Universidade do Minho
Escola de Engenharia

Luís Miguel Martins Pereira

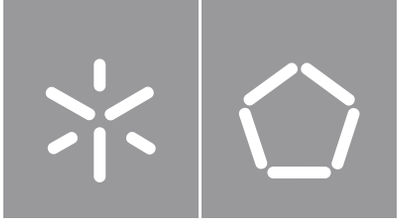
Sistema ciber-físico para deteção de colisões

janeiro de 2019

Sistema ciber-físico para deteção de colisões

Luís Pereira

UMinho | 2019



Universidade do Minho
Escola de Engenharia

Luís Miguel Martins Pereira

Sistema ciber-físico para deteção de colisões

Dissertação de Mestrado
Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação do
Professor Doutor Jorge Cabral

janeiro de 2019

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

AGRADECIMENTOS

Em primeiro lugar, quero agradecer aos meus pais, José Manuel e Elsa Margarida por serem os meus pilares durante todo este percurso, não deixando de parte o meu irmão Sérgio Emanuel.

Ao meu orientador, Professor Doutor Jorge Cabral, pela oportunidade de realizar esta dissertação, bem como, por toda a motivação e apoio oferecidos.

Um especial agradecimento a três amigas que me acompanham desde a adolescência, Tânias e Estela, pela amizade, confiança e presença na minha vida.

E por fim, a todos os meus colegas e amigos que conheci durante a passagem por esta Universidade, com ênfase nos amigos dos laboratórios de sistemas embebidos e IBS.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

O número de acidentes rodoviários em Portugal têm vindo a diminuir desde o início do século XXI, apesar do ligeiro aumento dos últimos anos. Anualmente, nas autoestradas verifica-se a ocorrência de cerca 2000 acidentes com vítimas mortais e feridos. Para detetar acidentes neste tipo de vias são utilizados sistemas de captação de imagens, enviadas para um operador, que as analisará. Atualmente já existem sistemas capazes de processar essas imagens e automaticamente informar as entidades responsáveis.

As *Wireless Sensor Networks* (WSNs) são redes que incorporam, essencialmente, um coordenador e vários nós sensores. A informação obtida pelos nós sensores é enviada para o coordenador, reencaminhada para um *gateway* e colocada numa plataforma *online*. Assim sendo, as WSNs também podem ser utilizadas para detetar acidentes nas autoestradas. Na ocorrência de um acidente, estas avisam um operador, e desta forma serem tomadas as devidas medidas de emergência.

Tendo isto em mente, o que se propõe nesta dissertação é o *refactoring* de um sistema de deteção de colisões com as guardas de segurança das autoestradas. Esta solução consiste na adaptação de uma rede de sensores sem fios, de forma a utlizar o protocolo 6LowPAN. Os diversos nós sensores estarão dispostos ao longo da guarda de segurança, cada um equipado com um acelerómetro, comunicando diretamente com o coordenador.

O sistema a implementar já tem um protótipo funcional integrado num projeto denominado *SustIMS*. Numa revisão posterior, este sistema foi submetido a uma modificação do microcontrolador. O objetivo desta dissertação consiste em alterar o sistema operativo, mantendo o mesmo *system-on-chip*. Com a utilização do Contiki, pretende-se fazer um *refactoring* do projeto já implementado, pois este sistema operativo utiliza IPv6 nas comunicações entre os vários dispositivos.

Palavras-chave: *Wireless Sensor Networks*, Contiki OS, IEEE 802.15.4, Deteção de colisões.

ABSTRACT

The number of road accidents in Portugal has been declining, since the beginning of the 21st century, despite the slight increase in recent years. On the highways occurred around 2000 accidents with fatalities and injuries per year. In order to detect accidents on the highways, images from video cameras are sent to an operator who will analyze them. Currently, there are systems capable of processing these images and automatically inform the responsible entities.

Wireless Sensor Networks (WSNs) are networks that essentially have a coordinator and several sensor nodes. The information obtained by the sensor nodes is sent to the coordinator, forwarded to a gateway and placed on a online platform. Therefore, a WSN can also be used to detect accidents on highways. When an accident occurs, they notify an operator to take the necessary emergency measures.

With this in mind, what is proposed in this master thesis is the refatoring of a collisions detection system with the security guards of highways. This solution consists on the adaption of a wireless sensor network, in order to use the 6LoWPAN protocol. The sensor nodes will be placed along the security guard, each equipped with an accelerometer, communicating directly with the coordinator.

The implemented system already has a functional prototype, integrated into a project called SustIMS. In a later review, this system had a modification on the microcontroller. The purpose of this dissertation is changing the operating system, but keeping the same system-on-chip. Therefore, with the use of Contiki, the goal is to make the refactoring of the already implemented project. The main advantage of this operating system resides in the use of IPv6 in communications between the devices.

Keywords: Wireless Sensor Networks, Contiki OS, IEEE 802.15.4, Collision Detection.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Equações

Capítulo 1

Introdução

Os acidentes rodoviários em Portugal têm apresentado um decréscimo desde o fim do XX, apesar de nos últimos anos, ter-se verificado um ligeiro aumento. Segundo a base de dados de Portugal Contemporâneo, rondam em 32 mil, os acidentes com vítimas anuais, apresentando uma média de 130 vítimas em cada 100 acidentes [1]. Como se pode verificar, para um país com aproximadamente 10,5 milhões de habitantes, estes números são bastante elevados. Uma das formas de os reduzir é apostar na prevenção. Em Julho de 2017, o secretário de Estado da Administração anunciou que serão tomadas medidas para travar este aumento dos acidentes. Uma das medidas seria a criação de mais ações de sensibilização e de fiscalização, de forma a consciencializar os automobilistas para os perigos das estradas [2].

Apesar de todo o esforço colocado na prevenção, este não se revela o suficiente para reduzir significativamente o número de acidentes rodoviários, sendo necessário tomar medidas aquando da ocorrência destes. Quando acontece um acidente, caso este não seja rapidamente detetado e devidamente identificado, para além dos possíveis danos materiais já existentes, podem ocorrer os chamados choques em cadeia. Nestas situações, os primeiros socorros devem ser prestados o mais rapidamente possível. Caso esse auxílio seja prestado tardiamente, o estado de saúde das vítimas pode vir a agravar-se, sendo muitas vezes a diferença entre a vida e a morte de algumas pessoas.

Nas autoestradas, um dos métodos de deteção de acidentes rodoviários consiste na utilização de sistemas de captação de imagens, que posteriormente são enviadas para um operador, funcionário de uma concessionária, que as analisará. Atualmente, já existem sistemas capazes

de processar essas imagens e automaticamente, informar as entidades responsáveis [3]. As redes de sensores sem fios também podem ser utilizadas para detetar acidentes nas autoestradas. Na ocorrência de um acidente, estas avisam um operador para que sejam tomadas as devidas medidas de emergência. Assim, é possível alertar os condutores da ocorrência do acidente, através dos painéis da autoestrada, possibilitando uma redução nas consequências provocadas por este.

Uma rede de sensores sem fios ou *Wireless Sensor Network* (WSN) incorpora um coordenador, vários nós sensores, podendo também conter *routers*. Os nós sensores adquirem informação do meio ambiente, os *routers* possuem a capacidade de retransmitir tramas, e o coordenador gere e recebe toda a informação proveniente da rede. A Figura 1.1 descreve o funcionamento de uma WSN [4]. Como é possível verificar, os dados provenientes de um nó sensor atravessam diversos dispositivos até chegarem à internet, e consequentemente, ao utilizador.

A criação de dispositivos de pequena dimensão, com capacidade de comunicação e de baixo consumo energético, já é uma realidade aplicada nas WSNs. Os nós, equipados com sensores, recolhem informações sobre o ambiente em que estão inseridos, e envia-as para um dispositivo central denominado coordenador [5]. Este, por usa vez, é conectado a um *gateway* capaz de se ligar à internet, que coloca os dados recolhidos numa plataforma online, como por exemplo, uma *cloud*.

