Universidade do Minho
Escola de Engenharia

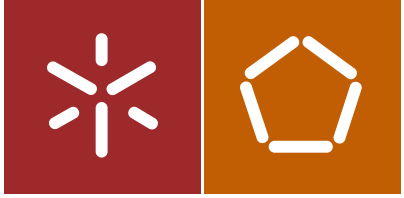
Henrique Castro Abreu

Monotorização das Condições Atmosféricas nas Autoestradas utilizando WSNs

dezembro de 2014

Henrique Castro Abreu Monotorização das Condições Atmosféricas nas Autoestradas utilizando WSNs

UMinho | 2014



Universidade do Minho
Escola de Engenharia

Henrique Castro Abreu

Monotorização das Condições Atmosféricas nas Autoestradas utilizando WSNs

Dissertação de Mestrado
Ciclo de Estudos Integrados Conducentes ao Grau de
Mestre em Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Jorge Cabral

dezembro de 2014

Anexo 3

DECLARAÇÃO

Nome: Henrique Castro Abreu

Endereço electrónico: henriqueabr@gmail.com

Telefone: 916911991

Número do Bilhete de Identidade: 13755683

Título dissertação

Monitorização de Condições Atmosféricas nas Autoestradas utilizando WSNs

Orientador:

Professor Doutor Jorge Cabral

Ano de conclusão: 2014

Ciclo de Estudos Integrados Conducentes ao Grau de Mestre em Engenharia Eletrónica Industrial e Computadores

---

Nos exemplares das teses de doutoramento ou de mestrado ou de outros trabalhos entregues para prestação de provas públicas nas universidades ou outros estabelecimentos de ensino, e dos quais é obrigatoriamente enviado um exemplar para depósito legal na Biblioteca Nacional e, pelo menos outro para a biblioteca da universidade respectiva, deve constar uma das seguintes declarações:

É AUTORIZADA A REPRODUÇÃO INTEGRAL DESTA TESE APENAS PARA EFEITOS DE INVESTIGAÇÃO, MEDIANTE DECLARAÇÃO ESCRITA DO INTERESSADO, QUE A TAL SE COMPROMETE.

Universidade do Minho, ___/___/_____

Assinatura: _____

Agradecimentos

Em primeiro lugar, gostaria de agradecer aos meus pais Henrique Abreu e Maria Castro, e à minha irmã Sílvia Abreu, por todo o apoio que me deram ao longo do meu percurso académico e pessoal. Sem eles não conseguiria, com todas as certezas, concluir esta etapa da minha vida.

Ao professor Jorge Cabral pela orientação nesta dissertação e pela confiança depositada em mim para a conclusão deste projeto. Não posso também deixar de agradecer ao João Brito pela ajuda e orientação na realização desta dissertação.

Quero fazer um agradecimento especial aos meus amigos de longos anos, Su, Inês, Pedro, Nando, Rafa, Ana e Vitinha, pela compreensão nos vários momentos em que não pude estar presente e pelo apoio e diversão que me foram proporcionando ao longo da minha vida.

Ao pessoal das tainadas, Novais, Tino, Herodes, Guevara, Fábio, Pimba e Raul, pelos momentos de descontração proporcionados. A eles um grande abraço!

Não podia deixar de agradecer ao pessoal do ESRG, em especial ao Hugo Gomes, que ao longo do ano me foram acolhendo e ajudando, cada um à sua maneira. Ao pessoal da sueca, do CS e aos demais, que me ajudaram ou na correção da dissertação ou nas dúvidas que foram surgindo no decorrer da mesma.

Por último, mas não menos importante, à minha namorada Patrícia Oliveira, pelos bons momentos, e pelo apoio, ajuda e compreensão nos menos bons.

A todos, muito obrigado!

Resumo

Ao longo dos anos, condições atmosféricas adversas nas autoestradas provocaram inúmeros acidentes rodoviários responsáveis por danos nas viaturas e infraestruturas civis, e infelizmente, acidentes fatais. Atualmente, apesar do esforço para evitar estas situações através de melhorias na segurança e avisos de condições menos favoráveis o número de acidentes ainda é demasiado elevado.

De forma a contribuir para a resolução deste problema, no âmbito do projeto *QREN SI I&DT SustIMS*, em co-promoção com a empresa Ascendi, é apresentado este sistema, de nome *HighWayMon*, que tem como objetivo a monitorização de condições atmosféricas nas autoestradas, com o intuito de ser possível prevenir algumas destas situações usando alguns sensores COTS. Para isso foi desenvolvida uma WSN (*Wireless Sensor Network*) capaz de obter as informações necessárias através de nós sensores, enviado-as para uma estação base responsável por interligar o sistema desenvolvido com outro capaz de processar e armazenar os dados adquiridos.

Através da modularidade do *hardware* e da funcionalidade *plug-and-play* inserida é possível realizar várias combinações dos nós sensores utilizando o mesmo *firmware*, permitindo adquirir dados de vários tipos de sensores que utilizem o mesmo tipo de interface.

Com o sucesso do projeto espera-se que seja possível usar este sistema para advertir os utilizadores das autoestradas dos perigos que podem encontrar ao longo da via, principalmente nos locais onde é previsto um maior número de acidentes.

Palavras-chave: WSN, Monitorização, Condições Atmosféricas, *Z-Stack*, Linguagem C, Modularidade do HW e *Plug-and-Play*.

Abstract

Over the years, adverse weather conditions on the highways led to several roads accidents causing huge damages in vehicles, infrastructures and, unfortunately, fatal accidents. Nowadays, despite the effort to overcome this situation by employing security improvements or warning signals, there are still to many accidents.

This thesis, under the project QREN SI I&DT SustIMS, with the company Ascendi, presents the HighWayMon system, that aims to provide a solution to the mentioned problems by proposing an autonomous weather monitoring system to report the conditions on the highways in order to be able to prevent accidents using COTS sensors. A Wireless Sensor Network (WSN) is developed to collect the necessary information by the sensor nodes and forward them to a base station responsible for connecting the network to other processing units for further processing.

The proposed system is also capable of performing various combination of devices using the same firmware through the developed hardware modularity and Plug-and-Play functionality. This allows the system to acquire only the necessary information from the network.

Upon finishing the project, it is expected to reduce the number of the accidents by issuing proper warning alerts to the drivers especially in the zones with the high risk of the accident.

Keywords: WSN, Monitoring, Weather Conditions, Z-Stack, C language, HW modularity and Plug-and-Play.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ADC	*Analog-to-Digital Converter*
AJAX	*Asynchronous JavaScript and XML*
API	*Application Programming Interface*
BPSK	*Binary Phase Shift Keying*
CCA	*Clear Channel Assessment*
COTS	*Commercial Of The Shelf*
CPU	*Central Processing Unit*
$CSMA-CA$	*Carrier Sense Multiple Acess - Collision Avoidance*
DARPA	*Defense Advanced Research Projects Agency*
DMA	*Direct Memory Access*
DSSS	*Direct Sequence Spread Spectrum*
EP	*End Point*
FFD	*Full Function Device*
FTDI	*Future Technology Devices International*
GTS	*Guaranteed Time Slot*
HWC	*HighWaymon Coordinator*
HWED	*HighWaymon End-Device*
HWG	*HighWaymon Gateway*
HWR	*HighWaymon Router*
I^2C	*Inter-Integrated Circuit*
IC	*Integrated Circuit*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
ISM	*Industrial Scientific and Medical*
ISR	*Interrupt Service Routine*
JSON	*JavaScript Object Notation*
LDO	*Low-DropOut regulator*
LE	*Luminosity Environmental*

LQI	*Link Quality Indicator*
LR − WPAN	*Low Rate Wireless Personal Area Netowrk*
MAC	*Media Access Control*
MCU	*Multipoint Control Unit*
MEMS	*Microelectromechanical Systems*
MISO	*Master Input Slave Output*
MIT	*Massachusetts Institute of Technology*
MOSI	*Master Output Slave Input*
MPPT	*Maximum Power Point Tracking*
NEMS	*Nanoelectromechanical Systems*
NOAA	*National Oceanic and Atmospheric Administration*
NWK	*Network*
OSI	*Open systems Interconnection*
OTA	*Over-The-Air*
PAN	*Personal Area Network*
PE	*Pressure Environmental*
PHY	*Physical Layer of the OSI model*
QPSK	*Quadrature Phase Shift Keying*
RAM	*Random Access Memory*
RFD	*Reduced Function Device*
ROM	*Read Only Memory*
SoC	*System on Chip*
SOSUS	*SOund SUrveillance System*
SPI	*Serial Peripheral Interface*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
THE	*Temperature and Humidity Environmental*
TI	*Texas Instruments*
TS	*Temperature Soil*
UART	*Universal Asynchronous Receiver/Transmitter*
UML	*Unified Modeling Language*
USB	*Printed Circuit Board*
VAX	*Virtual Address Extension*
WLAN	*Wireless Local Area Network*
WPAN	*Wireless Personal Area Network*
WSN	*Wireless Sensor Networks*
XML	*eXtensible Markup Language*

Capítulo 1

Introdução

Os acidentes rodoviários são umas das principais causas de mortalidade em todo o mundo e provocam prejuízos avultados nas viaturas e infraestruturas civis. Todos os dias há acidentes causados pelas mais diversas razões, entre elas, as condições atmosféricas desfavoráveis. Nos países nórdicos cerca de 25% dos acidentes fatais ocorreram em pisos escorregadios ou com gelo [14]. Nos Estados Unidos da América, entre 2008 e 2010, cerca de 900 pessoas morreram devido a estas mesmas razões [15]. No norte de Portugal, em 2008, 28% dos acidentes ocorreram com fatores atmosféricos de chuva ou situações de visibilidade reduzida (nevoeiro) [16].

Têm vindo a ser tomadas medidas de forma a tentar reduzir este tipo de acidentes, ou através de sinais luminosos que avisam os condutores quando existem condições atmosféricas desfavoráveis, ou por sistemas de monitorização colocados nas vias capazes de avisar os responsáveis das mesmas de alterações repentinas do piso que possam originar perigos para os condutores.

Mesmo assim, houve a necessidade de desenvolver sistemas que fossem capazes de prevenir os condutores rodoviários para as possíveis condições adversas existentes, bem como avisar os responsáveis pela manutenção quais as vias mais afetadas. Existe um enorme número de sistemas que já combatem este problema, tais como, o uso de câmaras de trânsito, o uso de variáveis obtidas nos sistemas de informação meteorológica, redes de sensores móveis.

Nesta dissertação foi desenvolvida uma WSN (*Wireless Sensor Network*) [7] para monitorizar as condições atmosféricas nas autoestradas. A WSN desenvolvida (nós e coordenador) nesta dissertação foi também utilizada no concurso europeu *Texas Instruments: Innovation Challenge Europe Analog Design Contest 2014*, onde o trabalho apresentado pela equipa do ESRG foi um dos 20 finalistas. Assim, esta dissertação está dividida em dois grandes temas: as WSNs e as condições atmosféricas. É importante referir que o principal alvo deste projeto é a Rede

Nacional de autoestradas, conhecida simplesmente por Autoestradas [16]. Desta forma, deixa de ser necessário a preocupação com a segurança dos peões, já que estas vias não foram projetadas para eles. Num projeto futuro é possível uma implementação semelhante deste sistema para as restantes estradas.

1.1 Objetivos

O principal objetivo desta dissertação passa por criar um sistema autónomo (a nível energético), de baixo consumo e sem fios, capaz de adquirir informação de variáveis físicas, de forma a monitorizar se as autoestradas constituem algum perigo para os utilizadores das mesmas, em caso de condições adversas.

Para isso, será desenvolvida uma WSN (coordenador e nós sensores) capaz de adquirir a informação necessária, utilizando para isso alguns sensores componentes COTS. Os dados recolhidos devem ser transmitidos para um dispositivo com conexão à internet, de forma a guardar a informação obtida. Para a utilização dos sensores pode ser necessário a implementação de *device drivers* e protocolos de comunicação.

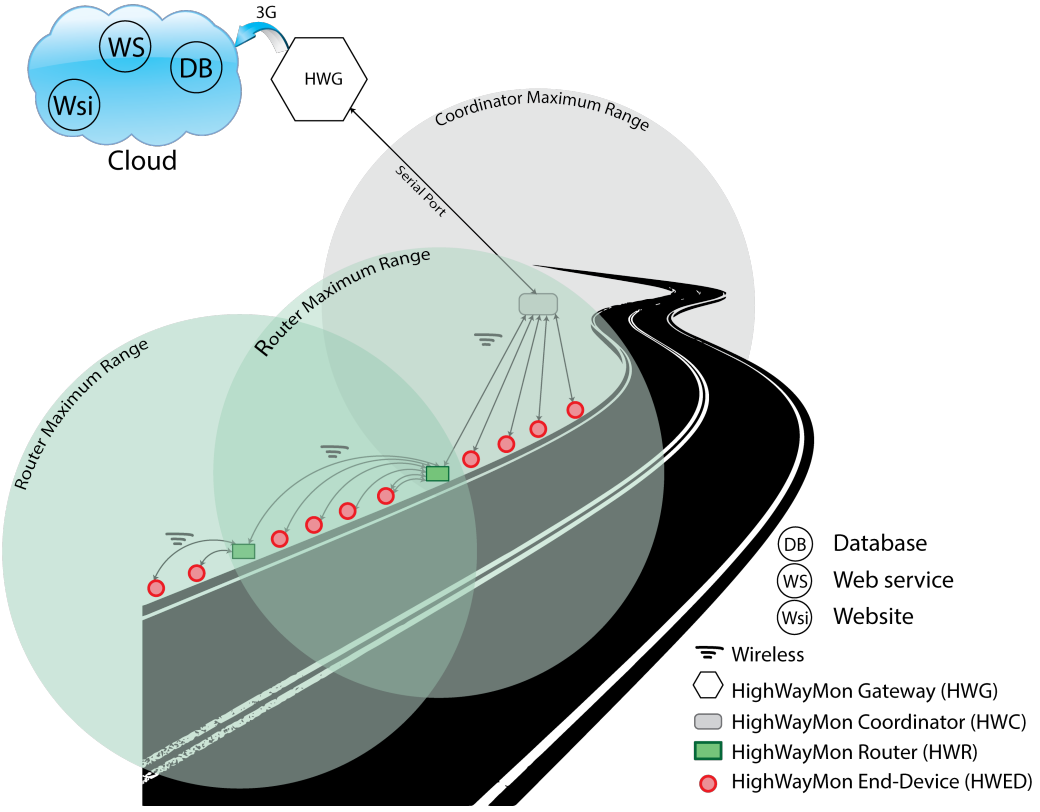


Figura 1.1: *Overview* do sistema.

Outro objetivo passa pela utilização de uma arquitetura modular do *hard-*

ware para que seja possível várias combinações do dispositivo (sensores utilizados, entre outros) sem ser necessário a criação de um dispositivo novo. Junto desta caraterística, inclui-se também a funcionalidade de *plug-and-play* permitindo aos utilizadores apenas montarem os módulos que pretendem sem ser necessário alguma configuração.

Na figura 1.1 é possível observar a visão geral do sistema proposto. Através dos HWEDs (*HighWayMon End-Devices*) é possível adquirir e enviar um conjunto de dados para o HWC (*HighWayMon Coordinator*) de forma a ser possível processá-los. O HWR (*HighWayMon Router*) é utilizado para expandir a rede, através do reencaminhamento de mensagens, tendo incorporada a funcionalidade de *harvesting* para obter uma maior autonomia da bateria. O HWC é responsável pela criação e configuração da rede, que tem como função a receção de todos os dados adquiridos pelos HWEDs e interligação com o HWG (*HighWayMon Gateway*), parte integrante de um sistema já desenvolvido que em conjunto com este projeto compõe uma solução IoT (*Internet of Things*). No capítulo 4 são apresentados os requisitos funcionais e não funcionais do projeto, e também, algumas restrições impostas ao mesmo.

1.2 Estrutura da dissertação

O capítulo 1 tem como propósito fazer uma introdução ao projeto, contextualizando o problema, explicando os objetivos propostos e apresentando a estrutura da dissertação.

No capítulo 2 é apresentado o estado da arte que referencia um pouco da história das WSNs, apresenta um conjunto de plataformas existentes no mercado e resume algumas das aplicações já existentes inseridas neste tema.

O capítulo 3 apresenta os principais conceitos teóricos das WSNs, focando as suas caraterísticas, unidades e a tecnologia *ZigBee*.

O capítulo 4 descreve a arquitetura, requisitos e funcionalidades do sistema. São apresentadas as caraterísticas de cada módulo, bem como todos os componentes necessários para o funcionamento do sistema. É feita também uma descrição da *Z-Stack* e do ambiente de desenvolvimento *IAR Embedded Workbench*.

No capítulo 5 são apresentados todos os passos efetuados na implementação do projeto, seguindo as especificações do capítulo anterior. É dividido em três partes, começando com a apresentação do *hardware* desenvolvido, em seguida por uma seção de UML que descreve as funcionalidades dos dispositivos e, por fim, as funcionalidades implementadas por *software*.

O capítulo 6 apresenta uma avaliação da implementação através dos resultados obtidos. É especificado o ambiente de testes e de seguida é explicado os testes mais relevantes.

Concluindo, o capítulo 7 apresenta as principais conclusões da dissertação, bem como o possível trabalho que se pode fazer futuramente.

Capítulo 2

Estado da Arte

Este capítulo tem como objetivo apresentar uma visão geral sobre o início da era das WSNs e de que forma foram evoluindo. São apresentados vários exemplos de plataformas criadas para facilitar o desenvolvimento de novas soluções, de forma a ser possível a redução do *time-to-market*. São ainda descritas algumas aplicações que de alguma forma incutiram interesse na forma como foram desenvolvidas e nas dificuldades que tiveram de ultrapassar para o sucesso do projeto.

2.1 História das WSNs

A história relativa às redes de sensores pode ser dividida em quatro fases. Em seguida é apresentada uma pequena descrição de cada uma delas [1].

2.1.1 Durante a guerra fria (década de 50)

As aplicações de defesa tem sido uma das principais razões para a investigação e desenvolvimento das redes de sensores, tal como muitas outras tecnologias. Durante a guerra fria foi criado um sistema de sensorização acústica denominado por SOSUS (*SOund SUrveillance System*) [17]. De forma a detetar submarinos soviéticos foram distribuídos, pelo fundo do oceano, um conjunto de sensores acústicos em localizações estratégicas. No entanto, com o passar do tempo outros sistemas de redes acústicas foram desenvolvidos para vigilância de submarinos.

Atualmente o SOSUS é usado pela administração nacional oceanográfica e atmosférica (NOAA) para monitorizar atividades no oceano, tais como, atividade animal e sísmica.

2.1.2 Início dos projetos avançados de defesa (década de 80)

Por volta de 1980 houve um grande desenvolvimento da investigação sobre redes de sensores através de programas patrocinados pela *Defence Advanced Research Projects Agency* (DARPA). O *Distributed Sensor Networks* (DSN) foi um desses programas e tinha como objetivo determinar se o recém criado protocolo TCP-IP poderia em conjunto com a *ARPAnet* (antecessor da internet) ser usado no contexto de redes de sensores. O DSN tinha como requisitos uma arquitetura de baixo custo, distribuída, autónoma e cooperativa em que a rede enviava a informação para o nó que melhor as podia utilizar. Este programa focava-se no rastreamento, processamento de sinal e computação distribuída. Eram utilizados sensores acústicos, protocolos de comunicação de alto nível, processamento e cálculos de algoritmos.

Houve duas instituições que estiveram envolvidas neste programa. Os investigadores da Universidade de *Carnegie Mellon* ficaram responsáveis pelo desenvolvimento de um sistema operativo que tivesse como caraterísticas uma rede transparente, permitisse a reconfiguração do sistema e *rebinding* (conetar-se novamente), criando desta forma o *Accent* [18]. Este sistema foi demonstrado num teste interior com fontes de sinais, sensores acústicos e computadores VAX conetados por *Ethernet*. Posteriormente este sistema evoluiu para o *Mach* [19]. Os investigadores do *Massachusetts Institute of Technology* (MIT) ficaram responsáveis pela técnica de processamento de sinal desenvolvendo o *Signal Processing Language and Interative Computing Environment* (SPLICE). Este sistema fazia a análise e desenvolvia algoritmos para redes de sensores distribuídos (DSNs).

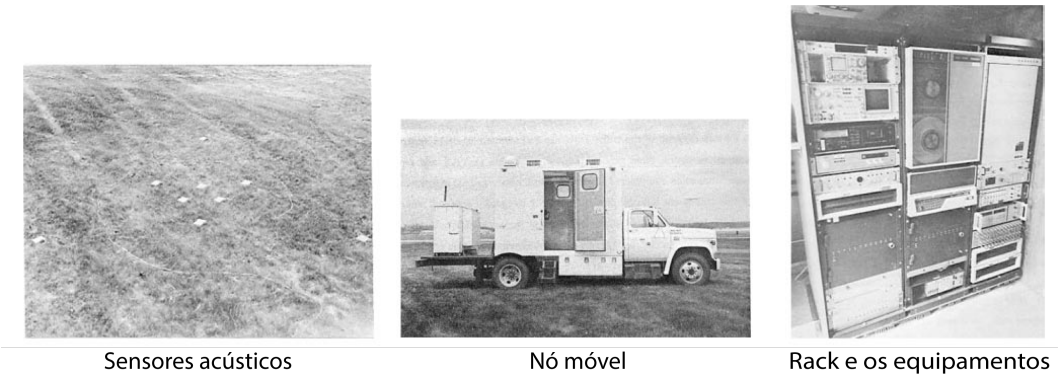


Figura 2.1: Componentes do teste do DSN por volta de 1985 [1].

Para demonstrar os resultados que tinham sido alcançados no projeto DSN, foi criado um teste que tinha como objetivo a monitorização acústica de aeronaves.

Todo o material utilizado foi personalizado daí o tamanho e peso associado ao mesmo. Foram usados conjuntos de sensores acústicos (nove microfones), um computador *PDP11/34* e vários processadores para processar os sinais acústicos. Na figura 2.1 é possível ver o equipamento utilizado e o tamanho dos dispositivos envolvidos.

Também por volta de 1980, o *Advanced Decision Systems* (ADS) [20] criou um algoritmo de procura de várias hipóteses para lidar com situações difíceis que envolviam a densidade do nó destino, falha nas deteções e falsos alarmes. Atualmente, este algoritmo é um *standard* para situações de problemas de procura.

2.1.3 Aplicações militares desenvolvidas entre 1980 e 1990

Na década de 80 os investigadores já imaginavam sensores de tamanho muito reduzido, no entanto, a tecnologia existente não estava pronta para suportar este tipo de desenvolvimento. Apesar disso, houve a necessidade de desenvolver componentes COTS para diminuir o tempo de desenvolvimento e os custos.

Na área militar já se via as grandes vantagens que as redes de sensores poderiam adicionar aos seus sistemas, daí que se tenha tornado uma componente muito importante no desenvolvimento de dispositivos para ambientes de guerra. Em ambiente de guerra o sistema de armas não continha necessariamente uma plataforma específica. Em vez disso, através do uso de sensores distribuídos, o sistema da arma e a plataforma colaboravam entre si e enviavam a informação para o nó apropriado. Desta forma, foi possível melhorar a deteção de tropas inimigas através de várias observações, geometria de diversidade fenomenológica, com raio de deteção superior e tempos de resposta mais rápidos.

A *Cooperative Engagement Capability* (CEC), desenvolvida pela marinha dos Estados Unidos da América, é um exemplo deste tipo de sistemas. Este consistia num conjunto de radares que detetavam alvos aéreos.

Outros exemplos de redes de sensores para ambientes de guerra incluem sensores acústicos para deteção de submarinos tais como o FDS e o *Advanced Deployable System* (ADS), e sistemas de sensores autónomos terrestres (UGS), [21] tais como, o *Tactical Remote Sensor System* (TRSS), o *Advanced Remote Ground Unattended Sensor* (ARGUS) e o *Remote Battlefield Acoustic and Seismic System* (REMBASS) [22].

2.1.4 Desenvolvimento de redes de sensores no século XXI

Durante anos muitos investigadores visionaram chegar ao que é possível obter atualmente sobre redes de sensores. Os avanços na comunicação e computação permitiram obter sensores pequenos e de baixo custo baseados em *microeletromechanical system* (MEMS) e futuramente *nanoeletromechanical system* (NEMS), redes sem fios e microprocessadores de baixo consumo e custo.

Devido à produção em larga escala a tecnologia necessária para o desenvolvimento de redes de sensores está comercialmente disponível e tende a ficar cada vez mais acessível.

A tabela 2.1 representa a evolução dos nós sensores, especificando algumas das suas caraterísticas.

Tabela 2.1: Evolução dos nós sensores ao longo de três gerações [1].

	Primeira Geração (1980 - 1990)	**Segunda Geração (2000-2005)**	**Terceira Geração (2005 - 2014)**
Tamanho	Caixa de sapatos ou maior	Baralho de cartas	Partícula de pó ou menor
Peso	Quilogramas	Gramas	Insignificante
Arquitetura do nó	Sensorização, comunicação e processamento separados	Sensorização, comunicação e processamento integrada	Sensorização, comunicação e processamento completamente integrada
Topologia	Ponto-a-ponto ou estrela	Servidor-cliente ou ponto-a-ponto	Ponto-a-ponto
Alimentação	Baterias grandes	Baterias do tipo AA	Solar, eólica ou outras baseadas em nanotecnologia
Durabilidade	Horas, dias ou mais	Dias até semanas	Meses até anos
Colocação do nó	Instalado manualmente ou "largado" pelo ar	Instalado manualmente	Embebido

2.2 Plataformas de Nós Sensores

Ao longo dos últimos anos tem havido um crescimento na investigação sobre o desenvolvimento de plataformas de redes de sensores. A maioria destas plataformas são usadas para investigar problemas de rede e capacidades de sensorização ou escalabilidade. No entanto, apenas algumas delas foram avaliadas para serem usadas em aplicações de larga escala.

Os maiores problemas de desenvolver os nós sensores são: (a) o custo (b) o tempo de desenvolvimento elevado e (c) a complexidade em escalar uma rede de sensores para centenas ou até milhares de nós.

Nesta secção serão discutidas algumas plataformas genéricas usando componentes COTS.

2.2.1 Nós Sensores Genéricos

Por nós genéricos entende-se nós que têm como objetivo incluir-se numa categoria geral de um nó sensor sem nenhuma aplicação específica. Foram avaliados alguns nós que se enquadram nesta categoria genérica e foram descritas as suas principais caraterísticas.

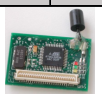
UC Berkeley Motes

A universidade de *Berkeley* foi uma das pioneiras na investigação e desenvolvimento de plataformas de nós sensores. O primeiro nó de sensores Motes surgiu através do projeto de tese de *Seth Hollar* em 2000, conhecido por *Macro Motes* ou *Cots Motes*. Posteriormente a esta versão original surgiram algumas expansões: *Wec Motes* e *RF Motes*. Tinham como caraterísticas o rádio *TR1000* e um processador *Atmel AT90LS8535*. Apesar do *RF Mote* ter um consumo de energia bastante pequeno é incapaz de obter distâncias de comunicação elevadas, ainda que tivesse capacidade para as atingir. Por sua vez, o *AT90LS8535* não consegue escrever na memória de programação, e por isso, é necessário um coprocessador para tratar apenas de reprogramação.

A geração seguinte é composta pelos nós *Rene* e *Rene2* que apresentam um desenho modular, o que permite a fácil integração de placas extras (p.e. placas de sensores). Nestes nós estão integrados o processador *ATMega163* e comparativamente à geração anterior foi aumentada a memória para 1 Kb e a sua flash para 16 Kb.

Os sucessores dos nós anteriores foram o *Mica* e o *Mica2* motes que apesar de

Tabela 2.2: Comparação entre alguns nós genéricos [13].

Mote type		WeC	Rene	Rene2	Mica	Mica2	Mica2Dot
Figura exemplo							
MCU	Chip	AT90LS8535	ATmega163L		ATmega103L	ATmega128L	
	Tipo	4 MHz, 8 bit	4 MHz, 8 bit		4 MHz, 8 bit	8 MHz, 8 bit	
	Memória programação (Kb)	8	16		128	128	
	RAM (Kb)	0.5	1		4	4	
Memória externa	Chip	24LC256			AT45DB014B		
	Tipo conexão	I^2C			SPI		
	Tamanho (Kb)	32			512		
Alimentação	Tipo	Pilha de moeda	2xAA				Pilha de moeda
	Capacidade	575	2850				1000
RF	Chip	TR1000				CC1000	
	Frequência	868/916MHz				868/916MHz, 433 ou 315 MHz	
	Taxa transmissão (Kbps)	10			40	38.4	

continuarem a usar a arquitetura modular tiveram algumas melhorias. Passaram a ter um processador *Atmel ATMega* 103 e um *ATMega128l*, respetivamente. Com o uso do *ATMega128l* deixa de ser necessário o uso de um coprocessador para escrever na memória do programa.

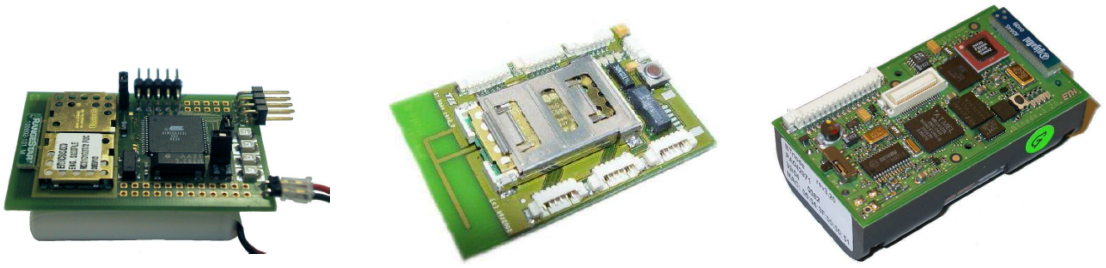
Muitas empresas usaram, e ainda usam, estes nós sensores para o desenvolvimento das suas WSNs. Na tabela 2.2 é possível ver algumas das diferenças entre as várias gerações de *Berkeley Motes*.

Telos

O nó *Telos* combina o processador *MSP420* da *Texas Instruments* [23] com o *Chipcon CC2420*. Este nó não continua com a arquitetura modular dos *Mica Motes* mas tem uma porta USB para facilitar na programação/*debug*, o que é uma vantagem na fase de desenvolvimento.

BTNode

Várias gerações de *BTNodes* [24] têm sido desenvolvidas pela *ETH Zurich* no projeto *Smart-Its* [25], como é possível ver na figura 2.2.



(a) BTNode1. **(b)** BTNode2. **(c)** BTNode3.

Figura 2.2: Gerações de *BTNodes* [2].

O *BTNode2* estava equipado com um processador *ATMega128l* e um módulo *Bluetooth* da *Ericsson ROK* 101 007. Continha ainda uma memória RAM externa de 60 Kb e um indicador de bateria carregada.

O *BTNode3* foi desenvolvido também pela ETH mas foi vendido comercialmente pela *Art of Technology* [26]. Contém um microprocessador *Atmel ATMega128l* com 244 Kb de memória RAM externa e tem dois rádios: o *Chipcon CC1000* e o *Zeevo ZV4002*. Esta nova geração tem uma estrutura modular de forma a ser possível a conexão de placas adicionais.

