

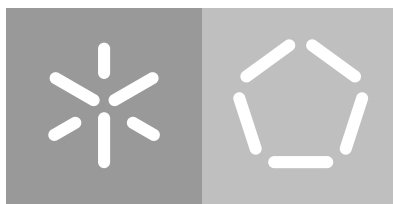
Universidade do Minho

Escola de Engenharia

Departamento de Eletrónica Industrial

Tiago Teixeira da Costa

Estação de Monitorização de Parâmetros de Qualidade da Água



Universidade do Minho

Escola de Engenharia

Departamento de Eletrónica Industrial

Tiago Teixeira da Costa

Estação de Monitorização de Parâmetros de Qualidade da Água

Dissertação de Mestrado

Mestrado Integrado em Engenharia Eletrónica
Industrial e Computadores

Trabalho efetuado sob a orientação do

Professor Doutor Paulo Cardoso

Professor Doutor António Duarte

Fevereiro de 2021

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos. Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho



**Atribuição
CC BY**

<https://creativecommons.org/licenses/by/4.0/>

Agradecimentos

Ao meu orientador, o Professor Paulo Cardoso, pela disponibilidade que sempre mostrou para mim, desde o momento em que pedi essa mesma orientação, pela ajuda na elaboração desta dissertação, e por uma especial aula no 1º ano, que marcou o meu percurso universitário.

Ao Professor António Duarte pelo lançamento do tema que guiou a dissertação, e pelos conhecimentos transmitidos sobre o assunto em questão.

Aos meus pais e ao meu irmão, pelo encorajamento constante ao longo do curso e pela inspiração que foram e são para mim.

À minha namorada, pelo apoio incondicional, pela paciência e pela força que sempre me transmitiu.

Ao Bruno, ao Henrique, ao Miguel e à Sandra, por terem sido muito mais do que meros colegas, e terem sido essenciais para o meu trajeto.

Todos foram importantes, de algum modo ou de outro, para que conseguisse atingir os meus objetivos, e a todos deixo o meu muito obrigado.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração. Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Resumo

Como sempre se soube ao longo dos tempos, a água é dos bens mais essenciais do planeta, tanto para o ser humano como para todos os animais que o rodeiam. A água faz funcionar ecossistemas inteiros e a sua ausência, ou o seu dano, pode destruir estes mesmos.

Mais recentemente, devido a factores como o crescimento das indústrias, a poluição que delas advém, a sobre-população do planeta, o sobre-uso dos recursos, entre outros, a qualidade da água tem decrescido e a sua possível futura escassez é vista como um problema.

Para evitar que a qualidade da água seja deteriorada, deve-se monitorizar esta mesma, de modo a ser possível agir quando necessário. Os processos existentes de análise da qualidade da água são executados em laboratório, o que requer recolha manual da amostra, disponibilidade temporal e financeira.

São, portanto, métodos complicados, numa época em que se tenta que nada o seja. Em plena era do fenómeno "smart", onde existem tecnologias IoT, sensorização, comunicação e processamento dos dados, deseja-se que algo tão importante como avaliar a qualidade da água seja algo simples, como tantas outras coisas, que outrora não eram, são.

A proposta passa pelo desenvolvimento do protótipo de um dispositivo IoT, de baixo consumo energético e de baixo custo, que analise variáveis da qualidade da água em tempo real, comunicando as mesmas para um outro dispositivo, que servirá de central de comando, com uma base de dados para o registo dos valores e das suas evoluções ao longo do tempo, com representação gráfica das mesmas, e também emitindo alertas, para tornar imediata a reação a variações indesejadas nas características da água.

Palavras-chave: Água; Central de comando; IoT; Sensorização;

Abstract

As it has always been known through the ages, water is one of the most essential things on the planet, not only to the human race but also to all the animals that surround it. Water makes entire ecosystems work, and its absence, or damage, can destroy those same ecosystems.

More recently, due to factors such as industry growth, the pollution that comes from it, overpopulation, overuse of natural resources, among others, water quality has seen a downfall and its possible future scarcity is seen as an issue.

To avoid water quality deterioration, one should monitorize it in order to make it possible to act when necessary. Current existing processes to make that analysis are executed in a laboratory, which requires manual collection, besides money and time availability.

These are, then, complicated methods, in an age where the goal is to make sure nothing is complicated. In the era of the "smart" phenomenon, when IoT technologies are available, sensorization, communication and data processing, one wishes that something as important as evaluating water quality is made simple, like so many things that once weren't but now they are.

The proposition is then to develop a prototype of an IoT device, with low power consumption and low cost, which analyses variables of water quality in real time, communicating the measurements in real time, wirelessly, to another device which will act as a command center, with a database to log the values and their evolution through time, and representing them graphically, and sending alerts to make possible an immediate reaction to unwanted fluctuations in the characteristics of the water.

Keywords: Command center; IoT; Sensorization; Water;

Conteúdo

1	Introdução	1
1.1	Contextualização	1
1.2	Motivação e objetivos	2
1.3	Organização da dissertação	2
2	Estado da arte	4
2.1	Monitorização da qualidade da água	4
2.2	Sistemas de computação	7
2.2.1	Sistemas embebidos	7
2.2.2	Internet Of Things (IoT)	8
2.2.3	Cyber-physical Systems (CPS)	8
2.2.4	Microcontroladores	9
2.2.5	Fonte de energia	11
2.2.6	Comunicação sem fios	12
2.3	Sensores para a monitorização de parâmetros da água	14
2.3.1	Sensor de pH	14
2.3.2	Sensor do TDS da água	15
2.3.3	Sensor de temperatura	15
2.3.4	Sensor de turbidez	16
2.3.5	Sensor fotoelétrico do nível de água	17
2.3.6	Sensor de pluviosidade	17
3	Análise	18
3.1	Estudo do problema	18
3.2	Casos de uso	19
3.3	Diagramas de sequência	20
3.4	Arquitetura do sistema	22
4	Design	23
4.1	Sensores	23
4.2	Fonte de energia	24
4.3	Comunicação	26
4.4	Sistema de computação	27
4.5	Central de comando	29
4.6	Arquitetura do sistema	30

5	Implementação e testes	32
5.1	Sensores	32
5.2	Fonte de energia	33
5.3	Comunicação	34
5.4	Sistema de computação	35
5.5	Estrutura física	37
5.6	Central de comando	38
5.7	Testes de integração	42
6	Conclusões	45
6.1	Projeto e solução	45
6.2	Discussão	45
6.3	Soluções de futuro	46
7	Bibliografia	47
a	Anexo	50

Lista de Figuras

Figura 2.1	Arduino Uno.	9
Figura 2.2	STM32F103C8T6.	10
Figura 2.3	Raspberry Pi 4.	11
Figura 2.4	Sensor de pH.	14
Figura 2.5	Sensor do TDS da água.	15
Figura 2.6	Sensor de temperatura à prova de água.	16
Figura 2.7	Sensor de turbidez analógico.	16
Figura 2.8	Sensor de nível.	17
Figura 2.9	Sensor de pluviosidade.	17
Figura 3.1	Casos de uso.	19
Figura 3.2	Diagrama de sequência.	20
Figura 3.3	Diagrama de sequência.	21
Figura 3.4	Diagrama de sequência.	21
Figura 3.5	Arquitetura abstrata de sistema IoT.	22
Figura 4.1	Painel solar.	25
Figura 4.2	Módulo de alimentação.	25
Figura 4.3	Módulo SIM808	26
Figura 4.4	STM32 Nucleo-144 development board.	27
Figura 4.5	Arquitetura de software do sistema de computação.	28
Figura 4.6	Fluxograma do sistema de computação.	28
Figura 4.7	Interface gráfica.	29
Figura 4.8	Fluxograma da central de comando.	30
Figura 4.9	Arquitetura final do sistema.	31
Figura 5.1	Sensores utilizados na implementação.	32
Figura 5.2	Painel solar e módulo de alimentação.	33
Figura 5.3	Módulo Bluetooth.	34
Figura 5.4	Resultado do teste de envio de mensagem.	35
Figura 5.5	Tasks criadas.	36
Figura 5.6	Árvore de ficheiros.	37
Figura 5.7	Projeto de modelo 3D.	38
Figura 5.8	Interface.	39
Figura 5.9	Menu de acesso.	40

Figura 5.10	Menu de edição de utilizadores.	40
Figura 5.11	Menu de edição dos valores de alarme.	41
Figura 5.12	E-mail de exportação.	41
Figura 5.13	E-mail de alerta.	41
Figura 5.14	Interface.	42
Figura 5.15	Comunicação com a central de comando.	43
Figura 5.16	Consumo energético em modo ativo.	43
Figura 5.17	Consumo energético em poupança de bateria.	43

Lista de Tabelas

Tabela 1	Características de projetos de monitorização.	7
Tabela 2	Análise comparativa de 3 microcontroladores.	11
Tabela 3	Análise comparativa de meios de transmissão.	13
Tabela 4	Requisitos do projeto.	19
Tabela 5	Tabela da Organização Mundial de Saúde.	23
Tabela 6	Lista de sensores.	24
Tabela 7	Planos de testes unitários dos sensores.	24
Tabela 8	Plano de testes da fonte de energia.	25
Tabela 9	Plano de testes do módulo de comunicação.	26
Tabela 10	Resultados dos testes unitários aos sensores.	33
Tabela 11	Resultado dos testes da fonte de energia.	34
Tabela 12	Resultado dos testes ao módulo de comunicação.	34
Tabela 13	Pinos de ligação entre os componentes e o microcontrolador.	36

Lista de Abreviaturas

IoT Internet Of Things

SoC System on Chip

ISR Interrupt Service Routine

RFID Radio-Frequency IDentification

NFC Near Field Communication

CPS Cyber-Physical System

TDS Total Dissolved Solids

WQI Water Quality Index

Capítulo 1

Introdução

Este primeiro capítulo serve para introduzir a dissertação. Inicialmente o problema é contextualizado, seguido pela motivação para resolver o problema e os objetivos que poderão ajudar a isso. Por fim é descrita a estrutura do presente documento.

1.1 Contextualização

Os computadores e os processadores não estão só fechados dentro de si. Hoje em dia, são portas para o mundo, em ligação com o exterior.

A Internet of Things está em claro crescimento e proliferação. Este crescimento verifica-se em vários aspetos. Existe crescimento ao nível do quão amplamente os dispositivos são aplicados, ao nível da capacidade, ao nível da fiabilidade, e num ponto de vista mais estatístico, ao nível do número de unidades existentes.

Para o desenvolvimento de um dispositivo do género, existem bastantes variáveis a decidir, e para cada variável existe um número de opções sempre em crescimento, em constante evolução e atualização.

Todavia, existem campos onde a força do IoT ainda não chegou. Um dos campos onde profissionais da área denotam essa ausência é, precisamente, a hidrologia.

Parte essencial do IoT é a sensorização e a comunicação. Existem cada vez mais sensores para cada vez mais variáveis, e a sua implementação torna-se incrementalmente mais fácil. As possibilidades de comunicação aumentam e tecnologias como a rede móvel e o Bluetooth estão mais evoluídas do que, por exemplo, há 10 anos.

Por fim, o processador. A capacidade de processamento aumentou e o tamanho dos processadores diminuiu. Estes dispositivos estão cada vez mais pequenos, o que está diretamente ligado à sua ubiquidade e massificação.

Sendo uma tecnologia tão promissora, pois permite potencialmente que todos os objetos estejam conectados, aumentando o conforto humano, é possível prever que o futuro passe em grande parte por ela.

1.2 Motivação e objetivos

O elevadíssimo valor da água para o Homem nunca foi questionável. A qualidade da água, dos rios e outros corpos de água menores, é importante para os ecossistemas onde se inserem, e para as indústrias que dela necessitam. Além do mais, é a água que acaba nas torneiras das casas.

Portanto, é importante que esta seja constantemente acompanhada, sendo desejável que isso aconteça em tempo real, de modo não dependente de recolha e análise em laboratório. Este acompanhamento deve servir para ser possível vislumbrar desníveis e valores fora do que se considera desejável e ideal. Possivelmente até enviando alertas, caso certos números sejam atingidos, de modo a ser possível uma rápida intervenção.

A isto, junta-se o mundo da tecnologia que permite este acompanhamento, que já foi brevemente introduzida em [1.1](#), e que será explorada ao longo de todo o documento, tendo foco no tema em mãos.

O objetivo é o de desenvolver uma estação de monitorização de parâmetros da qualidade da água. Para tal, as etapas passam primeiro pela identificação e descrição do que requer um sistema de monitorização e alerta do género. Depois disto, sabendo necessariamente que o sistema irá conter sensorização, segue-se o estudo do mundo dos sensores, focados na análise da água, e também das soluções computacionais que conseguiriam responder aos requisitos identificados para o sistema.

1.3 Organização da dissertação

No presente capítulo é feita a introdução, onde o tema é contextualizado, bem como o que motivou a dissertação e os objetivos da mesma, e o presente texto que explica a organização do documento.

No capítulo [2](#) é feito um levantamento do estado da arte, contextualizando conceitos e iniciando os primeiros passos para atingir o objetivo final, com o estudo dos sensores existentes no mercado e soluções computacionais, bem como tudo o que circunda um sistema como o que se ambiciona desenvolver.

Depois, no capítulo [3](#), segue-se a análise de tudo o que foi encontrado, listando as características do sistema, começando a idealizar de um modo abstrato, e delineando comportamentos que este deve ter.

De seguida, no capítulo [4](#), o sistema é desenhado com base na análise feita, são escolhidos os componentes que o sistema terá e como estes interagem entre si. O modo como o dispositivo e a central de comando serão implementados é detalhado.

Concluído o desenho dos componentes, no capítulo [5](#) é feita a implementação, com os respetivos testes. Esta implementação será feita em ambiente de laboratório e serve para se testar a funcionalidade e pertinência do sistema.

O capítulo 6, por fim, contém um resumo do trabalho desenvolvido, apreciações acerca do projeto e ideias de trabalho de futuro.

Capítulo 2

Estado da Arte

Ao longo deste capítulo será feito um levantamento tanto do significado de termos importantes para o projeto como de projetos e sensores existentes no mercado e no mundo acadêmico.

O capítulo começa com uma listagem de projetos existentes, cujo objetivo é o de poder fornecer uma noção de quais são as variáveis normalmente observadas, bem como as plataformas de computação e os meios de comunicação.

De seguida foi feita uma explicação de alguns termos, como os sistemas de computação, sistemas embebidos, IoT(Internet of Things) e CPS(Cyber-physical systems). São abordadas algumas alternativas de microcontroladores, de fontes de energia e de comunicação sem fios, para depois serem tomadas as decisões de design com base nessas recolhas.

Também para servir de base para a escolha do design, foram analisados alguns parâmetros da água, o que são, o que representam, como são avaliados e que sensores existem para essa avaliação.

2.1 Monitorização da qualidade da água

A monitorização da qualidade da água é essencial para a análise de um recurso importantíssimo como é a água. Devido a isso, já múltiplos projetos académicos e profissionais abordaram este tema. Serão, ao longo desta secção, caracterizados vários desses projetos, cuja função e objectivo é a análise de corpos de água dos mais variados géneros.

Em [1], com base na importância da Internet of Things, a sensorização remota, e a sua pluralidade de utilização, foi desenvolvido um projeto. Devido ao aumento da produção industrial a um nível global e ao sobre-uso dos recursos do planeta, a qualidade da água disponível tem deteriorado. O trabalho de investigação dos autores advém desta premissa, tendo então sido implementado um sistema de monitorização da qualidade da água, usando IoT e sensorização remota, nas ilhas Fiji, local onde foi desenvolvido. As variáveis observadas pelo sistema foram a temperatura, o pH e TDS.

Além dos sensores necessários à parametrização das variáveis, também foi usado, em termos de hardware, um conversor de analógico para digital (ADC), um microcontrolador não especificado, um cartão SD para armazenamento local dos valores e um módulo GSM.

O dispositivo oferece a possibilidade de apenas armazenar as recolhas em modo offline através do cartão SD, ou então enviar para um servidor FTP (File Transfer Protocol). O sistema incorpora, como é natural, modos de poupança de bateria, para estender o tempo de vida do mesmo, conservando a sua carga. Também os módulos, como o de comunicação com o cartão SD, são desativados quando não estão em uso.

Em [2], a premissa inicial é a de que é inegável a importância da monitorização da qualidade da água, tendo em conta o impacto que esta tem na saúde humana e nos ecossistemas. A ideia do projeto era a de desenvolver um sistema sem fios que assiste na monitorização contínua da condição da água tendo por base as variáveis listadas abaixo. Os sensores estão conectados a um microcontrolador que transmite os dados para uma base de dados, através de um módulo Wi-Fi. Com base nos parâmetros, a ideia passa depois por classificar a água usando um índice, o WQI (Water Quality Index). Para tal, foram observados o pH e a turbidez da água.

O microcontrolador utilizado foi o WeMos D1 R2, similar ao Arduino Uno, com a excepção de que tem um módulo Wi-Fi ESP-12F. Este é um módulo do tipo ESP8266 com um protocolo TCP/IP integrado que permite a qualquer microcontrolador aceder a uma rede Wi-Fi. Cada módulo ESP8266 vem pré-programado com firmware, que o permite ser utilizado sem qualquer programação, num dispositivo Arduino.

Em [3], serviu de motivação a estimativa de que cerca de 20 milhões de pessoas no Bangladesh não têm acesso a água potável. Foram escolhidos critérios, a temperatura, a turbidez, o pH e o oxigénio dissolvido, com grande impacto na qualidade da água, sendo que a velocidade deste tipo de tecnologia tem vantagem em relação ao método tradicional de colheita manual e análise em laboratório.

O modelo desenvolvido detecta a poluição remotamente, podendo-se depois tomar as decisões adequadas. Foi também criada uma rede de protótipos, com comunicação para uma base de dados, que contém visualização gráfica.

O sistema oferece também o envio de mensagens de texto, na eventualidade da existência de qualquer tipo de valor fora dos limites estabelecidos, de forma a permitir uma mais rápida resposta.

Foi utilizado o Arduino Mega 2580 pois este oferece os pinos, tanto digitais como analógicos, suficientes. Em termos de comunicação, foi utilizado o módulo SIM 808 GSM/GPRS. A conexão com o microcontrolador é simples, através de comandos AT e por meio da UART.

Em [4] partiu-se do problema da poluição das águas e do seu efeito nocivo na saúde. Na atualidade, a detecção de impurezas é feita em laboratório, e portanto, a utilização de dispositivos IoT corta esse gasto de tempo e de material, sendo um processo automatizado. As variáveis, temperatura, pH e condutividade, são recolhidas por Arduino e enviadas por mensagem de texto, através da tecnologia GSM. O hardware utilizado, como o módulo GSM e o modelo de Arduino, não é especificado.