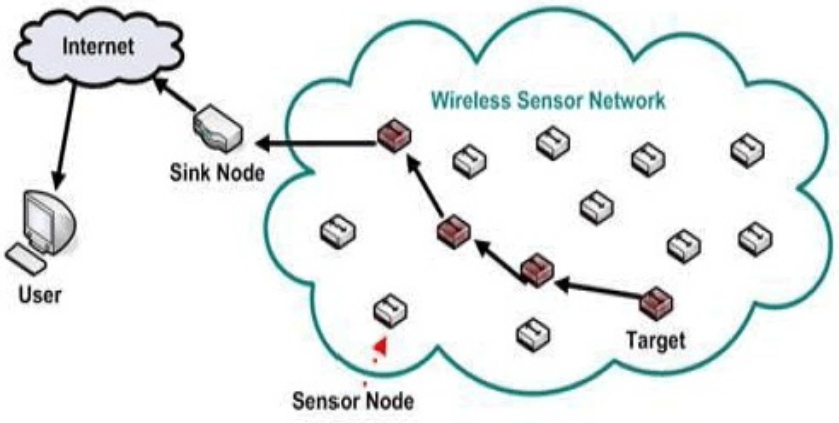


Figura 1.1: Rede de sensores sem fios

Um exemplo de uma WSN é o sistema de deteção de colisões integrado num projeto denominado *Sustainable Infrastructure Management System* (SustIMS). Este projeto visou o desenvolvimento de uma plataforma de gestão sustentável para infraestruturas rodoviárias com o objetivo de

gerir os principais elementos de uma infraestrutura incluindo obras de arte, pavimentos, taludes, muros, entre outros. Este, estava dividido em 3 partes, sendo a primeira, uma plataforma de gestão em que incluía uma base de dados e algoritmos para as tomadas de decisão, baseando-se nos dados recolhidos. A segunda parte é uma plataforma móvel que recolhe dados no local, sobre o estado de conservação de cada elemento. Finalmente, a terceira parte é composta pelos sistemas de monitorização baseados em redes de sensores sem fios que tanto monitoriza a estabilidade dos muros e taludes como deteta as colisões nos *rails* das autoestradas [6].

O SustIMS foi um projeto aprovado em 2012 pelo Quadro de Referência Estratégico Nacional (QREN), com o número de projeto 23113. O seu promotor foi a Ascendi tendo um investimento de 858.015 €e um incentivo de 451.792 €[7]. Em 2017, este projeto desenvolvido pela concessionária de autoestradas, em parceria com as Universidades do Minho e Nova de Lisboa, ganhou um dos *Global Road Achievement Awards* (GRAA). Atribuído pela *Internacional Road Federation* (IRF), o SustIMS é reconhecido como o melhor projeto mundial na categoria de "Preservação de Ativos & Gestão da Manutenção"[8].

1.1 Enquadramento

Uma WSN é composta, essencialmente, por nós sensores e um coordenador. De forma a prolongar a área abrangida por esta rede, serão colocados vários nós dispersos ao longo da guarda de segurança. Para além disto, estes terão de ser autónomos, sendo que o baixo consumo energético é uma necessidade para estes dispositivos. A Figura 1.2 representa a WSN utilizada no projeto SustIMS. Como é possível verificar, os nós sensores estão conectados ao coordenador via *wireless*, e este ligado a um *gateway* com acesso à Internet. Os dados gerados nessa rede enviados para serviços em *cloud*. Recorrendo a dispositivos eletrónicos, como um computador ou *smartphone*, é possível aceder a esses mesmos dados.

Nesta dissertação de mestrado pretende-se redesenhar a WSN de um sistema de deteção de colisões, aplicado em guardas de segurança das autoestradas. Esta WSN será composta por um coordenador e alguns nós sensores. Na revisão anterior do sistema integrado no projeto SustIMS, ambos os dispositivos possuíam um *System-on-Chip* (SoC) de 8 bits com um *transceiver RadioFrequência* (RF) capaz de enviar e receber informações sem fios. O nó sensor utilizava um acelerómetro para medir as acelerações, resultantes da vibração provocada pela colisão do

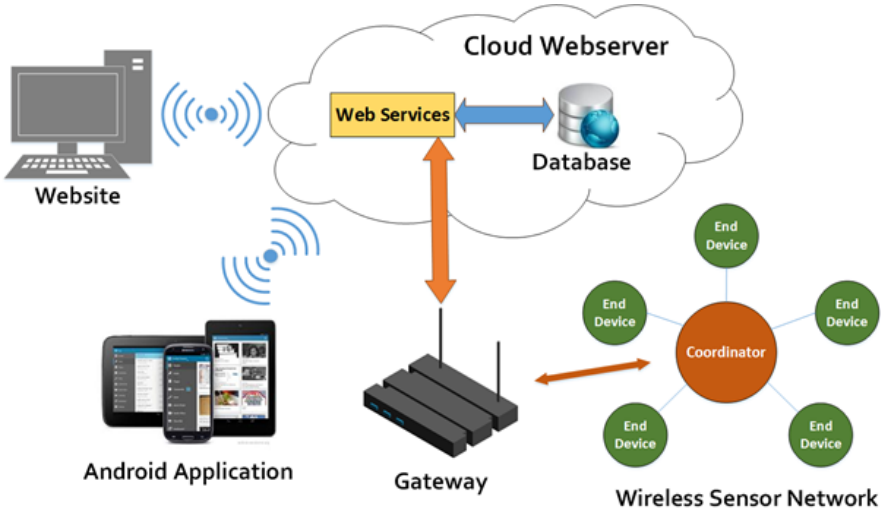


Figura 1.2: Visão geral da WSN do SustIMS

veículo com as guardas de segurança. O sistema operativo utilizado era a Z-Stack, nativa da *Texas Instruments* (TI).

O sistema proposto pretende alterar a unidade de processamento e o *software* do sistema já existente, com o intuito de generalizar as camadas superiores do sistema operativo, possibilitando uma adaptação futura de *hardware* mais rápida. Desta forma, será utilizado o novo *System-on-Chip* (SoC) CC2538 da TI que, para além do próprio processador de 32 bits, contém também um *transceiver* incorporado no mesmo SoC. Este é capaz de comunicar por *wireless* a 2.4 GHz, recorrendo aos protocolos IEEE 802.15.4 e 6LoWPAN. Incluirá também um acelerómetro, para detetar as colisões com o *rail*, e um amplificador de sinal de forma a aumentar o alcance da transmissão. Para a programação deste SoC será utilizado o sistema operativo Contiki tendo como vantagens de ser *open-source* e com um bom suporte online, através das comunidades de desenvolvimento [9].

Um recurso de assaz utilidade suportado pelo Contiki, é a capacidade dos nós se comunicarem utilizando o *Internet Protocol* (IP), mais concretamente, o IPv6. Este, por sua vez, não é exatamente o mesmo IPv6 que é utilizado na internet, mas um de tamanho reduzido com o *header* comprimido, de forma a se adequar às restrições impostas pelas WSNs. Este formato de IP é suportado pelo *IPv6 over Low-Power Wireless Personal Area Networks* (6LoWPAN). Com este recurso, teremos uma rede de sensores sem fios a comunicar entre si, da mesma forma com que, duas máquinas se comunicam na internet. Após implementada esta funcionalidade, a interligação das duas redes será mais simples.

