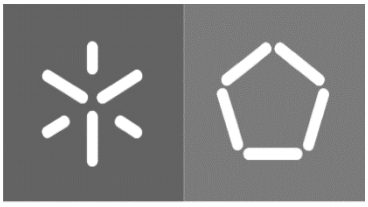
Universidade do Minho
Escola de Engenharia

Nelson Pedro Ferreira da Rocha

Desenvolvimento de sondas para monitorização morfodinâmica

Janeiro 2018



Universidade do Minho
Escola de Engenharia

Nelson Pedro Ferreira da Rocha

68582

Desenvolvimento de sondas para monitorização morfodinâmica

Dissertação de Mestrado

Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do:

Professor Doutor Marcos Martins

Professor Doutor José Pinho

DECLARAÇÃO

Nome: Nelson Pedro Ferreira da Rocha

Endereço eletrónico: nelsonrocha2727@gmail.com Telefone: 919005824

Bilhete de Identidade/Cartão do Cidadão: 14660874

Título da dissertação: Desenvolvimento de Sondas para Monitorização Morfodinâmica

Orientadores:

Professor Doutor Marcos Martins

Professor Doutor José Pinho

Ano de conclusão: 2018

Mestrado em Engenharia Eletrónica Industrial e Computadores

Universidade do Minho, _____/_____/_____

Assinatura:

AGRADECIMENTOS

Esta dissertação marca o final do curso e conclui um longo percurso de aprendizagem, e como não poderia deixa de ser existem pessoas às quais gostaria de deixar uma palavra de apreço e os meus mais sinceros agradecimentos.

As primeiras palavras de agradecimentos vão para a minha família, em especial aos meus pais e ao meu irmão, por todo o apoio prestado neste longo percurso académico, quer a nível emocional, pedagógico e até mesmo financeiro, que me permitiu chegar a esta etapa final.

Não menos importante, gostaria de agradecer ao meu orientador Professor Doutor Marcos Martins e ao meu coorientador Professor Doutor José Pinho por todo o apoio prestado, paciência e boa vontade que sempre disponibilizaram.

Outra palavra de apreço vai para os departamentos de Engenharia Eletrónica Industrial e Computadores e de Engenharia Civil da Universidade do Minho pela disponibilização de condições para poder concluir esta dissertação. Em especial, agradecer ao Sr. Carlos do DEI e ao Sr. Rui do DEC, que sempre se mostraram disponíveis para prestar auxílio.

Agradecer ainda a um dos maiores pilares ao longo destes anos que foram sem dúvida os colegas de curso: João Barros, Pedro Osório, Luís Costa, André Pereira, Pedro Pereira, Tiago Matos, Miguel Costa, Nicolas Almeida, João Sepúlveda, Christophe Barardo, ... (felizmente a lista é extensa!) por todos os momentos de camaradagem vividos, apoio, sugestões e ideias trocadas.

Agradecer por fim, àqueles amigos fora da universidade que sempre estiveram presentes também, o pessoal do grupo D.

A todos muito obrigado!

RESUMO

O principal objetivo desta dissertação é o desenvolvimento de um módulo sensorial que atue de forma autónoma em ambientes aquáticos para recolha de variáveis físicas, que não são facilmente acessíveis. O intuito é que essa recolha permita, posteriormente, a peritos analisar essa informação para perceber e antecipar alterações nestes ambientes. O problema assume um contorno de grande importância, uma vez que nos dias de hoje o ser humano está cada vez mais dependente dos ecossistemas aquáticos para a sua sobrevivência (recolha de bens alimentares, exploração de recursos naturais, prevenção de catástrofes naturais, ...), daí a sua monitorização ser fundamental para perceber previamente de que forma estes ecossistemas vão reagir e assim permitir aplicar abordagens corretas perante essas mesmas alterações. Para tal, nesta dissertação é proposto o desenvolvimento de uma sonda eficiente e sustentável que se dedique à recolha dessa informação em locais de difícil acesso. A sonda estará dotada de várias funcionalidades: (i) sensores, para a medição de pressão e temperatura debaixo de água; (ii) microcontrolador, que trata de todo o processamento inerente e faz a interface com os restantes periféricos; (iii) memória, para armazenar a informação recolhida de forma segura; (iv) placa de comunicação, para transmissão dos dados para uma estação de tratamento, através de um *webserver;* (v) alimentação, o sistema é munido de baterias que facultem o seu funcionamento de forma independente debaixo de água. A sonda que é proposta nesta dissertação tem a grande vantagem de ser de muito baixo custo quando comparada com as demais, como por exemplo os sistemas CTD [1][2, Ch. 14]. Esta sonda, oferece então soluções em resposta às desvantagens que outros equipamentos possam demonstrar e foi cuidadosamente concebida de forma a culminar num produto o mais *cost-efficient* possível.

Palavras-Chave: Monitorização, hidrografia, *data logger*, IoT, *cost-efficiency.*

ABSTRACT

The main objective of this thesis is the development of a sensorial module that operates autonomously in aquatic environments to collect physical variables, which are not easily accessible and that later allow experts to analyze this information in order to perceive and anticipate changes in these environments. The problem assumes contours of great importance, as human beings today are increasingly dependent on aquatic ecosystems for their survival (gathering of food, exploitation of natural resources, the presentiment of natural disasters, etc.), hence its monitoring is fundamental to realize in advance how these ecosystems will react and thus allow the human being to apply correct approaches to these same changes. For this, in this dissertation it is proposed the development of an efficient and sustainable probe that is dedicated to the extraction of this information of difficult access. The probe will be equipped with several functionalities: (i) sensors, for the measurement of pressure and temperature underwater; (ii) microcontroller, which handles all the inherent processing and interfaces with the other peripherals; (iii) memory, to safely store the information collected; (iv) communication board, for transmitting data to a treatment station, via a webserver; (v) power, the system is equipped with batteries that allow it to function independently underwater. The probe that is proposed in this dissertation has the great advantage of being of very low cost when compared with the others, as for example the CTD systems [1] [2, Ch. 14]. This probe then offers solutions in response to the disadvantages that other equipment can demonstrate and has been carefully designed to culminate in a product as cost-efficient as possible.

Keywords: Monitoring, hydrography, *data logger*, IoT, *cost-efficiency.*

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS, ACRÓNIMOS E SIGLAS

A

ADC. *Analog Digital Converter*

APN. *Acess Point Name*

B

BCD. *Binary Coded Decimal*

BPSK. *Binary Phase Shift Keying*

C

CPU. *Central Processing Unit*

CRC. *Cyclic Redundancy Check*

D

DOM. *Document Object Model*

F

FTP. *File Transfer Protocol*

G

GPS. *Global Positioning System*

GSM. *Global System for Mobile Communications*

GUI. *Graphical User Interface*

H

HTML. *HyperText Markup Language*

HTTP. *HyperText Transfer Protocol*

I

I^2C. *Inter-Integrated Circuit*

IP. *Internet Protocol*

N

NMEA. *National Marine Electronics Association*

P

PCB. *Printed Circuit Board*

PDN. *Packed Data Network*

PDP. *Packed Data Protocol*

PHP. *HyperText Preprocessor*

PIN. *Personal Identification Number*

PROM. *Programmable Read-Only Memory*

PUK. *Personal Unlocking Key*

PVC. *Polyvinyl Chloride*

R

RAM. *Random Acess Memory*

RTC. *Real Time Clock*

S

SDIO. *Secure Digital Input Output*

SMA. *Subminiature version A*

SPI. *Serial Peripheral Interface*

SQL. *Structured Query Language*

U

UART. *Universal Assynchronous Receiver Transmiter*

1. INTRODUÇÃO

Os oceanos e os cursos de água têm sido alvos de investigação e da engenharia há milhares de anos, contudo o estudo destas áreas só tem tomado proporções maiores desde o século XX, devido à necessidade de edificação de grandes obras hidráulicas.

Os primeiros relatos da preocupação do Homem com fenómenos desta natureza remontam a 3000 a.C., onde o rio Nilo foi represado para aumentar a produtividade agrícola de terras que até então eram estéreis, naquela que é assumida como sendo a barragem mais velha de grande escala do mundo, Sadd-El-Kafara, situada a sensivelmente trinta quilómetros da atual cidade de Cairo no Egipto [3]. Por volta dos anos 2500 a.C., enquanto os egípcios estavam empenhados nas construções das pirâmides, a civilização harapeana instalada no rio Indo, atual Paquistão, começava a edificar as primeiras cidades indianas, construindo altos diques para obstruir a entrada de água nas suas cidades [4]. Alguns anos mais tarde, a nordeste do rio Indo, subsistia um problema de controlo de cheias na antiga China, onde constantes inundações arrasavam o coração do território chinês, impedindo o desenvolvimento económico e social do país. Aqui as barragens e diques construídos ao longo do rio Amarelo eram ineficazes e a solução passou por, em vez de reprimir diretamente o fluxo do rio, criar um sistema de canais de irrigação que aliviasse as águas da inundação nos campos, e ainda para além disso, fazer um esforço adicional de dragar os leitos dos rios. O sucesso destas alternativas permitiu assim que a cultura chinesa florescesse ao longo das margens do rio [5]. Estas primeiras obras eram assentes em bases totalmente empíricas, as que tinham bom desempenho eram usadas como modelo para as futuras construções, as que por ventura colapsassem eram remodeladas baseado naquilo que se pensasse ser a causa do erro.

Mais recentemente, por volta do século XVII, o interesse do Homem no estudo desta matéria torna-se mais evidente devido ao trabalho de investigadores como Pierre Perrault, Edmé Mariotte e Edmond Halley, que conseguiram demonstrar, quantitativamente, algumas ideias já teoricamente insinuadas por nomes como Leonardo Da Vinci, que estavam subordinadas até então, relativas à existência de um ciclo hidrográfico [6]. Perrault mediu a precipitação na bacia hidrográfica do rio Sena e, estimando o caudal do rio, constatou que a chuva e a neve eram suficientes para explicar o fluxo do rio. Paralelamente, Perrault analisou a evaporação do rio e verificou que vários volumes de água se podiam perder para a atmosfera. Mariotte, também no rio Sena, combinou a velocidade e a área da secção transversal do rio para obter o caudal do rio confirmando os resultados de Perrault

e constatou ainda que os caudais das nascentes dos rios aumentavam consideravelmente aquando a queda de precipitação. Já Halley comprovou que a evaporação que ocorre no mar Mediterrâneo compensa a soma dos deflúvios de todos os rios que nele desaguam, justificando assim a conservação do nível das águas. Estes investigadores foram pioneiros no estudo de cursos de água, a partir desta daqui eclodiu um desenvolvimento da ciência hidrológica, como necessidade do desenvolvimento económico e tecnológico das diversas culturas humanas.

Os estudos destas áreas foram progredindo de forma notável e, no século XX, o empirismo é posto um pouco de lado, sendo a abordagem destas temáticas feita mais em bases teóricas, graças aos avanços da compreensão física dos processos hidrológicos e oceanográficos e pelo aparecimento de sistemas computorizados, especialmente os sistemas de informação geográfica [7].

Todavia, quando anteriormente construções feitas perto ou mesmo juntas aos rios resultavam em fracassos, era principalmente devido a estimativas insuficientes de previsão de cheias. Estes insucessos traziam consequências desastrosas, desde logo a destruição de propriedades, ferimento ou perda de vidas humanas, perda de produtividade e carência de lucros desses empreendimentos, que tinham repercussões sobre a economia das nações afetadas. Hoje em dia usando vários métodos analíticos e técnicas científicas, os especialistas através da recolha e análise de dados obtidos no campo conseguem cooperar na solução de problemas relacionados com a água, tal como a preservação do ambiente, prevenção de catástrofes naturais e gestão dos recursos hídricos.

Esta dissertação surge precisamente no âmbito do estudo desses processos hidrológicos, como sendo uma solução para a monitorização de grandezas físicas nestes ambientes, que oferece uma relação de eficiência e de preço o mais profícua possível para quem visa controlar e prever alterações nestes meios, em alternativa a soluções mais dispendiosas que existem.

1.1 Motivação

A resposta à necessidade de estudar variáveis físicas subaquáticas, advém do facto de que nos dias de hoje o ser humano estar cada vez mais dependente dos ecossistemas aquáticos. Pode-se dividir esta dependência em três grandes pontos [8]:

- Porque extrai-se bens alimentares, portanto há uma necessidade de conhecer os processos naturais que influenciam estes ecossistemas, tal como os agricultores precisam de saber as condições meteorológicas que se avizinham para cultivarem as suas terras.
- Porque simplesmente usa-se, como em construções nas costas marítimas, construções *offshore*, transporte, extração de petróleo e gás, efeitos de recreação (nadar, pescar, surfar, mergulhar, ...), pelo que há também necessidade de conhecer os processos naturais que afetam estas atividades.
- Porque influencia as condições climatéricas, nomeadamente a distribuição da chuva, secas, cheias, desenvolvimento de uma tempestade, furacões, ciclones, ... logo há interesse em perceber as relações entre o mar e a atmosfera de modo a analisar os padrões climatéricos para antecipar catástrofes naturais e ameaças à biodiversidade.

Dependentemente do tópico de estudo o módulo sensorial será uma mais valia para a hidrografia e para a oceanografia, em que o objetivo final é a previsão de acontecimentos naturais, isto porque, para prever um determinado acontecimento os especialistas precisam de aliar os seus conhecimentos teóricos aos dados recolhidos no campo, para que possam aplicar métodos numéricos próprios, que por sua vez concedem uma perceção de todo o sistema envolvente, para conduzir a estados futuros desse sistema criando assim uma previsão. Na figura seguinte está representado um esquema que sintetiza esse processo:

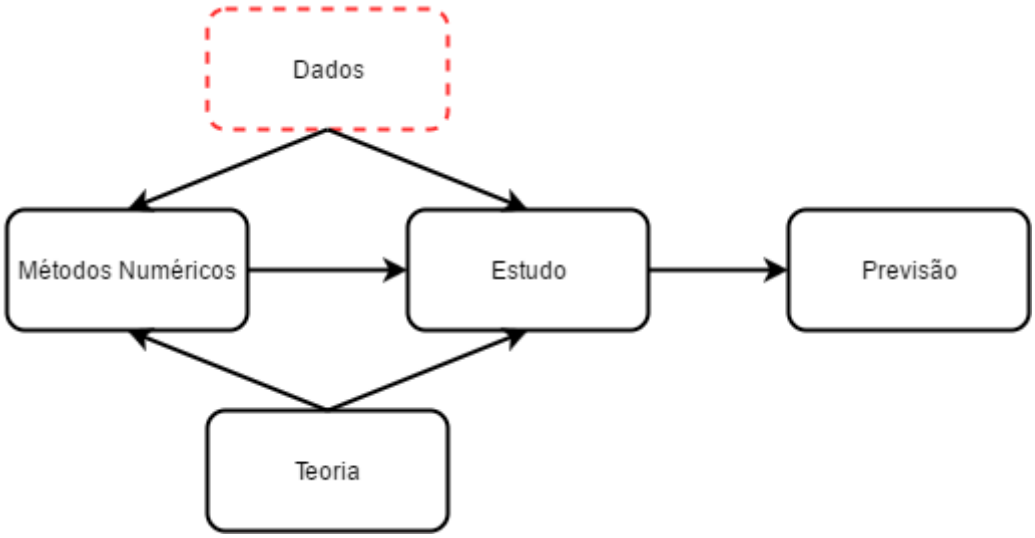


Figura 1.1 - Processo de Previsão

Sendo que a dissertação recairá sobre o bloco de dados, mais propriamente no que diz respeito à sua aquisição, tratamento e transmissão dos mesmos.

1.2 Objetivos

O principal objetivo desta dissertação passa por dar resposta à necessidade de produzir um módulo sensorial para a medição de variáveis físicas subaquaticamente com a melhor relação de *cost-efficiency*. A sonda proposta deverá então cumprir certos requisitos fundamentais para um correto e eficiente funcionamento da mesma:

- Apresentar impermeabilidade e resistência aquática;
- Medição de pressão (0 a 14 bar) e temperatura (-40°C a 50°C);
- Armazenamento da informação recolhida de forma detalhada numa unidade de memória;
- Operar em *Real Time Clock*;
- Comunicação via GSM para uma estação exterior;
- Geração de uma *back-end*;
- Sistema de alimentação para funcionamento independente;
- Possibilidade de configuração do ajuste de frequência de medição;
- Possibilidade de colocar periféricos do módulo em *sleep mode* para melhor autonomia;
- Incorporação de um sistema de GPS;
- Desenvolvimento de uma interface gráfica;

1.3 Estrutura da Dissertação

O conteúdo que se sucede nesta dissertação está organizado por diferentes capítulos, em que, cada um tem debruça-se sobre matérias distintas. O capítulo 2 é referente ao estado da arte, pelo que, neste caso, se examinam vários outros tipos de sondas usadas para efeitos semelhantes ao desta dissertação, bem como uma evolução cronológica destas. O capítulo 3 faz uma análise ao sistema, isto é, oferece uma visão da arquitetura global do sistema detalhando cada um dos diversos módulos e interfaces usados. O Capítulo 4 é o mais extenso e diz respeito a toda a implementação de *software* feita, onde cada um dos diversos módulos enunciados no capítulo anterior são dissecados de forma independente. O capítulo 5 refere-se mais à integração de *hardware* no sistema (antenas, PCB), mas não só, também do ensaio e estruturação de um invólucro impermeável. O

Capítulo 6 é relativo aos testes realizados, que comprovam o correto funcionamento do sistema como um só. E por fim, o Capítulo 7 sumariza as contribuições da tese e abre discussão para futuras melhorias que possam a vir ser implementadas.

2. ESTADO DA ARTE

O desenvolvimento de instrumentos de estudo de ambientes aquáticos surge por volta de 1960 por J. M. Snodgrass, que foi um engenheiro eletrónico e oceanógrafo físico [9], que até então eram algo obsoletos. Snodgrass aponta a segunda guerra mundial como sendo um grande impulsionador no desenvolvimento deste tipo de instrumentos, mais propriamente a guerra anti-submarina [10], que forçou o rápido desenvolvimento da acústica subaquática [11].

O primeiro instrumento usado por Snodgrass a apresentar sucesso foi o "bottom-sediment temperature-gradient recorder", este equipamento (figura 3), ainda que simples aos olhos de hoje, trouxe grandes resultados para a geofísica.

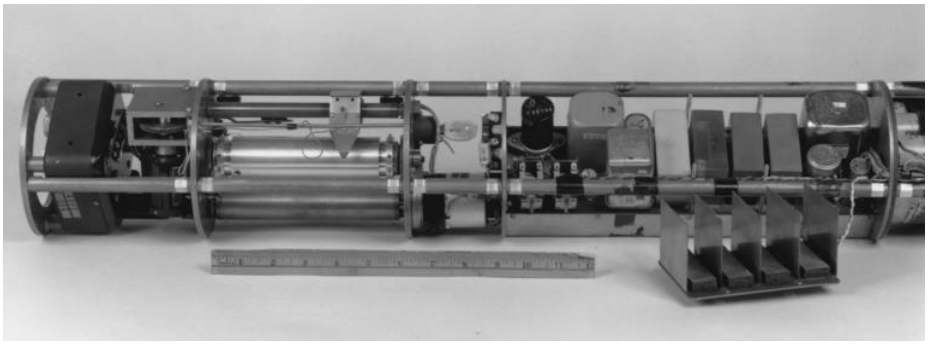


Figura 2.1 - Bottom-Sediment Gradient Temperature Recorder [12]

O aparelho calcula a diferença de temperatura entre dois pontos nos sedimentos devido aos dois termístores distanciados que tem acoplados a si. O aparelho foi usado em várias expedições apresentando resultados positivos relativos ao estudo oceanográfico, como por exemplo na expedição "MidPacific" em 1950 [13] e mais tarde em 1952 na expedição "Capricorn" [14].

Atualmente, a tecnologia evoluiu de tal forma que é possível explorar os sistemas aquáticos cada vez mais de forma sistemática, científica e de forma não invasiva. Esses avanços tecnológicos resultam no aparecimento de certas soluções para medição das grandezas físicas de pressão e temperatura, como é o caso dos CTD, BPR, AUV e o sistema ARGO.

2.1 CTD

De uma forma muito resumida e indo de encontro ao que a própria sigla significa um CTD mede condutividade, C, temperatura, T, e profundidade, D do inglês "depth", graças à incorporação de três sensores.

Apesar de o nome falar em profundidade, esta não a real variável que este tipo de equipamento mede, mas sim pressão. A relação entre pressão e profundidade é algo complexa, envolvendo densidade e compressibilidade da água, bem como a força do campo de gravidade local (função da latitude) [1].

A condutividade representa a capacidade da água de conduzir corrente elétrica. Água pura apresenta baixa condutividade elétrica, enquanto água com sais dissolvidos na forma de iões tende a conduzir mais corrente elétrica. Através da utilização de elétrodos é possível estimar a quantidade de sais dissolvidos na água [2][15].

A temperatura é medida a partir de alterações na resistência elétrica de um metal, que por sua vez expande ou contrai conforme as mudanças de temperatura na água [2][15].

A pressão é medida a partir de uma câmara preenchida por um fluido, no qual está inserida uma resistência. Uma das paredes dessa câmara é composta de uma membrana flexível. À medida que a pressão aumenta, a membrana pressiona o fluido dentro do compartimento e consequentemente altera a medida da corrente elétrica que passa pela resistência. A partir da medida de pressão é possível identificar a profundidade na qual encontra-se o equipamento [2][15].

Um exemplo de um CTD é o "SEACAT Profiler CTD".

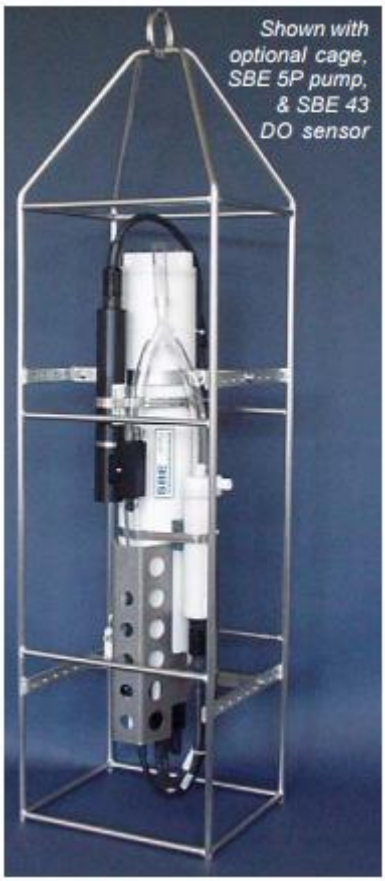


Figura 2.2 - SEACAT Profiler CTD [16, Fig. 1]

Este CTD da Sea-Bird Electronics apresenta alta precisão, resolução e confiança nos seus resultados e pode cobrir um vasto leque de aplicações. O equipamento destacado está continuamente de quatro em quatro segundos a fazer leituras novas, que são registadas dentro do próprio equipamento numa unidade de memória dedicada para esse efeito. Nove pilhas alcalinas do tipo D alimentam o sistema permitindo que este funcione de modo ininterrupto ao longo de sessenta horas, quando está a fazer scan com uma frequência de quatro hertz [16]. Toda a sua estrutura é envolvida e protegida por uma armação para amparar o choque mecânico.

Um outro exemplo de um CTD é o "CTDplus 100, 500 AND 1000" da empresa Sensoren Instrumente Systeme. Este CTD em muito se assemelha ao equipamento da Sea-Bird Electronics, destaca-se por ter uma interface de fácil acesso com o utilizador que que permite o ajuste de várias variáveis como por exemplo a frequência de amostragem, contudo não atinge profundidades tão grandes [17].

Figura 2.3 - CTDplus 500 [17, Fig. 1]

2.2 BPR

Um BPR, “bottom pressure recorder”, mede a pressão da coluna de água acima dele e é sensível o suficiente para prever mudanças milimétricas no nível da água. O instrumento pode ser usado para medir o aumento da água durante a atividade das marés, e também pode ser usado para detetar tsunamis. Quando um tsunami passa sobre um sensor de pressão de fundo, o instrumento deteta que a pressão da água acima dele é significativamente maior do que o normal, e pode alertar os especialistas para a anomalia [15].

Um exemplo de um BPR usado no nordeste pacifico para a averiguação de tsunamis, usa um transdutor de pressão fabricado pela Paroscientific, Inc [18].

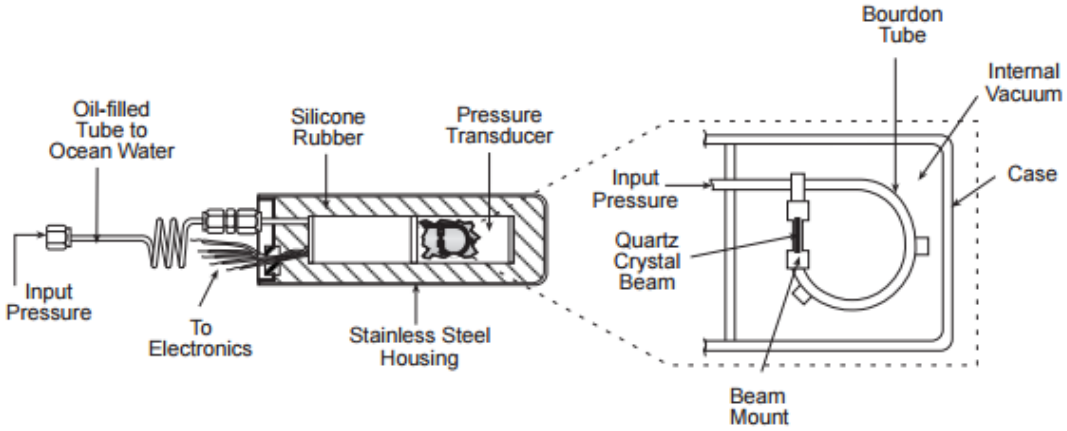


Figura 2.4 - Bottom Pressure Recorder

Esses transdutores usam um feixe de cristal de quartzo muito fino, eletricamente induzido a vibrar com a sua ressonância mais baixa. Tal como ilustra a figura 6, este oscilador é ligado a um tubo de Bourdon que está aberto numa extremidade e está em contacto com a água. Alterações na quantidade de fluido que passa sobre o instrumento provoca o aumento da pressão que faz com que o tubo de Bourdon se desenrole, e como que esticando o cristal de quartzo aumenta a frequência de vibração, ou inversamente, reduz a pressão, permitindo que o tubo de Bourdon enrole mais firmemente, comprimindo assim o cristal de quartzo e abaixando a sua frequência vibração. Estas mudanças de frequência vibratória do cristal de quartzo podem ser medidas precisamente pelo sistema eletrónico do medidor de tsunami e as mudanças de frequência são então convertidas nas correspondentes mudanças na altura do tsunami. Para períodos maiores que um minuto e assim por diante, e para implantações a profundidades de 5000 m, o transdutor é sensível a mudanças na altura da onda menor que um milímetro. Os novos equipamentos também fornecem informações relativas à temperatura muito fidedignas.