Família *Medusa*

Os nós da família Medusa têm a particularidade de usarem dois microprocessadores na sua arquitetura. As suas capacidades de computação foram divididas em duas categorias: computação de baixa ou alta frequência [3]. Para o primeiro caso é usado o microprocessador de 8 *bits Atmel ATMega128l*, que é responsável por processamentos "mais simples" e pelo funcionamento normal do nó. No segundo caso é usado um processador de 32 *bits*, o *AT91FR4081*, que é responsável por cálculos que requerem uma maior complexidade, e que um processador de 8 *bits* não seria capaz de as realizar com o desempenho necessário. Para além disso, este processador tem recursos para suportar diferentes sistemas operativos como o *Red Hat eCos* [27] e *uCLinux* [28].

Relativamente à comunicação, esta plataforma utiliza o já falado *TR1000* da *RF Monolithics*. Na figura 2.3 é possível observar o nó sensor *Medusa MK-2*.

Figura 2.3: Plataforma *Medusa MK-2* [3].

Eko Pro Series

A projeto *Eko Pro Series* foi desenvolvido pela empresa *Crossbow* na área da monitorização ambiental, mais concretamente para a monitorização de *crops*, microclimas e investigação ambiental. A rede em malha, baseada na tecnologia *XMesh* da *Crossbow* permite que todos os nós possam reenviar dados para outros nós, sendo que os nós que não têm funcionalidades de sensorização podem funcionar como repetidores.

Todos os nós monitorizam constantemente a vizinhança para cálculo de novas rotas alternativas. Desta forma, caso haja alguma rota bloqueada, irá usar outra diferente. Com esta topologia em malha existem duas grandes vantagens: é possível um maior alcance da rede (visto poderem retransmitir os dados com outros nós) e existe uma maior confiabilidade.

O nó *eKo eN2100*, representado na figura 2.4, integra uma placa com um processador e rádio da família *Crossbow IRIS*. A alimentação é feita através de baterias recarregáveis e um painel solar. Na tabela 2.3 é possível observar algumas das principais caraterísticas deste nó.

A este nó, podem-se ligar até quatro sensores compatíveis, que podem ser os já disponibilizados ou customizados. Os autores apostaram numa arquitetura *plug-and-play*, permitindo desta forma, o uso de sensores alternativos conforme necessário para o cliente. O nó ao fazer uma leitura no porto de sensorização irá identificar os sensores ligados após um *reset*. Na tabela 2.4 é possível observar os sensores já desenvolvidos pela empresa bem como algumas das suas caraterísticas.

Tabela 2.3: Caraterísticas do nó *eN2100*.

Nó eKo	**EN2100**	**EN2120**
Sensores		
Número sensores	4 sensores por nó	
Tempo de medição	Em cada 15 minutos	
Rádio		
Frequência	2.405 a 2.480 GHz	
Canais	16 canais	
Tecnologia sem fios	DSSS, IEEE 802.15.4	
Potência de saída	+3 dBm	+18 dBm para EUA; +10 dBm para Europa
Sensibilidade	-101 dBm	
Alcance	152 a 457 m	609 m a 3.2 Km
Antena	Dipolo, interna	
Alimentação		
Corrente	0.4 mA por cada 15 min (sem sensores)	0.5 mA por cada 15 min (sem sensores)
Painel solar	3.3 cm x 6.25 cm	
Bateria	3AA NiMh recarregáveis	
Duração	3 meses sem luz solar; Mais que 5 anos em campo	

Figura 2.4: Plataforma *Eko eN2100* [4].

Conclusão

Existe uma gama muito extensa de família de nós sensores quando se usa componentes COTS. O número de combinações que é possível fazer só com mi-

Tabela 2.4: Caraterísticas dos sensores da *Crossbow*.

Tipo	eS1101 - Temperatura e Humidade do Solo	eS1201 - Temperatura e Humidade do Ar
Fabricante	Humidade solo: *Watermark* Temperatura solo: *Davis*	*Sensirion SHT75*
Gama	Potencial hídrico: 0 aos 200cbar Temperatura: -40 até 70ºC	Humidade: 0 aos 100 %RH Temperatura: -40 até 70ºC
Precisão	+/- 5%	Humidade: +/- 3% Temperatura: +/- 2ºC
Tamanho cabo	4.8 m	6 m
Peso	272 g	226 g

croprocessadores e transmissores rádios é enorme. Para além dos nós referidos ao longo desta secção existem ainda os *MicaZ* desenvolvidos pela *Crossbow*, o projeto *Eyes*, os *Intel IMote*, entre outros.

Cada um dos nós apresentados foi fruto de uma pesquisa rigorosa do que existia no mercado para que se pudesse atingir os objetivos pretendidos. No momento da escolha do *hardware*, o *software* fica limitado pelas escolhas iniciais.

Nós genéricos são ideais para fins educacionais (principalmente os que apresentam arquitetura modular) e para empresas que não queiram ter os custos de desenvolvimento iniciais.

2.3 Aplicações de Sistemas de Monitorização

Nesta secção são apresentados alguns exemplos de aplicações desenvolvidas ao longo dos anos, e que utilizam WSNs para adquirir a informação necessária. Serão abordados vários tipos de aplicações, desde monitorização de habitats a monitorização rodoviária.

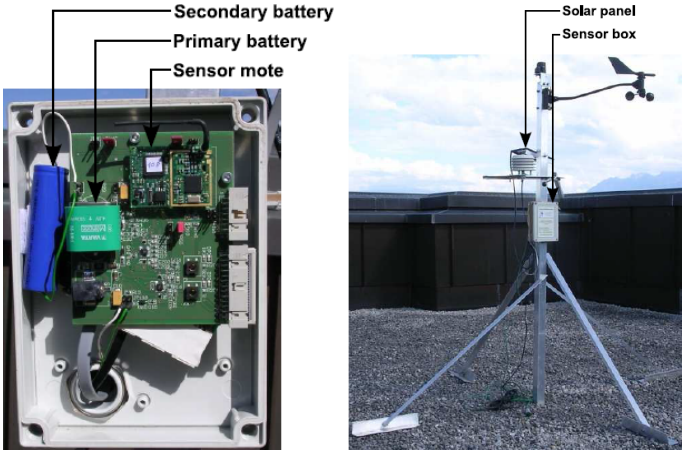
2.3.1 *SensorScope*

O projeto *SensorScope* [5] foi desenvolvido na *EPFL*, Suíça. Nas várias vertentes do *SensorScope* foi usada uma plataforma *Shockfish TinyNode* que é composta por um microcontrolador de 16 *bits*, o *MSP430*, e um transmissor rádio *Semtech XE1205*, que opera na banda dos 868 MHz e consegue obter taxas de transferência até 76 Kbps. O nó contém ainda 48 Kb de ROM, 10 Kb de RAM e

512 Kb de flash [5]. A plataforma escolhida pelos autores tem como vantagens o alcance (superior a 200m no exterior) e o baixo consumo de energia.

A alimentação deste sistema é feita recorrendo a um painel solar com dimensões de 162x140 mm e duas baterias: uma primária, do tipo *NiMH* com capacidade de 150 mAh e outra secundária, da tecnologia *Li-Ion*, com capacidade de 2200 mAh.

Para a estação de sensorização foi criada uma estrutura de alumínio com quatro pernas, onde o painel solar e os sensores estava fixados. A plataforma *TinyNode* estava fixada nesta estação inserida numa caixa isolada como é possível observar na figura 2.5.



(a) Caixa isolada. **(b)** Estação.

Figura 2.5: Componentes do *SensorScope* [5].

Relativamente à sensorização, a estação é capaz de acomodar até 7 sensores diferentes, dos quais alguns conseguem medir mais que uma variável física. No total, é possível medir 9 variáveis ambientais: temperatura e humidade do ar, temperatura à superfície, radiação solar, direção e velocidade do vento, precipitação, teor de água no solo e sução de água no solo [5].

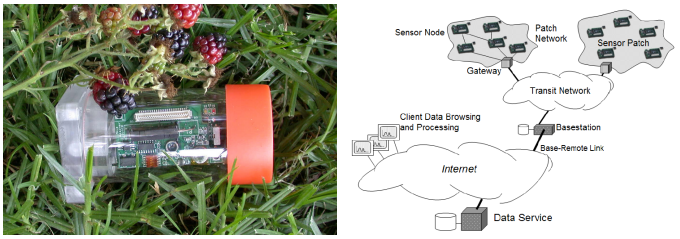
As estações de sensorização transmitiam a informação para uma estação base (*sink*), que por sua vez, através de um *gateway*, guardava a informação num servidor. É possível usar três tecnologias diferentes para conetar a estação base com o servidor: *GPRS*, *Wi-Fi* e *Ethernet*. A informação é então disponibilizada num interface gráfico baseado no *Google Maps* e num *website Microsoft's SensorMap*.

Foram desenvolvidos vários testes, sendo um deles realizado num glaciar localizado a 2500m de altitude em *Le Génépi*, Suíça. Este teste consistia no uso de 16 estações de sensores que tinham como objetivo adquirir a precipitação do local

e correlacionar esses dados com a temperatura e o vento, baseando-se na forma do terreno.

2.3.2 *Great Duck Island*

Durante o ano de 2002, os investigadores da Universidade de *Berkeley* e o Colégio do Atlântico desenvolveram a primeira WSN, com propósito de monitorização de *habitats*, numa ilha na costa de *Maine*. O principal objetivo era monitorizar o *habitat* de pequenas aves marinhas (*Leach's Storm Petrel*), nos primeiros sete meses após o seu nascimento, recorrendo para isso a 32 nós sensores. A WSN tinha como objetivo monitorizar de que forma estas aves usavam a sua toca e também o microclima do mesmo. Devido à grande sensibilidade destas aves, as WSNs são uma boa alternativa às frequentes visitas ao local para fazer as medições.



(a) Nó sensor. **(b)** Arquitetura da rede.

Figura 2.6: *Great Duck Island.* Nó sensor e arquitetura da rede [6].

Os nós sensores estão equipados com *Mica Motes*, desenvolvidos na Universidade de *Berkeley* (ver seção 2.2.1), equipados com uma placa de sensores ambientais capazes de adquirir informações relativas às seguintes variáveis: temperatura, luminosidade, pressão barométrica, humidade e movimento através de um sensor de infravermelho passivo. De forma a resistir às condições mais extremas, os nós sensores estão cobertos com uma capa fina capaz de proteger os contatos elétricos da água. No entanto, os sensores continuaram expostos de forma a manter a sua sensibilidade. Por fim, os nós foram inseridos numa estrutura acrílica para aumentar a sua robustez, como é possível observar na figura 2.6a.

Os nós sensores foram espalhados numa área de $60690m^2$ e os dados foram enviados para um servidor. Foi desenvolvida uma arquitetura em camada (*tiered architeture*) através da criação de grupos de nós sensores que transmitem a informação recolhida para um *gateway* do sensor *patch* respetivo. O *gateway* é parte de uma *transit network* que transmite a informação para uma estação base. Na figura 2.6b é possível observar a arquitetura da rede.

Junto com a informação recolhida dos sensores é enviado também o nível de bateria do nó respetivo, de forma a ter alguns cuidados relativos aos mesmos.

Apesar da duração do teste ter sido planeada para sete meses, muitos dos nós avariaram mais cedo. Isto deve-se a várias causas, entre elas: baterias esgotadas, desgaste dos nós e chuva. O desempenho da rede teve também algumas surpresas: devido à baixa taxa de transmissão era de prever que existissem muito poucas colisões, no entanto, observou-se que por alguma razão as tramas começaram a colidir [2].

2.3.3 *Zebranet*

Em Janeiro de 2004, iniciou-se o projeto *Zebranet* em que era pretendido a monitorização de zebras no seu *habitat* natural, nas planícies do *Kenya*. O principal objetivo era monitorizar a posição das Zebras, utilizando a tecnologia GPS, num período de um ano, sendo para isso usadas coleiras equipadas com um nó sensor.

Existem cerca de 35000 zebras nos 40000 Km de planícies de *Laikipia*, *Kenya* Central. Estas juntam-se em rebanhos pequenos ou grandes dependendo da sua espécie. Então, de forma a reduzir o número de coleiras a utilizar, apenas é necessário equipar as mesmas em alguns animais por grupo, visto que as velocidades e direções costumam ser semelhantes.

A localização dos animais é feita através de transmissores VHF que transmitem "*pings*". Para os receber é necessário um meio móvel terrestre ou aéreo que esteja a "ouvir" estes sinais.

O projeto *Zebranet* tem um cenário completamente diferente do falado anteriormente (*Great Duck Island*): (*a*) os nós sensores são móveis (*b*) a estação base é móvel (muda conforme o acampamento se move) (*c*) os nós sensores não estão em contato com a estação base, ou com a rede, durante todo o tempo. Devido ao risco de vandalismo não é possível ter estruturas fixas, por isso, foi necessário encontrar um sítio que fosse mais visitado pelos animais. O autor observou que os rebanhos costumavam encontrar-se em sítios com água. Através desta informação optou-se por uma topologia *peer-to-peer* com uma estratégia de *data dissemination*, isto é, os dados são reencaminhados de nó para nó quando estão dentro do raio de comunicação, e para a estação base quando algum desses nós estão dentro do raio.

O *Zebranet* sofreu algumas alterações desde o seu protótipo (versão 0.1 [29]) até às gerações seguintes alimentadas a energia solar (versões 1, 2 e 3 [30]). A plataforma inicial utilizava dois rádios, um para distâncias curtas e outro para longas. No entanto, esta caraterística não foi avante optando-se por utilizar apenas

um dispositivo rádio.

A versão 3 do *Zebranet* tem uma arquitetura um pouco diferente. Utiliza um microprocessador da TI, o *MSP430*, e o rádio adotado é o *MaxStream 9xStream*[2]. Devido aos altos consumos por parte do GPS e do rádio (comparativamente ao normalmente utilizado) é utilizada energia solar para carregar as baterias, e desta forma, ter uma maior duração do sistema.

No final 12 zebras utilizaram a coleira, funcionando de maneira autónoma nas planícies do *Kenya*.

2.3.4 *Traffic Pulse Technology*

Traffic Pulse Technology é um projeto desenvolvido pela *Traffic.com* [31] que tem como objetivo adquirir informações sobre variáveis físicas, através de uma rede de sensores, e posteriormente, processá-las e guardá-las num servidor, distribuindo as mesmas em diversas aplicações. Foi desenvolvido a pensar em ambientes abertos e com a possibilidade de adquirir informação em tempo real, permitindo desta forma monitorizar algumas mudanças nas variáveis, tais como temperatura e níveis de poluição, o mais rápido possível. De 2000 a 2013 este sistema esteve instalado ao longo de um grande número de autoestradas, englobando os principais estados dos EUA.

Figura 2.7: Nó sensor típico numa autoestrada [7].

A rede de sensores é responsável por adquirir velocidades, ocupação das faixas e pela contagem de veículos. Através desta informação é possível calcular médias de velocidade e tempos de viagem. Esta informação é transmitida para uma estação base a cada 60 segundos. Cada estação base tem acesso a informações extra (recolhidas através de câmaras, aviões e unidades móveis), tais como locais de

construção e acidentes, que em conjunto com a informação sensorial recolhida são uma mais valia para os utilizadores das aplicações disponibilizadas [7]. Na figura 2.7 é possível ver um nó tipico numa autoestrada.

Ao longo destes anos, a *Traffic Pulse* foi vendida a várias empresas diferentes. Em 7 de Março de 2007 foi vendida à *Navteq*, quase meio ano depois de ser anunciado [32]. Aproximadamente 1 ano depois, a 10 de Julho de 2008, a *Nokia* comprou a *Navteq*. Atualmente, o endereço *Traffic.com* foi mudado para *Here.com*, funcionando nos moldes anteriores, fornecendo informação sobre o tráfico nas maiores cidades.

2.3.5 *Barriera Attiva*

O projeto *Barriera Attiva* foca o aspeto ambiental das estradas, tendo como objetivo desenvolver e implementar um sistema ativo sem fios utilizando barreiras das auto estradas (WAGS).

As barreiras podem ser encontradas ao longo de várias autoestradas, e em algumas cidades, atuando de forma passiva na segurança das estradas evitando que os carros saiam fora da estrada em caso de colisão. Um dos principais objetivos é tornar estas barreiras num elemento ativo na segurança rodoviária, através da prevenção de colisões entre os carros e as barreiras e através de monitorização ambiental.

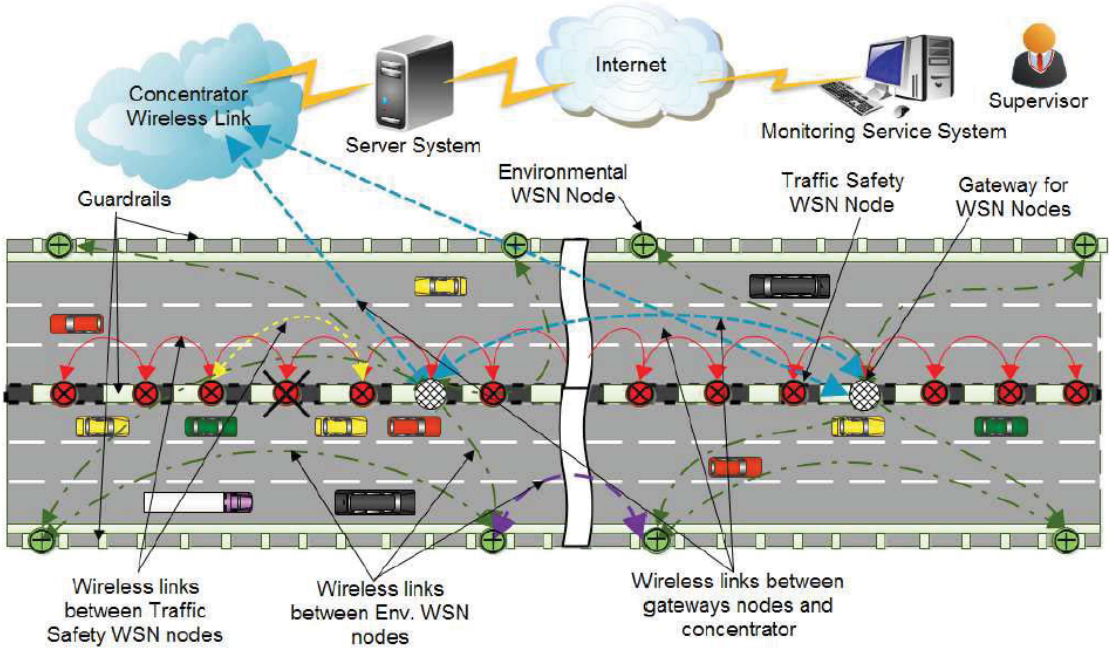


Figura 2.8: Arquitetura do sistema *Barriera Attiva* [8].

São utilizados três tipos de nós diferentes: (*a*) nós para monitorização ambiental, (*b*) nós para segurança rodoviária e (*c*) nós *gateway* como é possível observar na figura 2.8.

O nó sensor foi implementado usando uma plataforma *IRIS Mote* que utiliza um processador e rádio da *Atmel*, o *ATMega1281* e o *RF230*, respetivamente [33]. O subsistema de alimentação é feito através de uma bateria que é carregada usando energia solar.

Em relação ao sensores utilizados, o sistema é capaz de adquirir valores relativos às seguintes variáveis: monóxido de carbono (CO), dióxido de nitrogénio (NO2), dióxido de enxofre (SO2), material particulado, temperatura, humidade, luminosidade, som e vento.

Figura 2.9: Nó sensor aplicado numa barreira [8].

Para assegurar a escalabilidade no sistema, adotou-se uma estrutura flexível e modular para os nós sensores, sendo então possível usá-los em qualquer sítio que seja necessário sem ser preciso alterar o nó.

2.4 Conclusão

Existe um enorme número de aplicações envolvendo WSNs. Neste capítulo foram especificados cinco exemplos de redes de sensores capazes de monitorizar uma, ou várias, variáveis físicas, focando a arquitetura da rede e de que forma a informação é distribuída. Existem outras aplicações, com objetivos diferentes, bastante interessantes, e por isso mesmo serão referenciadas aqui: *RED-WINE* [34], *Cougar* [35], *Calamari* [36], *SECOAS* [37], Monitorização de um vulcão [38], entre outros.

O próximo capítulo é dedicado às principais caraterísticas que se deve ter em consideração no desenvolvimento de uma WSN. Com isto espera-se que os leitores possam compreender os principais conceitos em torno deste tema.

Capítulo 3

Wireless Sensor Networks

Neste capítulo será feita uma pequena introdução às WSNs, serão explicadas as suas principais caraterísticas, os componentes que compõem cada nó e as tecnologias sem fios presentes numa WSN, sendo aprofundada a tecnologia *ZigBee*.

3.1 Introdução

As WSNs têm como principal objetivo a aquisição de informações sobre o meio que as envolve. Estas existem normalmente para que seja possível obter informação sobre determinadas áreas, e desta forma, ser possível uma gestão das mesmas. Os dados recolhidos são processados e enviados para uma estação base, sendo normalmente disponibilizados numa plataforma informática.

Estas WSNs são constituídas por nós nós que incorporam diferentes componentes: sensorização, processamento, comunicação e alimentação. Estes nós cooperam entre si para que os dados recolhidos cheguem à estação base, e assim, as informações sejam processadas. Numa WSN podem existir centenas ou até milhares de nós. Alguns nós podem ter, para além da unidade de sensorização, uma unidade de atuação para que seja possível intervir sobre algumas circunstâncias.

Estes nós podem ser colocados aleatoriamente no meio que se quer monitorizar. No entanto para zonas de difícil acesso, em que até pode ser necessário ajuda aérea para a colocação dos mesmos, deve estar garantido o funcionamento do dispositivo por longos períodos de tempo. Para isso, o consumo de energia deve ser o mais otimizado possível garantindo, desta forma, uma maior durabilidade das baterias. Sendo a unidade de comunicação a que tem um maior consumo de energia, deve ser escolhido o melhor protocolo de encaminhamento para a rede sem fios que se quer desenvolver.

Existe um vasto número de aplicações possíveis em que podem ser usadas WSNs. Isto deve-se ao facto da existência de uma variedade significativa de sensores capazes de detetar variáveis físicas atualmente, tais como: temperatura, humidade, luminosidade, pressão atmosférica, som, aceleração, campos magnéticos, entre outros. Estas aplicações podem ser encontradas nas seguintes áreas: militar, transporte, saúde, monitorização de uma área, sensorização ambiental, monitorização estrutural, monitorização industrial e agricultura.

3.2 Caraterísticas de uma *WSN*

Existem vários desafios aquando da criação de uma WSN comparativamente às redes *ad hoc*. Visto que estas comunicam de forma *wireless* e sem infraestruturas, são exigidas certas caraterísticas para que a eficiência e o desempenho da rede seja o desejado, tais como [9]:

- **Tolerância a falhas:** visto que estes sistemas podem ser instalados em locais perigosos ou até inóspitos, deve-se garantir que o sistema é capaz de encontrar falhas e ser robusto o suficiente para que seja possível manter o sistema a funcionar mesmo havendo um grande número de falhas por parte dos nós. Para o protocolo de *routing* esta caraterística é crucial, pois, ele deve ser capaz de encontrar caminhos alternativos para reencaminhar os dados a serem transmitidos.
- **Escalabilidade:** as WSNs podem ter desde dezenas de nós até centenas de milhares. Por isso, é necessário que o desempenho da rede seja o melhor possível, mesmo que existam milhares de vizinhos no raio de comunicação de um nó.
- **Custos de produção:** o preço de cada nó é importante visto que estes dispositivos são, normalmente, descartáveis. Isto significa que quanto mais barato for o custo de produção de um nó, mais sucesso poderá ter no mercado.
- **Requisitos de hardware:** um nó sensor necessita sempre de pelo menos um sensor, um processador, uma unidade de comunicação e de alimentação. É possível adicionar mais funcionalidades ao nó, tais como sistema de localização ou um maior número de sensores, no entanto deve-se ter em conta que quantas mais funcionalidades forem adicionadas ao dispositivo maior será o consumo de energia e o custo de produção do mesmo.
- **Topologia:** apesar de as WSNs estarem a evoluir constantemente continuam a ter algumas restrições em termos de alimentação, memória, processamento e comunicação. Como um dos principais objetivos de uma WSN é um baixo

consumo de energia, uma topologia bem desenvolvida é uma mais valia para que o consumo de energia possa ser menor.

- **Meio de transmissão:** normalmente as WSNs utilizam comunicação rádio, no entanto, alguns sistemas adotam comunicação ótica ou laser de forma a tornar o sistema mais robusto e tentar evitar interferências.
- **Consumo de energia:** dado que alguns dispositivos podem estar em lugares de difícil acesso, pode ser impossível a troca de bateria. Por isso, são necessários cuidados no desenvolvimento tanto do *software* como do *hardware*. Dado que a comunicação rádio tem um elevado consumo de energia, comparativamente às outras unidades, pode ser possível em alguns casos a compressão dos dados de forma a poupar energia neste recurso. No entanto, pode ser usada mais energia para processar os dados ou filtrar os mesmos. Nesta caraterística deve ser encontrada a melhor forma de ter um sistema de baixo consumo tendo todas as funcionalidades necessárias.

3.3 Componentes de um nó sensor

Um nó sensor é composto por quatro componentes: comunicação, processamento, alimentação e sensorização como é possível ver na figura 3.1. Para além destas, pode-se também encontrar outros componentes, tais como gestão de energia que pode recorrer, por exemplo, a circuitos de carregadores solares ou piezoelétricos, ou sistemas de localização para ajudar a obter a posição do dispositivo. Em seguida serão apresentadas algumas informações referentes aos quatro principais componentes.

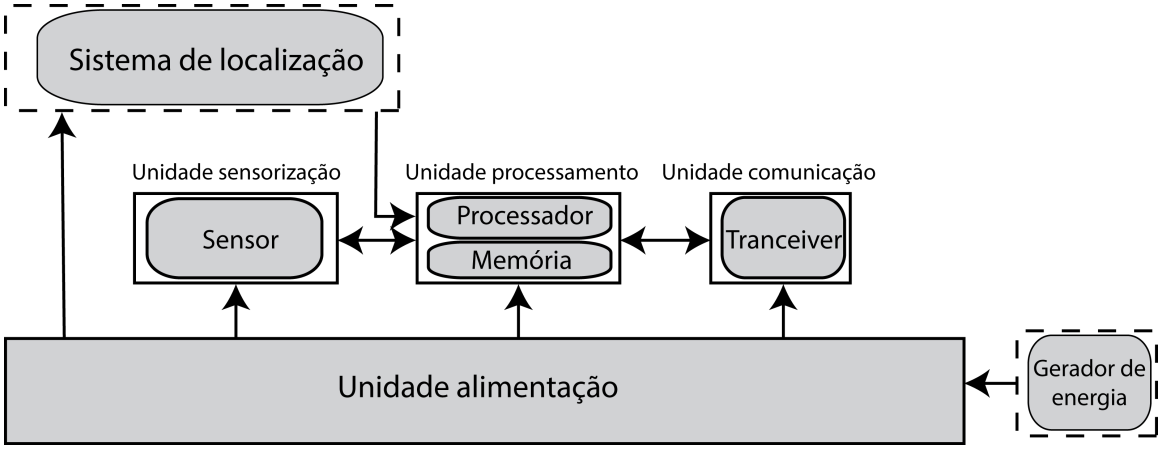


Figura 3.1: Componentes de um nó sensor [9].

3.3.1 Sensorização

A unidade de sensorização tem como princípio o uso de sensores para detetar uma grandeza física, tal como, temperatura, humidade, luminosidade, pressão atmosférica, aceleração, entre outros.

Atualmente, com o aparecimento de sensores MEMS está a surgir uma nova linha de sensores que está a revolucionar o mercado dos sensores de baixo consumo.

Existem dois interfaces disponíveis para sensores: analógico e digital. O analógico transforma a grandeza física medida num sinal elétrico variável que necessita de ser convertido para sinal digital através do uso de um ADC. Por outro lado, para facilitar este processo, existem sensores com interface digital que têm embutidos um sensor analógico que disponibiliza uma saída digital já numa escala apropriada. Normalmente, o interface destes sensores é um dos muitos protocolos de comunicação disponíveis (por exemplo, SPI, I^2C, UART, entre outros).

3.3.2 Processamento

Os microprocessadores modernos dispõem de vários periféricos internos, isto é, ADCs, temporizadores, entre outros. Dispõem também de uma boa capacidade no que refere à memória de dados (RAM) e à memória de código (*flash*). O tamanho que estes processadores atingem torna-os ideais para sistemas embebidos, como é o caso das WSNs. Alguns dos aspetos mais importantes passam por: consumo de energia, tensão elétrica, custo e suporte de periféricos.

Consumo

Nas aplicações baseadas em WSNs, é importante garantir, para os nós sensores, o mínimo consumo de energia possível de modo a aumentar a autonomia da WSN. A maioria dos nós estarão num estado adormecido (CPU deixa de executar e entra em modo *sleep*) 99% do tempo, e por isso, é importante saber qual é o consumo do microcontrolador nesse estado. Os microcontroladores atuais apresentam valores de consumo no modo *sleep* abaixo da dezena de μA. Enquanto o microcontrolador está em estado adormecido apenas tem de manter alguns registos, posições de memória e o tempo de sincronização para saber quando "acordar". Dependendo da taxa de amostragem, o consumo em modo ativo terá mais ou menos impacto no consumo médio do dispositivo. Para tempos de amostragem muito baixos o dispositivo estará, quase na totalidade do tempo, em modo *sleep*, pelo que o consumo em modo ativo não irá fazer grande diferença dado que apenas

acontece, normalmente, durante alguns milisegundos.

Tensão elétrica

A tensão a que o sistema é alimentado também é um fator importante no consumo de energia do dispositivo, pois, os transistores MOS apenas consomem nas transições (ligar/desligar), e portanto, quanto mais baixa for a tensão menor será o consumo, uma vez que o número de comutações é muito elevado (relógio do sistema). Nos microcontroladores tradicionais a alimentação era entre 2.7V e 3.3V, no entanto, atualmente existem microcontroladores a conseguir manter o funcionamento com apenas 1.8V de alimentação.

Suporte aos periféricos

Quase todos os fabricantes apresentam uma grande variedade de periféricos, que variam dentro da mesma família de microcontroladores. Alguns destes periféricos, como temporizadores/contadores, pinos de I/O de propósito geral, interrupções externas e porta série (UART) podem ser encontrados em quase todos os microcontroladores.

Os microcontroladores contêm também ADCs, comparadores analógicos e geradores de PWM, protocolo SPI, capazes de interagir com sensores analógicos.

Os interfaces digitais são importantes para comunicar com sensores que tem interface digital, ou com outros periféricos que usem protocolos de comunicação digitais. Normalmente, é possível encontrar suporte dos seguintes *standards*: SPI, I^2C e UART em que os dois primeiros são mecanismos síncronos e o último é assíncrono.

Memória

Os nós sensores necessitam de espaço para armazenar o código que implementa o protocolo de comunicações sem fios e para armazenar os dados adquiridos do sensor antes de serem processados e transmitidos. Atualmente é possível encontrar uma grande variedade de memórias *flash* e RAM sem restrição em termos de tamanho. No entanto, devido ao custo, estas memórias variam normalmente de 0 aos 256 Kb e de 1 aos 128 Kb, respetivamente.

3.3.3 Alimentação

A alimentação do sistema é uma das unidades mais importantes no nó. Normalmente, a sua capacidade é medida em miliampere-hora (mA-h), isto é, caso uma bateria tenha uma capacidade de 2000 mA-h e se o dispositivo tiver o consumo 20 mA a bateria deverá durar 100 horas. Saliente-se no entanto que a duração da bateria é influenciada por outros fatores, tais como: tempo de vida, a forma como é extraída a energia e a tolerância do sistema a descidas de tensão por parte da bateria. Devido a estes fatores, em alguns caso deixa de ser possível usar toda a capacidade da bateria, no entanto, o uso da mesma pode ser otimizado recorrendo a novas tecnologias de baterias e dispositivos de regulação de tensão.