1.2 Objetivos

Esta dissertação tem como principal objetivo redesenhar o nó sensor e o coordenador, de forma a dar suporte ao Contiki OS, e conseguir implementar o protocolo 6LoWPAN nas comunicações entre os vários dispositivos. Assim, com este sistema pretende-se:

- Sugerir uma implementação alternativa ao sistema existente, recorrendo ao Contiki OS;
- Manter a autonomia dos nós sensores, pois estes necessitam de ser autónomos e eficientes energeticamente;
- Aumentar o alcance de transmissão dos nós, proporcionando uma maior distância de comunicação;
- Implementar, na rede de sensores sem fios, uma comunicação utilizando o 6LoWPAN, suportado pelo Contiki, facilitando a conexão com a Internet.

1.3 Estrutura da dissertação

No capítulo 1 foi apresentada uma contextualização do problema, bem como a motivação que levou à elaboração deste sistema. Para além disso, o enquadramento no projeto SustIMS e os objetivos também são descritos. Assim, esta dissertação está repartida em 6 capítulos.

O capítulo 2 explica algumas tecnologias utilizadas, nomeadamente a norma IEEE 802.15.4 e o protocolo 6LoWPAN, bem como as diferenças deste com o ZigBee. Refere ainda os sistemas ciber-físicos, as redes de sensores sem fios e algumas utilizações destas.

No capítulo 3 é referida a especificação do sistema onde é descrito o *hardware* utilizado e dado um breve resumo do funcionamento do sistema.

O capítulo 4 descreve o *hardware* implementado e explica alguns detalhes da implementação do *software*. São apresentados ambos os dispositivos e o formato da mensagem.

O capítulo 5 mostra os testes realizados ao sistema, bem como os resultados destes.

Finalmente, o capítulo 6 apresenta as conclusões e o trabalho futuro. Com base nos resultados do testes, foi possível avaliar o projeto, podendo-o comparar com as versões anteriores.

Capítulo 2

Estado da Arte

No decorrer deste capítulo serão apresentados alguns protocolos de comunicação sem fios, destinados a dispositivos com recursos limitados. Posteriormente será abordado o tema dos sistemas ciber-físicos e das WSNs, bem como alguns exemplos destas. No final, é descrito o sistema operativo utilizado, Contiki OS.

2.1 Protocolos de comunicação sem fios

Inicialmente será apresentado o protocolo IEEE 802.15.4, a base dos *standards* conseguintes. Posteriormente, seguir-se-á o 6LoWPAN, um protocolo que utiliza IPv6 nas comunicações entre os dispositivos. Dado que esta dissertação propõe um *refactoring* de um protótipo implementado com tecnologia ZigBee, esta também será abordada. Adicionalmente, será referido o Thread, um protocolo com base em 6LoWPAN. No final deste subcapítulo será realizada uma pequena comparação entre as diversas tecnologias.

2.1.1 Norma IEEE 802.15.4

Em 1963, duas organizações de engenheiros juntaram-se com o objetivo de fundar o *Institute of Electrical and Electronics Engineers* (IEEE). Consiste na maior organização global profissional técnica, que se dedica ao avanço da tecnologia para benefício da humanidade [10]. Atualmente, já conta com cerca de 1600 conferências realizadas e mais de 426 mil membros espalhados por 160 países. Até agora, o IEEE já desenvolveu mais de 1600 projetos e normas [11].

De forma a se especializar nas diversas áreas, esta organização foi internamente dividida em vários *working groups*, que criam e desenvolvem *standards*. A participação nestes grupos é aberta a qualquer pessoa, desde que tenha conhecimentos profundos na área de determinado *working group*. De entre as várias áreas de especialização do IEEE, destacam-se as tecnologias de informação e a eletrónica de potência [12].

Um dos *working groups* é o IEEE 802 LAN/MAN que desenvolve *standards* de rede para redes locais, metropolitanas, entre outras. Este grupo foca-se nas redes com e sem fios para comunicação entre dispositivos eletrónicos. Desenvolveu *standards* tais como o IEEE 802.11, também conhecido por Wi-Fi, o IEEE 802.15.1 (primeiras versões do Bluetooth), e o IEEE 802.15.4, um protocolo utilizado em dispositivos *low power* [12].

O *standard* IEEE 802.15.4 define o funcionamento das LR-WPAN (*Low-Rate Wireless Personal Area Networks*). É essencialmente utilizado por sistemas de comunicação de baixa potência e baixo custo, operando com baixas taxas de transferência. Destina-se a dispositivos que pretendem comunicar a curtas distâncias e obter uma alta eficiência energética [13]. Esta tecnologia adequa-se bem aos dispositivos sensores de uma rede de sensores sem fios, pois cumprem com os requisitos fundamentais de uma rede deste tipo. Este *standard* define a camada física (PHY) e a subcamada MAC (*medium access control*).

2.1.1.1 Modelo de Rede

O *standard* IEEE 802.15.4 estabelece dois tipos de dispositivos físicos: o *full-function device* (FFD) e *reduced-function device* (RFD). Um FFD possui mais capacidades de memória e processamento, do que o outro tipo de dispositivos, sendo o único capaz de se comportar como coordenador de uma rede PAN (*personal area network*). Os RFDs são dotados de menores capacidades de processamento e memória. Normalmente são utilizados nas extremidades de uma rede, efetuando apenas ações simples, tais como, ler sensores ou comutar estados dos atuadores. Uma das suas funcionalidades na rede consiste na comunicação com outros dispositivos, sendo a sua configuração mais básica, a transmissão de dados para o coordenador, numa topologia em estrela [13].

O *Request for Comments* (RFC) 7228 é um documento do *Internet Engineering Task Force* (IETF), que define a terminologia para redes com dispositivos de recursos limitados. A tabela 2.1 apresenta 3 classes de dispositivos, tendo como base a sua capacidade de memória [14]. É de referir que as divisões entre as classes não são muito precisas. A **classe 0** classifica os

dispositivos com recursos bastante limitados, sendo a representativa dos RFD. A **classe 1** é para dispositivos com cerca de 100 kB de memórias RAM (*Random Access Memory*) e de código [15]. Nesta classe podem-se enquadrar ambos os tipos de dispositivos físicos. Finalmente, na **classe 2** encontram-se os dispositivos de melhores capacidades, os FFD.

Tabela 2.1: Classes dos dispositivos de recursos limitados

	Capacidade de armazenamento de dados	Capacidade de armazenamento de código
Classe 0	$\ll$ 10 kB	$\ll$ 100 kB
Classe 1	$\sim$ 100 kB	$\sim$ 100 kB
Classe 2	$\sim$ 50 kB	$\sim$ 250 kB

Uma rede IEEE 802.15.4 tem por base uma das duas topologias, estrela ou ponto a ponto, representadas na Figura 2.1. A topologia em estrela rege-se por uma comunicação direta entre os dispositivos das extremidades e o coordenador PAN. É uma organização dos dispositivos simples, em que o dispositivo central deve estar ligado a uma fonte de energia, mas possibilita que os restantes dispositivos tenham uma maior eficiência energética, permitindo uma alimentação através de baterias.