2.3 AUV

Ao longo das últimas décadas os engenheiros têm vindo a desenvolver tecnologias submergíveis capazes de combater os desafios que a exploração subaquática impõe aos seus exploradores. Os AUV, autonomous underwater vehicles, são um grande exemplo desses avanços tecnológicos [19].

Um destes veículos é o “Seaglider” [20] da “Kongsberg”. O “Seaglider” é um AUV desenvolvido para medição contínua e autónoma durante longos períodos de tempo de parâmetros. Pare além da hélice elétrica, o veículo contém estratégias de flutuabilidade e asas para criar movimento para a frente.

Figura 2.5 – Seaglider

O veículo move-se na água seguindo um padrão de dente de serra e vem muitas vezes à superfície para determinar a sua posição tal como mostra na figura 6. A navegação é realizada usando uma combinação de GPS quando vem à superfície com os sensores internos que monitorizam o comportamento do veículo durante os mergulhos. Já os sensores externos estão constantemente a varrer o oceano para determinar propriedades ambientais [19].

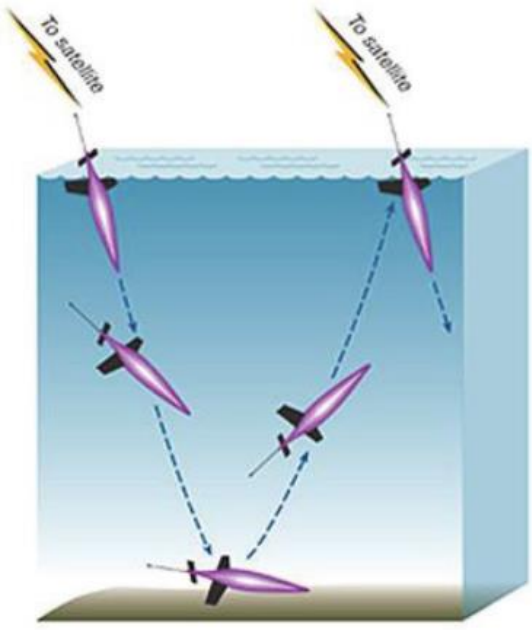


Figura 2.6 - Seaglider padrão de navegação [19, Fig. 5]

O “Seaglider” revolucionou a maneira como os dados oceanográficos são recolhidos. O seu novo método de propulsão usa muito pouca energia. Além disso, o veículo foi meticulosamente projetado para ser tão eficiente quanto possível. O resultado é uma ferramenta de recolha de dados que pode ser lançada ao mar durante meses, em vez das horas ou dias como os sistemas AUV mais comuns. Enquanto que a sua velocidade máxima é baixa, a resistência extremamente longa do veículo permite que ele atravesse milhares de quilómetros só num lançamento [21].

Em condições normais, o veículo aparecerá à superfície após cada mergulho para transmitir dados obtidos e receber comandos. Desta forma, o piloto do sistema pode monitorizar continuamente o desempenho da exploração e a saúde do veículo, fazendo os ajustes necessários. Além disso, o utilizador final pode obter os dados recolhidos em tempo quase real.

Entrando em detalhe mais técnico o “Seaglider” usa para medir temperatura um termístor SBE 3 da Sea-Bird Electronics entre as duas barbatanas e o sensor de pressão é o 211-75-710-05 1500PSIA da Paine Corporation que é do tipo strain gauge [20].

Alternativamente e para alcançar profundidades mais acentuadas existe um modelo muito semelhante da mesma empresa denominado de “Deepglider” (até cerca de 6000 metros de profundidade).

2.4 ARGO

ARGO é um sistema composto por um vetor de mais de 3000 sondas robóticas distribuídas geograficamente por todos os oceanos do planeta, que medem a temperatura, pressão, condutividade, entre outras características. Cada uma dessa sondas está preparada para atingir os 2000 metros de profundidade [22].

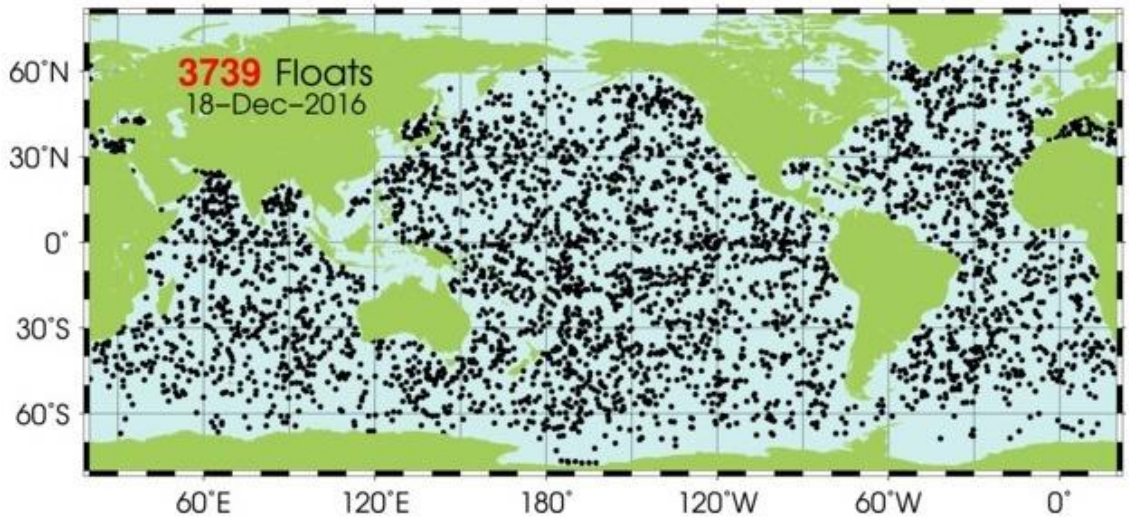


Figura 2.7 - Posição das sondas que transmitiram dados em 18 de dezembro de 2016 [23]

As sondas do sistema ARGO, são na verdade AUV, no sentido em que necessitam de ter a capacidade flutuar, submergir e emergir nos oceanos numa periodicidade pré-estabelecida. Para tal, as sondas atingem esses comportamentos alterando a sua própria densidade. Sendo que a densidade de qualquer objeto, é obtida dividindo a sua massa pelo seu volume, as sondas ARGO, mantêm sua massa constante, mas alteram o seu volume, usando pistões hidráulicos para injetar óleo mineral numa bolsa de borracha na base da sonda, que menos densas, sobem. Quando o período na superfície está completo, os pistões sugam o óleo e a sonda desce novamente [24]. Normalmente este processo ocorre de dez em dez dias.

A antena para comunicação com satélites é montada na parte superior da sonda. Uma vez na superfície, a sonda é basicamente uma boia, permitindo à antena se destacar na superfície para fins de comunicação. Os oceanos são salinos, portanto, condutores de eletricidade, tornando radio comunicação submersa bastante difícil.

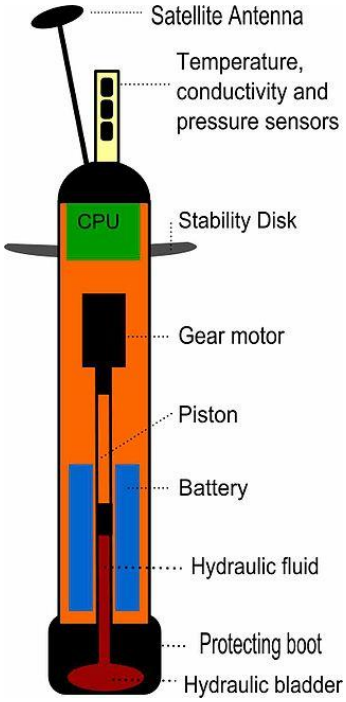


Figura 2.8 - Estrutura da sonda ARGO

A vida útil padrão de uma sonda do sistema Argo é de cinco anos. Depois das baterias se esgotarem, as sondas são comandadas para se afundarem ou mais raramente, flutuar até às praias.

2.5 Vantagem da sonda proposta

Apesar de já existirem alguns equipamentos capazes de se proporem a resolver esta temática, a sonda que é proposta nesta dissertação tem a grande vantagem de ser de muito baixo custo quanto comparada com as demais, como por exemplo os sistemas CTD [1][2, Ch. 14]. Para além do problema de custo que os outros equipamentos dispõem, existem outros aspetos em que estes não são nada práticos e eficazes, especialmente porque em alguns desses equipamentos a recolha de dados só é possível com a extração da sonda fora de água, porque têm escassez de armazenamento de dados, porque têm um consumo energético elevado, porque provocam grande impacto visual no ambiente, entre outros aspetos. Esta sonda, todavia, oferece solução em resposta a todos essas desvantagens e foi cuidadosamente concebida de forma a culminar num produto o mais *cost-efficient* possível.

3. METODOLOGIA E MÉTODOS

Nesta parte é importante fazer uma análise para identificar os vários blocos e tarefas que constituem o sistema a desenvolver. Na figura seguinte está representado um diagrama que compila esses blocos:

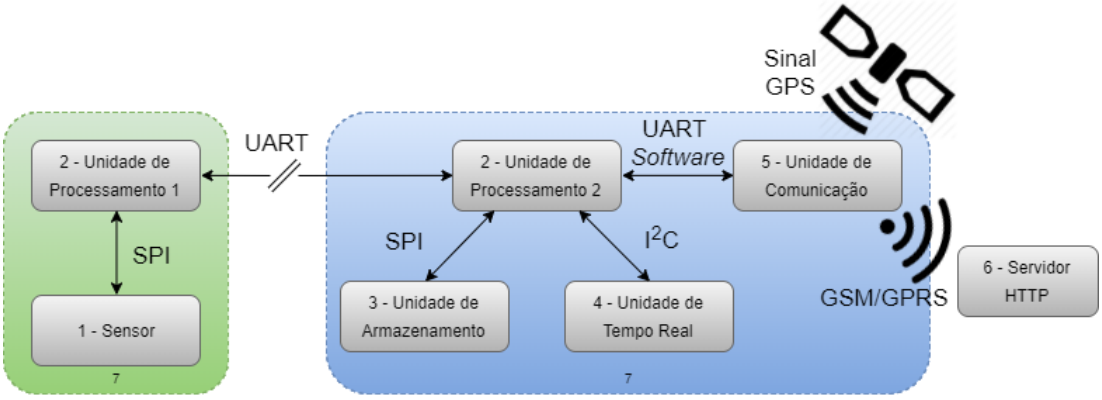


Figura 3.1 - Diagrama de blocos da sonda

Individualmente, cada bloco representa uma função específica, porém só a agregação de todos eles resulta na criação do protótipo final para resolução do problema proposto.

Entrando em detalhe na análise da Figura 3.1, cada bloco consistirá no seguinte:

1. Sensor

Este bloco é das partes mais importantes do projeto, uma vez que é o que trata da recolha das grandezas físicas de pressão e de temperatura, que os especialistas quererão analisar. O sensor escolhido é o MS5541C visto ser um sensor que satisfaz as condições às necessidades a que módulo estará sujeito, mais concretamente, este microsensor piezorestivo apresenta como ponto forte à aplicação em ambiente aquático o facto de ser à prova de água. Para além disso, este sensor apresenta uma gama dinâmica de 0 a 14 bar, o que permite medições a mais de 100 metros de profundidade e com uma resolução bastante boa de 1.2 mbar. Mais ainda, é um microsensor que funciona com baixa voltagem e tem um baixo consumo energético, que para um sistema em que se pretende que atue de forma independente durante algum período de tempo acaba por se tornar bastante vantajoso. O módulo sensorial comunica através de SPI.

2. Unidade de processamento

Nesta unidade pretende-se ter um microcontrolador capaz de corresponder às exigências de processamento da aplicação e com oferta de soluções ao nível das interfaces de comunicação que são impostas pelos restantes periféricos. Existem vários microcontroladores capazes de dar uma resposta a estes requisitos, contudo o *Arduino nano* apresenta-se como o mais *cost-efficient* e aquele com menores dimensões o que proporciona uma melhor gestão do tamanho do protótipo final.

3. Unidade de armazenamento

Os dados recolhidos pelo sensor são do interesse da aplicação que se mantenham guardados e registados de uma forma segura para posterior análise por parte de especialistas. Para tal, o adaptador/leitor de cartões da *SparkFun* aliado a um tradicional cartão de memória *microSD* seria uma solução fidedigna de armazenamento dos dados proporcionando um *backup* em caso de alguma falha, bem como proporcionando uma extensão da escassa memória RAM do microcontrolador. Esta unidade tal como o microsensor será ligada ao barramento de SPI. Para tal a unidade de processamento deverá incorporar uma biblioteca que suporte ficheiros de formato FAT32 e que permita a leitura e escrita no cartão de memória.

4. Unidade de Tempo Real

A necessidade de saber identificar o espaço temporal no qual uma amostra foi recolhida suscitou a integração de um módulo de *Real Time Clock.* Dessa forma, foi adicionado o módulo DS3231 que, ligado por I2C, irá permitir que nunca seja perdida a data e a hora durante a execução do programa. Este módulo, para além de baixo custo, é também de dimensões reduzidas e extremamente preciso.

5. Unidade de comunicação

A transmissão dos dados recolhidos para uma estação assume um dos papéis fulcrais no projeto. Para tal propõe-se o uso do módulo GSM/GPRS SIM808. A ideia passa por adquirir um cartão SIM a uma das operadoras telefónicas com cobertura nacional e com um plafond de dados

móveis, que não tem de ser necessariamente muito grande, transmitir os dados recolhidos a partir da própria sonda para um servidor. Desta forma é mais fácil e mais cómodo analisar os dados extraídos da sonda a partir de um *website* ou mesmo através de uma interface gráfica que interaja igualmente com o servidor. Este módulo de comunicação, que comunica por UART com o microcontrolador, oferece ainda o suporte para operar em Real Time Clock, o que é bastante útil para a análise dos especialistas. O mesmo módulo permite ainda a incorporação de um sistema de GPS na sonda, e uma vez que potencia desta funcionalidade, ela é também integrada no sistema.

6. Servidor

Este bloco diz respeito à implementação de uma aplicação *back-end* e de uma aplicação *front-end,* ou dito de outra forma, diz respeito ao gerenciamento e estruturação dos dados recebidos e enviados pela sonda. Resume-se então, à interação de *scripts* PHP com a base de dados e também à criação de uma interface gráfica, que seja amigável para utilizador, isto é, que seja eficiente, intuitiva e fácil de navegar. A análise dos dados recolhidos e o que os mesmos representam não é da competência desta dissertação.

7. Encapsulamento

Este bloco, fisicamente, deverá encapsular todos os blocos que estão dentro dele e resume-se ao design e produção de um invólucro que ofereça resistência e impermeabilidade à água desenvolvido em PVC. Dentro do encapsulamento estará também uma fonte de alimentação que permita o sistema funcionar de forma autónoma durante um largo período de tempo.

Quando ao modo e às ferramentas usadas no desenvolvimento do protótipo, foram usadas várias linguagens de programação (C, C++, C#, HTML, PHP, SQL) e ferramentas de *software* (Atmel Studio, Visual Studio, Fiddler). Desde logo todo o código, em C, das unidades de processamento foi desenvolvido no IDE Atmel Studio ao contrário daquilo que era a escolha mais óbvia que seria o Arduino IDE, pela simples razão de que, sendo esta uma dissertação em engenharia eletrónica industrial e computadores seria mais desafiante e ao mesmo tempo permitiu programar de certa forma a um mais baixo nível, ao nível dos registos, facultando eventuais otimizações que de outra forma não seriam possíveis. Depois, todo o código desenvolvido relativo à aplicação *back-end*, impôs

a aplicação de conhecimentos das linguagens de PHP e SQL. Nesta fase, há uma ferramenta que se mostrou muito útil, Fiddler, onde foi possível testar o funcionamento da aplicação *back-end,* nomeadamente o modo como esta se comportava ao ser submetida a diferentes tipos de requisitos HTTP. Por fim, a aplicação relativa à interface gráfica foi desenvolvida em C# no IDE Visual Studio da Microsoft.

4. IMPLEMENTAÇÃO DO SISTEMA

Este capítulo é o mais extenso desta dissertação e possivelmente o mais importante, na medida que, é detalhado todo o processo de implementação realizado, descrito de uma forma passível de ser facilmente reproduzido. Nesse sentido, este capitulo descreve pormenorizadamente a maneira como todo o sistema foi abordado, todas as técnicas usadas e alguns fundamentos teóricos, bem como foi implementado.

4.1 Sensor

Por forma a recolher dados relativos à pressão e à temperatura debaixo de água escolheu-se usar o sensor MS5541C. O MS5541C contém um microsensor piezoresistivo e tem incorporado uma interface eletrónica. A sua principal função é converter a tensão de saída analógica não compensada obtida pelo sensor de pressão piezoresistivo num valor digital de 16 *bits*, bem como fornecer o valor digital também de 16 bits de temperatura. Como a tensão de saída do sensor de pressão depende fortemente da temperatura e das tolerâncias do processo, é necessário compensar esses efeitos, pelo que o processo de compensação deve ser feito por *software* usando um microcontrolador externo. Para a medição de ambos valores de pressão e temperatura o sensor usa o mesmo ADC, sigma delta. Durante as duas medições o sensor só estará ligado por curto espaço de tempo para reduzir o consumo de energia.

Cada módulo é individualmente calibrado de fábrica a duas pressões e duas temperaturas. Como resultado, 6 coeficientes necessários para compensar as variações do processo e as variações de temperatura são calculados e armazenados na memória PROM de 64 *bits* do módulo. Esses 64 bits são divididos em quatro *words* de 16 *bits* cada e devem ser lidas pelo *software* do microcontrolador e utilizadas para obter os valores reais de pressão e temperatura.

Uma grande vantagem do uso deste sensor é ter um circuito integrado diretamente adaptado que evita o uso de componentes externos e permite alcançar um consumo energético muito baixo. Esta característica faz com que o sensor seja uma preferência predileta para sistemas portáteis com alimentação feita por bateria. Apesar do seu baixo consumo, o sensor não perde no seu desempenho e apresenta uma precisão e resolução altas, que o torna também adequado para aplicações industriais e automotivas. A possibilidade de compensar o sensor com *software* permite ao utilizador adaptá-lo à sua aplicação especificamente.

A comunicação entre o sensor e um microcontrolador externo é feita por uma interface serial de três fios fácil de usar. Esta particularidade permite que praticamente qualquer microcontrolador possa ser usado, visto que não há qualquer tipo de células de interface específicas.

4.1.1 Especificações

Em termos de especificações de performance do sensor, pode-se sintetizar as suas informações técnicas na seguinte tabela:

Tabela 1 - Especificações MS5541C

		Símbolo	Mínimo	Típico	Máximo	Unidade
Alcance Pressão		p	0		14	bar
Resolução Pressão				1.2		mbar
Alcance Temperatura		T	-40		85	°C
Resolução Temperatura			0.005	0.01	0.015	°C
Tensão de Alimentação		V_{DD}	2.2	3.0	3.6	V
Corrente	Média	I_{avg}		4		uA
	Conversão	I_{sc}		1		mA
	Standby	I_{ss}			0.1	uA
Sinal Clock Externo		MCLK	30.000	32.768	35.000	kHz
Duty Cycle MCLK			40/60	50/50	60/40	%
Clock Interface Serial		SCLK			500	kHz
Resolução ADC				16		bits
Tempo de Resposta				35		ms

Dentro destas especificações verifica-se que o sensor em causa cumpre os requisitos necessários para a aplicação onde se pretende ser inserido. Para além demais, este módulo oferece resistência à humidade e à água. O MS5541C possui uma tampa de proteção metálica preenchida com gel de silicone para proteção reforçada contra humidade. As propriedades deste gel garantem o correto funcionamento do sensor mesmo quando em contacto com a água.

Fisicamente o dispositivo é realmente pequeno (6.2mm x 6.4mm), o que o torna mais prático de instalar, e apresenta um esquema de disposição de pinos que o permite ser facilmente acoplado a uma PCB. Na figura seguinte é possível observar o seu *pinout*:

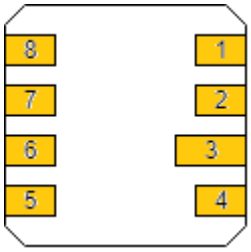


Figura 4.1 - Pinout MS5541C

A descrição dos seus pinos pode ser observada na tabela abaixo:

Tabela 2 - Pinout MS5541C

Número	Nome do pino	Função
1	SCLK	Clock interface série
2	GND	Ground
3	PV	Programação voltagem negativa
4	PEN	Enable programação
5	VDD	Tensão de alimentação
6	MCLK	Master clock (32.768kHz)
7	DIN	Entrada dados
8	DOUT	Saída de dados

Os pinos três e quatro são só usados pelo fabricante para propósitos de calibração e não devem ser conectados.

4.1.2 Interface Série

O MS5541C comunica com microcontroladores ou outros sistemas digitais através de uma interface série síncrona de três fios, muito semelhante a SPI. O sinal de SCLK inicia a comunicação, com a transmissão a acontecer na subida do SCLK quando o microcontrolador quer enviar os dados recebidos pelo sensor. Por outro lado, quando o microcontrolador quiser receber os dados enviados pelo sensor, a transmissão deve ocorrer na descida do SCLK.

Tabela 3 - SCLK na transmissão entre uC e sensor

µC	Sensor	SCLK
Enviar	Receber	
Receber	Enviar	

O SCLK é gerado pelo próprio microcontrolador. O valor digital fornecido pelo MS5541C no pino DOUT, ou é o resultado de uma conversão (valores da pressão ou temperatura não compensados), ou são dados de calibração de *software*. Para além disso, o sinal em DOUT é também usado para indicar o estado da conversão. A seleção do que se pretende ler do sensor é acessível pelo envio duma trama respetiva no pino DIN, que corresponda à instrução desejada.

Todas as comunicações começam com a trama da instrução desejada no pino DIN. O dispositivo não necessita de um *chip select*. Em vez disso, existe uma sequência START (três *bits* a 1) antes da sequência SETUP sendo esta seguida ainda de uma sequência STOP (três *bits* a 0). A sequência de SETUP consiste em quatro *bits* que designam a leitura da pressão, temperatura ou dados de calibração. Por exemplo, no caso de querer ler a pressão a trama enviada deverá ser 1-1-1-1-0-1-0-0-0-0, porém como no protocolo SPI só é possível enviar um *byte* de cada vez, a trama terá de ser divida em dois e adicionados '0' antes e depois da trama para perfazer os dois *bytes*.

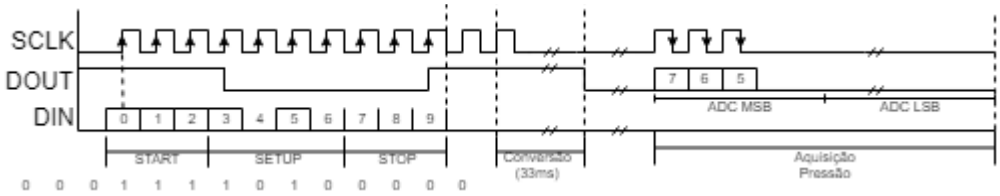


Figura 4.2 - Comunicação entre µC e sensor

Assim sendo, as sequências de todas as tramas possíveis estão representadas na seguinte tabela:

Tabela 4 - Tramas disponíveis na comunicação com o sensor

Trama	Abreviatura	Valor hexadecimal
Conversão do valor de pressão	D1	0x0F40
Conversão do valor de temperatura	D2	0x0F20
Leitura 1ª word calibração	W1	0x1D50
Leitura 2ª word calibração	W2	0x1D60
Leitura 3ª word calibração	W3	0x1D90
Leitura 4ª word calibração	W4	0x1DA0
Reset		0x155540

No caso de ler a pressão, D1, ou ler a temperatura, D2, o módulo reconhece o início de uma conversão do nível lógico baixo para o nível lógico alto no pino DOUT, como mostra na Figura 4.2. Mais dois pulsos de *clock* são necessários em SCLK depois da mudança lógica em DOUT, depois disso SCLK é mantido a 0 até haver nova transição em DOUT, desta vez com a passagem do nível lógico alto para o nível lógico baixo indicando assim o fim de uma conversão. A partir deste momento o microcontrolador pode ler os 16 *bits* de dados correspondentes gerando mais dezassete pulsos de *clock*. É importante sempre ler o resulto da última conversão antes inicializar uma nova.

A sequência de *reset* é especial, uma vez que o módulo reconhece o seu padrão independentemente do estado em que se encontrar. Por essa razão pode ser usado para reiniciar a comunicação entre o microcontrolador e o sensor a qualquer momento, caso a sincronização entre este se tenha perdido. DOUT deve estar a 1 durante esta sequência de vinte e um *bits*. É de resto, aconselhável o envio desta sequência antes do envio de uma sequência que envolva conversão para evitar situações em que o protocolo é desligado permanentemente devido a possíveis interferências elétricas.

Neste tipo de ligação existe ainda mais uma linha à qual é necessário prestar atenção, MCLK. O MCLK ao contrário do SCLK não é necessário para a transmissão de dados, mas sim para o ADC incorporado no módulo. Cabe ao microcontrolador gerar este sinal, que corresponde a um pulso de 32.768kHz com 50% de *duty cycle*. Neste caso em particular foi usado o *timer* 2 do Arduino nano para gerar uma onda quadrada com a frequência e o *duty cycle* pretendidos.

4.1.3 Medição de Pressão e Temperatura

Para ler os valores reais de pressão e temperatura existe um determinado procedimento que se deve seguir, assim como um método de compensação que deverá ser feito por *software*.

Primeiramente as *words* 1 a 4 devem ser lidas do sensor via interface série. Com isto, é possível obter os valores dos coeficientes 1 a 6 (C1 a C6). Estes valores de coeficientes de compensação são extraídos por lógica binária e operações de *shifting*, isto porque, o padrão de bits apresentado nas *words* 1 a 4, não se encontra de forma inteiramente coerente. Por exemplo, W1 contém valores correspondentes aos coeficientes C1 e C2, W2 aos coeficientes C2 e C5, W3 aos coeficientes C3 e C5 e W4 por sua vez diz respeito aos coeficientes C4 e C6. Cada um destes coeficientes de compensação representa uma determina variável que será usada no processo implementado por software para compensação dos valores de pressão e temperatura: (i) C1, sensibilidade da pressão; (ii) C2, offset de pressão; (iii) C3, coeficiente de temperatura da sensibilidade de pressão; (iv) C4, coeficiente de temperatura do offset de pressão; (v) C5, temperatura de referência; (vi) C6, coeficiente de temperatura da temperatura.