Tecnologias de baterias

O comportamento de uma bateria varia de bateria para bateria. Os três tipos de baterias mais comum são as alcalinas, as de lítio e as de *NiMH* [39]. em seguida serão apresentadas as principais caraterísticas de cada uma destas tecnologias.

As baterias alcalinas são as mais populares do mercado, são usadas em inúmeras aplicações eletrónicas. Estas baterias são de baixo custo, são de fácil acesso e são excelentes para aplicações de baixo consumo em temperaturas ambientes. Apesar da sua popularidade, estas baterias têm alguns aspetos negativos, isto é, através da temperatura ambiente e do consumo médio de corrente que a bateria tem de fornecer, a capacidade da mesma pode diminuir drasticamente.

Existem vários tipos de baterias de lítio, no entanto as que mais se assemelham às alcalinas, pelo tamanho AA, são as baterias $LiFeS^2$. Estas baterias são um substituto das alcalinas mas têm uma duração maior, conseguem fornecer uma corrente muito maior e tem um desempenho melhor para temperaturas diferentes. No entanto, estas baterias não são recarregáveis existindo um perigo elevado ao tentar fazê-lo.

Por fim, existem as baterias *NiMH* que são muito semelhantes às alcalinas e mantém as suas caraterísticas durante muitas recargas. As principais desvantagens são o seu peso e o facto de ter uma densidade de energia menor, isto é, quanto menor a densidade, maior é a bateria para a mesma capacidade.

Reguladores de tensão

Para contrariar as variações de tensão impostas pelas baterias, podem ser usados reguladores de tensão que são responsáveis por receber uma tensão de entrada variável e produzir uma tensão de saída constante e estável. Atualmente é

possível, através do uso de conversores com *boost*, receber uma tensão de entrada inferior à que se pretende na sua saída. No entanto, a eficiência destes dispositivos ainda não é perfeita, sendo umas das principais desvantagens. Apesar das perdas envolvidas, já é possível encontrar reguladores com uma eficiência de energia aceitável, rondando os 92%.

Recorrendo a energias renováveis é possível utilizar baterias recarregáveis, sendo para isso necessário incluir alguma eletrónica para gestão de energia, como por exemplo, através do uso da família BQ da *Texas Instruments*.

3.3.4 Comunicação

A unidade de comunicação é a unidade mais crítica em termos de consumo de todo o dispositivo. Isto deve-se ao facto de ser a unidade que tem um consumo de energia maior, variando entre 15 mA e as centenas de mA. Esta unidade é responsável pela ligação dos vários nós na rede sendo constituída pelo sistema de transmissão, receção, amplificação e antena.

Comunicação por rádio frequência (RF)

Numa WSN este tipo de comunicação está sujeita a ruído e por isso podem existir perdas significativas de sinal dependendo da distância entre os nós e a existência, ou não, de obstáculos entre eles. Existem mais fatores que podem afetar o alcance de transmissão, entre eles a sensibilidade do recetor, a eficiência da antena e o ganho.

3.4 Tecnologias sem fios

Esta secção tem como propósito uma descrição geral das tecnologias sem fios mais utilizadas focando-se principalmente na tecnologia *ZigBee/IEEE 802.15.4*.

3.4.1 Introdução

No início do ano de 2003 o protocolo *IEEE 802.15.4* foi aprovado depois de muitos esforços ao logo dos anos. A organização do *IEEE 802* focou-se, primeiramente, no desenvolvimento de protocolos com taxas de transmissão médias ou elevadas antes de estruturar um protocolo que respondesse às necessidades de sistemas de baixo consumo e com taxas de transmissão mais baixas. A falta de um protocolo que desse suporte a este tipo de comunicações era um impedimento

para os fabricantes, pois, os protocolos com taxas superiores tinham um preço demasiado elevado para este tipo de sistemas.

Com isto surge o tipo de rede WPAN (*Wireless Personal Area Network*) *802.15* que prevê a ligação de dispositivos distanciados até 300 m. Este padrão existe para salvaguardar as especificações de redes pessoais com baixo consumo de energia e custo reduzido. Existem três classes mais importantes dentro desta categoria: a *IEEE 802.15.1*^{TM}*/Bluetooth* que tem uma transferência de dados média e foi projetada para remover qualquer tipo de cabos em dispositivos eletrónicos; a *IEEE Std 802.15.3*^{TM} que é adequada para sistemas que necessitam de uma qualidade de serviço elevada; e a *IEEE 802.15.4*, também conhecida por LR-WPAN (*Low Rate Wireless Personal Area Netowrk*) que foi desenvolvida para dar suporte a aplicações com um consumo energético, taxa de transferência e complexidade inferiores às restantes classes. A figura 3.2 representa o espaço de operação dos protocolos 802 WPAN e WLAN.

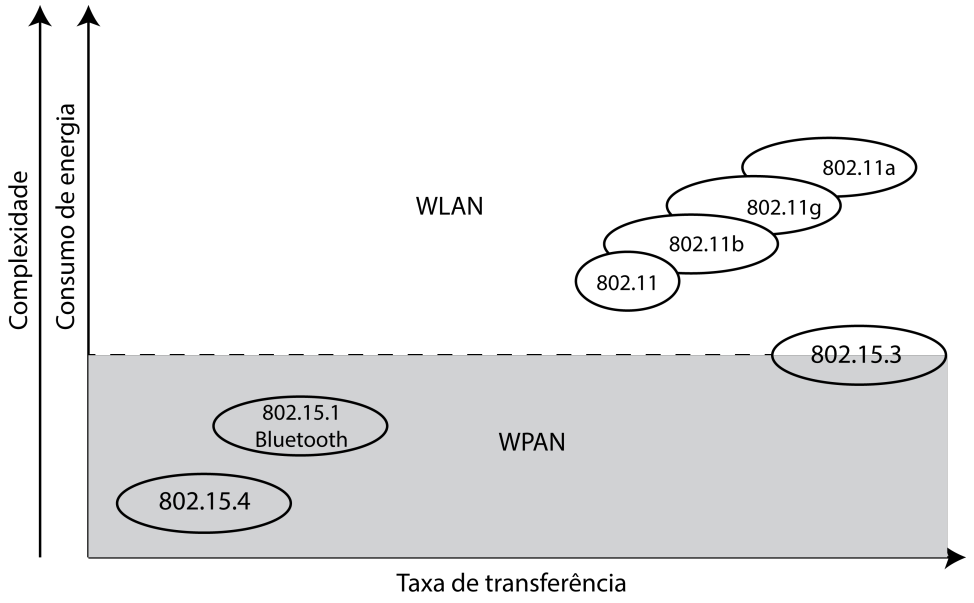


Figura 3.2: Espaço de operação dos protocolos 802 WLAN e WPAN.

Através da tabela 3.1 é possível concluir que o padrão *IEEE 802.15.4* tem um consumo bastante inferior relativamente aos outros padrões apresentados. No entanto, em certos casos, é possível pensar se a *ZigBee* (baseado no *standard* 802.15.4) e o *Bluetooth* são "concorrentes". O que realmente torna o protocolo *IEEE 802.15.4* mais adequado a sistemas de baixo consumo é o seu consumo de energia em modo adormecido, pois, consegue atingir consumos 100 vezes inferior aos do *Bluetooth*.

É importante descrever as principais vantagens do protocolo *IEEE 802.15.4*.

Tabela 3.1: Comparação entre uma LR-WPAN e o protocolo *Bluetooth* e *Wi-Fi*.

	Tecnologias sem fios		
	LR-WPAN	**Bluetooth**	**Wi-Fi**
Débito	250 Kbps	1 Mbps	300 Mbps
Consumos	Tx > 30mA standby: < 10 uA	Tx > 400mA standby: < 0,2 mA	Tx > 400mA standby: < 20 mA
Pilha protocolar	≈ 32 K b	≈ 250 Kb	≈ 1 Mb
Vantagens	Consumo, latência, número de nós e custo	Interoperabilidade e ausência de cabos	Elevada taxa de transferência
Principais aplicações	Monitorização e controlo sensores	Periféricos de PCs, telemóveis e PDAs	Internet, transferência de arquivos, vídeo/áudio
Alcance de transmissão (metros)	1-100+	1-10+	1-100+

Este padrão suporta dispositivos com funcionalidades reduzidas (RFD), o que possibilita obter consumos menores, ou seja maior autonomia para dispositivos alimentados a bateria. Este padrão permite, também, uma grande variedade de topologias desde estáticas a dinâmicas sendo possível passar grandes quantidades de tempo sem existir comunicação entre os dispositivos. A pilha protocolar da *ZigBee* é também adequada para estes sistemas, pois, adiciona um menor nível de complexidade na mesma de forma a que os dispositivos que a utilizam não necessitem de muitos recursos.

3.4.2 *ZigBee*

Depois da visão geral sobre as tecnologias sem fios mencionadas na secção 3.4.1, chegou-se à conclusão que o protocolo *zigbee* é o mais adequado para sistemas de baixo consumo, como o proposto nesta dissertação. É então interessante aprofundar o conhecimento sobre esta tecnologia. E por isso mesmo, será descrito nesta secção o funcionamento global do protocolo.

O protocolo *Zigbee* surgiu em Dezembro de 2004 sendo, no entanto, apresentado a público pela primeira vez no dia 27 de Julho de 2005. Foi criado pela *Zigbee*^{TM} *Alliance* que é composta por mais de 200 empresas a nível internacional e mais de 20 países distintos.

O surgimento deste protocolo deve-se à necessidade de uma norma de redes

sem fios que focasse dispositivos de controlo e sensores. Desta forma foi possível associar a transmissão de dados sem fios com um consumo energético reduzido e com elevada fiabilidade.

Estando presente em aplicações de várias áreas distintas, o protocolo *Zigbee* contém diversas caraterísticas que o torna distinto dos restantes, dando assim suporte às razões que levaram à sua criação [40]:

- Consumo de energia reduzido (*low-power*);
- Pilha protocolar simples o que significa interfaces de baixo custo (*low-cost*);
- Possibilidade de um número de nós elevado por rede, podendo ultrapassar as 65000 unidades;
- Suporte de diferentes topologias: estrela (*star*), malha (*mesh*) e árvore (*cluster tree*);
- O tempo de ligação à rede é menor, comparativamente às outras tecnologias, apresentando assim baixas latências (*low-latency*);
- Existem apenas dois estados de operação: modo ativo e modo adormecido. Desta forma não é necessário a escolha do modo mais adequado para a operação como no caso do *bluetooth*;
- Permite dois modos de operação da rede: modo *beaconing* e *non-beaconing*;
- Elevada segurança, com recurso a uma encriptação de 128 *bits*;
- Duas classes de dispositivos físicos:
  - *Full Function Device* (FFD) - Normalmente desempenha a função de coordenador da rede tendo acesso aos seus dispositivos. É possível usá-lo em qualquer topologia suportada pelo protocolo *Zigbee*, no entanto, estes dispositivos têm um consumo de energia mais elevado (estão sempre ligados) e têm uma construção mais complexa;
  - *Reduced Function Device* (RFD) - Não podem ser usados como coordenador da rede, podendo apenas comunicar com os mesmos. Normalmente são usados no modo adormecido para obter consumos de energia inferiores. Estão limitados a uma topologia em rede e são dispositivos de construção mais simples;
- Operação em três bandas de rádio: $868MHz$ (Europa), $915MHz$ (Estados Unidos) e $2.4GHz$ (Global);

3.4.2.1 Arquitetura protocolar *Zigbee / IEEE 802.15.4*

O protocolo *ZigBee* é baseado no modelo OSI (*Open systems Interconnection*) composto por sete camadas. No entanto, a *ZigBee* apenas usa as camadas necessárias para as funcionalidades desejadas. Na figura 3.3 é possível visualizar a

arquitetura protocolar *ZigBee/IEEE 802.15.4*.

Cada camada comunica com a camada adjacente através de pontos de acesso SAPs (*Service Access Points*).

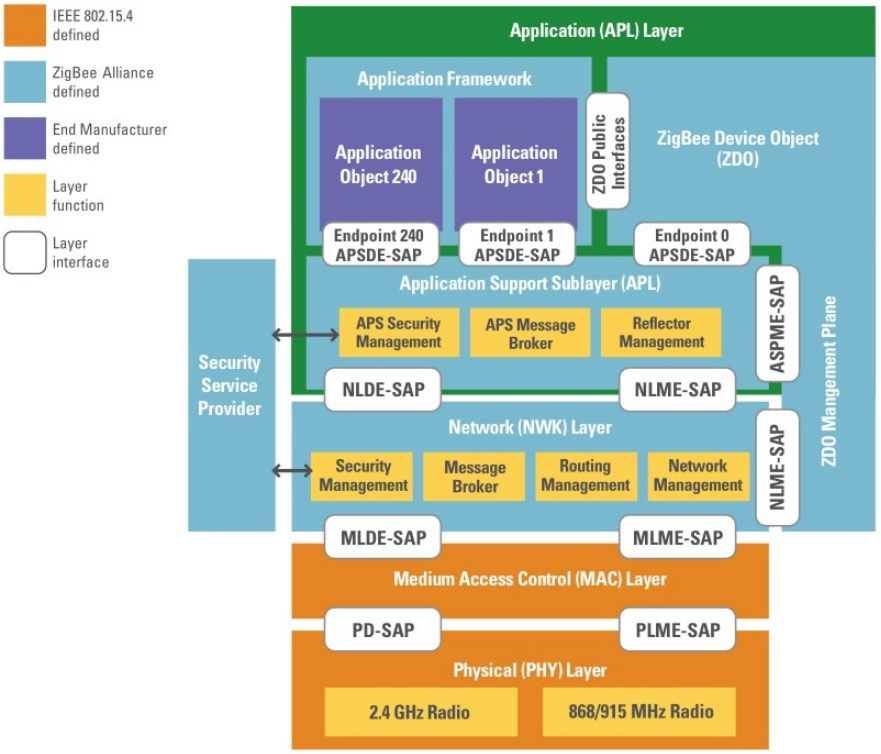


Figura 3.3: Arquitetura protocolar *ZigBee/IEEE 802.15.4* [10]

As duas camadas inferiores, camada física (PHY) e camada de acesso ao meio (MAC) são definidas pelo *standard IEEE 802.15.4*. O protocolo *ZigBee* constrói sobre estas, definindo as camadas de rede superiores, ou seja, a camada de rede (NWK) e a camada de aplicação onde se incluem a subcamada de suporte aplicacional (APS), o objeto de dispositivo *ZigBee* (ZDO), os objetos da aplicação e a *application framework* (AF).

A camada física é responsável por especificar os componentes de interface da rede, os seus parâmetros e modo de operação. Inclui ainda funcionalidades para suportar a operação da camada de acesso ao meio, tais como: deteção da energia recebida (RED), indicador de qualidade de sinal (LQI) e verificação de canal livre (CCA).

A camada MAC controla, tal como o nome indica, o acesso ao meio através de dois modos de operação: *beaconing* e *non-beaconing*. Para além disto, distingue também o tipo de dispositivos permitidos na rede e a estrutura de tramas admissível.

A camada de rede é responsável pela configuração da rede, a descoberta de novos dispositivos, associações e desassociações, manutenção da topologia, gestão da camada de acesso ao meio e gestão de segurança.

A camada APS tem como principal caraterística a manutenção de tabelas de encaminhamento de forma a manter os vários dispositivos interligados consoante os seus serviços e necessidades. Esta camada assume também funções de encaminhamento de mensagens entre dispositivos.

A subcamada ZDO é um objeto da aplicação em todos os nós. Tem como responsabilidades a manutenção dos dispositivos, incluí o papel do dispositivo (coordenador, *router* ou *end-device*) e estabelece uma relação segura entre os vários dispositivos.

3.4.2.2 Camada física (PHY)

Esta camada executa essencialmente o papel de controlador da transmissão e receção de dados através do meio físico envolvente, neste caso o ar.

O protocolo *ZigBee* opera em três frequências de operação [7], normalmente usadas em LR-WPANs, conhecidas como ISM (*Industrial, Scientific and Medical*) que são isentas de licenciamento. Através da tabela 3.2 é possível observar as frequências utilizadas, bem como as taxas de transferência e número de canais para cada uma das bandas utilizadas.

Tabela 3.2: Frequências de operação do protocolo *ZigBee*

Frequências (MHz)	Canais	Taxa de transferência ($Kbps$)	Isenção de licenciamento
869	1	20 - 100	Europa
902 - 928	10	40 - 250	América e Austrália
2400 - 2485	16	250	Mundial

A frequência de cada canal depende da banda utilizada. Através das expressões seguintes consegue-se encontrar o valor da frequência (Fc) do canal (K).

$$Fc = 868.3MHz \quad \text{,para K} = 0 \tag{3.1}$$

$$Fc = 906 + 2(K-1)MHz \quad \text{,para K} = 1,2,...10 \tag{3.2}$$

$$Fc = 2405 + 5(K-1)MHz \quad \text{,para K} = 11,...26 \tag{3.3}$$

Em termos de modelação, esta varia com a banda utilizada. No caso da banda ser de 868 MHz ou 915 MHz é utilizada a modulação BPSK. Caso a banda seja de 2.4 GHz utiliza a modulação O-16QPSK.

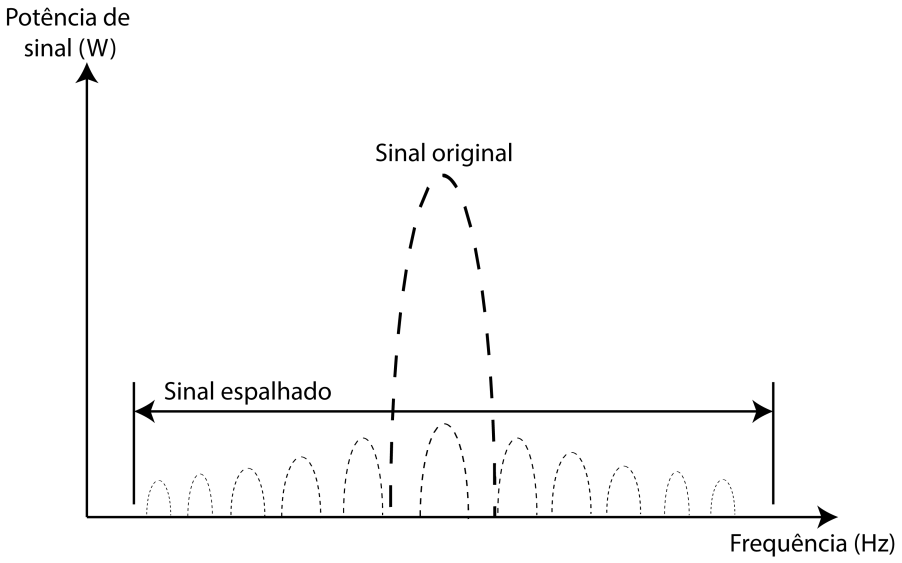


Figura 3.4: Técnica *Direct Sequence Spread Spectrum* [11]

Na camada física é utilizada a técnica de DSSS (*Direct Sequence Spread Spectrum*) o que possibilita um interface de custo reduzido com altos níveis de integração. A DSSS transforma um sinal original com um potência de sinal elevada e espalha a potência do sinal sobre uma faixa de frequência superior. Na figura 3.4 é possível observar uma representação desta técnica.

Esta camada possibilita ainda outras funcionalidades, tais como a ativação ou desativação do *transceiver* de radiofrequência, que pode variar dentro de três modos de funcionamento: a transmitir, a receber ou adormecido; o detetor de energia recebida (RED) que faz uma estimativa do sinal no canal selecionando. Desta forma é possível saber qual o canal que se encontra em melhor condição para transmitir e receber; indicador da qualidade da ligação, que é um pacote que permite avaliar a qualidade do sinal; verificação de canal livre, que identifica se o canal está livre ou ocupado. Pode funcionar dentro de três modos de operação: modo deteção de energia, modo *carrier sense* e modo *carrier sense* com deteção de energia.

3.4.2.3 Camada de acesso ao meio (MAC)

A camada de acesso ao meio [7][41] oferece o interface entre a camada anterior (PHY) e a camada de rede definida pela *ZigBee*. Deve ser capaz de controlar

o acesso aos canais de RF sendo necessário para isso o uso de mecanismos de prevenção de colisões, neste caso o CSMA-CA (*Carrier Sense Multiple Acess - Collision Avoidance*).

Em termos de funcionalidades, esta camada, caso o dispositivo seja um coordenador, deve gerar *beacons* para a rede. No caso de outros dispositivos, deve sincronizá-los na rede. É responsável também pela gestão dos canais de acesso através do GTS (*Guaranteed Time Slot*). Deve garantir que a ligação entre duas entidades MAC, de dois dispositivos diferentes, é confiável e também fornecer serviços de associação e desassociação à rede PAN. Apesar de fornecer mecanismos de segurança, a camada MAC deixa para as camadas superiores a decisão de qual o nível de segurança que deve ser usado.

Modos de operação

Existem dois modos de operação da rede: o modo *beaconing* e modo *non-beaconing*. Estes dois modos apresentam caraterísticas diferentes que deverá ser decidido através da aplicação em questão.

No modo *beaconing* o coordenador transmite periodicamente uma *frame* que é usada pelos outros dispositivos presentes na rede para efeitos de sincronização. Normalmente, o uso deste modo de operação é usado quando a aplicação tem como requisito que o coordenador use bateria.

No modo *non-beaconing* o coordenador está sempre acordado o que possibilita que os dispositivos possam enviar dados a qualquer momento. Este modo de operação tem um consumo de energia superior, daí que não é aconselhado o uso de bateria por parte do coordenador, alimentado o mesmo de forma a que se garanta uma alimentação permanente.

Endereçamento da rede

O protocolo *IEEE 802.15.4* define três tipos de identificadores, de forma a definir endereços de destino e origem:

- **Endereço MAC -** este endereço é composto por 64 *bits* sendo os 24 primeiros "oferecidos" ao fabricante do dispositivo. O endereço MAC é único para cada dispositivo a nível mundial.
- ***Short address* -** este endereço é composto por 16 *bits* e é utilizado na identificação de um nó. Desta forma, é possível obter uma comunicação direta com os dispositivos.
- **PANID -** é composto por 16 *bits* e permite fazer a separação de redes dife-

rentes dentro do mesmo local. Desta forma, é possível duas redes diferentes utilizarem o mesmo canal sem interferências.

Tipos de dispositivos

A rede *ZigBee* diferencia três tipos de dispositivos lógicos: coordenador, *router* e *end-device*. A distinção dos vários dispositivos é feita através do *hardware* que utiliza e da sua capacidade. Tal como foi explicado na secção 3.4.2, estes dispositivos lógicos estão englobados dentro de duas classes: FFD e RFD.

Tabela 3.3: Diferenças entre as duas classes de dispositivos físicos.

Dispositivo físico	Coordenador	*Router*	*End-Device*
FFD	Sim	Sim	Sim
RFD	Não	Não	Sim

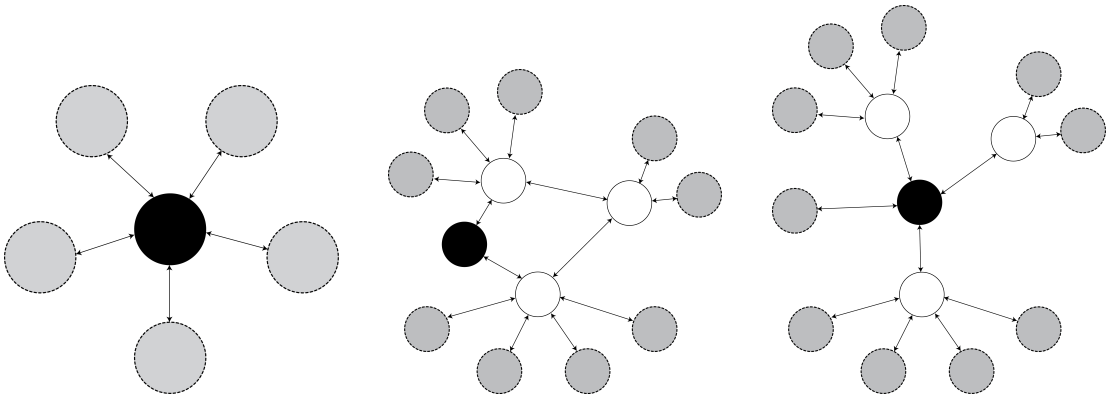
O **coordenador** é o responsável pela criação da rede. Assim, apenas poderá existir um dispositivo deste tipo por rede. Este dispositivo é o que tem acesso a mais informações na rede. Visto a rede necessitar de estar sempre ativa, o coordenador nunca poderá adormecer, o que significa que o seu consumo de energia é superior aos dispositivos RFD.

O **router** é um dispositivo FFD em que a sua principal caraterística é o reencaminhamento de mensagens por parte de outros dispositivos que não estejam dentro do raio de comunicação do coordenador. Desta forma, é possível o router servir de intermediário para que o alcance da rede seja superior. No entanto, estes dispositivos podem desempenhar funções de controlo e monitorização, tal como os *end-devices*. Apesar de bastante úteis, estes dispositivos podem não ser necessários dependendo da aplicação em causa, por isso, são opcionais.

O ***end-device*** apresenta, essencialmente, funções de controlo e monitorização. Estes dispositivos podem ser FFD ou RFD, no entanto, na maioria dos casos são dispositivos alimentados a bateria pelo que é aconselhável, por razões energéticas, que estes possam "adormecer", pertencendo assim aos dispositivos RFD. Em termos de comunicação, estes dispositivos apenas podem comunicar com *routers* ou coordenador, por isso, não é possível transmitir dados com outros *end-devices*.

Topologias

A estrutura de uma WSN incluí diferentes topologias para comunicação rádio. Algumas destas topologias serão discutidas nesta secção com o intuito de distinguir qual o seu comportamento, e também, as suas vantagens e desvantagens. Na figura 3.5 é possível ver as topologias que serão descritas em seguida.



(a) Estrela **(b)** Malha **(c)** Árvore

Figura 3.5: Topologias de uma WSN.

Uma rede em estrela (*Single Point-to-Multipoint*) é constituída por dois tipos de nós, um único nó (coordenador) responsável pela gestão da rede com capacidades para enviar e/ou receber mensagens de outros nós, e os nós sensores (*End-devices*) que só podem comunicar com o nó responsável pela rede. Existem várias razões para a utilização desta topologia, tais como: simplicidade na implementação, baixo consumo de energia nos *end-devices* (já que estes não podem comunicar entre si) e as latências de comunicação são relativamente baixas visto que a comunicação é direta entre dispositivos.

Na rede em malha todos os nós do tipo FFD (coordenador/*routers*) conseguem enviar e/ou receber mensagem entre si. Com este tipo de rede pretende-se que os nós possam usar outro nó vizinho como intermediário caso não estejam no raio de comunicação da estação base. A principal vantagem deste tipo de topologia é que o raio de comunicação não é limitado, isto é, caso um nó por alguma razão não esteja no raio de comunicação da base, basta adicionar um nó intermediário (ou mais se necessário) entre os dois para que estes consigam comunicar. Para além desta vantagem, ganha-se ainda uma grande escalabilidade e redundância no sistema. A principal desvantagem na utilização deste tipo de topologia é o grande consumo de energia que é necessário para a comunicação entre os dispositivos. A razão disto deve-se ao fato de uma grande parte do consumo por parte do nó ser

derivado da comunicação rádio. Outra desvantagem é que quanto maior for o número de saltos (intermediários) necessários para chegar à estação base maior será a latência da mensagem.

A topologia em árvore é semelhante à rede em malha, no entanto, o coordenador assume o papel de nó central da rede. A distribuição de dados e as mensagens de controlo efetuam-se de uma forma hierárquica. Ao contrário da rede em malha, todos os dados terão de passar pelo coordenador, não deixando o router com nenhuma função para além de intermediário.

3.5 Conclusão

Neste capítulo foram apresentados os principais conceitos de uma WSN. Com esta informação é possível agora definir todo o sistema que se pretende desenvolver. No próximo capítulo é apresentada a arquitetura do sistema, as funcionalidades pretendidas e os componentes que serão usados para a sua implementação. Por último, é feita uma pequena descrição da *stack* utilizada, sendo esta baseada no protocolo *ZigBee* apresentado neste capítulo.

Capítulo 4

Especificação do sistema

Com este capítulo pretende-se a especificação de toda a WSN. Será feita uma pequena descrição geral sobre o sistema a ser implementado bem como quais os seus requisitos funcionais e não funcionais. É feita também uma descrição de todos os módulos que compõem a WSN, uma especificação do SoC utilizado, e quais os sensores e ferramentas utilizadas para implementar com sucesso este projeto.

4.1 Requisitos do projeto

Os requisitos deste sistema podem ser funcionais e não funcionais. O requisito funcional define as funções do sistema, enquanto que os requisitos não funcionais focam-se no seu desempenho, segurança, usabilidade, entre outros, podendo, na maioria das vezes, impor restrições ao sistema.

Na especificação do sistema, foram inseridas algumas restrições ao mesmo. Estas consistiam na utilização do microcontrolador *CC2530* e da ferramenta *Z-Stack* desenvolvidas pela TI.

Requisitos funcionais

- Uma WSN inclui os seguintes dispositivos: *end-device*, *router* e coordenador;
- Cada *end-device/router* deve ser capaz de adquirir informação sobre pelo menos uma das seguintes variáveis físicas:
  - Temperatura do ar (ºC);
  - Humidade do ar (%RH);
  - Pressão atmosférica (hPa);
  - Temperatura do solo (ºC);
  - Luminosidade (lux);

- Um *router* pode ter funcionalidades do *end-device*;
- A WSN suporta um sistema de roteamento dinâmico;
- Cada *end-device/router* tem de enviar o nível de bateria do dispositivo em cada SAMPLE (mensagem enviada no protocolo WPAN) ou caso um evento seja ativado;
- Cada *end-device/router* tem de enviar um evento caso detete alguma aceleração não admitida pelo sistema (alarme anti-roubo por exemplo)

Requisitos não funcionais

- Todos os *end-device/router* são alimentados a bateria;
- Cada nó suporta tempos de amostragem diferentes;
- O *router* é alimentado a bateria e utiliza um carregador solar para carregá-las, com o objetivo de aumentar a autonomia das mesmas;
- O coordenador está conetado ao *gateway*;
- Utilizar o SoC *CC2530*;
- Utilizar a *Z-Stack*.

4.1.1 Arquitetura do *HighWayMon*

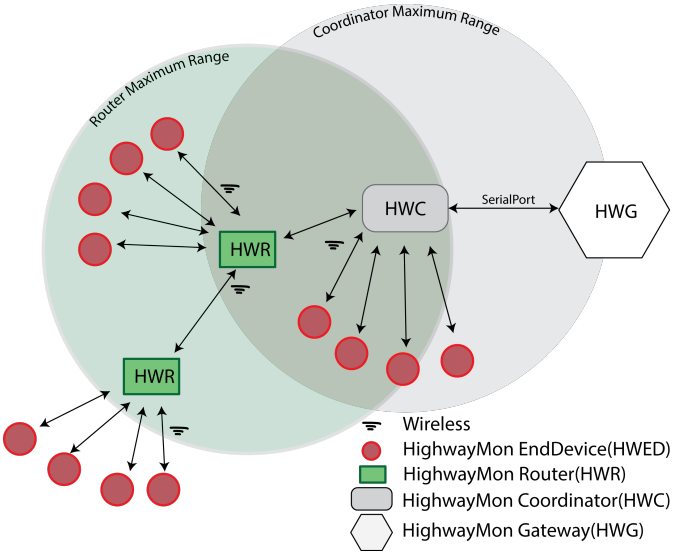


Figura 4.1: Arquitetura do *HighWayMon*.