O projeto desenvolvido em [5] ambiciona ser de baixo custo, em tempo real e IoT. As variáveis sensoriais, de temperatura e pH, são colectadas por um controlador e enviadas através de um módulo especializado. Os dados ficam depois armazenados num servidor acessível através de endereço IP.

O microcontrolador utilizado é o LPC2148, que é baseado num CPU ARM7TDMI-S de 32 bits. A escolha foi feita devido ao seu pequeno volume e baixo consumo energético. O módulo de comunicação Wi-Fi escolhido foi, novamente, o ESP8266.

Em [6] é apresentado um sistema de monitorização da qualidade da água multi-paramétrico, desenvolvido na Universidade de Bristol e desenhado para o porto da cidade e demonstra, com sucesso, a executabilidade da recolha em tempo real e em alta frequência de dados da qualidade da água, sendo exibidos online em tempo real. O sistema tem também em aberto a possibilidade de uma maior integração no sistema urbano de gestão de água, melhorando desse modo a sua eficiência. As variáveis observadas são a temperatura, o pH, o oxigénio dissolvido, a TDS e a turbidez.

É também importante observar que o dispositivo é alimentado a energia solar, através da utilização de painéis, permitindo reduzir a necessidade de intervenção humana, e a comunicação é feita por Wi-Fi através da rede pública da cidade.

Em [7] a finalidade foi a de desenvolver um sistema de monitorização de água, em tempo real, para monitorizar qualidade e quantidade. Está, então, o dispositivo dividido em duas partes: um medidor de quantidade e um medidor de qualidade. O medidor de quantidade tem como objectivo monitorizar a quantidade de água consumida por uma residência, notificando tanto o consumidor como a autoridade competente. O medidor de qualidade, por sua vez, verifica a pureza da água recebida pelo consumidor, avaliando os parâmetros de temperatura, pH, turbidez e condutividade. Este sistema garante prevenir riscos para a saúde do utilizador, bem como potenciais ameaças de contaminação por uma terceira parte. Um sistema online oferece todas estas medições para visualização em tempo real. Quaisquer violações, tanto em limites de utilização ou ao nível da qualidade da água, é imediatamente notificada por SMS às partes interessadas, e um sinal de alerta é gerado pelo sistema.

O sistema de medição de quantidade é implementado numa Raspberry Pi 3B, enquanto que a avaliação da qualidade é feita por um Arduino Uno, conectado em série com a Raspberry Pi. A comunicação com a Internet é feita através do módulo Wi-Fi da Raspberry.

A seguinte tabela agrega as informações recolhidas dos vários projetos analisados. As variáveis são sempre de um grupo de 5, temperatura, turbidez, pH, TDS e oxigénio dissolvido, variando de projeto para projeto as escolhas dos autores. Também é possível ver que a escolha principal de microcontrolador é o Arduino, e a comunicação de dados divide-se entre GSM e Wi-Fi.

	[1]	[2]	[3]	[4]	[5]	[6]	[7]
Temperatura	X		X	X	X	X	X
Turbidez		X	X			X	X
pH	X	X	X	X	X		X
TDS	X			X		X	X
Oxigénio dissolvido			X			X	
Controlador	-	Arduino	Arduino	Arduino	LPC 2148	-	Raspberry /Arduino
Transmissão	GSM	Wi-Fi	GSM	Wi-Fi	Wi-Fi	Wi-Fi	Wi-Fi

Tabela 1: Características de projetos de monitorização.

2.2 Sistemas de computação

Um sistema de computação engloba duas partes, o hardware e software. O hardware é um conjunto de dispositivos que possuem capacidades para executar um programa, software, processando informações.

Ao longo da secção encontrar-se-á um detalhar dos vários componentes relacionados com os sistemas de computação, já com relativa orientação para o projeto.

2.2.1 Sistemas embebidos

Os sistemas embebidos são uma combinação de partes de hardware e de software, bem como outros componentes que se juntam originando produtos como telemóveis, leitores de música, *routers* de rede, ou até mesmo sistemas de orientação de aviões [8].

Os autores de [9] definem um sistema embebido como um sistema computador especializado que é normalmente integrado como parte de um sistema maior. Consiste numa combinação de componentes de hardware e software de modo a formar um motor computacional que irá executar uma função específica. Ao contrário dos sistemas de *desktop* que são desenhados para uma função geral, os sistemas embebidos são restringidos na sua aplicação.

O aspecto da integração dos sistemas embebidos em sistemas maiores é corroborada por Grischa Liebel [10], acrescentando ainda que são utilizados para controlar ou monitorizar diretamente esse mesmo sistema onde se inserem, através de dispositivos de hardware específicos.

Os sistemas embebidos têm sido utilizados na indústria automóvel, nos sistemas industriais automatizados, na construção, no treino da automação, na aviação, entre muitos outros campos, diz Richard Zurawski [11]. Por exemplo, têm surgido tendências para o uso de sistemas embebidos na área da automatização industrial para implementar complexos dispositivos inteligentes específicos da área que integram sensores e atuadores através da conversão de sinal, processamento de

senal e de dados, e funções de comunicação. O crescimento dos sistemas embebidos também se tem verificado bastante na indústria automóvel. Na era moderna, os veículos contêm centenas de microcontroladores.

2.2.2 Internet Of Things (IoT)

A IoT (Internet of Things) é a interconexão em rede de objetos físicos, capazes de executar recolha, com a tecnologia própria, e transmissão de dados para outros dispositivos, através da Internet[12]. Devido ao rápido crescimento das tecnologias associadas, como os microcontroladores e os sensores, a IoT tem aberto oportunidades para cada vez mais aplicabilidades, tendo vindo a ganhar atenção por parte de investigadores e profissionais em todo o mundo, e vindo também a ganhar rapidamente terreno no cenário das modernas telecomunicações sem fios[13].

A força maior da era IoT é o alto impacto que tem no dia a dia. Do ponto de vista de um utilizador particular, a introdução da IoT pode ser visível tanto no mundo do trabalho como num ambiente doméstico. Neste contexto, a domótica, vida assistida, e-saúde, aprendizagem aprimorada, são apenas alguns dos exemplos de possíveis aplicações nas quais este paradigma terá um papel principal no futuro próximo. De um modo similar, da perspectiva das empresas, as maiores consequências aparentes serão visíveis em campos como a automação industrial, logística, gestão de negócio/processo[14].

Uma das mais estudadas aplicações da IoT são as cidades inteligentes. As cidades ambicionam para atingir objetivos de desenvolvimento sócio-económico e qualidade de vida. O conceito de cidades inteligentes poderá ajudar na resposta a esse desafio[15]. O propósito da IoT, neste contexto, é o de recolher dados que permitam aos governantes, com base nesses dados, desenhar uma melhor utilização dos recursos públicos, aumentando a qualidade dos serviços oferecidos aos cidadãos[13]. O paradigma IoT é, portanto, benéfico e particularmente atrativo para administrações locais e regionais que o adotem num estado inicial, ganhando características tecnológicas de ponta, e servindo como catalisadores para a adoção do paradigma numa escala maior[16].

2.2.3 Cyber-physical Systems (CPS)

O termo CPS (Cyber-Physical Systems) deriva da palavra "cibernética", que se trata da junção de controlo, computação e comunicação. CPS é a ligação entre o ciber-espço e os processos físicos [17].

Em 2009, Wayne Wolf [18] teorizava acerca da futura popularização do termo, prevendo que se ouviria falar nele no futuro. Considerou que os sistemas CPS seriam o próximo passo da computação, e definiu-os como um co-design de controlo e computação.

Em [19] são listadas 3 sugestões para o desenvolvimento de sistemas ciber-físicos. Recomenda-se que os sistemas tenham abstração e arquiteturas que permitam um design modular. A confiança

e a segurança são características fulcrais dos CPS. Por fim, os componentes utilizados deverão ser altamente fidedignos, seguros e reconfiguráveis, características estas que se devem estender também ao sistema na sua globalidade.

2.2.4 Microcontroladores

Microcontroladores são computadores, num único módulo integrado, que contêm um processador, memória e periféricos. A memória pode ser de vários tipos, RAM, Flash, EEPROM, entre outras. Os periféricos são programáveis e permitem a entrada e saída de dados, como comunicação e ligação de pinos a dispositivos.

Para o desenvolvimento do dispositivo protótipo, serão analisados 3 microcontroladores dos mais utilizados em projetos do género, com diferentes graus de complexidade, cada um com as suas características e todos diferentes entre si. São eles o Arduino2.2.4, a STM322.2.4 e a Raspberry Pi2.2.4.

Arduino

O Arduino trata-se de uma plataforma muito disseminada no mundo da prototipagem eletrónica, devido à sua facilidade de utilização, pinos adequados para projetos desse género, baixo custo, entre outras características. Possui um ambiente de programação amigável, mascarando a complexidade do C/C++, que faz do Arduino uma excelente porta para o mundo do desenvolvimento de protótipos, nomeadamente dispositivos IoT. Além disso, pode também correr linguagem C/C++.

Existem várias placas de Arduino, mas em termos físicos as valências são praticamente as mesmas, variando apenas as quantidades disponíveis. São, então, compostas pelo controlador, da Atmel, pinos programáveis de entrada e saída, tanto digitais como analógicos, e interface série ou USB. É comum ver-se em projetos a utilização de 'shields', que são placas de extensão que atribuem ao Arduino outras capacidades, nomeadamente de comunicação em rede.



Figura 2.1: Arduino Uno.

STM32

A família STM32 é um grupo de microcontroladores de 32 bits. Desenvolvidas pela STMicroelectronics, as placas variam em capacidades e em tamanhos. São, também, muito utilizados no mundo da prototipagem eletrônica, sendo já para um nível mais avançado. Um ponto positivo comparativamente como a plataforma Arduino é a capacidade de *debugging*. Os processadores utilizados são da empresa ARM, e são rodeados de RAM, memória de flash e vários periféricos, nomeadamente pinos de entrada e saída, tanto digitais como analógicos, comunicação de série, entre outros, também variando de placa para placa.

O desenvolvimento de software para esta família é feito em C/C++, podendo ser utilizado o sistema operativo FreeRTOS. O FreeRTOS é um sistema operativo criado com o objetivo de ser pequeno e simples. Com o código escrito em C, oferece funções de manipulação para tarefas, meios de comunicação entre tarefas, timers, etc., tudo o que se espera encontrar num sistema operativo de tempo real. Como vantagem, oferece também capacidades de baixo consumo energético, o que num projeto do género é uma vantagem interessante.

Através da utilização da camada de abstração de hardware fornecida pela marca, é garantida portabilidade entre placas. Também a marca oferece o ST32CubeMX, IDE que assiste no desenvolvimento de código, nomeadamente na inicialização de pinos e outras configurações. Maioritariamente, as placas permitem utilizar vários métodos de repouso, que garantem um baixo consumo energético.

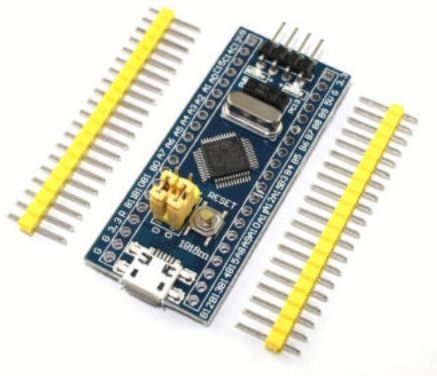


Figura 2.2: STM32F103C8T6.

Raspberry Pi

A Raspberry Pi tem como características fortes a sua portabilidade, graças às pequenas dimensões, e o seu preço reduzido, para as capacidades que tem.

Trata-se de uma unidade central, por exemplo, ARM de 64 bits, na última geração, a Raspberry Pi 4, rodeada de uma RAM, pinos de entrada e saída programáveis, ligação Ethernet, ligação Wi-Fi e portas USB, entre outras características. As portas USB permitem a ligação de dispositivos de

entrada como o teclado e o rato. Pode correr vários sistemas operativos, como o Raspberry Pi OS, Ubuntu, Windows 10 IoT Core e RISC OS.



Figura 2.3: Raspberry Pi 4.

Conclusão

A análise individual dos 3 microcontroladores permitiu obter várias características, que servirão de referência para a comparação entre os microcontroladores, no momento da escolha de qual a utilizar no projeto.

	Arduino	STM32	Raspberry Pi
Principal aplicação	Códigos simples, encadeados, com poucas tarefas.	Com FreeRTOS, ordenar facilmente a execução de tarefas independentes.	Executar programa que beneficie da robustez e complexidade do sistema operativo de propósito geral.
Consumo energético	>45mA	>110mA	>500mA
Porta série	Sim	Sim	Sim
Wi-Fi	Requer módulo	Requer módulo	Sim
Bluetooth	Requer módulo	Requer módulo	Sim

Tabela 2: Análise comparativa de 3 microcontroladores.

2.2.5 Fonte de energia

Num contexto de sistemas computacionais a funcionar num ambiente fora do controlo humano, deseja-se que o consumo energético seja o mais baixo possível. Ao mesmo tempo, é desejável que haja maneiras de carregar as baterias sem ser necessária intervenção humana. Para tal, serão analisadas ao longo dos próximos parágrafos as energias renováveis.

Energias renováveis são aquelas que têm origem nos recursos naturais da terra, como sol, vento, água e calor. As mais utilizadas são a energia solar, energia eólica, energia hidráulica e energia geotérmica.

A energia solar, que tem como fonte o sol, e é captada através de painéis fotovoltaicos, sendo a energia é armazenada em baterias, ou utilizada diretamente. Os painéis são compostos por células fotovoltaicas, que criam uma diferença de potencial por ação da luz, beneficiando do efeito fotovoltaico para absorver a energia do sol, fazendo fluir a corrente elétrica entre duas camadas com cargas opostas. Em Portugal, um painel de 1 metro quadrado gera, em média, entre 120 e 150 Watt.

Também viável seria a utilização de energia eólica. Esta origina do vento, e, através de ventoinhas em movimento, recolhe a energia cinética do mesmo, transformando-a em energia elétrica. Processo em tudo similar à energia hidráulica, cuja fonte de energia cinética é o movimento do corpo de água, nomeadamente rio ou mar.

A conversão da energia cinética em energia elétrica é feita através do gerador que é rodado pelas pás da ventoinha. Este gerador é composto por um ou mais ímanes, que, ao rodarem, produzem uma variação magnética que induz corrente elétrica numa bobina, que é direcionada para a bateria ou diretamente para uso.

Olhando ao contexto onde o dispositivo estará inserido, em Portugal, as energias solar e eólica são intermitentes, a hidráulica requer movimento constante das águas, e apenas a geotérmica é constante mas de difícil captação.

2.2.6 Comunicação sem fios

Hoje em dia, com a evolução das tecnologias IoT, previamente contextualizadas, os meios de comunicação sem fios são cada vez mais. A título de exemplo, entre muitos outros mais, existem as tecnologias ZigBee, 6LowPan, Bluetooth, Wi-Fi, LTE (Long Term Evolution), 2G(GSM), 3G, 4G, e começa agora a surgir o 5G [20].

O 6LowPAN combina o último protocolo de Internet (IPv6) e as redes pessoais sem-fios de baixo consumo energético (LoWPAN). Permite que pequenos dispositivos com capacidade de processamento transmitam informação de um modo sem fios através de um protocolo de Internet. Trata-se de um grande adversário do ZigBee. O ZigBee, por sua vez, foi desenvolvido utilizando o padrão 802.15.2, mas, ao contrário do 6LowPAN, não tem facilidade em comunicar com outros protocolos. Um dos principais benefícios do ZigBee é o facto de poder estar praticamente sempre em modo de baixo consumo energético, diminuindo bastante os gastos energéticos do sistema. É tipicamente implementado numa rede sem fios, com vários nós, que combinados entre si podem atingir maiores distâncias[21].

O Bluetooth clássico é uma rede sem fios de curto alcance, que funciona na frequência de 2.4 GHz. É utilizado para a transferência de ficheiros entre dispositivos, ligação a dispositivos sem fios como colunas, auscultadores, teclados, entre outros, e é, devido à comodidade de, por exemplo, não ter de ligar auscultadores aos dispositivos, uma tecnologia em crescimento. Existe uma versão de baixo gasto energético, o BLE - *Bluetooth Low Energy* - cuja característica adicional é a conservação energética em dispositivos que transmitem menos informação. Isto é conseguido através de um modo de repouso que é mantido sempre que não está a ocorrer comunicação.

Wi-fi é uma família de protocolos de rede sem fios, geralmente utilizada para acesso à Internet. Nessa utilização, veio substituir o cabo de rede como meio de acesso, devido à sua comodidade. É estimado que são produzidos anualmente mais de 3 mil milhões de dispositivos com capacidades Wi-Fi. Pode-se também, através da Internet, proceder à comunicação entre dispositivos, e é desse modo que se insere nas tecnologias IoT. Hoje em dia, dispositivos como frigoríficos têm já ligação à Internet, o que é bastante representativo da evolução que ocorreu. Entre outros, os mais comuns dispositivos que dispõem desta tecnologia são os computadores, fixos e portáteis, os telemóveis e *tablets*, as *smart TVs* e as impressoras sem fios.

O LTE, 2G(GSM), 3G, 4G e 5G podem-se agrupar, pois são evoluções da mesma tecnologia. Tratam-se de redes móveis maioritariamente desenhadas para comunicação através de telemóveis, são também utilizados para outros dispositivos, nomeadamente o multibancos, sistemas de alarme, e outros que, por motivos de segurança, necessitam de uma comunicação permanente e global que o Wi-Fi não consegue garantir. E é essa a principal vantagem que trazem. Isso obviamente traz um custo energético e a comunicação ser feita com custos. O 3G tratou-se da primeira rede móvel de alta velocidade, e desde aí tem-se vislumbrado uma evolução no campo, sendo que o 5G promete trazer uma globalidade de comunicações nos próximos anos, estando a decorrer a sua propagação pelo mundo.

	Zigbee	6LowPAN	Bluetooth	Wi-fi	Redes móveis
Utilização típica	Monitorizar e controlar	Monitorizar e controlar	Conectar dispositivos	Ampla	Ampla
Alcance	10-100m	10-100m	15-30m	Varia	Global
Consumo	30mA	Ínfimo	30mA	NA	Alto
Bitrate	250kbps	250kbps	1Mbps	Varia	NA
Custo uso	NA	NA	NA	NA	Existe

Tabela 3: Análise comparativa de meios de transmissão.

Estes métodos de comunicação são divisíveis pelo seu alcance [22]. As redes móveis tratam-se de um meio de comunicação de largo alcance, enquanto que as restantes listadas oferecem alcances limitados. Inversamente ao que acontece ao nível do alcance, as redes móveis são as que consomem mais energia para trabalhar.

Tratam-se todos de escolhas válidas, sendo que a escolha para cada projeto IoT deve ter em conta as especificidades do mesmo, não havendo uma resposta correta que se aplique a todos por igual.