Para ler o valor de pressão o microcontrolador tem de ler o valor de 16 bits correspondente à pressão, D1, e o valor de temperatura, D2. Depois disso, a pressão compensada é calculada através de um algoritmo que usa os valores de D1, D2 e C1 a C6. Todo o processo e algoritmo pode ser visualizado no seguinte fluxograma:

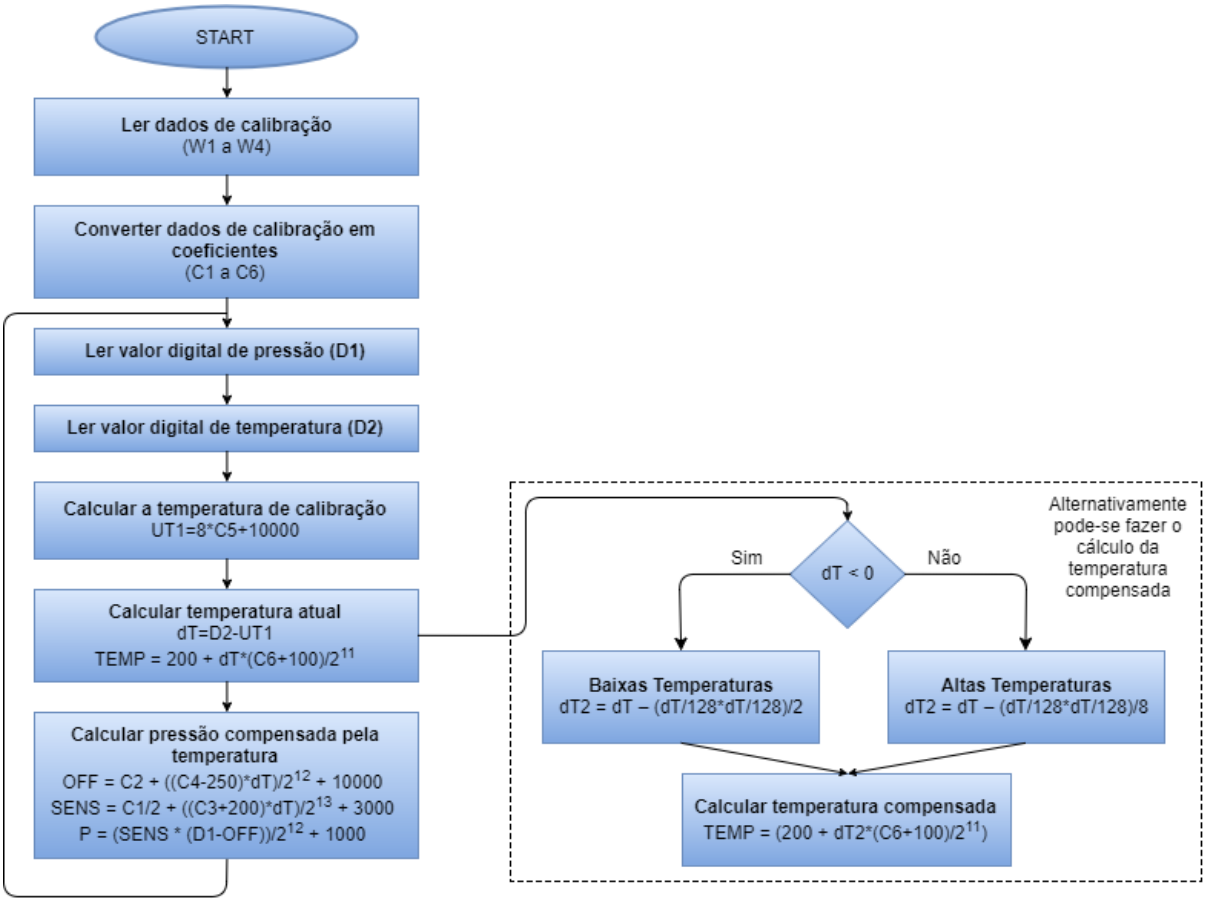


Figura 4.3 – Fluxograma cálculo temperatura e pressão atuais com algoritmo de compensação

Para obter uma alta precisão na medida da temperatura em toda a gama dinâmica do sensor é recomendável compensar a não linearidade da saída do sensor de temperatura. Esta compensação é alcançável corrigindo o valor pressão e temperatura medidos através de um fator de correção de segunda ordem.

4.1.4 Filtragem das Leituras de Pressão

As leituras dos valores de pressão obtidos do sensor apresentam algum ruído ao longo do tempo de medição. Nesse sentido, a aplicação de um filtro com o intuito de suavizar e remover algum desse ruído torna-se útil, devolvendo assim, um sinal mais congruente e fidedigno.

A aplicação de um filtro de média é uma técnica comum para remover ruído em processamento de imagens maioritariamente, mas é aqui utilizada para suavizar o sinal de pressão. Este processo divide primeiramente o sinal de pressão em janelas de dez amostras de pressão e computoriza a média dos valores de pressão dentro de cada janela. Estas médias formam o esqueleto do sinal filtrado, mais suavizado. O sinal de real de pressão é muitas vezes composto por

segmentos lineares, assim, é possível reconstruir os pontos intermédios do sinal suavizado por interpolação linear consecutiva das médias [25]. Para efeitos de ilustração, a Figura 4.4 mostra o sinal de pressão original, onde o eixo do x representa o tempo entre cada amostra, em segundos, e o eixo do y o valor de pressão correspondente medido em bar.

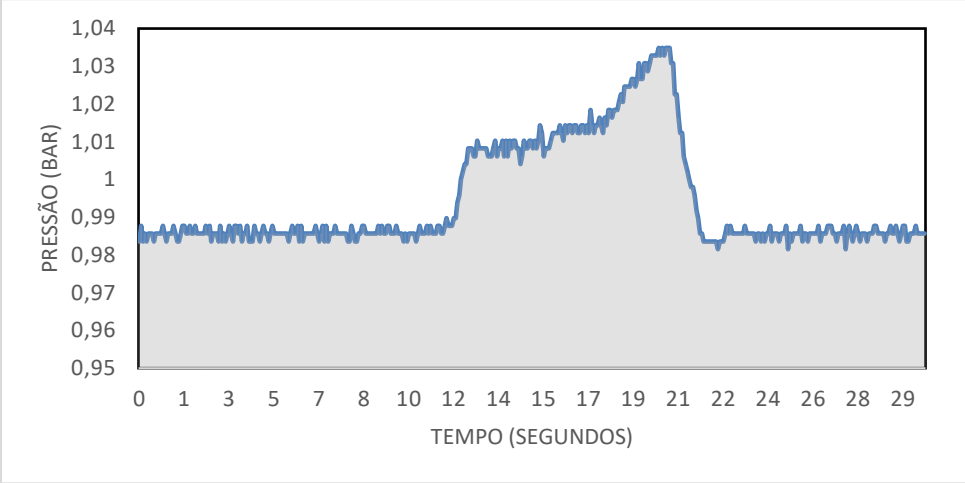


Figura 4.4 - Sinal de Pressão Original

Aplicando o filtro de média produz-se um sinal de pressão menos ruidoso e mais verossímil, como se pode observar na Figura 4.5.

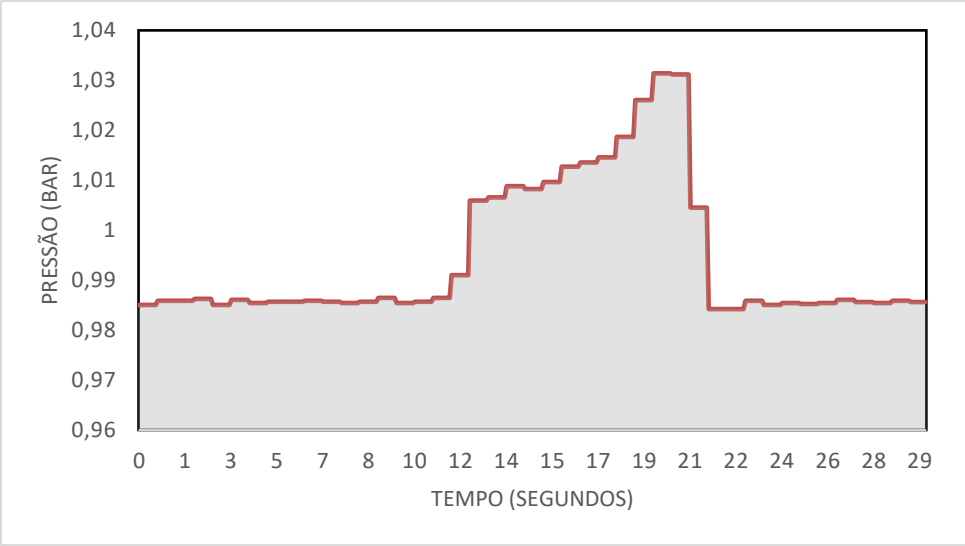


Figura 4.5 - Sinal de Pressão Filtrado

4.2 Armazenamento

Neste projeto um cartão microSD de 8Gb é usado como dispositivo de armazenamento dos dados recebidos pela sonda. O principal requisito nesta parte é permitir operar com ficheiros do sistema FAT32. Para tal, o cartão deverá ser formatado previamente no formato FAT32 antes de se estabelecer a interface entre este e o microcontrolador.

O Secure Digital Memory Card, SDC, nos dias de hoje é um cartão cada vez mais disponível a um preço módico e que oferece uma boa capacidade de armazenamento para qualquer tipo de sistema embebido, para além que ainda suporta SPI o que torna a interface mais simples. O SDC foi desenvolvido como sendo compatível com o Multi Media Card, MMC. Existem também versões de tamanho reduzido, como RS-MMC, miniSD e microSD com a mesma função.

Um SDC divide-se em dois importantes blocos semicondutores: *memory core* e um controlador. O *memory core* é onde os dados são realmente guardados, quanto maior for este bloco maior capacidade de armazenamento o cartão tem. Já o controlador é o responsável pela interpretação de certos comandos standard SD entre o microcontrolador externo e a *memory core*, tal como escrever ou ler dados.

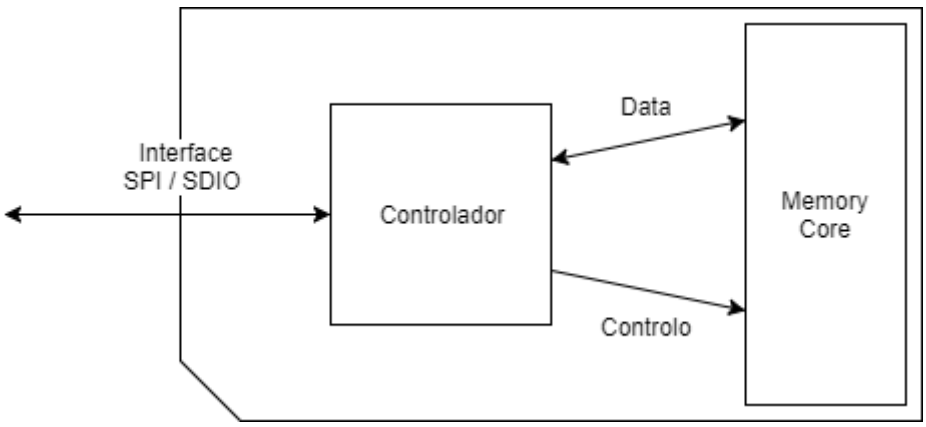


Figura 4.6 – Esquema básico da arquitetura de um SDC

Na figura e tabela abaixo, Figura 4.7 e Tabela 5, é ainda possível observar o pinout de um microSD. Um microSD deverá operar com tensões de pelo menos entre os 2.7V e os 3.6V.

Figura 4.7 - Pinout microSD

Tabela 5 - Pinout microSD

Número	SDIO	SPI
1	DAT2	
2	DAT3	CS
3	CMD	DI
4	VDD	VDD
5	CLK	SCLK
6	VCC	VCC
7	DAT0	DO
8	DAT1	

O cartão pode comunicar com um microcontrolador através de SPI ou SDIO. O SDIO é usado para comunicações a velocidades mais altas, porém neste projeto a interface é feita através de SPI pelo simples facto de o microcontrolador escolhido, Arduino nano, ter incorporado um módulo de *hardware* SPI.

4.2.1 Camadas Funcionais do cartão SD

Resumidamente um cartão SD possui um controlador interno que descodifica os comandos provenientes da interface em série e um memory core que não é mais que uma memória flash sobre

a qual o sistema de ficheiros, FAT32, é implementado. Sobre estes dois blocos internos do cartão SD assentam três camadas essenciais e distintas:

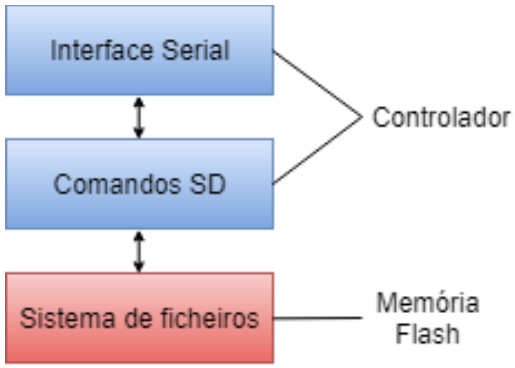


Figura 4.8 - Conceito de camadas SD

O devido acesso a estas três camadas é essencial se o objetivo é operar com ficheiros no formato FAT32.

4.2.2 Interface Série

O protocolo de SPI é usado nesta interface série e é uma alternativa à interface nativa nos MMC/SDC, o SDIO. Com este protocolo os cartões podem facilmente estabelecer uma interface com a maioria dos microcontroladores. Ainda que existam quatro diferentes modos de operação de SPI, este tipo de cartões funciona no modo 0, ou seja, quando a fase e polaridade do *clock* estão configurados no nível lógico 0.

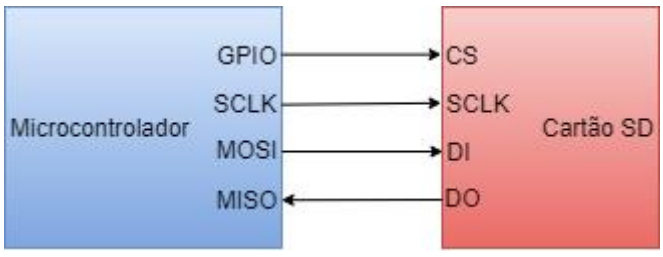


Figura 4.9 - Típica conexão SPI entre uC e cartão SD

Neste tipo de interface o microcontrolador é visto como o *master* e o cartão SD é visto como sendo o *slave*. O microcontrolador é quem inicia a transmissão de dados e que gere o pulso de *clock.* Através do respetivo pino de CS, *chip select,* o microcontrolador pode escolher qual o dispositivo que pretende comunicar. A transmissão de dados propriamente dita é feita com o microcontrolador a enviar dados para o cartão através do pino MOSI, *master out slave in*, e a receber dados do cartão pelo pino MISO, *master in slave out*.

4.2.3 Comandos SD

O cartão SD só aceita alguns comandos padrão. Com esses comandos o microcontrolador pode ler registos do cartão SD bem como escrever e ler na memória deste.

Nesta comunicação série os dados são transmitidos *byte* a *byte.* O comando enviado pelo microcontrolador consiste numa trama que tem um tamanho fixo de 6 *bytes* (figura 15) e o cartão só está pronto para receber um comando se o pino DO, *data out*, está no nível lógico alto. Depois do comando ser enviado é a vez do cartão enviar um comando em resposta e passar o pino DI, data in, para o nível lógico alto, enquanto que sendo o pulso de *clock* gerado pelo microcontrolador, o microcontrolador envia constantemente um 0xFF até ao *byte* recebido ser válido. O pino CS, *chip select*, deve passar de 1 para 0 até a transmissão terminar.

Byte 1			Byte 2~5	Byte 6	
7	6	5 0	31 0	7	0
0	1	Comando	Argumento do Comando	CRC	1

Figura 4.10 - Típico comando SD

O CRC só é usado em dois comandos quando a transmissão é feita em SPI (CMD0 e CMD8). O tipo de CRC usado é o 7-bit CRC que é dado por um polinómio $x^7 + x^3 + 1$. O seu cálculo é obtido da seguinte forma [26]:

$$G(x) = x^7 + x^3 + 1$$

$$M(x) = 1st\ bit * x^{39} + 2nd\ bit * x^{38} + \cdots + last\ bit\ before\ CRC * x^0$$

$$CRC[6 \ldots 0] = Resto\ de\ [(M(x) * x^7)/G(x)]$$

Na tabela seguinte pode-se observar alguns dos comandos usados genericamente para ler, escrever e inicializar o cartão, bem como o seu argumento, o tipo de resposta e a descrição de cada um deles.

Tabela 6 - Comandos SD relevantes

Comando	Argumento	R	Abreviação	Descrição
CMD0	Nenhum	R1	GO_IDLE_STATE	Reset de software
CMD1	Nenhum	R1	SEND_OP_COND	Ativa o modo de inicialização
CMD8	0x000001AA	R7	SEND_IF_COND	Verifica tensão fornecida (SDC v2)
CMD10	Nenhum	R1	SEND_CID	Lê o registo CID (ID do cartão)
CMD12	Nenhum	R1	STOP_TRANSMISSION	Para de ler dados
CMD16	Tamanho	R1b	SET_BLOCKLEN	Altera o tamanho do bloco R/W
CMD17	Endereço	R1	READ_SINGLE_BLOCK	Lê um único bloco
CMD18	Endereço	R1	READ_MULTIPLE_BLOCK	Lê múltiplos blocos
CMD24	Endereço	R1	WRITE_BLOCK	Escreve num único bloco
CMD25	Endereço	R1	WRITE_MULTIPLE_BLOCK	Escreve em múltiplos blocos
CMD55	Nenhum	R1	APP_CMD	Prefixo dos comandos ACMD<n>
CMD58	Nenhum	R3	READ_OCR	Lê registo OCR
ACMD41	0x40000000	R1	APP_SEND_OP_COND	Ativa o modo de inicialização (SDC)

No que diz respeito à resposta do comando, dependendo do tipo do índice de comando enviado existem vários tipos de formatos de resposta que podem ser obtidos tais como: R1, R2, R3 e R7. O tipo de resposta R1, figura abaixo, é o mais comum e consiste num byte, que em caso de sucesso é 0x00 e na ocorrência de algum o bit correspondente a esse tipo de erro é ativo retornando um valor diferente de 0x00.

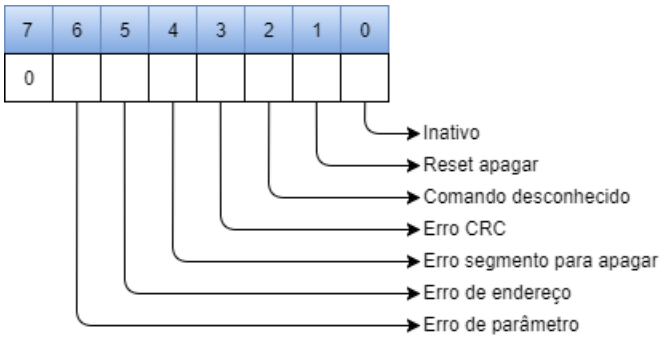


Figura 4.11 - Resposta R1

Por vezes o tipo de resposta obtido é R1b, que apenas significa que demora mais tempo a obter a resposta que R1. As respostas R3 e R7 consistem numa trama composta por R1 seguidos de 32bits e são dedicadas para os comandos CMD58 e CMD8 respetivamente.

Figura 4.12 - Resposta R3

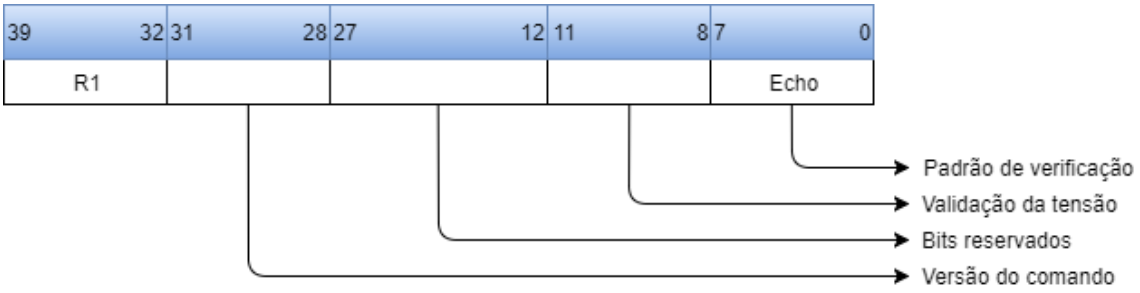


Figura 4.13 - Resposta R7

Inicialmente os cartões MMC/SDC começam a operar no seu modo nativo. Para os inicializar em modo SPI há um certo procedimento que se deve seguir.

Depois de o cartão ser alimentado a pelo menos o mínimo da sua tensão requerida de operação espera-se um milissegundo para esta estabilize. O *clock* do SPI deve ter valor compreendido entre 100kHz a 400kHz. Colocam-se os pinos DI e CS a 1 e aplicam-se setenta e quatro ou mais pulsos de *clock* a SCLK. Nesta fase o cartão está a operar no seu modo nativo e pronto para aceitar o primeiro comando, que corresponde ao CMD0 com o CS a 0. Isso fará com haja um *reset* de *software* e o cartão entre no modo SPI enviando uma resposta R1 de 0x01, *Idle*

State, que significa precisamente que está pronto para correr o processo de inicialização. Uma vez que este comando é enviado e como sendo um comando nativo é imprescindível o campo de CRC na trama de envio, depois de entrar em SPI a funcionalidade de CRC é desativada, exceto para o CMD8.

Enquanto está a decorrer o processo de inicialização envia-se o CMD8, para perceber se o cartão está a operar fora do alcance de tensão suportado, caso o *byte* R1 retorne um comando ilegal significa que o cartão é um SDC v1 ou um MMC, se não é um SDC v2. Não havendo nenhuma incongruência no que diz respeito à alimentação prossegue-se com o envio do comando ACMD41 (SDC) ou CMD1 (MMC) para começar o processo de inicialização. O cartão estará corretamente inicializado quando o *bit* de *Idle State* na resposta R1 passar a 0, ou seja, quando a resposta R1 assumir o valor de 0x00. A partir deste momento o cartão começa a aceitar comandos de escrita e leitura.

No seguinte fluxograma é possível perceber de uma forma mais compreensível todo o processo de inicialização:

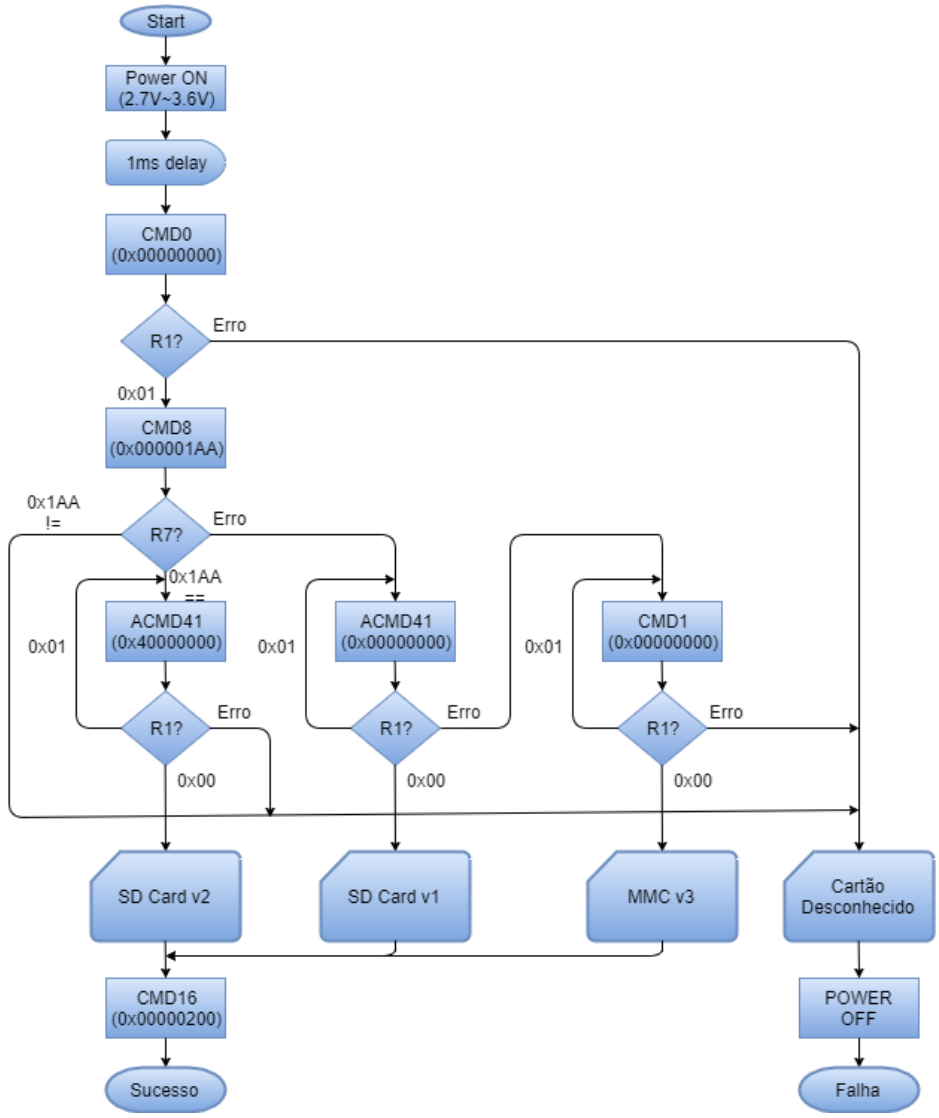


Figura 4.14 - Fluxograma Inicialização em SPI de SDC/MMC

Depois de concluído este processo deve-se ajustar a frequência do SCLK para o valor maior possível com o intuito de maximizar a performance dos comandos de escrita e leitura que interagem com a camada do sistema de ficheiros.

Numa ação que envolta transmissão de dados, um ou mais blocos de dados serão enviados ou recebidos depois de uma resposta a um comando válida. O bloco de dados é transmitido num pacote que consiste num *token*, bloco de dados e CRC.

Os dados podem ser lidos da memória flash do cartão SD usando os seguintes comandos:

- CMD17 – READ_BLOCK – Lê um único bloco de dados
- CMD18 – READ_MULTIPLE_BLOCK – Lê múltiplos blocos de dados

READ_BLOCK:

Para ler um único bloco de dados envia-se o comando CMD17, aonde se descrimina o endereço a partir do qual se pretende ler. Depois de este ser aceite (R1 = 0x00), esse bloco de dados será transmitido ao microcontrolador através do seguinte pacote de dados:

Token CMD17 (1 byte)								Bloco de dados (2 - 512 bytes)	CRC (2 bytes)
1	1	1	1	1	1	1	0	Dados do utilizador	Valor CRC

Figura 4.15 - Pacote de dados de leitura

Se ocorrer algum erro durante o processo de leitura, o microcontrolador, em vez desse pacote, receberá apenas um *byte* que corresponde ao *token* de erro de dados, que indica o tipo de erro que ocorreu:

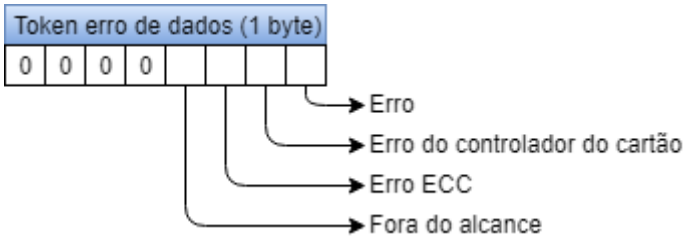


Figura 4.16 - Token erro de dados

READ_MULTIPLE_BLOCK:

Correspondente ao CMD18, este comando apenas difere do anterior pelo facto que, o microcontrolador vai ler blocos consecutivamente a partir do endereço indicado no argumento do comando em vez de apenas um bloco. Para terminar a leitura o CMD12 deve ser transmitido.