Este projeto é composto por uma WSN responsável por adquirir informação sensorial, tais como, temperatura e humidade do ar, pressão atmosférica, luminosidade e temperatura do solo, através de HWEDs e HWRs. As informações

adquiridas são enviadas para o HWC que está conetado a um *gateway* com conexão à internet, responsável por enviar estas informações para uma base de dados. Na figura 4.1 estão representados os principais componentes do sistema.

Com a presente dissertação pretende-se apenas o desenvolvimento de uma WSN que consiga enviar as informações recolhidas para o resto do sistema, sendo necessário para isso, um protocolo compatível com o HWG. Isto é feito através do formato JSON (*JavaScript Object Notation*), permitindo a troca de informação entre o HWC e o HWG.

Através da utilização desta arquitetura é possível a criação de um sistema com o intuito de monitorizar as condições das autoestradas, podendo assim, ajudar a minimizar os acidentes nestas vias.

4.2 *Wireless Sensor Network*

A presente secção apresenta a arquitetura da WSN a ser desenvolvida, os módulos que compõem cada dispositivo, bem como as suas funcionalidades. São também definidos os dados do protocolo porta série e WPAN responsáveis por transmitir as informações desde os sensores até ao HWG.

4.2.1 Modularidade do HW

No planeamento da presente dissertação, foi optada uma arquitetura modular para o *hardware* a ser desenvolvido. Esta modularidade tem várias vantagens, desde um sistema flexível e com grande escalabilidade, passando pela abstração do projeto em pequenos subsistemas tornando a sua integração mais fácil. Visto este ser um projeto educacional, torna-se interessante este tipo de arquitetura, sendo possível, caso necessário, a alteração de partes específicas do sistema sem ser necessário a sua completa remodelação.

Na tabela 4.1 é possível observar a subdivisão do sistema nas suas várias partes. É então possível diferenciar 6 módulos: *HWUSB*, *HWCC2530*, *HWPOWER*, *HWSENSORBUS*, *HWACCELEROMETER* e *HWHARVESTING*. Em seguida será apresentada uma breve descrição de cada módulo.

- ***HWUSB*:** Este módulo tem o objetivo de oferecer um interface simples ao *HWCC2530*. Desenvolvido especificamente para o HWC, a utilização deste módulo permite a alimentação do dispositivo, bem como um interface simplista utilizando a porta série (via USB).
- ***HWCC2530*:** Este módulo é comum a todos os dispositivos da WSN. É o

core do dispositivo, tendo integrado apenas o módulo *GB2530* que contém o *SoC CC2530* da *Texas Instruments*.

- ***HWPOWER*:** O principal objetivo deste módulo é fornecer uma alimentação estável ao dispositivo em causa (apenas para HWED e HWR) através do uso de uma bateria e de algum *hardware* necessário para o melhor funcionamento do dispositivo. Para além desta função, é ainda responsável por dar suporte a algumas funcionalidades, tais como: I^2C *Enable*, responsável pela ativação dos sensores que utilizam o protocolo I^2C e o ADC *Enable*, que tem como objetivo a medição do nível da bateria através do ADC.
- ***HWSENSORBUS*:** Este módulo tem como objetivo a ligação física dos sensores ao dispositivo sendo totalmente composto por conetores.
- ***HWACCELEROMETER*:** Neste módulo é possível encontrar um acelerómetro, responsável por detetar acelerações não admitidas pelo sistema.
- ***HWHARVESTING*:** O módulo *harvesting* tem como objetivo ajudar o sistema a prolongar o tempo de vida, através do uso de um painel solar para carregar a bateria.

Tabela 4.1: Especificação dos módulos desenvolvidos.

	HWUSB	*HWCC2530*	*HWPOWER*	*HWSENSORBUS*	*HWACCELEROMETER*	*HWHARVESTING*
HWC	X	X				
HWR		X	X	X	X	X
HWED		X	X	X	X	X
Obrigatório - Opcional - Suportado						

De forma a ligar todos os módulos foram utilizados dois conetores de 20 pinos (2x10) machos e dois fêmea. Desta forma é possível interligar todos os módulos pretendidos de uma forma simples. Os conetores usados são apresentados na figura 4.2. Têm o espaçamento de 1.27mm do tipo SMT, com duas filas de 10 pinos.

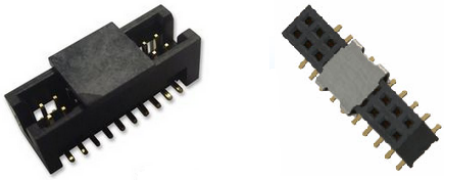


Figura 4.2: Conetores para interligação dos módulos: macho (esquerda) e fêmea (direita).

Com a exceção dos módulos das extremidades (em cima e em baixo) são utilizados dois conetores de cada para que seja possível realizar a ligação de forma correta. Na figura 4.3 está representado o esquemático destas ligações, podendo

já ser observado o propósito de cada pino. Por exemplo, os pinos P0_5 e P1_2 referem-se ao pino de dados e de relógio, respetivamente, do protocolo de comunicação I^2C.

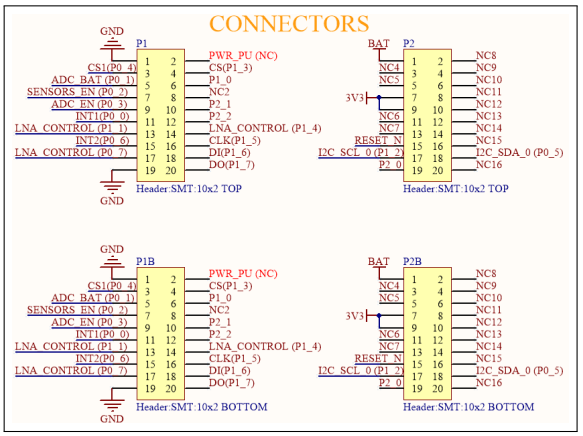


Figura 4.3: Esquemático da ligação entre os módulos.

No caso do *HWCC2530*, que é colocado sempre no topo, apenas são utilizados os conetores fêmea. Por isso, em termos de esquemáticos, apenas são utilizados os que estão designados como *BOTTOM*.

No caso do *HWUSB* apenas são utilizados dois conetores macho, pois, este estará sempre na parte inferior do dispositivo não necessitando dos conetores na parte inferior. Neste caso, apenas serão usados os conetores com a designação TOP.

O módulo *HWPOWER* é um caso mais especial porque é possível usá-lo em duas situações diferentes. A primeira é a forma definitiva, sendo a base do dispositivo e por isso apenas é necessário, tal como o módulo *HWUSB*, os conetores machos. A segunda é na fase de desenvolvimento em que é necessário utilizar a placa de desenvolvimento para o programar ou até mesmo fazer *debug*. Neste caso é necessário ter os quatro conetores (dois superiores e dois inferiores) para que seja possível ligar o dispositivo à programadora *SmartRF05EB*.

4.2.1.1 *HWUSB*

O módulo *HWUSB* foi desenvolvido apenas para o HWC de forma a poder interligar este dispositivo com o HWG através de uma USB. No desenvolvimento deste módulo foram especificados dois subsistemas principais: um circuito integrado denominado *FT232RL* [42] e um LDO (*Low-DropOut regulator*).

Na figura 4.4 está representado o esquemático do circuito utilizado pelo FTDI (*Future Technology Devices International*) que segue as recomendações do desenho

do IC. Representado por *USB1* está o conetor USB que coneta este sistema com um externo (HWG).

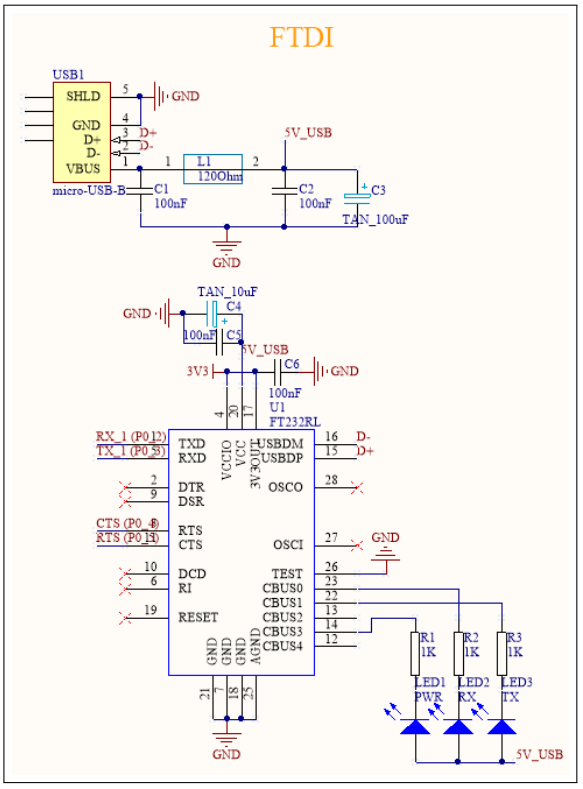


Figura 4.4: Esquemático do circuito FTDI.

São ainda utilizados três LEDs que servem para sinalizar algumas situações. O *LED1* sinaliza quando o dispositivo está ligado, o *LED2* serve para sinalizar que o dispositivo está a receber dados e, por fim, o *LED3* que sinaliza quando está a ser feita uma transmissão.

De forma a garantir uma alimentação estável no dispositivo, foi utilizado um LDO (*TPS63060* [43]) desenvolvido pela TI. Assim, é garantido um funcionamento normal do sistema através de um valor de tensão estável no valor desejado (3.3V) e um valor de corrente suficiente para que o microprocessador possa ter um desempenho normal.

Na figura 4.5 encontra-se o esquemático necessário para a utilização deste IC. A alimentação deste dispositivo é feita através do conetor *USB1* que estará ligado a uma porta série. A tensão fornecida pela porta USB é aproximadamente 5V, no entanto é possível através do FTDI regular a tensão para a pretendida (3.3 V). A utilização do LDO é necessária para obter um débito de corrente estável de forma a manter o funcionamento normal do dispositivo. Caso não se utilizasse este IC poderia o dispositivo não funcionar corretamente em alguns casos (picos

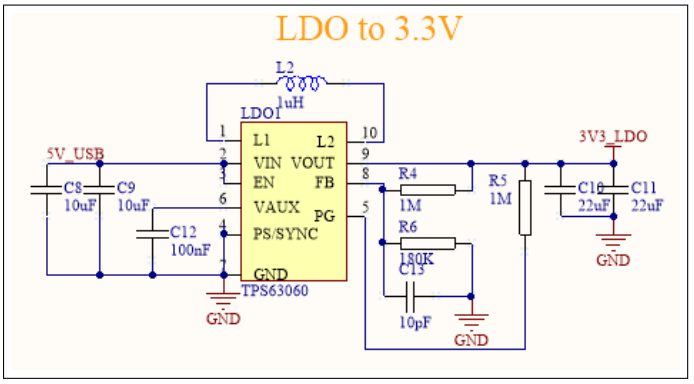


Figura 4.5: Esquemático do circuito LDO.

de corrente). Através das resistências R4 e R6 é possível variar o valor de saída de 2.5V até 8V, pelo que através da equação 4.1

$$Vout = Vin * R6/(R4 + R6) \tag{4.1}$$

é possível chegar aos seguintes valores das resistências: R4 = 1M e R6 = 180K para um valor de tensão de 3.3 V.

Foi ainda implementada a funcionalidade de *reset* ao dispositivo, sendo para isso utilizado um botão, ligado ao pino *RESET_N*, para que em caso de erro fosse possível interromper microprocessador. Na figura 4.6 encontra-se o circuito desenhado.

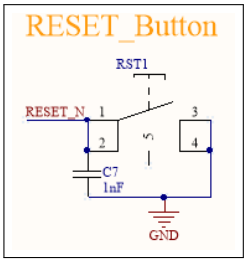


Figura 4.6: Esquemático do circuito *RESET*.

4.2.1.2 *HWCC2530*

Este módulo tem como objetivo interligar o *GB2530*, que contém o microprocessador *CC2530* e o *range extender RFX2401*, com o resto do sistema. Para isso, apenas é necessário fazer as ligações (figura 4.7) os respetivos pinos dos conetores, de forma a assegurar a total compatibilidade do sistema. É utilizado também um par de condensadores de acoplamento de forma a estabilizar a tensão de alimentação.

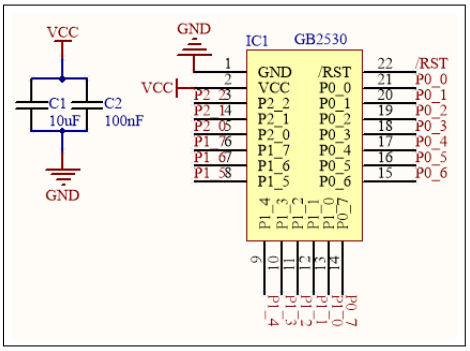


Figura 4.7: Esquemático do *GB2530*.

Na seção 4.3 é feita uma descrição mais pormenorizada do processador *CC2530* e do módulo *GB2530*.

4.2.1.3 *HWPOWER*

O módulo *HWPOWER* é comum nos dispositivos HWR e HWED. Tem como objetivo a alimentação do dispositivo, medição do nível de bateria e a ativação do barramento I^2C para a utilização dos sensores.

Tal como no módulo *HWUSB* é utilizado um LDO (*TPS63060*) de forma a estabilizar a tensão de entrada e garantir corrente suficiente para que o microprocessador trabalhe de forma normal (figura 4.8).

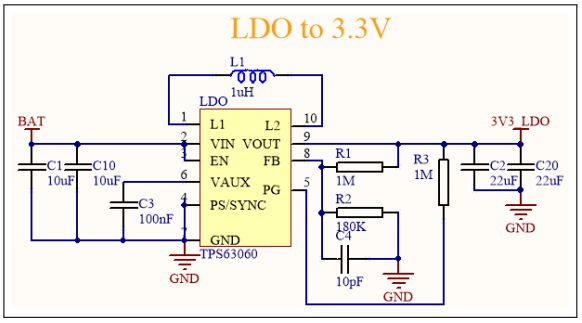


Figura 4.8: Esquemático do LDO.

De forma a minimizar o consumo dos sensores no sistema, foi implementada uma funcionalidade que permite ativar ou desativar o barramento I^2C e desta forma, a alimentação dos sensores. Para isso, é utilizado um sinal digital do microprocessador que dependendo do seu valor (0V ou 3.3V) ativa ou desativa o barramento através do uso de dois *mosfets* (um do tipo P e outro do tipo N). A gate do *mosfet Q2A* (representado na figura 4.9) está conetado ao microprocessador, e quando $VGS<VTH$ o *Q2A* fica ao corte o que significa que $VOUT=VDS=VDD$. Nesse caso, o *Q2B*, que é um *mosfet* do tipo P, também estará na sua zona de

corte, o que significa que irá funcionar como um interruptor aberto. No caso de o *VGS>VTH* o circuito terá a funcionalidade contrária alimentando desta forma os sensores.

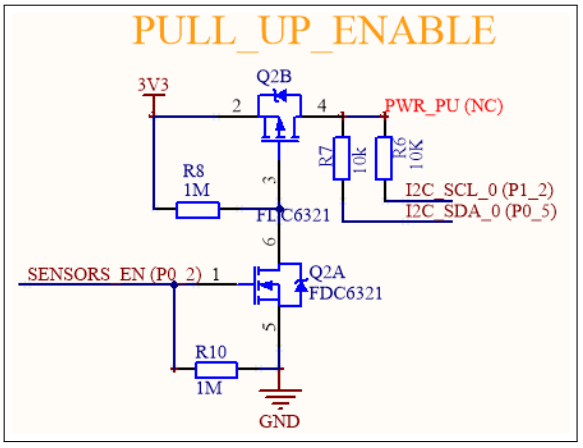


Figura 4.9: Esquemático do circuito de *enable/disable* do I^2C.

Outra funcionalidade necessária é a medição do nível de bateria do dispositivo. Para isso foi criado um circuito, que utilizando o ADC do *CC2530*, é capaz de fazer essa medição. De forma a habilitar ou desabilitar a medição é utilizado um *mosfet* do tipo P (*NX3008PBK*). Visto que o ADC do microprocessador tem um valor máximo que pode medir (1.15V), foi necessário utilizar um divisor resistivo de forma a comparar esse valor com o valor máximo da bateria (3.7V). Para o cálculo das resistências é utilizada a equação 4.2. Na figura 4.10 é possível observar o esquemático do circuito desenhado.

$$VADC = VBat * R29/(R29 + R31) \tag{4.2}$$

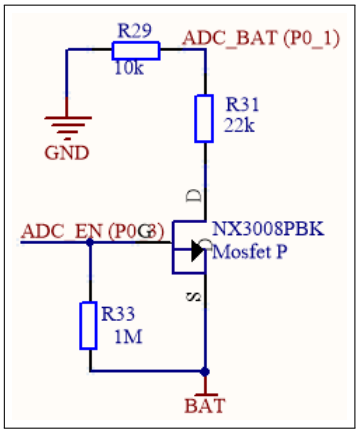


Figura 4.10: Esquemático do circuito de medição do nível de bateria.

Ainda no módulo *HWPOWER* é utilizado o conetor P3 (Figura 4.11) onde é possível ligar a bateria para alimentar todo o dispositivo. Ainda neste esquemático encontra-se um *switch* que tem como objetivo a configuração da alimentação do dispositivo de duas formas. A primeira utiliza a bateria para o alimentar e a segunda é através da programadora (*SmartRF05EB*). Esta funcionalidade foi desenhada apenas direcionando o tipo de alimentação pretendido conforme a posição do *switch*.

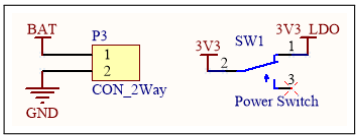


Figura 4.11: Esquemático do *switch* e conetor da bateria.

4.2.1.4 *HWSENSORBUS*

O módulo *HWSENSORBUS* serve apenas como barramento para os sensores de I^2C que se pretende utilizar. É possível utilizar quantos módulos destes quanto necessário, no entanto, como nesta dissertação apenas serão utilizados no máximo quatro sensores em simultâneo (que utilizem o protocolo I^2C), só é necessário um módulo *HWSENSORBUS*.

É possível observar o esquemático deste módulo na figura 4.12 sendo este apenas composto por conetores de quatro pinos. Cada conetor tem quatro tipo de dados diferentes (mas iguais entre todos): *POWER_PU* (*VDD*) e *GND* responsáveis pela alimentação do sistema, *I²C_SDA* e *I²C_SCL* utilizados para o protocolo de comunicação.

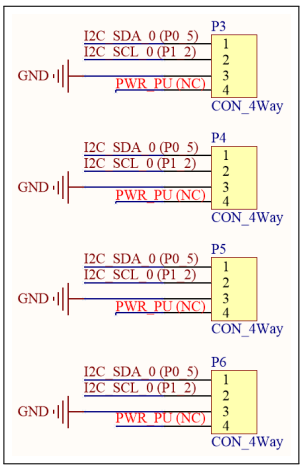


Figura 4.12: Esquemático do *HWSENSORBUS*.

4.2.1.5 *HWACCELEROMETER*

O módulo *HWACCELEROMETER* suporta um acelerómetro de três eixos (*LIS331DLH*) que utiliza o protocolo SPI para comunicar com o microprocessador. Na figura 4.13 pode-se observar o esquemático desenhado em que são apenas feitas as conexões necessárias para o bom funcionamento do acelerómetro.

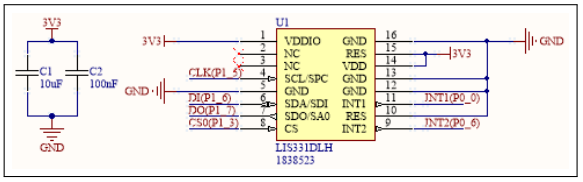


Figura 4.13: Esquemático do acelerómetro *LIS331DLH*.

Para o protocolo SPI foram utilizados 4 pinos: *CLK* (P1_5), *DI* (P1_6), *DO* (P1_7) e *CS0* (P1_3). Foram ainda feitas as conexões para utilizar os pinos P0_0 e P0_6 como interrupção externa em caso de alguma aceleração ser detetada. Dependendo da configuração do acelerómetro podem ser usadas as duas interrupções, uma ou nenhuma. Nesta dissertação optou-se por utilizar apenas uma, a *INT1*.

4.2.1.6 *HWHARVESTING*

Este módulo é responsável por implementar a funcionalidade de *harvesting* no sistema, que através do uso de um painel solar, é possível adquirir energia para carregar a bateria. Isto é feito através do uso do IC *BQ25504* capaz de gerir a energia de entrada obtida, conseguindo sempre a máxima potência possível através do *MPPT* (*Maximum Power Point Tracking*) dinâmico integrado, e desta forma carregar a bateria.

Na figura 4.14 está representado o esquemático do circuito a implementar, tendo em consideração os circuitos exemplos e o suporte oferecido pela TI.

É possível estabelecer três pontos importantes de *threshold* através de resistências. Com as resistências *RUV1* e *RUV2* é possível estabelecer um *threhold* inferior para o nível de bateria (*VBAT_UV*). Para o nível superior da bateria são usadas as resistências *ROV1* e *ROV2* (*VBAT_OB*). Por último tem-se o *VBAT_OK* que permite uma tensão programável, independentemente dos níveis de bateria anteriores, de forma a indicar se o nível da bateria está num valor aceitável.

Com a utilização deste módulo pretende-se aumentar a duração das baterias nos dispositivos, que apesar de ser apenas fundamental nos HWR podem também

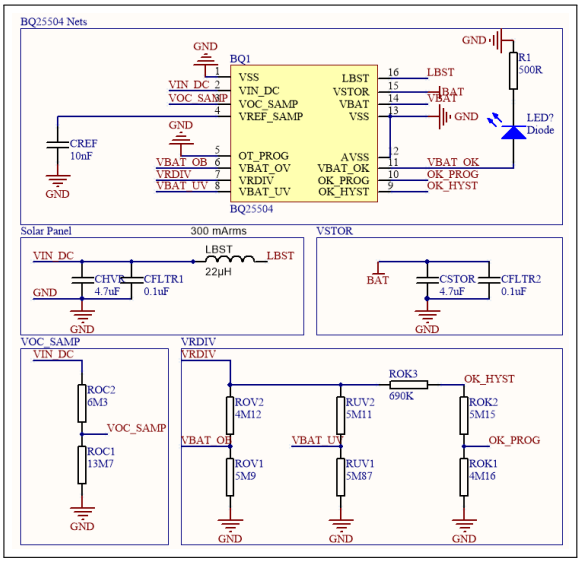


Figura 4.14: Esquemático do módulo *HWHARVESTING*.

ser usados pelos HWED.

4.2.2 *Highway Coordinator* (HWC)

O HWC é a base de toda a rede, isto é, é responsável pela criação da mesma fazendo um "*scan*" ao ambiente procurando por redes já existentes. Posteriormente, escolhe um canal e um identificador de rede (também chamado de PAN ID), tratando de criar então a nova WSN. O *PAN ID* é único para cada WSN, desta forma, é possível definir quais os dispositivos que pertencem à rede criada. Para além das funcionalidades referidas, o HWC também é responsável pela ligação da WSN ao resto do sistema, sendo para isso necessário usar o protocolo porta série. Para o formato dos dados, será usado o formato JSON, visto o HWG estar à espera de uma mensagem com este tipo de formato.

4.2.2.1 Protocolo Porta Série

Este protocolo é relativo à informação transmitida entre o coordenador e o *gateway* do sistema. A mensagem deve estar no formato JSON, como referido anteriormente, e será dividida em dois grupos: informação relativa à rede e informação relativa à aplicação.

A informação a ser transmitida pela porta série relativa à aplicação é a mesma do protocolo WPAN. A única diferença será transmitir esses dados em formato JSON. A informação relativa à rede terá várias variáveis que serão descritas de seguida:

- *WPAN ID*
- *Group ID*
- *Cluster ID*
- Endereço MAC do *Gateway*
- *EndPoint*
- *Broadcast*
- Qualidade da ligação
- Correlação
- RSSI
- Segurança
- Número de sequência

JSON

O formato JSON [44] foi criado inicialmente por *Douglas Crockford*. O JSON tem vindo a substituir o XML e o AJAX devido à sua simplicidade, facilidade para os humanos de escrever e ler e simples para as máquinas de gerarem e fazerem o *parser*. JSON é um formato de texto completamente independente da linguagem utilizada, no entanto, usa convenções que são reconhecidas por alguns programadores da família C: C, C++, C#, *Java*, *JavaScript*, *Perl*, *Python* e outros, tornando este formato ideal para transferência de dados entre linguagens.

O JSON é construído sobre duas estruturas, uma coleção de pares nomes/valor e uma lista ordenada de valores. Um objeto, neste formato, consiste em pares nome/valor desordenados. O objeto começa com { (braço esquerdo) e termina com } (braço direito). Cada nome é seguido por : (dois pontos) e cada par é separado por , (virgula).

4.2.3 *Highway End-Device* (HWED)

Este dispositivo é responsável por adquirir todas as informações necessárias relativas ao ambiente envolvente, tais como: temperatura e humidade do ar, luminosidade, pressão atmosférica e temperatura do solo. Para isso serão necessário utilizar sensores de baixo consumo com um protocolo de comunicação digital. Na figura 4.15 é possível observar a arquitetura do HWED. Este é composto por dois módulos obrigatórios (*HWPOWER* e *HWCC2530*), dois módulos opcionais (*HWACCELEROMETER* e *HWSENSORBUS*) e tem suporte para o módulo *HWHARVESTING*.

Relativamente aos sensores utilizados o HWED dá suporte a quatro tipos de

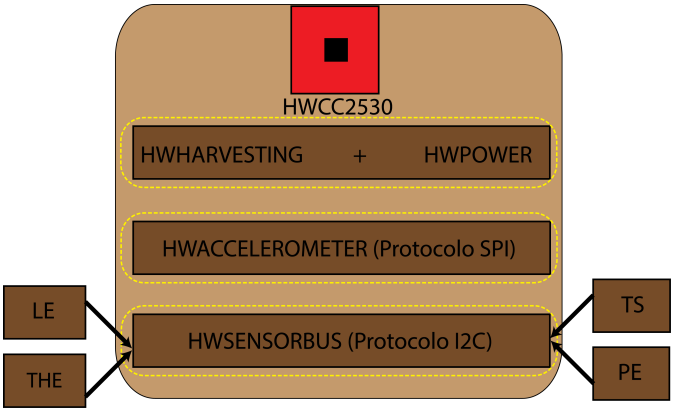


Figura 4.15: Arquitetura do HWED.

sensores que utilizam o protocolo I^2C. Em seguida será dada uma breve explicação sobre cada um deles. Na secção 4.4 será feita uma explicação mais detalhada sobre cada sensor.

- **THE:** Este módulo está responsável pela medição da temperatura e humidade do ar. Para isso será utilizado o *SHT21* desenvolvido pela *Sensirion*.
- **LE:** O objetivo deste módulo é a aquisição da luminosidade envolvente através de um sensor de luminosidade, o *MAX44009* desenvolvido pela *MAXIM*.
- **PE:** Este módulo tem como objetivo adquirir a pressão barométrica do ambiente envolvente, sendo para isso utilizado o sensor *BMP180* desenvolvido pela *Bosch Sensortec*.
- **TS:** Este módulo está responsável por adquirir a temperatura do solo. Para isso será utilizado o sensor *TMP100* desenvolvido pela TI.

Um dos requisitos do projeto é um sistema anti-roubo. Para isso será utilizado um acelerómetro (*HWACCELEROMETER*) que caso detete uma aceleração deve enviar um evento, através de uma interrupção do MCU, sendo possível, desta forma, monitorizar se o dispositivo foi mexido ou não. Este sensor utiliza o protocolo digital SPI para comunicar com o sistema.

Umas das principais vantagens do uso destes dispositivos é a possibilidade de utilizar o modo *sleep*. Desta forma o consumo de energia decresce drasticamente, sendo viável a utilização de WSNs.

4.2.3.1 Protocolo WPAN

O designado protocolo WPAN é referente aos dados que são transmitidos entre os dispositivos que adquirem informação dos sensores (HWR e HWED) e o coordenador da rede. De forma a generalizar os dados transmitidos entre dispositivos optou-se pela criação de uma estrutura que agrupasse todas as variáveis

utilizadas. Em seguida é possível observar todas as informações que se pretendem adquirir:

- ID do pacote
- Endereço MAC do dispositivo
- Taxa de amostragem (segundos)
- Tipo de dispositivo
- Informação sobre o nível da bateria (V)
- Informação relativa ao acelerómetro
- Informação sobre ao sensor de temperatura e humidade do ar (ºC/%RH)
- Informação relativa ao sensor de luminosidade (lux)
- Informação sobre o sensor de pressão (hPa)
- Informação relativa ao sensor de temperatura do solo (ºC)

É importante referir que este protocolo deve ser capaz de incluir, ou não, as informações relativas aos sensores, isto é, caso um sensor não esteja conetado não deve ser enviado o campo relativo a esse sensor. Para isso é utilizada a variável *packet ID* de 16 *bits*, em que cada *bit* corresponde a um tipo de dados. Ao enviar os dados é feito um teste a esta variável que define quais os dados sensoriais que se devem incluir. Nesta dissertação apenas serão usados seis *bits*, no entanto, ainda existe espaço para adicionar mais sensores caso necessário. Na tabela 4.2 são apresentados quais os *bits* correspondentes a cada tipo de dados.

Tabela 4.2: Correspondência dos *bits* do *packet ID* com as suas variáveis.

Variável	**Bit correspondente**
Bateria	0x01
Aceleração	0x02
Temperatura e Humidade do ar	0x04
Luminosidade	0x08
Pressão	0x10
Temperatura do solo	0x20

4.2.4 *Highway Router* (HWR)

Apesar de ser um dispositivo opcional, o uso deste tipo de equipamento permite obter um maior alcance, em termos de distância, da WSN. Os *routers* são

dispositivos que necessitam de estar sempre em modo ativo, por isso, é necessário que a sua alimentação tenha o melhor desempenho possível para que a duração da bateria seja maior. Visto isto, os HWR deverão ter suporte de um painel solar para carregar a sua bateria. Esta medida é crucial para o sucesso do dispositivo.

Para além desta caraterística, própria dos *routers*, o HWR é capaz de se comportar da mesma forma que um HWED, isto é, pode adquirir informação sensorial (se necessário). Desta forma, é possível desenvolver um nó igual tanto para o HWR como para o HWED, tornando o sistema modular relativamente ao hardware utilizado.

4.3 *CC2530*

O *System-on-Chip CC2530* [12], desenvolvido pela *Texas Instruments*, combina o poder do MCU 8051 com um *transceiver* de radiofrequência compatível com o *standard* IEEE 802.15.4. O *transceiver* RF funciona na banda do 2.4 GHz podendo atingir taxas de transferência até 250 Kbps. Através do uso deste componente COTS é possível desenvolver novas aplicações com *time-to-market* reduzido, mantendo os custos e consumos reduzidos.

Figura 4.16: *System-on-Chip CC2530* [12].

A memória de dados do microprocessador é de 8 Kb, sendo única para todas as versões existentes. No total existem quatro versões: *CC2530F32/64/128/256* que correspondem a 32/64/128/256 Kb de memória *flash*, sendo utilizada para guardar o *firmware* desenvolvido.