2.3 Sensores para a monitorização de parâmetros da água

Esta secção contém uma listagem de parâmetros da água. Para cada parâmetro é explicado o que este representa, como a medição é feita e os sensores existentes no mercado para esse parâmetro. O objetivo deste levantamento é que depois sirva de referência na escolha dos sensores a aplicar no dispositivo.

As variáveis a estudar são o pH, TDS(sólidos totais dissolvidos), temperatura, turbidez. Também são apresentados sensores que detetem a presença de água.

2.3.1 Sensor de pH

O pH é o potencial Hidrogénio e consiste numa escala que exprime a acidez ou alcalinidade de uma solução.

O seu cálculo é feito através do logarítmico negativo da concentração de iões de Hidrogénio, expressada na unidade de moles por decímetro cúbico.

Uma solução que apresente um pH de 7 é considerada neutra, inferior a este valor, entre 0 e 7 é considerada ácida, e acima, até 14 é considerada alcalina.

Para uma avaliação menos específica, o pH é medido com o auxílio de indicadores que assumem cores mediante os valores de pH, existem também sensores como o exposto abaixo.

Este sensor trata-se de um sensor analógico, que traduz o pH da água numa voltagem correspondente.

As suas gamas de funcionamento ideal e de medição estão listadas abaixo, bem como a precisão e tempo de resposta.

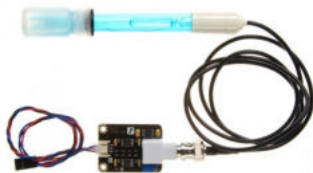


Figura 2.4: Sensor de pH.

- Gama de medição : 0-14 pH;
- Valor energético : 5.00V;
- Tamanho : 43mmx32mm;
- Temperatura de funcionamento: 0-60 °C;
- Precisão : $\pm 0.1\text{pH}$ (25 °C);
- Tempo de resposta : $\leq 1\text{min}$.

2.3.2 Sensor do TDS

A variável TDS indica quantos miligramas de sólidos solúveis estão dissolvidos por litro de água. Em geral, quanto maior o valor da TDS, mais sólidos solúveis estão dissolvidos na água, e portanto menos limpa é a água. Este valor pode ser usado como referência para a pureza da água. Os sólidos podem ser o carbonato, bicarbonato, cloreto, sulfato, fosfato, nitrato, cálcio, magnésio e sódio.

A medição desta variável é feita através da análise da condutividade da água, que está diretamente relacionada com os sólidos que nela estão dissolvidos, pois estes criam condições para que a água consiga conduzir corrente elétrica.

Para esta medição existem um sensor que faz isso mesmo. Trata-se também de um sensor analógico, que retorna em corrente o valor de condutividade que encontra, e que depois, através do cálculo fornecido nos documentos, corresponde ao nível de TDS da água.



Figura 2.5: Sensor do TDS da água.

- Gama de TDS : 0~100ppm;
- Precisão de TDS: $\pm 10\%$ F.S.;
- Voltagem de entrada: 3.3 5.5V;
- Voltagem de saída: 0 2.3V;
- Tamanho do sensor: 42 * 32mm;
- Tamanho da sonda: 83cm.

2.3.3 Sensor de temperatura

A temperatura é uma variável que representa a energia cinética média de cada grau de liberdade de cada uma das partículas de um sistema em equilíbrio térmico. Os átomos estão em constante vibração, e o nível de temperatura é proporcional a esse nível de vibração. Quanto maior for a agitação dos átomos, maior a temperatura.

A medição da temperatura pode ser feita de alguns modos. Utilizando um termopar, que é um par de fios metálicos, conectados a uma extremidade, criando uma diferença de tensão termoelétrica, que indica a temperatura. Através de um sensor de temperatura de resistência, que se trata de um rolamento de fios, que ao aquecer indicam a variação da temperatura ambiente. Ou através de termistores, que possuem sensibilidade à alteração de temperatura. São semicondutores com resistência elétrica proporcional à temperatura.

O sensor abaixo é à prova de água, com a sonda de temperatura selada e permite medir com precisão temperatura em ambientes húmidos. Possui um sensor DS18B20 com interface digital

1-Wire que devolve a leitura de valores numa *word* de 9 a 12-bits (configurável). Consegue funcionamento ideal a praticamente qualquer temperatura humanamente razoável, e é utilizado muito frequentemente em projetos do género.



Figura 2.6: Sensor de temperatura à prova de água.

- Temperatura de funcionamento: $(-55)-(+125)$ °C;
- Precisão : ± 0.55 °C de -10°C a $+85^{\circ}\text{C}$.

2.3.4 Sensor de turbidez

A turbidez é uma propriedade física da água, que é visível através da redução da transparência devido à presença de materiais em suspensão que não permitem a passagem de luz por ela. O seu aumento é causado pela subida da presença desses materiais, normalmente orgânicos.

A medição é feita através da transmissão de um feixe de luz, para um fototransistor recetor, que avalia a dispersão entre eles, que em águas mais transparentes é mínima, sendo maior em águas com mais turbidez.

O sensor encontrado é analógico, devolvendo uma tensão mediante o valor avaliado, através do método descrito anteriormente.



Figura 2.7: Sensor de turbidez analógico.

- Tempo de resposta: $<500\text{ms}$;
- Temperatura de funcionamento: $(5)-(90)$ °C

2.3.5 Sensor do nível

Este sensor fotoelétrico do nível da água opera usando os princípios óticos. Similarmente ao que acontece no sensor de turbidez, existe uma transmissão e receção de um feixe de luz. A presença de água afeta esta comunicação, sendo então possível depreender que o sensor está a ser atravessado por ela.

As vantagens deste sensor são a sua boa sensibilidade e a ausência da necessidade de calibração. A sonda é resistente à ferrugem e tolera altas temperaturas e alta pressão. Sendo digital, se o sensor entrar em contacto com a água, dará um input positivo.



Figura 2.8: Sensor de nível.

- Temperatura de funcionamento: $(-25)-(+125)^\circ\text{C}$.

2.3.6 Sensor de pluviosidade

Esta deteção é feita com dois módulos, uma placa coletora e uma placa de análise. A deteção é feita na placa coletora e interpretada pela placa de análise. A resistência da placa coletora é afetada pela quantidade da água na sua superfície, pois a água é condutora de energia. Quanto mais húmida estiver a placa, menos resistência irá apresentar, e então aumenta a voltagem transmitida pela placa de análise.

O sensor tem um output de nível baixo se detetar presença de gotas, a partir de um valor definido num potenciômetro.



Figura 2.9: Sensor de pluviosidade.

- Voltagem de entrada: 3.3 5V;
- Saída: nível alto ou nível baixo;
- Tamanho do sensor: 5 * 4 cm.

Capítulo 3

Análise

Ao longo deste capítulo é feita uma reflexão das informações recolhidas no capítulo 2. É feito um estudo do problema, que permite retirar os requisitos que são vistos como necessários para o projeto.

São especificados os casos de uso, que são as interações que os atores têm com o sistema.

De seguida estão os diagramas de sequência, que correspondem cada um a um caso de uso, e representam a sequência de processos que o sistema deve ter após a interação por parte do ator.

3.1 Estudo do problema

Com base na contextualização feita, é possível delinear o problema em mãos e daí tirar as características que a solução terá de oferecer.

Deseja-se que o disposto seja capaz de recolher, periodicamente, parâmetros da qualidade da água, portanto devem-se definir quais serão esses parâmetros. Após essa recolha, o dispositivo deve transmitir, a longa distância, os mesmos dados para uma central de comando, pelo que importa definir o meio de transmissão. Estas duas partes ocorrerão num dispositivo IoT, sendo necessário escolher qual a unidade de processamento e qual a fonte de energia, ambicionando sempre o máximo de eficiência energética, que permita a auto-suficiência energética do dispositivo, sem ser necessário interação humana.

A central de comando que complementa o dispositivo deverá ser capaz de receber os dados, processar os mesmos, guardar os valores recebidos, disponibilizando-os através de uma interface que permita ao utilizador definir níveis de aviso, a partir dos quais deverá ser enviado um alerta.

A fiabilidade de ambos deve ser alta, o dispositivo deve ser compacto e fácil de instalar, bem como a central de comando deve ser simples e intuitiva.

Todos estes aspetos estão resumidos na tabela visível abaixo.

Requisitos
Acordar periodicamente
Fornecer valores precisos das variáveis de qualidade da água
Guardar dados na memória do dispositivo
Transmitir valores a longa distância
Ser energeticamente auto-suficiente
Disponibilizar valores graficamente na central de comando
Definir valores de alarme
Alertar quando valores forem ultrapassados

Tabela 4: Requisitos do projeto.

3.2 Casos de uso

O diagrama de casos de uso identificam as interações que os atores podem ter com o sistema. Neste caso existem 3 atores, o utilizador, a central de comando e o sistema de computação.

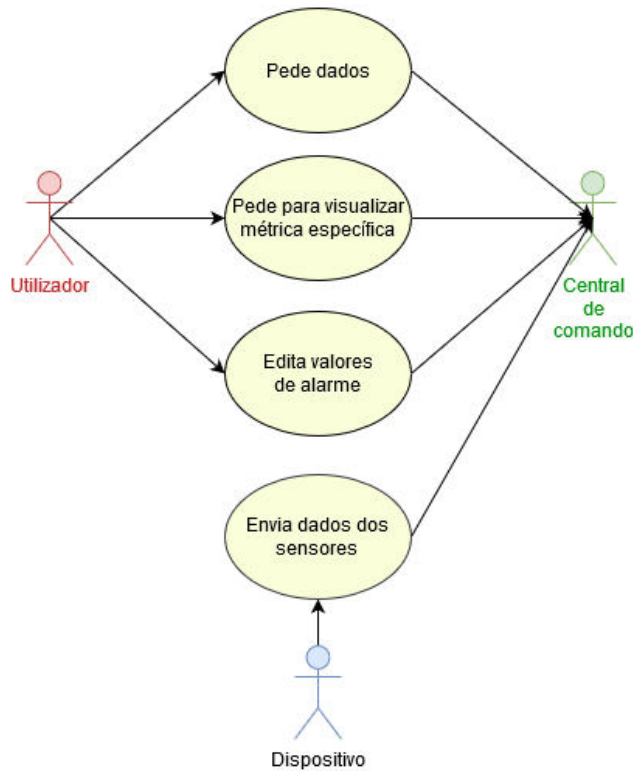


Figura 3.1: Casos de uso.

O utilizador apenas interage diretamente com a central de comandos. Poderá requerer à central de comando que esta faça uma recolha de dados, que exiba uma variável específica, de entre as

várias a ser observadas, e pode editar os valores de alarme, que são os valores afetos a cada métrica, que quando ultrapassados levam ao envio de uma notificação.

O dispositivo, por sua vez, interage com o sistema enviando os dados que ele próprio recolheu, dos valores atualizados lidos pelos sensores de qualidade da água.

3.3 Diagramas de sequência

Um diagrama de sequência representa a sucessão de processos que ocorrem após uma interação por parte de um ator. Serve para especificar como os intervenientes no sistema se influenciam sequencialmente após essa interação inicial.

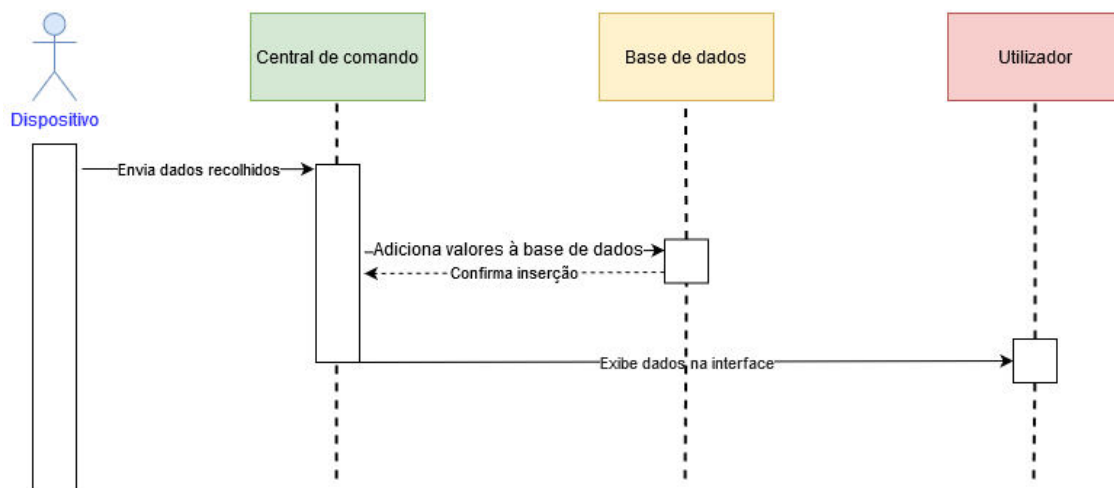


Figura 3.2: Diagrama de sequência.

A interação presente na figura 3.2 tem o dispositivo como ator inicial. O dispositivo, na periodicidade definida, após recolher os dados dos sensores, interage com o sistema enviando estes mesmos dados para a central de comando. A central de comando adiciona os valores recebidos, já convertidos, à base de dados, que devolve à central de comando uma confirmação do sucesso da inserção.

Por fim, a central de comando torna visíveis os dados ao utilizador, através da interface.

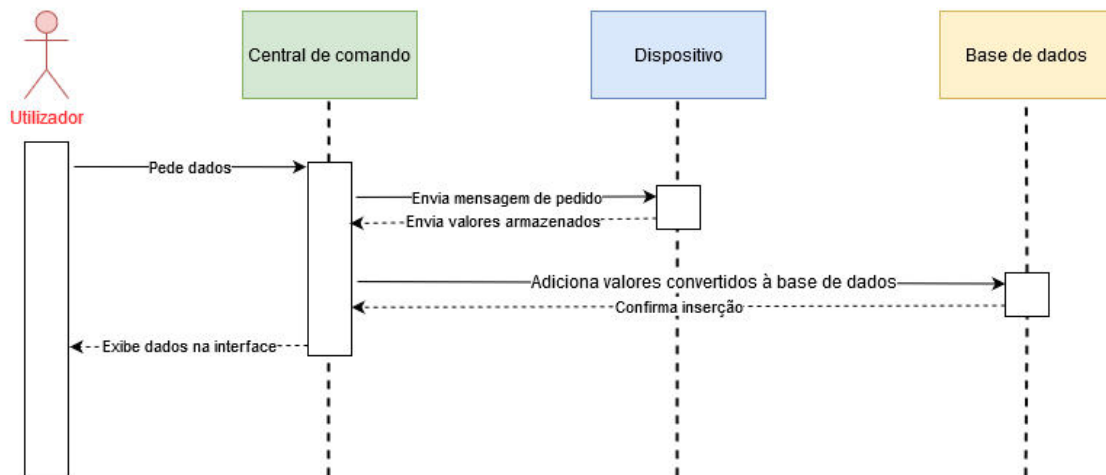


Figura 3.3: Diagrama de sequência.

A sequência visível na imagem 3.3 tem início com o pedido de dados por parte do utilizador. Ao receber este pedido, a central de comando envia uma mensagem de pedido ao dispositivo, que faz com que o dispositivo envie todos os dados que tem armazenados.

Ao receber os valores, a central de comando adiciona-os à base de dados, já convertidos para valores reais. Esta inserção na base de dados é confirmada pela base de dados.

Por fim, a central de comando torna visíveis os dados ao utilizador, através da interface.

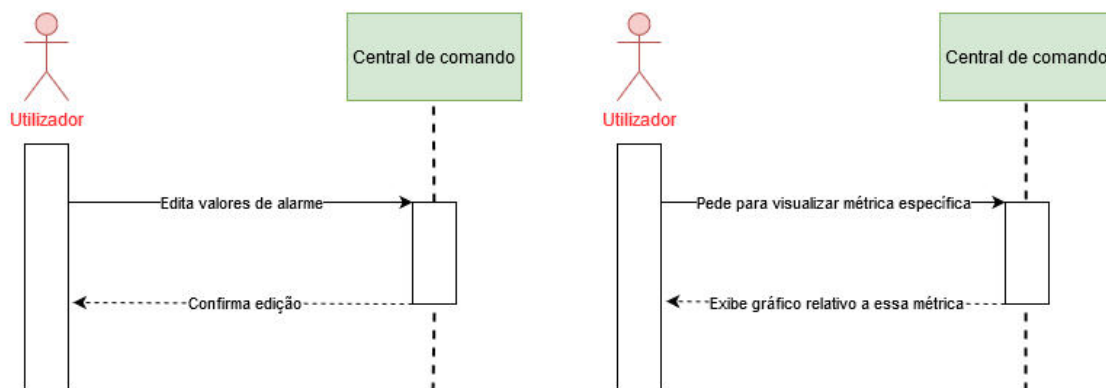


Figura 3.4: Diagrama de sequência.

Na figura 3.4 estão duas interações do utilizador com a central de comando. Na primeira, à esquerda, o utilizador edita os valores de alarme, na central de comando, e a central de comando devolve uma confirmação dessa edição. O envio de mensagem de alarme será agora feita quando os valores lidos ultrapassarem as novas definições.

Na interação à direita, o utilizador requer à central de comando a exibição do gráfico afeto a uma variável específica. Ao receber este pedido, a central de comando exhibe o gráfico relativo a essa métrica.

3.4 Arquitetura do sistema

Como se pode ver no capítulo 2, a solução para uma questão deste género passa por um dispositivo IoT, capaz de recolher e transmitir as métricas desejadas. Em geral, composto por um microcontrolador, equipado por sensores responsáveis pela aquisição dos dados, e por um dispositivo de transmissão de dados.

A transmissão é feita para uma central de comando, normalmente um computador, que será o local onde se vão armazenar os valores recebidos. Esta central será a interface, que permita ao utilizador ver os valores recebidos. A conversão de valores digitais ou analógicos pode ser feita tanto no microprocessador como no recetor.

As decisões ao nível do design do sistema passam, então, pela seleção dos sensores a aplicar ao problema, do sistema de computação, da fonte de energia, do método de comunicação e da central de comando. Um sistema abstrato deste género será algo como o da seguinte imagem:

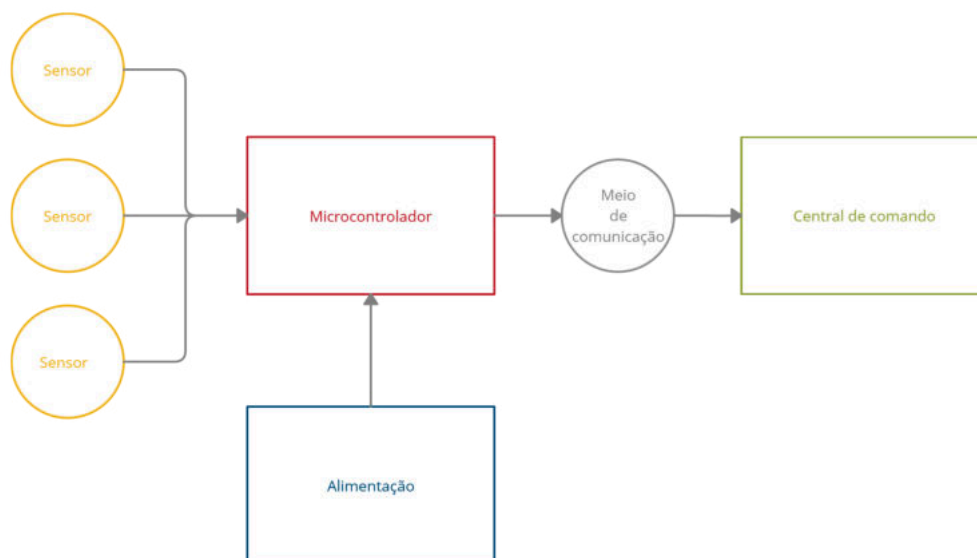


Figura 3.5: Arquitetura abstrata de sistema IoT.

