dates.append(time_)
103     phs.append(row[5])
104     #phs = testeData_ph(phs)
105   return dates, phs
106
107 # Get 'x' samples of historical data - EC
108 def getHistData_ec (numSamples):
109   result = curs.execute("SELECT * FROM wq_data ORDER BY timestamp DESC
        LIMIT "+str(numSamples))
110   data = result.fetchall()
111   dates = []
112   ecs = []
113   for row in reversed(data):
114     time_ = str(datetime.fromtimestamp(int(row[2])))
115     dates.append(time_)
116     ecs.append(row[6])
117   return dates, ecs
118
119
120 # Test data for cleanning possible "out of range" values
121 def testeData(dos, orps, phs, ecs):
122   n = len(dos)
123   for i in range(0, n-1):
124     if (dos[i] < 0 or dos[i] >20):
125       dos[i] = dos[i-2]
126     if (orps[i] < 0 or orps[i] >500):
127       orps[i] = dos[i-2]
128     if (phs[i] < 0 or phs[i] >14):
129       phs[i] = phs[i-2]
130     if (ecs[i] < 50 or ecs[i] > 1500):
131       ecs[i] = ecs[i-2]
132   return dos, orps, phs, ecs
133
134
135 # Test data for cleanning possible "out of range" values
136 def testeData_do(dos):
137   n = len(dos)
```

```
138  for i in range(0, n-1):
139   if (dos[i] < 0 or dos[i] >20):
140    dos[i] = dos[i-2]
141  return dos
142
143 # Get Max number of rows (table size)
144 def maxRowsTable():
145  for row in curs.execute("select COUNT(id) from  wq_data"):
146   maxNumberRows=row[0]
147  return maxNumberRows
148
149 # Get sample frequency in minutes
150 def freqSample():
151  times, dos, orps, phs, ecs = getHistData (5)
152  fmt = '%Y-%m-%d %H:%M:%S'
153  tstamp0 = datetime.strptime(times[0], fmt)
154  tstamp1 = datetime.strptime(times[1], fmt)
155  freq = tstamp1-tstamp0
156  freq = int(round(freq.total_seconds()/60))
157  print("Frequencia em minutos", freq)
158  return (freq)
159
160 # Get number of samples for a range time in minutes
161 def numsample(rt):
162  ns = int(rt / freqSamples)
163  if (ns > maxRowsTable()):
164    ns = maxRowsTable() - 1
165  return (ns)
166
167 global freqSamples
168 freqSamples = freqSample()
169
170 # main route
171 @app.route("/")
172 @app.route("/home")
173 @app.route("/index")
174 def index():
175     rangeTime = 1440
176     numSamples = numsample(rangeTime)
177     time, do, orp, ph, ec = getLastData()
178     templateData = {
179    'time'  : time,
180       'do'   : do,
181       'orp'   : orp,
182    'ph'   : ph,
183    'ec'   : ec,
184       'freq'  : freqSamples,
185       'rangeTime' : rangeTime,
186       'numSamples' : numSamples
```

```
187  }
188      return render_template('index_bs.html', **templateData)
189
190 @app.route("/teste01")
191 def teste01():
192  return render_template('teste01.html')
193
194 @app.route("/historical")
195 def historical():
196      return render_template('index_bs_range.html')
197
198 @app.route('/historical', methods=['POST'])
199 def my_form_post():
200      global numSamples
201      global freqSamples
202      global numS
203      rangeT = int (request.form['rangeT'])
204      print("Range Time ---------- =", rangeT)
205      numS = numsample(rangeT)
206      templateData = {
207        'time'     : getLastReading(),
208        'freq'   : freqSamples,
209        'rangeT' : rangeT,
210     'numS'   : numS
211  }
212      return render_template('index_bs_range_res.html', **templateData)
213
214 @app.route('/plot/do/<ns>')
215 def plot_do(ns):
216  n = int(ns)
217  times, dos = getHistData_do(n)
218  ys = dos
219  fig = Figure()
220  axis = fig.add_subplot(1, 1, 1)
221  axis.set_title("DO [mg/L]")
222  axis.set_xlabel("Sample")
223  axis.grid(True)
224  xs = range(n)
225  axis.plot(xs, ys)
226  canvas = FigureCanvas(fig)
227  output = io.BytesIO()
228  canvas.print_png(output)
229  response = make_response(output.getvalue())
230  response.mimetype = 'image/png'
231  return response
232
233 @app.route('/plot/orp/<ns>')
234 def plot_orp(ns):
235  n = int(ns)
```

```
236  times, orps = getHistData_orp(n)
237  ys = orps
238  fig = Figure()
239  axis = fig.add_subplot(1, 1, 1)
240  axis.set_title("ORP [mV]")
241  axis.set_xlabel("Sample")
242  axis.grid(True)
243  xs = range(n)
244  axis.plot(xs, ys)
245  canvas = FigureCanvas(fig)
246  output = io.BytesIO()
247  canvas.print_png(output)
248  response = make_response(output.getvalue())
249  response.mimetype = 'image/png'
250  return response
251
252 @app.route('/plot/ph/<ns>')
253 def plot_ph(ns):
254  n = int(ns)
255  times, phs = getHistData_ph(n)
256  ys = phs
257  fig = Figure()
258  axis = fig.add_subplot(1, 1, 1)
259  axis.set_title("pH [Sorensen Scale]")
260  axis.set_xlabel("Sample")
261  axis.grid(True)
262  xs = range(n)
263  axis.plot(xs, ys)
264  canvas = FigureCanvas(fig)
265  output = io.BytesIO()
266  canvas.print_png(output)
267  response = make_response(output.getvalue())
268  response.mimetype = 'image/png'
269  return response
270
271 @app.route('/plot/ec/<ns>')
272 def plot_ec(ns):
273  n = int(ns)
274  times, ecs = getHistData_ec(n)
275  ys = ecs
276  fig = Figure()
277  axis = fig.add_subplot(1, 1, 1)
278  axis.set_title("EC [$\mu$S/cm]")
279  axis.set_xlabel("Sample")
280  axis.grid(True)
281  xs = range(n)
282  axis.plot(xs, ys)
283  canvas = FigureCanvas(fig)
284  output = io.BytesIO()
```

```
285  canvas.print_png(output)
286  response = make_response(output.getvalue())
287  response.mimetype = 'image/png'
288  return response
289
290 if __name__ == "__main__":
291     app.run(debug=True)
```

Listagem X.5: main.py

*

motor.py

```
1 # ----------------------------------------
2 # I.P. Beja AquaQ2 2021
3 # Aluno n. 756
4 # ----------------------------------------
5 from sqlalchemy import create_engine
6 from sqlalchemy.orm import sessionmaker
7
8 def gerar_motor():
9  driver_bd = "postgresql+psycopg2"
10  user_bd   = "postgres"
11  passwd_bd = "aquaq2barba2021"
12  ip_bd     = "192.168.0.106"
13  port_bd   = "35432"
14  bd_bd     = "aquaq0"
15  base_de_dados = driver_bd + "://" + user_bd + ":" + passwd_bd + "@" +
        ip_bd + ":" + port_bd + "/" + bd_bd
16  motor = create_engine(base_de_dados, echo=True)
17  return motor
```

Listagem X.6: motor.py

index_bs.html

```
1 <!-- I.P. Beja AquaQ2 2021-->
2 <!-- Aluno n. 756-->
3 <!DOCTYPE html>
4 <html lang="pt">
5 <head>
6   <title>AquaQ2 - Water Quality Monitoring</title>
7   <meta charset="utf-8">
8   <meta name="viewport" content="width=device-width, initial-scale=1">
9   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
      bootstrap/3.4.1/css/bootstrap.min.css">
```

```
10    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/
        jquery.min.js"></script>
11    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/
        bootstrap.min.js"></script>
12    <style>
13   h3 {text-align: center;}
14   p {text-align: center;}
15   div {text-align: center;}
16    </style>
17  </head>
18  <body>
19  <!--Atualiza a página a cada 2 minutos-->
20   <script>
21    window.setTimeout(function () {
22      window.location.reload();
23    }, 120000);
24    </script>
25  <div class="jumbotron text-center">
26    <h2>Surface Water Quality Monitoring</h2>
27    <p>AquaQ2 2021 </p>
28  </div>
29   <div class="row">
30       <div id="g1" class="col-sm-3" style="background-color:rgb(248,
        248, 250)"></div>
31       <div id="g2" class="col-sm-3" style="background-color:rgb(248,
        248, 250)"></div>
32       <div id="g3" class="col-sm-3" style="background-color:rgb(248,
        248, 250)"></div>
33    <div id="g4" class="col-sm-3" style="background-color:rgb(248, 248,
        250)"></div>
34     </div>
35     <div class="row">
36       <div class="col-sm" style="background-color:rgb(248, 248, 250)">
37      <!-- <h2> Last Sensors Reading: {{ time }} ==> <a href="/"class="
        button">REFRESH</a></h2> -->
38        <h3> Last Sensors Reading: {{ time }} ==> <a href="/" class="
        btn btn-default">REFRESH</a></h3>
39      <h4> Frequency: {{ freq }} minutes  Range Time: 1 day minutes  N.