O controlador de I/O é responsável pelos pinos de propósito geral, sendo possível configurá-los para o que for necessário, por exemplo, como entrada ou saída, se estão sob controlo do *software* ou do *hardware* e até mesmo se o pino tem uma resistência *pullup* ou *pulldown*.

Através do controlador DMA de cinco canais, é possível aceder à memória, para leitura ou escrita, sem ser necessário interromper o CPU. Desta forma, é possível uma maior eficiência do sistema, já que a maioria dos periféricos, tais como, núcleo AES, controlador da flash, USARTs, *timers* e ADC, utilizam este método de acesso aos dados.

Este microprocessador dispõe de 2 USARTs configuráveis que servem para ser usadas como UART ou SPI (*master* e *slave*), criando a possibilidade de interligação de outros periféricos ou até mesmo sistemas.

Existem ainda vários módulos importantes no sistema, tais como, os *Timers* 1, 2, 3 e 4, um núcleo AES para encriptação de dados, um ADC de oito canais para adquirir sinais analógicos e convertê-los para digital, entre outros como é possível observar na figura 4.17.

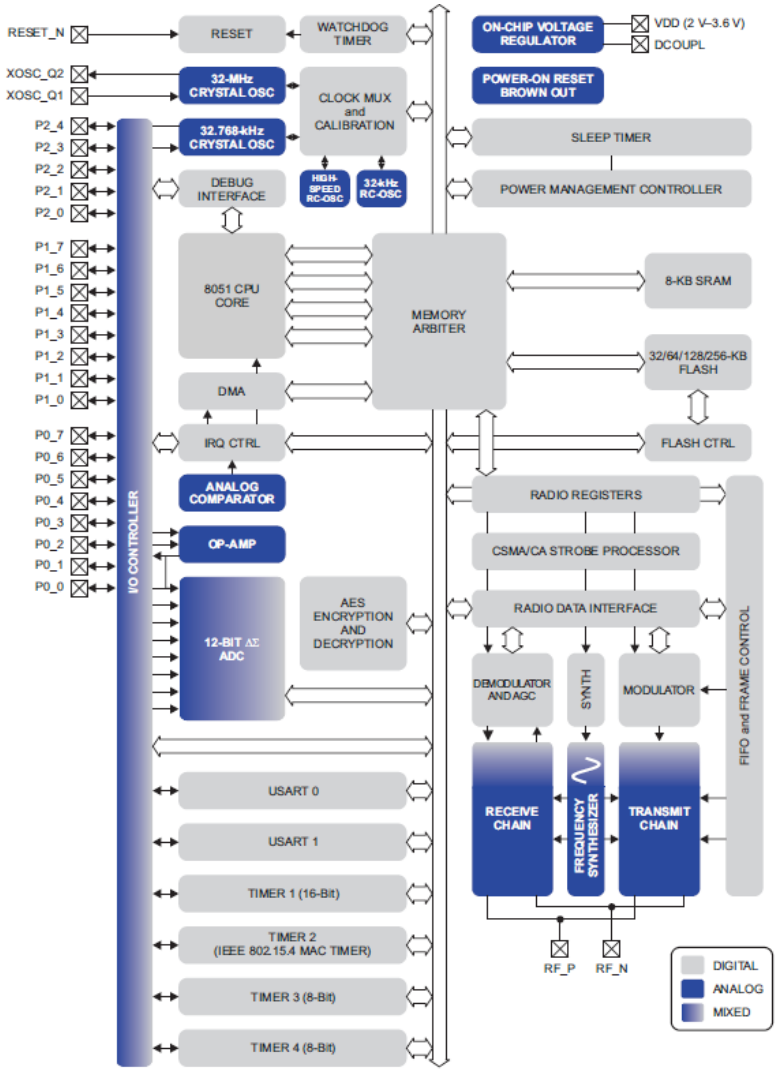


Figura 4.17: Arquitetura do *CC2530*.

4.3.1 *CC2530 Evaluation Modules*

Na proposta dissertação foram usados dois módulos diferentes em que ambos tem como base o *CC2530* da TI. A razão da escolha destes dois módulos deve-se

ao fato de um, o *CC2530-91*, já ter sido desenvolvido no Grupo de Investigação de Sistema de Embebidos, totalmente testado e com bons resultados. O outro módulo (*HWCC2530*), contém o *SoC GB2530* de uma marca chinesa, e como tem de ser testado pode influenciar o teste dos outros módulos. Por isso, o *CC2530-91* será usado para testar todos os módulos desenvolvidos de forma a minimizar erros, enquanto o *HWCC2530* será usado apenas depois de totalmente testado.

Tabela 4.3: *Pinout* dos portos do *CC2530* nos conetores P3 e P4.

P3				**P4**			
Pin	CC2530 Port		***Pin***	***Pin***	***CC2530 Port***		***Pin***
1			2	1			2
3	P0.4	P1.3	4	3			4
5	P0.1	P1.0	6	5			6
7	P0.2		8	7			8
9	P0.3	P2.1	10	9	VDD		10
11	P0.0	P2.2	12	11	VDD		12
13	P1.1	P1.4	14	13			14
15	P0.6	P1.5	16	15			16
17	P0.7	P1.6	18	17	P1.2	P0.5	18
19		P1.7	20	19	P2.0		20

A razão da utilização do *HWCC2530* deve-se ao fato do módulo ser mais compacto e os raios de comunicação entre dispositivos serem superiores aos do *CC2530-91*, o que neste projeto é crucial para reduzir drasticamente o número de dispositivos no sistema. No entanto, os consumos do módulo *GB2530* são superiores o que em algumas situações é uma desvantagem. A WSN desenvolvida pode usar qualquer um dos módulos, sendo um fator a decidir dependendo do local onde for feita a instalação da mesma, isto é, no caso do alcance não ser um fator importante é possível obter uma duração superior do dispositivo com a utilização do módulo *CC2530-91*. Em seguida serão apresentadas as caraterísticas do módulo *CC2530-91* e do SoC integrado no *HWCC2530*. Na tabela 4.3 é possível observar o *pinout* do *CC2530* relativamente aos conetores P3 e P4.

CC2530-91

Este módulo, apresentado na figura 4.18, combina o *CC2530* com o *range extender CC2591* baseado nos esquemáticos do *CC2530-91EM* disponível pela TI. O *CC2591* foi desenvolvido para *transceivers* RF na gama dos $2.4GHz$ e para SoCs da TI. Este fornece um maior poder de amplificação e com baixo ruído aumentando assim o desempenho de aplicações sem fios.

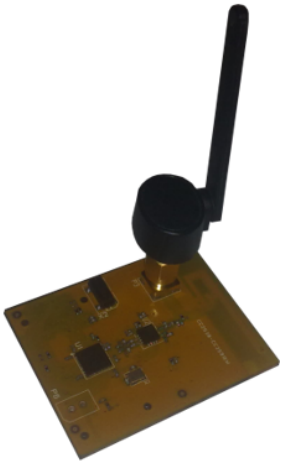


Figura 4.18: Módulo *CC2530-91* desenvolvido pelo Grupo de Sistemas Embebidos.

HWCC2530

O *HWCC2530* é composto pelo módulo *GB2530-S* [45] desenvolvido pela empresa *GBAN Electronic Technology*. Este módulo utiliza o SoC *CC2530* da TI, falado anteriormente, que em conjunto com um *range extender*, o *RFX2401* [46], conseguem obter um módulo *low-power* com o raio de comunicação superior ao apresentado anteriormente. Na figura 4.19 é possível observar o módulo *GB2530*.

Figura 4.19: Módulo *GB2530*.

Na tabela 4.4 são apresentadas as principais caraterísticas deste módulo comparativamente ao módulo *CC2530-91*. De notar que umas das principais vantagens

deste módulo é o seu maior alcance, comparativamente ao *CC2530-91*, podendo atingir os 2000m de alcance entre dispositivos.

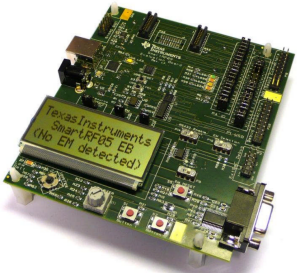
Tabela 4.4: Caraterísticas do módulo *GB2530-S* e comparação com o módulo *CC2530-91*.

Desempenho	***GB2530-S***	***CC2530-91***
Chips	*CC2530 + RFX2401*	*CC2530 + CC2591*
Serial Data Rate	até 115.2 Kbps	até 115.2 Kbps
Taxa transmissão RF	250 Kbps	250 Kbps
Corrente ao receber (típica) (RX modo ativo, CPU *idle*)	43.6 mA	27 mA
Corrente ao transmitir (típica) (TX ativo a 1dBm, CPU *idle*)	138.6 mA	166 mA
Frequência de rádio		
Banda de frequência	2400 - 2483.5 MHz	2400 - 2483.5 MHz
Potência de Sinal	-13 dBm a 22.5 dBm	-10 dBm a 20 dBm
Sensibilidade	-110 dBm	-98.8 dBm (Alto ganho) -90.4 dBm (Baixo ganho)
Alcance	600 - 2000m	600 - 1600m

4.3.2 *SmartRF05 Evaluation Board*

A programadora *SmartRF05EB* utilizada neste projeto foi desenvolvida no Grupo de Investigação de Sistema Embebidos da Escola de Engenharia da Universidade do Minho. Este dispositivo tem como base a placa de desenvolvimento *SmartRF05EB* [47] desenvolvida pela TI, que já dá o melhor suporte aos produtos de RF por eles desenvolvidos. No entanto, e porque houve a necessidade de desenvolver uma nova placa, optou-se por incluir melhorias e novas funcionalidades à placa já existente. A principal preocupação a ter em conta foi que este dispositivo continuasse a dar suporte a todos os dispositivos de baixo consumo da família RF da TI. A tabela 4.5 ilustra as alterações efetuadas entre a placa pelo Grupo de Sistemas Embebidos comparativamente à da TI.

Tabela 4.5: Alterações efetuadas à *SmartRF05RB*.

SmartRF05EB	**Nova *SmartRF05EB***
	
LCD 3x16 carateres	Trocou-se por um 1.8in color TFT 128*160 com interface SPI e suporte para cartão *MicroSD*
UART Interface alta velocidade USB 2.0	Combinação dos dois através do USB 2.0 *HUB TUSB2036* da TI
Serial Flash	Removido, pode ser usado como componente externo caso necessário
Potenciómetro	Removido, pode ser usado como componente externo caso necessário
Dimensão: 128 x 113 mm	Reduzido para 110 x 73 mm
2 baterias do tamanho AA	Removido o suporte para baterias AA. Adicionado suporte para baterias Lipo e carregador com proteção

4.4 Sensores

Nesta secção será apresentada toda a informação importantes sobre os sensores utilizados na realização deste projeto. Como já referido anteriormente, são usados 5 sensores responsáveis por monitorizar algumas variáveis físicas do ambiente que as envolve.

Para determinar qual o melhor sensor a utilizar realizou-se uma pesquisa relativamente aos sensores disponíveis no mercado para cada uma das áreas em que se pretende medir alguma variável física. A tabela 4.6 apresenta alguns dos sensores encontrados bem como as suas principais caraterísticas.

Após análise dos sensores anteriores, e tendo como objetivo utilizar senso-

Tabela 4.6: Comparação entre vários sensores disponíveis no mercado.

Sensores	Protocolo de comunicação	Resolução	Alcance
Temperatura e Humidade			
SHT21	I^2C	Humidade: 8-12 *bits* Temperatura: 12-14 *bits*	Humidade: 0 a 100% Temperatura: -40 a 120ºC
Si7005	I^2C	Humidade: 12 *bits* Temperatura: 14 *bits*	Humidade: 0 a 100% Temperatura: -40 a 85ºC
Temperatura			
DS18B20	*1-Wire*	12 *bits*	-55 a 125ºC
CAT6095	I^2C	12 *bits*	-40 a 125ºC
STTS75	I^2C	9-12 *bits*	-55 a 125ºC
TMP100	I^2C	9-12 *bits*	-40 a 125ºC
Luminosidade			
TSL2561	I^2C	16 *bits*	Variável
MAX44009	I^2C	0.045 lux	0.045 a 188000 lux
Pressão Atmosférica			
LPS331AP	I^2C ou SPI	24 *bits*	260 a 1260 mbar
BMP180	I^2C ou SPI	16-19 *bits*	300 a 1100 mbar

res já existentes no departamento, foram escolhidos os componentes descritos em seguida.

4.4.1 *SHT21* (THE)

O módulo THE é composto pelo sensor desenvolvido pela *Sensirion*, o *SHT21* [48]. Este sensor permite adquirir informação sobre a temperatura e humidade relativa envolvente, permite calibração e é alimentado na mesma gama de tensões do restante sistema, disponibilizando estas informações através do interface digital I^2C. Na figura 4.20 é possível observar o sensor referido. Através da adição de um *filter cap* com proteção *IP67* [49], o *SHT21* está protegido completamente contra poeira e contra efeitos de emersão entre 15cm e 1m.

Na tabela 4.7 estão representadas algumas das principais caraterísticas deste sensor. Através do endereço único que este sensor possuí é possível comunicar com ele usando o barramento I^2C, tendo este sensor o papel de *slave* no sistema. Após a inicialização do protocolo I^2C e da configuração do sensor (caso necessário)

Figura 4.20: Sensor de temperatura e humidade *SHT21*.

Tabela 4.7: Principais caraterísticas do sensor *SHT21*.

	Temperatura	**Humidade**
Protocolo	I^2C	
Resolução	11 - 14 bits	8 - 12 bits
Gama de valores	-40ºC a 125ºC	0-100%RH
Alimentação	2.1 a 3.6V	
Endereço	0x20	
Comando	0xE3	0xE5

é possível interagir com o mesmo enviando o endereço do dispositivo (0x20) e selecionar o modo de escrita. Logo de seguida é enviado um comando que tem como objetivo identificar a variável que se pretende ler. No caso da temperatura é enviado o comando 0xE3 e para a humidade o comando 0xE5. A partir deste momento só falta ler a informação recolhida pelo sensor, sendo para isso necessário reinicializar o protocolo I^2C selecionando desta vez o modo de leitura. O *slave* irá então enviar a informação através de 3 *bytes*, em que o último é usado para verificar a integridade dos dados (CRC de 8 *bits*). Para finalizar este processo é necessário parar o protocolo I^2C. Depois de obtidos estes dados é possível calcular os respetivos valores de temperatura e humidade através das equações 4.3 e 4.4.

$$temperatura(^{\circ}C) = 175.72 * (valor\_lido/65536) - 46.85 \tag{4.3}$$

$$humidade(\%RH) = 125 * (valor\_lido/65536) - 6 \tag{4.4}$$

4.4.2 *MAX44009* (LE)

Desenvolvido pela *Maxim Integrated* o sensor de luminosidade, *MAX44009* [50], é um componente integrante do módulo LE. Este sensor tem interface digital I^2C, é alimentado dentro da gama de tensões do restante sistema e é ideal para um

vasto número de aplicações. Na figura 4.21 é possível observar o sensor referido.

Figura 4.21: Sensor de luminosidade *MAX44009*.

Na tabela 4.8 estão representadas as principais caraterísticas deste sensor, bem como algumas informações para a utilização do mesmo.

Tabela 4.8: Principais caraterísticas do sensor *MAX44009*.

	Luminosidade
Protocolo	I^2C
Resolução	0.045 Lux
Gama de valores	0.045 a 188,000 Lux
Alimentação	1.7V a 3.6V
Endereço	0x4A ou 0x4B
Comando	0x03 e 0x04

Através do barramento de I^2C é possível comunicar com o *MAX44009*, sendo para isso necessário haver uma inicialização do I^2C e do próprio sensor, configurando-o caso necessário. Em seguida é enviado o endereço escolhido, sendo possível escolher entre 0x4A e 0x4B, em modo de escrita. Imediatamente a seguir é enviado o comando respetivo aos valores mais significativos dos dados, o 0x03. Para efetuar a leitura é necessário voltar a iniciar o I^2C e enviar o endereço novamente, desta vez em modo de leitura. A partir deste momento é possível ler o *byte* mais significativo e concluindo com a paragem do I^2C. Até este momento apenas se tem o *byte* mais significativo, sendo ainda necessário ler o menos significativo. Para isso apenas é preciso repetir o que foi explicado anteriormente alterando o valor do comando para 0x04.

Para chegar ao valor final em lux é preciso fazer alguns cálculos para obter o valor *exponent* e *mantissa* da fórmula 4.5. Através da fórmulas 4.6 e 4.7 estes

dados são adquiridos, sendo então possível obter o valor final.

$$luminosidade(Lux) = 2^{exponent} * mantissa * 0.72 \tag{4.5}$$

$$exponent = (HighByte \ \& \ 0xF0) >> 4 \tag{4.6}$$

$$mantissa = ((HighByte \ \& \ 0x0F) << 4) \mid (LowByte \ \& \ 0x0F) \tag{4.7}$$

4.4.3 *BMP180* (PE)

O módulo PE é composto pelo sensor de pressão barométrica e temperatura *BMP180* [51]. Este sensor *low power* desenvolvido pela *Bosch Sensortec* tem interface digital I^{2}C, é totalmente calibrado e é alimentado na gama de tensões semelhante ao resto do sistema. Este sensor contém uma unidade de controlo com E^{2}PROM que contém os dados para a calibração do *BMP180*. Na figura 4.22 é possível observar o sensor em questão.

Figura 4.22: Sensor de pressão e temperatura *BMP180*.

Na tabela 4.9 estão representadas algumas das principais caraterísticas deste sensor.

Tabela 4.9: Principais caraterísticas do sensor *BMP180*.

	Temperatura	**Pressão**
Protocolo	I^{2}C	
Resolução	16 *bits*	16 a 19 *bits*
Gama de valores	-40ºC a 85ºC	300 a 1100 hPa
Alimentação	1.8 a 3.6V	
Endereço	0x77	

De forma a obter os valores pretendidos, é necessário utilizar o barramento I^{2}C para comunicar com o sensor. Primeiramente, deve-se proceder à inicialização

do barramento e do *BMP180*, sendo possível nesta altura obter todas as informações relativas ao sensor, tais como: endereço e o ID do *chip*. De forma a ser possível fazer a calibração dos valores obtidos, é necessário ler da memória E^2PROM 11 valores de 16 *bits* guardando-os para uso futuro. Em seguida procede-se à leitura da temperatura sem compensação através da inicialização do I^2C e envio do endereço em modo de escrita, seguindo-se do comando respetivo e do valor a ser escrito no mesmo. No caso da temperatura é o comando 0xF4 e os dados 0x2E. Após um pequeno tempo de espera dependendo da resolução reinicia-se o I^2C e envia-se o endereço respetivo desta vez em modo de leitura. Em seguida procede-se à leitura de dois *bytes* que correspondem à temperatura sem compensação, finalizando com a paragem do I^2C. De forma a obter-se a temperatura nos valores em ºC devem utilizar-se as equações disponíveis no *datasheet* ([51]) do *BMP180*. Para a obter as informações relativas à pressão utiliza-se o mesmo processo mas com os dados relativos à variável pretendida.

Apesar de a pressão atmosférica ser a informação pretendida, este sensor fornece também a temperatura à sua volta, por isso, será utilizada esta informação para saber se a temperatura obtida pelo sensor de temperatura do ar está num valor próximo da adquirida por este sensor.

4.4.4 *TMP100* (TS)

De forma a adquirir informação da temperatura do solo utilizou-se o sensor *TMP100* [52] desenvolvido pela *Texas Instruments*. Este sensor dispõe de interface digital I^2C, é de baixo consumo e pode ser alimentado na gama de tensões do restante sistema. Na figura 4.23 é possível observar o respetivo sensor.

Figura 4.23: Sensor de temperatura *TMP100*.

Na tabela 4.10 estão representadas as principais caraterísticas deste sensor, bem como algumas informações para a utilização do mesmo.

Tabela 4.10: Principais caraterísticas do sensor *TMP100*.

	Temperatura
Protocolo	I^2C
Resolução	9 a 12 *bits*
Gama de valores	-40ºC a 125ºC
Alimentação	2.7V a 5.5V
Endereço	0x48

Através do barramento de I^2C é possível comunicar com o *TMP100*. Uma das vantagens do sensor apresentado é a possibilidade de poder selecionar diferentes endereços para o mesmo sensor. Isto é feito através das ligações de dois pinos, o A0 e o A1. Neste caso será usado o endereço 0x48 para comunicar com o sensor, que corresponde aos dois pinos ligados ao GND. O resto do processo é bastante semelhante aos já apresentados anteriormente. Após a inicialização do I^2C envia-se o endereço do *slave* em modo de escrita e envia-se também o registo que se pretende ler (0x00 para o registo de temperatura). Em seguida reinicia-se o I^2C e envia-se o endereço em modo de leitura e procede-se à leitura dos dois bytes que contém os dados pretendidos.

É possível configurar o sensor de forma a selecionar qual a resolução que se pretende, bem como, outras opções disponíveis. Para isso é utilizado o registo 0x01 e caso se pretenda a resolução máxima envia-se o valor 0x60.

Para obter o valor da temperatura em ºC utiliza-se a equação 4.8 para obter os valores desejados.

$$temperatura(^{\circ}C) = valor\_lido * 0.0625 \tag{4.8}$$

4.4.5 *LIS331DLH*

No caso do acelerómetro foi utilizado um já existente no departamento, o que facilita a sua utilização devido ao conhecimento prévio existente. Foi usado então o *LIS331DLH* desenvolvido pela *STMicroeletronics*. Este acelerómetro de baixo consumo, dispõe de um interface digital SPI ou I^2C, uma resolução de 12 *bits* e permite medir acelerações de +-2g a +-8g. O acelerómetro gera um sinal de interrupção se a aceleração detetada ultrapassar um valor previamente programado (*threshold*). Na figura 4.24 é possível observar o acelerómetro *LIS331DLH*.

Figura 4.24: Acelerómetro *LIS331DLH*.

4.5 Baterias

A escolha da bateria a utilizar é uma decisão muito importante, pois, existem várias variáveis a ter em consideração, tais como a tensão, capacidade, corrente de carregamento e descarga da bateria. Para alimentar os dispositivos da WSN foram escolhidas duas baterias: uma para os HWR e outra para os HWED.

A razão das escolha de duas baterias deve-se ao fato de os HWRs necessitarem de uma bateria recarregável, devido ao módulo *harvesting* que se pretende implementar. No entanto, para os HWEDs esta caraterística da bateria não é um requisito, podendo-se dar mais importância à capacidade da mesma.

4.5.1 *SAFT VHT AA*

As baterias escolhidas para os HWRs foram as *SAFT VHT AA* (figura 4.25), apresentando boas caraterísticas para a solução proposta.

Figura 4.25: Bateria *SAFT VHT AA*.

As principais caraterísticas desta bateria podem ser observadas na tabela 4.11. Como o sistema necessita de uma alimentação de aproximadamente 3V utilizou-se três pilhas em série de forma a obter uma tensão resultante de 3.6V e uma capacidade de 1100 mA-h. A bateria descrita suporta temperaturas dos -10 aos +55ºC e, devido ao uso de carregamento solar, é necessário que as mesmas sejam recarregáveis.

Tabela 4.11: Caraterísticas da *SAFT VHT AA*.

Caraterísticas elétricas	
Alimentação (V)	1.2
Capacidade (mA-h)	1100
Dimensões	
Diâmetro (mm)	13.9
Altura (mm)	48.9
Condições de carregamento	
Permanente (mA)	30 a 55

4.5.2 *SAFT LS 17500*

Para alimentar os HWEDs foi utilizada a bateria *SAFT LS 17500* [53] da *Saft Batteries*, que se destaca da anterior pela sua capacidade superior. No entanto, estas baterias não são recarregáveis, não podendo ser usadas nos HWRs. Apresentam uma tensão aos seus terminais de 3.6V e uma capacidade de 3400 mA-h. Tem uma gama de temperatura de funcionamento elevada, dos -60ºC aos +85ºC e apresenta também um tamanho reduzido (5cm x 1,7cm), ideal para este tipo de sistemas. Na figura 4.26 é possível observar a bateria em questão.

Figura 4.26: Bateria *SAFT LS 17500*.

4.6 *IAR Embedded Workbench IDE*

Este IDE é um ambiente de desenvolvimento de *software* para microprocessadores de baixo consumo que, devido à sua pequena capacidade de memória e consumos, necessitam de funcionalidades muito bem definidas. Com o uso deste

ambiente é possível desenvolver código em várias linguagens de programação, tais como C, C++ ou *assembly*, para um extensa variedade de microprocessadores. Esta *toolchain* tem integrada ainda algumas ferramentas importantes que ajudam no desenvolvimento de projetos: um compilador C/C++, um assemblador, um *linker*, um editor, um gestor de projetos com o *Make utility* e o *debugger IAR C-SPY*.

Dependendo da arquitetura de microprocessadores que se pretende utilizar, o *IAR* dispõe uma vertente específica. Nesta dissertação é utilizada a versão *IAR Embedded Workbench* para 8051, pois os microprocessadores da *Texas Instruments* são compatíveis com a arquitetura *Intel* 8051.

4.7 *Z-Stack*

A *Z-Stack* [54] consiste num conjunto de APIs desenvolvidas pela *Texas Instruments*, baseadas no protocolo de comunicações *ZigBee*, que tem como objetivo desenvolver, de um modo simples, todas as funcionalidades desejadas neste projeto. A *Z-Stack* é destinada a soluções *low-power*, *low-cost*, *low data rate* e *highly reliable*.

É composto por sete camadas com funcionalidades diferentes, como é possível observar na figura 4.27, nomeadamente: ZMAC, NWK, AF, ZDO, APS, OSAL e HAL.

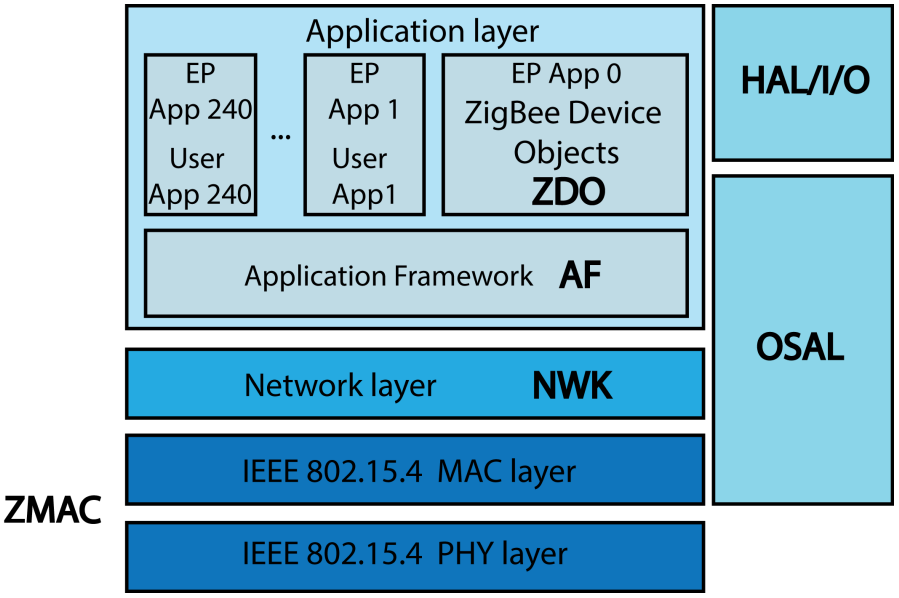


Figura 4.27: Arquitetura da *Z-Stack*.

A camada ZMAC disponibiliza um interface entre a norma *IEEE 802.15.4*

e a camada de rede (NWK), que fornece serviços de dados e informações sobre a rede para as camadas superiores. Englobadas na camada de aplicação tem-se a camada AF e a ZDO. A *Application Framework* está responsável pela gestão dos EPs (*EndPoints*) e pelo envio e receção dos dados. Para cada EP é necessário efetuar um registo do dispositivo na AF através das funções disponibilizadas pela *Z-Stack*. Desta forma, é possível definir o tipo de endereçamento que se pretende utilizar, o número do EP, definir a versão do dispositivo, o ID do mesmo e o número de *clusters*. A estrutura da trama enviada via OTA (*Over-The-Air*) está representada na figura 4.28. Como é possível observar na imagem esta é constituída por 13 campos, sendo que 12 deles são relativos à rede e o último (*AppData*) é relativo aos dados enviados. Esta *AppData* é composta por uma estrutura que contém os seguintes dados: (*a*) número de sequência do dispositivo, (*b*) tamanho da trama e (*c*) dados transmitidos.

16bits	16bits	struct	16bits	8bits	8bits	8bits	8bits	8bits	8bits	32bits	8bits	n bits
groupId	clusterId	srcAddr	macDestAddr	endPoint	wasBroadcast	LinkQuality	Correlation	RSSI	SecurityUse	timestamp	nwkSeqNum	AppData

Figura 4.28: Estrutura da trama enviada via OTA.

A camada ZDO dispõe um conjunto de APIs ao nível da aplicação para controlar e monitorizar, através do ZDP (*ZigBee Device Profile*), os seguintes serviços:

- Descoberta de dispositivos e serviços;
- HWED/HWR *Bind* e *Unbind*;
- Serviços de gestão da rede;
- Inicialização da rede.

Existem várias opções disponíveis para a ZDO baseadas nas opções de compilação definidas e no tipo de dispositivo. Na tabela 4.12 está representado as opções suportadas para os diversos estados de cada dispositivo, bem como o seu significado.

Usando como exemplo o *DEV_NWK_ORPHAN*, esta opção é utilizada quando o dispositivo perdeu a conexão com o seu pai, isto é, por alguma razão deixou de estar no raio de comunicação do mesmo, ou houve uma avaria. Através desta *flag* é possível programar o dispositivo para tentar criar uma conexão com outro dispositivo.

Por fim tem-se as camadas OSAL (*Operating System Abstraction Layer*) e HAL (*Hardware Abstraction Layer*). Estas camadas são utilizadas para separar o desenvolvimento de H/W do desenvolvimento de S/W. A camada OSAL fornece mecanismos de *scheduling* (*tasks* e eventos), gestão de memória e funcionalidades para as mensagens. Devido às suas limitações, o OSAL não é considerado exa-

Tabela 4.12: Estados de cada dispositivo disponíveis pela camada ZDO.

Opção	Definição
DEV_HOLD	Inicializado - não começou de forma automática
DEV_INIT	Inicializado - não está conetado a nada
DEV_NWK_DISC	À descoberta de PANs para se conetar
DEV_NWK_JOINING	A conetar-se a uma PAN
DEV_NWK_REJOIN	Reconetar a uma PAN (Apenas para HWED)
DEV_END_DEVICE_UNAUTH	Conetado mas ainda não foi autenticado
DEV_END_DEVICE	Iniciado como dispositivo depois da autenticação
DEV_ROUTER	Dispositivo conetado, autenticado e é um *router*
DEV_ZB_COORD	Iniciado como coordenador da rede
DEV_NWK_ORPHAN	Dispositivo perdeu informação sobre o seu pai

tamente um sistema operativo. A camada HAL fornece uma fácil programação do *hardware* isolando o *software* das especificidades do mesmo. Por último tem-se a ISR (*Interrupt Service Routine*), responsável pelos serviços de interrupção. Na figura 4.29 é possível observar de que forma a *Z-Stack* e a aplicação interagem com estas camadas.