Capítulo 4

Design

Capítulo onde cada um dos blocos especificados anteriormente na arquitetura são definidos. Esta definição é feita com o suporte da pesquisa de estado da arte, que serve de base para a justificação das decisões. Estes blocos são os sensores, o sistema de computação, a fonte de energia, o meio de comunicação e a central de comando.

4.1 Sensores

Em termos de qualidade da água, seguindo a tabela da Organização Mundial de Saúde para a qualidade da água, selecionaram-se os sensores de Turbidez, pH e TDS.

Parâmetros	Gama de valores	Unidade
Turbidez	5-10	NTU
pH	6.5-8.5	pH
TDS	192-512	ppm

Tabela 5: Tabela da Organização Mundial de Saúde.

A temperatura, como se viu no estado da arte, é uma variável muito analisada, e muito importante pois pode influenciar os outros parâmetros, como por exemplo o pH, bem como afetar a vida animal e das plantas. Portanto foi também inserido o sensor que analisa esta variável.

Em termos de hidrodinâmica, para se poder ter noção da altura da água em determinado momento, a proposta passa pela montagem de uma estrutura com mais do que um sensor fotoelétrico do nível da água, dando uma indicação se certas alturas foram atingidas ou não, agindo como pontos de verificação.

Também em termos hidrodinâmicos, a chuva pode afetar a subida do nível da água. Para a detetar, foi escolhido um sensor de pluviosidade, no topo do dispositivo.

Sensor
Sensor de pH
Sensor do TDS
Sensor de temperatura
Sensor de turbidez analógico
Sensor Fotoelétrico Nível de Água
Sensor de pluviosidade

Tabela 6: Lista de sensores.

Planos de testes dos sensores

Com base na documentação disponível para cada sensor, foram definidos alguns testes unitários para se poder testar o normal funcionamento destes. Nos sensores de TDS e de turbidez, devido à ausência de amostras no momento de desenvolvimento do protótipo, apenas serão testadas as suas gamas de voltagem de saída. Os restantes testes são feitos com amostras conhecidas, como o pH de águas vendidas em loja e a temperatura ambiente da sala. No caso dos sensores com saída lógica, a presença e ausência de água e chuva, para os sensores de nível e pluviosidade, respetivamente, determinarão o nível lógico da saída.

Sensor de	Input	Output expectável
pH	pH de 5,7	89mV de output
	pH de 9,5	1147mV de output
TDS	Medir TDS	Output entre 0 e 2.3V
Temperatura	Ambiente ($\simeq 20^{\circ}\text{C}$)	191 hexadecimal
Turbidez	Medir turbidez	Output entre 0 e 4.5V
Nível da água	Presença de água	Nível 1 lógico
	Ausência de água	Nível 0 lógico
Pluviosidade	Presença de chuva	Nível 0 lógico
	Ausência de chuva	Nível 1 lógico

Tabela 7: Planos de testes unitários dos sensores.

4.2 Fonte de energia

O microcontrolador requer energia e esta deve ser fornecida autonomamente e sem interação humana. Das 3 possibilidades, eólica, hidráulica e solar, a mais adequada, ao ambiente onde o dispositivo deve ser inserido, é a energia solar.

Além disto, em termos de hardware trata-se de uma alternativa mais simples, que apenas requer que a placa esteja em exposição solar. A necessidade de peças em movimento, no caso das energias

hidráulica e eólica, poderia apresentar um problema para a auto-suficiência desejada, pois seriam mais fáceis de danificar do que a placa solar.



Figura 4.1: Pannel solar.



Figura 4.2: Módulo de alimentação.

Especificações:

- Voltagem de funcionamento: 6V;
- Corrente de funcionamento: 833mA;
- Superfície de vidro endurecido.

Especificações:

- Voltagem de entrada do painel solar: 6-24V;
- Voltagem de saída por USB: 5V;
- Corrente de saída por USB: 1A;

A voltagem e corrente oferecidas por esta opção são mais do que suficientes para alimentar o dispositivo, sendo então uma opção ideal. Carecendo de informação sobre durabilidade sem sol.

A ligação entre o painel e o módulo de alimentação é feita via *jack* DC-002. O módulo alimenta a bateria recarregável de 3.7V. A energia acumulada na bateria, irá alimentar o microcontrolador, através de cabo micro USB.

Plano de testes da fonte de energia.

Input	Output expectável
Presença de luz solar	Output de energia 6V
Input de energia 6V	Carregamento da bateria
Conexão por cabo USB	Output de saída 5V/1A

Tabela 8: Plano de testes da fonte de energia.

Na tabela estão os testes a executar no módulo de fonte de energia do dispositivo, que passam por verificar a conversão, por parte do painel, de luz solar em eletricidade, verificar também se esta eletricidade está a carregar a bateria com sucesso, e por fim se esta bateria, ligada ao dispositivo, lhe fornece a energia que ele necessita.

4.3 Comunicação

A escolha do método de comunicação recaiu para o GSM, pois métodos como o Bluetooth e o Wi-Fi estão dependentes da proximidade. Tratando-se de um dispositivo que se encontrará numa localização remota, apenas o GSM se apresenta como resposta ao problema. O problema que a tecnologia oferece é o seu custo de uso, mas é uma troca necessária em termos de custo-benefício.

Mais especificamente, o módulo selecionado é o SIM808. É um módulo dois em um, com GSM e GPS, suporta redes GSM/GPRS de quatro bandas e combina com tecnologia GPS para navegação satélite.

Oferece consumos energéticos muito baixos em modo de poupança energética e é integrado com um circuito de carregamento, aguentando tempos de *standby* muito longos. Além disto, é controlado por comandos AT através da UART e funciona tanto em 3.3V como em 5V.

Especificações:



- Suporta relógio em tempo real
- Consumo de 1 mA em modo de poupança energética
- Cartão SIM standard

Figura 4.3: Módulo SIM808

Plano de testes do módulo de comunicação

Input	Output expectável
Envio de comandos AT	Sucesso no controlo do módulo
Envio de "Hello World"	Sucesso no envio

Tabela 9: Plano de testes do módulo de comunicação.

Na tabela estão presentes os testes unitários a executar ao módulo de comunicação passam por testar a sua capacidade de processar os comandos AT, que servem para controlar o módulo, e por testar também o envio de uma mensagem para o recetor.

4.4 Sistema de computação

O sistema de computação a escolher teria de possuir certas características, nomeadamente o armazenamento dos dados, um bom número de pinos de ligação compatíveis com os sensores, compatível com comunicação série, e, não menos importante, baixo consumo energético, de forma a maximizar a vida da sua unidade de alimentação.

Analisando os 3 microcontroladores estudados no estado da arte, o Arduino, a STM32 e a Raspberry Pi, considerou-se que complexidade e sofisticação da Raspberry Pi não é necessária para o projeto e até indesejada. Comparativamente ao Arduino, os microcontroladores do grupo STM32 oferecem a possibilidade de utilização de FreeRTOS, o que permite a utilização de tarefas. A gestão de tarefas que é possível com o FreeRTOS é muito útil para o problema, que oferece também características de baixa latência, então a escolha recaiu sobre os microcontroladores do grupo STM32.

O FreeRTOS da SMT32 permite também uma futura expansão de código simples, adicionando tarefas mais facilmente do que na programação em C do Arduino. Em termos de hardware, a STM32 possui mais recursos periféricos e capacidade de processamento e de memória superiores, o que também faz com que a STM32 esteja mais apta a futuras expansões do que o Arduino.

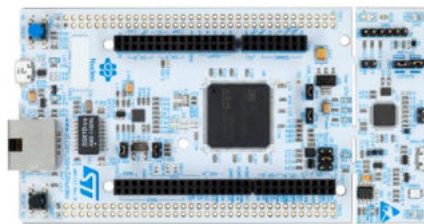


Figura 4.4: STM32 Nucleo-144 development board.

Mais especificamente, a placa utilizada é a STM32F767ZI, mas, e é uma das valências da família STM32, o código poderá ser implementado nas restantes placas dessa mesma família, sendo totalmente compatível.

A placa será responsável pela obtenção dos valores dos sensores e o seu envio para o recetor, bem como outras funções extra.

O tratamento dos dados recolhidos ficará a cargo da central de comando, convertendo os valores de tensão em valores reais das variáveis, para poupar energia no dispositivo.

Arquitetura de software

Na figura abaixo está a arquitetura do software implementado no microcontrolador. Como já foi definido, o dispositivo é responsável por controlar os sensores através da sua aquisição e enviar os valores através da porta série. Estas funções fazem parte do middleware.

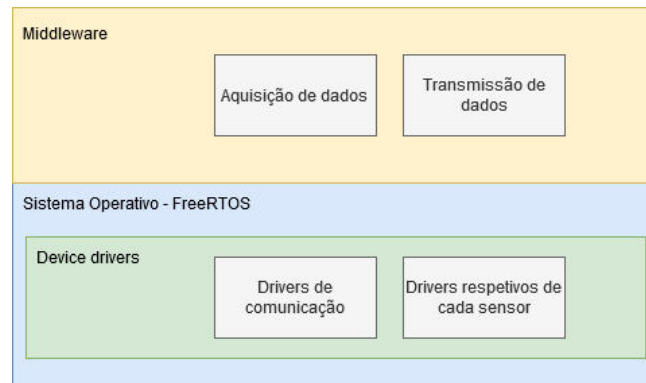


Figura 4.5: Arquitetura de software do sistema de computação.

Havendo uma grande vantagem no FreeRTOS, este foi escolhido e complementado com o software de baixo nível, que são as funções necessárias para a leitura de cada sensor e para o módulo de comunicação - os 'device drivers'.

Fluxograma



Figura 4.6: Fluxograma do sistema de computação.

O microcontrolador encontrar-se-á maioritariamente em modo de baixo consumo energético. Ao acordar por interrupção do timer, passará à leitura dos sensores. No caso de acordar devido a um pedido da central de comando, deverá proceder ao envio dos dados acumulados na estrutura de dados.

4.5 Central de comando

Como ficou definido no sistema genérico, terá de haver uma central de comando para obter e converter os dados dos sensores em valores reais das métricas respectivas. Servirá também para armazenar essas métricas em base de dados e servirá de interface ao utilizador. Esta central será desenvolvida em C#, devido à facilidade que oferece no desenvolvimento, bem como, com a sua biblioteca de pacotes, executar várias funcionalidades como por exemplo o envio de e-mails de alarme quando os valores máximos forem ultrapassados.

A interface desejada deverá mostrar a evolução das variáveis ao longo do tempo em modo gráfico, como na figura abaixo.



Figura 4.7: Interface gráfica.

Além dos gráficos relativos às variáveis, a interface deve permitir ao utilizador executar um pedido de valores ao dispositivo, proceder à exportação dos dados tanto em tabela, como o envio dessa tabela por e-mail, além de ajustar os valores a partir dos quais a central de comando deve accionar os mecanismos de alarme.

Fluxograma

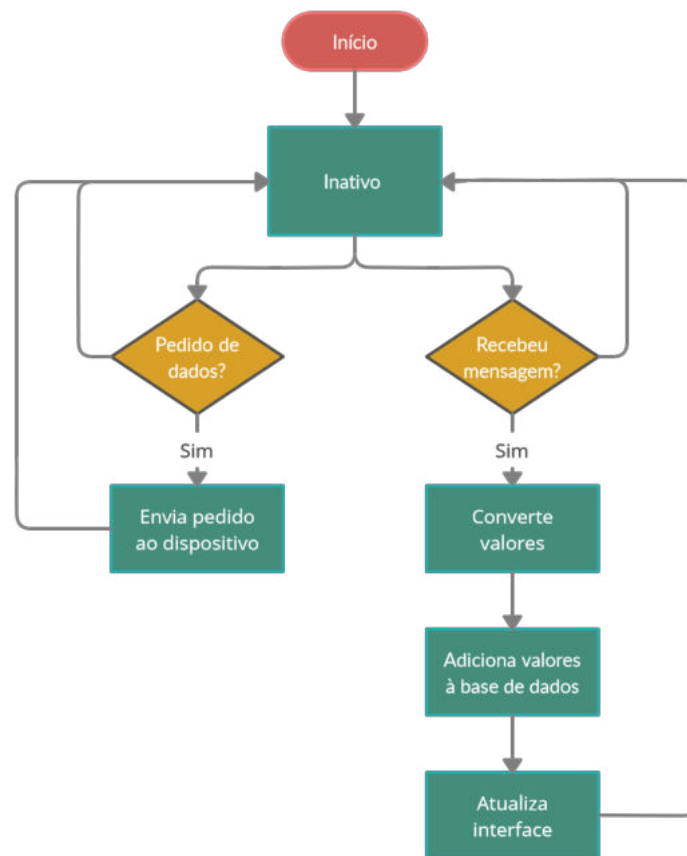


Figura 4.8: Fluxograma da central de comando.

A central de comando, à semelhança do que acontece com o dispositivo, também estará majoritariamente inativa, correndo apenas em dois casos. Primeiro, se receber dados irá tratar dos mesmos, convertendo, adicionando à base de dados e atualizando a interface. Segundo, se receber um pedido de dados por parte do utilizador. Nesse caso irá reencaminhar o mesmo para o dispositivo, com uma mensagem pré-definida através da porta série.

4.6 Arquitetura do sistema

Tendo definido os 5 elementos da arquitetura abstrata de um sistema IoT proposta pode-se então definir a arquitetura de hardware.

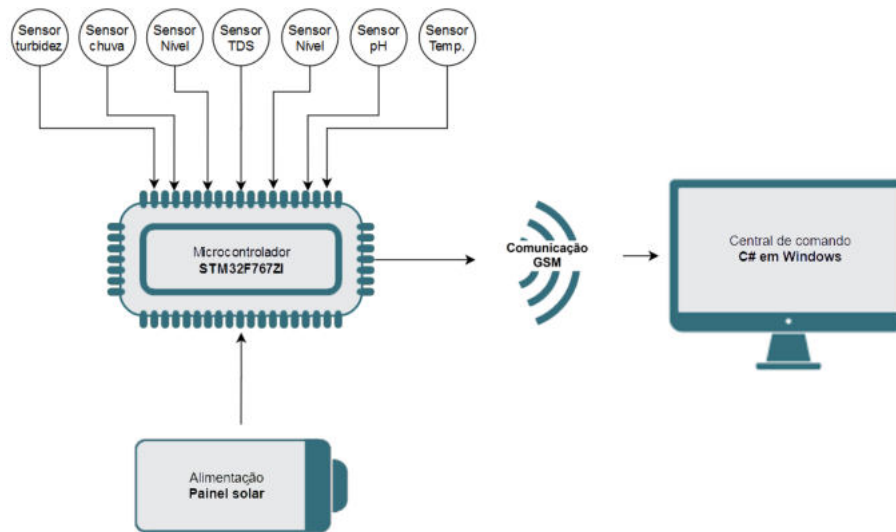


Figura 4.9: Arquitetura final do sistema.

Será composto por 7 sensores, sendo eles de temperatura, de pH, de turbidez, de TDS, dois sensores de nível e um sensor de pluviosidade, conectados a um microcontrolador STM32. Este microcontrolador será alimentado com energia solar, recolhida por um painel solar.

A comunicação será assegurada através de tecnologia GSM, conectada a uma central de comando desenvolvida em C# a correr num computador Windows.

Capítulo 5

Implementação e Testes

Neste capítulo são expostos os processos de implementação, em laboratório, do sistema e os testes inerentes a isso. É seguida a ordem que foi escolhida no capítulo anterior. Primeiro foram implementados e testados os sensores individualmente, seguidos pela fonte de energia e o meio de comunicação. Tudo isto, juntamente com o sistema de computação, perfaz o dispositivo desejado para o problema.

Foi também implementada a central de comando, e no final foram executados testes de integração, de forma a ser possível avaliar as capacidades e funcionalidades do sistema.

5.1 Sensores



(a) Sensor de pH.



(b) Sensor de TDS.



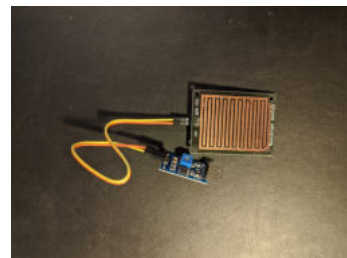
(c) Sensor de temperatura.



(d) Sensor de turbidez.



(e) Sensor de nível da água.



(f) Sensor de pluviosidade.

Figura 5.1: Sensores utilizados na implementação.

Para a implementação dos sensores, foram utilizados os definidos no capítulo 4. São eles, o sensor de pH, o sensor de TDS, o sensor de temperatura, o sensor de turbidez, o sensor de nível da água e o sensor de pluviosidade.

Para todos os sensores foram realizados testes unitários, utilizando o plano de testes definido. Os testes foram considerados de sucesso, pois os resultados corresponderam ao esperado, fazendo com que os sensores se confirmem como adequados para o projeto.

Sensor de	Input	Output expectável	Output
pH	pH de 5,7	89mV de output	≈ 90mV
	ph de 9,5	1147mV de output	≈-1150mV
TDS	Medir TDS	Output entre 0 e 2.3V	250mV
Temperatura	Temperatura ambiente	191h	190h
Turbidez	Medir turbidez	Output entre 0 e 4.5V	4.21V
Nível da água	Presença de água	Nível 1 lógico	Nível 1 lógico
	Ausência de água	Nível 0 lógico	Nível 0 lógico
Pluviosidade	Presença de chuva	Nível 0 lógico	Nível 0 lógico
	Ausência de chuva	Nível 1 lógico	Nível 1 lógico

Tabela 10: Resultados dos testes unitários aos sensores.

5.2 Fonte de energia

Foi definido que a energia a ser utilizada seria a solar. Então foi utilizado um painel solar conectado a um módulo de alimentação, que armazena a energia convertida pelo painel.