        Samples: {{ numSamples }} </h4>
40       </div>
41   </div>
42
43   <div class="row">
44    <h3> <a href="/historical" class="btn btn-default">Historical DATA</
        a></h3>
45   </div>
46      <div class="row">
47          <div class="col-md-3" style="background-color:rgb(248, 248,
        250)">
```

```
48                <img src="/plot/do/{{numSamples}}" alt="Image
    Placeholder" width="100%">
49            </div>
50            <div class="col-md-3" style="background-color:rgb(248, 248,
    250)">
51                <img src="/plot/orp/{{numSamples}}" alt="Image
    Placeholder" width="100%">
52            </div>
53            <div class="col-md-3" style="background-color:rgb(248, 248,
    250)">
54                <img src="/plot/ph/{{numSamples}}" alt="Image
    Placeholder" width="100%">
55    </div>
56    <div class="col-md-3" style="background-color:rgb(248, 248, 250)">
57     <img src="/plot/ec/{{numSamples}}" alt="Image Placeholder" width="
      100%">
58     </div>
59  </div>
60  <p> @2021 - AquaQ2  </p>
61      <script src="../static/raphael-2.1.4.min.js"></script>
62      <script src="../static/justgage.js"></script>
63      <script>
64    var g1, g2, g3, g4;
65    document.addEventListener("DOMContentLoaded", function(event) {
66     g1 = new JustGage({
67      id: "g1",
68      value: {{do}},
69      valueFontColor: "black",
70      titleFontColor: "black",
71      labelFontColor: "black",
72      min: 0,
73      max: 20,
74      decimals: 1,
75      title: "DO",
76      label: "mg/L"
77     });
78
79     g2 = new JustGage({
80      id: "g2",
81      value: {{orp}},
82      valueFontColor: "black",
83      titleFontColor: "black",
84                    labelFontColor: "black",
85      min: 0,
86      max: 500,
87      decimals: 1,
88      title: "ORP",
89      label: "mV"
90     });
```

```
91      g3 = new JustGage({
92       id: "g3",
93       value: {{ph}},
94       valueFontColor: "black",
95       titleFontColor: "black",
96                   labelFontColor: "black",
97       min: 0,
98       max: 14,
99       decimals: 1,
100      title: "pH",
101      label: "E.Sorensen"
102     });
103     g4 = new JustGage({
104      id: "g4",
105      value: {{ec}},
106      valueFontColor: "black",
107      titleFontColor: "black",
108      labelFontColor: "black",
109      min: 50,
110      max: 1500,
111      decimals: 1,
112      title: "EC",
113      label: "microS/cm"
114     });
115   });
116     </script>
117   </div>
118 </div>
119 </body>
120 </html>
```

Listagem X.7: index_bs.html

index_bs_range.html

```
1  <!-- I.P. Beja AquaQ2 2021-->
2  <!-- Aluno n.756-->
3  <!DOCTYPE html>
4  <html lang="pt">
5  <head>
6    <title>AquaQ2 - Water Quality Monitoring</title>
7    <meta charset="utf-8">
8    <meta name="viewport" content="width=device-width, initial-scale=1">
9    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
       bootstrap/3.4.1/css/bootstrap.min.css">
10   <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/
       jquery.min.js"></script>
```

```
11   <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/
       bootstrap.min.js"></script>
12   <style>
13  h3 {text-align: center;}
14  p {text-align: center;}
15  div {text-align: center;}
16   </style>
17 </head>
18 <body>
19      <div class="jumbotron text-center">
20          <h1>Surface Water Quality Monitoring</h1>
21          <p>AquaQ2 2021</p>
22      </div>
23     <div class="row">
24       <div class="col-sm" style="background-color:rgb(248, 248, 250)">
25        <h3> Last Sensors Reading: {{ time }}</h3>
26       </div>
27  </div>
28  <div class="row">
29   <div class="col-sm" style="background-color:rgb(248, 248, 250)">
30    <h3> HISTORICAL DATA </h3>
31    <p> Enter range of time to be retrieved in minutes (Sample
       Frequency: {{ freq }} minutes)
32    <form method="POST">
33    <input name="rangeT" value= {{rangeT}}>
34    <input type="submit">
35    </form></p>
36   </div>
37  </div>
38 <hr>
39  <p> @2021 - AquaQ2  </p>
40      <script src="../static/raphael-2.1.4.min.js"></script>
41      <script src="../static/justgage.js"></script>
42 </div>
43 </div>
44 </body>
45 </html>
```

Listagem X.8: index_bs_range.html

Apêndice XI

Tecnologias de Comunicação Sem Fios

Este Apêndice foi escrito para servir de suporte teórico no desenvolvimento da dissertação. Os objetivos desta tornam essencial o conhecimento das tecnologias de comunicação sem fios de baixo consumo, mais conhecidas pela terminologia inglesa de *L*ow Power Wide Area Network - *L*PWAN.

As tecnologias de comunicação sem fios permitem que os dispositivos comuniquem entre si e com a Internet, para o efeito, cada um dos protocolos disponíveis para gerir essas comunicações, possui vantagens e desvantagens. Decidir que tipo de tecnologia sem fios, utilizar num projeto pode tornar-se uma tarefa difícil. Atualmente existe um grande número de tecnologias sem fios disponíveis no mercado que podem ser implementadas em produtos de *hardware* para a comunicação na Internet das Coisas (IoT) e *Machine to Machine* (M2M).

Classificação das redes sem fios quanto ao alcance

As redes sem fios (Wireless Lan) - são uma alternativa às redes convencionais com fios, fornecendo as mesmas funcionalidades, mas de forma mais flexível, de fácil configuração e com boa conectividade, tanto é áreas residenciais como em campo aberto.

WPAN (Wireless Personal Area Network) - rede de área pessoal sem fios, permite conectar vários dispositivos de curto alcance, normalmente alguns metros. Entre outras, estão associadas às tecnologias: NFC, RFID, Bluetooth, BLE, Zigbee, Z-Wave e UWB. As WPAN são projetadas para pequenas distâncias, baixos consumos de energia, baixo custo e baixas taxas de transferência.

WLAN (Wireless Local Area Network) – rede de área local que utiliza as ondas rádio para estabelecer conexão à Internet ou a outras redes, normalmente associadas às tecnologias WIFI, padrão IEEE 802.11. Utilizadas em ambientes residenciais, empresas e em lugares públicos, com alcances típicos de até 100 metros.

WMAN (Wireless Metropolitan Area Network) - rede metropolitana sem fios, com um alcance que pode atingir algumas dezenas de quilómetros, permite, por exemplo, estabelecer conexão entre redes de escritórios de uma mesma empresa ou de campus universitário. É normalmente associada ao padrão 802.16 (WIMAX).

WWAN (Wireless Area Network) - rede disponibilizada em longas distância sem fios, é uma tecnologia que as operadoras de comunicações móveis utilizam para criar as redes de transmissão. Oferecem uma ampla cobertura, na ordem das centenas de quilómetros, custos razoáveis, altos níveis de segurança, acesso a espectro dedicado e simplicidade de gestão. Por esse motivo, são tecnologias muito atraentes no moderno cenário de conectividade de IoT. Normalmente associadas às tecnologias: 3G, 4G, LTE, LORA, SIGFOX, NB-IoT e 5G.[102]

Near-Field Communication

Comunicação por campo de proximidade, o NFC comunica utilizando campos eletromagnéticos compartilhados entre duas bobinas, enquanto quase todas as outras tecnologias sem fios emitem ondas de rádio. Enquadra-se nas redes de curto alcance (Wireless Personal Area Network – WPAN).

Principais caraterísticas de uso: baixo consumo energético, topologia ponto-a-ponto, faixa de operação de aproximadamente 10cm, padrão de comunicação ISO/IEC 18000-3, opera na frequência 13.56 Mhz, taxa de transferência de 424 Kbps, permite apenas uma rede com dois nós. O uso mais comum da NFC é nos sistemas de pagamento sem contato. Embora os dados de pagamento sejam encriptados, o alcance operacional extremamente curto da NFC também ajuda a eliminar a possibilidade de alguém próximo invadir a transação. [85]

RFID

Identificação por radio frequência (RFID) é uma tecnologia de comunicação sem fios, que utiliza ondas de rádio para transmitir dados entre um leitor e um tag. Enquadra-se nas redes de curto alcance (WPAN).