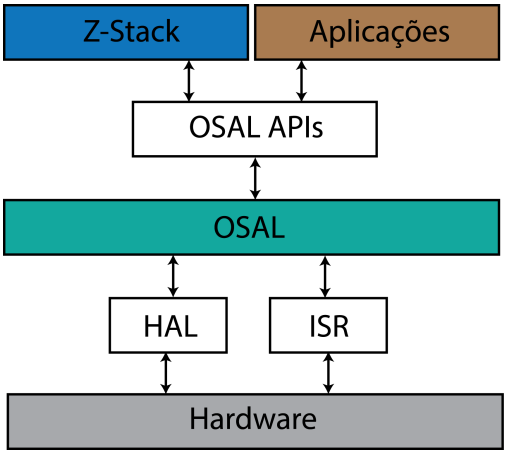


Figura 4.29: Interação da *Z-Stack* e da aplicação com as camadas OSAL e HAL.

4.8 Conclusão

No decorrer deste capítulo foram apresentadas as caraterísticas mais importantes do sistema a implementar. Devido à modularidade do hardware pretendida e da funcionalidade de *plug-and-play*, será necessário desenvolver vários módulos com funcionalidades diferentes, dividindo assim o sistema *HighWayMon* em vários subsistemas, o que simplifica de alguma forma o desenvolvimento do sistema.

No capítulo 5 será possível observar os módulos desenvolvidos, e também, de que forma foram implementadas as funcionalidades pretendidas.

Capítulo 5

Implementação

Este capítulo está dividido em três secções que, em conjunto, demonstram a implementação do projeto *HighWayMon*, sendo estas o desenvolvimento do *hardware*, a modulação do sistema através de UML e as funcionalidades desenvolvidas no *software*.

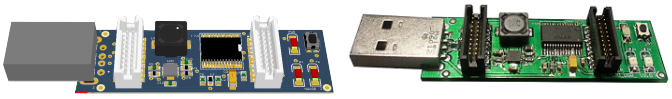
Na componente de *hardware* são apresentados todos os módulos desenvolvidos, os dispositivos finais e os sensores utilizados. Através de modelos UML é feita uma descrição de como o sistema funciona, apresentando através de diagramas de sequência, as interações efetuadas de forma a implementar as principais funcionalidades. Na componente de *software* é apresentado de que forma estas funcionalidades foram implementadas.

5.1 *Hardware*

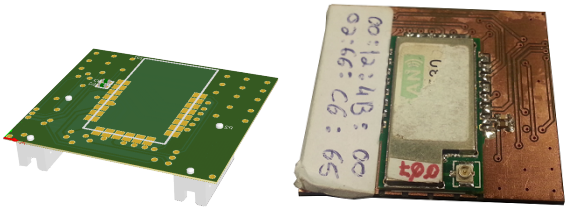
Nesta seção serão apresentados todos os módulos desenvolvidos para a presente dissertação, tendo em conta as especificações descritas no capítulo anterior. Para a implementação do hardware em questão foi utilizada a ferramenta CAD *Altium Designer* para desenvolver as PCBs pretendidas. Para além das PCBs desenvolvidas será apresentado o resultado final de cada dispositivo. Todos os módulos foram desenvolvidos e soldados na Universidade do Minho, no Grupo de Sistemas Embebidos.

5.1.1 Módulos

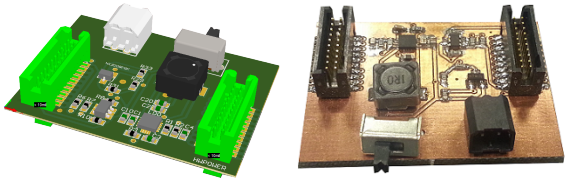
Tal como referido anteriormente, todos os módulos foram desenhados utilizando a ferramenta *Altium Designer*, proporcionando um ambiente capaz de implementar todos os requisitos necessários para o desenvolvimento das PCBs.



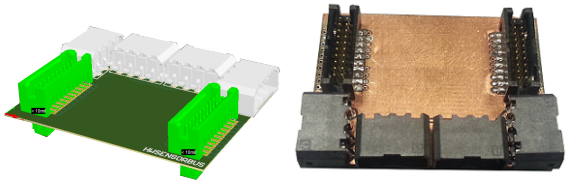
(a) HWUSB: Modelo 3D e PCB final.



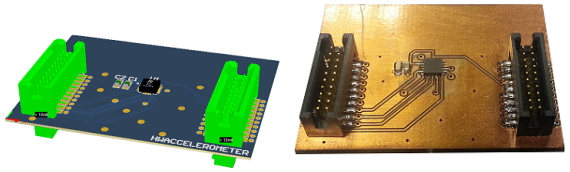
(b) HWCC2530: Modelo 3D e PCB final.



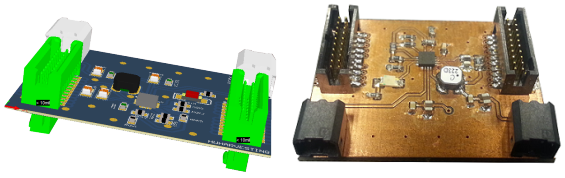
(c) HWPOWER: Modelo 3D e PCB final.



(d) HWSENSORBUS: Modelo 3D e PCB final.



(e) HWACCELEROMETER: Modelo 3D e PCB final.



(f) HWHARVESTING: Modelo 3D e PCB final.

Figura 5.1: Módulos desenvolvidos.

Após desenhadas as PCBs foi necessário produzir as mesmas, utilizando as máquinas disponíveis no departamento para a realização dessa tarefa. Em seguida foi necessário soldar todos os componentes. Para isso foram utilizadas as estações de soldar disponíveis no laboratório do grupo de Sistemas Embebidos. Em

alguns casos, mais concretamente para os ICs *TPS63060*, *LIS331DLH*, *SHT21*, *MAX44009* e *BMP180*, foi necessário utilizar o processo de *reflow* para soldar estes componentes à respetiva PCB, devido aos seus pinos estarem por baixo do IC.

Na figura 5.1 é possível observar todos os módulos desenvolvidos. Ao lado esquerdo encontra-se o modelo 3D da PCB obtido no *Altium Designer* e no lado direito o resultado final da mesma com todos os componentes já soldados.

5.1.2 Sensores

De forma a adquirir as informações atmosféricas foi necessário criar pequenos módulos com os sensores que foram escolhidos. No total foram desenvolvidos 4 módulos, um para cada tipo de sensor (THE, LE, PE e TS). Em cada um deles foi utilizado um conetor de forma a conetar o módulo respetivo com o *HWSENSORBUS*. Na figura 5.2 é possível observar todos os módulos de sensores desenvolvidos.

(a) THE **(b)** LE **(c)** PE **(d)** TS

Figura 5.2: Módulos dos sensores desenvolvidos.

5.1.3 Dispositivos Finais

Utilizando os módulos desenvolvidos é possível criar vários dispositivos diferentes, por exemplo, pode não ser necessário utilizar o acelerómetro e, nesse caso, não é preciso utilizar o módulo respetivo. Na figura 5.3 é possível observar o modelo 3D de todos os módulos que se podem utilizar nos dispositivos HWED e HWR.

Após desenvolvidos todos os módulos foi possível juntar os que se pretendiam utilizar, de forma a montar o dispositivo final. Na figura 5.4 são apresentados dois dos dispositivos já com os módulos agrupados. Na figura 5.4a está o HWC com dois dos módulos desenvolvidos, o *HWUSB* e o *HWCC2530*. Na figura 5.4b encontra-se o HWED com os seguintes módulos: *HWPOWER*, *HWSENSORBUS*, *HWACCELEROMETER* e *HWCC2530*.

O hardware foi totalmente testado e validade, estando a ser utilizado no projeto SustIMS.

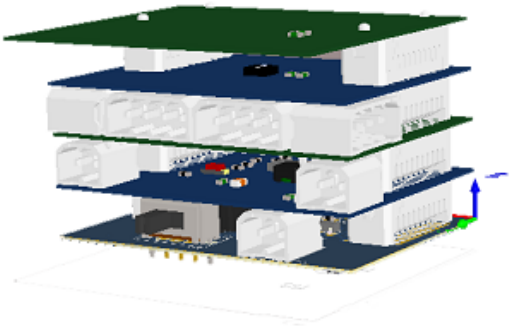


Figura 5.3: Modelo 3D do dispositivo HWED/HWR.

(a) HWC **(b)** HWED

Figura 5.4: Resultado final dos dispositivos desenvolvidos.

5.2 UML

Nesta secção serão apresentados modelos dos principais componentes da WSN e dos dispositivos que a compõe. Será focada a criação e descoberta de redes WPAN, bem como, os eventos utilizados na presente dissertação.

5.2.1 Criação da WPAN

Na figura 5.5 está representado o diagrama de sequência para a criação de uma WPAN. Este processo pode apenas ser efetuado pelo coordenador e tem como objetivo a criação da rede WPAN para que seja possível outros dispositivos conetarem-se a ela. No diagrama é representado quais as tarefas de cada camada para a criação da rede.

A atividade da camada ZDO começa com um pedido para formar a rede para a camada NWK, que por sua vez, faz um pedido à camada MAC para fazer um *scan* passivo de forma a detetar os canais (realizada pela camada PHY). Após a confirmação da camada PHY e MAC, a camada NWK pede um *scan* ativo de forma a abrir os canais. Quando esta ação é confirmada a aplicação é notificada

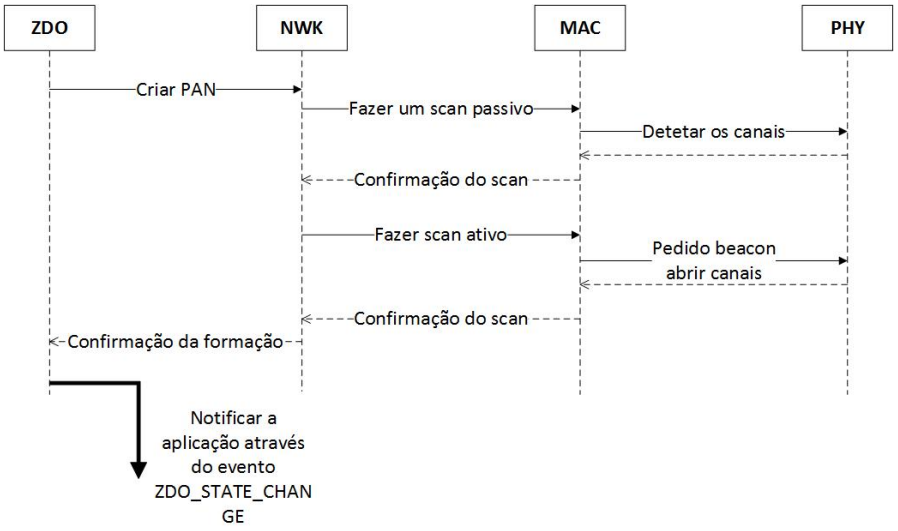


Figura 5.5: Diagrama de sequência da formação de uma WPAN.

através do evento *ZDO_STATE_CHANGE*.

A definição da PAN ID e do canal a ser usado pode ser feita de maneira autónoma ou pode ser definida previamente, pelo programador.

5.2.2 Descoberta da WPAN

Após a rede estar criada, os dispositivos podem conetar-se à mesma. Na figura 5.6 está representado o processo de descoberta da WPAN através de um diagrama de sequência, com as ações de cada camada da *stack*. Em primeiro lugar é necessário analisar se existe alguma rede a que se possa ligar, isto é, se existe uma rede com um PAN ID compatível e se existe um canal disponível para utilizar. Após a confirmação da existência de uma rede, é enviado o pedido de *join* para que o dispositivo consiga conetar-se à mesma. Neste processo é feita uma associação (*binding*) entre os dispositivos e registada numa tabela que contém todas as associações do dispositivo.

Na presença de *HWRs*, estes podem ser intermediários entre o HWC e os HWEDs. Nesse caso essa associação é feita através do router.

5.2.3 Processamento de eventos

Os HWEDs e HWRs são compostos por dois eventos semelhantes mais ativados de forma diferente. O evento síncrono, tal como o nome indica, ocorre de forma regular, sendo para isso utilizado um temporizador que assinala a taxa de amostragem da *SAMPLE* a adquirir, isto é, a aquisição das variáveis físicas pre-

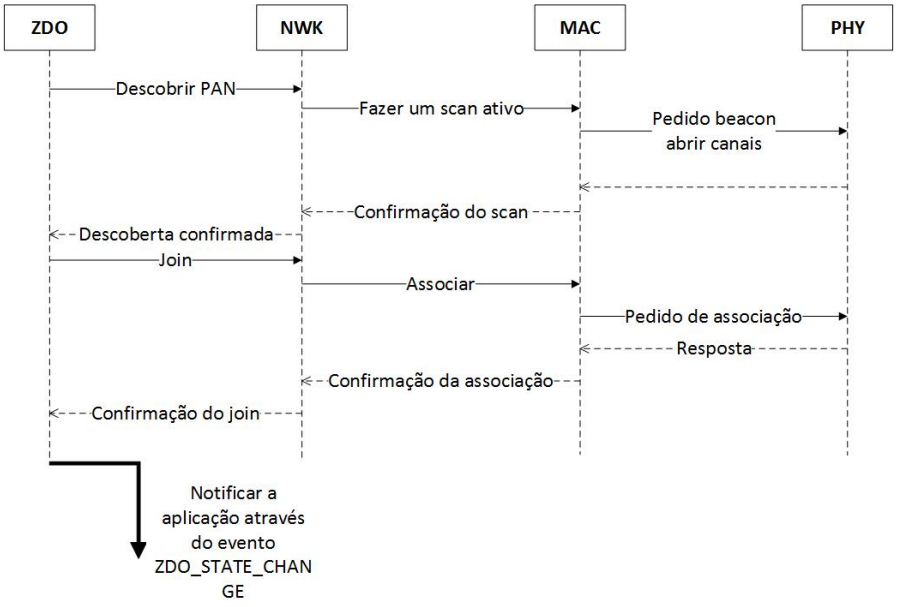


Figura 5.6: Diagrama de sequência da descoberta de uma WPAN.

tendidas. No entanto, o evento assíncrono é ativado através de uma interrupção do acelerómetro quando forem detetadas acelerações não admitidas pelo sistema.

Na figura 5.7 é possível observar o processamento efetuado em ambos os eventos. Apesar de estes eventos terem funcionalidades diferentes, o seu processamento é muito semelhante. Após o evento ter sido ativado, são feitas todas as leituras necessárias aos sensores que compõem o dispositivo, bem como o seu nível de bateria. No caso do evento assíncrono, para além destas informações, é enviado o valor em módulo dos três eixos da aceleração detetada. Após essa informação ser adquirida é enviada a estrutura de todos os dados através do protocolo WPAN para o HWC, que irá receber essa informação, processá-la e enviá-la para o HWG através do protocolo porta série.

5.2.4 Funcionalidades do HWC

O comportamento do HWC pode ser dividido em duas partes: a inicialização do dispositivo e a parte ativa após este ter sido inicializado (como é possível observar na figura 5.8). Na primeira fase é feito o registo do dispositivo na camada AF, sendo um processo comum para todos os dispositivos que queiram fazer parte da rede. Em seguida é inicializado o protocolo porta série e criada a rede WPAN. Na fase ativa do dispositivo podem acontecer dois casos diferentes. O primeiro passa pela conexão dos dispositivos que se querem conetar à rede, é feito um pedido por parte dos mesmos e se forem reunidas as condições para se efetuar uma ligação, os

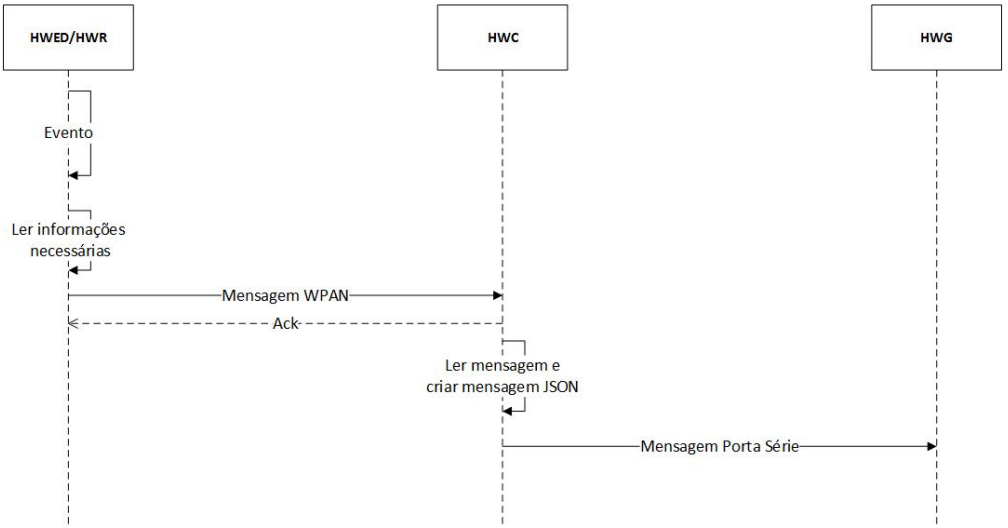


Figura 5.7: Diagrama de sequência do processamento de eventos síncronos e assíncronos.

dispositivos passam a fazer parte da rede criada, sendo esta funcionalidade gerida pela *Z-Stack*. O segundo caso é a receção de mensagens por parte dos dispositivos HWED/HWR. As mensagens recebidas são processadas e enviadas por porta série no formato JSON.

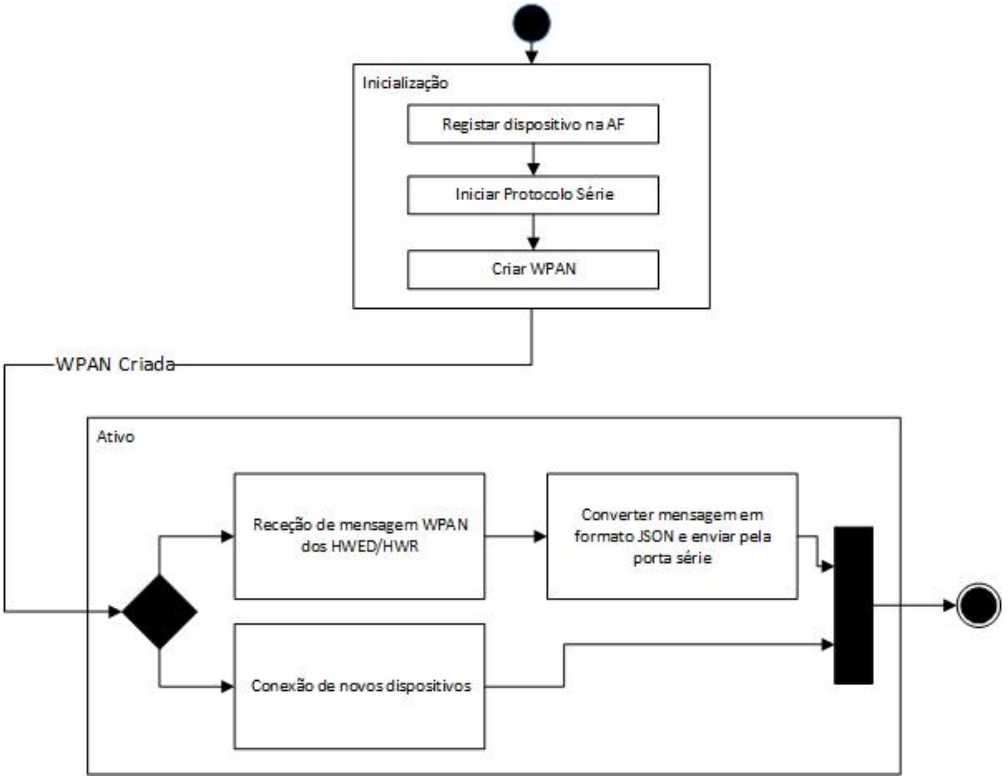


Figura 5.8: Diagrama de atividades do HWC.

5.2.5 Funcionalidades do HWED

A figura 5.9 tem como objetivo representar as funcionalidades do HWED. Tal como o HWC pode ser dividido em duas componentes, a inicialização do dispositivo e o seu modo ativo.

De forma a inicializar estes dispositivos é necessário o seu registo na camada AF, a configuração da sua alimentação como sendo a bateria (o dispositivo pode adormecer) ou, no caso dos HWR, como estando sempre ligados. É também feita a inicialização das interrupções, do ADC, do acelerómetro e do barramento de I^2C. Por último, os dispositivos tentam conetar-se à rede através de um pedido de *join*.

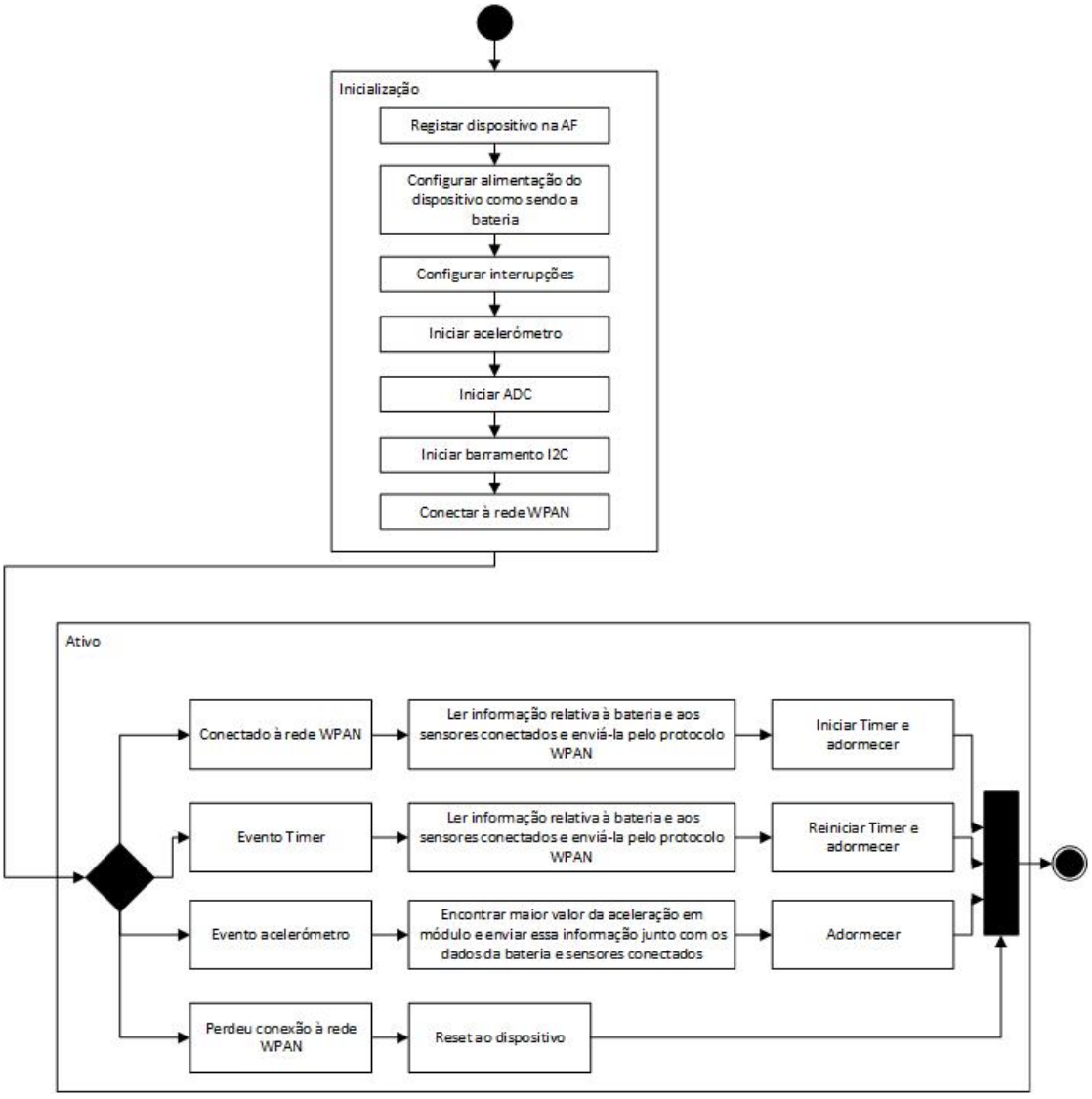


Figura 5.9: Funcionalidades do HWED.

No seu modo ativo existem quatro casos que em conjunto podem definir o

funcionamento destes dispositivos. Após conexão à rede, será feita uma leitura dos sensores conetados ao dispositivo, da bateria e serão enviados esses dados para o HWC através do protocolo WPAN. Após este processo, é iniciado o *timer* responsável pelos eventos síncronos e o dispositivo entra em modo *sleep*. Nos eventos do *timer* este processo é repetido, sendo para isso necessário reiniciar a contagem do mesmo. No caso dos eventos assíncronos (interrupção por parte do acelerómetro), é encontrado o máximo valor da aceleração em módulo e, em conjunto com os dados dos sensores conetados e da bateria, é enviada essa informação pelo protocolo WPAN. Após o envio, o dispositivo volta ao estado de *sleep*. O último caso tem como objetivo definir o comportamento do dispositivo quando este perde a ligação com o seu "pai". No caso disto acontecer, é feito um *reset* do dispositivo até ser possível estabelecer uma ligação.

5.3 *Software*

A presente secção é referente ao *software* desenvolvido no decorrer do projeto de forma a suportar todas as funcionalidades requeridas. Serão apresentadas as principais implementações realizadas para suportar os requisitos.

O *software* está dividido em duas tarefas: *AP_Task* e *ED_Task*. A primeira é referente ao HWC e tem como objetivo a criação da WSN com todas as configurações necessárias, a receção dos dados recolhidos pelos HWED e HWR e a criação de uma mensagem em formato JSON enviando-o pela porta série. A *ED_Task* suporta todos os dispositivos que possam adquirir informação sensorial, mais concretamente o HWED e o HWR. Um dos principais objetivos no desenvolvimento do *software* era dar suporte ao *plug-and-play* conseguido através da modularidade do *hardware*, isto é, utilizar o mesmo *firmware* para todas as combinações possíveis num dispositivo. Desta forma, é apenas necessário uma versão do *firmware* independentemente dos sensores utilizados.

Em seguida será feita uma descrição mais detalhada de cada tarefa, explicando de que forma são implementadas as funcionalidades pretendidas.

5.3.1 *AP Task*

A tarefa do coordenador é composta por dois métodos essenciais para qualquer tarefa OSAL: um para efetuar todas as inicializações necessárias e outro para lidar com os eventos que possam suceder.

O método de inicialização implementado tem como objetivo registar o dis-

positivo na AF da *Z-Stack*, neste caso como coordenador, inicializar o dispositivo, iniciar o protocolo série e definir o nível de transmissão do TX. A função denomina-se *AccessPoint_task_Init* e tem como argumentos o ID da *task*.

Listagem 5.1: Método responsável por inicializar a tarefa do coordenador.

```
1  void AccessPoint_task_Init( byte task_id ){
2    taskID = task_id;
3
4    //1. Register application in ZStack AF
5    AF_task_registration_register(&taskID);
6    zgDeviceLogicalType = ZG_DEVICETYPE_COORDINATOR;
7    ZDOInitDevice(0);
8    //2. Register Serial Port
9    uart_init();
10   //3. Set TX power to the maximum
11   ZMacSetTransmitPower(TX_PWR_PLUS_19);
12   printf("HWC Initialized in PANID=0x%x\n",ZDAPP_CONFIG_PAN_ID)
       ;
13   return;
14 }
```

Para lidar com os eventos que possam suceder, é utilizada a função *AccessPoint_task_ProcessEvent* de forma a processar corretamente qual a melhor forma de lidar com eles. Podem ser definidos mais 15 eventos para além do obrigatório (*SYS_EVENT_MSG*). Nesta tarefa não são necessários mais eventos, pois, apenas existem eventos relativos a mensagens (recebidas neste caso). Quando este evento é ativado existem várias opções possíveis dependendo se recebeu uma mensagem, ou enviou uma mensagem, ou se houve uma mudança no estado da rede, entre outras. No caso do HWC apenas existe a opção de receção de mensagem dentro de um *switch* (*AF_INCOMING_MSG_CMD*), que irá processar a mensagem recebida e enviar toda a informação no formato JSON pela porta série.

Listagem 5.2: Função responsável por processar os eventos do HWC.

```
1  UINT16 AccessPoint_task_ProcessEvent( byte task_id, UINT16
       events ){
2    volatile afIncomingMSGPacket_t *MSGpkt;
3      if ( events & SYS_EVENT_MSG ){
4      MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive( taskID
         );
5      while ( MSGpkt != NULL){
6        switch ( MSGpkt->hdr.event ){
7        case AF_INCOMING_MSG_CMD:
```

```
8           printJsonString((afIncomingMSGPacket_t *)MSGpkt, (
                appProto *) MSGpkt->cmd.Data);
9           break;
10        }
11        osal_msg_deallocate( (uint8 *)MSGpkt );
12        MSGpkt = (afIncomingMSGPacket_t *)osal_msg_receive(
              taskID );
13      }
14      return (events ^ SYS_EVENT_MSG);
15    }
16    return 0;
17  }
```

Configuração da WPAN

Através do ficheiro de configuração *f8wconfig.cfg* é possível definir previamente qual o valor a atribuir à PAN ID que se pretende criar. Para isso apenas é necessário alterar o valor da variável *ZDAPP_CONFIG_PAN_ID*. Por omissão o valor é 0xFFFF, o que significa que ao alterar este valor, os *HWEDs* e *HWRs* terão de ter o valor da PAN ID alterado para se poderem conetar à rede. Na figura 5.10 é possível observar a parte do ficheiro de configuração correspondente à definição do ID da rede.

```
/* Define the default PAN ID.
 *
 * Setting this to a value other than 0xFFFF causes
 * ZDO_COORD to use this value as its PAN ID and
 * Routers and end devices to join PAN with this ID
 */
-DZDAPP_CONFIG_PAN_ID=0xDCAD
```

Figura 5.10: Definição do PAN ID da rede.

Configuração do canal

No ficheiro de configuração é também possível definir qual o canal que se pretende utilizar. Existem no total 26 canais, no entanto, como a gama pretendida é de 2.4 GHz, é possível usar desde o canal 11 ao 26 sendo que por omissão é usado o canal 11 (0x0B), tal como é apresentado na figura 5.11.

```
/* Default channel is Channel 11 - 0x0B */
// Channels are defined in the following:
//         0      : 868 MHz     0x00000001
//         1 - 10 : 915 MHz     0x000007FE
//        11 - 26 : 2.4 GHz     0x07FFF800
//
//-DMAX_CHANNELS_868MHZ     0x00000001
//-DMAX_CHANNELS_915MHZ     0x000007FE
//-DMAX_CHANNELS_24GHZ      0x07FFF800
//-DDEFAULT_CHANLIST=0x04000000  // 26 - 0x1A
//-DDEFAULT_CHANLIST=0x02000000  // 25 - 0x19
//-DDEFAULT_CHANLIST=0x01000000  // 24 - 0x18
//-DDEFAULT_CHANLIST=0x00800000  // 23 - 0x17
//-DDEFAULT_CHANLIST=0x00400000  // 22 - 0x16
//-DDEFAULT_CHANLIST=0x00200000  // 21 - 0x15
//-DDEFAULT_CHANLIST=0x00100000  // 20 - 0x14
//-DDEFAULT_CHANLIST=0x00080000  // 19 - 0x13
//-DDEFAULT_CHANLIST=0x00040000  // 18 - 0x12
//-DDEFAULT_CHANLIST=0x00020000  // 17 - 0x11
//-DDEFAULT_CHANLIST=0x00010000  // 16 - 0x10
//-DDEFAULT_CHANLIST=0x00008000  // 15 - 0x0F
//-DDEFAULT_CHANLIST=0x00004000  // 14 - 0x0E
//-DDEFAULT_CHANLIST=0x00002000  // 13 - 0x0D
//-DDEFAULT_CHANLIST=0x00001000  // 12 - 0x0C
-DDEFAULT_CHANLIST=0x00000800  // 11 - 0x0B
```

Figura 5.11: Definição do canal utilizado.