Relativamente aos comandos de escrita, os dados podem ser escritos na memória flash do cartão SD usando os seguintes comandos:

- CMD24 – WRITE_BLOCK – Escreve dados num único bloco
- CMD25 – WRITE_MULTIPLE_BLOCK – Escreve dados em múltiplos blocos

WRITE_BLOCK:

Depois de se enviar o CMD24, com o endereço pretendido no argumento, um bloco de dados será escrito na memória flash do cartão. Esse bloco de dados segue num pacote de dados igual ao descrito na Figura 4.15. Para cada bloco de dados escrito, o cartão envia um *byte* de dados para microcontrolador, cujo conteúdo consiste no seguinte:

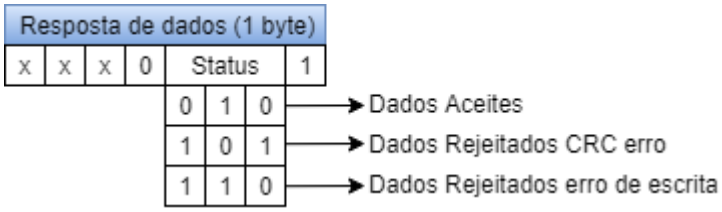


Figura 4.17 - Resposta de dados

WRITE_MULTIPLE_BLOCK:

Este comando escreve múltiplos blocos de dados sequencialmente a partir do endereço indicado. Depois do comando CMD25 ser aceite, o microcontrolador envia um ou mais pacotes de dados para o cartão. O pacote de dados é semelhante ao da Figura 4.15, porém difere no *token*.

Token CMD25 (1 byte)								Bloco de dados (1 - 512 bytes)	CRC (2 bytes)
1	1	1	1	1	1	0	0	Dados do utilizador	Valor CRC

Figura 4.18 - Pacote de dados escrita em múltiplos blocos

O processo de escrita continua até que um novo e distinto *token* seja enviado para parar a transmissão.

4.2.4 Sistema de ficheiros

O sistema de ficheiros usado nos SDC/MMC é o sistema FAT. As especificações dos cartões definem os tipos FAT como: FAT12 para 64MB, FAT16 para 128MB a 2GB, FAT32 para 4GB a 32GB e exFAT para 64GB a 2TB. Neste projeto pretende-se implementar o sistema FAT32, esta secção explica em detalhe a implementação do sistema de ficheiros FAT32 no cartão SD.

FAT32 significa File Allocation Table 32, isto é, tem uma tabela de alocação de ficheiros de 32 bits de tamanho. Todos os dados de um ficheiro são codificados na memória do cartão e o FAT mantém a localização do próximo bloco correspondente à localização do bloco atual.

Sectors:

Um sector é a unidade de bloco de dados mais pequena na memória do cartão para ler ou escrever. Um sector no FAT32 normalmente tem 512 bytes.

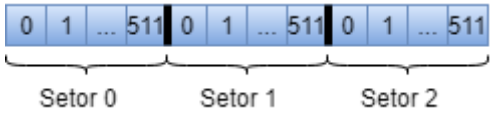


Figura 4.19 - Setores

Clusters:

O agregado de setores sucessivos resulta num cluster. O número de sectors por cluster deve ser uma potência de dois (1, 2, 4, ... 128) dependendo do tamanho total do sistema de ficheiros.

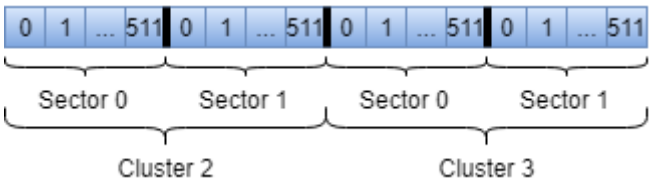


Figura 4.20 – Clusters

Todos os dados de um ficheiro gravado na memória flash do cartão estão distribuídos aleatoriamente nesta em clusters, tal como destaca o seguinte exemplo:

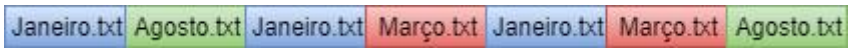


Figura 4.21 - Distribuição de clusters na memória

Cabe à tabela de alocação de ficheiros, FAT, compreender a localização do próximo cluster correspondente ao cluster do ficheiro atual.

FAT32 File System Format:

O ficheiro do sistema FAT32 é guardado dentro da memória do cartão num formato especificamente definido. Consiste numa série de sectors no início e é seguido depois por clusters, que são dedicados para armazenar os dados do utilizador.

Figura 4.22 - Formato FAT32

MBR:

O MBR, Master Boot Record, é o mesmo sector para praticamente todos os sistemas operativos. Fica localizado no primeiro sector da memória e é o primeiro pedaço de código a ser lido. Para além de código de inicialização contém uma tabela de partições, que define as diferentes secções dentro da memória.

Figura 4.23 - Master Boot Record Sector

Os dezasseis *bytes* correspondentes a cada partição detêm informação sobre as respetivas partições, tal como onde fica o primeiro sector da partição. Os dois últimos *bytes* do MBR dizem têm um valor específico (0xAA55), que serve para verificar se o sector que está a ser lido corresponde ao MBR ou não.

Boot Sector:

O boot sector é o primeiro sector dentro de uma partição e alberga todos os detalhes que dizem respeito ao sistema de ficheiros dentro da partição respetiva tal como: *bytes* por sector, sectors por cluster, número de tabelas de alocação de ficheiros, o endereço para o primeiro cluster da partição (normalmente chamado cluster 2) e ainda o endereço do FS Info Sector.

FS Info Sector:

A localização do FS Info Sector é obtida da leitura dos bytes 47 e 48 do Boot Sector. A sua função é armazenar o estado atual dos clusters, mais propriamente o número total de clusters livres no momento e o endereço do próximo cluster que esteja livre.

FS Directory Sector:

Este é o primeiro sector dentro do primeiro cluster do sistema de ficheiros. Este bloco é igualmente chamado de Root Directory. É de resto também, o primeiro sector com dados do utilizador na partição. Aqui são criadas entradas com as informações relativas aos ficheiros do utilizador em conjuntos de 32 *bytes*, pelo que um directory sector suporta 16 estruturas para entradas de ficheiros diferentes. Um cluster que seja usado para guardar estas entradas não serve para guardar os dados propriamente ditos, somente aponta para o cluster onde o conteúdo do ficheiro realmente começa. Na figura seguinte é possível observar o conteúdo de cada uma dessas entradas de 32 *bytes*.

0 10	11 19	20 21	22 25	26 27	28 31
Nome Ficheiro	...	Endereço 1º Cluster HIGH	...	Endereço 1º Cluster LOW	Tamanho Ficheiro

Figura 4.24 - Estrutura de entrada de ficheiro

Tabela de Alocação de Ficheiros (FAT32):

A tabela de alocação de ficheiros de 32 bits, consiste numa sequência de sectors no início da partição onde a cada 32 bits é guardado o endereço corresponde ao próximo bloco de dados, cluster, do ficheiro ao qual se está a aceder. Esta tabela tem tantas mais entradas quantos mais blocos de memória forem alocados na partição. Resumidamente a tabela de alocação de ficheiros serve como elo de ligação entre os vários blocos de dados aleatoriamente distribuídos na memória. Esquematicamente este é o processo que ocorre quando se acede a um ficheiro, por exemplo de texto:

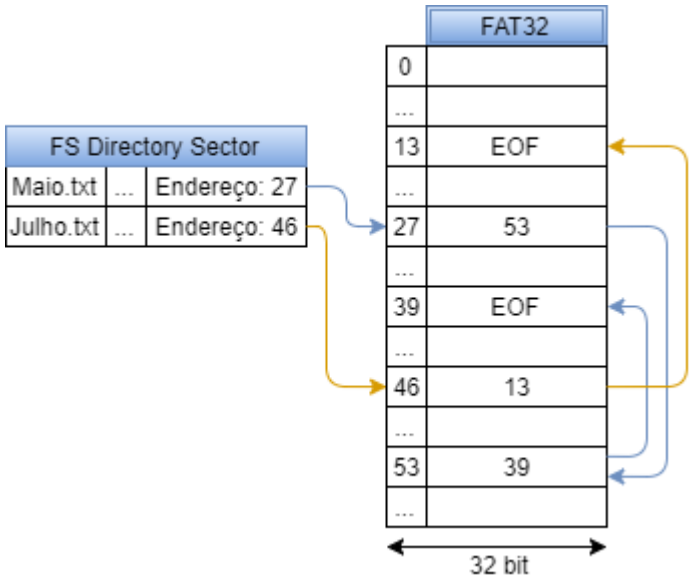


Figura 4.25 - Tabela de Alocação de Ficheiros de 32 bits

4.3 Real Time Clock

A necessidade de saber o tempo a que uma leitura foi feita forçou à integração de um módulo com um relógio de tempo real. Neste caso, optou-se pelo módulo DS3231. O DS3231 é um RTC de baixo custo com um cristal incluído. Para além disso, este dispositivo possui uma entrada para bateria (pilhas CR2032) para manter uma rigorosa precisão no tempo medido mesmo quando a fonte de energia principal do sistema é desligada.

O RTC mantém informação relativa aos segundos, minutos, horas, dias da semana, data, meses e anos. A data ao fim de cada mês é automaticamente ajustada quando se tratam de meses com menos de trinta e um dias e inclui ainda correções para os anos bissextos. Este relógio pode operar tanto no formato de vinte e quatro horas como no formato de doze horas. O dispositivo é ainda dotado de outras funcionalidades como programação de dois alarmes e geração de uma onda quadrada programável. Todos os dados e endereços são transferidos num barramento de I^2C.

4.3.1 Especificações

Para um correto funcionamento do RTC existe um conjunto de condições de operação que devem ser respeitadas. Essas condições bem como algumas das suas características são descritas na tabela abaixo apresentada:

Tabela 7 - Especificações DS3231

		Símbolo	Mínimo	Típico	Máximo	Unidade
Tensão de alimentação		V_{CC}	2.3	3.3	5.5	V
Corrente	Ativa	I_{CCA}			200	µA
	Standby	I_{CCS}			110	µA
Frequência Clock		f_{SCL}	0		100	kHz

Apesar do chip do DS3231 apresentar vários pinos, a placa de breakout que é usada neste projeto sintetiza o acesso a somente quatro pinos, os suficientes para garantir o correto funcionamento da transferência de dados pelo barramento de I^2C. Estes quatros pinos são descritos na seguinte tabela:

Tabela 8 - Pinout DS3231

Número	Pino	Função
1	VCC	Fonte de alimentação primária
2	SCL	Clock interface série
3	SDA	Entrada e saída de dados
4	GND	Ground

4.3.2 Interface I^2C

O DS3231 suporta um barramento de I^2C bidirecional e um protocolo de transmissão de dados. Neste tipo de transmissão o microcontrolador é o transmissor (master) e o módulo RTC é o recetor (*slave*). Cabe ao microcontrolador gerar um sinal de *clock*, SCL, controlar o acesso ao barramento e gerar as condições de START e STOP da comunicação. As comunicações no barramento são feitas através da linha de entrada SCL e da linha de entrada/saída SDA do dispositivo. Dentro das especificações do barramento, SCL não deverá ter uma frequência maior que 100kHz (modo standard), todavia pode atingir frequências mais altas com um máximo de 400kHz se o módulo estiver a operar no modo rápido.

Para se estabelecer uma correta ligação através de I^2C algumas condições de operação devem ser verificadas. Primeiramente as duas linhas do barramento devem permanecer no nível lógico alto. Para iniciar a comunicação propriamente dita, o sinal de SDA deve baixar para o nível lógico baixo enquanto o sinal na linha SCL se mantém alto (condição de START). Contrariamente, quando se pretende parar a comunicação, o sinal de SDA deve transitar do nível lógico baixo para o nível lógico alto e, mais uma vez, o sinal de SCL deve continuar alto (condição de STOP).

Uma transmissão de dados só é completamente válida quando após um executado o START, o sinal da linha de SDA se mantém estável durante o período de tempo em que o *clock* está a 1. Os dados representados na linha SDA devem mudar o seu valor quando o *clock* está a 0. Desta forma, existe um pulso de *clock* por cada bit de dados transferido. Uma transmissão só fica completa quando se verifica uma condição de STOP. O número de *bytes* transmitidos entre a condição de START e a condição de STOP não é limitada e é determinada pelo microcontrolador. Toda a informação é transmitida *byte* a *byte* e o recetor acusa a receção com um nono *bit*.

O módulo RTC quando devidamente endereçado deve reagir com um *acknowledge* após a receção de um *byte*. Para tal, o microcontrolador tem de gerar um pulso de *clock* extra, para esse

bit adicional. O módulo deve baixar o sinal na linha SDA de modo que, quando o pulso de *clock* extra estiver no seu nível lógico alto, a linha SDA esteja estabilizada em 0. O microcontrolador deve sinalizar o fim da transmissão de dados para o módulo não gerando este pulso de *clock* extra e neste caso, o módulo deve deixar a linha SDA no estado alto para habilitar o microcontrolador da capacidade de criar a condição de STOP.

Posto isto, o DS3231 pode operar em dois modos: o modo de receção e o modo de transmissão.

No modo de receção, a transferência de dados deverá ser intercalada entre uma condição de START e uma condição de STOP. O reconhecimento do endereço do módulo é feito por hardware depois da receção do byte que contém o endereço e o bit de direção. Este byte é o primeiro byte a ser transmitido do microcontrolador e contém nele sete bits correspondentes ao endereço do DS3231, 1101000, seguido de um bit de direção que indica se vai ser feita uma leitura ou uma escrita no módulo, pelo que neste caso este bit terá o valor de 0 (escrita). Depois de desencriptar o primeiro byte relativo ao endereço o módulo responde com um *acknowledge* e está preparado para receber o próximo byte que diz respeito a um endereço, que define o apontador para a posição na memória RAM onde a data começará a ser gravada, mais uma vez, o módulo reage com um *acknowledge*. O microcontrolador pode agora enviar os bytes de dados que bem entender e só terminará quando decidir enviar a condição de STOP.

No modo de transmissão, o primeiro *byte* é tratado como se estivesse no modo de receção, no entanto, o *bit* de direção aqui tem o valor de 1 e indica que a transferência de dados acontece no sentido reverso. Recebido o primeiro *byte* o DS3231 reage como habitualmente com um *acknowledge*. Posto isto, o DS3231 começa a transmitir dados partindo da posição onde o apontador para a posição da memória RAM foi definido. O DS3231 deve receber um *not acknowledge* para terminar a transmissão.

4.3.3 Tempo Real

As informações relativas ao tempo e ao calendário atuais são obtidas pela leitura dos bytes dos registos apropriados. Contudo, estes registos devem ser inicializados ou definidos pelo microcontrolador. Ao escrever nos endereços 0x00 a 0x06 do módulo RTC pode-se definir uma data e hora, onde o primeiro registo, 0x00, é referente ao registo dos segundos e assim sucessivamente até ao registo 0x06 que diz respeito ao registo do ano. Uma vez definidos estes registos pode-se lê-los sempre que necessário. O conteúdo de cada um destes registos está no formato BCD,

codificação binária decimal. O DS3231 pode ser executado no modo de 12h ou no modo de 24h, bastando para isso alternar o bit 6 do registo das horas.

Quando se está a escrever ou a ler nos registos de tempo e data, buffers secundários são usados para prevenir erros relativos à atualização dos registos internos. Para ler os registos de tempo e de data, estes buffers secundários são sincronizados com os registos internos após a ocorrência de uma condição de START e quando o apontador para a posição de memória aponta para 0x00 (segundos). A informação é lida a partir destes registos secundários, enquanto o *clock* continua o seu ciclo. Isto faz com que, seja eliminada a necessidade de reler os registos principais quando acontece uma atualização dos seus valores durante uma leitura.

Para questões de otimização e simplicidade de desenvolvimento de código declara-se uma estrutura para melhor acesso a cada um dos registos do DS3231 (segundos, minutos, ..., ano). Para inicializar o módulo RTC, primeiramente habilita-se a comunicação I^2C aonde se define a frequência de SCL. A partir daqui o módulo está pronto para ser escrito ou lido e cada um destes métodos processa-se de maneira semelhante: estabelece-se a condição de START, envia-se para módulo o seu endereço de identificação no barramento, define-se o apontador dos registos para o registo dos segundos, depois depende do modo de operação, se for escrita envia-se *bytes* de dados, se for leitura espera-se a receção dos dados e envia-se ou não o bit *acknowledge*, e por fim, estabelece-se a condição de STOP. Na figura seguinte está ilustrada uma situação em que se descreve o processo de escrita e leitura do DS3231.

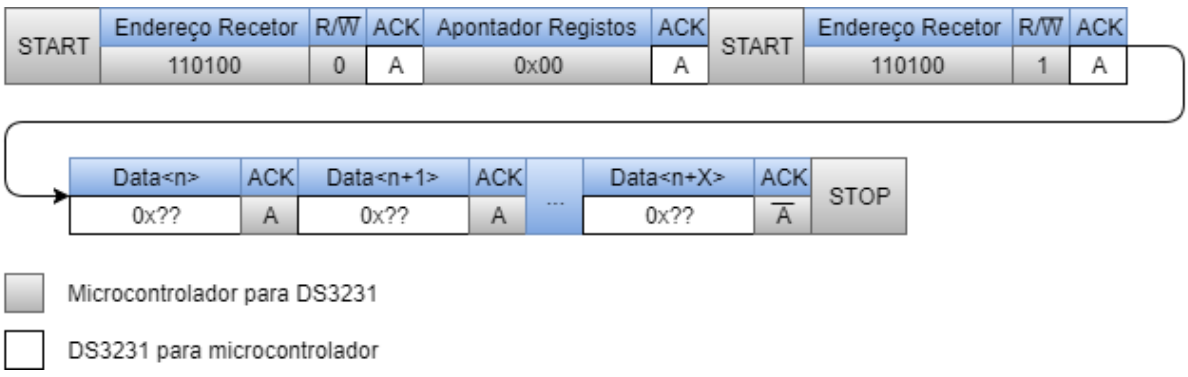


Figura 4.26 - Processo de escrita e leitura do DS3231

De notar que a informação escrita e lida nos registos do DS3231 deve estar no formato BCD, isto é, por exemplo, a sequência de dados: 0x12, 0x32, 0x48, 0x01, 0x15, 0x08, 0x17 representa 12 horas, 32 minutos e 48 segundos de segunda-feira dia 15 de agosto de 2017.

4.4 Comunicação GSM/GPRS

Todas as comunicações realizadas pela sonda, quer seja a transmissão de dados para o exterior ou quer seja a receção de dados para dentro da sonda, acontecem graças à integração do módulo de comunicação GSM/GPRS SIM808. O módulo SIM808 é um módulo três em um que incorpora as funcionalidades de GSM, GPS e *Bluetooth*. O controlo e o acesso a esta placa de comunicação são feitos através de comandos AT via UART.

O GPRS é um aprimoramento da infraestrutura da rede GSM existente e permite o transporte de dados por pacotes, também chamada de comutação de pacotes, permitindo assim, uma taxa mais elevada de transferência de dados em comparação com as tecnologias anteriores que usavam comutação por circuito. O GPRS introduz um número de novos elementos funcionais que suportam o princípio de transmissão *end-to-end* de pacotes de dados baseados em IPs.

4.4.1 Especificações

O SIM808 suporta uma rede *quad-band* de GSM/GPRS nas frequências 850, 900, 1800, 1900MHz. O termo *quad-band* exprime precisamente a capacidade de o dispositivo operar em quatros frequências diferentes. No contexto de aplicações móveis esta funcionalidade oferece ao utilizador uma amplificada capacidade de *roaming*, o que significa que o módulo funcionará em qualquer ponto do globo onde o serviço de GSM esteja disponível.

No que diz respeito ao GPRS, os dispositivos dotados de GPRS são classificados segundo categorias de performance, pois nem todos os dispositivos móveis são designados para desempenhar o mesmo tipo de serviço. Como tal, são separados em três classes de acordo com as suas capacidades de se conectar aos serviços GSM e GPRS: *class A*, *class B* e *class C* [27]. O módulo em causa insere-se na *class B*, o que significa que ele pode ser conectado a serviço GPRS e a um serviço de GSM (voz, SMS), mas não pode usar os dois em simultâneo. Durante o uso do serviço GSM, ou seja, quando está a decorrer uma chamada de voz ou está a ser processada uma SMS, o serviço de GPRS é suspendido e só é retomado quando o serviço GSM terminar.

Os dispositivos GPRS são ainda organizados sobre uma outra categoria que os distingue pela taxa de transmissão de dados que eles podem suportar, as chamadas *multislot classes* [28]. As *multislot classes* determinam a velocidade de transferência de dados nas direções de *uplink* e *downlink* e correspondem a um valor entre 1 e 45 que a rede usa para alocar canais de rádio nessas

direções de ligação. O SIM808 insere-se na classe 12 o que lhe faculta uma conectividade no máximo de 85.6kbps quer em velocidade de *download* quer em velocidade de *upload*.

Uma outra das apelativas funcionalidades do módulo SIM808 é o suporte de RTC. Todavia, como a interface entre microcontrolador e o módulo de comunicação é feito por comandos AT via UART, o processo de leitura do valor atual do tempo torna-se um pouco lento e desagradável quando por exemplo se pretende armazenar dados em períodos de amostragens relativamente pequenos. Por essa razão, optou-se por um módulo à parte de RTC para esse efeito, o DS3231 que pode ser observado com mais detalhe no capítulo 4.3.

Para além do mecanismo de GSM/GPRS, o SIM808 é ainda dotado de um mecanismo de GPS, que permite munir a sonda com um sistema de *tracking* da sua posição num determinado espaço e suporta o protocolo da NMEA (vide capítulo 4.7).

A nível mais de interface, o SIM808 sustenta um suporte para cartões SIM standards. O cartão SIM permite a autenticação do cliente na rede, para que o aparelho possa procurar a rede GSM na qual está registado e dessa forma mediante o plafond definido com a operadora permitir as várias transferências de dados que possam ocorrer. O SIM808 possui também três conectores de antena uFL, para a antena GSM, GPS e *Bluetooth*, respetivamente. Possui ainda um botão que permite a ligação da placa manualmente e LEDs que indicam o módulo está ligado ou não e o seu estado da conexão à rede. Para conectar a bateria existe um conector XM-2.0mm, no qual se podem ligar baterias com tensões entre os 3.4V e os 4.4V, o que se torna conveniente para ligar baterias Li-Po de 3.7V.

Relativamente a algumas características elétricas do módulo pode-se observar a seguinte tabela:

Tabela 9 - Especificações SIM808

	Símbolo	Mínimo	Típico	Máximo	Unidade
Tensão de Entrada	V_{BAT}	3.4	3.7	4.4	V
Corrente de pico	I_{peak}	0		2	A
Corrente média	I_{avg}	2		500	mA

Os valores de corrente consumida dependem do modo de operação do módulo (no modo *sleep* gastará relativamente menos).

Por último, mas não menos importante segue-se uma tabela relativa ao mapeamento dos pinos do SIM808 e as suas respetivas funções.

Tabela 10 - Pinout SIM808

Pino	Função
VBAT	Pino ligado diretamente à bateria Li-Po e que pode ser usado para fazer drive para o microcontrolador
GND	Ground
VIO	Referência do nível lógico (5V, 3.3V)
DTR	Pino usado para acordar o módulo do modo *sleep*
PWR	Permite ligar e desligar o módulo por *software*
RI	Identifica a existência de mensagens ou chamadas novas
TX	Saída de dados (UART)
RX	Entrada de dados (UART)
RST	Permite fazer reset ao módulo

4.4.2 Ativação do Módulo

Antes de realizar alguma ação com o módulo é necessário ligá-lo fisicamente. O SIM808 dispõe de várias maneiras de habilitar essa ligação, desde logo pressionando o botão de *power* existente na placa de comunicação, porém este método é muito pouco ortodoxo e torna-se impraticável quando se pretende que o sistema funcione de uma forma totalmente autónoma. Assim sendo, o módulo é ligado alternativamente por *software.* Essa ligação é realizada usando o pino "PWR" presente do módulo, ao qual uma transição no nível lógico de um sinal proveniente do microcontrolador por mais de um segundo é suficiente para que habilitar a ligação do módulo e consequentemente a ligação da sua porta série, possibilitando assim as mais diversas configurações e operações que o módulo de comunicação suporta através de comandos AT.

Uma vez que o módulo está alimentado, resta desbloquear o cartão SIM para finalizar o que diz respeito ao capitulo da ativação do módulo. O modo de desbloqueio do cartão SIM não é mais que a inserção do número de identificação pessoal, PIN. Como qualquer cartão SIM facultado por uma operadora, ele vem com o seu respetivo PIN e PUK. Aqui, tal como acontece à semelhança dos telemóveis aquando da inserção do PIN, uma sequência de três tentativas erradas resulta no bloqueio do cartão e o seu desbloqueamento só fica possível com a introdução do PUK, o que requer

especial atenção quando o modo de inserção do PIN deixa de ser feito de forma manual e passa a ser feito de forma automática (*software*). A entrada código PIN é então feita recorrendo a um comando AT, no qual é acoplado uma string de quatro caracteres correspondente a esse mesmo código, *"AT+CPIN="****"\r"*. O comando retornará uma mensagem correspondente ao sucesso ou insucesso da ação.

4.4.3 Registo na Rede GPRS

Para usar a rede GPRS, o terminal deve registar-se na rede GPRS, fazer o chamado *attach*. Durante um *attach*, a rede verifica se o utilizador é autorizado e este pode ser combinado com GSM, pois trata-se de um módulo de comunicação *class* B. As atividades relacionadas a um terminal GPRS são caracterizadas por três estados diferentes: *idle*, *standby* e *ready*. No estado *idle*, o módulo assenta sobre a rede GSM. A estação móvel pode receber dados por comutação de circuito e estabelecer atualizações sobre a sua localização. Nesta fase, o módulo de comunicação comporta-se como qualquer outro dispositivo GSM e ainda não está anexado ao serviço GPRS.