Principais caraterísticas de uso: baixo consumo energético, topologia em estrela, faixa de operação de aproximadamente 10 cm, padrão de comunicação definido pela família de

normas técnicas, ISO/IEC 18000, taxa de transferência entre os 10-100 Kbps. Apresenta-se em três níveis de frequências, com caraterísticas distintas: Low Frequency (LF), que opera em frequências de 125 KHz a 134 KHz, alcance no máximo 10 cm; High Frequency (HF), opera na frequência de 13.56 MHz, o alcance típico desta faixa de frequência pode chegar até 1 m; Ultra-High Frequency (UHF), que opera em frequências entre os 865 MHz e 960 MHz, para esta faixa de frequência, em condições normais o alcance pode atingir cerca de 5 a 6 metros (entre paredes) e 30 metros ou mais (sem obstáculos), em condições ideais, permite ligar até 7 nós. Geralmente utilizado em forma de etiqueta, anexada a um objeto, veículo, animal ou pessoa, possibilitando a sua identificação e rastreamento sem a necessidade de contato visual ou físico.[133]

Bluetooth e BLE

Bluetooth e Bluetooth *Low Energy* (BLE) são tecnologias sem fios utilizadas para transferir dados a curtas distâncias, utilizando sinal de radio frequência. Enquadram-se nas redes de curto alcance (WPAN), padrão de comunicação IEEE 802.15.1.

Principais caraterísticas de uso: baixo consumo energético (mais baixo para o BLE), permite a utilização em topologias de rede ponto-a-ponto, estrela ou malha, faixa de operação de aproximadamente 10 m para o bluetooth padrão e superior a 100 m para o BLE (dependendo das condições), taxa de transferência de aproximadamente 2.1 Mbps para o bluetooth padrão 1 Mbps para o BLE, ambos operam na mesma frequência, 2.4 Ghz, o bluetooth padrão comporta uma rede com um máximo de 7 nós (ativos, 256 no total) enquanto o BLE, praticamente sem limite (rede mesh até 32,767). A tecnologia bluetooth é comum na utilização de equipamentos de fitness, relógios inteligentes e na automação doméstica.[59]

Zigbee

Designa um conjunto de especificações para a comunicação sem fios entre dispositivos eletrónicos, com foco na baixa potência de operação, na baixa taxa de transmissão de dados e no baixo custo de implementação. Enquadra-se nas redes de curto alcance (WPAN). Padrão de comunicação IEEE 802.15.4.

Principais caraterísticas de uso: permite topologia em estrela e em malha, a distâncias de cerca de 300m em linha de vista e cerca de 75-100m entre paredes, opera nas frequências 915MHz/2.4 GHz, taxa de transferências entre 20 e 250 Kbps, consumindo muito pouca energia. Permite comunicações com excelente imunidade a interferências e taxas de transferência de dados entre os 20Kbps e os 250Kbps.É uma tecnologia sem fios proprietária, a sua especificação está a cargo de um consórcio industrial não lucrativo de produtores e outras companhias, que juntos se designam por ZigBee Alliance. Tem foco nos mercados de controlo e automação de edifícios.[44]

Z-Wave

É uma tecnologia sem fios proprietária (adquirida pela Silicon Labs em 2018) que concorre principalmente com o Zigbee e o BLE no mercado de automação residencial. Enquadra-se nas redes de curto alcance (WPAN). Padrão de comunicação IEEE 802.15.4.

Principais caraterísticas de uso: enquanto o BLE e o Zigbee, utilizam a "popular" banda de 2,4 GHz, o Z-Wave utiliza uma banda de abaixo de 1GHz, taxa de transferência de cerca de 100 Kbps e um alcance de cerca de 150m. Existem duas vantagens significativas da frequência portadora mais baixa: maior alcance e interferências reduzidas. As bandas de frequência utilizadas pelo Z-Wave tendem a ser muito menos lotadas. A desvantagem da frequência mais baixa da operadora é uma velocidade de transmissão de dados mais baixa, que acaba sendo quase 10 vezes mais lenta que o Bluetooth LE. O Z-Wave suporta redes de malha menores, apenas até 232 dispositivos, o que é mais que suficiente para a maioria das aplicações.[141]

6LoWPAN

6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) é o grupo de desenvolvimento da IETF (Internet Engineering Task Force), que cria e mantêm as especificações que permitem utilizar IPv6 nas redes IEEE 802.15.4. Enquadra-se nas redes de curto alcance (WPAN).

Principais caraterísticas de uso: baixo consumo de energia, frequência de operação 2.4 Ghz, alcance de aproximadamente 100 metros e uma taxa de transferência de dados de cerca de 250Kbps. O 6LoWPAN é essencialmente um novo concorrente do Zigbee. A principal diferença é que o 6LoWPAN é uma rede baseada em IP. O 6LoWPAN assenta na ideia de que a internet é inteiramente construída em IP (IP enable), o que significa que cada dispositivo (Host) Low Power deverá ter um IP tornando-se também uma parte no mundo da Internet ou «Internet of Things». O 6LoWPAN é um aliado da IoT, pode ser implementado em sistemas embarcados, por exemplo: smartphones, sensores pessoais, automação residencial, logística, transporte, medidores inteligentes de energia elétrica, infraestrutura de redes, etc.[48]

UWB

Banda Ultra larga (Ultra Wide Band) é uma tecnologia com modo de funcionamento muito semelhante, à tecnologia RFID e BLE, utilizando tags e sensores. Enquadra-se nas redes de curto alcance (WPAN), assenta no padrão IEEE 802.15.3.

Principais caraterísticas de uso: permite transmitir informação a taxas de transmissão que podem ultrapassar os 100 Mbps, numa grande largura de banda, baixa taxa de ruído e a curtas distâncias. Exemplos de utilização: impressoras, rato, teclado ou leitores de

mp3, com o máximo 10 metros de distância. A sua principal desvantagem é ser muito dispendiosa a sua implementação, devido às caraterísticas de suas ondas, a cobertura fica comprometida, o que exige uma grande quantidade de sensores fixos, aumentando o custo. Além disso, existe também o alto custo das tags que serão colocadas em cada elemento que precisa ser rastreado. É uma tecnologia promissora, utilizada por exemplo, em áreas militares, indústrias automobilísticas, hospitais e outros espaços ou serviços que necessitam de informações precisas.[129]

WIFI

(Wireless Fidelity), "fidelidade sem fios" ou wireless é uma tecnologia de comunicação sem fios, que utiliza ondas de rádio (RF) para permitir a comunicação entre dispositivos. Esta tecnologia é normalmente utilizada para interconetar routers da Internet com dispositivos como computadores, tablets e telefones. Enquadra-se nas redes de locais sem fios (Wireless Lan Area Network – WLAN) e assenta nos padrões IEEE 802.11.

Principais caraterísticas de uso: pode utilizar as frequência rádio de 2,4 GHz e 5 GHz, taxas de transferência entre 65 Mbps a 450 Mbps (802.11n), alcance de cerca de 100m, topologia em estrela ou ponto a ponto, consumo alto de energia.

Ainda no âmbito desta tecnologia, existe o WiFi Direct. O WiFi Direct tira partido da mesma tecnologia básica que o WiFi tradicional. Utiliza a mesma frequência e oferece largura de banda e taxas de transferência semelhantes mas, não requer um ponto de acesso, permitindo que dois dispositivos tenham uma ligação direta (ponto-a-ponto) semelhante ao Bluetooth, sem a sobrecarga de um ponto de acesso. A vantagem do WiFi Direct, por exemplo em relação ao Bluetooth, é principalmente, apresentar taxas de transferência cerca de cem vezes mais elevadas. Esta velocidade tem um preço e esse preço é principalmente maior consumo de energia.[135]

LoRa/LoRaWAN

Lora (abreviatura de Long-Range) é uma tecnologia de radio frequência que permite a comunicação sem fios a longas distâncias, enquadra-se nas redes de longo alcance (Low Power Wireless Area Network – LPWAN).

Principais caraterísticas de uso: baixo consumo energético, topologia em estrela (também pode ser utilizado em comunicações ponto a ponto), alcance de distâncias superiores a 10 Km, implementado sobre a norma IEE802.15.4, taxas de transferência que variam entre os 0.3 e 50Kbps e opera na frequência 868 MHz (Europa). LoRa refere-se à tecnologia subjacente e LoRaWAN refere-se ao protocolo de rede da camada superior, que implementa os detalhes de funcionamento, segurança, qualidade do serviço, ajustes de potência visando maximizar a duração da bateria dos módulos, e os tipos de aplicações tanto do lado do

módulo quanto do servidor. É uma tecnologia sem fios propriedade atualmente da empresa Semtech. [3]

MIOT

é um protocolo desenvolvido especificamente para grandes implementações IoT industriais e comerciais. Enquadra-se nas redes de longo alcance (LPWAN).