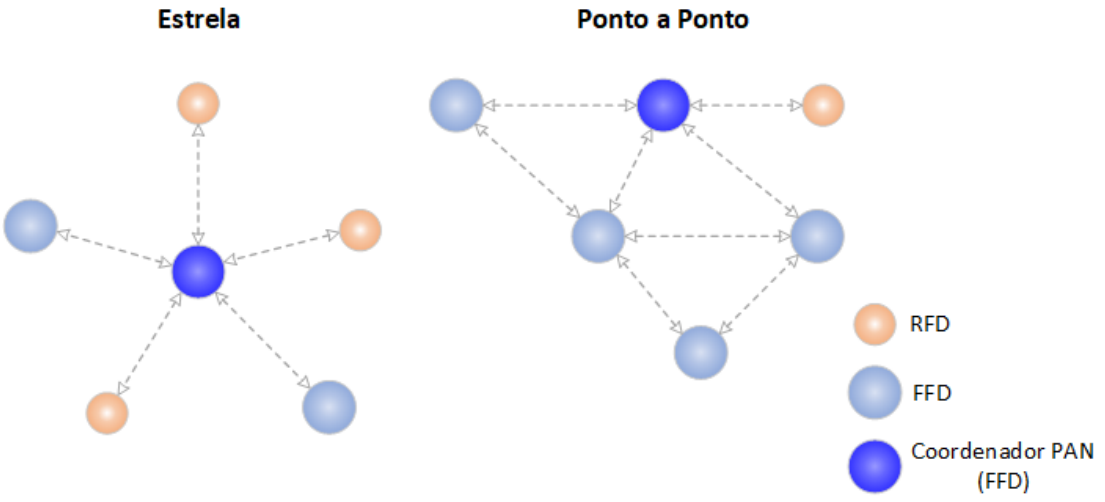


Figura 2.1: Topologias de rede

Numa topologia ponto a ponto, os dispositivos FFD comunicam entre si com o objetivo de determinada mensagem chegar ao coordenador PAN. Ao contrário da topologia em estrela, para além da aquisição de dados do meio ambiente, os dispositivos FFD também poderão funcionar

como retransmissores das mensagens recebidas. Uma rede ponto a ponto possui a capacidade de se auto-organizar, potenciando uma maior cobertura da rede [16].

2.1.1.2 Camada Física (PHY)

A camada física é a base do modelo Open System Interconnect (OSI), e como tal, é responsável por comunicar diretamente com a interface rádio. Assim sendo, as tarefas a que esta camada tem a cargo são as seguintes:

- Ativação e desativação da interface rádio;
- Deteção de energia (ED) em determinado canal;
- Indicação da qualidade de ligação (LQI) para cada trama;
- Seleção de um canal correspondente a uma frequência;
- Transmissão e receção de dados.

A primeira versão do protocolo IEEE 802.15.4, lançada em 2003, suportava a operação nas bandas de 868 MHz e 915 MHz, utilizando a modulação BPSK com taxas de transferência de 20 kb/s e 40 kb/s. Suportava também a banda de 2.4 GHz, utilizando a modulação O-QPSK com uma taxa de transferência de 250 kb/s [17]. Mas nem todas as bandas podem ser usadas livremente. A banda de 868 MHz só é de utilização livre na Europa e a de 915 MHz só é permitida na América do Norte. Ao contrário destas, a utilização da banda de 2.4 GHz é permitida em todo o globo terrestre. Estas restrições de localização são impostas pelas bandas de rádio *Industrial, Scientific and Medical* (ISM).

Em 2006 foi lançada uma nova versão cujo principal avanço foi o aumento no número de canais. Foi possível aumentar para 3 o número de canais na banda de 868 MHz e de 10 para 30 canais na banda de 915 MHz. Por outro lado, a banda de 2450 MHz manteve-se nos 16 canais. Tal evolução foi conseguida com a introdução de 2 técnicas de modulação numa nova banda opcional de 868/915 MHz [18]. Com a adição de novas opções na camada física, deixou de haver apenas o conceito de canal, passando a existir o conceito de página. Foram reservadas 32 páginas com, no máximo, 26 canais cada. Desta forma, os canais da primeira versão ficaram mapeados na página zero, as novas adições, nas páginas 1 e 2, e as restantes, reservadas para uso futuro. As opções PHY introduzidas foram as seguintes [19]:

- Banda opcional de 868/915 MHz com técnica de espalhamento espectral *Direct Sequence Spread Spectrum* (DSSS) e modulação *Offset Quadrature Phase-Shift Keying* (O-QPSK);
- Banda opcional de 868/915 MHz baseado em *Parallel Sequence Spread Spectrum* (PSSS) usando *Binary Phase-Shift Keying* (BPSK) e *Amplitude Shift Keying* (ASK).

No ano a seguir surgiu uma nova versão que visava encontrar alternativas na camada física. Desta forma, a versão de 2007 acrescentou 2 novas especificações PHY: *Direct Sequence ultrawideband* (UWB), destinada a operar nas faixas de 3 GHz, 5 GHz e 6 GHz a 10 GHz, e *chirp spread spectrum* (CSS) colocando a frequência de 2450 MHz a operar numa taxa de transferência de 1000 kb/s. A grande vantagem do UWB é a capacidade de fornecer taxas de transferência de até 27,4 Mb/s.

Em 2009 foram lançadas algumas emendas que possibilitaram uma maior cobertura geográfica. Estas emendas visavam a compatibilidade de operação nas bandas de frequência disponíveis na China e no Japão. Assim sendo, foram adicionadas as seguintes especificações [19]:

- Banda de 780 MHz com modulação *Offset Quadrature Phase-Shift Keying* (O-QPSK);
- Modulação *M-ary Phase Shift Keying* (MPSK) na banda de 780 MHz;
- Banda de 950 MHz *Direct Sequence Spread Spectrum* (DSSS) usando modulação *Binary Phase-Shift Keying* (BPSK);
- Banda de 950 MHz baseada na modulação *Gaussian Frequency-Shift Keying* (GFSK).

A taxa de transferência na banda de 780 MHz é de 250 kb/s nas 2 modulações existentes enquanto que na banda de 950 MHz, é de 20 kb/s e 100 kb/s nas modulações BPSK e GFSK, respetivamente. No ano de 2011, estas emendas foram colocadas numa versão final. Na tabela 2.2 estão apresentados alguns parâmetros sobre a camada física, tais como, a frequência, o limite de banda, modulação e velocidade de transmissão [20]. Cada frequência da primeira coluna representa um determinado número de canais. Cada canal corresponde a uma frequência fixa compreendida pelos valores do limite de banda.

Tabela 2.2: Frequências permitidas pela norma IEEE 802.15.4-2011

Frequência (MHz)	Limites de banda	Modulação	Taxa de transferência (kb/s)
780	779-787	O-QPSK; MPSK	250
868	868-868.6	ASK; BPSK; O-QPSK	20;100;250
915	902-928	ASK; BPSK; O-QPSK	40;250
950	950-956	BPSK; GFSK	20;100
2450 DSSS	2400-2483	O-QPSK	250
2450 CSS	2400-2483	CSS/DQPSK	250;1024
UWB sub-GHz	250-750	BPM/BPSK	110;850; 6800;27240
UWB low band	3244-4742		
UWB high band	5944-10234		

2.1.1.3 Subcamada MAC

A subcamada *Medium Access Control* (MAC) interliga a camada física com a camada de rede. As principais funcionalidades são as seguintes [20]:

- Geração dos *beacons frames* no caso de ser o coordenador da rede;
- Sincronização dos mesmos;
- Suporte à associação e desassociação da rede PAN;
- Permite serviços de segurança sobre os dados;
- Gestão do mecanismo GTS (*Guaranteed Time Slots*);
- Implementação do mecanismo *Carrier Sense Multiple Access with Collision Avoidance* (CSMA-CA).

A comunicação de dispositivos IEEE 802.15.4 no mesmo canal é possível, devido ao mecanismo CSMA-CA. Antes do dispositivo enviar uma mensagem, este verifica se o meio está livre, utilizando

uma de duas técnicas: análise da energia espectral na frequência a utilizar, *Energy Detection* (ED), ou através da análise ao tipo de sinal presente no canal, *(Clear Channel Assessment* (CCA). Caso o meio esteja ocupado, o dispositivo aguarda um tempo aleatório, e tenta novamente o envio. Este processo é repetido até o meio estar livre ou o número de tentativas for excedido.