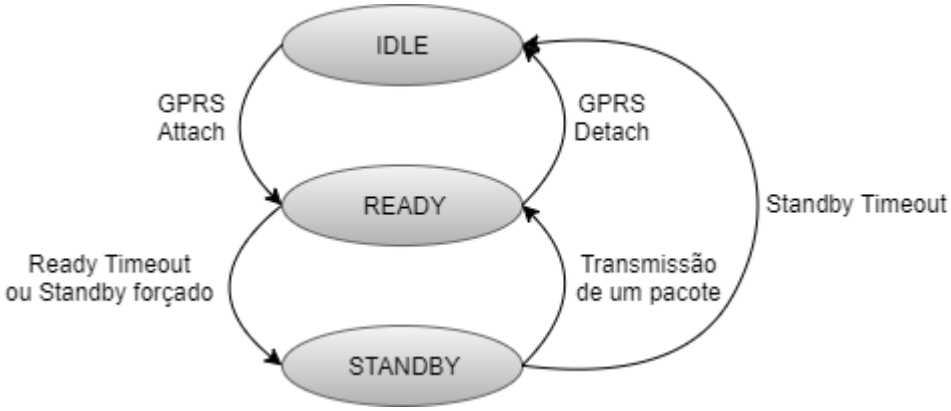


Figura 4.27 - Modelo de Estados de uma estação móvel GPRS [29]

No estado *ready,* a rede está ciente da existência da estação móvel e é possível transmitir e receber pacotes PDP bem como ativar ou desativar o contexto PDP. O módulo fará a transição para o estado *standby* caso não ocorra transmissão de dados durante um período de tempo estabelecido. O processo de *detach* trará o módulo para o estado de *idle* novamente.

Por fim, no estado *standby* a estação móvel continua *attached* à rede e só transitará de novo para o estado *ready* quando houver a transmissão ou receção de um pacote de dados por parte da estação móvel.

A descrição do procedimento de *attach* pode ser ilustrada e compreendida no seguinte esquema presente na Figura 4.28.

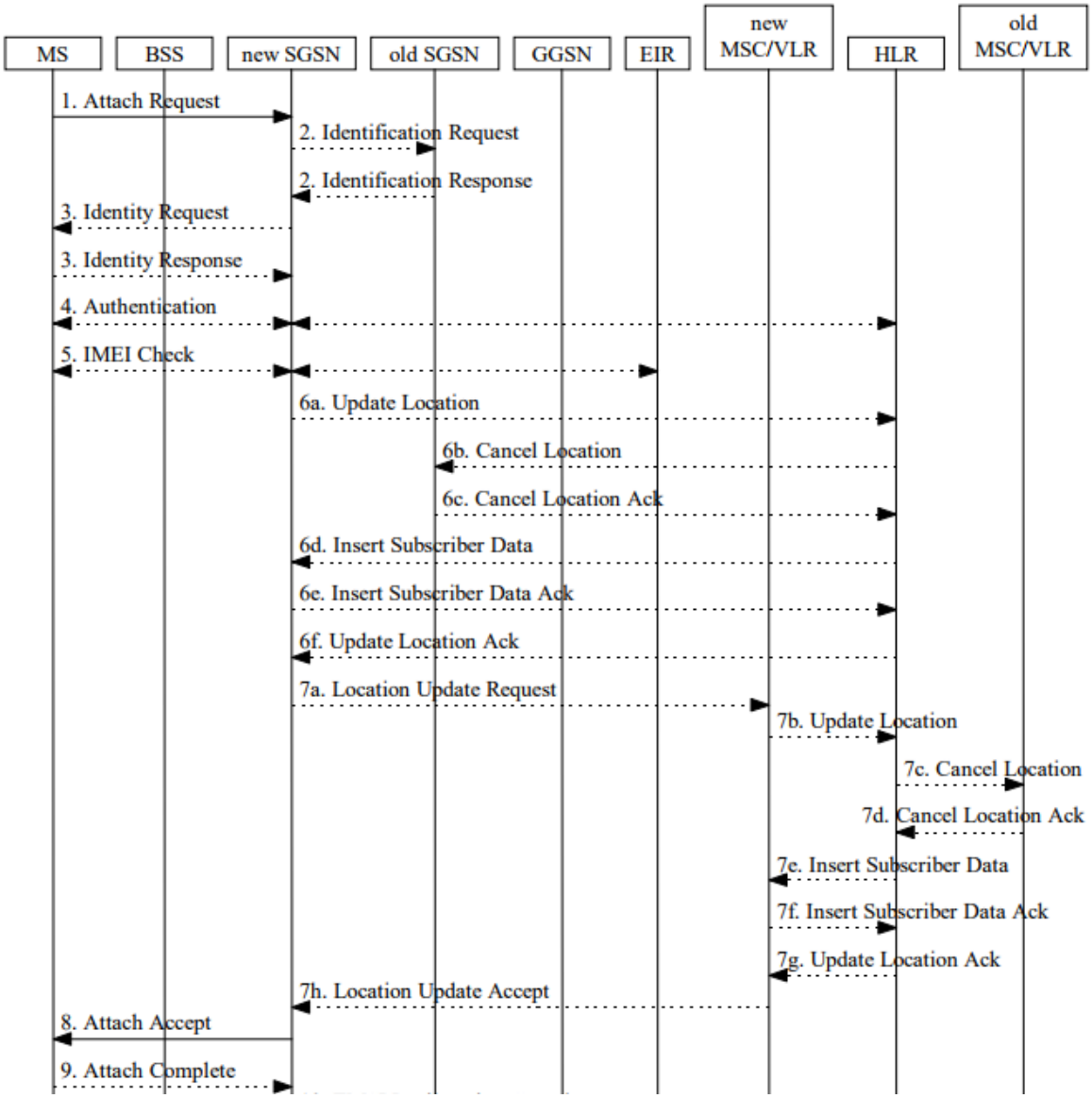


Figura 4.28 - Processo de "attach" GPRS [30]

Em jeito de síntese, este comportamento de *attach* funciona de uma maneira sucinta, da seguinte forma: a estação móvel, neste caso o SIM808, envia uma mensagem de *attach* para o SGSN[1] que contém parâmetros tais como: IMSI (*International mobile subscriber identity*) e o tipo de *attach.* Se o SGSN tiver mudado desde a último *attach* feito pelo módulo e se este for desconhecido

[1] SGSN é o nó de suporte do serviço GPRS que medeia o acesso a recurso da rede em nome dos subscritores móveis. É responsável pela entrega de pacotes de dados recebidos e transmitidos da estação móvel. O SGSN armazena informação relativa à localização e aos perfis de utilizador registados.

ao SGSN, o SGSN requer alguns passos adicionais para obter o IMSI, desde logo, contactará o SGSN antigo para obter o IMSI, caso o SGSN antigo não conceda o IMSI, o SGSN novo requererá à estação móvel que o conceda. Uma vez que o IMSI é conhecido o SGSN inicia o processo de autenticação enviando uma autenticação para o módulo. Se o SGSN mudou desde o último *attach* à rede, o novo SGSN começa a atualizar a localização recorrendo ao HLR[2]. O HLR envia um pedido de cancelamento de localização para o SGSN antigo e envia os dados da subscrição para o SGSN novo inserindo uma mensagem de dados do subscritor. Depois disso, finalmente, o SGSN envia uma aprovação de *attach* para a estação móvel e esta responde com um *acknowlegde*.

4.4.4 Configuração do *Bearer*

Na especificação de um padrão de telecomunicações como é o caso da tecnologia GSM, o primeiro passo é, naturalmente, a configuração dos serviços oferecidos pelo sistema. Na terminologia GSM, os serviços de telecomunicação podem ser categorizados nos serviços *bearer* (Existem outros tipos de serviços como é o caso dos teleservicos e os serviços suplementares). *Bearer* no sentido etimológico da palavra significa alguém ou algo que transmite para outro lugar algum tipo de mensagem, assim sendo o serviço *bearer* é um serviço que permite a transmissão de sinais de informação entre interfaces de rede. Estes serviços oferecem ao utilizador a capacidade necessária para transmitir sinais apropriados entre certos pontos de acesso. Para habilitar a funcionalidade deste tipo de serviços existem alguns aspetos que devem ser respeitados, tais como a definição do tipo de conexão e um APN único.

O tipo de conexão a configurar é imediato, pretende-se uma conexão GPRS, e dessa forma, aquando da configuração do *bearer* dever-se-á passar o parâmetro "GPRS" no comando AT referente ao estabelecimento do tipo de ligação.

Quanto à definição do APN é um pouco mais complexa. Uma conexão GPRS é estabelecida por referência ao nome do ponto de acesso, APN. O APN trata-se, normalmente, do *gateway* entre uma rede móvel (GSM, GPRS, 3G, 4G, ...) e a rede de internet pública. Um sistema de rede móvel como este deve ser configurado com um APN para que, a operadora responsável por fazer a ligação à internet interprete esse identificador e determina o tipo de conexão de rede que deve ser estabelecida. Mais especificamente, o APN identifica a rede de pacotes de dados, PDN, através da

[2] HLR significa *Home Location Register* e é a base de dados principal da informação permanente de um subscritor para uma rede móvel.

qual o usuário de redes móveis quer comunicar. Para além disso, o APN pode ser usado para definir o tipo de serviço para a qual o PDN é usado (WAP, MMS, ...). Todo o trabalho pesado é feito pela operadora, no entanto é necessário ter a certeza que as configurações atribuídas estão corretas, para entrar na rede correta, da maneira correta. Dependendo do modo como a rede da operadora usada está estruturada, diferentes configurações têm de ser usadas obrigatoriamente.

A estrutura de um APN consiste em duas partes distintas: um identificador de rede e um identificador da operadora [31], tal como se pode observa na figura seguinte.

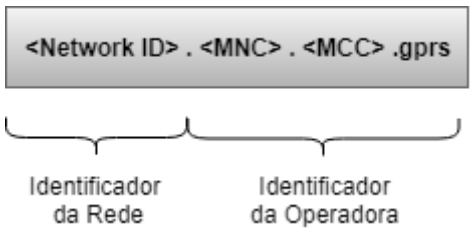


Figura 4.29 - Estrutura de um APN

O identificador de rede esclarece a que rede externa deverá o GGNS ser conectado e é obrigatória a sua inclusão aquando da definição de um APN. Por sua vez, o identificador da operadora consiste em três campos separados por um ponto que definem em que PLMN o GGNS está situado, mas este não é de carácter obrigatório [32], onde <MNC> diz respeito ao código da rede móvel e <MCC> ao código do país [33]. Estes dois códigos juntos identificam uma rede de operadora móvel única. Especificamente, nesta dissertação foi usado o APN "net2.vodafone.pt", este APN contém somente um identificador de rede com três campos e nestes três campos o identificador de rede é tratado como um nome de domínio internet. Neste caso, "vodafone.pt" é um nome de um domínio internet registado e como tal, tendo um domínio registado como sendo parte do identificador de rede, a singularidade pode ser assegurada ao trocar APNs entre redes ou mesmo dentro de uma rede.

Existem dois modos de aplicação IP para o SIM808: APPTCP e SAPBR. APPTCP e SAPBR podem funcionar simultaneamente. Quando está a operar no modo APPTCP, estabelece uma comunicação assente em TCP ou UDP. Quando está a operar no modo SAPBR, contém uma aplicação FTP ou HTTP.

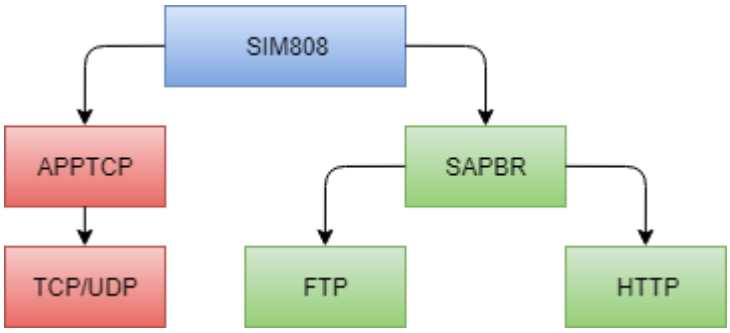


Figura 4.30 - Estrutura das aplicações IP no SIM808

Este esquema supracitado serve para entender a estrutura do comando AT que permite a configuração do *bearer*. Assim sendo a nível de código a configuração é feita da seguinte forma: (i) Tipo de conexão: *"AT+SAPBR = 3, 1, "Contype", "GPRS"\r"*; (ii) Definição do APN: *"AT+SAPBR= 3, 1, "APN", "net2.vodafone.pt"\r"*; (iii) *Bearer* aberto: *"AT+SAPBR =1,1 \r"*.

4.4.5 Aplicação HTTP

O módulo SIM808 contém uma pilha TCP/IP embebida que é orientada através de comandos AT e que permite aplicações aceder facilmente aos serviços de internet FTP ou HTTP. Uma grande vantagem desta solução é que elimina a necessidade de o fabricante da aplicação implementar a sua própria pilha TCP/IP, minimizando assim o custo e o tempo imprescindíveis para integrar conectividade de internet numa aplicação nova ou já existente.

O HTTP é um protocolo de pedido e resposta, o que significa que um determinado cliente envia uma solicitação a pedir uma determinada página ou ficheiro e o servidor envia de volta essa página ou ficheiro, os chamados HTTP *requests* [34].

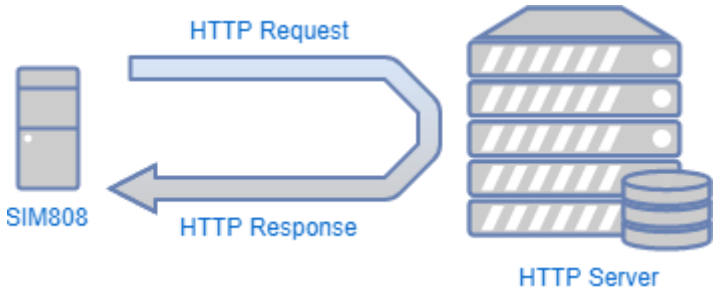


Figura 4.31 - HTTP Requests entre SIM808 e o servidor

Um cliente HTTP inicia um *request* estabelecendo uma ligação TCP para uma porta particular de um servidor. O servido ao captar essa mensagem de *request* retorna uma linha de estado,

"HTTP/1.1 200 OK", e uma mensagem particular própria. O corpo dessa mensagem é normalmente o recurso solicitado.

O conteúdo da mensagem de *request,* segundo Fielding et al [34], é composta ordenadamente por uma linha inicial (*request line*), linhas de cabeçalho (*request header*), uma linha separadora branca obrigatória e o corpo da mensagem caso seja preciso (ver Figura 4.32 e Figura 4.33). A *request line* é organizada em três partes separadas por um espaço: o método, a identificação do URI[3] e a versão do HTTP utilizador (normalmente HTTP/1.1, pois é a versão atual).

O protocolo HTTP define vários métodos a inserir na *request line* que indicam a ação a ser realizada no recurso especificado, ou seja, os métodos definem o que o servidor deve fazer com o URL recebido. Dentro desses métodos destaque para os métodos GET e POST.

O método GET é provavelmente o método de HTTP *request* mais comum. Este método solicita uma representação do recurso especificado. Quando implementado corretamente, um método GET deve ser idempotente, isto é, múltiplos *requests* feitos ao mesmo recurso devem ter o mesmo resultado que um *request* apenas e não deve alterar nenhum valor no servidor em causa. Por outras palavras, o método GET deve ter um efeito de apenas leitura para recuperar dados e não para os alterar.

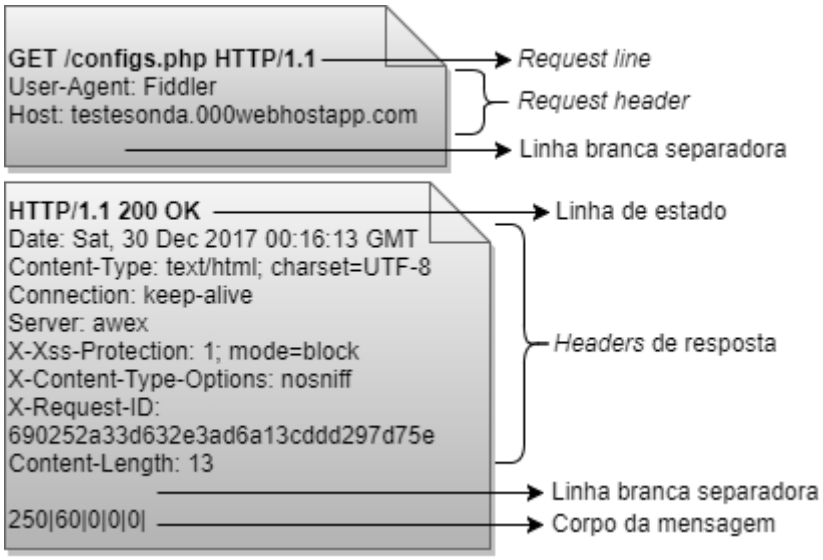


Figura 4.32 - HTTP Request e respetiva resposta - GET

[3] O *Uniform Resource Identifier,* URI, indica qual o recurso que a requisição pede. No protocolo HTTP, o URI é o URL, *Uniform Resource Locator,* composto pela identificação do protocolo, pelo endereço do servidor e pelo documento solicitado.

Um *request* bem definido resulta numa resposta com sucesso que tem um corpo da mensagem correspondente ao recurso solicitado.

O método POST envia dados para o servidor para serem processados no recurso especificado. Os dados a serem tratados são incluídos no corpo do comando. Este método é usado quando é necessário enviar dados para um servidor para serem processados, recorrendo normalmente a um PHP *script* que está presente no URL introduzido. O método POST requer sempre que as informações submetidas sejam incluídas no corpo da mensagem e formatadas como uma *query string*. O formato da *query string* deverá listar as variáveis em estudo pela ordem em que elas aparecem no *script,* em que o nome da variável e o seu respetivo valor é separado por um "=" e o conjunto da variável nome/valor separado das demais por um e comercial, "&" [35], para melhor elucidação deste formato pode-se observar a Figura 4.33. Para além da compreensão da *query string* este método requer ainda alguns parâmetros adicionais no seu *header* tais como a *Content-Length,* que especifica o tamanho da mensagem, e o *Content-Type.* O *Content-Type* diz respeito ao formato da mensagem, na verdade, é ele que define o formato da *query string* e neste caso é usado neste projeto o formato "application/x-www-form-urlenconded".

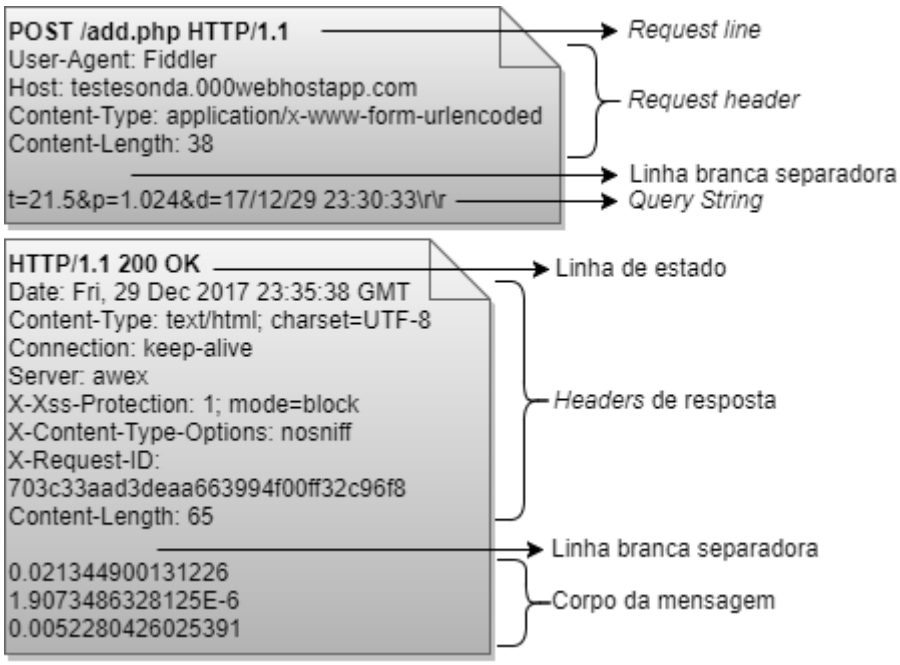


Figura 4.33 - HTTP Request e respetiva resposta - POST

É por causa destas particularidades que o método POST oferece maior segurança no que diz respeito à transferência de dados quando comparado ao método GET, uma vez que no método GET os dados são anexados ao URL, ficando assim expostos e visíveis ao usuário.

Em ambos os métodos a linha inicial, linha de estado, de uma resposta HTTP, através de um código de retorno e de uma frase explicativa, proporciona ao cliente informação relativa ao *request* executado. O código de retorno é formado por três dígitos, onde o primeiro desses três dígitos indica o tipo de estado a que pertence [34]. Existem cinco tipos de estados diferenciados pelo sentido de *feedback* que têm.

Tabela 11 - Códigos de retorno HTTP

Código de retorno	***Feedback***	**Exemplo**
1xx	Informação	100 Continue
2xx	Sucesso	200 OK
3xx	Redireccionamento	301 Not Modified
4xx	Erro no cliente	404 Not Found
5xx	Erro no servidor	502 Bad Gateway

Toda a aplicação HTTP que acontece no microcontrolador pode ser traduzida no seguinte fluxograma, que descreve a sequência operacional dessa aplicação baseada em comandos AT.

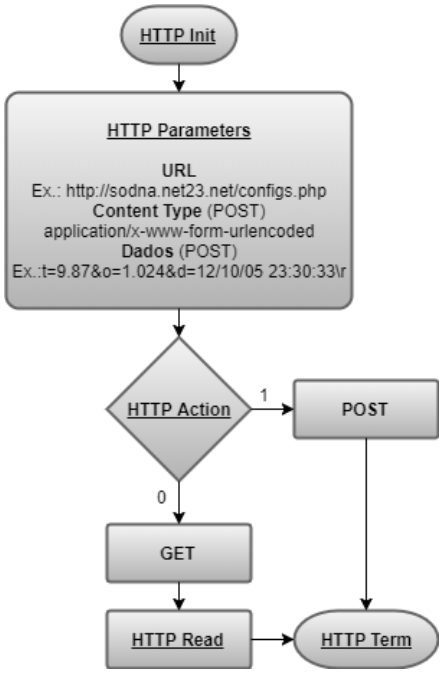


Figura 4.34 - Fluxograma Aplicação HTTP

4.5 Aplicação *back-end*

O conceito de *back-end* está muitas vezes associado como sendo tudo o que diz respeito ao servidor, isto é, faz referência a tudo aquilo que o utilizador não consegue visualizar, como a base de dados ou os *scripts* alocados no servidor. Na aplicação *back-end* é gerenciado e estruturado o conteúdo enviado pela sonda, para que este possa depois ser apresentado mais tarde na aplicação de *front-end* e providenciar uma apreciação ao utilizador.

4.5.1 Base de Dados

Uma base de dados não é mais que um repositório de dados em que estes são armazenados de uma forma organizada. Neste caso em concreto, a base de dados estará constituída por dados relativos à leitura do sensor (pressão e temperatura), configurações de frequências de amostragens, e outros dados relativos ao estado da sonda (nível de bateria, qualidade do sinal GSM, posição de latitude e longitude), para que posteriormente possa ser feita a troca dessas informações por um utilizador, que remotamente através da aplicação *front-end*, pretenda comunicar com a sonda ou vice-versa.

O gerenciamento da base de dados é feito pelo MySQL. O MySQL é precisamente um dos sistemas de gerenciamento de base de dados mais populares da atualidade e usa a linguagem SQL como interface. Graças ao MySQL toda os dados são armazenados sobre a forma de um conjunto de informação estruturada. Para além de ser rápido, robusto, de múltiplos utilizadores e de múltiplos processos, é um sistema onde a informação pode ser guardada em várias tabelas, em vez de estar numa única partição, facilitando assim mais rapidez de acesso e flexibilidade [36]. Para além de tudo isso, o MySQL tem a grande vantagem de ser grátis e *open-source.*

Para adicionar, aceder ou processar os dados é preciso um sistema de controlo e gestão que funcione independentemente ou como parte de outras aplicações (por exemplo, através de uma aplicação PHP).

O MySQL pode ainda usar da potencialidade de uma ferramenta de administração chamada PhpMyAdmin, que tal como o nome indica, foi totalmente desenvolvido em PHP para que possa funcionar independentemente da plataforma em que se encontra o servidor *web*. Através desta ferramenta é possível controlar todas as bases de dados existentes como se existisse uma só, criar e eliminar bases de dados, bem como tabelas e alterar os seus campos. É possível ainda exportar informação de uma base de dados em vários formatos (SQL, XML, ...) para, por exemplo, criar uma cópia de segurança.

4.5.2 Interação PHP *Scripts* com a Base de Dados

A linguagem PHP foi especialmente desenhada para habilitar o desenvolvimento de páginas web. A grande vantagem do PHP é oferecer uma excelente conectividade com várias bases de dados, entre elas MySQL. A combinação PHP e MySQL é de resto das combinações mais comuns no desenvolvimento web.

Muitas das funcionalidades da linguagem PHP são concedidas graças a extensões, *plug-ins* que são carregados quando se inicia o servidor web. O acesso à base de dados é feito por meio dessas extensões, no caso, como se pretende aceder a uma base de dados MySQL, usa-se a extensão MySQLi.

O primeiro passo é conectar à base de dados. Esse passo é possível usando a função MySQLi mysqli_connect(), que necessita no mínimo de argumentos como o nome do *host,* nome do utilizador da base de dados e a respetiva palavra-passe para poder estabelecer uma conexão. De modo a selecionar qual a base de dados sobre a qual as *queries* vão correr é necessário um quarto parâmetro, transmitido através da função mysqli_select_db(), com um argumento correspondente

ao respetivo nome dessa base de dados. Depois disso, faz-se uma verificação se a conexão foi bem sucedida. Existem várias razões pelas quais uma conexão pode ser mal sucedida, as mais comuns são a introdução de credenciais erradas ou carga excessiva de uso da base de dados que pode levar a que esta refuse novas conexões.

Um passo mais avançado é fazer transformar esta configuração de conexão à base de dados numa função, para que mais tarde se possa aceder à base de dados em qualquer *script* PHP de uma forma mais simples sem ter que experienciar o processo todo de novo. Importante lembrar que, a menos que seja declarada a função mysqli_close(), a base de dados permanecerá acessível durante o tempo que esse *script* demore a correr.

Uma vez estabelecida a conexão à base de dados é possível começar a aplicar-lhe *queries.* A maneira mais simples e prática de fazer isso é usando a função mysqli_query().

```
01.  //Conexão Base de Dados
02.  $link=Connection();
03.
04.  //Query Base de Dados
05.  $query = "SELECT * FROM Conf"
06.  $result = mysqli_query($link,$query);
```

Figura 4.35 - Query Base de dados

De uma maneira bastante intuitiva, o valor de "$result" será *false* se a *query* falhar, caso contrário terá o valor correspondente à query em causa. No caso de sucesso, para *queries* como "SELECT", será retornado um objeto mysqli_result, e para outras *queries,* tal como "INSERT", será retornado *true*.