Principais caraterísticas de uso: muito baixo consumo, taxa de transferência na ordem dos 512 bps, opera na frequência de 868 MHz, o que lhe assegura poucas possibilidade de interferências, suporta até um milhão de nós em simultâneo, permite alcançar distâncias até 15kms e conetividade móvel a velocidades superiores a 120 km/h. Esta tecnologia é adequada para monitorizar grandes sistemas técnicos ou áreas de difícil acesso, comunicações M2M, medidores inteligentes (leituras remotas e controle remoto do consumidor)), controlo e monitorização de sinais vitais de grandes grupos de pessoas(atletas, socorristas, etc).[71]

NB-IOT

(Narrow-Band/"Faixa Estreita") é uma tecnologia móvel, desenvolvida pela organização 3GPP, como uma tecnologia de comunicações sem fios para IoT, compatível com as tradicionais redes móveis e destinada apenas à transmissão de quantidades muito pequenas de dados. Enquadra-se nas redes de longo alcance (LPWAN).

Principais caraterísticas de uso: implementado sobre a norma IEE802.15.4, baixo consumo energético, baixa latência, topologia em estrela (também pode ser utilizado em comunicações ponto a ponto), alcance de distâncias superiores a 10 Km, taxas de transferência que variam entre os 0.3 e 50 Kbps, opera na frequência 868 MHz (Europa) e permite interligar até 50.000 de nós. oferece conexões móveis de alta qualidade e acesso direto à Internet.[59]

SIGFOX

é uma tecnologia proprietária, que permite implementar redes sem fios em faixa-estreita ou ultra-faixa-estreita. Enquadra-se nas redes de longo alcance (LPWAN).

Principais caraterísticas de uso: baixo consumo, operação nas faixas de frequências de 862 a 928 MHz, cobertura de 3 a 10 km em áreas urbanas e 30 a 50 km, em áreas rurais e taxas de transmissão que variam entre 100 bps e 600 bps. A Rede Sigfox possui restrições no respeita à frequência de envio de dados, o tamanho das mensagens podem no máximo utilizar 12 bytes por pulso, o que dificulta a sua utilização em aplicações que necessitem de grandes transmissões de dados. [113] [112]

	C. Energia	T. Dados	Tipo	Alcance	Frequencia
NFC	mt baixo	424 Kbps	WPAN	10 cm	13,56 MHz
RFID	mt baixo	40 Kbps	WPAN	10 m	125Khz, 13,56Mhz
Bluetooth	baixo	2-3 Mbps	WPAN	50m	2.4 GHz
Bluetooth LE	mt baixo	1 Mbps	WPAN	50m	2.4 GHz
UWB	baixo	1 Gbps	WPAN	10m	6-8.5 GHz
ZigBee	baixo	250 Kbps	WPAN	100m	915MHz/2.4 GHz
Z-Wave	baixo	100 Kbps	WPAN	150m	868/908 MHz
6LowPAN	baixo	250 Kbps	WPAN	100m	2.4 GHz
WiFi/WiFi Direct	alto	100-250Mbps	WLAN	100m	2.4 GHz/5 GHz
Mioty	baixo	512bps	LPWAN	15km	868 MHz
LoRa/LoRaWAN	baixo	50 Kbps	LPWAN	30km	868 MHz/915 MHz
NB-IOT	moderado	250kps	LPWAN	20km	várias
Sifgfox	baixo	100 bps	LPWAN	30km	862-728 MHz
GSM/GPRS	moderado	10/40 Kbps	WAN	35 km	850 MHz/1.9 GHz
LTE	alto	High	WAN	>100 Km	várias
LTE-M	baixo	1 Mbps	LPWAN	>100 Km	várias
WiMax	alto	1 Gbps/365 Mbps	MAN	>100 Km	várias

Comparação de algumas caraterísticas - Redes sem fios

EC-GSM-IoT

Padrão baseado no LTE, um melhoramento da tecnologia GSM, utilizado em aplicativos M2M ou IoT, visando complementar os dispositivos existentes (WPAN). Padronizado pelo 3GPP (3rd Generation Partnership Project), enquadra-se nas redes de longo alcance (LPWAN), permite taxas de transferência de cerca de 40 Kbps, opera na frequência dos 850 MHz/1.9 GHz e oferece uma cobertura de cerca de 15 Km.[117]

LTE e LTE-M

Long-Term Evolution (LTE), tecnologia móvel 4G, padrão é desenvolvido e padronizado pelo 3GPP (3rd Generation Partnership Project) com o objetivo de disponibilizar um padrão para comunicação sem fios de alta velocidade para dispositivos móveis, telefones e terminais de dados, baseados nas tecnologias GSM e UMTS, dando neste caso, prioridade ao tráfego de dados em detrimento do tráfego de voz, suportando velocidades de dados muito mais rápidas que o GSM.

O LTE enquadra-se nas redes de longo alcance (WAN), permite taxas de transferência de cerca de 10 Mbps e opera na frequência dos 20 MHz. O LTE-M é uma abreviação de LTE (Long Term Evolution) Cat-M1, enquadra-se nas LPWAN. É um subconjunto da tecnologia móvel LTE, otimizada para dispositivos de baixa taxa de transferência de dados (até 200Kbps). Utiliza pequenas baterias, o que permite reduzir o consumo de energia comparativamente ao LTE padrão. Esta tecnologia é utilizada em dispositivos da Internet das Coisas que precisam de se ligar diretamente a uma rede móvel 4G. Opera nas frequências entre os 1.4 e os 20 Mhz. [127]

WiMax

O padrão 802.16 oferece taxas de transferência e alcance capazes de competir com as tecnologia por cabo.

A velocidade máxima que o WiMAX nos oferece hoje em dia, é cerca de 1 Gbps para locais fixos e até 365 Mbps para clientes móveis. Permite comunicações sem fios até cerca de 100km[136]. Têm sido adicionados novos melhoramentos no suporte às aplicações M2M, nomeadamente, suporte para vários recursos básicos de M2M, como transmissão de baixa potência. Segundo Lou Frenzel (2017) [37] é uma opção superior para a Internet industrial das coisas.

Complemento LoRa e LoRaWAN

LoRa é uma tecnologia de modulação rádio baseada em *ChirpSpread Spectrum* (CSS) com *Forward Error Correction* (FEC) integrada.*CSS* é uma técnica de espalhamento de espectro que usa toda a largura de banda do canal para transmitir o sinal. *FEC* (protocolo de controlo de erros) é uma técnica usada para controlar erros na transmissão de dados em canais de comunicação não confiáveis ou ruidosos. Um *Chirp* é um sinal no qual a frequência aumenta ou permanente com o tempo.

LoRaWAN é um protocolo de comunicação rádio digital que que foi desenvolvido para a "Internet das coisas" e possui algumas caraterísticas que a tornam ideal para este projeto, tais como baixo consumo de energia, longo alcance e mecanismos de segurança que protegem as transmissões. "LoRaWAN" refere-se à camada de acesso ao meio (segunda camada do modelo OSI), "LoRa" refere-se à camada física de rádio (primeira camada do modelo OSI). A única tecnologia de comunicação, até à data, que verificava todos os requisitos foi a LoRaWAN. Tem longo alcance o que reduz significativamente o número de gateways necessário (2-5Km em ambiente urbano, 15Km em ambiente rural), tem um modo de consumo muito reduzido (Classe A), suporta um grande número de dispositivos por gateway (cerca 7800 por interface de rádio) e é possível ter controlo completo sobre a rede. Estas vantagens sobrepõem-se às suas limitações, tais como possuir uma taxa de transmissão baixa, não suportar o protocolo IP ou necessitar de vários servidores de suporte. A Figura 8-1 mostra um diagrama da arquitetura de rede LoRaWAN.

Dispositivos

A sua ativação pode ser feita de duas maneiras: Utilizando a Ativação pelo ar, OTAA (Over-The-Air Activation), feita quando um dispositivo é implantado ou redefinido; ou pela Activation By Personalization (ABP) "ativação personalizada", ativação feita na etapa de configuração.

Apêndice XII

Protocolos de comunicação aplicados à IoT

Este Apêndice foi escrito para servir de suporte teórico no desenvolvimento da dissertação.

Protocolos IoT da Camada de Transporte

A camada de transporte é responsável pela transferência de dados entre os equipamentos, independente da aplicação, do tipo, topologia ou configuração das redes físicas. Faz a ponte entre a camada aplicacional e física. A camada de nível aplicacional envia os dados contidos nos pacotes para as aplicações de destino, e as camada de nível físico encarrega-se da forma como os dados serão transmitidos pela rede. A transferência de dados nesta camada pode ser realizada mediante a utilização de conexões ou através de datagramas. Os protocolos podem ou não oferecer confiabilidade, garantia de entrega, controle de fluxo, entre outros. Embora existam vários protocolos da camada de transporte, como por exemplo, RTP, DCCP, SCTP ou RSVP, os dois mais utilizados são o UDP e o TCP.