O acesso ao meio é baseado numa combinação de acesso aleatório e acesso agendado [21]. Nos dois casos, é o coordenador PAN que controla os acessos recorrendo a um dos dois métodos: *beacon-enabled* ou *nonbeacon-enabled*. No primeiro método, o coordenador envia tramas para sincronizar os dispositivos da rede. Estas tramas, denominadas de *beacons*, são enviadas e modo síncrono pelo coordenador e o intervalo de tempo delimitado por 2 *beacons* consecutivos, constitui uma supertrama. Esta inclui um *Contention Access Period* (CAP), um *Contention Free Period* (CFP) opcional, podendo também ter um período de tempo inativo. A supertrama é dividida em 16 *slots* de tempo e a sua estrutura é definida pelo coordenador, podendo ser composta por:

- **Apenas por um período de contenção (CAP)**: Os dispositivos que se pretendem conectar à rede PAN devem usar o mecanismo *slotted* CSMA-CA sendo que todas as transmissões devem terminar antes do fim do período ativo.

- **Por um CAP e um período de contenção livre (CFP)**: Este último período é constituído por *Guaranteed Time Slots* (GTSs), espaços de tempo que podem ser maiores que um *slot*, para permitir o funcionamento de aplicações de baixa latência.

Em ambos os casos, e por questões energéticas, poderá existir um período inativo em que o coordenador é colocado no modo *low-power*. Na Figura 2.2 encontra-se um exemplo da estrutura de uma supertrama [20]. Como por ser verificado, esta contém um período de contenção livre com dois GTSs e um período inativo.

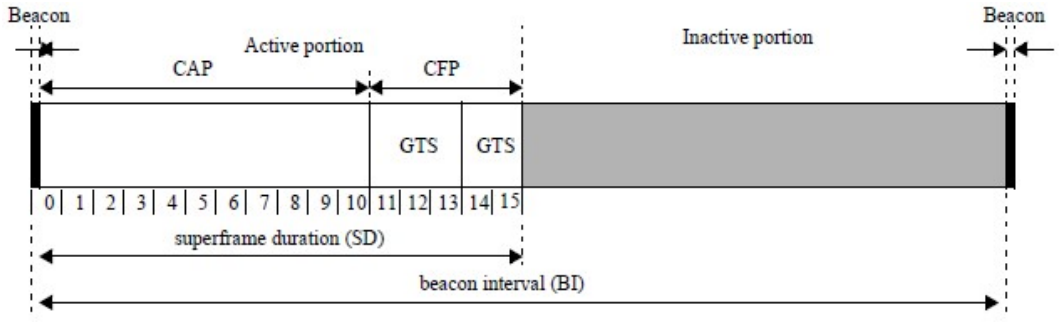


Figura 2.2: Exemplo da estrutura de uma supertrama

No modo *nonbeacon-enable* não existem *beacons* nem supertramas [22]. O acesso ao meio é completamente aleatório, utilizando para o efeito, o mecanismo *unslotted* CSMA-CA. A grande vantagem deste modo é a possibilidade de suspensão dos nós sensores por períodos de tempo maiores. Assim sendo, o uso do modo *beacon-enable* só deverá ser usado em aplicações bastante específicas, pois o consumo energético é superior quando se utiliza supertramas.

Estrutura da trama MAC

O *standard* IEEE 802.15.4 definiu quatro tipos de estruturas de mensagens para a subcamada MAC: os *beacon frames*, mensagens de dados, de *acknowledged*, e de comandos MAC [20].

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Figura 2.3: Formato genérico de uma trama MAC

A estrutura de uma mensagem MAC, representada na Figura 2.3 [20], assenta em 3 principais campos: o cabeçalho ou MHR (*MAC header*), os dados, e o rodapé ou MFR (*MAC footer*). No cabeçalho, os primeiros 2 *bytes* indicam o tipo da trama e o modo de endereçamento. O restante espaço está reservado para o número de sequência, para os endereços de destino e origem, seguido de um campo auxiliar de segurança. Os dados seguem a seguir, e no final da trama, o MFR contém o FCS (*frame check sequence*) calculado a partir do cabeçalho e dos dados.

O tamanho máximo de uma trama é de 127 *bytes*. È de salientar que, para tramas de dados, a camada MAC requere, pelo menos, 5 *bytes* fixos mais 4 *bytes* mínimos de endereçamento. Assim, no máximo, cada trama pode albergar 118 *bytes* de dados [17].

2.1.2 6LoWPAN

A norma *IPv6 over Low-Power Wireless Personal Area Networks* (6LoWPAN) nasceu de uma ideia, em que o protocolo IP (*Internet Protocol*) deveria ser aplicado até nos mais pequenos dispositivos [23]. Em 2007, o grupo IETF publicou esta norma, permitindo que dispositivos com suporte ao *standard* IEEE 802.15.4 pudessem enviar mensagens baseadas no protocolo *Internet Protocol version 6* (IPv6) [24], mais concretamente nos RFC 4419 [25] e RFC 4944 [26]. Inicialmente, idealizou-se aplicar este conceito apenas nas LR-WPAN na banda de 2.4 GHz, mas tem-se vindo a adaptar em outras tecnologias, tais como, *Sub-1 GHz low-power RF*, *Bluetooth*, entre outras [27]. Mais recentemente, este protocolo tem vindo a ser implementado em plataformas *Field-Programmable Gate Array* (FPGA) [28].

Porquê IPv6?

Nos dias de hoje, a Internet ainda é essencialmente baseada em IPv4 (*Internet Protocol version 4*) e dado que utiliza endereçamento de 32 bits, apenas é possível obter cerca de quatro mil milhões de endereços únicos. Acontece que, mesmo com o uso de técnicas de tradução como o NAT (*Network Address Translation*), em 2011, os endereços IPv4 já tinham sido todos atribuídos [27]. Para resolver este problema, surge em 2006 uma nova versão do protocolo da Internet, o IPv6. Utiliza endereçamento de 128 bits suportando cerca de $3,4 \times 10^{38}$ dispositivos conectados. Para além disso, o uso do IP apresenta as seguintes vantagens [29]:

- **Segurança**: O IPv6 suporta o uso da autentificação e encriptação *IP Security* (IPsec) [30] e de serviços *web* que geralmente utilizam *sockets* seguros ou mecanismos de segurança na camada de transporte.
- **Serviços *Web***: A utilização destes serviços é vasta, como por exemplo, o TCP (*Transmission Control Protocol*) ou o HTTP (*Hypertext Transfer Protocol*).
- **Tamanha da trama**: A quantidade de dados que podem ser enviados numa trama é relativamente grande (1280 *bytes* no IPv6).

Numa rede, a colocação de dispositivos que utilizam IP nas suas comunicações, simplifica o modelo de conectividade [23]. Em redes que utilizam normas como a *Zigbee* ou *WirelessHART*,

a sua ligação com o protocolo da internet é realizada através de *gateways* complexos. Com a utilização do 6LoWPAN, a conexão pode ser efetuada com recurso a um simples *router*. Do ponto de vista da segurança e da necessidade de endereços, o IPv6 tornou-se numa tendência inevitável de uso nas WSNs e no próspero mundo da *Internet of Things* (IoT) [31].

2.1.2.1 Arquitetura do 6LoWPAN

Uma WPAN (*Wireless Personal Area Network*) é uma rede que interliga dispositivos num determinado espaço onde as conexões são *wireless* [32]. Numa *Low Power Wireless Personal Area Networks* (LoWPAN), só existe tráfego de dados entre dispositivos internos. Por essa razão, as LoWPAN são redes *stub*. Normalmente, estas redes apenas têm um caminho de interligação com a Internet [33]. Uma 6LoWPAN é constituída por redes LoWPAN que utilizam IPv6. Uma particularidade destas redes é a partilha do mesmo prefixo de endereçamento, nomeadamente, nos primeiros 64 bits.