Para *queries* do tipo "SELECT" (e alguns outros tipos semelhantes) será retornado então um objeto mysqli_result tal como mencionado em cima. Para obter dados relativos às linhas de uma tabela da base de dados é necessário um passo adicional para captá-los. Aqui entra a função mysqli_fetch_array(), que obtém as linhas de um objeto mysqli_result como um vetor associado, onde os índice desse vetor correspondem aos nomes das colunas da base de dados. Desta forma, ter-se-á um vetor com as linhas retornadas da base de dados, todavia caso não seja retornada nenhuma linha, este vetor estará naturalmente vazio.

De modo a concluir a interação PHP com a base de dados deve-se usar a função mysqli_close() no final do *script* em causa. Esta é uma função extremamente importante pois fecha a conexão à base de dados do servidor. O *script* estará sempre a correr caso ela não seja chamada

e a abertura de várias conexões à base de dados MySQL sem as dar por terminadas pode resultar no comportamento defeituoso dessa conta.

4.5.3 Manipulação da Aplicação HTTP em PHP

Dentro do subcapítulo "4.4.5 – Aplicação HTTP" já observamos de que modo o microcontrolador da sonda emite e recebe dados usando o protocolo da camada de aplicação HTTP, mais detalhadamente debruçou-se sobre o funcionamento dos HTTP *requests* e dos seus métodos POST e GET. Contudo, o modo como essas solicitações são interpretadas na aplicação *back-end* não foi especificado. Neste subcapítulo faz-se essa análise.

Na ocorrência de um método POST, tal como vimos anteriormente, os dados chegaram sobre a forma de uma *query string*, igual à que é possível observar Figura 4.33. Esta informação deverá ser alvo de um processo de *parsing* para transformá-la numa *query* SQL e introduzi-la numa das tabelas geradas na base de dados. O PHP providencia uma variável super global chamada "$_POST", para aceder a toda a informação enviada por um HTTP *request* com o método POST. Quando se usa uma *query string* do tipo "application/x-www-form-urlencoded" a variável "$_POST" passa a ser um vetor associado onde os seus índices dizem respeito ao nome da variável e os seus respetivos conteúdos ao valor dela. Por exemplo, usando mais uma vez o exemplo da Figura 4.33, "$_POST" será: $_POST['t']=21.5, $_POST['p']=1.024 e assim sucessivamente.

Posto isto, o processo de transformação numa *query* SQL torna-se mais fácil. Primeiramente faz-se uma verificação se a variável pretendida foi de facto transmitida, recorrendo à funcionalidade de isset(), e caso os seus valores sejam diferentes de *null,* durante um ciclo iterado pelo número de vezes que uma *query string* foi enviada vai-se preenchendo a respetiva tabela da base de dados, através da *query* SQL "INSERT".

```php
if (isset($_POST["t"])) $t = $_POST["t"];
else $t = null;

if (isset($_POST["p"])) $p = $_POST["p"];
else $p = null;

if (isset($_POST["d"])) $d = $_POST["d"];
else $d = null;

for($i=0; $i<count($t); $i++)
{
    $query = "INSERT INTO `Variables` (`Temperatura`, `Pressao`, `Data`)
        VALUES ('".$t[$i]."','".$p[$i]."', '".$d[$i]."')";

    mysqli_query($link,$query);
}
```

Figura 4.36 – Exemplo tratamento de um HTTP POST

Quando a solicitação feita ao servidor se trata de HTTP *request* com um método GET, o que se pretende é ler informação, ou extrair informação do servidor, pelo que a parte de *parsing* deixa de ser necessária, até porque, no corpo de um *request* do tipo GET não vem nenhuma informação para ser tratada. O que acontece num ficheiro PHP aquando da ocorrência de uma solicitação deste tipo, em vez de o *script* fazer o tratamento da *query string* e a sua inserção numa tabela, é extraída a informação pretendida de uma tabela, usando a *query* SQL "SELECT", e impressa no *script*. Para imprimir essa informação faz-se um *echo* do resultado da *query* "SELECT".

4.5.4 Estrutura e organização dos ficheiros PHP

Nesta fase é importante perceber de que forma os *scripts* PHP estão organizados e que função desempenham na aplicação de *back-end*. Para tal pode-se observar a figura seguinte, que contém uma ilustração de todos ficheiros alocados no servidor.

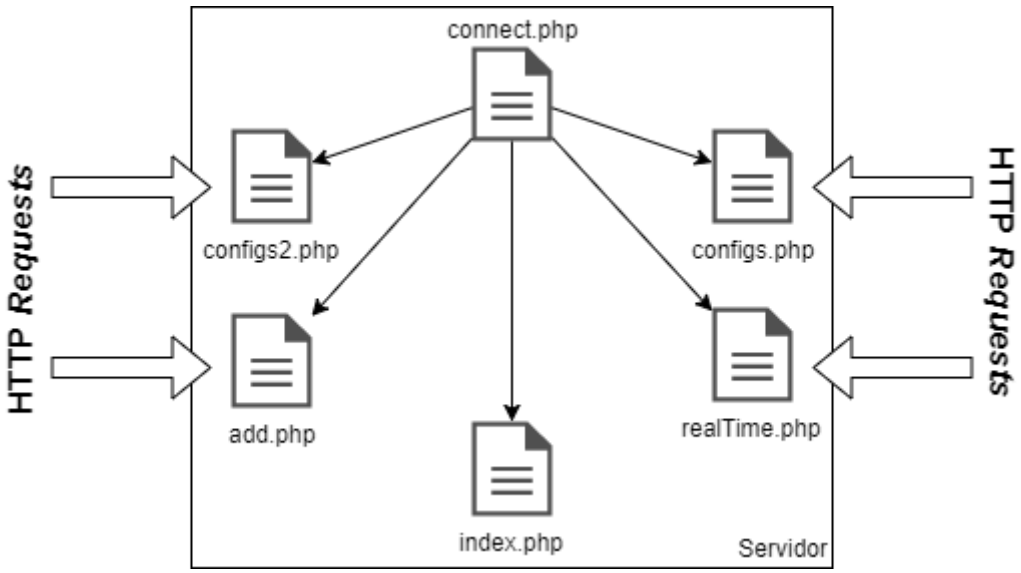


Figura 4.37 - Organização ficheiros PHP

Todos os ficheiros PHP incluem o *script* "connect.php". Este *script* não é mais que uma função na qual é estabelecida uma conexão à base de dados, onde estão as credenciais de acesso, daí ela ser incluída no início de cada dos restantes *scripts.*

Em "add.php" é onde são recebidos os dados recolhidos pela sonda (temperatura, pressão e a respetiva data e hora da amostra), e na qual se faz a inserção desses mesmos dados na base de dados, e nele só são associados *requests* do tipo POST.

Tudo o que diz respeito a configurações e informações relativas ao estado da sonda é lidado pelos ficheiros "configs.php" e "configs2.php". A estes ficheiros já estão associados *requests* do tipo POST e do tipo GET, ora para armazenar informação no servidor, ora para extraí-la do servidor e a disponibilizar a quem a solicitar. O "configs.php" é o ficheiro no qual o utilizador através da aplicação *front-end* envia os dados e relativos à frequência de amostragem (temperatura e pressão) e à frequência de envio dos dados para o servidor (POST), e no qual a sonda lê então esses valores definidos pelo utilizador (GET). O "configs2.php" por sua vez, é o ficheiro no qual a sonda envia dados relativos ao seu estado (nível de bateria, nível do sinal GSM, coordenadas latitude e longitude e espaço livre no cartão SD) para o servidor (POST), e no qual o utilizador vê na aplicação *front-end* os valores correspondentes a esses estados (GET).

O "realTime.php" é usado para a sonda obter a data e hora atuais (GET), e dessa forma acertar os valores do módulo DS3231, para que desta forma a sonda esteja munida da capacidade de conceder o tempo exato a que uma amostra foi recolhida. Basicamente a sonda ao fazer um GET

deste recurso, vai provocar ao mesmo tempo que haja um comando SQL de "UPDATE" que altere a entrada relativa ao tempo atual na tabela da base de dados. Este *script* só necessita de correr uma vez, ao início, durante todo o processo de recolha de dados da campanha da sonda.

Por fim existe ainda o ficheiro "index.php". Este é o ficheiro local que automaticamente abre se o visitante não especificar no URL a página que quer aceder. No caso em particular, apresenta uma tabela com todos os dados recolhidos pela sonda até então. Para tal, o *script* faz uso das propriedades da linguagem SQL para apresentar os dados de forma ordenada pela o tempo de recolha e das propriedades da linguagem HTML, para construir a tabela e apresentar os dados de forma mais visível.

4.6 Aplicação *front-end*

Uma aplicação *front-end* é uma aplicação na qual o utilizador interage diretamente. O *front-end* é responsável por recolher os dados de entrada do utilizador, que possam ser submetidos de várias formas, e processá-los para adequá-los a uma especificação em que o *back-end* os possa empregar. Além de receber entradas geradas pelo utilizador, o *front-end* é também usado para expor informação útil para esse mesmo utilizador, pelo que uma boa aplicação *front-end* requer um especial cuidado com a experiência do usuário.

Esta é uma das partes que não sendo fulcral, se mostra uma ferramenta útil, tanto para o desenvolvimento do projeto em si, como para o uso da sonda de uma forma mais simples, em que há toda uma camada de abstração entre o utilizador e o que acontece na sonda.

4.6.1 Aplicação C#

A aplicação *front-end* consiste numa interface gráfica de utilizador feita em C#. A linguagem C# tem todas as características de qualquer preponderante linguagem moderna. Em C#, uma das maneiras de criar uma interface gráfica é usando *Windows Forms,* que são bibliotecas GUI da *framework* .NET da Microsoft*.* Os elementos das *Windows Forms* são componentes reutilizáveis, que encapsulam a funcionalidade da interface, e que são familiares a programadores de GUI.

O elemento básico da maior parte das GUI desenvolvidas sobre a forma de *Windows Forms* é tal como o próprio nome sugere a janela [37]. Essencialmente tudo o que é visível na aplicação *front-end* (botões, ícones, caixas de texto, ...) são janelas. Por causa disto, muitas destas janelas e controlos presentes nas *Windows Forms* têm as mesmas propriedades. Por exemplo, todos eles têm uma propriedade chamada "Text", todavia a maneira como eles fazem uso dessa propriedade depende da sua especificação no caso.

Desenvolver uma aplicação deste género torna-se inteligível assim que alguns conceitos básicos são compreendidos. Esta secção abrange alguns desses conceitos e dá um ponto de partida sobre a qual o desenvolvimento da aplicação se iniciou.

Um dos pontos iniciais cruciais é declarar no início do projeto da aplicação alguns dos *namespaces* usados. O mais comum é o *namespace* "System", que carrega todas as bibliotecas básicas de uma classe. Por exemplo, o *namespace* "System.Windows.Forms" suporta todas as definições de janelas e controlos das *Windows Forms.* Outro exemplo é o "System.Drawing" que dá acesso à funcionalidade gráfica do sistema operativo.

Em seguida, vem a declaração da classe e com ela a instanciação de objetos necessários ao programa. Um desses membros da classe que é praticamente obrigatório é um objeto do tipo "Container", que pertence ao *namespace* "System.ComponentModel". Este objeto não participa na representa gráfica do programa, contudo faz imenso trabalho em segundo plano para suportar *timers, multithreading* e até mesmo para fazer limpeza do programa quando ele fecha [37].

Quando uma classe é criada, o seu construtor tem de estar inerente. No caso em concreto, o construtor chama um método chamado InitializeComponents(). Esta função cria e instancia todas as janelas e controlos para gerar a interface gráfica do programa. Basicamente, nesta função define-se os valores iniciais das propriedades dos elementos da aplicação. Sem querer entrar muito em detalhe, cada elemento possui variadas propriedades que indicam o modo como se apresentam na GUI. Por exemplo, existe a propriedade de "Location", que especifica a posição X e Y na janela principal onde o elemento irá aparecer, a propriedade de "AutoSize", que aceita um valor booleano para ver se o elemento deve ou não ajustar o seu tamanho mediante o seu conteúdo, a propriedade de "TabIndex", que quando existem múltiplos elementos numa janela se torna conveniente premindo a tecla "Tab" permite alternar entre os vários elementos consoante o índice que lhes for atribuído, a propriedade de "Anchor", que diz onde determinado elemento reside, isto é, se está unido a outro elemento ou não, entre muitas outras propriedades.

Debruçando agora sobre a função Main(), esta é a função que simplesmente põe o programa a correr. Ela chama uma função estática chamada Run(), que pertence à classe "Application" e deve aceitar como parâmetro uma nova instância da classe que gera a *Window Form*. Quando a aplicação *front-end* está a correr, o utilizador pode ver a janela que se encontra na figura seguinte.

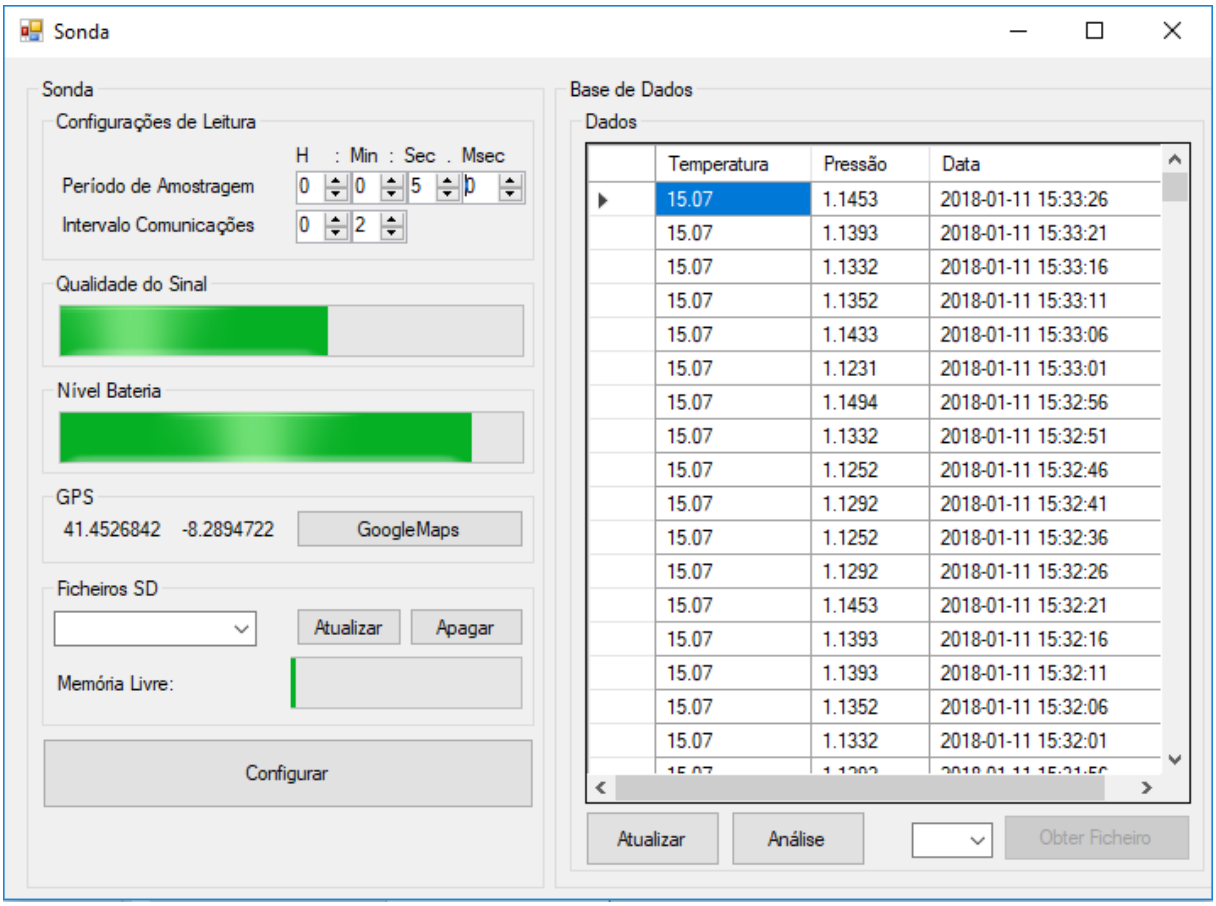


Figura 4.38 - Aplicação front-end

Olhando para a Figura 4.38, rapidamente se repara na presença de vários tipos de controlos (botões, barras de progresso, etc.). Cada um destes elementos tem variadas características especificas e serve um propósito único. Não vale a pena entrar em detalhe sobre a função que cada elemento desempenha, pela simples razão de serem autoexplicativos. Contudo é evidente pela imagem que existe uma abundante gama de componentes gráficos e tipos de controladores nas aplicações *Windows Forms* e que estes podem ser combinados para criar uma aplicação relativamente sofisticada.

4.6.2 Eventos

Um evento é uma mensagem enviada por um objeto para sinalizar a ocorrência de uma ação. Esta ação pode ser gerada pelo utilizador clicando por exemplo num botão, ou pode ser causada por outra lógica do programa como mudar o valor de uma propriedade. O que gera o evento é chamado de *event sender.* Tipicamente um evento faz parte do *event sender,* por exemplo, o evento

"Click" é um membro da classe "Button". Para definir um evento, deve ser especificado aquando da sua declaração o *type* de *delegate* usado para gerar o evento.

Um *delegate* identifica a função que vai lidar com o evento. No contexto dos eventos, um *delegate* age como um intermediário entre a fonte do evento e o método que cuida desse evento [38]. A .NET *framework,* providencia um tipo de *delegate* que suporta vários cenários chamado "EventHandler".

Para responder a um evento, define-se um método para lidar com o evento no recetor do evento. Este método deve subscrever ao tipo de *delegate* previamente definido. Aqui são realizadas ações que são solicitadas aquando a geração de um evento, como recolher dados de entrada do utilizador após clicar num botão.

Neste caso em particular, são gerados essencialmente três tipos de eventos: cliques nos botões, valores alterados nas caixas numéricas das configurações de leitura e interrupção do *timer.* Nos cliques dos botões as ações realizadas pelos métodos respetivos são evidentes baseando no texto que cada um desses botões dispõe, por exemplo, clicando no botão "Atualizar" faz uma renovação dos dados extraídos até então. Nos valores das caixas numéricas das configurações de leitura estão a ser gerados eventos a cada alteração dos seus valores para impedir que seja introduzido um valor abaixo ou acima da capacidade de leitura da sonda. Já na interrupção do timer está a ser gerado um evento que quando passa um período de tempo estipulado (no caso cinco minutos) há uma atualização das variáveis de estado da sonda.

4.6.3 Classe WebClient

Na aplicação *front-end* existe a primordialidade de permitir que o utilizador seja capaz de alterar parâmetros de configuração na sonda (frequências de amostragem, ...) e que paralelamente possa monitorizar algumas variáveis de estado relativas ao comportamento da sonda (nível de bateria, ...). Já se observou que, a sonda faz esta troca de informação com o servidor, pelo que o objetivo nesta fase é que a própria aplicação gráfica seja capaz de igualmente fazer troca de dados do utilizador com o servidor, a fim de que, da próxima vez que a sonda comunicar com o servidor haja informação renovada para lidar com ela.

No fundo, mesmo na interface gráfica existe a necessidade de aplicar os métodos GET e POST, relativos aos HTTP *requests,* para ler conteúdo de um determinado URL, ou para postar e atualizar informação na base de dados. Uma das soluções para resolver esta necessidade é usar a classe WebClient da .NET *framework.* Esta classe é apenas um revestimento sobre outra classe da

.NET, a classe HttpWebRequest. Isto significa que, a classe WebClient tem propriedades e métodos definidos que habilitam a interação direta com servidores usando HTTP.

Para inserir novos dados no servidor é usado o método UploadValues(). Este método envia uma variável do tipo NameValueCollection[4] para o servidor. Se o *Content-Type* não for especificado, ele assume o seu valor *default* "application/x-www-form-urlencoded". Para um recurso HTTP, que é o caso, esta função usa o método POST. O resto do processo é depois lidado com a aplicação *back-end.* Contudo este processo só é desencadeado quando é premido o botão "Configurar" da aplicação. Aqui o programa fará uma leitura dos valores representados na caixa de grupo "Configurações de leitura", que serão então os valores carregados para o servidor.

Para ler as variáveis de estado da sonda, a aplicação precisa de ler a *string* exposta no recurso ".../configs2.php". Para conseguir extrair essa informação, o método DownloadString() é usado. Basicamente, este método vai descarregar o todo conteúdo da página *web* como sendo uma *string*, todavia esta pagina *web* foi propositadamente desenhada para que todo o seu conteúdo fosse uma única e simples *string* para facilitar o processo de *parsing* na aplicação *front-end.* Este processo de leitura do recurso enunciado é feita automaticamente e periodicamente recorrendo a um *timer* que vai executar essa acção de cinco em cinco minutos e irá alterar os valores das barras de progresso e as caixas de texto relativas às coordenadas geográficas com os valores correspondentes.

4.6.4 Processos

De uma maneira simplificada iniciar um processo é correr outra aplicação sobre a aplicação desenvolvida. Há muitos tipos de situações em que esta funcionalidade é útil, neste caso, a utilidade surge com o interesse de querer mostrar, mediante as coordenadas geográficas relativas à posição da sonda no momento, uma página *web* que mostra no GoogleMaps a localização da sonda. Isto pressupõe que, neste caso a aplicação que se pretende correr por cima da aplicação *front-end* é um navegador de internet.

A .NET *framework* oferece uma solução eficiente para resolver este problema. Existe dentro do *namespace* "System.Diagnostics" uma classe chamada "Process", que desempenha um papel importante no arranque de novos processos. Esta classe permite arrancar e parar um processo do sistema local.

---

[4] NameValueCollection representa uma associação de chaves (*string*) e de respetivos valores (*string*), que podem ser acedidos pela chave ou por um índice.

É possível abrir uma página *web* através do método Start() da classe, bastando para isso passar o URL na função como sendo um argumento [39]. Aqui, com o auxílio de operadores de concatenação de *strings* gera-se um URL distinto para cada posição. Ao invocar o método Start() com o URL pretendido, automaticamente será aberto o navegador de internet padrão que apresentará a respetiva página *web,* tal como se pode observar de resto na Figura 4.43.

4.6.5 HTML *Parser*

Na aplicação *front-end* é também possível examinar os valores contidos na base de dados relativos às medições realizadas. O método de apresentação destes valores passa pela decomposição do documento HTML criado no recurso "índex.php".

Para tal, é usada uma biblioteca elaborada com código da .NET *framework,* chamada "HTMLAgilityPack". Este é um analisador HTML ágil que gera um DOM de leitura/escrita [40]. Ao gerar um DOM, gera-se uma representação e uma interação com objetos em documentos HTML [41]. Os nós de cada documento são organizados numa topologia hierárquica e podem ser endereçados e manipulados pelo uso de métodos aplicados sobre os objetos.

Esta ferramenta mostrou-se vantajosa na medida em que, permite analisar um documento HTML, selecionar os seus nós e manipular o seu conteúdo [42]. O processo de aquisição do ficheiro HTML até que este tenha o seu conteúdo processado e apresentado na aplicação para o utilizador poder ver inicia-se com um método que permite adquirir o ficheiro HTML a partir de um recurso da internet. Este método é invocado de forma assíncrona, pois por vezes pode haver uma grande quantidade de dados para descarregar e dessa forma evita-se que a interface gráfica fique inoperacional durante esse período de tempo. Uma vez obtido o documento, seleciona-se uma lista nós pretendida, bastando para isso, usar um método que através do qual é passado sobre forma de parâmetro a expressão que compreende o XPath. Basicamente, existirão três listas de nós, uma para temperatura, outra para pressão e outra para a data e hora da amostra. Estas listas terão ainda de ser manipuladas no sentido de lhes remover codificação HTML inerente (ex.: " ") e dessa maneira ficar apenas com o valor da variável representada sobre a forma de *string.* Posto isso, é só adicionar à tabela previamente criada do tipo "DataTable" os respetivos valores para que possam ser expostos ao utilizador.

4.6.6 Ferramenta de Análise de Dados

Com o propósito de expressar visualmente os dados extraídos foi incorporada na interface gráfica uma ferramenta de análise de dados, que através do esboço de gráficos facilita a compreensão desses mesmos dados, permitindo assim uma análise mais rápida e objetiva.

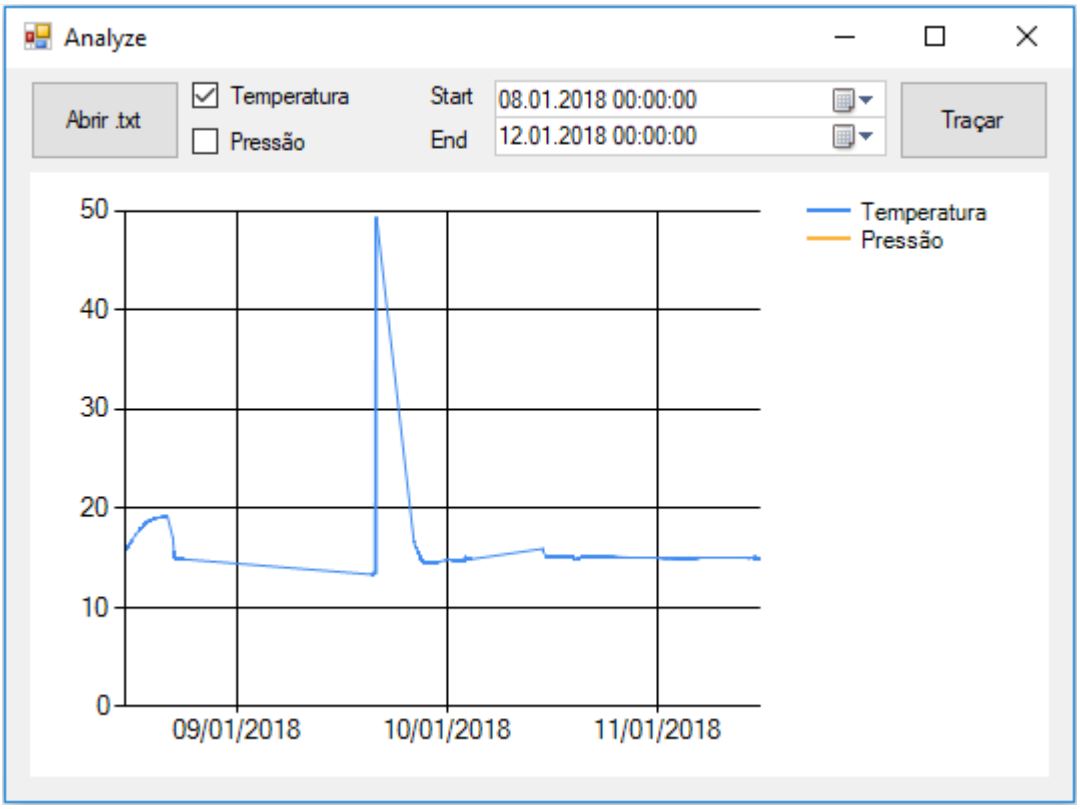


Figura 4.39 – Ferramenta de Análise de Dados

O funcionamento desta ferramenta é bastante simples e foi sendo concebida seguindo um caminho que a tornasse intuitiva e autoexplicativa.