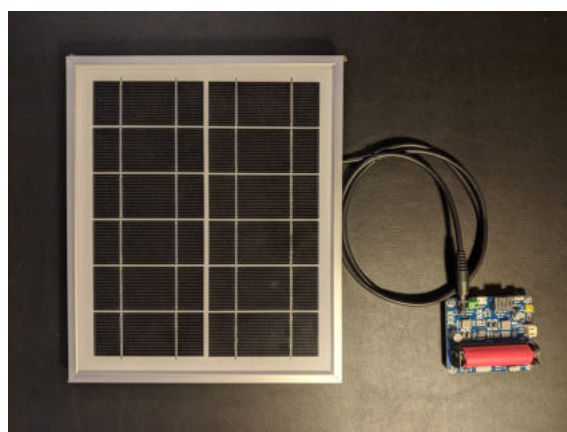


Figura 5.2: Painel solar e módulo de alimentação.

Foram executados os testes unitários aos dois componentes, verificando-se que o painel consegue, com sucesso, converter a energia solar em energia elétrica, e também que o módulo de alimentação armazena a energia na bateria. O resultado da integração destes dois componentes permitiu

alimentar o dispositivo durante vários dias, desde que com exposição solar diária, correspondendo ao desejado.

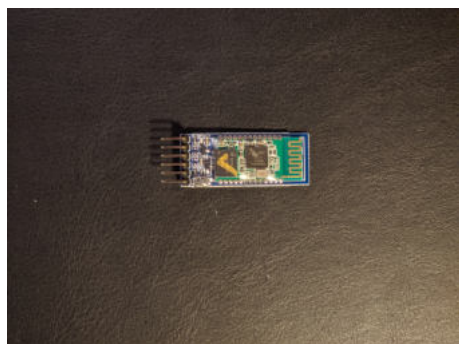
Input	Output expectável	Output
Presença de luz solar	Output de energia 6V	Output de energia 6V
Output de energia 6V	Carregamento da bateria	Carregamento da bateria
Conexão por cabo USB	Output de saída 5V	Output de saída 5V

Tabela 11: Resultado dos testes da fonte de energia.

5.3 Comunicação

Tendo a implementação do protótipo sido feita em ambiente de laboratório, foi utilizado um módulo de comunicação por Bluetooth no lugar do módulo GSM selecionado no capítulo 4. Esta escolha justifica-se pela ausência da necessidade de comunicação de longa distância no ambiente de prototipagem, pelo facto de que a comunicação é também ela feita pela porta série, fazendo com que a passagem para a utilização de GSM seja fácil, e ainda a redução dos custos inerentes à comunicação GSM.

Mais especificamente, foi utilizado o módulo HC-05. A sua utilização e implementação é simples, após a configuração o dispositivo conecta-se de um modo rápido e automático.



Especificações:

- Alcance: 10 metros;
- Consumo baixo de energia.

Figura 5.3: Módulo Bluetooth.

Os testes conduzidos ao módulo foram os mesmos que foram desenhados para a utilização de GSM, pois ambos são controláveis por comandos AT e ambos comunicam pela porta série. Tratam-se de testes de integração, pois resultam já da ligação física entre o módulo e o microcontrolador.

Input	Output expectável	Output
Envio de comandos AT	Sucesso no controlo	Sucesso no controlo
Envio de "Hello World"	Sucesso no envio	Sucesso no envio

Tabela 12: Resultado dos testes ao módulo de comunicação.

Na imagem é visível a comunicação entre os dispositivos, que resulta do teste de envio da mensagem "Hello World", com sucesso, entre o microcontrolador e a máquina onde irá correr a central de comando.



Figura 5.4: Resultado do teste de envio de mensagem.

5.4 Sistema de computação

No sistema de computação é feita a integração com os módulos dos sensores, comunicação e fonte de energia, e é desenvolvido todo o código responsável pela recolha de dados e processamento dos mesmos. Também no código é imposto o modo de poupança energética.

Ligação aos módulos

A tabela contém os pinos onde os módulos dos sensores, comunicação e energia serão ligados à placa para teste. Isso irá influenciar os portos que serão ativados em código - e vice-versa, pois mediante as condições energéticas, poder-se-á condicionar as escolhas.

No total dos 7 sensores, 3 são digitais e devolvem um valor binário, sendo que os restantes devolvem um valor variável, sendo então necessário o uso de pinos ADC.

A comunicação é feita através da UART, então o módulo terá de estar conectado aos respetivos pinos. Por fim, a alimentação é feita por USB, utilizando a entrada micro USB disponível na placa.

A seguinte tabela servirá como guia no passo da definição de pinos, na criação do projeto no programa STM32CubeMX.

Componente	Função	Pino	Componente	Função	Pino
Sensor Turbidez	VCC	5V	Sensor pH	VCC	5V
Sensor Turbidez	GND	GND	Sensor pH	GND	GND
Sensor Turbidez	ADC	PC_3	Sensor pH	ADC	PB_1
Sensor Chuva	VCC	3.3V	Sensor TDS	VCC	3.3V
Sensor Chuva	GND	GND	Sensor TDS	GND	GND
Sensor Chuva	GPIO	PA_4	Sensor TDS	ADC	PC_3
Sensor nível 1	VCC	5V	Sensor temperatura	VCC	5V
Sensor nível 1	GND	GND	Sensor temperatura	GND	GND
Sensor nível 1	GPIO	PA_6	Sensor temperatura	GPIO	PA_5
Sensor nível 2	VCC	5V	Comunicação	VCC	3.3V
Sensor nível 2	GND	GND	Comunicação	GND	GND
Sensor nível 2	GPIO	PC_11	Comunicação	UART RX	PB_10
Alimentação	-	Micro USB	Comunicação	UART TX	PB_11

Tabela 13: Pinos de ligação entre os componentes e o microcontrolador.

Desenvolvimento de código

No desenvolvimento do projeto de código foi utilizada a ferramenta STM32CubeMX. Esta é uma ferramenta gráfica fornecida pela STMicroelectronics que facilita a configuração do projeto. É feita a escolha do microprocessador, configuração dos periféricos, middleware, e por fim é gerado um projeto em C a executar em Keil uVision, que é um IDE de desenvolvimento de código, que oferece capacidades de gestão de projeto e debugging.

Tasks				
Task Name	Priority	Stack Size...	Entry Function	Allocation
defaultTask	osPriorityNormal	128	StartDefaultTask	Dynamic
ReadTurb	osPriorityHigh	128	read_turb	Dynamic
ReadTDS	osPriorityHigh	128	read_tds	Dynamic
ReadTemperature	osPriorityHigh	128	read_temperature	Dynamic
ReadPH	osPriorityHigh	128	read_ph	Dynamic
DataTransmissio	osPriorityAboveN...	128	data_processing	Dynamic
DataProcessing	osPriorityAboveN...	128	data_processing	Dynamic
ReadLevel1	osPriorityHigh	128	read_level1	Dynamic
ReadLevel2	osPriorityHigh	128	read_level2	Dynamic
ReadRain	osPriorityHigh	128	read_rain	Dynamic

Figura 5.5: Tasks criadas.

O primeiro passo aquando da criação de um projeto no STM32CubeMX é a escolha do microcontrolador, da família STM32. De seguida, procede-se à definição dos pinos, mediante as suas ligações físicas e funções que devem ter.

Também dentro do STM32CubeMX, tendo sido escolhido o sistema operativo FreeRTOS, é possível definir as tasks. Além dos seus nomes identificativos, é importante a definição do seu grau de prioridade, também obedecendo ao definido pelo diagrama de prioridades de tasks. Tendo tudo isto em conta, foram criadas as diversas tasks.

Existe uma tarefa para a leitura de cada sensor, bem como uma tarefa de processamento, que recebe os dados das tarefas de leitura e armazena os mesmos numa estrutura de dados, e depois uma tarefa de transmissão, que através da porta série envia os dados recolhidos.

Em termos de poupança energética, para tornar o programa cíclico, não sendo necessário o mesmo estar sempre a correr, foi configurado um timer que consome pouca energia, LPTIM1, responsável pelo acordar do dispositivo em intervalos de tempo definidos. Também o consumo energético foi definido no STM32CubeMX, seleccionando a opção de baixo consumo que este oferece. Esta opção gera duas funções, PreSleepProcessing e PostSleepProcessing, que são as funções chamadas tanto à entrada como à saída do modo de baixo consumo.

A geração do projeto STM32CubeMX, vai dar origem a um projeto de Keil uVision, com as várias sub-diretorias e ficheiros, .c de código, e .h de header.

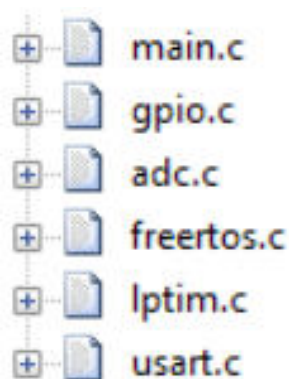


Figura 5.6: Árvore de ficheiros.

É nesse projeto que o código afeto à dissertação é escrito, atribuindo ao dispositivo as capacidades que se desejam e que foram descritas no capítulo 4.

5.5 Estrutura física

A estrutura final ficou como é visível na imagem abaixo. Por não ser um dos focos do projeto, não garante segurança ao nível da resistência à água, que requereria conhecimentos extra. Todavia, serve como referência para uma possível execução por parte de alguém com esses mesmos conhecimentos.

A estrutura foi desenhada no programa SketchUp, de desenho 3D, sendo o modelo gerado o visível na próxima imagem.

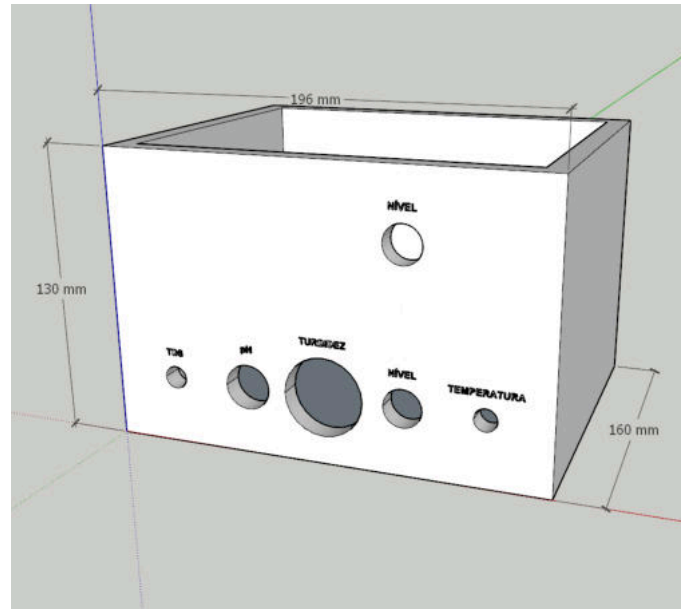


Figura 5.7: Projeto de modelo 3D.

Após a impressão do modelo 3D, a estrutura correspondeu ao esperado, sendo que o passo seguinte foi a colocação de todos os elementos do hardware. Os sensores de qualidade da água estão ao nível do sensor de nível colocado na posição mais abaixo. Isto permite que se tenha certeza que a água está a tocar nos sensores, pois o sensor de nível estará ativo quando isto se verifica.

O sensor de pluviosidade está por cima, por questões gravíticas óbvias. O sensor de nível na posição 2 situa-se mais acima, e servirá como aviso da subida do nível da água.

5.6 Central de comando

Sendo o dispositivo local um desktop standard a correr Windows, não contém implementação de hardware. O software desenvolvido terá como objetivo receber dados, tratar e guardar os mesmos, e por fim disponibilizar através de uma interface.

O desenvolvimento, conforme decidido, foi feito em C#. A interface final é a disponibilizada na imagem seguinte.

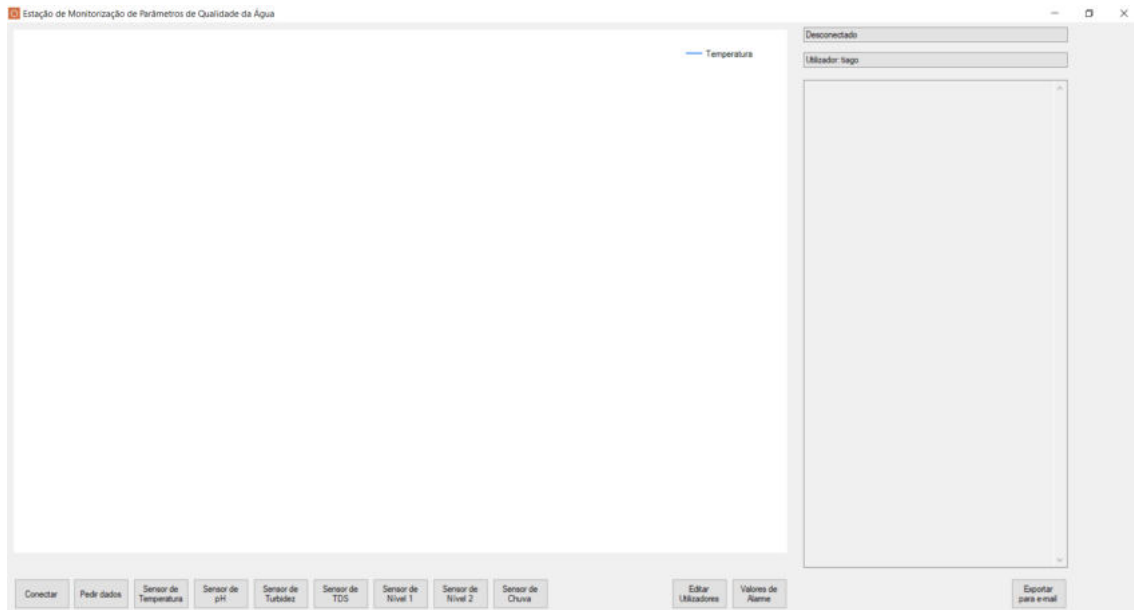


Figura 5.8: Interface.

Numa breve explicação, e do ponto de vista do utilizador, existem as seguintes interações:

- O botão imediatamente à esquerda abaixo do gráfico, permite ao utilizador tentar conectar com o dispositivo, para o caso dessa ligação não ter sido conseguida;
- O botão seguinte serve para o utilizador requerer dados ao dispositivo que esteja conectado com o computador;
- O conjunto de botões relativos às variáveis permitem trocar de gráfico, vendo o gráfico afeto a cada um dos sensores enumerados;
- À direita abaixo do gráfico, o primeiro botão é exclusivo para o administrador e serve para abrir a janela de edição da lista de utilizadores;
- Imediatamente ao lado está o botão de edição dos valores que farão com que o sistema envie um e-mail de aviso;
- O painel à direita serve para um utilizador mais experiente conseguir detetar anomalias nos valores enviados pelo dispositivo. Podem transmitir, por exemplo, problemas com os sensores;
- Acima do painel existem duas caixas de texto informativas, uma informa se o dispositivo está conectado e a outra diz qual o utilizador que está com sessão iniciada neste momento;
- Abaixo do painel, o botão "exportar para e-mail", como o próprio nome indica, permite exportar o ficheiro, que é atualizado a cada receção de dados, para um e-mail à escolha do utilizador.

Para aceder à interface, o utilizador terá de fazer o login. O programa traz um administrador predefinido que servirá para um primeiro início de sessão.



Figura 5.9: Menu de acesso.

Depois do login efetuado, o administrador tem acesso a um menu de manipulação da lista de utilizadores. Poderá criar um utilizador próprio, atribuindo a si capacidades de administrador, que passam pela exclusividade de acesso à lista.

Também poderá adicionar utilizadores, cujo acesso permitirá visualizar os dados, bem como as funções listadas previamente.

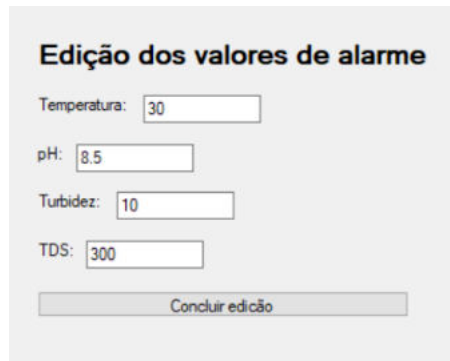


Figura 5.10: Menu de edição de utilizadores.

Valores de alarme

Conforme foi proposto nos objetivos do projeto, o sistema deve possuir mecanismos de alarme, para notificar caso certos valores sejam ultrapassados.

Para tal, é necessário que estes valores sejam definidos, existindo então um menu para essa definição.



Edição dos valores de alarme

Temperatura:

pH:

Turbidez:

TDS:

Figura 5.11: Menu de edição dos valores de alarme.

O menu é exclusivo ao administrador, que definirá os valores máximos a partir dos quais deve ser notificado via e-mail.

Envio de e-mails

O sistema procede ao envio de um e-mail em duas situações. A primeira pode ser vista na figura 5.12, que corresponde ao e-mail enviado quando o utilizador deseja exportar o documento que contém os dados guardados.

A segunda situação é a presente na figura 5.13. Mediante os valores definidos pelo utilizador, quando esses valores são atingidos, este e-mail é automaticamente enviado. Serve de alerta para ser possível agir perante esta alteração de valores. O e-mail informa também qual a variável que foi ultrapassada.

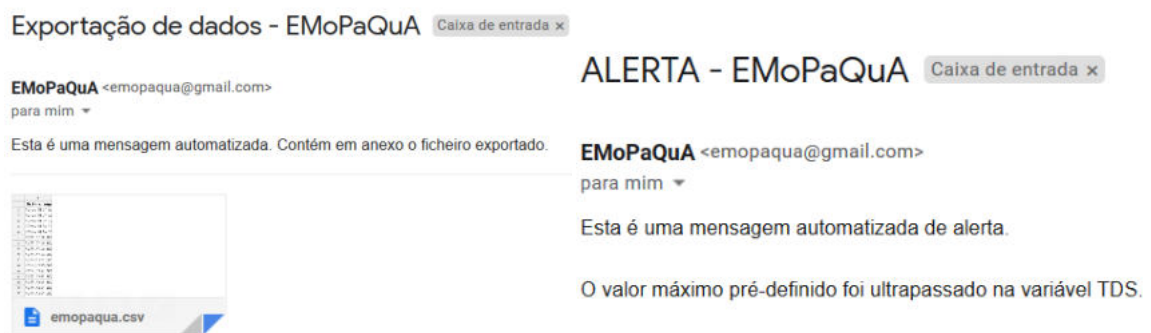


Figura 5.12: E-mail de exportação.

Figura 5.13: E-mail de alerta.

5.7 Testes de integração

No momento da implementação dos diferentes módulos foram realizados testes unitários aos mesmos, para verificar se se encontram em normal funcionamento. Depois da conexão de todos os elementos do sistema, definidos na arquitetura no capítulo 4, é necessário testar o funcionamento do sistema na sua globalidade.

Testes de comunicação de valores

Os testes da comunicação de valores foram executados após o dispositivo estar concluído, de forma a verificar se os valores recolhidos eram válidos e se a própria comunicação prosseguia sem sobresaltos.