Nas soluções e projetos IoT, o protocolo da camada transporte mais utilizado é o UDP. Esta opção deve-se essencialmente a uma dimensão mais reduzida dos pacotes, comparativamente com o protocolo TCP. O cabeçalho dos pacotes do protocolo TCP utiliza um número mais elevado de bits do que o do UDP. O cabeçalho do protocolo TCP indica o *Sequence number* e o *Acknowledgment Number*, que permitem garantir a ordem e a confirmação de entrega do pacote [41]. Já no cabeçalho do protocolo UDP esses dados não aparecem, uma vez que cada pacote UDP é único e indivisível. A diminuição do número de bits utilizados permite reduzir a largura de banda necessária para transmissão dos pacotes e reduzir as necessidades de processamento no encapsulamento (reduzindo também o consumo energético dos dispositivos).

O Protocolo UDP (*User Datagram Protocol*)

Situa-se na camada de transporte, descrito na RFC 768 e permite à camada de aplicação encaminhar datagramas encapsulados em pacotes IPv4 ou IPv6. Porém, este protocolo não apresenta garantias de entrega de pacotes no destino, não é confiável. Caso sejam necessárias garantias de entrega de pacotes, é necessário proceder à implementação de várias estruturas de controlo, tais como *timeouts*, retransmissões, *acknowledgements*, controlos de fluxo, etc. É um serviço sem conexão, pois não necessita manter um relacionamento entre o cliente e o servidor. Assim, o cliente UDP pode criar um *socket*, enviar dados para um servidor e em seguida enviar um outro datagrama com o mesmo *socket* para um outro servidor. Da mesma forma, através de um único socket é possível a um servidor poder receber informação vinda de diversos clientes. O UDP também fornece os serviços de *broadcast* e *multicast*, permitindo que um único cliente envie pacotes para vários outros clientes da rede [41]. O protocolo UDP não suporta SSL/TLS. Em alternativa, para fornecer segurança nas transferências de dados, utiliza o DTLS, *Datagram Transport Layer Security*. A utilização do protocolo UDP faz-se principalmente nos serviços que não precisam de garantir a chegada de pacotes. *Streamings* de vídeo e serviços de voz sobre IP são exemplos de utilização UDP.

Protocolo TCP (*Transmission Control Protocol)*

O TCP é um protocolo de nível da camada de transporte e é sobre este que assentam a maioria dos protocolos da camada de aplicação, como o SSH, FTP, HTTP. Faz a ponte entre dos dados transferidos entre as camadas físicas e as camadas aplicacionais, e garante confiabilidade, sequência correta de entrega de pacotes e verificação de erros [79].

Embora não seja o mais adequado para projetos IoT, continua a ser muito utilizado neste tipo de projetos. A utilização deste protocolo verifica-se mais na comunicação entre plataformas do que nas comunicações entre dispositivos IoT. A sua utilização deve-se, em grande parte, à grande utilização que se faz da *internet*, que é suportada por este, e ao facto de muitas das plataformas IoT alojadas na *cloud* comunicarem através de APIs HTTP (REST).

Protocolos IoT da Camada de Aplicação

Com a massificação da comunicação entre dispositivos e plataformas *IoT*, foi necessário recorrer à utilização de conceitos e protocolos que exigem baixa largura de banda e baixo processamento de dados. Embora os protocolos mais adaptados para *IoT* sejam o MQTT e o CoAP, existem muitos projetos IoT implementados com recurso ao protocolo HTTP (*Hypertext Transfer Protocol*) e ao conceito RESTful APIs (*Representational State Transfer*). A utilização desta última, deve-se em grande parte à utilização massiva que esta tem

no desenvolvimento das soluções *Web* modernas, justificada pela sua simplicidade e pelo conhecimento já adquirido por parte dos programadores.

Tanto o MQTT como o CoAP são normas abertas, mais adequados para ambientes com restrições de processamento, de largura de banda e de consumo energético, do que o HTTP (REST), que consome mais recursos e necessita de mais recursos por parte dos dispositivos e das redes. O MQTT oferece flexibilidade nos padrões de comunicação e atua apenas como um canal para dados binários, enquanto que o CoAP foi projetado para interoperabilidade com a *Web* [17].

Em projetos *IoT* é comum recorrer-se à utilização de APIs (*Application Programming Interface*) para efetuar e controlar a transferência de informação entre os sensores, *gateways*, atuadores e plataformas *IoT* de controlo/gestão. Uma API é um conjunto de padrões de programação que permitem a construção de aplicativos mais seguros e eficazes. Sendo invisível para os utilizadores e funcionando como suporte às aplicações informáticas, esta interface permite definir comportamentos e permissões de acesso aos sistemas, aumentando a segurança no acesso aos dados e diminuindo os riscos de manipulação de dados [64].

MQTT (*Message Queue Telemetry Transport)*

Desenvolvido pela IBM no final dos anos 90, com base no protocolo TCP/IP, acabou por se tornar um padrão para comunicações IoT. É um protocolo de mensagens com suporte para a comunicação assíncrona e utiliza um modelo de publicação/assinatura [2].

Embora a maioria dos serviços da *Web* sejam executados via HTTP, este padrão tem algumas limitações, nomeadamente o facto de ser síncrono, de ser unidirecional, de ser um-para-um e ser um protocolo pesado com muitos cabeçalhos e regras.

O protocolo MQTT define dois tipos de entidades na rede: um *message broker* e os clientes. O *broker* é um servidor que recebe todas as mensagens dos clientes e, em seguida, direciona essas mensagens para os clientes de destino relevantes. Um cliente é qualquer dispositivo que possa interagir com o *broker* e receber mensagens. A grande vantagem do MQTT é a simplicidade. Não há restrições quanto ao tipo de tópico ou de mensagem que se pode usar. As comunicações do cliente com o *broker* podem ser protegidas, criptografadas através de TLS, ou com um método de criptografia e um mecanismo de atualização de chaves [109].

O *MQTT* suporta tanto a autenticação baseada no nome de utilizador/password, como a autenticação baseada no certificado TLS.

O adaptador tenta autenticar o dispositivo utilizando estes mecanismos na seguinte ordem: certificado e utilizador/palavra-passe.

Na autenticação utilizador/password quando um dispositivo necessita autenticar-se utilizando este mecanismo, tem de fornecer um nome de utilizador e palavra-passe para es-

tabelecer ligação ao *MQTT Broker*, esteverifica as credenciais fornecidas pelo cliente em relação às credenciais registadas no serviço de credenciais.

Quando um dispositivo utiliza um certificado *TLS* para se autenticar, o *MQTT Broker* tenta determinar a quem o dispositivo pertence, com base no DN (*Distiguished Name*) do emissor contido no certificado. Para seja bem-sucedida, o certificado de confiança do servidor necessita ser configurado através de um registo produzido por uma autoridade de certificação de confiança. O certificado do cliente do dispositivo será então validado usando o certificado registado, confirmando assim implicitamente a quem o dispositivo pertence.

Qualidade do Serviço

A qualidade do serviço (QoS) define o balanço entre a performance e a consistência na entrega das mensagens. O nível de qualidade do serviço zero não garante a entrega da mensagem, o nível um garante que a mensagem é entregue pelo menos uma vez podendo ser entregue a mesma mensagem mais do que uma vez (duplicada) e o nível dois garante que a mensagem é entregue apenas uma vez. Foi utilizado o nível dois de qualidade do serviço, de modo a garantir que, por exemplo, as leituras recebidas dos nós sensores e os comandos enviados para os nós sensores cheguem ao destinatário e sejam entregues apenas uma vez.

Tópicos

Os tópicos do protocolo MQTT são grupos de comunicação que podem ser criados pelos clientes. Posteriormente outros clientes podem subscrever a esses tópicos, passando a fazer parte do grupo. Quando uma mensagem é enviada para um determinado tópico, todos os clientes que subscreveram esse tópico vão receber a mensagem. Os tópicos podem ter vários níveis separados por uma barra (/) e suportam carateres "wildcard" que permitem que os clientes subscrevam a todos os tópicos que correspondam à regra. O caractere "+" representa um nível com qualquer nome e o caratere "#" representa qualquer número de níveis com qualquer nome. Os tópicos são criados e utilizados pelo LoRa App Server para enviar e receber mensagens dos nós sensores. Foi criado um tópico para enviar mensagens e outro para receber por cada nó sensor de cada aplicação. O LoRa App Server também publica mensagens de estado dos nós sensores, mensagens de erro dos nós sensores e mensagens de confirmação de receção de mensagens (acknowledgement), sendo possível configurar o "caminho" para esses tópicos. Os tópicos são criados e utilizados automaticamente para a comunicação entre a LoRa Gateway Bridge e o LoRa Server.