Para observar uma determinada evolução temporal da temperatura ou da pressão ou das duas em simultâneo, começa-se por abrir um ficheiro de texto com os dados a analisar bastando para isso clicar no botão "Abrir .txt". Ao clicar aqui abrir-se-á um explorador de ficheiros do Windows que permite escolher o ficheiro a ler, por *default* o explorador começa na pasta no qual será provável estar o ficheiro pretendido (geralmente o ficheiro a abrir será o .txt gerado ao clicar no botão "Obter .txt" da *form* precedente e nesse sentido por *default* o explorador abre na pasta aonde esse ficheiro foi guardado). O utilizador poderá depois, através das caixas de *check*, selecionar o tipo de dados que quer analisar (temperatura, pressão) e com o selecionador de data e hora restringir o intervalo

de tempo de amostras a visualizar. Apurados os parâmetros de configuração da análise basta clicar no botão “Traçar” e o gráfico correspondente será exibido.

4.7 Sistema de Posicionamento Global

O sistema de posicionamento global, comumente chamado de GPS, é um sistema que fornece informações a um aparelho recetor de posição e de tempo em qualquer lugar da Terra. O GPS não necessita que o utilizador envie qualquer tipo de dados e opera independentemente de qualquer ligação telefónica ou de internet.

O módulo de comunicação SIM808 fornece uma solução de GPS na banda de frequência L1 para sistemas totalmente autónomos oferecendo uma boa aquisição do sinal e rigor no rastreamento.

Todas as funções do GPS são controladas por comandos AT via porta série. Aqui o GPS tem dois modos de operação que de forma inteligível são o modo ligado e o modo desligado. Quando é necessário ligar o GPS, o mecanismo de GSM deve de igual forma estar ligado, pois no módulo SIM808 o mecanismo de GPS é controlo pelo de GSM. Depois de ligado o dispositivo passa a ser um recetor de GPS e tratará automaticamente de adquirir e procurar satélites de GPS. No modo inativo, a fonte de alimentação interna para o GPS é desligada e o consumo de corrente é realmente muito baixo.

4.7.1 L1 C/A

A frequência L1 consiste num sinal de 1.57542GHz, mais especificamente, este módulo faz uso de um sinal disponível nesta frequência para uso civil, o L1 C/A, que resulta da modulação da onda portadora, L1, com a soma do código C/A e da mensagem de navegação.

O código C/A é um sinal de frequência 1.023MHz com uma taxa de transmissão de 1023Mbps (uma mensagem por cada milissegundo), disponível no canal L1. Este código consiste num PRN que aparenta ser aleatório, mas que na verdade é gerado por um algoritmo conhecido e cada satélite tem o seu próprio C/A garantido desta forma a distinção de um satélite entre os demais. A mensagem de navegação, também conhecida por *NAV data,* consiste numa *stream* de *bits* de uns e zeros a 50Hz que contém informação relativa a efemérides, tempo da semana, correções de tempo, almanaque e dados da ionosfera do respetivo satélite. A mensagem de navegação é adicionada ao código C/A usando o *modulo 2 adder,* esse processo de adição não é mais que o uso de um XOR, ou exclusivo, e pode ser observado na figura seguinte o seu funcionamento:

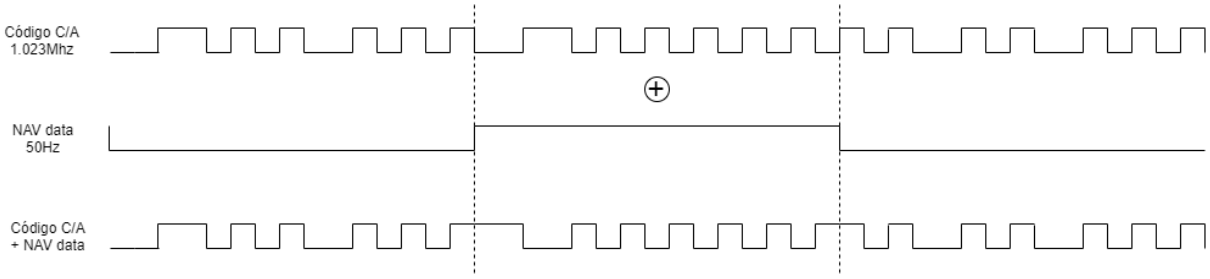


Figura 4.40 - L1 C/A - Modulo 2 adder

De modo que a informação contida neste sinal viaje desde o satélite até ao recetor de GPS é necessário modular o sinal numa onda portadora. Essa modulação é feita usando a técnica de BPSK e a onda L1 como onda portadora.

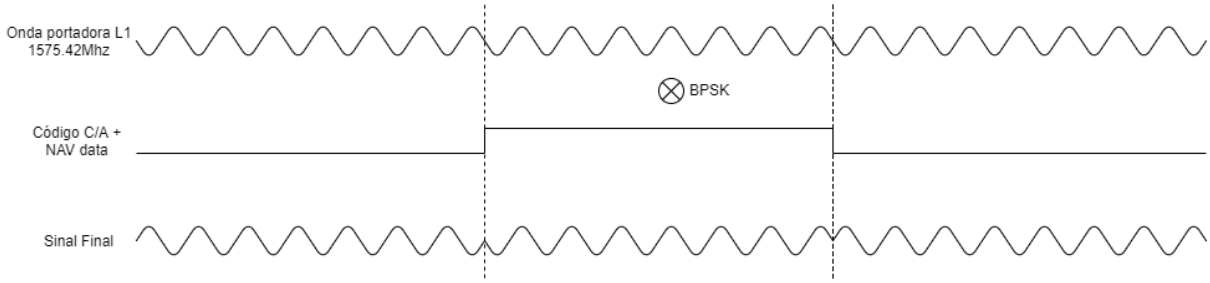


Figura 4.41 - L1 C/A - Modulação BPSK

4.7.2 Formato NMEA

Uma vez habilitada a funcionalidade do GPS, o módulo capta a informação relativa ao posicionamento atual apresentando-a em mensagens que seguem o formato definido pela NMEA. Esta associação desenvolveu um padrão de mensagens com vista a permitir uma comunicação de dados pronta e satisfatória entre instrumentos marinhos eletrónicos, equipamentos de navegação e equipamentos de comunicação, quando estes estão ligados entre si por uma interface apropriada [43]. A comunicação do recetor de GPS é feita dentro desta especificação. A ideia principal da NMEA é enviar uma mensagem que seja autónoma, diferenciável e independente de qualquer outra mensagem.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

Figura 4.42 - Exemplo de formato de uma mensagem da NMEA

Cada mensagem começa com o caracter "$" e termina com os respetivos caracteres de *carriege return* e *line feed*. A seguir ao primeiro caracter, os dois que se seguem imediatamente dizem respeito ao prefixo, que basicamente diz qual o tipo de mensagem que está a ser usado, que neste caso se expõe como sendo "GP", visto que esse é o prefixo associado a recetores de GPS. Este prefixo é seguido de mais uma sequência de três caracteres, que indicam o tipo de conteúdo da mensagem. Entre os modos suportados por este recetor GPS estão: "$GPGGA", "$GPGLL", "$GPGSA", "$GPGSV", "$GPRMC", "$GPVTG" e "$GPZDA" [44]. Destes modos foi usado o "$GPGGA" sendo que dele é possível obter a toda a informação essencial para determinar a posição da sonda. Esta mensagem pode ser analisada e decomposta tendo em conta a mensagem apresentada na Figura 4.42 e a seguinte tabela:

Tabela 12 - Conteúdo da mensagem $GPGGA

Valor	Significado	Descrição
$GPGGA	ID da mensagem	Cabeçalho do protocolo
123519	Tempo UTC	hhmmss
4807.038	Latitude	ddmm.mmmmmm
N	Indicador N/S	N=Norte, S=Sul
01131.000	Longitude	ddmm.mmmmmm
E	Indicador E/W	E=Este, W=Oeste
1	Indicador Qualidade de GPS	Não deve ser 0
08	Satélites Usados	0 a 12
0.9	HDOP	Degradação da posição horizontal
545.4	MSL Altitude	Altitude (base nível médio do mar)
M	Unidade	Metros
46.9	Separação geoide	Altura entre geoide e elipsoide global
M	Unidade	Metros
***47**	Checksum	Começa sempre por "*"
	<CR><LF>	Final da mensagem

O conteúdo dos restantes tipos de protocolo pode ser observado na referência [44].

4.7.3 Conversão para Decimal

O sistema de coordenadas definido pela NMEA são os graus minutos, [(d)ddmm.mmmm, onde 'd' significa graus e 'm' significa minutos]. Ainda que os valores apresentados sobre este formato representem a latitude e a longitude numa determinada posição, o seu formato não é prático para a aplicação em causa. Para tal procede-se à conversão do valor para o seu respetivo valor de graus decimais (dd).

O processo de conversão é bastante simples. Sendo que existem sessenta segundo num grau, dividem-se os minutos representados neste formato por sessenta e adiciona-se o resultado aos graus já existentes no formato. Deve-se, no entanto, ter especial cuidado com o sinal do resultado final. No caso da latitude, se o indicador N/S for sul, significa que o resultado final terá sinal negativo. O mesmo acontece na longitude, caso o indicador E/W apontar para oeste.

Mais uma vez, usando o exemplo da Figura 4.42, verifica-se que os valores convertidos da latitude e da longitude corresponderão a 48.1173 e 11.517 respetivamente. Este formato torna-se mais prático e conveniente à interface de utilizador desenvolvida, pois nela obtendo as coordenadas desta forma permite o redireccionamento através de uma hiperligação até à página web, propriedade da *Google Maps*, que mostra a posição da sonda no planisfério.

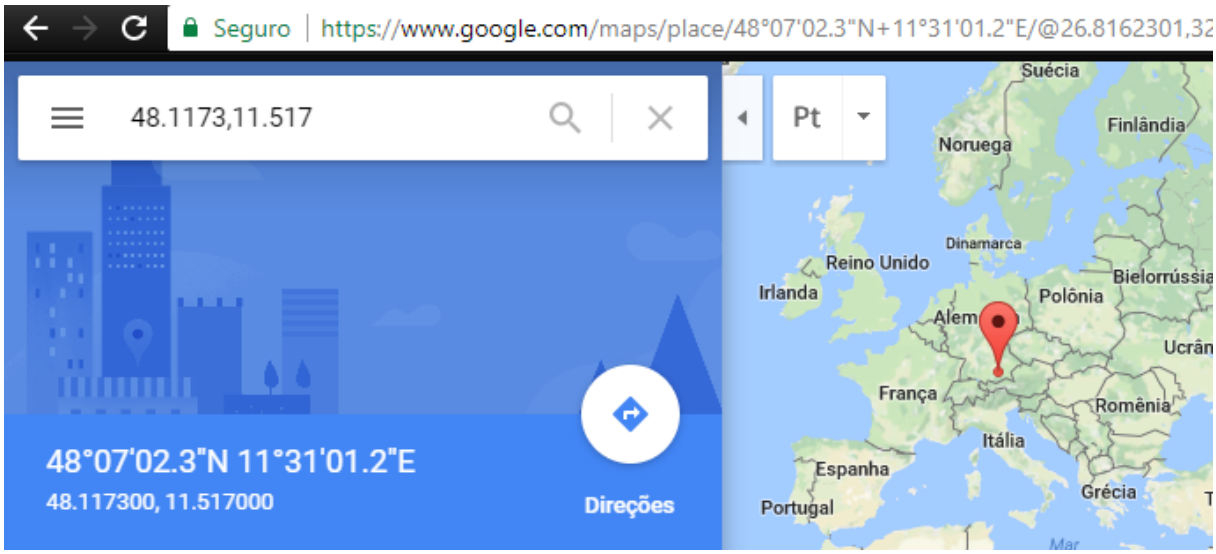


Figura 4.43 – Exemplo de resultado no Google Maps face às coordenadas obtidas

Na Figura 4.43 é possível constatar que as coordenadas aleatoriamente atribuídas no exemplo da Figura 4.42, correspondem na verdade uma localidade próxima de Munique na Alemanha.

4.8 Outras Funcionalidades

Este subcapítulo diz respeito à implementação de algumas funcionalidades ainda que não totalmente importantes, no sentido em que não faziam parte dos requisitos iniciais, acrescentam valor ao projeto.

4.8.1 Sincronização de Frequências de Leitura e Comunicação

De modo a periodicamente fazer leituras do sensor ou transmitir dados para o servidor são usados *timers* integrados no microcontrolador. Uma vez que os microcontroladores operam sobre uma frequência de *clock* predefinida, todos eles têm uma forma de configurar *timers* [45].

Basicamente um *timer* é um registo, mas um registo que aumenta ou diminui o seu conteúdo automaticamente. O microcontrolador usado neste projeto, Arduino nano, é dotado de dois tipos de *timers,* de 8 *bits* e de 16 *bits,* que permitem respetivamente contar 256 instruções ou 65536 instruções. Por esta razão os *timers* são também chamados por vezes de contadores. Quando um contador chega ao seu valor máximo, dá-se um *overflow* e o registo volta a 0. Outra das características mais importantes dos *timers* é que operam de forma independente do CPU, o que significa que, correm paralelamente o que torna a contagem mais precisa.

O valor do registo dos *timers* muda a cada pulso de *clock*. No caso, o microcontrolador usado tem um XTAL com uma frequência de 16MHz. Baseando na cógnita fórmula do calculo do período, T=1/f, significa que com esta frequência de XTAL cada *tick* do *clock* demora 0.0000625 milissegundos. Todavia, se o utilizador pretender estabelecer um intervalo de tempo, por exemplo, de um segundo entre leituras do sensor, indica que são precisos 15999999 pulsos de *clock*. Ora o registo do *timer* não conseguirá atingir este valor e fará sucessivos *overflows*, mesmo que seja o *timer* de 16 *bits,* pelo que, terão de ser usadas outras alternativas, como o uso de *prescalers* ou de interrupções.

O *prescaling* é uma técnica de divisão de frequência. A frequência do CPU, no entanto não é reduzida e mantem-se a mesma, 16MHz, o que acontece é uma derivação dessa frequência para correr no *timer*. Contudo, esta técnica não é completamente gratuita, no sentido em que há um *trade-off* entre a resolução e a duração [45]. A arquitetura AVR que está inerente aos Arduinos permite divisões da frequência de *clock* por 8, 64, 256, 1024, em que, uma maior divisão resulta num alcance de um intervalo de tempo maior, mas uma resolução inferior.

Outro aspeto associado aos *timers* são as interrupções. As interrupções são acionadas assim que certos requisitos se estabeleçam. Por exemplo, quando o registo que está a fazer uma contagem num *timer* e consente um *overflow,* pode ser acionada uma interrupção configurando previamente um *bit* de um registo especializado que a faça disparar. Dentro de uma interrupção, a função *main* é guardada e parada, e são realizadas as instruções pedidas na interrupção, que podem ser, por exemplo, incrementar um contador ou alternar o estado de um pino.

Entrando mais concretamente dentro da implementação feita, a sincronização do período de leituras do sensor é feita usando o *timer 0* e a sincronização do período de envio de dados para o servidor é feita pelo *timer 2.* A sonda retira do servidor os intervalos de tempo definidos pelo utilizador e inicializa os *timers,* escolhendo o *prescaler* mais adequado e ativando quais as interrupções a usar associadas aos *timers* (interrupção *overflow* e de comparação).

4.8.2 UART por *Software*

Uma das maiores limitações do Arduino nano é o facto de só possuir uma porta série. Isto torna-se um problema quando tem que comunicar com um módulo de comunicação GSM que funciona por UART, quando se tem de transferir dados com outro microcontrolador por UART e ainda quando se pretende fazer *debug* do código num terminal no computador. A implementação de uma porta série por *software* surge pela necessidade de puder fazer estas três coisas em simultâneo, dedicando assim uma porta série por *software* só para, exclusivamente, estabelecer uma comunicação entre o microcontrolador e o módulo de comunicação GSM.

O maior problema que surge na implementação de uma UART por *software* é ajustamento do tempo. Pode existir apenas um pequeno atraso, mas este resultará numa acumulação de atrasos que pode provocar o não envio ou receção de um *bit* resultando numa transmissão sem sentido. Portanto, o uso de um *timer* pode ser uma boa alternativa, no caso, foi usado o *timer 1.* O modo de envio é simples, é usada uma interrupção de comparação (TX) [46]. Já o modo de receção é um pouco mais complexo, para determinar quando há dados para ler com mais precisão é usada a interrupção de captura de entrada e depois faz-se a análise dos *bits* individuais com a outra interrupção de comparação livre do *timer* (RX) [46]. Isto significa que, o UART por *software* é *full-duplex* e usa os pinos do Arduino "IPC1" e "OC1A" para fazer essa transmissão.

4.8.3 Comunicação Bidirecional entre µControladores

Fisicamente a sonda é constituída por dois microcontroladores. Um microcontrolador que vai para debaixo de água e faz as leituras do sensor e outro microcontrolador que se encontra na superfície que faz todo o restante processamento necessário tal como: gravar os dados para o cartão SD, comunicar com o servidor, etc. Este microcontrolador que fica submerso está devidamente estancado para garantir renúncia à entrada de água, e está constantemente a fazer leituras do sensor em *burst mode.* Quando for requisitado pelo microcontrolador principal dados, interrompe as suas leituras e faz a transmissão dos valores obtidos pelo sensor mais recentes. Esta interação bidirecional entre microcontroladores é feita de forma assíncrona por dispositivos de *hardware* que são comuns aos dois microcontroladores, UART.

De todos os protocolos de comunicação disponíveis para estabelecer uma interação entre os microcontroladores, este tipo de conexão foi adotado, porque a distância que separa os dois microcontroladores ronda os oito metros de distância e após alguma pesquisa reparou-se que, quando o barramento é maior, este apresenta melhores resultados comparando por exemplo com outros tipos de protocolos como SPI ou I^2C [47]. Claro que, estas conjeturas são suscetíveis de serem pouco precisas, pois há diversos fatores que determinam o comprimento máximo do barramento (velocidade, impedância da linha, ...). Todavia aliada a esta pesquisa, foi comprovado de forma empírica que de facto o UART é uma alternativa viável, pelo menos há distância proposta.

Assim sendo, o microcontrolador que está ligado ao sensor está constantemente a fazer leituras e quando é detetado na sua interrupção de receção da porta série um determinado caracter especial definido previamente, interrompe o processo de leituras entra na interrupção e ativa uma *flag* que vai depois provocar o envio da leitura mais recente feita. Este envio é feito *byte* a *byte*, pelo que são transmitidos oito *bytes* (dois *floats,* um alusivo à temperatura, outro à pressão). Do ponto de vista do outro microcontrolador, ele envia esse determinado caracter definido e reconhecido entre ambos periodicamente, seguindo a frequência de amostragem imposta pelo utilizador, e aguarda uma resposta, ou seja, espera que cheguem os oito *bytes* na sua interrupção da porta série.

4.8.4 *Watchdog*

A ideia desta implementação é que nunca seja necessária na verdade. Se o *watchdog* tiver que atuar significa que algo de errado se passou no decorrer do programa, ou seja, esta implementação é de pura precaução e não de necessidade. Para um sistema que se propõe a atuar de forma autónoma por períodos de tempo alargados é bom ter uma maneira de fazer *reset* ao

sistema caso alguma coisa corra mal, não comprometendo assim o propósito da campanha, relembrando que, a sonda para além de estar a operar de forma autónoma, pode estar também em sítios de difícil acesso.

O *watchdog* é um *timer* especial que pode ser ativo em qualquer parte do código. Quando ele é ativo deve ser garantido que uma certa sequência de instruções é completa dentro de um espaço de tempo definido. Este espaço de tempo ou *delay* pode ser configurando alterando valores nos registos do próprio *timer* do *watchdog.* No caso de as instruções serem realizadas dentro desse espaço de tempo definido, o *watchdog* deve ser desligado e a execução do programa continua, ou, no caso de não conseguir realizar as instruções a tempo, o sistema inteiro reinicia evitando assim, que o programa bloqueie ou o sistema se desligue.

5. ESTRUTURAÇÃO DO INVÓLUCRO

Neste capítulo é abordado com mais detalhe a organização relativa aos componentes mais físicos integrados na sonda.

5.1 Desenho de PCBs

Com vista a tornar o produto final mais simples e estável, foi desenvolvida uma PCB para suportar mecanicamente os componentes e para eletricamente interligá-los através de pistas de cobre impressas numa placa não condutora.

Existem vários outros programas de design de PCBs, mas optou-se por usar um que está cada vez mais a ser utilizado e se está a tornar cada vez mais popular em toda a comunidade de design, o Altium Designer. O design do esquemático é bastante semelhante quando comparado com outros programas do género, contudo é no editor do *layout* da PCB que ele se destaca.

O processo de criação da PCB começa pelo desenho do esquema elétrico do circuito, para instruir o programa quais os componentes que são usados e como estão conectados uns com os outros. Depois disso, prepara-se o *layout* da PCB e para tal é necessário transferir o diagrama esquemático previamente desenhado num desenho da placa de circuito impresso. É nesta fase que se define a orientação e tamanho das pistas de cobre que percorrem a placa.

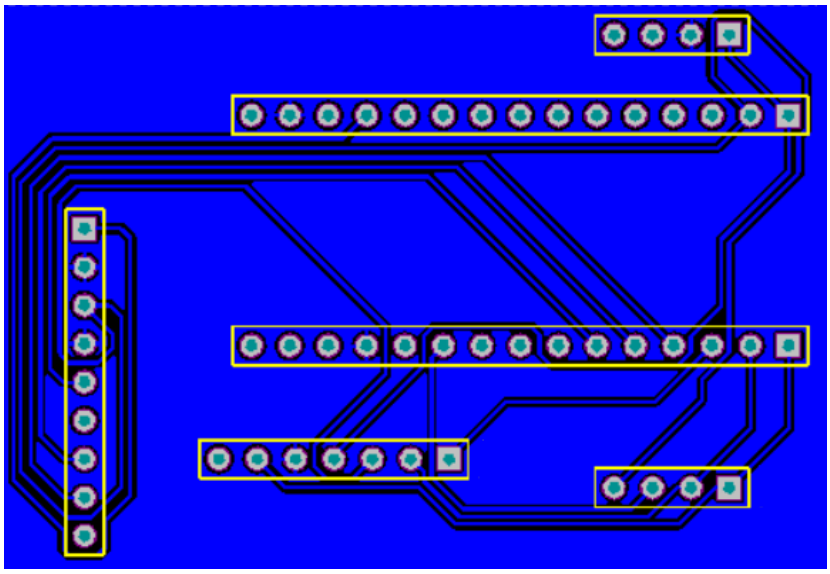


Figura 5.1 - PCB Sonda

5.2 Antenas

Na totalidade, o sistema da sonda está dotado de duas antenas, uma antena GSM e uma antena GPS.

A antena GSM, é uma antena omnidirecional com um conector SMA de tamanho compacto e fácil de usar. A sua base magnética permite que ela seja facilmente aplicada à superfície de um material ferroso. Tem uma frequência de operação que vai desde os 900MHz aos 1800Mhz, o que suporta o requisito da aplicação GSM em causa. A extensão da antena é um cabo com três metros de comprimento do tipo RG174, que é cobre, o que faz com que o cabo tenha uma menor impedância em comparação com outros tipos de cabo. O ganho do sinal da antena pode chegar até aos 2.5dBi.

Por sua vez, a antena GPS também está equipada com um conector SMA. Há semelhança da antena GSM, a antena GPS é igualmente compacta e fácil de usar e possui uma base magnética para aplicar em qualquer material metálico. A sua frequência de operação encontrar-se nos 1575.42MHz, o que vai de encontro à frequência do sinal L1 descrito no capítulo 4.7. O cabo é também do tipo RG174 e a antena apresenta um ganho de sinal na ordem dos 26dBi.

Como as duas antenas estão munidas de um conector do tipo SMA e a placa de comunicação SIM808 possui apenas conectores do tipo fêmea U.FL, isto significa que, foi necessário emparelhar aos cabos da antena um adaptador de SMA para U.FL para que a sonda pudesse captar as radiofrequências.

5.3 Impermeabilidade do Invólucro

Uma vez que há uma parte do sistema que deve ficar submerso (parte que faz as leituras do sensor), deve ser garantida a estanquicidade e resistência à pressão até dez metros de profundidade.

Para tal, foi idealizado um invólucro em PVC para isolar o conteúdo eletrónico da água. O PVC é um produto de grande versatilidade e cheio de potencialidade de aplicações e de baixo custo. Além disso, garante resistência à água e outros líquidos, gases, fungos, choques e oferece bom isolamento térmico e elétrico. A nível de operação a diferentes condições de pressão, continua a oferecer uma boa relação de resistência com a profundidade. Segundo as especificações *standard* para PVC da ASTM, as recomendações máximas de operação de pressão e os valores mínimos de pressão para rutura do PVC, às condições que a sonda se deve inserir, nunca serão atingidas [48]. De resto na figura seguinte, é possível observar essa padronização feita pela ASTM.

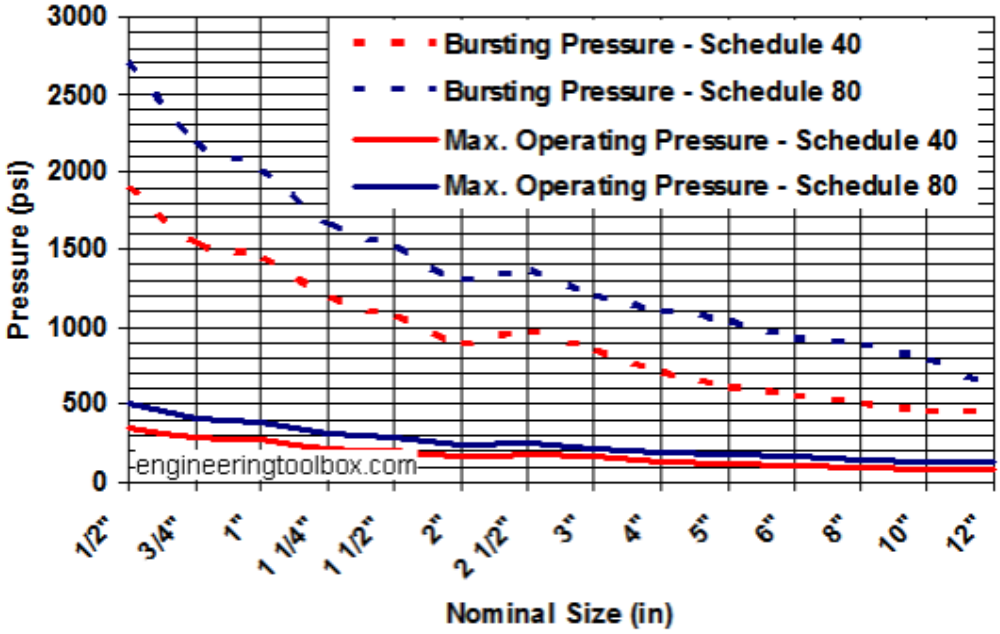


Figura 5.2 - Condições de Operação para Tubos PVC [48]

"Schedule 40" e "Schedule 80" fazem referência à grossura das paredes dos tubos PVC, onde o 80 será naturalmente mais espesso que o de 40. O tubo usado é do tipo 40, e tem 2cm de diâmetro (3/4''), pelo que imaginando que ele esteja submerso àquela que será a profundidade máxima, 10 metros, o tubo estará sujeito a uma pressão de aproximadamente 1 bar, valor esse, que está muito longe do que o tubo pode suportar na teoria, 20 bar, pelo que esta margem é tão grande que mesmo que haja algum erro neste cálculo jamais estará comprometida a integridade da sonda a esta profundidade.