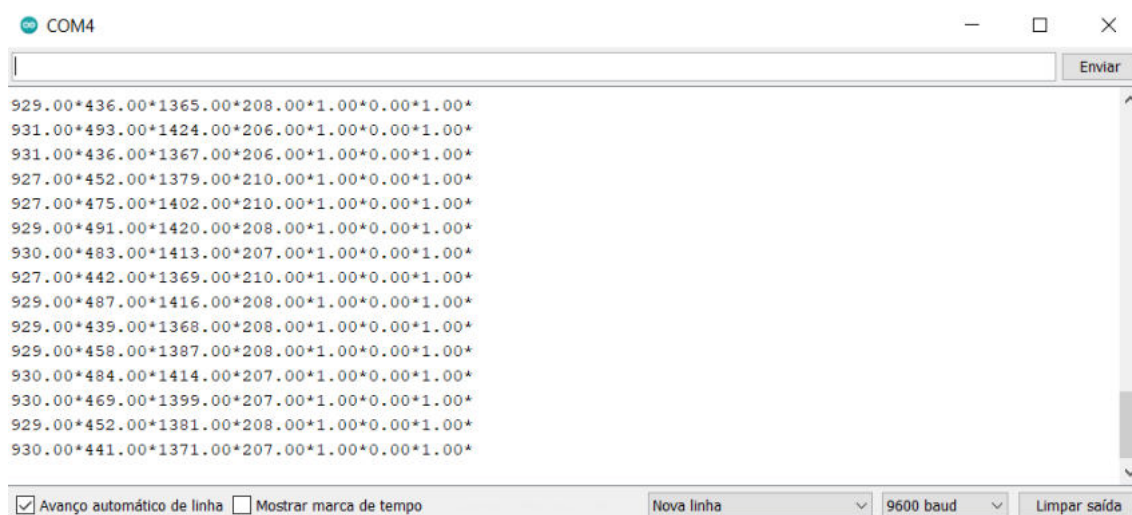


Figura 5.14: Interface.

Na imagem é visível o envio contínuo das mensagens com os valores dos vários sensores, sendo que correspondem ao expectável, considera-se um teste de sucesso. O microcontrolador tratou de enviar os dados, recolhidos com qualidade, no intervalo definido.

Testes de comunicação com a central de comando

Tendo a comunicação sido testada através do terminal, este teste serviu para verificar o funcionamento da central de comando, ao nível da receção e conversão de dados, bem como o armazenamento desses mesmos dados num ficheiro de tabela.

Na figura 5.15, no lado direito é possível visualizar os dados convertidos em números, com o minuto em que foram recolhidos e as unidades SI aplicáveis. Ao seu lado, à esquerda, está visível o gráfico da variável seleccionada, que no caso é a temperatura, podendo o utilizador alterar qual seria o gráfico a visualizar.

O teste decorreu com total sucesso, pois, da parte do dispositivo, os dados foram recolhidos e enviados, e, da parte da central de comando, foram recebidos, convertidos, exibidos e armazenados.

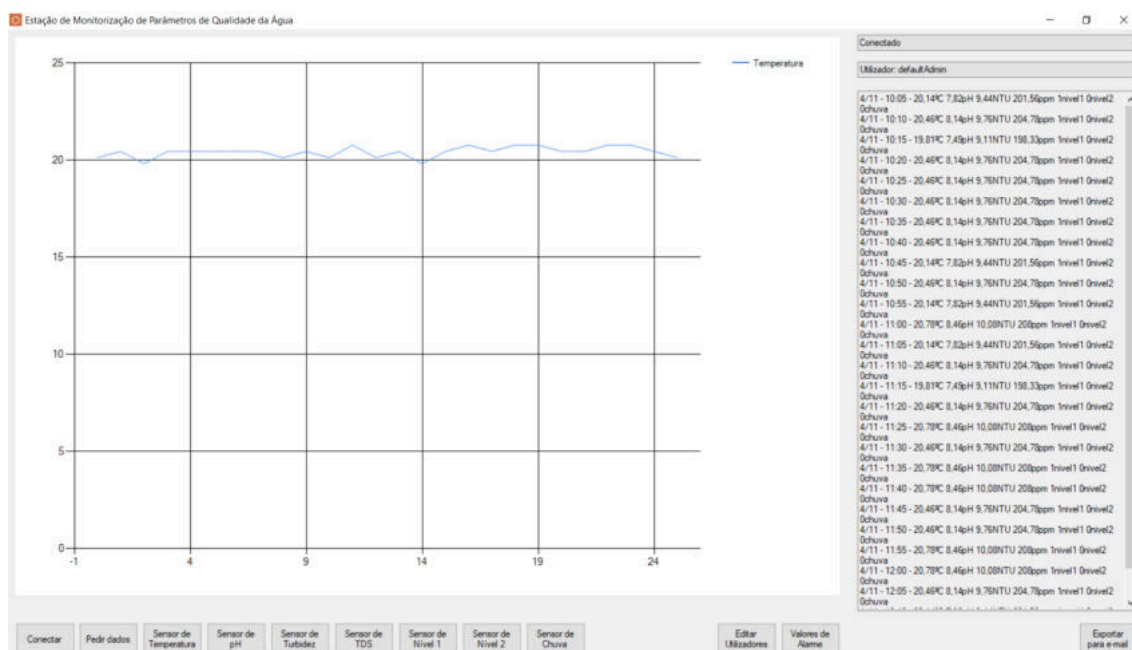


Figura 5.15: Comunicação com a central de comando.

Testes de consumo energético

Nas figuras 5.16 e 5.17 podem-se ver os resultados dos testes ao consumo energético do dispositivo, sendo que à esquerda está a corrente consumida pelo dispositivo enquanto está ativo e à direita quando entra no modo de baixo consumo. Os valores são os sublinhados pela linha vermelha, em cada imagem.

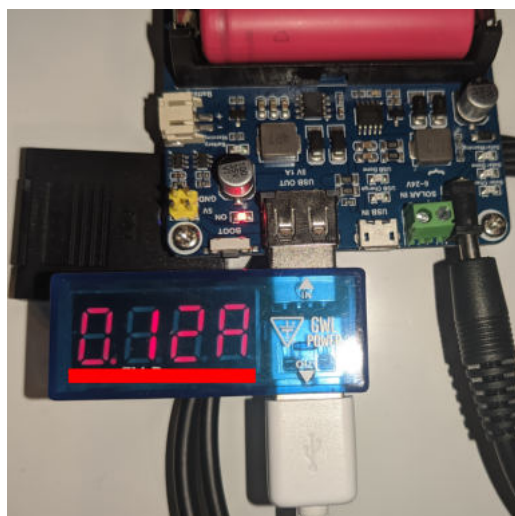


Figura 5.16: Consumo energético em modo ativo.

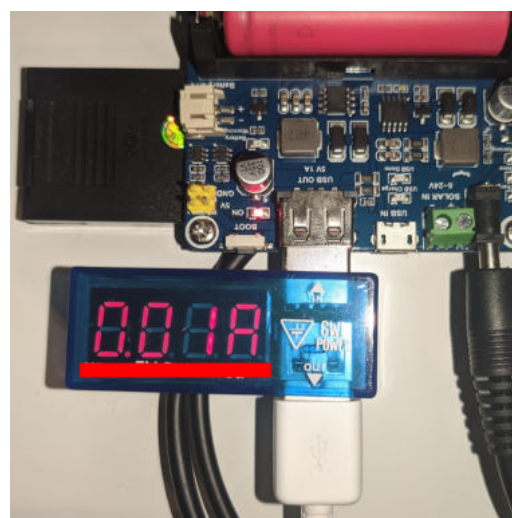


Figura 5.17: Consumo energético em poupança de bateria.

Considera-se então que as definições de baixo consumo energético aplicadas ao projeto tiveram sucesso, pois existe uma redução grande no consumo por parte do dispositivo.

Teste de duração energética

Para avaliar se de facto a fonte de energia seleccionada satisfaz as questões energéticas do problema, foi executado um teste à duração que o dispositivo atinge sem qualquer intervenção humana ao nível da carga. Tratou-se de um teste limitado, de 2 dias. O dispositivo foi colocado num local com exposição solar, como aconteceria na sua aplicação real.

Os resultados deste teste foram satisfatórios. No fim destes 2 dias, onde o dispositivo manteve o seu normal funcionamento, os LEDs indicadores de carga ainda assinalavam o máximo. Pode-se então concluir que a capacidade de fornecimento de energia por parte do painel solar e o seu módulo de armazenamento é suficiente para o caso.

Ao nível da duração energética, só a instalação do dispositivo protótipo, no local, permitirá tirar conclusões em relação à capacidade, por exemplo em dias com total ausência de sol.

Capítulo 6

Conclusões

Neste capítulo são tiradas ilações em relação ao trabalho desenvolvido, com uma explicação do que foi feito, seguido de uma apreciação, e por fim são apresentadas possíveis soluções de futuro de expansão e melhoria do projeto.

6.1 Projeto e solução

O projeto tinha como objetivo o desenvolvimento de uma estação de monitorização de parâmetros da água. Para atingir isso, foi feito um extensivo trabalho de pesquisa ao nível do estado da arte, tanto ao nível tecnológico como ao nível científico, afeto à hidrologia.

A pesquisa efetuada permitiu chegar à solução de desenvolver um dispositivo IoT. Este dispositivo deveria possuir sensores de pH, temperatura, turbidez e TDS, bem como sensores de chuva e de nível da água, de modo a permitir, ter uma ideia muito ligeira de alguns aspetos físicos. A comunicação escolhida foi a rede móvel GSM, pois é o meio de transmissão com maior alcance e o único capaz de satisfazer o problema da colocação do dispositivo numa localização remota. Como sistema de computação foi selecionado um microcontrolador STM32, aproveitando a possibilidade que este oferece de utilizar FreeRTOS. Por fim, para a central de comando, foi criado um projeto em C#, que permite o envio de avisos através de e-mail, que era também um dos objetivos da dissertação.

6.2 Discussão

Comparativamente com os projetos no mercado e no mundo académico, de solução ao mesmo problema, a solução apresentada é como bastante interessante, pois utiliza uma unidade de processamento mais fiável e expansível do que as habituais.

Considera-se que a intenção do projeto foi atingida, e o trabalho desenvolvido poderá servir como base para a criação de um produto.

Além disto, mantém-se a ideia inicial da pertinência do projeto. É um mercado pouco explorado e necessitado, e o projeto desenvolvido poderá ser uma resposta a essa necessidade.

6.3 Soluções de futuro

Vários aspetos poderiam complementar o que foi desenvolvido. Por exemplo, para central de comando, utilizar algo como a Raspberry Pi, que permite a criação de uma interface, a um preço muito mais barato do que um computador normal, além de permitir também as comunicações que este permite, como o Wi-Fi e Bluetooth.

Como é também prática corrente nos dias de hoje, o dispositivo poderia ser complementado, além ou em vez da central de comando, por uma app móvel. É também uma das caras principais do IoT, a ligação dos dispositivos ao telemóvel, e poderia ser interessante complementar o sistema com isso mesmo.

Além disto, no caso de um dispositivo sem intenção de expandir, a placa seleccionada poderia ter sido outra de tamanho inferior, pois existem placas da marca com valências suficientes para aplicar este projeto e com medidas bastante inferiores, que diminuiriam o tamanho do produto. A placa escolhida foi utilizada por se tratar apenas de um produto protótipo, e de modo a diminuir os custos inerentes ao desenvolvimento do mesmo. Apesar disso, o programa poderia ser implementado noutra, pois existe essa portabilidade.

Não se tratando de pontos que falharam para atingir os objetivos propostos, tratam-se certamente de opções que tornariam o sistema mais interessante.

Capítulo 7

Bibliografia

- [1] AN Prasad, K Al Mamun, FR Islam, and Haq Haqva. Smart water quality monitoring system. In *2015 2nd Asia-Pacific World Congress on Computer Science and Engineering (APWC on CSE)*, pages 1–6. IEEE, 2015.
- [2] SI Samsudin, SIM Salim, K Osman, SF Sulaiman, and MIA Sabri. A smart monitoring of a water quality detector system. *Indonesian Journal of Electrical Engineering and Computer Science*, 10(3):951–958, 2018.
- [3] Bishwajit Paul et al. *Sensor based water quality monitoring system*. PhD thesis, BRAC University, 2018.
- [4] Akshatha A.C., Jevita Zeena D’Silva, Likhithashreedevi, and Madhumith Karanth. Arduino based water quality monitoring system. Karnataka State Council for Science and Technology, 2017.
- [5] Y. Lasya, D. Deeksha, B. Srikanth, and K. Manoj. *Real Time Water Quality Monitoring Through IoT*. PhD thesis, Institute of Aeronautical Engineering, India, 2016.
- [6] Yiheng Chen and Dawei Han. Water quality monitoring in smart city: A pilot project. *Automation in Construction*, 89:307–316, 2018.
- [7] Manish Kumar Jha, Rajni Kumari Sah, MS Rashmitha, Rupam Sinha, B Sujatha, and KV Suma. Smart water monitoring system for real-time water quality and usage monitoring. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 617–621. IEEE, 2018.
- [8] James K Peckol. *Embedded systems: a contemporary design tool*. John Wiley & Sons, 2019.
- [9] Robert Oshana and Mark Kraeling. *Software engineering for embedded systems: Methods, practical techniques, and applications*. Newnes, 2019.

- [10] Grisca Liebel, Nadja Marko, Matthias Tichy, Andrea Leitner, and Jörgen Hansson. Model-based engineering in the embedded systems domain: an industrial survey on the state-of-practice. *Software & Systems Modeling*, 17(1):91–113, 2018.
- [11] Richard Zurawski. *Embedded Systems Handbook 2-Volume Set*. CRC press, 2018.
- [12] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101, 2012.
- [13] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.
- [14] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
- [15] Hans Schaffers, Nicos Komninos, Marc Pallot, Brigitte Trousse, Michael Nilsson, and Alvaro Oliveira. Smart cities and the future internet: Towards cooperation frameworks for open innovation. In *The future internet assembly*, pages 431–446. Springer, 2011.
- [16] Dana Cuff, Mark Hansen, and Jerry Kang. Urban sensing: out of the woods. *Communications of the ACM*, 51(3):24, 2008.
- [17] Edward Ashford Lee and Sanjit A Seshia. *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.
- [18] Wayne H Wolf. Cyber-physical systems. *IEEE Computer*, 42(3):88–89, 2009.
- [19] Radhakisan Baheti and Helen Gill. Cyber-physical systems. *The impact of control technology*, 12(1):161–166, 2011.
- [20] Glenn Schatz. The complete list of wireless iot network protocols. *Link Labs*, 2016.
- [21] Brian Ray. 6lowpan vs. zigbee: Two wireless technologies explained. *Link Labs*, 2014.
- [22] Shadi Al-Sarawi, Mohammed Anbar, Kamal Alieyan, and Mahmood Alzubaidi. Internet of things (iot) communication protocols. In *2017 8th International conference on information technology (ICIT)*, pages 685–690. IEEE, 2017.
- [23] Ministério do Ambiente. Sistema nacional de informação de recursos hídricos, 1995.
- [24] Ahsan Ullah, Shahzabin Reza Mimu, Ayndrila Biswas, Md Golam Rasel, and Mourita Mozib. Smart iot based launch activities monitoring system for reducing the accident occurs on the river. *Journal of Sensor Research and Technologies*, 1(1, 2), 2019.

- [25] Kevin Ashton et al. That 'internet of things' thing. *RFID journal*, 22(7):97–114, 2009.
- [26] Daniel Giusto, Antonio Iera, Giacomo Morabito, and Luigi Atzori. *The internet of things: 20th Tyrrhenian workshop on digital communications*. Springer Science & Business Media, 2010.
- [27] Vaishnavi V Daigavane and MA Gaikwad. Water quality monitoring system based on iot. *Department Electronics & Telecommunication Engineering, Advances in Wireless and Mobile Communications*, 10(5):1107–1116, 2017.
- [28] V Vaishnavi, RC Varshitha, and M Tejaswini. Literature survey on smart water quality monitoring system. *International Journal of Innovations in Engineering and Science*, 2018.