COAP (*Constrained Application Protocol*)

Tal como o protocolo HTTP, o CoAP é um protocolo de transferência de dados. Ao contrário do HTTP, o CoAP foi projetado tendo em conta as restrições de capacidade de processamento e de largura de banda. O CoAP é executado sobre UDP. Clientes e servidores comunicam através de datagramas sem ligação. O CoAP permite que *broadcast* e *multicast* UDP sejam utilizados para o endereçamento, seguindo um modelo cliente/servidor. Os clientes fazem solicitações e os servidores devolvem respostas (GET, PUT, POST e DELETE) [64]. O CoAP foi projetado para operar com o HTTP e a *Web REST* por meio de *proxies* simples. Como o CoAP é baseado em datagramas pode ser usado tanto em SMS como noutros protocolos de comunicação baseados em pacotes. A nível de aplicação QoS, as solicitações e mensagens de resposta podem ser marcadas como "*confirmable*" ou "*nonconfirmable*". Mensagens confirmadas são respondidas pelo recetor com um pacote *ack* e mensagens não-confirmadas são esquecidas. Como o CoAP trabalha sobre UDP e não sobre TCP, o SSL/TLS não está disponível para fornecer segurança. Neste caso utiliza-se o DTLS, *Datagram Transport Layer Security*, que fornece as garantias semelhantes ao TLS, mas para transferências de dados por UDP [73]. Embora o CoAP não exija o IPv6, é mais fácil de implementar em ambientes IP, onde os dispositivos são diretamente rastreáveis [64].

WebSockets

Historicamente, desenvolver aplicações *web* que precisem de comunicação bidirecional entre cliente e servidor era significado de um abuso do protocolo HTTP, devido às incessantes requisições ao servidor *web* para verificar se havia algum dado novo, além da criação de duas conexões TCP, uma para enviar mensagens ao cliente e outra para receber mensagens [45]. Uma simples solução para esse problema foi usar uma única conexão TCP para comunicar em ambas direções. É esta basicamente a definição de *WebSockets*. O protocolo *WebSocket* foi projetado para substituir todas as outras tecnologias existentes de comunicação bidirecional na *Web* que usam o protocolo HTTP, principalmente para beneficiar a infraestrutura existente.

HTTP/REST (*Hypertext Transfer Protocol/Representational State Transfer*)

O protocolo HTTP é utilizado para servir o *website*, é utilizada a segunda versão deste protocolo, chamada HTTP/2, mas caso o *web* browser não suporte esta versão o *website* é servido pelo protocolo HTTP 1.1, perdendo os benefícios do protocolo HTTP/2, tais como a multiplexagem da ligação TCP, que consiste em utilizar a mesma ligação para efetuar múltiplos pedidos em paralelo reduzindo a latência e a utilização de recursos, ou

a compressão de cabeçalhos, que reduz o tamanho dos pedidos e respostas reduzindo a utilização de largura de banda da rede.

HTTP é um protocolo da camada de aplicação do modelo OSI utilizado para transferência de dados na rede mundial de computadores, a *Wide Web*, utilizado para transferir dados de hipermédia (imagens, sons e textos). É um protocolo de comunicação massificado pela utilização em transferência de páginas HTML[46]. Para que a transferência de dados seja realizada, o protocolo HTTP necessita estar agregado a outros dois protocolos de rede: TCP (Transmission Control Protocol) e IP (*Internet Protocol*). Embora não possa ser considerado um protocolo IoT, devido às suas necessidades de computação e de largura de banda, mesmo assim verifica-se que existem muitos projetos IoT desenvolvidos com recurso ao protocolo HTTP, principalmente devido a grande quantidade de programadores que trabalham atualmente com o mesmo no desenvolvimento de solução *Web*. O REST, considerado um conceito e não um protocolo, é atualmente a principal base de comunicação com a maior parte das APIs existentes na Internet. Neste sentido, aplicativos que são RESTful são aqueles cujas APIs seguem um conjunto universal de requisitos de arquitetura, de modo que várias linguagens de programação se possam interligar facilmente, através de uma abordagem unificada. As APIs REST trabalham via HTTP para executar ações, como, por exemplo, POST, GET, PUT e DELETE. Estas ações podem ser mapeadas para as funções SQL CREATE, SELECT, UPDATE e DELETE, conhecido como CRUD [41]. O REST desempenha um papel crucial e tornou-se um "protocolo" normalizado, interpretado por quase todos os terminais e servidores *web* [41].

RESUMO

Os protocolos MQTT e CoAP são dois protocolos desenvolvidos e otimizados para ambientes IoT. O protocolo HTTP, embora não se possa considerar adequado a projetos IoT, é muito utilizado nesta área, principalmente devido a sua simplicidade e a grande quantidade de programadores que possuem conhecimentos de desenvolvimento nesta tecnologia, de modo que o REST é considerado um conceito e não um protocolo.

O MQTT é um protocolo de comunicação «muitos-para-muitos» utilizado essencialmente para transmitir mensagens entre vários clientes através de um intermediário central *broker*. Os clientes publicam mensagens e o servidor decide para onde os deve encaminhar, mediante a utilização de rotas.

O CoAP é, principalmente, um protocolo um-para-um criado para transferir informações entre um cliente e um servidor. Embora tenha capacidade para a observação de recursos, o CoAP é mais adequado para um modelo de transferência de estados e não para comunicações baseadas em eventos [18]. Enquanto o HTTP é um protocolo que necessita estar agregado a outros dois protocolos de rede: TCP (Transmission Control Protocol) e IP (Internet Protocol).

As APIs REST trabalham via HTTP para executar ações. Estão massificadas nas

soluções *web* modernas e a transferência de dados pode ser efetuada através de JSON ou XML. REST é um conceito stateless, ou seja, os clientes ligam-se e consomem dados a pedido através da API, transmitindo dados sem que seja necessária uma conexão aberta [132].

Protocolos de Comunicação Serie

Entre outras, na comunicação entre sensores e MCU's é utilizada comunicação série. Este tipo de comunicação pode ser utilizada através de diferentes protocolos. Dois dos principais protocolos são o UART e o I2C [67].

Protocolo UART

O UART (Universal Asynchronous Receiver Transmitter) utiliza uma comunicação assíncrona, o que significa que numa comunicação entre dois dispositivos ambos necessitam de possuir o mesmo ritmo de transmissão de dados. Não é utilizado um clock em comum pelos vários dispositivos. Este tipo de comunicação é caraterizada pela capacidade de todos os dispositivos serem capazes de comunicar entre eles, tanto do ponto de vista de transmissão, como de receção. Através do protocolo UART é possível de se obter uma comunicação intitulada de full-duplex. Neste tipo de comunicação existe um transmissor, TX, e um recetor, RX.

Tabela XII.1: Protocolos de Comunicação Série ([60])

Protocolo	Taxa de comunicação (bps)	Sentido de Transmissão	Método	Nº Fios	Tensão	Nº Max de dispositivos
UART	1200 a 115200	Full-Duplex	Assíncrono	2	0 a 5	1
SPI	0 a 10M (depende do dispositivo)	Full-Duplex	Síncrono	3 + (1 para cada slave)	0 a 5V	não há
I^2C	100k ou 400k	Half-Duplex	Síncrono	2	0 a 5V	127 ou 1024
RS 232	1200 a 115200	Full-Duplex	Síncrono ou Assíncrono	2	-25 a +25	1
1 wire	0 a 16,3k ou 0 a 142k	Half-Duplex	Assíncrono	1	2,8 a 6,0	256
USB	1,5M a 4,8G	Half-Duplex (2.0) ou Full-Duplex (3.0)	Assíncrono	2	0 a 20	127

I2C – Inter-Integrated Circuit

O I2C (Inter-Integrated Circuit) utiliza uma comunicação síncrona, e utiliza um clock em comum por todos os dispositivos do sistema. Utiliza um bus de 2 cabos, o SCL (Serial Clock Line) e o SDA (Serial Data Line), ao qual cada dispositivo está ligado em paralelo. As comunicações são do tipo master-slave, no entanto pode existir mais do que um só master. Cada dispositivo possui um id pré-definido, de forma a que o master saiba com quem comunicar ...

Vantagem: possibilita comunicar com vários dispositivos utilizando poucos fios, além de possibilitar que mais de um mestre controle os escravos.

Funcionamento:
Ligação: este protocolo utiliza apenas dois pinos, SDA que é o sinal de dados e SCL o clock. É possível concluir que este protocolo é *half-duplex*, porque contém apenas um pino para envio de dados, e síncrono, pois usa um pino de clock.

SPI – Serial Peripheral Interface

Vantagem: não há limite para o número de escravos, a comunicação é full-duplex e possui boa velocidade de comunicação.
Funcionamento: em primeiro lugar o mestre gera um *clock* e seleciona através do pino SS com qual dispositivo será efetuada a comunicação. Em seguida os dados são enviados para o dispositivo de destino pelo pino MOSI e então, o dispositivo escravo envia uma resposta (se necessário) ao mestre pelo pino MISO.
MOSI – Master Output Slave Input
MISO – Master Input Slave Output
Ligação: existem dois tipos de ligação para o protocolo SPI, ligação paralela e ligação em cascata.