A arquitetura 6LoWPAN define 3 topologias: *ad hoc*, simples e extendida. Uma LoWPAN *ad hoc* não está conectada à Internet sendo que os dispositivos apenas comunicam entre si. Uma LoWPAN simples possui um *edge router*, funcionando como *gateway*, para interligação com outra rede IP. Por fim, uma rede que possui múltiplos *edge routers* conectados a uma ligação *backbone*, é denominada de LoWPAN extendida. Na Figura 2.4 está apresentada uma arquitetura 6LoWPAN simples pois possui apenas um *gateway* para interligação com a Internet [34].

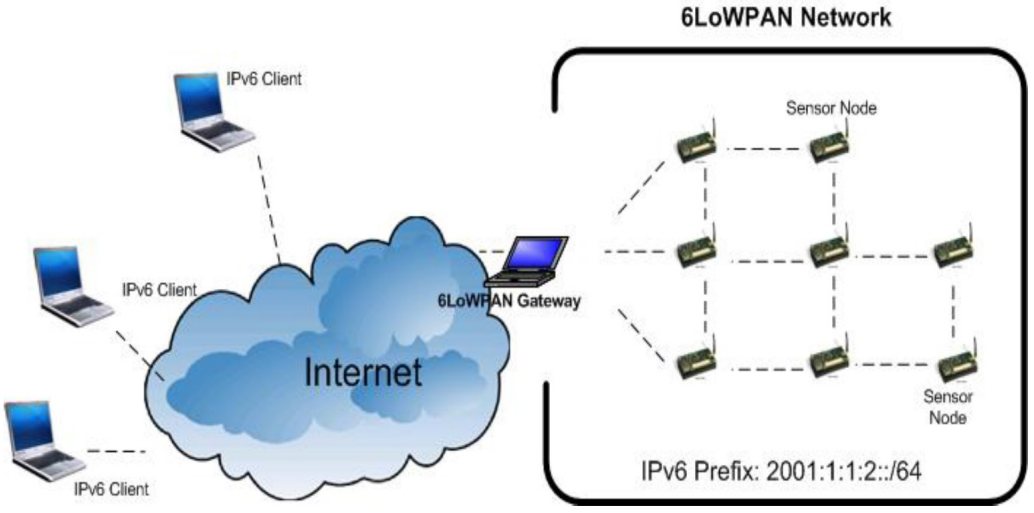


Figura 2.4: Exemplo de uma arquitetura 6LoWPAN simples

2.1.2.2 Pilha Protocolar

A Figura 2.5 apresenta uma comparação entre as pilhas protocolares IP e 6LoWPAN, nas camadas definidas pelo *Internet Model* [29]. As camadas física e de *link* de dados correspondem às camadas física e MAC da norma IEEE 802.15.4, respetivamente. Incorporado na camada de rede, o 6LoWPAN, é dividido em duas partes, numa camada de adaptação (LoWPAN) e numa implementação IPv6. Desta forma, as necessárias compressões são realizadas na camada de adaptação, interligando o protocolo IEEE 802.15.4 com o IPv6.

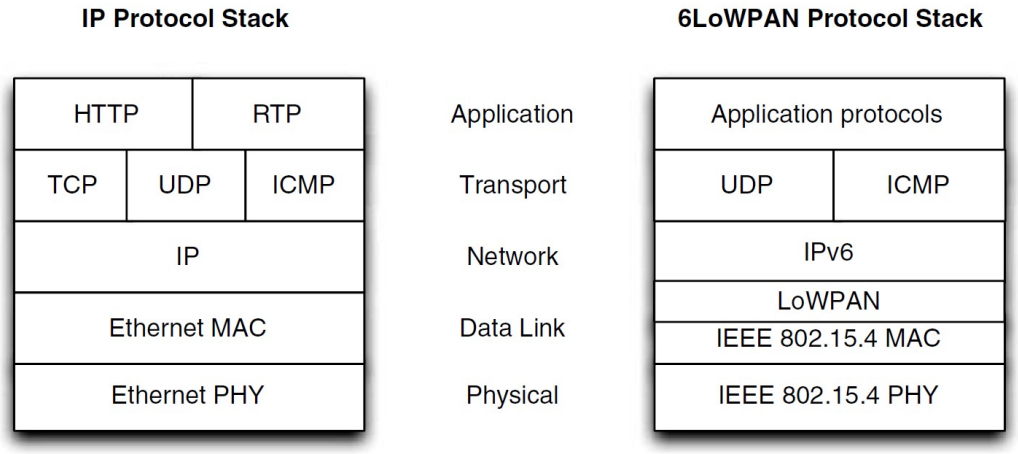


Figura 2.5: Pilhas protocolares do IP e do 6LoWPAN

Na camada de transporte, é utilizado o protocolo *User Datagram Protocol* (UDP) que também pode ser comprimido [29]. Apesar de na Figura 2.5 não estar representado, o protocolo *Transmission Control Protocol* (TCP) também é utilizado, mas devido à sua complexidade e baixa eficiência em comunicações de banda larga reduzida, a sua utilização é menor. Já o *Internet Control Message Protocol version 6* (ICMPv6) é um protocolo da camada de rede e é utilizado, por exemplo, para o *Neighbor Discovery* (ND). A camada de aplicação localiza-se no topo da pilha protocolar, podendo estar implementado protocolos como o *HyperText Transfer Protocol* (HTTP) ou o *Constrained Application Protocol* (CoAP).

2.1.2.3 6LoWPAN como Camada de Adaptação

A integração do IP em redes LoWPAN era impensável por muitos desenvolvedores, pois assumiam que este protocolo era demasiado pesado para os dispositivos de recursos limitados, característicos deste tipo de redes [35]. O 6LoWPAN desmitificou essa ideia introduzindo uma camada de

adaptação, entre a camada de *link* de dados (subcamada MAC IEEE 802.15.4) e a camada de rede. Desta forma, é possível enviar tramas IPv6 através de rádios IEEE 802.15.4.

Aquando do desenvolvimento do 6LoWPAN, o IETF focava-se essencialmente na eficiência energética em redes *Low-Power and Lossy Networks* (LLNs). Assim sendo, esta norma especifica três importantes mecanismos [35]:

- **Compressão do cabeçalho**: Os parâmetros que podem ser obtidos através da trama IEEE 802.15.4, ou mesmo subentendidos a partir do contexto, são omitidos no cabeçalho do 6LoWPAN.

- **Fragmentação e remontagem**: Alguns dados têm de ser divididos em várias tramas devido ao *Maximum Transmission Unit* (MTU) do IPv6 ser de 1280 *bytes* e das tramas IEEE 802.15.4 ser apenas de 127 *bytes*.

- **Encaminhamento de tramas na segunda camada**: Uma trama, até chegar ao destino, necessita de ser reencaminhada por outros dispositivos da rede. Esse processo pode ser realizado pela camada de *link* de dados ou pela camada de rede.

Compressão do cabeçalho

Em conexões ponto a ponto, as compressões são *status-based*, em que o cabeçalho da trama a enviar apenas contem as diferenças em relação ao cabeçalho anterior. Em redes dinâmicas, em que os saltos necessários para um pacote chegar ao destino podem estar constantemente em mudança, tal já não pode ser aplicado. Desta forma, surge a necessidade de criar tramas com cabeçalhos independentes [36]. O conceito chave do 6LoWPAN consiste no uso de compressões *stateless* e de contexto partilhado. Cada trama é comprimida de forma independente e não é influenciada por cabeçalhos anteriormente enviados.

A eliminação de informação redundante no cabeçalho das várias camadas é um dos métodos aplicados por esta compressão. Os campos que contêm os tamanhos do UDP/IPv6 e dos endereços IPv6 são omitidos, pois podem ser derivados de camadas inferiores [27]. O outro método consiste na total remoção de campos que, por definição, possuem o mesmo valor. É o caso do *Traffic Class* e da *Flow Label*, em que os seus valores são sempre zero [36]. Assim, os cabeçalhos das 3 camadas superiores (rede, transporte e aplicação) podem ser reduzidos a

apenas alguns *bytes*.