Protocolo Porta Série

Após uma mensagem ser recebida e o evento ser despoletado é necessário processar a mensagem de forma a enviar as informações pela porta série no formato pretendido (JSON). Para isso é utilizada a função *printJsonString()* que consiste no print desses valor já no formato correto. O protocolo é dividido em duas partes: as informações relativas à rede (função *ntw_protocol_printJSONString*) e as informações da aplicação (função *app_protocol_printJSONString*). No código apresentado na listagem seguinte é possível observar de que forma a informação da rede é impressa.

Listagem 5.3: Processamento da informação relativa à rede

```
1  sprintf(buff,"\"NTW\" : {");
2  puts(buff);
3  sprintf(buff,"\"WPANID\":\"0x%X\",", ZDAPP_CONFIG_PAN_ID);
4  puts(buff);
5  sprintf(buff,"\"GROUPID\":\"%d\",", pkt->groupId);
6  puts(buff);
7  sprintf(buff,"\"CLUSTERID\":\"%d\",", pkt->clusterId);
8  puts(buff);
9  sprintf(buff, "\"GATEWAYMAC\":\"%s\",", ntw_MacAddress_toString
       (macaddr , mac_str) );
10 puts(buff);
11 sprintf(buff, "\"ENDPOINT\":\"%d\",", pkt->endPoint );
12 puts(buff);
13 sprintf(buff, "\"BROADCAST\":\"%d\",", pkt->wasBroadcast );
14 puts(buff);
```

```
15 sprintf(buff, "\"LINKQUALITY\":\"%d\",", pkt->LinkQuality );
16 puts(buff);
17 sprintf(buff, "\"CORRELATION\":\"%d\",", pkt->correlation );
18 puts(buff);
19 sprintf(buff, "\"RSSI\":\"%d\",", pkt->rssi );
20 puts(buff);
21 sprintf(buff, "\"SECURITY\":\"%d\",", pkt->SecurityUse );
22 puts(buff);
23 sprintf(buff, "\"SEQNUMBER\":\"%d\"", pkt->nwkSeqNum );
24 puts(buff);
25 //End the JSON Message
26 sprintf(buff,"}");
27 puts(buff);
```

No caso da informação relativa à aplicação o processo é semelhante, no entanto, a informação relativa aos sensores é tratada de forma diferente. Para cada tipo de informação é testado através do *pck_id* da mensagem se a informação dessa variável física está nos dados enviados. Desta forma, caso um sensor não esteja conetado, não são enviadas as informações relativas a esse sensor. Devido à sua extensão, o código relativo aos sensores não está representado, no entanto, é semelhante ao da bateria pelo que é possível compreender o funcionamento do mesmo.

Listagem 5.4: Processamento da informação relativa à aplicação

```
1  //0. Clean up the buff
2  puts("\"APP\" : {");
3  //Buff the MAC Address & Sampling Rate & NodeType
4  sprintf(buff, "\"DEVICEMAC\":\"%s\",",
5          ntw_MacAddress_toString(appPacket->macaddr, mac));
6  puts(buff);
7  //Buff the Sampling Rate
8  sprintf(buff, "\"SAMPLING_RATE\":\"%d\",",
9          appPacket->samplingRate_sec);
10 puts(buff);
11 //Buff the NodeType
12 sprintf(buff, "\"DEVICETYPE\":\"%d\"",
13         appPacket->nodeType);
14 puts(buff);
15 missing_comma = true;
16 //Buff the battery
17 if ( appPacket->pck_id & BATTERY ){
18   CONTROL_MISSINGCOMMA(missing_comma, buff);
19   sprintf(buff, "\"BATTERY\":\"%f\"",
```

```
20            appPacket->batt.battery );
21    puts(buff);
22  }
23  //Buff the acceleration
24    (...)
25  //Buff the Temperature & Humidity
26    (...)
27  //Buff the Luminosity
28    (...)
29  //Buff the Temperature & Pressure
30    (...)
31  //Buff the TEMPERATURE
32    (...)
33  //End the JSON Message
34  puts("}");
```

5.3.2 *ED Task*

À semelhança do *software* desenvolvido para o HWC, o *ED_Task* é também composto por duas funções principais por onde passa a inicialização do sistema e processamento de eventos. Através da função *EndDevice_task_Init* é feito o registo da aplicação na AF, adquirido o valor do endereço MAC do dispositivo, a inicialização da UART, configurada a interrupção para o acelerómetro bem como a inicialização do mesmo, configurado o ADC e inicializado o barramento I^2C. No caso da função *EndDevice_task_ProcessEvent* é feito o processamento de todos os eventos que possam ocorrer no dispositivo. Tal como referido na tarefa do HWC, é possível criar até 15 eventos manualmente. Ao desenvolver o *firmware* optou-se por utilizar quatro eventos para além do obrigatório, que serão responsáveis por processar a informação dos sensores do dispositivo, tanto para o acelerómetro como para os sensores de monitorização ambiental.

Na listagem seguinte é apresentada uma parte da função que tem como objetivo tratar dos eventos relativos ao dispositivo.

Listagem 5.5: Função responsável por inicializar a tarefa dos End-devices.

```
1  switch ( MSGpkt->hdr.event ){
2    case AF_DATA_CONFIRM_CMD:
3         if ( ((afDataConfirm_t *) MSGpkt)->hdr.status != 0 ){ //
               No Mac Acknowledge received
4             if(nNetworkRetry-- == 0)
5               SystemResetSoft();
6             } else
```

```
7                 nNetworkRetry = NETWORK_PKG_RETRY;
8               break;
9     case ZDO_STATE_CHANGE:
10          ntw_state = (devStates_t)(MSGpkt->hdr.status);
11          switch(ntw_state){
12              case DEVICE_TYPE:
13                   SERIAL_APP_PRINT("Network Initilized\n");
14                   sensorSampling();
15                   periodicSamplingStart();
16                   halDigioIntEnable(&accPin_cnf);
                        halDigioIntEnable(&assynKeepAlive_cnf);
17                   PWR_MEASUREMENT_KEEPALIVE_START();  //Macro for
                        use cases power consumption
18              break;
19              case DEV_NWK_ORPHAN:
20                   SystemResetSoft(); //Reboot system
21                   break;
22              default:
23                   SERIAL_APP_PRINT("Network Not Connected\n");
24                   periodicSamplingStop();
25                   halDigioIntDisable(&accPin_cnf);
                        halDigioIntEnable(&assynKeepAlive_cnf);
26              break;
27          }
28    }
```

No evento *SYS_EVENT_MSG* existem duas formas de lidar com eventos deste tipo: uma para indicador se a mensagem foi corretamente transmitida (*AF_DATA_CONFIRM_CMD*) e outra para identificar as mudanças de estado do dispositivo (*ZDO_STATE_CHANGE*). Dentro da última opção existem uma série de acontecimentos relativamente ao estado do mesmo. Através do *DEVICE_TYPE* é possível saber se o dispositivo se conetou com sucesso à rede e com a flag *DEV_NWK_ORPHAN* é possível saber quando o dispositivo fica órfão.

Para o acelerómetro são utilizados dois eventos com objetivos diferentes (*EVENT_ACC* e *EVENT_ACC_SAMPLING*) que em conjunto permitem descobrir o pico máximo do impacto em vez de apenas ler uma vez o valor. Quando a interrupção do acelerómetro é ativada pela primeira vez, é guardado o valor medido e são feitas várias amostragens posteriormente de forma a saber qual o maior valor em módulo medido. Caso o valor medido seja maior que o valor anterior, é guardado esse valor e comparado com o seguinte. Após 9 amostragens é enviado o maior valor encontrado bem como qualquer informação adicional, como valores sensoriais, bateria, entre outros. É ainda utilizado outro evento (*EVENT_ACC_REBOOT*)

de forma a reiniciar o acelerómetro, e assim, todos os valores por defeito do mesmo.

Por último é usado um evento (*EVENT_PERIODICSAMPLING*) de forma síncrona que obtém o valor dos sensores disponíveis, bem como o valor da bateria, e envia esses valores para o HWC.

Configuração do modo *sleep*

O modo *sleep* é essencial no dispositivo HWED. Isto deve-se ao fato de estes serem alimentados através de uma bateria, o que significa que quanto menor for o consumo do sistema maior será a duração da bateria e, por conseguinte, será necessária uma menor manutenção do dispositivo.

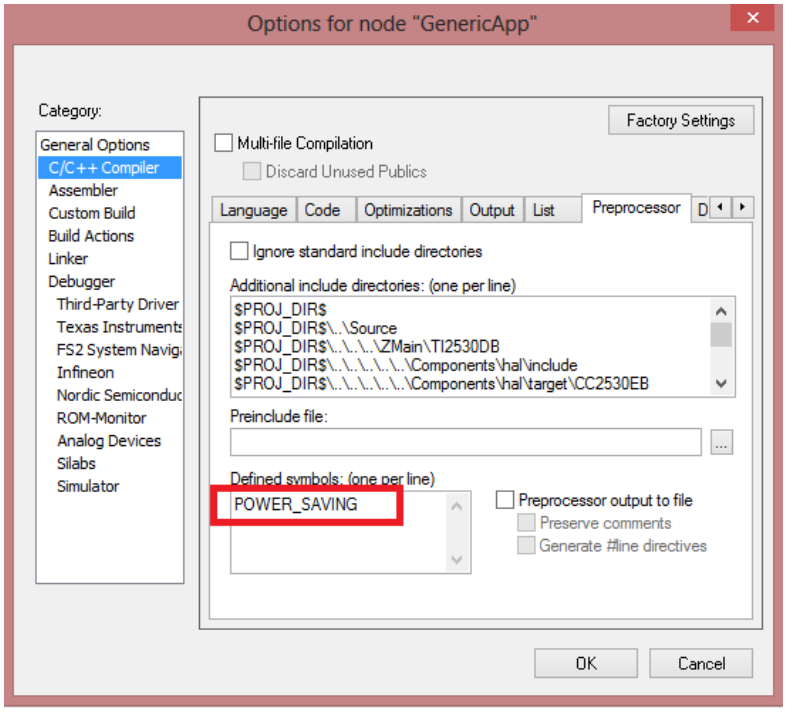


Figura 5.12: Opção de compilação para modo *sleep* do HWED.

Este modo é implementado através da opção de compilação *POWER_SAVING*, representado na figura 5.12. Quando não existe mais nenhum evento a ser processado no ciclo principal da OSAL e a funcionalidade de conservação de energia está ativada, o sistema decide quando deve "adormecer". Para isso é necessário a opção de compilação estar ativada e o RX estar desligado no estado *idle*, sendo necessário neste último caso alterar o parâmetro *RFD_RCVC_ALWAYS_ON* para *FALSE* no ficheiro de configuração *f8wConfig.cfg*.

Barramento I²C

O protocolo I²C foi totalmente implementado em *software* através da utilização de dois pinos de GPIO. A simplicidade deste protocolo, devido à utilização de apenas dois pinos, permite interligar vários dispositivos (*slaves*) num único barramento controlados por um único dispositivo (*master*). É baseado numa comunicação pergunta-resposta em que o *master* geralmente escolhe o sensor com que pretende comunicar (através do endereço do mesmo) e faz o pedido dos dados. O *slave* por sua vez responde com os dados pretendidos depois de efetuar o processo necessário para os obter.

O barramento de I²C é utilizado para comunicar com os sensores de monitorização ambiental e obter os valores que estes conseguem adquirir. Através da utilização de um pino (P0_2) do microprocessador é possível ativar ou desativar este barramento, permitindo, desta forma, um menor consumo de energia por parte dos sensores. Isto é feito através de dois *defines*, como se mostra na listagem 5.6.

Listagem 5.6: Defines responsáveis pela configuração do pino de enable.

```
1  #define I2C_SENSORS_ON(){ i2cEnable.dir = HAL_DIGIO_OUTPUT;\
2                            i2cEnable.initval = 1;           \
3                            halDigioConfig(&i2cEnable);      \
4                            DelayMs(10);                     \
5  }
6  #define I2C_SENSORS_OFF(){ i2cEnable.dir = HAL_DIGIO_INPUT;\
7                            i2cEnable.initval = 0;           \
8                            halDigioConfig(&i2cEnable);      \
9                            halDigioSetPortInputMode(&i2cEnable,
                                pulldown);\
10 }
```

Visto o número de sensores no dispositivo não ser fixo, é necessário identificar quais os sensores que o compõe, de forma a enviar para o coordenador apenas a informação estritamente necessária. Para isso, é testado se os sensores estão ligados e se a comunicação entre os sensores ocorreu da maneira correta. Será usado como exemplo o sensor SHT21 responsável por adquirir informação sobre a temperatura e humidade do ar.

Listagem 5.7: Função responsável por adquirir a informação sensorial do sensor SHT21.

```
1  static void i2cBus_SHT21(appProto * sensorValues){
2    uint8 buffTmp[3];
3    uint8 buffHd[3];
4    uint8 command_T=SHT21_TRGGER_T_CMD;//command for temperature
5    uint8 command_RH=SHT21_TRGGER_RH_CMD;//command for humidity
```

```
6    if(
7            HalI2CInitSHT21() == true &&
8            HalI2CReceiveSHT21(&command_T, buffTmp, 3) == true &&
9            HalI2CReceiveSHT21(&command_RH, buffHd, 3)
10             ){
11             sensorValues->pck_id |= TEMPERATURE_HUMIDITY;
12             sensorValues->tmphumidity.temperature = SHT21
                   GetTemperature(buffTmp);
13             sensorValues->tmphumidity.humidity = SHT21GetHumidity
                   (buffHd);
14           }
15   }
```

Caso uma das funções utilizadas retorne falso, não é guardado nenhum valor e é assumido que o dispositivo não está ligado ou que ocorreu um erro de comunicação. Caso não aconteça nenhum deste casos, tudo ocorreu na normalidade e os dados foram adquiridos. Em seguida é alterado o *bit* correspondente ao sensor de temperatura e humidade na variável *pck_id* e são feitos os cálculos para obter os valores na unidades pretendidas.

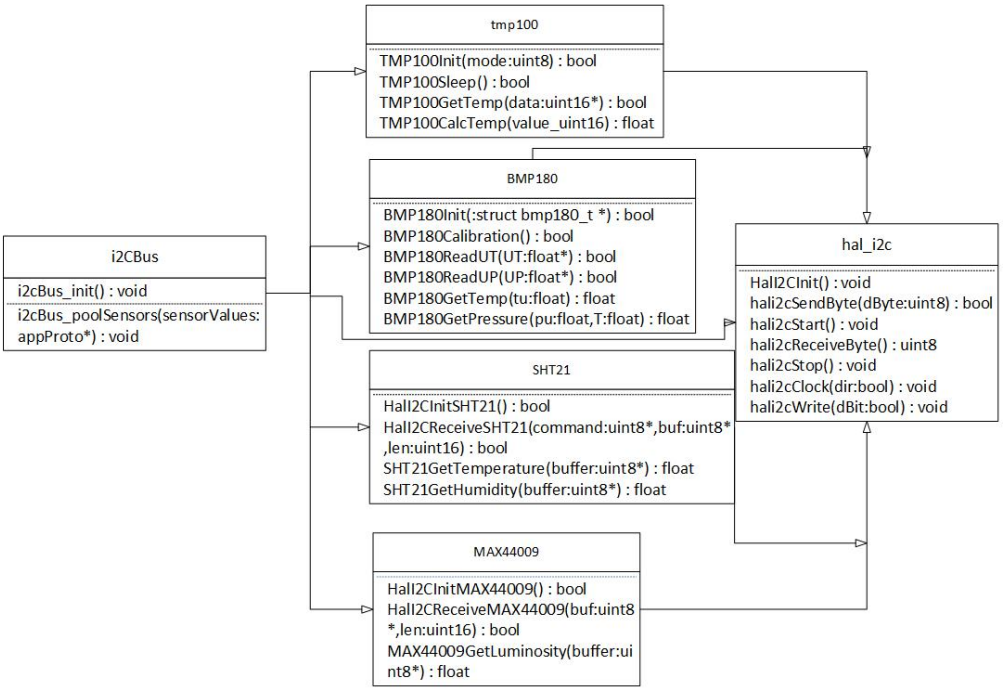


Figura 5.13: Diagrama de classes dos drivers desenvolvidos para os sensores que utilizam o barramento I^2C.

O mesmo processo é efetuado para os restantes sensores utilizando as funções respetivas. Desta forma garante-se um sistema *plug-and-play* generalizado independentemente dos sensores utilizados e minimizando os dados transmitidos caso

estes não estejam presentes.

Na figura 5.13 é apresentado o diagrama de classes dos drivers desenvolvidos para os sensores que utilizam o protocolo de comunicação I^2C.

Protocolo SPI

O protocolo SPI (*Serial Peripheral Interface*) tem como objetivo a criação de um barramento síncrono que permite a ligação de vários dispositivos a um único controlador. Através do uso do *chip select* (CS) e do *clock* (SCLK) o master consegue controlar o fluxo dos dados do barramento. A linha MOSI (*Master Output Slave Input*) é usada quando se quer que o *master* envie os dados e o *slave* os receba, enquanto que a linha MISO (*Master Input Slave Output*) tem o processo inverso.

A *Z-Stack* oferece suporte para este protocolo, o que permite utilizar as APIs existentes. Na listagem seguinte é possível observar o HAL do protocolo.

Listagem 5.8: APIs do protocolo SPI.

```
1 void spi_init( void );
2 void spi_start(uint8 cs);
3 void spi_stop(void);
4 void spi_write(uint8 * buff,uint16 len);
5 uint8 spi_read(uint8 * buff,uint16 len);
```

Protocolo WPAN

O protocolo WPAN é utilizado para transmitir a informação adquirida pelos nós sensores e enviá-la para o HWC. Para isso é utilizada uma estrutura para guardar todos os dados necessários. A definição da mesma está representada na listagem seguinte. Para cada sensor é utilizada uma estrutura própria de forma a organizar de uma melhor forma os dados adquiridos. Para além dos dados dos sensores é necessário também enviar nesta estrutura outros dados relevantes, tais como, o tempo de amostragem, o tipo de dispositivo, o nível de bateria, o endereço MAC e o *Packet Id* responsável por informar o HWC do tipo de dados presentes (relativamente aos sensores).

Listagem 5.9: Estrutura utilizada para enviar todos os dados relativos aos sensores.

```
1 typedef struct appProto_t{
2   uint16 pck_id;
3   unsigned char macaddr[8];
4   unsigned int samplingRate_sec;
```

```
5    uint8 nodeType;
6    battProto batt;
7    accProto acc;
8    tempHumidityProto tmphumidity;
9    luminosityProto light;
10   tempPressureProto tmppressure;
11   tempProto temperature;
12 }appProto;
```

Para enviar estes dados OTA (*Over-The-Air*) é utilizada a função *send_packet* que tem como argumentos um apontador do tipo *void* (que irá apontar para a estrutura) e o tamanho dos dados a enviar. A função *AF_DataRequest*, disponibilizada pela camada AF, recebe todas as informações necessárias para transmitir os dados e caso estes sejam enviados com sucesso devolve o valor 1.

Listagem 5.10: Função utilizada para transmissões OTA.

```
1  static bool send_packet(void * data, uint16 size){
2    afAddrType_t coordenator_address = {
3     NULL,
4    (afAddrMode_t)Addr16Bit,
5     ENDPOINT,
6     0x00
7    };
8   return (ntw_state == DEVICE_TYPE &&
9      AF_DataRequest(&coordenator_address,
10        AF_task_registration_GetApplicationNetworkDetails(),
11        CLUSTERID,
12        size,
13        (unsigned char *) data,
14        &n_transactions,
15        AF_DISCV_ROUTE,
16        AF_DEFAULT_RADIUS) == afStatus_SUCCESS) ? true:false;
17 }
```

5.4 Conclusão

Este capítulo teve como objetivo, numa primeira fase, apresentar o *hardware* desenvolvido, primeiro com a apresentação de todos os módulos, e depois, com os dispositivos criados. Em seguida foi representado o comportamento de alguns eventos do sistema, através de diagramas de sequência. Por fim, foram apresentadas as principais funcionalidades do sistema desenvolvidas em *software*.

No capítulo 6 é possível observar alguns dos testes desenvolvidos, de forma a testar as funcionalidades implementadas neste projeto.

Capítulo 6

Testes e resultados

Neste capítulo serão apresentados os testes efetuados ao projeto desenvolvido nesta dissertação. Será também feita uma pequena descrição do ambiente em que decorreram os testes, especificando os componentes utilizados e as ferramentas envolvidas.

6.1 Ambiente de testes

No decorrer dos testes experimentais torna-se necessário caraterizar o ambiente em que os mesmos acontecem. Para isso é necessário especificar os vários componentes do teste realizado, tal como *hardware*, ferramentas de *software*, entre outros.

Relativamente ao *hardware* foram utilizados os módulos desenvolvidos nesta dissertação formando assim três tipos de dispositivos: HWED, HWR e HWC. Foram ainda utilizadas três placas de desenvolvimento, duas *RasberryPie* e uma *Kontron*, com funcionalidades de HWG de forma a integrar o projeto desenvolvido com o restante sistema.

O *firmware* foi compilado utilizando o *IAR Embedded Workbench* e inserido nos microprocessadores respetivos.

6.2 Testes desenvolvidos

Numa primeira fase são feitos testes isolados a cada componente do sistema, onde cada módulo é estado de forma independente. São efetuadas medições de consumo energético, testes ao protocolo série RS232, entre outros. Por último é feito um teste de integração do sistema, que tem o intuito de juntar a WSN

desenvolvida com o HWG já desenvolvido anteriormente. Desta forma é possível efetuar um teste funcional ao sistema completo.

6.2.1 Teste à bateria *SAFT VHT AA*

O primeiro teste realizado passa por adquirir a curva caraterística da bateria de forma a saber qual é a zona ativa da mesma, permitindo usar essa informação para se averiguar quando a mesma se encontra sem carga.

Na figura 6.1 está representado o resultado do teste efetuado. Este teste consistiu em enviar SAMPLES com um curto intervalo de tempo entre elas (0.2s) de forma a determinar-se qual a duração da bateria e a respetiva curva. Neste teste foram utilizados os módulos *HWCC2530* e *HWACCELEROMETER*, pelo que não estão incluídos os sensores do barramento I^2C nem o módulo de *energy harvesting*.

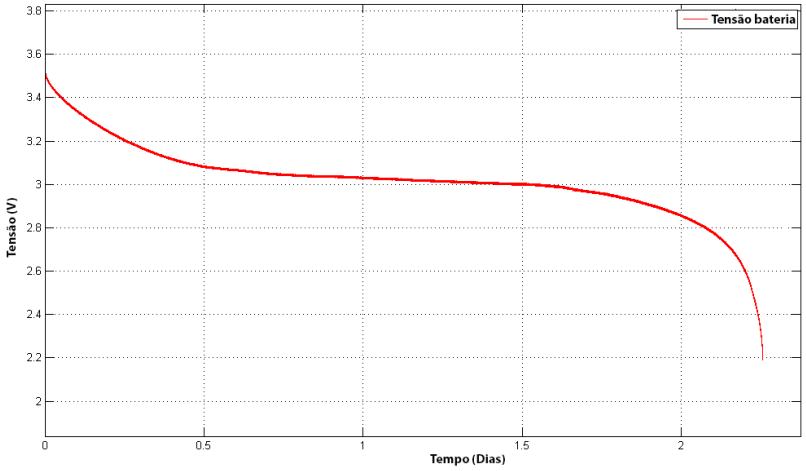


Figura 6.1: Curva caraterística da bateria *SAFT VHT AA*.

Este teste teve a duração de aproximadamente 54 horas. A tensão da bateria variou dos 3.5V aos 2.2V tendo começado a cair rapidamente a partir dos 2.9V. Isto significa que a partir deste valor deve-se notificar que a bateria se encontra com um baixo valor de tensão sendo possível substituí-la caso necessário.

6.2.2 Teste ao protocolo porta série

Para validar o funcionamento do protocolo série RS232, foi criado um teste que tinha como objetivo enviar uma trama de dados por este protocolo. Na figura 6.2 é apresentado o resultado obtido neste teste.

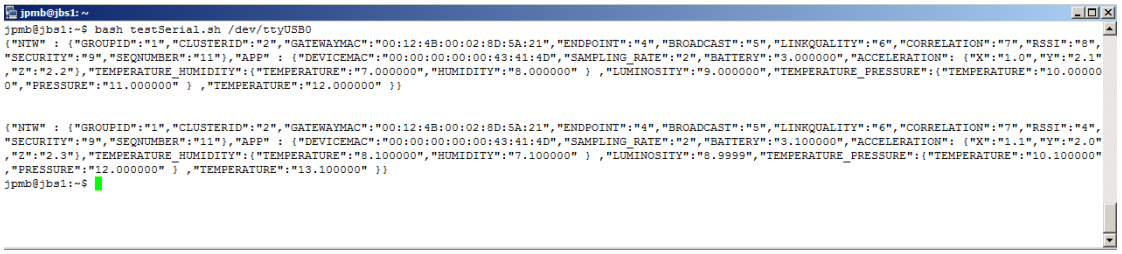


Figura 6.2: Duração do estado SAMPLE.

Como é possível observar todos os dados são recebidos de forma correta, com informação em todos os campos do protocolo escolhido pela aplicação e suportados pelo *firmware*. No entanto, os valores dos campos obtidos não são valores reais obtidos pelo sistema, serve apenas para apresentar o formato pelo qual estes são enviados (JSON).

6.2.3 Barramento I^{2}C

Na ausência de periférico I^{2}C no *SoC CC2530*, foi necessária a implementação do mesmo em *software*, por *bit banging*, sendo por isso,necessário efetuar um teste ao barramento de forma a verificar se os dados estavam a ser transmitidos corretamente. O teste consiste na aquisição de amostras por parte dos sensores conetados a cada 250 ms, onde para isso é feito o *enable* ao barramento I^{2}C. Para fazer este teste foi utilizado o multímetro digital *MS04100* da *Tektronix*.

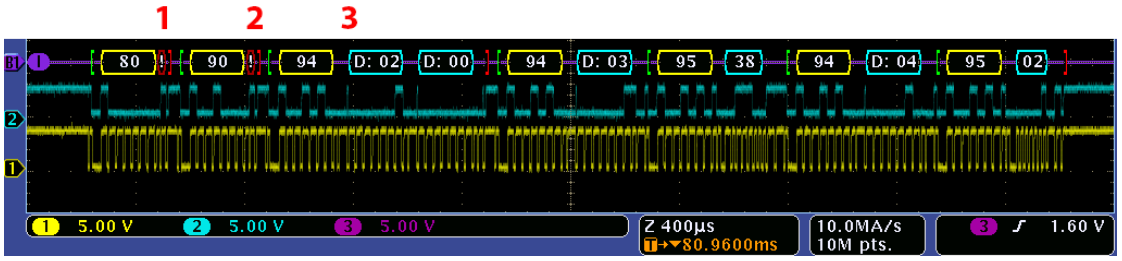


Figura 6.3: Comunicação com os sensores de I^{2}C.

Na figura 6.3 é possível observar uma parte dos comandos recebidos durante uma aquisição da informação sensorial, onde estava apenas conetado o sensor de luminosidade *MAX44009*, que tem o endereço 0x94. Para saber quais os sensores que estão conetados ao dispositivo é necessário tentar comunicar com os mesmos. É feita então uma inicialização do sensor, e caso o mesmo não esteja presente, é facilmente detetado devido à não resposta ACK ao comando enviado. Numa primeira tentativa é testado se o sensor de temperatura e humidade SHT21, com o endereço 0x80, está conetado. Como é possível observar no instante 1 o *ackno-*

wledge não é recebido o que significa que o sensor não está ligado e, então, deve tentar comunicar com o próximo. O seguinte é o sensor de temperatura TMP100, que tem como endereço 0x48, em que o mesmo acontece no instante 2. No momento 3 já existe o sensor de luminosidade pelo que são feitos os pedidos necessários para obter o respetivo valor. Por último, é feito o mesmo teste ao sensor BMP180, ao que é recebido um erro na confirmação da inicialização (figura 6.4 instante 4).

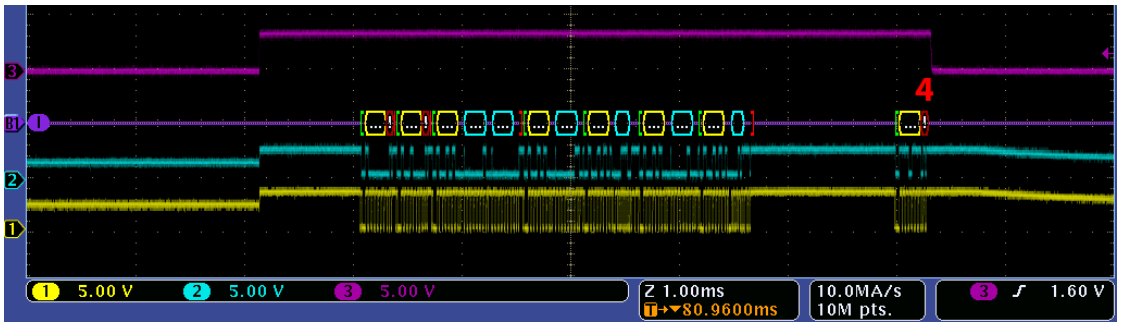


Figura 6.4: Pino de *enable* do barramento de I^2C.

Na figura 6.4 é possível ver a comutação de *enable* antes da aquisição dos sensores e após a mesma, provando que esta funcionalidade apresenta o correto funcionamento.

6.2.4 Teste de integração do sistema

De forma a testar a interação entre todos os módulos e dispositivos, foi criado o teste de integração entre os módulos do sistema e os HWG já instalados no laboratório e desenvolvidos pelo grupo de trabalho onde este projeto se insere. As WSNs desenvolvidas são conetadas aos HWGs já desenvolvidos no departamento. Na figura 6.5 é apresentado o *overview* do teste realizado.

É composto por três WSN diferentes criadas por três HWC distintos com PANIDs diferentes para cada uma das redes. A especificação dos componentes utilizados podem ser observados na tabela 6.1. Foram utilizadas diversas combinações diferentes com os módulos, no entanto, era importante determinar o funcionamento do dispositivo com todos os sensores e, por isso, utilizou-se o HWED 1 para determinar qual a duração do dispositivo.