Apêndice A

Anexo

Código do dispositivo

Código escrito para o dispositivo de monitorização, na linguagem C.

```
main.c
1  int main ( void )
   {
3      HAL_Init () ;
      SystemClock_Config () ;
5      MX_GPIO_Init () ;
      MX_ADC1_Init () ;
7
      MX_USART3_UART_Init () ;
9      init_UART3 () ;

11     HAL_PWR_EnableWakeUpPin ( PWR_WAKEUP_PIN4 ) ;

13     MX_FREERTOS_Init () ;

15     osKernelStart () ;

17     while ( 1 )
   {
19     }
   }
```

Listing A.1: int main(void)

Responsável por chamar as funções de configuração, para a camada de abstração de hardware, o clock do sistema, os pinos GPIO e ADC, a comunicação da porta série, os pinos de interrupção e o próprio sistema operativo, FreeRTOS. De um modo resumido, inicializa o programa.

gpio.c

```

void MX_GPIO_Init(void)
2 {
    GPIO_InitTypeDef GPIO_InitStruct = {0};

4
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOF_CLK_ENABLE();
6    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
8    __HAL_RCC_GPIOG_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();
10

    HAL_GPIO_WritePin(GPIOB, LD3_Pin|LD2_Pin, GPIO_PIN_RESET);

12

    HAL_GPIO_WritePin(USB_PowerSwitchOn_GPIO_Port,
14 USB_PowerSwitchOn_Pin, GPIO_PIN_RESET);

    GPIO_InitStruct.Pin = USER_Btn_Pin|EXTI_LEVEL2_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
16    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

18

20    GPIO_InitStruct.Pin = GPIO_RAIN_Pin|GPIO_LEVEL1_Pin|DS18B20_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
22    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

24

    GPIO_InitStruct.Pin = USB_PowerSwitchOn_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
26    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
28    HAL_GPIO_Init(USB_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

30

    GPIO_InitStruct.Pin = USB_OverCurrent_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
32    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(USB_OverCurrent_GPIO_Port, &GPIO_InitStruct);

34

36    HAL_NVIC_SetPriority(EXTI15_10_IRQn, 5, 0);
    HAL_NVIC_EnableIRQ(EXTI15_10_IRQn);
38
}

```

Listing A.2: void MX_GPIO_Init(void)

A função inicializa os pinos GPIO conforme definido no STM32 Cube, sejam entradas ou saídas, tanto booleanas como leituras de conversão analógica para digital.

adc.c

```

1 void MX_ADC1_Init(void)
2 {
3     ADC_ChannelConfTypeDef sConfig = {0};
4
5     hadc1.Instance = ADC1;
6     hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV2;
7     hadc1.Init.Resolution = ADC_RESOLUTION_12B;
8     hadc1.Init.ScanConvMode = ADC_SCAN_DISABLE;
9     hadc1.Init.ContinuousConvMode = ENABLE;
10    hadc1.Init.DiscontinuousConvMode = DISABLE;
11    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
12    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
13    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
14    hadc1.Init.NbrOfConversion = 1;
15    hadc1.Init.DMAContinuousRequests = DISABLE;
16    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
17    if (HAL_ADC_Init(&hadc1) != HAL_OK)
18    {
19        Error_Handler();
20    }
21
22    sConfig.Channel = ADC_CHANNEL_3;
23    sConfig.Rank = ADC_REGULAR_RANK_1;
24    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
25    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
26    {
27        Error_Handler();
28    }
29 }

```

Listing A.3: void MX_ADC1_Init(void)

Esta função inicializa os pinos de ADC de acordo com o configurado no STM32 Cube. Como os pinos escolhidos foram do grupo ADC1, de modo a ser necessário menos código, apenas a função de inicialização relativa a esse mesmo grupo está presente.

freertos.c

```

1 void MX_FREERTOS_Init(void) {
2     osMutexDef(xADCMutex);
3     xADCMutexHandle = osMutexCreate(osMutex(xADCMutex));

```

```

5  osMutexDef(xBufferMutex);
   xBufferMutexHandle = osMutexCreate(osMutex(xBufferMutex));

7

   osSemaphoreDef(xLowPowerSemaphore);
9  xLowPowerSemaphoreHandle = osSemaphoreCreate(osSemaphore(
xLowPowerSemaphore), 1);

11  osMessageQDef(xTemperatureQueue, 10, double);
   xTemperatureQueueHandle = osMessageCreate(osMessageQ(
xTemperatureQueue), NULL);

13

   osMessageQDef(xPHQueue, 10, double);
15  xPHQueueHandle = osMessageCreate(osMessageQ(xPHQueue), NULL);

17  osMessageQDef(xTDSQueue, 10, double);
   xTDSQueueHandle = osMessageCreate(osMessageQ(xTDSQueue), NULL);

19

   osMessageQDef(xTurbQueue, 10, double);
21  xTurbQueueHandle = osMessageCreate(osMessageQ(xTurbQueue), NULL);

23  osMessageQDef(xLevel1Queue, 10, double);
   xLevel1QueueHandle = osMessageCreate(osMessageQ(xLevel1Queue), NULL
);

25

   osMessageQDef(xLevel2Queue, 10, double);
27  xLevel2QueueHandle = osMessageCreate(osMessageQ(xLevel2Queue), NULL
);

29  osMessageQDef(xRainQueue, 10, double);
   xRainQueueHandle = osMessageCreate(osMessageQ(xRainQueue), NULL);

31

   osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0,
128);
33  defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

35  osThreadDef(ReadTurb, read_turb, osPriorityHigh, 0, 128);
   ReadTurbHandle = osThreadCreate(osThread(ReadTurb), NULL);

37

   osThreadDef(ReadTDS, read_tds, osPriorityHigh, 0, 128);
39  ReadTDSHandle = osThreadCreate(osThread(ReadTDS), NULL);

41  osThreadDef(ReadTemperature, read_temperature, osPriorityHigh, 0,
128);
   ReadTemperatureHandle = osThreadCreate(osThread(ReadTemperature),
NULL);

```

```

43     osThreadDef(ReadPH, read_ph, osPriorityHigh, 0, 128);
45     ReadPHHandle = osThreadCreate(osThread(ReadPH), NULL);

47     osThreadDef(DataProcessing, data_processing, osPriorityAboveNormal,
0, 128);
    DataProcessingHandle = osThreadCreate(osThread(DataProcessing),
NULL);

49

51     osThreadDef(ReadLevel1, read_level1, osPriorityHigh, 0, 128);
    ReadLevel1Handle = osThreadCreate(osThread(ReadLevel1), NULL);

53     osThreadDef(ReadLevel2, read_level2, osPriorityHigh, 0, 128);
    ReadLevel2Handle = osThreadCreate(osThread(ReadLevel2), NULL);

55

57     osThreadDef(ReadRain, read_rain, osPriorityHigh, 0, 128);
    ReadRainHandle = osThreadCreate(osThread(ReadRain), NULL);
}

```

Listing A.4: void MX_FREERTOS_Init(void)

Função de inicialização do sistema operativo. Como tal, pois a base do FreeRTOS são as tarefas, nesta função são criadas as mesmas, com as suas configurações correspondentes. Além disso, também os componentes de comunicação entre tarefas, como os semaphores, mutexes e queues, são inicializados. respective handles.

```

uint16_t ReadADC(unsigned char channel)
2 {
    uint16_t temp;
4     static ADC_ChannelConfTypeDef sConfig;

6     switch(channel){
        case 1: sConfig.Channel = ADC_CHANNEL_5;                //ADC channel
of the temperature sensor
8         break;
        case 2: sConfig.Channel = ADC_CHANNEL_9;                //ADC channel
of the pH sensor
10        break;
        case 3: sConfig.Channel = ADC_CHANNEL_13;               //ADC channel
of the TDS sensor
12        break;
        case 4: sConfig.Channel = ADC_CHANNEL_3;                //ADC channel
of the Turb sensor
14        break;
    }
}

```

```

    case 5: sConfig.Channel = ADC_CHANNEL_TEMPSENSOR;    //ADC channel
             of the internal temperature sensor
16 break;
    default: printf("Wrong ADC");                        //Invalid ADC
             channel
18 break;
}
20 sConfig.Rank = ADC_REGULAR_RANK_1;
    sConfig.SamplingTime = ADC_SAMPLETIME_3CYCLES;
22 HAL_ADC_ConfigChannel(&hadc1, &sConfig);

    HAL_ADC_Start(&hadc1);
    HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY);
24 temp = HAL_ADC_GetValue(&hadc1);
    HAL_ADC_Stop(&hadc1);
26 return temp;
28 }

```

Listing A.5: uint16_t ReadADC(unsigned char channel)

Função responsável pela leitura e retorno do valor do pino ADC. É utilizada para a leitura dos sensores analógicos, e tem todos os seus acessos protegidos por um mutex, de forma a evitar que mais do que uma tarefa aceda a ela

```

void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)
2 {
    if (GPIO_Pin == GPIO_PIN_13)
4 {
        Return_Tasks();
6 }
    if (GPIO_Pin == GPIO_PIN_11)
8 {
        flaglevel2=1;
10 Return_Tasks();
    }
12 }

```

Listing A.6: void HAL_GPIO_EXTI_Callback (uint16_t GPIO_Pin)

Função de resposta à interrupção - ISR - para as duas interrupções externas que foram configuradas, a do nível 2 e a do botão localizado na placa, cuja utilidade será forçar uma recolha de dados;

```

void Return_Tasks(void)
2 {

```

```

    UBaseType_t uxSavedInterruptStatus;
4   uxSavedInterruptStatus = taskENTER_CRITICAL_FROM_ISR();
    xTaskResumeFromISR( ReadTemperatureHandle );
6   xTaskResumeFromISR( ReadPHHandle );
    xTaskResumeFromISR( ReadTDSHandle );
8   xTaskResumeFromISR( ReadTurbHandle );
    xTaskResumeFromISR( ReadLevel1Handle );
10  xTaskResumeFromISR( ReadLevel2Handle );
    xTaskResumeFromISR( ReadRainHandle );
12  xTaskResumeFromISR( DataProcessingHandle );
    taskEXIT_CRITICAL_FROM_ISR( uxSavedInterruptStatus );
14 }

```

Listing A.7: void Return_Tasks(void)

Função chamada pelas funções cujo objetivo é fazer correr as tarefas de leitura de valores dos sensores.

```

void StartDefaultTask( void const * argument )
2 {
    for (;;)
4     {
        vTaskSuspend( NULL );
6     }
}

```

Listing A.8: void StartDefaultTask(void const *argument)

Trata-se da função genérica gerada pelo sistema, que nada faz e corre quando não existe nenhuma outra tarefa a correr.

```

void HandleGPIO( GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, QueueHandle_t
    xQueue )
2 {
    double value = -1;
4   value = HAL_GPIO_ReadPin ( GPIOx, GPIO_Pin );           // Read value from
    GPIO
    xQueueSend( xQueue, ( void * ) &value, portMAX_DELAY ); // Send
    the value to the respective queue
6
}

```

Listing A.9: void HandleGPIO(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, QueueHandle_t xQueue)

Função genérica de leitura de um pino de GPIO e envio do valor do mesmo através de uma queue. Toma como argumentos o porto e o pino, para fazer a leitura, e também a queue através da qual deve ser feito o envio dos valores.

```

1 void HandleADC (int channel , QueueHandle_t xQueue)
  {
3     uint16_t adcValue = 0;
    double value=0;
5     xSemaphoreTake( xADCMutexHandle , portMAX_DELAY);
        //Try to obtain the ReadADC Mutex
    adcValue=ReadADC ( channel);
        //Read the ADC value of the sensor
7     value=(double) adcValue;
    xSemaphoreGive( xADCMutexHandle );
        //Release the ReadADC Mutex
9     xQueueSend( xQueue ,( void * ) &value , portMAX_DELAY );
        //Send the value to the respective queue
11 }

```

Listing A.10: void HandleADC(int channel, QueueHandle_t xQueue)

Função genérica de leitura de um pino de ADC e envio do valor do mesmo através de uma queue. Toma como argumentos o canal do ADC, que faz parte necessariamente do grupo ADC1, para fazer a leitura, e também a queue através da qual deve ser feito o envio dos valores.

```

1 void read_turb(void const * argument)
  {
3     for (;;)
        {
5         HandleADC (Turb , xTurbQueueHandle);
        vTaskSuspend ( NULL );
7     }
  }

```

Listing A.11: void read_turb(void const * argument)

Função associada à tarefa de leitura do sensor de turbidez, tratando-se de um sensor analógico, procede à chamada da função genérica com os devidos argumentos.

```

void read_tds(void const * argument)
2 {
    for (;;)
4     {
        HandleADC (TDS , xTDSQueueHandle);
6     vTaskSuspend ( NULL );
    }

```

```

    }
8 }

```

Listing A.12: void read_tds(void const * argument)

Função associada à tarefa de leitura do sensor de TDS, tratando-se de um sensor analógico, procede à chamada da função genérica com os devidos argumentos.

```

void read_ph(void const * argument)
2 {
    for (;;)
4 {
        HandleADC (pH, xPHQueueHandle);
6        vTaskSuspend ( NULL );
    }
8 }

```

Listing A.13: void read_ph(void const * argument)

Função associada à tarefa de leitura do sensor de pH, tratando-se de um sensor analógico, procede à chamada da função genérica com os devidos argumentos.

```

void read_temperature(void const * argument)
2 {
    uint8_t Presence = 0;
4    uint8_t Temp_byte1 , Temp_byte2;
    uint16_t TEMP;
6    float Temperature = 0;

8    for (;;)
    {
10        Presence = DS18B20_Start ();
        DWT_Delay (1);
12        DS18B20_Write (0xCC);
        DS18B20_Write (0x44);
14        DWT_Delay (800);
        Presence = DS18B20_Start ();
16        DWT_Delay (1);
        DS18B20_Write (0xCC);
18        DS18B20_Write (0xBE);
        Temp_byte1 = DS18B20_Read ();
20        Temp_byte2 = DS18B20_Read ();
        TEMP = (Temp_byte2 << 8) | Temp_byte1;
22        Temperature = (float)TEMP/16;
    }
}

```

```

24  xQueueSend( xTemperatureQueueHandle ,( void * ) &Temperature ,
    portMAX_DELAY );
    }
26 }

```

Listing A.14: void read_temperature(void const * argument)

Função associada à tarefa de leitura do sensor de temperatura, tratando-se de um sensor digital de 8-bit, é executada uma sequência de comunicações com o sensor, de modo a obter o valor final.

```

void read_level1(void const * argument)
2 {
    for (;;)
4 {
        HandleGPIO( GPIOA , GPIO_PIN_6 , xLevel1QueueHandle );
        // Call function to read and send GPIO value
6 vTaskSuspend( NULL );
        //Suspend the task after reading
    }
8 }

```

Listing A.15: void read_level1(void const * argument)

Função associada à tarefa de leitura do sensor de nível na posição 1, tratando-se de um sensor digital binário, procede à chamada da função genérica com os devidos argumentos.

```

void read_level2(void const * argument)
2 {
    for (;;)
4 {
        HandleGPIO( GPIOA , GPIO_PIN_11 , xLevel2QueueHandle );
        // Call function to read and send GPIO value
6 vTaskSuspend( NULL );
        //Suspend the task after reading
    }
8 }

```

Listing A.16: void read_level2(void const * argument)

Função associada à tarefa de leitura do sensor de nível na posição 2, tratando-se de um sensor digital binário, procede à chamada da função genérica com os devidos argumentos.

```

void read_rain(void const * argument)
2 {
    for (;;)
4 {

```

```

        HandleGPIO ( GPIOA , GPIO_PIN_4 , xRainQueueHandle );
        // Call function to read and send GPIO value
6      vTaskSuspend ( NULL );
        //Suspend the task after reading
    }
8  }

```

Listing A.17: void read_rain(void const * argument)

Função associada à tarefa de leitura do sensor de chuva, tratando-se de um sensor digital binário, procede à chamada da função genérica com os devidos argumentos.

```

void data_processing ( void const * argument )
2 {
    for ( ;; )
4 {
        if ( uxQueueMessagesWaiting ( xRainQueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xLevel1QueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xLevel2QueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xTemperatureQueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xTDSQueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xPHQueueHandle ) !=0 &&
            uxQueueMessagesWaiting ( xTurbQueueHandle ) !=0 )
6 {
            xQueueReceive ( xTemperatureQueueHandle , &( temperature_buffer[i]
            ), ( TickType_t ) 10 );
8            xQueueReceive ( xPHQueueHandle , &( ph_buffer[i] ), ( TickType_t )
            10 );
            xQueueReceive ( xTurbQueueHandle , &( turb_buffer[i] ), (
            TickType_t ) 10 );
10            xQueueReceive ( xTDSQueueHandle , &( tds_buffer[i] ), ( TickType_t
            ) 10 );
            xQueueReceive ( xLevel1QueueHandle , &( level1_buffer[i] ), (
            TickType_t ) 10 );
12            xQueueReceive ( xLevel2QueueHandle , &( level2_buffer[i] ), (
            TickType_t ) 10 );
            xQueueReceive ( xRainQueueHandle , &( rain_buffer[i] ), (
            TickType_t ) 10 );
14            i ++;
            vTaskSuspend ( NULL );
16        }
    }
18 }

```

Listing A.18: void data_processing(void const * argument)

Agrega todos os valores dos sensores, enviados pelas tarefas de leitura através das queues de mensagem criadas para o efeito, e guarda os mesmos na estrutura de dados.

```

void data\_sending(void const * argument)
2 {
    for(int iteracao <= i, iteracao=0, iteracao++)
4     {
        printf("%0.2f*%0.2f*%0.2f*%0.2f*%0.2f*%0.2f*%0.2f*\r\n",
            temperature_buffer[iteracao], ph_buffer[iteracao], turb_buffer[
            iteracao], tds_buffer[iteracao], level1_buffer[iteracao], level2_buffer
            [iteracao], rain_buffer[iteracao]);
6     }
    i=0;
8 }

```

Listing A.19: void data_sending(void const * argument)

Percorre a estrutura que contém os dados acumulados e envia os mesmos. Além disto, limpa a lista para não enviar valores repetidos e limpar o espaço na memória.

```

void PreSleepProcessing(unsigned long ulExpectedIdleTime)
2 {
    HAL_SuspendTick();
4     HAL_PWR_EnterSTOPMode(PWR_LOWPOWERREGULATOR_ON, PWR_STOPENTRY_WFI);
}

```

Listing A.20: void PreSleepProcessing(uint32_t *ulExpectedIdleTime)

Chamada quando nenhuma tarefa está a ser executada, parando então os ticks do relógio de modo a poupar energia.

```

void PostSleepProcessing(uint32_t *ulExpectedIdleTime)
2 {
    HAL_ResumeTick();
4 }

```

Listing A.21: void PostSleepProcessing(uint32_t *ulExpectedIdleTime)

Chamada quando qualquer tarefa vai ser executada, após um período sem execução das mesmas. Reinicia os ticks do relógio.

lptim.c

```

1 void MX_LPTIM1_Init(void)
{
3     hlptim1.Instance = LPTIM1;
    hlptim1.Init.Clock.Source = LPTIM_CLOCKSOURCE_APB_CLOCK_LPOSC;

```

```

5  hlptim1.Init.Clock.Prescaler = LPTIM_PRESCALER_DIV1;
   hlptim1.Init.Trigger.Source = LPTIM_TRIGSOURCE_0;
7  hlptim1.Init.Trigger.ActiveEdge = LPTIM_ACTIVEEDGE_RISING;
   hlptim1.Init.Trigger.SampleTime =
       LPTIM_TRIGSAMPLETIME_DIRECTTRANSITION;
9  hlptim1.Init.OutputPolarity = LPTIM_OUTPUTPOLARITY_HIGH;
   hlptim1.Init.UpdateMode = LPTIM_UPDATE_IMMEDIATE;
11 hlptim1.Init.CounterSource = LPTIM_COUNTERSOURCE_INTERNAL;
   if ( HAL_LPTIM_Init(&hlptim1) != HAL_OK )
13 {
       Error_Handler();
15 }
}

```

Listing A.22: void MX_LPTIM1_Init(void)

Função que configura o timer de baixo consumo energético, com a devida fonte, prescaler, gatilho de ativação, entre outros.

```

void HAL_LPTIM_MspInit( LPTIM_HandleTypeDef* lptimHandle )
2 {
   GPIO_InitTypeDef GPIO_InitStruct = {0};
4   if ( lptimHandle -> Instance == LPTIM1 )
   {
6       __HAL_RCC_LPTIM1_CLK_ENABLE();
       __HAL_RCC_GPIOG_CLK_ENABLE();
8       GPIO_InitStruct.Pin = GPIO_PIN_14;
       GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
10      GPIO_InitStruct.Pull = GPIO_NOPULL;
       GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
12      GPIO_InitStruct.Alternate = GPIO_AF3_LPTIM1;
       HAL_GPIO_Init(GPIOG, &GPIO_InitStruct);
14  }
}

```

Listing A.23: void HAL_LPTIM_MspInit(LPTIM_HandleTypeDef* lptimHandle)

Ativa o timer, fazendo-o correr.

```

1 void LPTIM1_IRQHandler( void )
   {
3       HAL_LPTIM_IRQHandler(&LptimHandle);
       HAL_NVIC_ClearPendingIRQ( LPTIM1_IRQn );
5  }

```

Listing A.24: void LPTIM1_IRQHandler(void)

Função pré-definida de resposta à interrupção, recarrega o timer e retoma o programa.

usart.c

```
1 void init_UART3 ()
2 {
3     HAL_UART_Receive_IT(&huart3 , &UART3Rx_Buffer [ UART3Rx_index ] , 1);
4 }

```

Listing A.25: void init_UART3(void)

Função que configura a interrupção do pino RX, de forma a receber o caractere através da porta série.

```
void MX_USART3_UART_Init( void )
2 {
3     huart3.Instance = USART3;
4     huart3.Init.BaudRate = 9600;
5     huart3.Init.WordLength = UART_WORDLENGTH_8B;
6     huart3.Init.StopBits = UART_STOPBITS_1;
7     huart3.Init.Parity = UART_PARITY_NONE;
8     huart3.Init.Mode = UART_MODE_TX_RX;
9     huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
10    huart3.Init.OverSampling = UART_OVERSAMPLING_16;
11    huart3.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
12    huart3.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
13    if ( HAL_UART_Init(&huart3) != HAL_OK )
14    {
15        Error_Handler();
16    }
17 }

```

Listing A.26: void MX_USART3_UART_Init(void)

Inicializa a USART3 (que foi a escolhida no Design), ativa os seus serviços de interrupção e permite que o printf seja feito através da mesma, via TX.

```
1 void HAL_UART_RxCpltCallback ( UART_HandleTypeDef* huart )
2 {
3     if ( huart->Instance == USART3 )
4     {
5         if ( UART3Rx_Buffer [ UART3Rx_index -1 ] == 't' )
6         {
7             Return_Tasks();
8             UART3Rx_index=0;
9         }
10        else UART3Rx_index++;
11    }
12 }

```

```

11     UART3Rx_index &= ~(1<<7);
13     HAL_UART_Receive_IT(&huart3, &UART3Rx_Buffer[UART3Rx_index], 1);
        //Reset the interrupt for UART3
    }
15 }

```

Listing A.27: void HAL_UART_RxCpltCallback(UART_HandleTypeDef* huart)

Função pré-definida de resposta à receção de um carater.