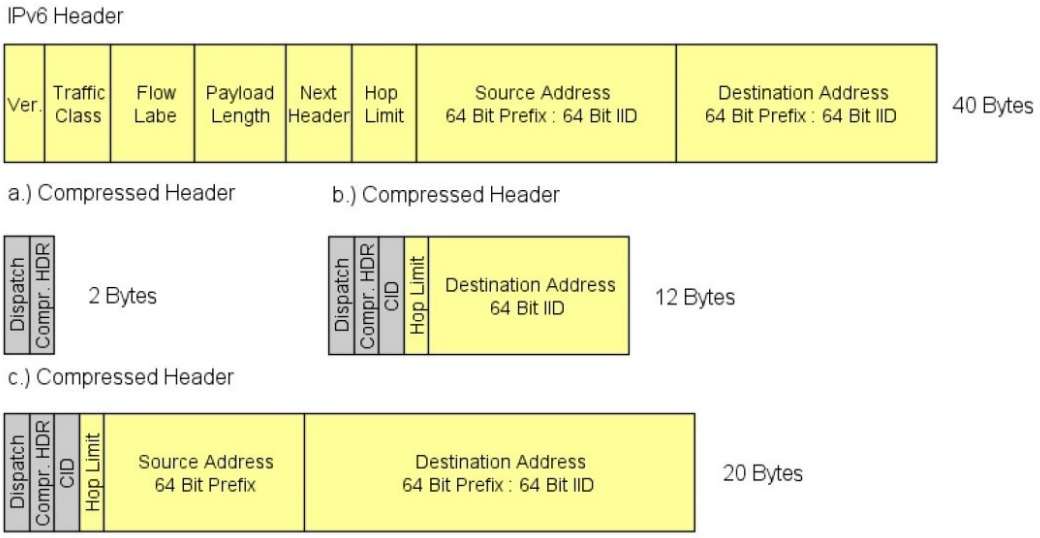


Figura 2.6: Exemplos de cabeçalhos 6LoWPAN

A Figura 2.6 apresenta alguns exemplos da compressão do cabeçalho IPv6, bem como uma comparação deste, com o cabeçalho IPv6 normal [36]. Se a comunicação for apenas dentro da mesma rede 6LoWPAN, o cabeçalho IPv6 pode ser reduzido até 2 *bytes* (alínea a). Quando a comunicação é entre duas redes, existem dois cenários. Se o prefixo da rede for conhecido, a redução é para 12 *bytes* (alínea b), senão, apenas é conseguida uma atenuação de 50% relativamente aos 40 *bytes* utilizados no IPv6 normal (alínea c) [27].

Os dados encapsulados pelo 6LoWPAN são precedidos por uma pilha de cabeçalhos, nomeadamente, os cabeçalhos de endereçamento *Mesh*, *Broadcast*, Fragmentação, IPv6 e UDP [37]. De forma a identificar os possíveis cabeçalhos na trama, foi definido o campo *Dispatch* de apenas 1 *byte*. A Tabela 2.3 apresenta as 4 categorias de cabeçalhos identificados pelos primeiros 2 *bits* desse campo [38][39]:

- ***No 6LoWPAN*** (00): Estas tramas, que não são 6LoWPAN, são descartadas permitindo a coexistência de outros protocolos na mesma rede.
- ***Dispatch*** (01): Enquadra-se nesta categoria os cabeçalhos IPv6 compressos e de *multicasting*.
- ***Mesh Addressing*** (10): Permite que tramas IEEE 802.15.4 possam ser encaminhadas pela camada de enlace de dados.

- **Fragmentação** (11): É utilizado quando os dados a enviar não cabem numa simples trama IEEE 802.15.4.

Tabela 2.3: Tipos de cabeçalho 6LoWPAN

First 2 bits		Following bit combinations	
No 6LOWPAN	00	xxxxxx	Any combination
Dispatch	01	000000	Additional Dispatch byte follows
		000001	Uncompressed IPv6 Addresses
		000010	LOWPAN_HC1 compressed IPv6
		010000	LOMPAN_BC0 broadcast
		1xxxxx	LOWPAN_IPHC
Mesh Addressing	10	xxxxxx	Any combination
Fragmentation	11	000xxx	First Fragmentation Header
		100xxx	Subsequent Fragmentation Header

Nos restantes 6 bits do campo *Dispatch* é especificado alguns detalhes do cabeçalho, nomeadamente, o tipo de compressão utilizado ou se é o primeiro fragmento. No RFC4944 [26], a compressão LOWPAN_HC1 utiliza oito bits para especificar os possíveis cabeçalhos compressos, cujos quais, serão acoplados a este *byte*. Os primeiros dois grupos de dois *bits* especificam os endereços de origem e destino, respetivamente. O bit 4 indica a existência do *Traffic Class* e do *Flow Label*, os bits 5 e 6 especificam o protocolo do cabeçalho de transporte (UDP, ICMP ou TCP), e finalmente, o bit 7 indica a presença do codificador HC2 [37]. Por sua vez, este codificador possibilita a redução do cabeçalho UPD, de oito para quatro *bytes*. O TCP e o ICMP não são comprimidos.

Em 2011, no RFC6282 [40], surge um novo codificador, LOWPAN_IPHC, que vem melhorar a compressão do cabeçalho, aquando da comunicação com endereços globais e *multicast*. Utiliza 3 *bits* para o *Dispatch Type* e 13 bits para o campo de codificação LOWPAN_IPHC. No melhor caso, o cabeçalho IPv6 pode ser reduzido até 2 *bytes*, e quando o pacote precisa de efetuar múltiplos saltos para chegar ao seu destino, o LOWPAN_IPHC consegue reduzi-lo até 7 *bytes* [38].

Os cabeçalhos de um pacote IPv6 também existem numa trama 6LoWPAN, mas de uma

forma mais compacta. Uma técnica de compressão destes cabeçalhos, providenciada pelo 6LoWPAN, é o LOWPAN_NHC, em que cada *next-header* é identificado por um *standard* de *bits*. Na Figura 2.7 está apresentado um exemplo do 6LoWPAN/UDP com uma compressão do cabeçalho IPv6, LOWPAN_IPHC, seguido de uma compressão UDP, LOWPAN_NHC [41].

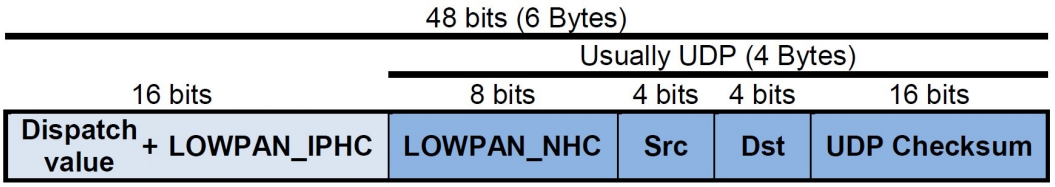


Figura 2.7: Exemplo de cabeçalho compresso 6LoWPAN/UDP

O cabeçalho UDP pode ser reduzido com recurso a este codificador. O valor do comprimento utilizado pelo UDP é eliminado, pois o seu valor pode ser determinado a partir de camadas inferiores. Já o *checksum* pode ser retirado se as camadas superiores assim o permitirem. As portas de origem e destino são comprimidas até 4 bits cada, permitindo apenas portas num intervalo de 61616 a 61631. No melhor dos casos, o cabeçalho UDP pode reduzido até 2 *bytes*. Salienta-se que, comparando com o *standard* IPv6/UDP, este codificador consegue uma redução do cabeçalho, de 48 para apenas 6 *bytes* [29].