Em seguida serão apresentados os resultados referentes ao HWED 1 que utiliza a bateria especificada para o dispositivo bem como todos os sensores suportados pelo *firmware* desenvolvido. Este dispositivo teve aproximadamente a duração de 8 dias com um tempo de amostragem de 1 segundo e o envio de 720403 mensagens para o HWC da respetiva rede. Na figura 6.6 é possível observar os

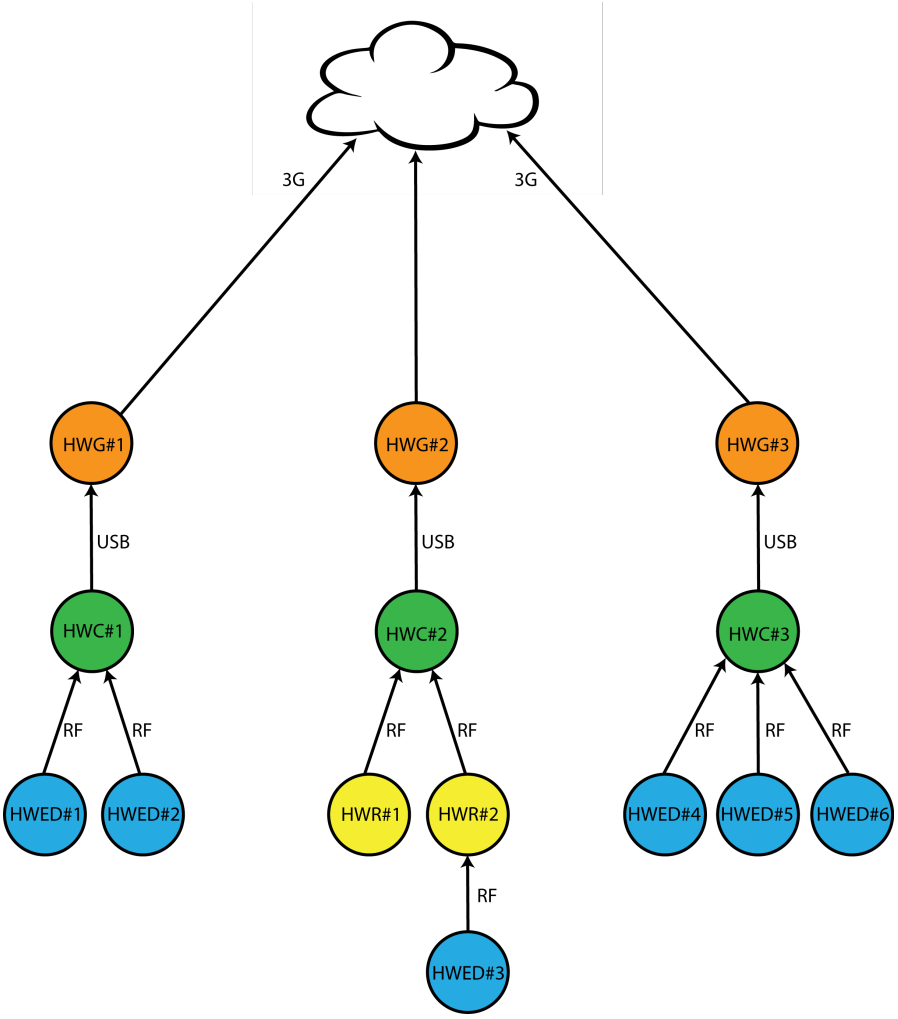


Figura 6.5: Esquema do teste de integração.

valores lidos pelo ADC do nível da bateria. Apesar de não ser possível observar a curva na sua totalidade, devido a um erro numa resistência do circuito do ADC, consegue-se saber a partir de que valor é que o dispositivo deixou de funcionar.

Para que os valores da bateria pudessem ser lidos pelo ADC, foi criado um circuito do tipo divisor resistivo, que divide os valores obtidos pela constante 3,2. Os valores apresentados pela figura 6.6, recolhidos pelo ADC, representam portanto a tensão nominal da bateria dividida por essa constante obtida.

De forma a observar os valores obtidos pelos sensores ao longo do teste, foram criados gráficos para cada um deles e feita uma descrição dos resultados adquiridos. Na figura 6.7 são apresentados os valores obtidos pelo sensor de temperatura e humidade *SHT21*.

Dado que o teste foi efetuado em laboratório, e as condições do meio ambiente não foram reproduzidas na totalidade (variações de temperatura e humidade), o

Tabela 6.1: Especificação dos dispositivos utilizados no teste de integração.

WSN	Dispositivo	Tempo Amostragem	Detalhes
1	HWED 1	1 seg	Bateria: SAFT VHT AA; Sensores: THE, LE, PE, TS, Acelerómetro;
1	HWED 2	2 min	Bateria: PL903048
1	HWC 1	NA	-
1	HWG 1	NA	RasberryPI
2	HWR 1	1 min	Bateria: PL903048; Sensores: THE e LE; Painel Solar
2	HWR 2	1 min	Bateria: PL903048; Sensores: THE e LE
2	HWED 3	2 min	Bateria: PL903048; Sensores: Acelerómetro
2	HWC 2	NA	-
2	HWG 2	NA	RasberryPI
3	HWED 4	30 min	Bateria: PL903048; Sensores: Acelerómetro
3	HWED 5	1h	Bateria: PL903048; Sensores: Acelerómetro
3	HWED 6	1h30 min	Bateria: PL903048; Sensores: Acelerómetro
3	HWC 3	NA	-
3	HWG 3	NA	Kontron

resultado, com valores mais ou menos constantes, consegue mostrar com mais evidência apenas as variações de temperatura entre o dia e a noite (os picos com maior valor correspondem ao período do dia, e os de menor valor ao período da noite). Consegue-se também retirar da figura que os 8 picos obtidos correspondem, efectivamente, aos 8 dias em que o teste decorreu.

Em seguida, na figura 6.8, é apresentado o resultado do sensor de luminosidade *MAX44009*. Os valores tem uma variação desde 10 lux aé 430 lux, aproximadamente. Na figura, é possível diferenciar as alturas do dia em que as luzes do laboratório estão apagadas ou ligadas. Estes dados poderiam variar significativamente caso o sensor estivesse na direção de uma lâmpada, no entanto, é possível observar a variação de intensidade da luz esperada.

Na figura 6.9 são apresentados os valores obtidos pelo sensor *BMP180* da

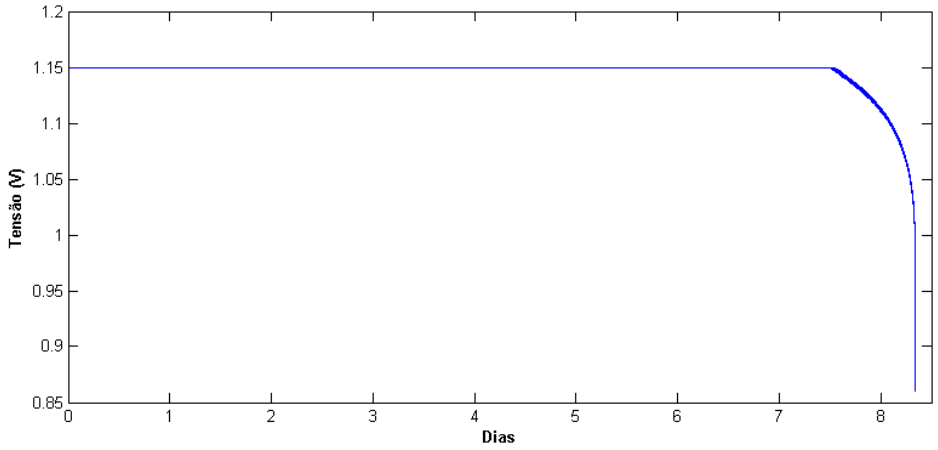


Figura 6.6: Gráfico do nível de bateria obtido pelo ADC.

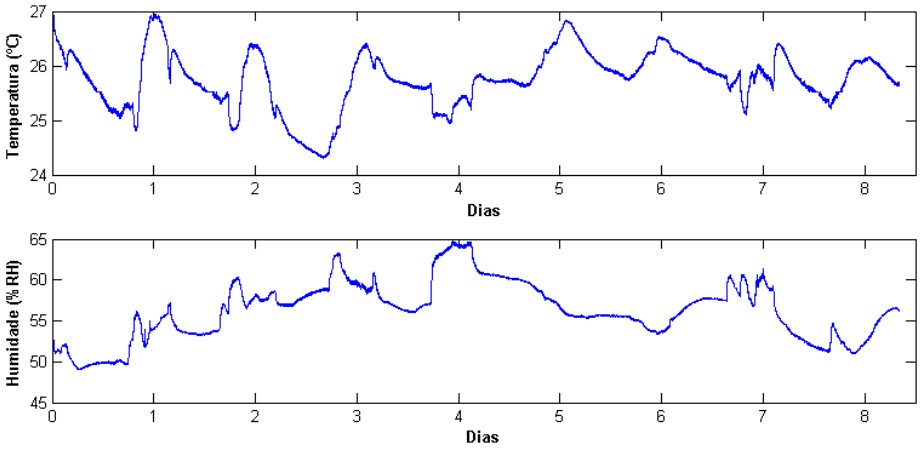


Figura 6.7: Valores adquiridos pelo sensor de temperatura e humidade *SHT21* no dispositivo HWED 1.

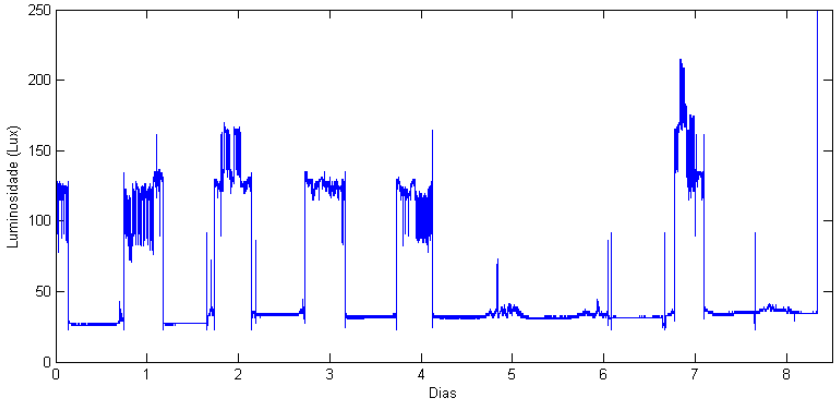


Figura 6.8: Valores adquiridos pelo sensor de luminosidade *MAX44009* no dispositivo HWED 1.

temperatura e pressão atmosférica. Ao longo dos 8 dias a temperatura e pressão tiveram variações esperadas, sendo possível visualizar os diferentes dias na figura relativa à temperatura. A pressão variou de forma sinusoidal ao longo da duração do teste.

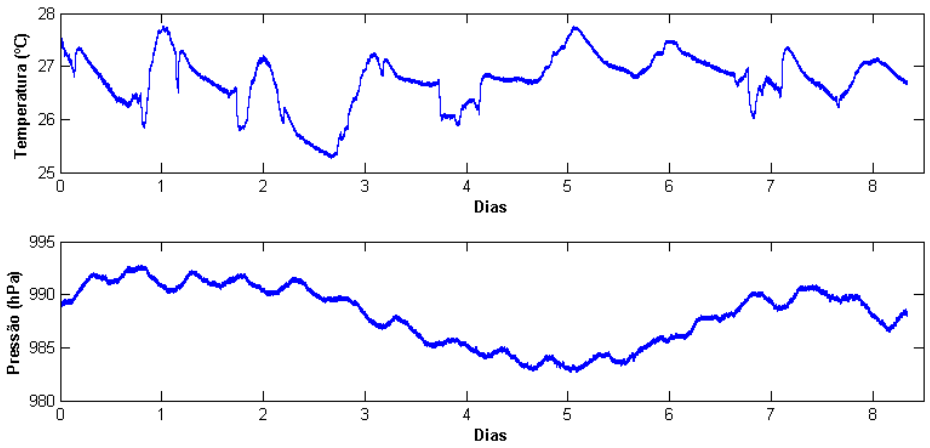


Figura 6.9: Valores adquiridos pelo sensor de temperatura e pressão atmosférica *BMP180* no dispositivo HWED 1.

Por último, são apresentados os resultados do sensor de temperatura *TMP100*. Através da figura 6.10 é possível observar as variações que ocorreram ao longo dos oito dias de teste.

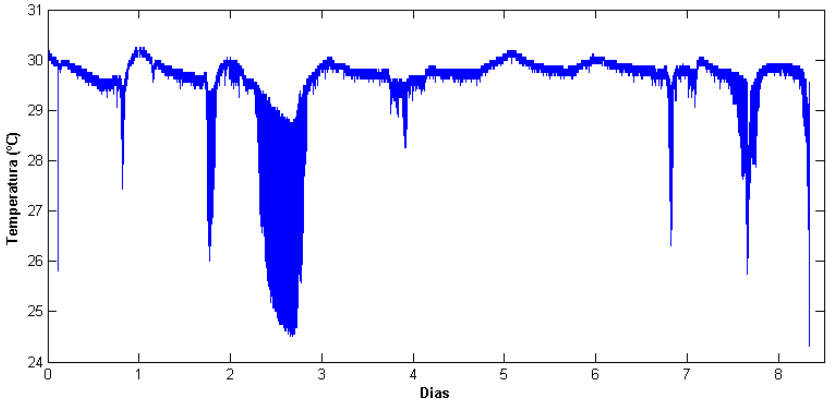


Figura 6.10: Valores adquiridos pelo sensor de temperatura *TMP100* no dispositivo HWED 1.

De notar que existem variações de temperatura de sensor para sensor. Isto pode ser causado por diversas razões, tais como, o *filter cap* do *SHT21* ou até mesmo as propriedades do sensor. No entanto, as variações da temperatura são semelhantes.

6.3 Medição de consumo

De forma a obter-se o consumo médio de corrente do dispositivo foi necessário utilizar o osciloscópio digital *MS04100* da *Tektronix* [55]. Foi criado um *firmware* de testes para um módulo HWED que envia pacotes de dados via RF com intervalos de tempo entre pacotes de 250ms. Desta forma, foi possível saber o consumo médio do dispositivo em várias fases e, também, confirmar se o valor definido para a taxa de amostragem estava correto. Como é possível verificar na Figura 6.11 existem quatro fases distintas do dispositivo, *sleep*, ativo, RX e TX. De forma a encontrar o consumo médio do sistema foi necessário calcular o consumo em cada uma destas fases.

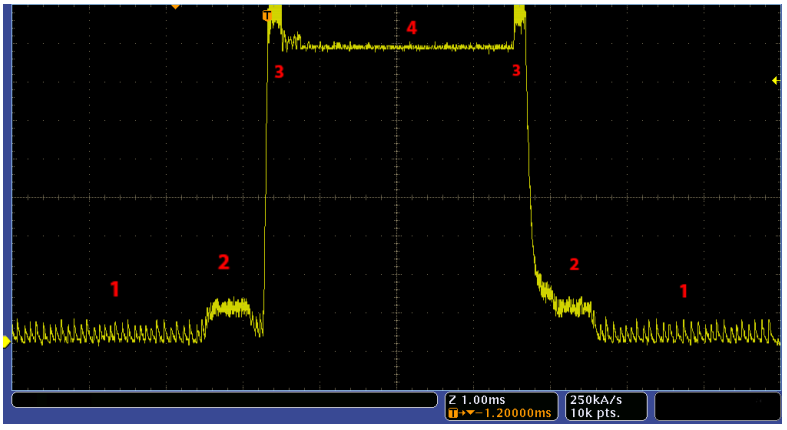


Figura 6.11: Onda caraterística do envio de uma SAMPLE.

A tabela 6.12 apresenta os consumos obtidos em cada uma das etapas, bem como a sua duração, de forma a ser possível calcular o valor da corrente em relação ao tempo.

Tabela 6.2: Consumo médio de corrente e descrição dos vários estados.

Intervalo	**Descrição**	**Corrente (mA)**	**Tempo (ms)**
1	Modo *sleep*	0,307	-
2	*MCU* modo ativo, 16 MHz *clock*	5,66	0,8
3	Modo RX	31	0,2
4	Transmissão do pacote, modo TX	23,77	3

Como é possível ver pelos valores apresentados na tabela referida anteriormente, o consumo médio do dispositivo no modo ativo, para a duração obtida, é de 92,77 mA. Imagine-se uma taxa de amostragem de hora em hora, ou seja, a cada hora o dispositivo apenas estará em modo ativo durante 5 ms (verificado na Figura 6.11), significando que o dispositivo se encontra em modo *sleep* o tempo restante, com um consumo médio de 0,307 mA. De notar que o tempo em modo ativo pode variar dependendo de quantos sensores estão conetados ao dispositivo, atrasos da resposta do ACK, entre outros. O valor do consumo médio de corrente do dispositivo, com uma taxa de amostragem de uma hora, pode ser confirmado recorrendo à equação 6.1

$$I\_médio[mAh] = (I\_sleep[mA] * tempo[h] + I\_ativo[mA] * tempo[h])/3600 \tag{6.1}$$

Sabendo o consumo médio por hora do dispositivo, e recorrendo à informação disponível da bateria relativamente à capacidade da mesma, consegue-se prever o tempo de vida de cada HWED. Neste caso, para a bateria escolhida (*SAFT LS 17500* com capacidade de 3400 mAh) e para o tempo de amostragem de 1h, a previsão do tempo de vida é obtida tal como mostra a equação 6.2. É possível então saber que o dispositivo tem uma duração de aproximadamente 11074,9 horas, que corresponde a 461 dias.

$$tempo\_vida[h] = capacidade\_bateria[Ah]/consumo\_sistema[A] \tag{6.2}$$

Durante o mesmo teste, achou-se relevante saber se a taxa de amostragem definida no *firmware* correspondia, efetivamente, à taxa de amostragem real. Na figura 6.12 está representado o tempo entre dois pacotes, e como se pode observar o tempo entre as duas tramas é de 250 ms (valor que foi definido).

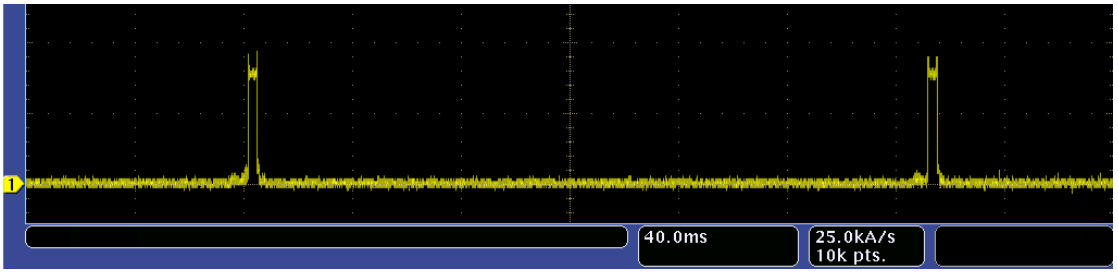


Figura 6.12: Tempo entre duas *SAMPLES*.

Sendo este teste efetuado sem ter conetado qualquer sensor foi feito um teste com o objetivo de obter o consumo médio de corrente de todos os sensores, pesar de não ser uma parte muito significativa no consumo médio. A tabela 6.3 apresenta os resultados obtidos para cada um dos sensores.

Tabela 6.3: Consumo médio de corrente dos sensores suportados.

Sensor	**Corrente (μA)**
SHT21	0,17
MAX44009	0,32
BMP180	0,68
TMP100	0,15
LIS331DHL	500

6.4 Conclusão

Através deste capítulo foi possível apresentar os principais resultados obtidos após implementação do projeto. Através de testes unitários e de um teste de integração foi demonstrado que o sistema foi implementado com sucesso, tendo incorporado todas as funcionalidades propostas. Na última secção foi apresentado o consumo do dispositivo HWED, bem como dos módulos dos sensores que se podem utilizar.

Capítulo 7

Conclusões e Trabalho futuro

Este último capítulo tem como objetivo apresentar as conclusões do autor sobre o projeto, sugerindo algumas modificações na arquitetura e funcionalidade do sistema, de forma a dar continuação ao trabalho realizado.

7.1 Conclusões

A presente dissertação tinha como objetivo a criação de um sistema capaz de monitorizar as condições atmosféricas das autoestradas, de forma a poder diminuir o número de acidentes que acontecem nessas vias.

Após a realização do projeto é possível afirmar que os objetivos iniciais propostos para a realização do mesmo foram efetivamente cumpridos, e que o sistema consegue obter informação sensorial relativamente às seguintes variáveis físicas: temperatura do ar, humidade do ar, luminosidade, pressão atmosférica, temperatura do solo e acelerações.

De forma a comunicar com os sensores foi necessário desenvolver os drivers dos mesmos, bem como, oferecer suporte ao protocolo I^2C através de *bit banging* visto que o processador utilizado (*CC2530*) não oferecia suporte a este tipo de comunicação, suportando apenas o acelerómetro (*LIS331DLH*) que utiliza o protocolo SPI.

Foi também incorporada a funcionalidade de roteamento dinâmico, possibilitando desta forma estender o alcance da rede através do uso dos dispositivos HWR, que podem possuir também as funcionalidades do HWED devido à modularidade do HW. Esta tarefa foi realizada em conjunto com a equipa do ESRG responsável pelo desenvolvimento do projeto *SustIMS*.

Por último, realça-se a funcionalidade de *plug-and-play* que permite aos utili-

zadores não se preocuparem com configurações do dispositivo independentemente dos sensores adicionados, isto é, não é necessária qualquer preocupação por parte do utilizador na utilização dos sensores suportados, apenas é necessário ligar ao dispositivo.

7.2 Trabalho futuro

Apesar de todos os objetivos terem sido cumpridos é possível melhorar o sistema atual, seja pela adição de novas funcionalidades ou sensores, ou pela alteração do *hardware* de forma a inserir melhorias nessa componente.

A primeira sugestão passa por fazer uma certificação de qualidade e segurança a todo o sistema, possibilitando torná-lo mais seguro e assegurar a qualidade de todos os componentes que integram o sistema.

Visto o barramento de I^2C ter sido implementado por *bit-banging*, dado a falta de suporte pelo *CC2530*, sugere-se um melhoramento do mesmo, testando os seus limites e potencialidades. Numa outra abordagem, sugere-se ainda a substituição do processador por um que dê suporte ao barramento de I^2C, por exemplo, o *CC2538* da TI.

A terceira sugestão passa pela alteração da fonte de energia para carregar as baterias. Atualmente o sistema utiliza a luz proveniente do sol para realizar esta tarefa. No entanto, devido a poeiras, chuva e outras condições desfavoráveis, o painel solar pode facilmente ficar tapado impossibilitando a aquisição da energia solar de forma correta. Por isso, sugere-se a inclusão de uma outra fonte de energia renovável como uma fonte piezoelétrica, podendo ser vantajoso para o sistema.

A última sugestão passa pelo suporte a novos sensores que de alguma forma podem ser um contributo para a monitorização das vias rodoviárias. Em forma de exemplo, seguem algumas sugestões: sensor de CO2, vento, precipitação e gelo (ou uma forma de poder antecipar a sua formação).

Referências Bibliográficas

[1] C. Chong and S. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1219475

[2] M. Leopold, "Sensor Network Motes:: Portability & Performance," no. December, 2007. [Online]. Available: http://curis.ku.dk/ws/files/14772764/Martin_Leopold_Thesis.pdf

[3] a. Savvides and M. Srivastava, "A distributed computation platform for wireless embedded sensing," *Proceedings. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pp. 220–225. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1106774

[4] T. Crossbow, "eKo Pro Series System brochure."

[5] G. Barrenetxea, F. Ingelrest, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange, "SensorScope: Out-of-the-Box Environmental Monitoring," *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 332–343, Apr. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4505485

[6] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications - WSNA '02*, p. 88, 2002. [Online]. Available: http://portal.acm.org/citation.cfm?doid=570738.570751

[7] K. Sohraby, D. Minoli, and T. Znati, *Wireless sensor networks: technology, protocols, and applications*, 2007. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:No+Title#0http://books.google.com/books?hl=en&lr=&id=I3bJGo690SUC&oi=fnd&pg=PR5&dq=Wireless+Sensor+Networks:+Technology,+Protocols+and+applications&ots=olOp-QUvSt&sig=gcxGG3IFten19-sRcSGBNqFAOhg

[8] L. D. Vito and V. Cocca, "Wireless active guardrail system for environmental

measurements," *Environmental Energy ...*, vol. 6, pp. 50–57, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6348403

[9] M. Matin and M. Islam, "Overview of Wireless Sensor Networks," in *Wireless Sensor Networks - Technology and Protocols*, 2012. [Online]. Available: http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/overview-of-wireless-sensor-network

[10] Chipdesignmag, "ZigBee Wireless Sensor Applications for Health, Wellness and Fitness," 2009, [Acedido em 14-02-2014]. [Online]. Available: http://chipdesignmag.com/sld/files/2009/06/zigbee_wirelesshealth1.jpg

[11] L. d. C. Antunes, "Identificação de pessoas numa portaria virtual," 2012. [Online]. Available: http://repositorio.ipl.pt/handle/10400.21/2405

[12] Texas Instruments, "A True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications," p. 36, 2011.

[13] F. M. C. Zhao and L. S. U. Guibas, *Wireless Sensor Networks: An information processing approach*. Elsevier, 2004.

[14] P. Jonsson, "Classification of road conditions: From camera images and weather data," in *Computational Intelligence for Measurement Systems and Applications (CIMSA), 2011 IEEE International Conference on*, 2011, pp. 1–6.

[15] L. Stormhighway.com and {Midwest 64 Multimedia}, "Icy Road Safety.com, Icy Road Fatality Statistics," [Acedido em 23-11-2013]. [Online]. Available: http://www.icyroadsafety.com/fatalitystats.shtml

[16] J. O. da Costa, E. F. Freitas, P. A. A. Pereira, and M. A. P. Jacques, "Acidentes rodoviários das estradas nacionais de Portugal : estudo da associação entre as variáveis recolhidas," 2011. [Online]. Available: http://repositorium.sdum.uminho.pt/handle/1822/15483

[17] C. Bryan and C. Nishimura, "MONITORING OCEANIC EARTHQUAKES win4 SOSUS: AN EXAMPLe FROM THE CARIBBEAN," *tos.org*, vol. 4, no. 1, pp. 4–10, 1940. [Online]. Available: http://www.tos.org/oceanography/archive/8-1_bryan.pdf

[18] R. Rashid and G. Robertson, *Accent: A communication oriented network operating system kernel*, 1981. [Online]. Available: http://dl.acm.org/citation.cfm?id=806593

[19] R. Rashid and H. Tokuda, "Mach: a system software kernel," *Computing Systems in Engineering*, no. 4864, pp. 1–3, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0956052190900045

[20] Department of the Navy, "Advanced Deployable System Ocean Tests Program

Definition and Risk Reduction Phase," 1998.

[21] Z. Haig, "Networked unattended ground sensors for battlefield visualization," *TECHNOLOGY*, vol. 3, no. 3, pp. 387–399, 2004. [Online]. Available: http://baranta.zmne.hu/docs/Volume3/Issue3/pdf/07haig.pdf

[22] L. communications, "Remotely Monitored Battlefield Sensor System-II," pp. 3–4.

[23] TI, "Texas Instruments," [Acedido em 06-08-2014]. [Online]. Available: http://www.ti.com/

[24] BTNodes, "BTnodes - A Distributed Environment for Prototyping Ad Hoc Networks," [Acedido em 06-08-2014]. [Online]. Available: http://www.btnode.ethz.ch/Main/Purchase

[25] E. Zurich, "Smart-ITs Project," [Acedido em 17-08-2014]. [Online]. Available: http://www.smart-its.org/

[26] Art of Techbology, "Art of Technology." [Online]. Available: http://art-of-technology.ch/

[27] R. H. ECos, "Red Hat eCos," [Acedido em 07-08-2014]. [Online]. Available: http://ecos.sourceware.org/

[28] UCLinux, "Embedded Linux/Microcontroller Project," [Acedido em 07-08-2014]. [Online]. Available: http://www.uclinux.org/

[29] P. Juang, H. Oki, Y. Wang, and M. Martonosi, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," *ACM Sigplan ...*, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=605408

[30] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware Design Experiences in ZebraNet."

[31] Traffic.com, "Traffic Pulse Technology," [Acedido em 17-08-2014]. [Online]. Available: http://here.com/traffic

[32] GPS Lodge, "NAVTEQ to Acquire Traffic.com," 2006, [Acedido em 15-08-2014]. [Online]. Available: http://www.gpslodge.com/archives/008272.php

[33] MEMSIC, "Wireless Sensor Networks Nodes," 2014, [Acedido em 18-08-2014]. [Online]. Available: http://www.memsic.com/wireless-sensor-networks/

[34] Sistra, "RED-WINE," [Acedido em 17-08-2014]. [Online]. Available: http://www.sistrasrl.it/prodotti/traffico

[35] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *ACM Sigmod Record*, 2002. [Online]. Available: http://dl.acm.org/citation.cfm?id=601861

[36] K. Whitehouse, "The design of calamari: an ad-hoc localization system

for sensor networks," *University of California at Berkeley*, 2002. [Online]. Available: http://www.cs.virginia.edu/~whitehouse/research/localization/whitehouse02calamari.pdf

[37] M. Britton and L. Sacks, "The SECOAS Project — Development of a Self-Organising , Wireless Sensor Network for Environmental Monitoring."

[38] G. Werner-Allen and K. Lorincz, "Deploying a wireless sensor network on an active volcano," *Internet Computing ...*, no. April, pp. 18–25, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1607983

[39] S. Online, "A Practical Guide to Battery Technologies for Wireless Sensor Networking," 2008, [Acedido em 02-07-2014]. [Online]. Available: http://www.sensorsmag.com/networking-communications/batteries/a-practical-guide-battery-technologies-wireless-sensor-netwo-1499

[40] ZigBee Alliance, "ZigBee Specification Overview," [Acedido em 17-07-2014]. [Online]. Available: http://www.zigbee.org/Specifications/ZigBee/Overview.aspx

[41] L. S. Committee, "Part 15.4: wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (LR-WPANs)," *IEEE Computer Society*, vol. 2006, no. September, 2003. [Online]. Available: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Part+15.4:+Wireless+Medium+Access+Control+(MAC)+and+Physical+Layer+(PHY)+Specifications+for+Low-Rate+Wireless+Personal+Area+Networks+(WPANs)#0

[42] FTDI Chip, "FT232R USB UART IC," pp. 1–43, 2010.

[43] Texas Instruments, "HIGH INPUT VOLTAGE BUCK-BOOST CONVERTER WITH 2A SWITCH CURRENT TPS63060," 2012. [Online]. Available: http://www.ti.com/lit/ds/symlink/tps63060.pdf

[44] Json.org, "Introducing JSON," [Acedido em 15-09-2014]. [Online]. Available: http://json.org/

[45] GBAN Electronic Technology, "GB2530-S High Power Zigbee Module(CC2530+RF2401,SMT)," [Acedido em 06-09-2014]. [Online]. Available: http://www.gban.cn/en/product_show.asp?id=47

[46] RFaxis, "RFX2401," [Acedido em 06-09-2014]. [Online]. Available: http://www.rfaxis.com/downloads/RFX2401.pdf

[47] Texas Instruments, "SmartRF05 Evaluation Board User's Guide," p. 62, 2010, [Acedido em 28-08-2014]. [Online]. Available: http://www.ti.com/lit/ug/swru210a/swru210a.pdf

[48] Sensirion, “Datasheet SHT21 - Humidity and Temperature Sensor,” pp. 1–12, 2010.

[49] DSM&T CO. INC, “IP Rating Chart,” [Acedido em 26-08-2014]. [Online]. Available: http://www.dsmt.com/resources/ip-rating-chart

[50] Maxim Integrated, “MAX44009 Industry ’ s Lowest-Power Ambiente Light Sensor with ADC,” pp. 1–20.

[51] Bosch Sensortec, “BMP180 Digital pressure sensor,” p. 28, 2013.

[52] Texas Instruments, “TMP100-Q1 - Digital Temperature Sensor With I2C Interface,” no. September, p. 20, 2011.

[53] Saft, “Primary lithium battery LS 17500,” no. July, 2010.

[54] Texas Instruments, “Z-Stack,” 2012, [Acedido em 14-04-2014]. [Online]. Available: http://www.ti.com/tool/z-stack

[55] Tektronix, “MSO/DPO4000B Mixed Signal Oscilloscope,” [Acedido em 22-10-2014]. [Online]. Available: http://www.tek.com/oscilloscope/mso4000-dpo4000