Código da central de comando

Código escrito para a central de comando, utilizando C#.

Início de sessão

```

1  public void login_Click(object sender, EventArgs e)
    {
2      DataTable dt = new DataTable();
3      SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT(*) FROM
        listadeutilizadores WHERE utilizador='" + utilizadortextbox.Text + "
        ' AND palavrapasse='" + passwordtextbox.Text + "'", con);
4      sda.Fill(dt);
5      if (dt.Rows[0][0].ToString() == "1")
6      {
7          Form1 frm1 = new Form1();
8          frm1.Activated += new EventHandler(frm1_Activated);
9          frm1.FormClosed += new FormClosedEventHandler(frm1_FormClosed);
10         frm1.Show();
11         this.Hide();
12     }
13     else
14     {
15         MessageBox.Show("Conta nao encontrada", "ERRO",
16             MessageBoxButtons.OK);
17     }
18 }

```

Listing A.28: private void login_Click(object sender, EventArgs e)

Reação ao botão de login. O programa verifica se o nome de utilizador e a palavra-passe existem na base de dados que é guardada de modo encriptado. Se sim, dá acesso à interface principal.

Não existe opção de criação de conta, pois apenas o administrador tem essa capacidade.

Edição da lista de utilizadores

```

private void userselect_SelectedIndexChanged(object sender, EventArgs e)
2 {
    username= userselect.Text;
4    usertxtbox.Text = username;

    using (var connection = new SqlConnection(@"Data Source =.\
6    SQLEXPRESS; Initial Catalog = emopaqua; Integrated Security = True")
    {
8        connection.Open();
        using (var q = new SqlCommand("select palavrapasse ,nomecompleto
        , administrador from listadeutilizadores where utilizador='" +
        username + "'", connection))
10    {
        SqlDataReader rdr = q.ExecuteReader();
12        // Fill the string with the values retrieved
        while (rdr.Read())
14    {
        passwordtxtbox.Text = rdr["palavrapasse"].ToString();
16        fullnametxtbox.Text = rdr["nomecompleto"].ToString();
        admin = rdr["administrador"].ToString();

18        if (admin == "True") MakeAdmin.Checked = true;
20        else if (admin == "False") MakeAdmin.Checked = false;
        }
22    }
    }
24 }

```

Listing A.29: private void userselect_SelectedIndexChanged(object sender, EventArgs e)

Aquando de uma alteração na seleção do utilizador na primeira caixa, todas as outras são atualizadas. Isto é, o nome de utilizador, palavra-passe e nome completo associados ao utilizador são colocados visíveis.

```

private void savechanges_Click(object sender, EventArgs e)
2 {
    using (var connection = new SqlConnection(@"Data Source =.\
4    SQLEXPRESS; Initial Catalog = emopaqua; Integrated Security = True")
    {
        using (var q = new SqlCommand("insert into listadeutilizadores
6        values ('' + usertxtbox.Text + "',' + passwordtxtbox.Text + "',' +
        + fullnametxtbox.Text + "',' + admin + ");", connection))
        {

```

```

        q.Connection.Open();
8      q.ExecuteNonQuery();
    }
10 }
}

```

Listing A.30: private void savechanges_Click(object sender, EventArgs e)

As caixas de texto com o nome de utilizador, palavra-passe e nome completo são editáveis, portanto o administrador poderá alterar estes dados, bem como se é administrador ou não. Após isto acontecer, deverá premir o botão que guarda estas alterações. Esta função é a resposta a esse premir de botão, alterando na base de dados.

```

1 private void RemoveUser_Click(object sender, EventArgs e)
{
3     using (var connection = new SqlConnection(@"Data Source =.\
        SQLEXPRESS; Initial Catalog = emopaqua; Integrated Security = True")
    {
5         using (var q = new SqlCommand("delete from listadeutilizadores
            where utilizador='" + userselect.Text + "';", connection))
        {
7             q.Connection.Open();
            q.ExecuteNonQuery();
9         }
        }
11     usertxtbox.Text = "";
    passwordtxtbox.Text = "";
13     fullnametxtbox.Text = "";
    MakeAdmin.Checked = false;
15 }

```

Listing A.31: private void RemoveUser_Click(object sender, EventArgs e)

Ao premir este botão, o utilizador é removido da base de dados e as caixas de texto são limpas, pois eram afetas a um utilizador que já não existe.

```

1 private void AddUser_Click(object sender, EventArgs e)
{
3     userselect.SelectedIndex = -1;

5     usertxtbox.Text = "";
    passwordtxtbox.Text = "";
7     fullnametxtbox.Text = "";
    MakeAdmin.Checked = false;
9 }

```

Listing A.32: private void AddUser_Click(object sender, EventArgs e)

Ao premir este botão, as caixas de texto são limpas, sendo que o que será inserido nas mesmas serão os dados do novo utilizador, confirmado após o premir no botão de guardar alterações.

Listing A.33: private void Conclusion_Click(object sender, EventArgs e)

Botão de conclusão, fecha a janela e reabre a janela de interface de visualização de dados.

Interface de visualização de dados

```

1 public void serialport_connect(String port, int baudrate, Parity
   parity, int databits, StopBits stopbits)
   {
3     DateTime dt = DateTime.Now;
     usart = new System.IO.Ports.SerialPort(port, baudrate, parity,
     databits, stopbits);
5     try
     {
7         usart.Open();
         connected.Text = "Conectado";
9         connection_state = true;
         usart.DataReceived += new SerialDataReceivedEventHandler(
         usart_DataReceived);
11    }
     catch (Exception ex) {
13        connection_state = false;
         connected.Text = "Desconectado";
15    }
   }

```

Listing A.34: public void serialport_connect(String port, int baudrate, Parity parity, int databits, StopBits stopbits)

A funcionalidade principal do programa de interface é, mais do que qualquer outra coisa, comunicar com o dispositivo. Esta é a função que irá criar a ligação, através da utilização do namespace da Microsoft, System.IO.Ports.

```

private void usart_DataReceived(object sender,
   SerialDataReceivedEventArgs e)
2 {
   MethodInvocation mi = new MethodInvocation(this.ReceivedProcessing);
4   this.BeginInvoke(mi, null);
}

```

Listing A.35: private void DataReceived(object sender, SerialDataReceivedEventArgs e)

Função de interrupção, quando da recepção de uma mensagem por parte do dispositivo invoca a função ReceivedProcessing.

```

1 private void ReceivedProcessing()
2 {
3     int hour = DateTime.Now.Hour, minute = DateTime.Now.Minute, day =
    DateTime.Now.Day, month = DateTime.Now.Month;
4     string phrase = usart.ReadExisting();
5     string[] words = phrase.Split(' ');
6     double temp, ph, turb, tds;
7     int level1=1, level2, rain=1;
8     string strFilePath = @"D:\emopagua.csv";
9     StringBuilder sbOutput = new StringBuilder();

10
11     temp = (((double.Parse(words[0].ToString(), CultureInfo.
    InvariantCulture) / 4095) * 3300) - 760) / 2.5) + 25;
12     ph = 3.5 * double.Parse(words[0].ToString(), CultureInfo.
    InvariantCulture) * (5.0 / 4095);
13     turb = ((((-1120.4 * (double.Parse(words[2].ToString(), CultureInfo.
    InvariantCulture))) * (5.0 / 4095) * (double.Parse(words[2].
    ToString(), CultureInfo.InvariantCulture))) * (5.0 / 4095)) +
    (5742.3 * (double.Parse(words[2].ToString(), CultureInfo.
    InvariantCulture)) * (5.0 / 4095)) - 4352.9);
14     tds = (133.42 * ((double.Parse(words[3].ToString(), CultureInfo.
    InvariantCulture)) * (5.0 / 4095) * ((double.Parse(words[3].ToString()
    , CultureInfo.InvariantCulture)) * (5.0 / 4095) * ((double.Parse(
    words[3].ToString(), CultureInfo.InvariantCulture)) * (5.0 / 4095) -
    255.86 * ((double.Parse(words[3].ToString(), CultureInfo.
    InvariantCulture)) * (5.0 / 4095) * ((double.Parse(words[3].ToString()
    , CultureInfo.InvariantCulture)) * (5.0 / 4095) + 857.39 * ((
    double.Parse(words[3].ToString(), CultureInfo.InvariantCulture)) *
    (5.0 / 4095)) * 0.5))))));
15     level1= int.Parse(words[4].ToString(), CultureInfo.InvariantCulture
    );
16     level2= int.Parse(words[5].ToString(), CultureInfo.InvariantCulture
    );
17     rain= int.Parse(words[6].ToString(), CultureInfo.InvariantCulture);

18
19     if (maxtemp < temp || maxph < ph || maxtds < tds || maxturb < turb)
20         this.mail_alert();
21

```

```

    sbOutput.Append(day + "/" + month + ";;" + hour + ":" + minute + "
    ;;" + temp.ToString("#.##") + ";" + ph.ToString("#.##") + ";" + turb
    .ToString("#.##") + ";" + tds.ToString("#.##") + ";" + level1 + ";"
    + level2 + ";" + rain + ";\n");
23
    File.AppendAllText(strFilePath, sbOutput.ToString());
25
    txtReceive.AppendText(day + "/" + month + " - " + hour + ":" +
    minute + " - " + temp.ToString("#.##") + "C " + ph.ToString("#.##") +
    "pH " + turb.ToString("#.##") + "NTU " + tds.ToString("#.##") + "
    ppm " + level1 + "nivel1 " + level2 + "nivel2 " + rain + "chuva \r\n
    ");
27
    this.chart1.Series["Temperatura"].Points.AddXY(i, temp);
29
    this.chart2.Series["pH"].Points.AddXY(i, ph);
    this.chart3.Series["Turbidez"].Points.AddXY(i, turb);
31
    this.chart4.Series["TDS"].Points.AddXY(i, tds);
    this.chart5.Series["Nivel 1"].Points.AddXY(i, level1);
33
    this.chart6.Series["Nivel 2"].Points.AddXY(i, level2);
    this.chart7.Series["Chuva"].Points.AddXY(i, rain);
35
    i++;
}

```

Listing A.36: private void ReceivedProcessing()

Função que processa então a mensagem recebida. Através de funções pré-definidas, divide as strings com informações temporais e os vários valores, cada um afeto a cada sensor, converte-os para dados reais. Depois disto, adiciona os valores aos gráficos e também ao ficheiro .csv. Por fim verifica se ultrapassam os valores definidos pelo utilizador, o que faria com que o e-mail automático fosse enviado.

```

private void connectattempt_Click(object sender, EventArgs e)
2 {
    serialport_connect("COM7", 9600, System.IO.Ports.Parity.None, 8,
    System.IO.Ports.StopBits.One);
4
}

```

Listing A.37: private void connectattempt_Click(object sender, EventArgs e)

No caso do dispositivo estar desconectado, o utilizador pode utilizar este botão para tentar novamente estabelecer conexão.

```

1 private void DataRequest_Click(object sender, EventArgs e)
{

```

```

3     if ( connection_state )
        usart . Write ( " ti " );
5 }

```

Listing A.38: private void DataRequest_Click(object sender, EventArgs e))

Função de resposta a um clique no botão de pedido de dados ao dispositivo. Irá enviar uma string que, conforme definido no código presente no dispositivo, fará com que este recolha novos valores dos sensores e, conjuntamente com os dados que foram acumulados ao longo do tempo, com o espaçamento temporal definido, os envie para a interface.

```

1 private void buttonTemperature_Click ( object sender , EventArgs e )
{
3     chart1 . Show ( ) ;
    chart2 . Hide ( ) ;
5     chart3 . Hide ( ) ;
    chart4 . Hide ( ) ;
7     chart5 . Hide ( ) ;
    chart6 . Hide ( ) ;
9     chart7 . Hide ( ) ;
}

```

Listing A.39: private void buttonTemperature_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de temperatura, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonPH_Click ( object sender , EventArgs e )
2 {
    chart1 . Hide ( ) ;
4     chart2 . Show ( ) ;
    chart3 . Hide ( ) ;
6     chart4 . Hide ( ) ;
    chart5 . Hide ( ) ;
8     chart6 . Hide ( ) ;
    chart7 . Hide ( ) ;
10 }

```

Listing A.40: private void buttonPH_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de pH, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonTurbidity_Click ( object sender , EventArgs e )
2 {
    chart1 . Hide ( ) ;

```

```

4      chart2.Hide();
      chart3.Show();
6      chart4.Hide();
      chart5.Hide();
8      chart6.Hide();
      chart7.Hide();
10 }

```

Listing A.41: private void buttonTurbidity_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de turbidez, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonTDS_Click(object sender, EventArgs e)
2 {
      chart1.Hide();
4      chart2.Hide();
      chart3.Hide();
6      chart4.Show();
      chart5.Hide();
8      chart6.Hide();
      chart7.Hide();
10 }

```

Listing A.42: private void buttonTDS_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de TDS, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonLevel1_Click(object sender, EventArgs e)
2 {
      chart1.Hide();
4      chart2.Hide();
      chart3.Hide();
6      chart4.Hide();
      chart5.Show();
8      chart6.Hide();
      chart7.Hide();
10 }

```

Listing A.43: private void buttonLevel1_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de nível presente no nível 1, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonLevel2_Click(object sender, EventArgs e)

```

```

2 {
    chart1.Hide();
4    chart2.Hide();
    chart3.Hide();
6    chart4.Hide();
    chart5.Hide();
8    chart6.Show();
    chart7.Hide();
10 }

```

Listing A.44: private void buttonLevel2_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de nível presente no nível 2, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void buttonRain_Click(object sender, EventArgs e)
2 {
    chart1.Hide();
4    chart2.Hide();
    chart3.Hide();
6    chart4.Hide();
    chart5.Hide();
8    chart6.Hide();
    chart7.Show();
10 }

```

Listing A.45: private void buttonRain_Click(object sender, EventArgs e)

Reação ao clique no botão de invocação do gráfico do sensor de chuva, simplesmente oculta todos os outros gráficos e coloca esse como visível.

```

private void EditUsers_Click(object sender, EventArgs e)
2 {
    UserList editUsers = new UserList();
4    editUsers.Show();
}

```

Listing A.46: private void EditUsers_Click(object sender, EventArgs e)

Botão de abertura da janela de edição da lista de utilizadores. Deve estar apenas visível para o administrador.

```

1 private void alarmvalues_Click(object sender, EventArgs e)
{
3    txtReceive.Hide();
    editconclusion.Show();

```

```

5     edittemperature.Show();
    editPH.Show();
7     edittds.Show();
    edittitle.Show();
9     editturb.Show();
    temperaturemax.Show();
11    tdsmax.Show();
    turbmax.Show();
13    phmax.Show();
15 }

```

Listing A.47: private void alarmvalues_Click(object sender, EventArgs e)

Ao premir este botão, surge um menu de edição dos valores de alarme, sendo que o mesmo surge no local do painel de dados não tratados.

```

1 private void MailExport_Click(object sender, EventArgs e)
{
3     MailExport.Hide();
    MailBox.Show();
5     MailSend.Show();
}

```

Listing A.48: private void MailExport_Click(object sender, EventArgs e)

Ao clicar este botão, será apresentado o pequeno menu de envio do ficheiro .csv por e-mail

```

private void MailSend_Click(object sender, EventArgs e)
2 {
    MailExport.Show();
4    MailBox.Hide();
    MailSend.Hide();
6
8    var mailMessage = new MimeMessage();
    mailMessage.From.Add(new MailboxAddress("EMoPaQuA", "empoaqua@gmail
.com"));
10    mailMessage.To.Add(new MailboxAddress("default", MailBox.Text));
    mailMessage.Subject = "Exportacao de dados - EMoPaQuA";
12
    var body = new TextPart("plain")
14    {
        Text = "Esta e' uma mensagem automatizada. Conte'm em anexo o
ficheiro com os dados exportados."
16    };
}

```

```

18     var attachment = new MimeTypePart("image", "gif")
19     {
20         Content = new MimeContent(File.OpenRead(@"D:\emopagua.csv")),
21         ContentDisposition = new ContentDisposition(ContentDisposition.
Attachment),
22         ContentTransferEncoding = ContentEncoding.Base64,
23         FileName = Path.GetFileName(@"D:\testfile.csv")
24     };

26     var multipart = new Multipart("mixed");
27     multipart.Add(body);
28     multipart.Add(attachment);
29     mailMessage.Body = multipart;

30     using (var smtpClient = new SmtpClient())
31     {
32         smtpClient.Connect("smtp.gmail.com", 465, true);
33         smtpClient.Authenticate("empaqua@gmail.com", "EMoPaQuA2020");
34         smtpClient.Send(mailMessage);
35         smtpClient.Disconnect(true);
36     }
37     attachment.Content.Stream.Close();
38 }

```

Listing A.49: private void MailSend_Click(object sender, EventArgs e)

Utilizando o namespace MailKit da Microsoft, que permite em C# enviar e receber e-mails, entre outras funções relacionadas, a função trata de enviar o ficheiro .csv para o endereço indicado.