RS 232

Permite comunicar a maior distância do que protocolo UART, além de ser um protocolo full-duplex. Funciona de forma similar ao protocolo UART, ou seja, é enviado 1 start bit + 8 bits de dados + 1 stop bit. A diferença está nas tensões que o protocolo utiliza e a possibilidade de adicionar mais pinos para verificar as informações enviadas.

1-Wire

Vantagem: utiliza apenas 1 pino para a comunicação, cada dispositivo tem um endereço único.
Funcionamento: este protocolo funciona em 3 fases. A primeira fase é responsável por habilitar a comunicação e identificar os escravos ligados, a segunda fase seleciona qual escravo receberá os comandos e, por fim, na terceira fase ocorre a leitura ou escrita de dados.
Ligação: pode ser feita de duas formas, parasita ou com alimentação externa.

Apêndice XIII

Realização do Sistema

Este apêndice tem como objetivo ilustrar e completar informação apresentada no Capítulo 5.

Tecnologia de Contentores

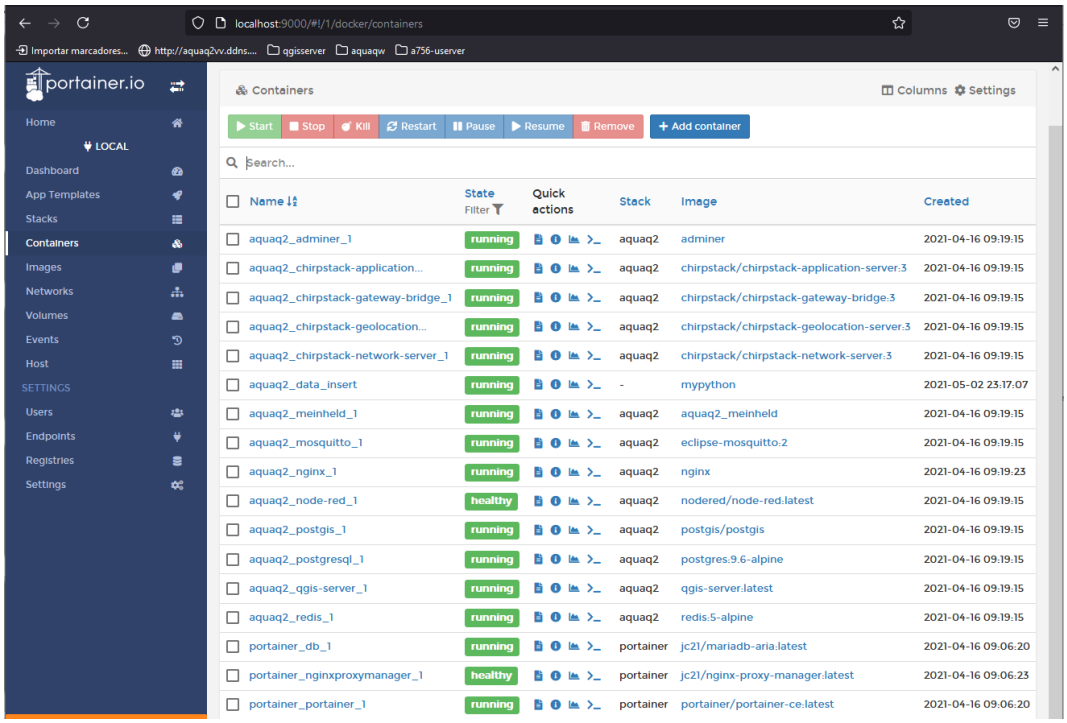


Figura XIII.1: Visualização através do *Portainer* da lista de todos os *contentores Docker* que compõem o sistema. A partir daqui, é possível realizar várias operações, por exemplo: parar, apagar, reiniciar qualquer contentor.

Chirpstack

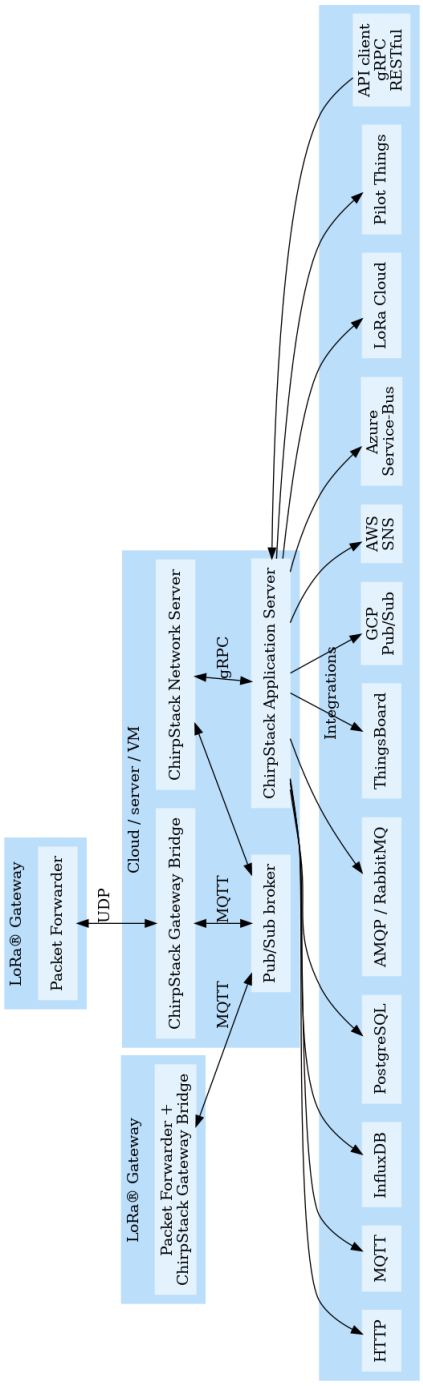


Figura XIII.2: Diagrama LoRaWAN - Diagrama original da empresa *Semtech*, constituído de uma forma geral pelos componentes abordados no Capítulo 4.

Fonte: (https://www.semtech.com/lora/what-is-lora)

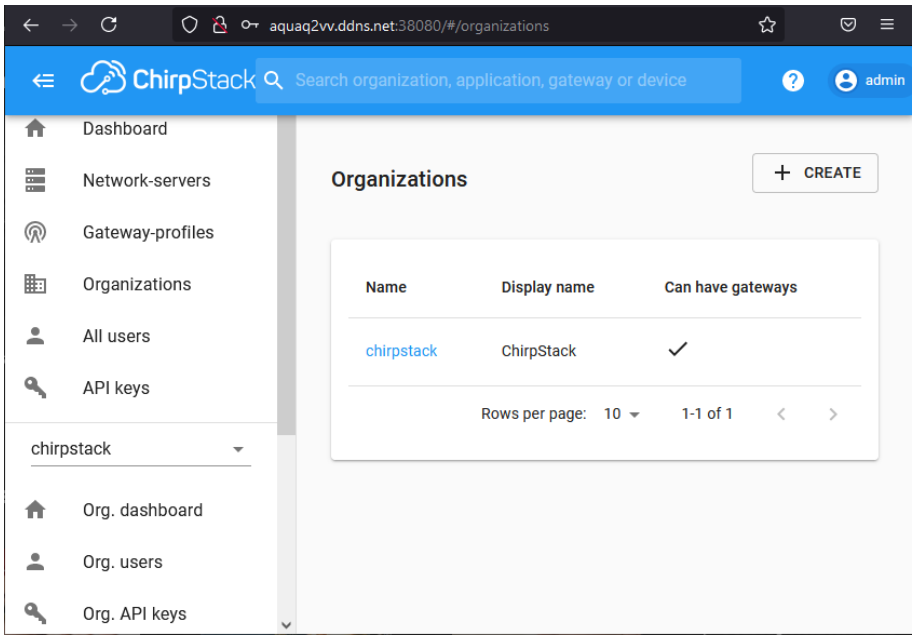


Figura XIII.3: Interface *LoRa Application Server* - Permite configurar e monitorizar todos componentes do sistema *Chirpstack*

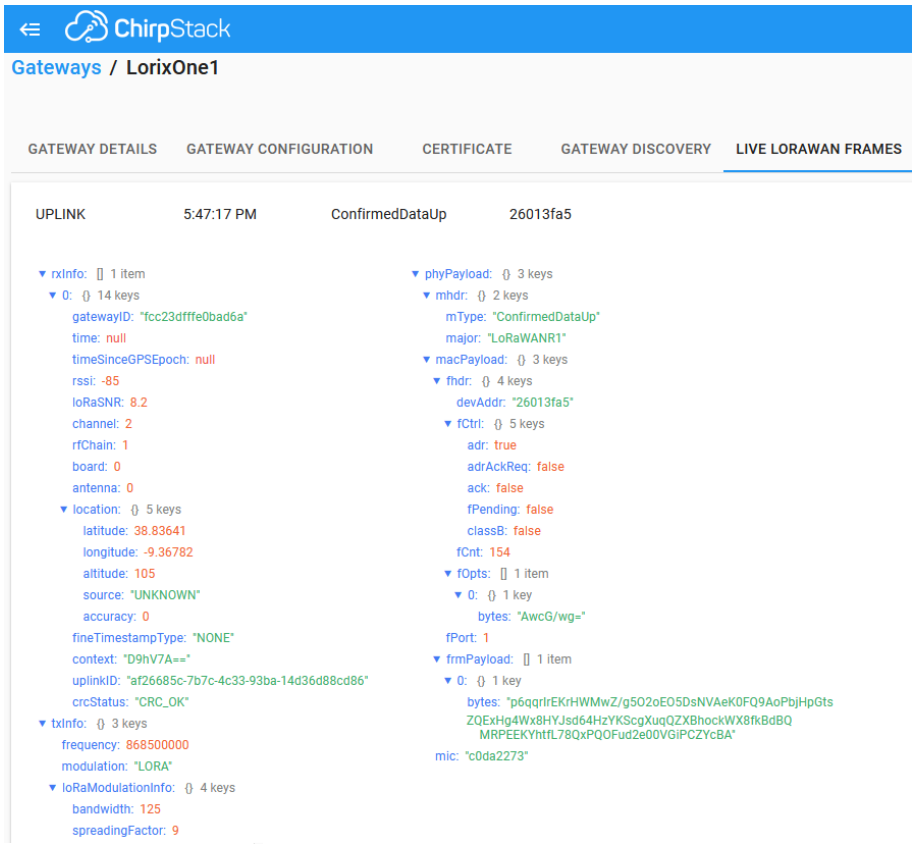


Figura XIII.4: Live LoRaWAN Frames

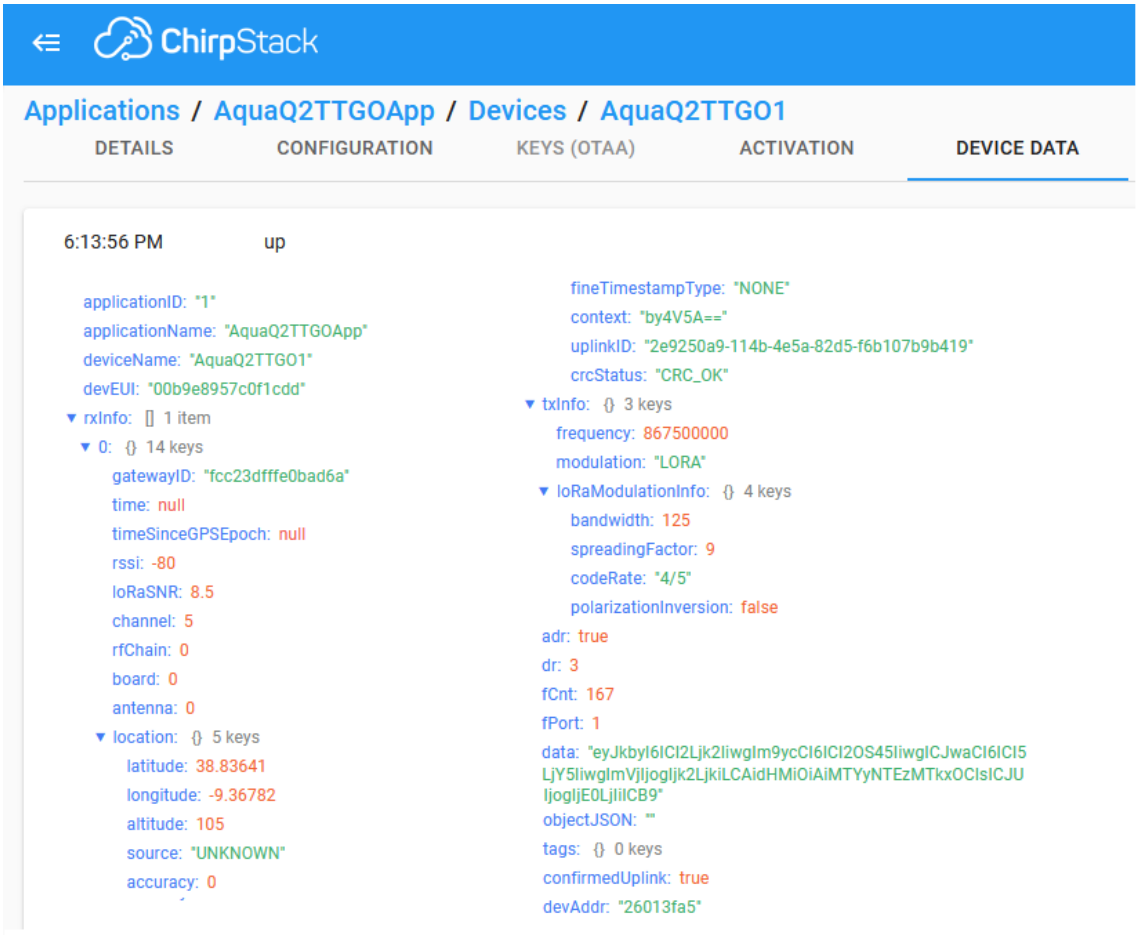


Figura XIII.5: Dados do dispositivo de aquisição de dados de qualidade da água

Base de Dados do Sistema LoRa Chirpstack

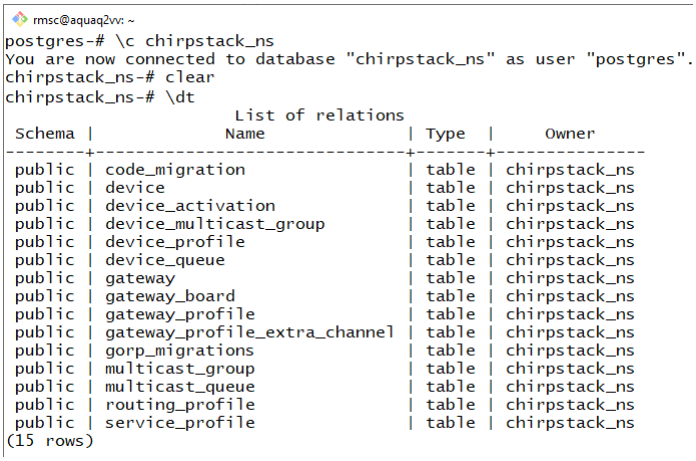


Figura XIII.6: SGBD PostgreSQL utilizado pelo LoRa Application Server

```
rmsc@aquaq2vv: ~
postgres=# \c chirpstack_as
You are now connected to database "chirpstack_as" as user "postgres".
chirpstack_as=# \dt
                    List of relations