A montagem do sensor requer a perfuração de um buraco de 6mm no próprio PVC, de modo a que a parte do sensor que faz as leituras propriamente dita fique exposta e em contato com a água. Todavia, entre a perfuração e o sensor é aplicada uma camada de silicone para vedar a possibilidade de entrada de água. Por sua vez, no lado verticalmente oposto a esta perfuração existe outra, para a entrada do cabo, CAT5, que é encarregue da comunicação UART com o outro microcontrolador. Da mesma forma, é aplicado silicone para garantir impermeabilização. Nesta perfuração é ainda aplicado um bucim para que, além de ajudar à estanquicidade, evite que o cabo exerça uma força excessiva na dobra e comprometa a integridade dos fios dentro dele.

A caixa que está à superfície é aonde reside o microcontrolador principal, bem como as baterias e antenas. Esta é uma caixa sem embute do tipo IP65, visto que, este índice de proteção é

suficiente já que não terá de ficar submersa [49]. Nela é ainda feita uma perfuração para ligar a ficha IP68 que virá do cabo.

6. TESTES E RESULTADOS

Neste capítulo são apresentados alguns resultados feitos em laboratório e no campo obtidos usando a integração do sistema implementado nos dois capítulos precedentes. Todavia, antes do teste do sistema como um todo, os diversos módulos constituintes do sistema foram individualmente testados e verificado se os seus comportamentos eram os desejados.

6.1 Cenários de Teste

Mediante os testes que foram feitos em laboratório ou no campo, os cenários de testes e, essencialmente, as suas condições ambientais a que a sonda ficou sujeita mudaram.

6.1.1 Testes de laboratório

Os testes experimentais foram realizados num canal hidráulico da Gunt, modelo HM 162, presente no departamento de engenharia civil da Universidade do Minho. Este canal permite estudar escoamentos com superfície livre. O canal possui uma secção retangular com 0.3 metros * 0.45 metros * 10.0 metros de desenvolvimento e inclinação variável. A adição de água é efetuada por uma bomba centrífuga com capacidade máxima para 150 m^3/h e o caudal medido por um medidor de caudal magnético.

Figura 6.1 - Canal Hidráulico Gunt HM 162

Esta geometria e características do canal garantem condições semelhantes àquelas que são por vezes sentidas em ambientes naturais, como por exemplo alguns rios.

6.1.2 Testes de campo

Embora o cenário de laboratório gere condições um pouco semelhantes àquelas que se observam num rio, existem, contudo, adversidades que podem surgir imprevisivelmente fruto da natureza. Nesse sentido, levou-se a sonda até ao rio Lima, em Lanheses mais propriamente, onde foi testado o funcionamento da sonda. Aqui, aproveitou-se e fizeram-se as medições da sonda desenvolvida nesta dissertação em paralelo com uma sonda comercial, a sonda TG2050, para comparar.

Figura 6.2 - Local de Teste de Campo - Lanheses

6.2 Resultados

Com o intuito de obter alguns resultados que comprovem o correto funcionamento da sonda, foram realizados alguns ensaios nos quais se fez variar o nível de água no canal e verificar se essa variação se traduz numa variação da pressão plausível.

Na tabela seguinte é possível ter uma ideia do que determinada coluna de água em metros significa na unidade de pressão de bar.

Tabela 13 - Tabela de Conversão mH20 para bar

mH_2O	bar
0.05	0.0049
0.1	0.0098
0.15	0.0147
0.2	0.0196
0.25	0.0245
0.3	0.0294
0.5	0.0490
1	0.0980
2	0.1961
5	0.4903
10	0.9806

Durante um período de sensivelmente quatro horas e meia foi regulada e monitorizada a altura do nível de água no canal. A sonda foi inserida dentro do canal e registou o seguinte resultado:

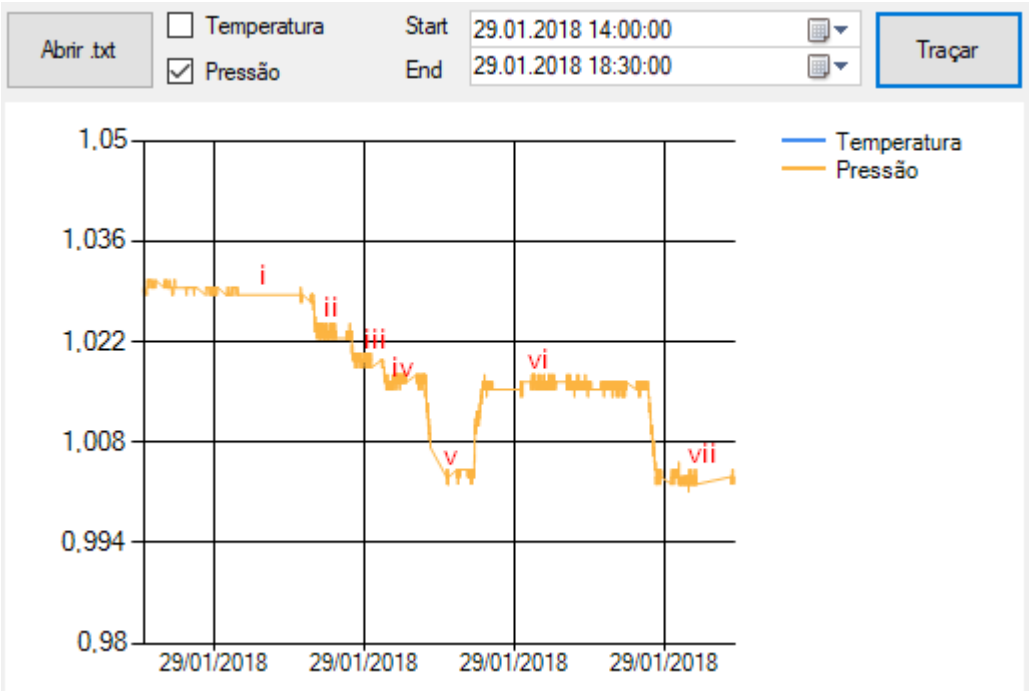


Figura 6.3 - Teste Variação da Pressão

O valor da pressão atmosférica no momento do ensaio rondava os 1.0032bar.

Em numeração romana estão destacados na imagem alguns eventos gerados com o intuito de verificar o comportamento da sonda. O significado de cada um desses eventos pode ser conhecido atentando na seguinte interpretação:

i) Marca o início da leitura. A sonda é depositada no canal já com água a circular. Desde o sensor até à tona da água vão sensivelmente vinte e seis centímetros. Na teoria a esta altura de coluna de água o valor de pressão sentido deveria ser de 1.02869bar, que na verdade não é impreciso.

ii) Provocou-se uma descida de cinco centímetros no nível de água no canal. Isto significa que, a pressão medida dever-se-ia ajustar para a ordem dos 1.0237bar, e de facto, a medição obtida segue esse valor.

iii) Fez-se mais uma descida do nível de água do canal, desta vez de três centímetros. Ou seja, o sensor está neste momento a dezoito centímetros abaixo da linha de água, o que acumulado à pressão atmosférica deve dar um valor a rondar os 1.020bar, o que confere.

iv) Este passo é exatamente igual ao passo iii), isto é, provocou-se uma descida de três centímetros de novo. E mais uma vez concorda com o valor que supostamente na teoria se deveria obter, 0.0179bar.

Figura 6.4 - Sonda no Canal Hidráulico

v) Neste evento observa-se pelo valor que o gráfico mostra que esse valor está muito perto do valor da pressão atmosférica, 1.0032bar. Na verdade, isto é coerente, pois neste evento esvaziou-se na totalidade o canal, pelo que o sensor passar a estar a medir somente pressão atmosférica.

vi) Voltou a ligar-se o canal. Ao ligar o canal, ele volta ao nível de água que tinha antes de ter sido desligado. Nesse sentido era espectável que, o valor de pressão medido fosse igual ao do evento iv) e pela observação do gráfico constata-se que isso aconteceu.

vii) Neste último evento simplesmente voltou-se a desligar o canal, pelo que o valor de pressão lido voltou a ser o valor correspondente à pressão atmosférica.

Uma vez que o correto funcionamento da sonda foi verificado em laboratório passou-se para os testes em campo. Aqui, a sonda foi mergulhada num rio e fez medições paralelamente com uma sonda comercial. Registaram-se os seguintes resultados:

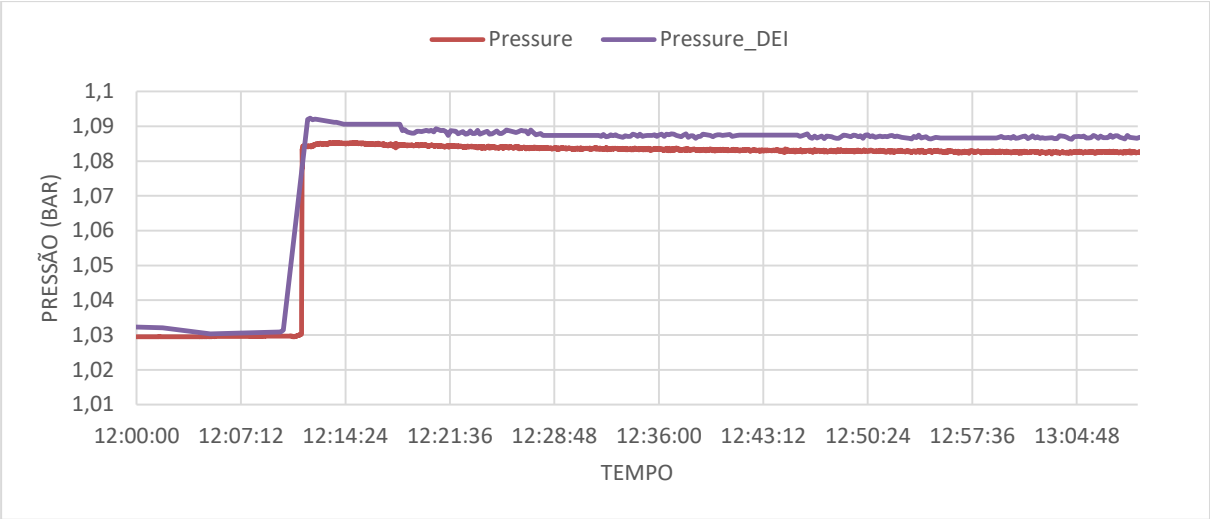


Figura 6.5 - Resultados em Campo – Pressão

Como se pode observar, a sonda desenvolvida, "Pressure_DEI", segue o padrão captado pela sonda TGR-2050, o que por si só é bom indicador, uma vez que esta é uma sonda comercial e que se admite ser fiável. Algumas incongruências na leitura do gráfico também podem ser justificadas pelo facto de na altura das medições a sonda TGR-2050 estar a funcionar à frequência de amostragem de 1Hz, ou passo que a sonda desenvolvida estava a funcionar à frequência de amostragem de 0.1Hz, ainda que, esta sonda desenvolvida tenha a capacidade de operar até 4Hz. Essa é de resto, a frequência de amostragem suficiente para detetar a passagem de ondas.

Outro aspeto relevante é observar também a evolução temporal da temperatura nesta mesma campanha:

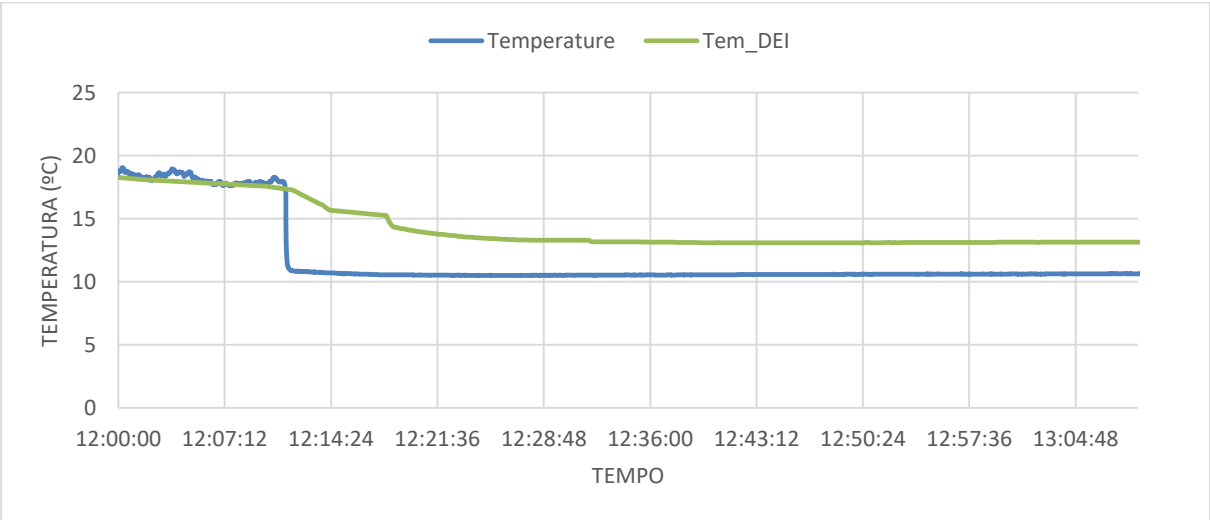


Figura 6.6 - Resultados em Campo – Temperatura

Neste gráfico pode-se observar que em ambas as sondas o valor da temperatura desceu rapidamente mal as sondas foram inseridas dentro de água e que em ambas convergiram para um valor de temperatura estável ao fim de algum tempo. O facto de esse valor de temperatura para o qual as sondas convergiram não ser o mesmo pode ser justificado por, apesar de terem sido mergulhadas à mesma profundidade, terem sido colocadas em pontos do rio diferentes.

7. CONCLUSÕES

Chegando à conclusão da dissertação é altura de sumarizar as contribuições deste projeto e discutir as perspetivas futuras.

7.1 Sumário

Desde sempre o Homem mostrou grande interesse em explorar os cursos de água e os oceanos, no entanto só desde o último século esse interesse se mostrou mais evidente. Aliás, o ser humano está na verdade cada vez mais dependente destes recursos naturais. E essa evidência manifesta-se ao observar o esforço empregue pelo Homem ao desenvolver ininterruptamente novos meios e formas de conhecer e perceber os processos naturais que ocorrem nestes ecossistemas. Existe uma ávida vontade de substituir tecnologias obsoletas, de forma a que, cada vez mais seja mais fácil e precisa a extração de informação deste tipo de meios, como de resto se pôde observar pela análise ao estado da arte que foi enunciado nesta dissertação. De facto, esta preocupação é necessária, pois nos dias de hoje, nestes espaços recolhem-se bens alimentares, fazem-se construções, extraem-se recursos naturais (petróleo, ...), realizam-se transportes, fazem-se estudos de previsão de fenómenos naturais, etc. E é aqui que o propósito desta dissertação aparece como uma mais valia. Ainda que já existam aparelhos de medição capazes de estudar estes meios, o protótipo aqui desenvolvido tem como principal vantagem propor-se a fazer igualmente essas tarefas, mas com uma relação de custo e eficiência melhor.

Para além do facto de ser uma sonda de um custo consideravelmente mais baixo quando comparada com as demais, ela oferece ainda mais vantagens. Desde logo, uma dessas vantagens é o facto de a extração de dados capturados não envolver a remoção da sonda da água para o terreno, uma vez que ela está dotada da capacidade de autonomamente estabelecer uma comunicação com um servidor e transferir os dados exatamente a partir do local em que se encontrar. Depois outros dos problemas que podem estar associados a estes de equipamentos é o facto de terem pouca capacidade de armazenamento, porém nesta sonda existe um grande volume de armazenamento e não é falacioso dizer que em todas as numerosas medições que foram feitas nem dez por cento da totalidade da unidade de armazenamento foi ocupada. Outra característica favorável a esta sonda é o consumo energético, em que com suporte de bases empíricas se assume que tenha uma autonomia capaz de realizar campanhas de forma autónoma por sensivelmente uma

semana. É verdade que provavelmente as sondas existentes no mercado oferecem maior robustez, contudo muitas vezes, com essa robustez vem um impacto visual e ambiental notório, e já com esta sonda esse aspeto é praticamente insignificante e continua a oferecer uma resistência razoável.

Todo o documento redigido surge sobre a forma de um documento que para além de fornecer conhecimento teórico, explora com detalhe cada um dos diversos módulos que constituem o sistema, resultando dessa forma igualmente num documento técnico que instrui o leitor sobre diversas matérias a um nível mais prático. Esta dissertação aborda variados conceitos e tecnologias ligadas por exemplo, aos sistemas embebidos, às telecomunicações, sistemas de informação e até mesmo às microtecnologias.

Uma das maiores adversidades encontradas no desenvolvimento deste projeto, na ótica do autor desta dissertação, resulta possivelmente também numa das mais valias dele. O facto de se desenvolver um sistema embebido coordenado por um microprocessador de relativamente poucos recursos, ATmega328p, obrigou à exploração quase que exaustiva de todos esses recursos para se conseguir atingir todos os requisitos do projeto. Nesse sentido, esta dissertação também instrui a quem pretenda usar microcontroladores da família AVR em projetos de outra natureza, pois aqui praticamente todas as potencialidades dele foram exploradas.

De resto, dizer ainda que, o sistema como um só é capaz de cumprir todos os objetivos propostos inicialmente.

7.2 Perspetivas Futuras

Ainda que, do ponto de vista do cumprimento dos requisitos principais do projeto tenham sido superados os objetivos propostos, existem sempre funcionalidades que possam ser implementadas ou melhoramentos que possam ser feitos.

Debruçando um pouco sobre o estado da arte pode-se verificar que muitos dos equipamentos semelhantes não se limitam a medir só pressão e temperatura, alguns deles, medem outro tipo de grandezas como a salinidade por exemplo. Logo, esta poderá ser um dos melhoramentos futuros aplicando à sonda mais tipos de sensores para consequentemente medir mais tipos de grandezas. Esta não deverá ser uma dificuldade acrescida, uma vez que da maneira que o sistema está implementado, ele funciona quase como um *gateway* e facilmente podem ser adicionados novos sensores tirando proveito, por exemplo, dos barramentos de SPI e I^2C.

De relembrar que nesta dissertação somente foi desenvolvido um protótipo, pelo que a nível de construção do invólucro, este apresenta uma estrutura, ainda que segura e testada, pouco robusta e essa poderia ser uma das melhorias no sentido de redesenhar uma estrutura nova.

Uma das grandes lacunas que acabou por ficar por preencher nesta dissertação, foi a concretização de testes de campo. Ainda que ficou provado que a sonda funciona corretamente em testes realizados em laboratório, este não irá ser o ambiente sobre o qual ela irá atuar. Nesse sentido ficou a faltar perceber o comportamento da sonda num ambiente não controlado e sujeito a vários agentes imprevisíveis.

8. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Sea-Bird Scientific, "CTDs Explained Measure Conductivity and Temperature with an Oceanographic CTD," 2016.

[2] R. H. Stewart and R. Stewart, "PHYSICAL OCEANOGRAPHY PHYSICAL OCEANOGRAPHY Introduction to Introduction to Introduction to Physical Oceanography."

[3] L. W. Mays, *Ancient water technologies*. Springer, 2010.

[4] "Early River Valley Civilizations, 3500 B.C. - 450 B.C." [Online]. Available: https://www.cusd200.org/cms/lib/IL01001538/Centricity/Domain/267/_files/World_Civ_Chapter_2.pdf. [Accessed: 19-Sep-2017].

[5] S. O. Science and D. R. Montgomery, "Emperor Yu's Great Flood."

[6] J. C. I. Dooge, "Concepts of the hydrological Cycle. Ancient and modern Les concepts des cycles hydrologiques. Anciens et modernes," 2001.

[7] I. Pinto, "Introdução aos Sistemas de Informação Geográfica (SIG)." 2009.

[8] D. Ames Baker, "Ocean Instruments and Experiment Design."

[9] T. Snodgrass Bratt and M. Snodgrass Betancourt, "Frank Edwin Snodgrass," 1920.

[10] O. Cote and H. Sapolsky, "Antisubmarine Warfare after the Cold War Executive Summary."

[11] R. S. Winokur, "Naval oceanography contributions to underwater acoustics—The Cold War era," *J. Acoust. Soc. Am.*, vol. 137, no. 4, pp. 2306–2307, Apr. 2015.

[12] "Bottom Sediment Temperature Gradient Recorder, electronic chassis of BSTGR | Digital Collections | UC San Diego Library." [Online]. Available: https://library.ucsd.edu/dc/object/bb8738319m. [Accessed: 17-Dec-2016].

[13] "Mid-Pacific Expedition, 1950 (Material de arquivo, 1950) [WorldCat.org]." [Online]. Available: http://www.worldcat.org/title/mid-pacific-expedition-1950/oclc/35846844. [Accessed: 17-Dec-2016].

[14] "Shipboard Report, Capricorn Expedition 26 September 1952-21 February 1953. SIO Reference 53-15," 1952.

[15] "Ocean Sense Welcome to Ocean Sense!"

[16] "SEACAT Profiler CTD SBE 19plus V2."

[17] "CTDplus 100, 500 AND 1000 Brochure."

[18] M. C. Eble and F. I. Gonzalez, "Deep-Ocean Bottom Pressure Measurements in the Northeast Pacific."

[19] S. Wood, “26 Autonomous Underwater Gliders.”

[20] C. C. Eriksen *et al.*, “Seaglider: A Long-Range Autonomous Underwater Vehicle for Oceanographic Research,” *IEEE J. Ocean. Eng.*, vol. 26, no. 4, 2001.

[21] S. A. Jenkins *et al.*, “Underwater Glider System Study,” *Scripps Inst. Oceanogr.*, 2003.

[22] “On The Design and Implementation of Argo A Global Array of Profiling Floats.”

[23] “Argo - part of the integrated global observation strategy.” [Online]. Available: http://www.argo.ucsd.edu/. [Accessed: 19-Dec-2016].

[24] “Argo FAQ.” [Online]. Available: http://www.argo.ucsd.edu/FAQ.html#find. [Accessed: 15-Dec-2016].

[25] Y.-H. T. Chen, “Mapping Hidden Water Pipelines using a Mobile Sensor Droplet.”

[26] “SD Specifications Part 1 Physical Layer Simplified Specification Technical Committee SD Card Association,” 2006.

[27] A. Freedman, “Handoff in GSM/GPRS Cellular Systems.”

[28] J. Cai and D. J.Goodman, “General Packet Radio Service in GSM,” *IEEE Commun. Mag.*, 1997.

[29] C. Bettstetter, L. J. Vogel, and J. Eberspächert, “GSM Phase 2+ general packet radio service GPRS: Architecture, protocols, and air interface,” *IEICE Trans. Commun.*, vol. E83–B, no. 2, pp. 117–118, 2000.

[30] ETSI, “Digital cellular telecommunications system (Phase 2+); General Packet Radio Service (GPRS); Service description; Stage 2.” 1997.

[31] TSGC, “TS 123 003 - V14.5.0 - Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Numbering, addressing and identification (3GPP TS 23.003 version 14.5.0 Release 14),” 2017.

[32] K. Kymalainen, “3rd Generation Partnership Project; Technical Specification Group Core Network and Terminals; Numbering, addressing and identification (Release 13).”

[33] IEEE 802.16, “Use of the IEEE 802.16 Operator ID with IEEE Std 802.16 Wireless Metropolitan Area Networks.”

[34] R. Fielding *et al.*, “RFC2616 - Hypertext transfer protocol–HTTP/1.1,” *Internet Eng. Task Force*, pp. 1–114, 1999.

[35] W3C, “Forms in HTML documents.” [Online]. Available: https://www.w3.org/TR/REC-html40/interact/forms.html#form-content-type. [Accessed: 06-Jan-2018].

[36] “Tecnologias” [Online]. Available: http://www4.di.uminho.pt/~gepl/SIEP/tecnologias.htm.

[Accessed: 14-Jan-2018].

[37] J. Mayo, *C# unleashed*. Sams, 2002.

[38] J. G. R. Sathiaseelan and N. Sasikaladevi, *Programming with C# .NET*. Prentice Hall of India, 2009.

[39] A. Singh, "Process.Start() in C#." [Online]. Available: https://www.mindstick.com/Articles/628/process-start-in-c-sharp-with-examples. [Accessed: 18-Jan-2018].

[40] "NuGet Gallery | HtmlAgilityPack 1.6.15." [Online]. Available: https://www.nuget.org/packages/HtmlAgilityPack/. [Accessed: 25-Jan-2018].

[41] H. Borland and J. Kilmer, "Document Object Model ™ (DOM)," *I Can*, pp. 192–231, 2001.

[42] S. Gupta, G. Kaiser, D. Neistadt, and P. Grimm, "DOM-based content extraction of HTML documents," *Proc. twelfth Int. Conf. World Wide Web WWW 03*, p. 207, 2003.

[43] "National Marine Electronics Association NMEA 0183 Standard For Interfacing Marine Electronic Devices COPYRIGHT© NMEA 2002," 2002.

[44] "SIM808_GPS_Application Note_V1.00 Smart Machine Smart Decision," 2014.

[45] "Introduction to AVR Timers » maxEmbedded." [Online]. Available: http://maxembedded.com/2011/06/introduction-to-avr-timers/. [Accessed: 27-Jan-2018].

[46] P. Dannegger, "Software UART - Mikrocontroller.net." [Online]. Available: https://www.mikrocontroller.net/topic/38928. [Accessed: 28-Jan-2018].

[47] "NodeMCU y Arduino: ¿Cómo interconectarlos? UART, i2C y SPI - BBits." [Online]. Available: https://borrowbits.com/2017/11/como-comunicar-arduino-con-nodemcu-parte-i-conexion-serieuart/. [Accessed: 28-Jan-2018].

[48] ASTM, "Designation: D1785 – 12 Standard Specification for."

[49] Fantech, "Ingress Protection Rating," 2008. [Online]. Available: http://www.fantech.com.au/images/PDF/Catalogue/IP.pdf. [Accessed: 29-Jan-2018].