 Schema |          Name           | Type  |     Owner
--------+-------------------------+-------+---------------
 public | api_key                 | table | chirpstack_as
 public | application             | table | chirpstack_as
 public | code_migration          | table | chirpstack_as
 public | device                  | table | chirpstack_as
 public | device_keys             | table | chirpstack_as
 public | device_multicast_group  | table | chirpstack_as
 public | device_profile          | table | chirpstack_as
 public | gateway                 | table | chirpstack_as
 public | gateway_ping            | table | chirpstack_as
 public | gateway_ping_rx         | table | chirpstack_as
 public | gateway_profile         | table | chirpstack_as
 public | gorp_migrations         | table | chirpstack_as
 public | integration             | table | chirpstack_as
 public | multicast_group         | table | chirpstack_as
 public | network_server          | table | chirpstack_as
 public | organization            | table | chirpstack_as
 public | organization_user       | table | chirpstack_as
 public | service_profile         | table | chirpstack_as
 public | user                    | table | chirpstack_as
(19 rows)
```

Figura XIII.7: SGBD PostgreSQL utilizado pelo LoRa Application Server

```
rmsc@aquaq2vv: ~
127.0.0.1:6379> client list
id=3 addr=172.19.0.6:33134 fd=8 name= age=72422 idle=6
id=4 addr=172.19.0.9:56228 fd=9 name= age=72422 idle=6
id=5 addr=127.0.0.1:37862 fd=10 name= age=33 idle=0
```

Figura XIII.8: SGBD Redis - Lista de Clientes. O *LoRa Network Server* (172.19.0.12) e o *LoRa Application Server* (172.19.0.4).

```
rmsc@aquaq2vv: ~
rmsc@aquaq2vv:~$ docker exec -it aquaq2_redis_1 redis-cli --stat
------- data ------ -------------------- load --------------------
keys       mem      clients blocked requests            connections
276        939.39K  3       0       73210 (+1)          4
277        1019.48K 3       0       73227 (+17)         4
276        939.30K  3       0       73242 (+15)         4
275        939.19K  3       0       73243 (+1)          4
275        939.19K  3       0       73244 (+1)          4
```

Figura XIII.9: SGBD - Estatísticas do SGBD Redis.