Fragmentação

Um dos grandes problemas da utilização do IPv6 nas redes LR-WPAN é o tamanho máximo do pacote a transmitir. O MTU do IPv6 é de 1280 *bytes* enquanto que o do IEEE 802.15.4 é de 127 *bytes* apenas. Como visto anteriormente, as técnicas de compressão do cabeçalho conseguem aumentar a quantidade de dados que é possível enviar numa trama, mas, nem sempre é o suficiente. Na melhor das hipóteses, cada trama 6LoWPAN que utilize o *standard* IEEE 802.15.4, pode albergar, no máximo, cerca de 100 *bytes* de dados [42]. Caso a quantidade de dados a enviar exceda esse limite, é necessário fragmentar as tramas.

A fragmentação é uma técnica permitida por este *standard* em que cada fragmento deve conter a mesma "*tag*". Para distinguir os pacotes subsequentes, é utilizado um campo de *offset* (1 *byte*), transportado apenas no segundo e seguintes pacotes [37]. Apesar desta funcionalidade ser bastante útil em redes em malha, é de notar que, a sua utilização deve ser evitada. Um dos problemas associados reside na reconstrução da trama que pode estar propensa a erros. Caso

se perca um dos fragmentos, a remontagem não é realizada, e como tal, todos os fragmentos têm de ser enviados novamente [42].

Routing

Durante o envio de um pacote, desde a origem até ao seu destino, por vezes é necessário que este efetue vários saltos. É o *routing* que define o trajeto que a trama deve percorrer. Dependendo da camada onde se encontra implementado este mecanismo, são definidas duas categorias: *mesh-under* ou *route-over* [27]. Neste primeiro, a segunda camada recorre aos endereços MAC, ou aos *short addresses* de 16 bits, para encaminhar os pacotes. No *route-over*, é aplicado o endereçamento IP implementado na terceira camada [36]. Na Figura 2.8 está apresentada uma comparação entre estas duas categorias de *routing* [35].

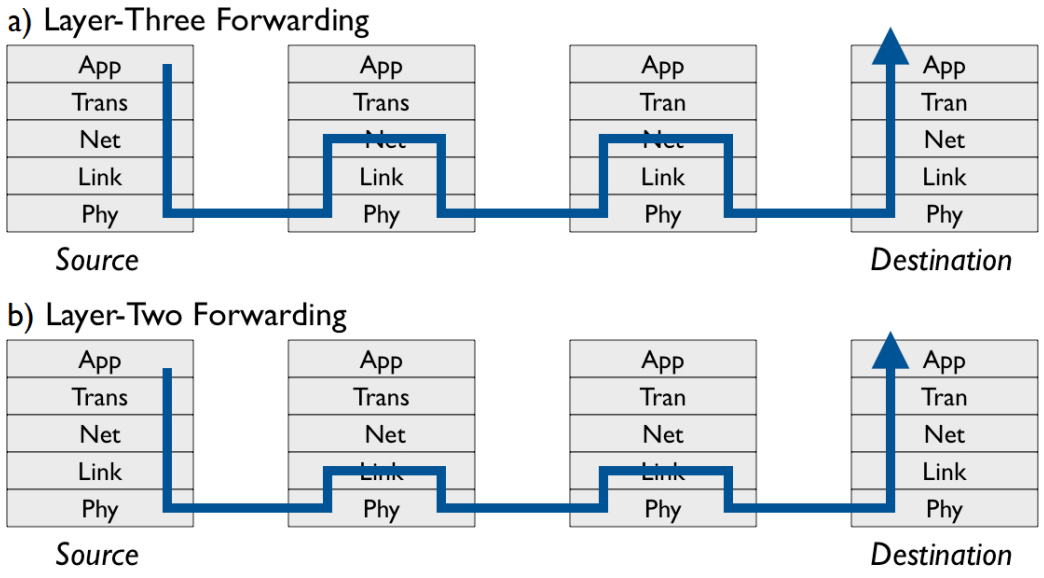


Figura 2.8: Route-over(a) *vs* Mesh-under(b)

Num sistema *mesh-under*, o encaminhamento das tramas ocorre de forma transparente. De forma a oferecer compatibilidade com os protocolos IPv6 implementados em camadas superiores, como por exemplo, a deteção de endereços duplicados, os pacotes são enviados num domínio *broadcast*. Como as mensagens são enviadas para todos os dispositivos na rede, pode ocorrem uma sobrecarga de informação na rede. Logo, este tipo de sistema apenas deve ser usado em redes com poucos dispositivos. [27].

Por outro lado, um sistema *route-over* utiliza a camada IP para o *routing*, em que cada dispositivo atua como um *router* IP. Desta forma, cada nó sensor possui algumas das funcionalidades de um router IP, tais como, *Neighbor Discovery* ou configuração de endereços [36]. A utilização do endereçamento IP é um princípio básico para a independência de camadas inferiores, facilitando a integração em redes de grandes dimensões.

Um dos protocolos para redes *route-over* é o *Routing Protocol for Low power and Lossy networks* (RPL). O grupo do IETF, *Routing over Low-power and Lossy links* (ROLL), padronizou um protocolo baseado em IPv6, independente da camada de enlace de dados, para dispositivos com recursos limitados. Suporta os mais variados modelos de tráfego de dados, nomeadamente, multiponto-ponto e ponto-multiponto, aquando do envio de dados de dentro da rede 6LoWPAN para um dispositivo central e vice-versa. Para além disso, também permite o tráfego ponto-ponto [39].

A base do RPL consiste na utilização de *Destination-Oriented Direct Acyclic Graphs* (DODAGs). A particularidade de uma *Direct Acyclic Graph* (DAG) é a de que todos os dispositivos estão orientados duma forma em que não existem ciclos no encaminhamento dos dados. Uma DODAG, que se encontra apresentada na Figura 2.9 [39], é apenas uma DAG orientada a um único destino, o DAG *root* [43].

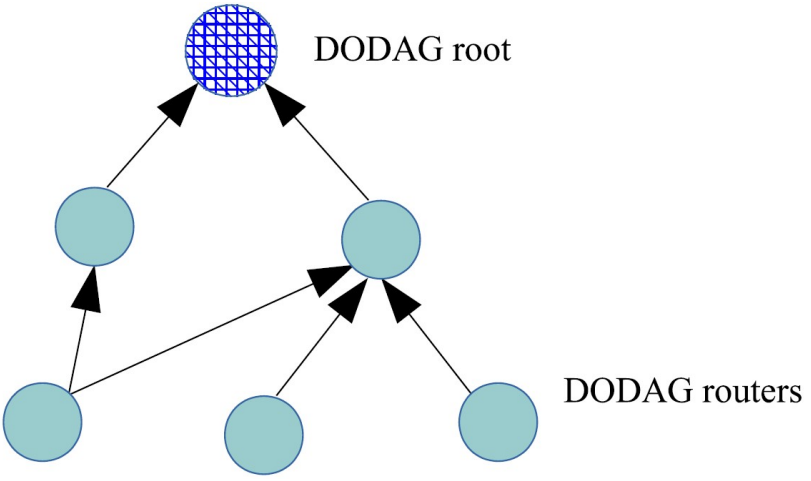


Figura 2.9: Topologia de uma DODAG

Este protocolo recorre a 4 tipos de mensagens. A mais importante é a *DODAG Information Object* (DIO) pois contém informações sobre o *rank* atual do nó, ou seja, a distância entre este e a raiz, o que consequentemente, levará à escolha do caminho preferencial para o encaminhamento

de dados [39]. O tipo de mensagem *DODAG Advertisement Object* (DAO) propaga informações de destino no sentido filhos para pais, ou seja, dos *routers* mais afastados em direção à raiz [44]. Dependendo do modo de operação, as mensagens DAO são encaminhadas de forma diferente:

- No modo ***non-storing***, estas mensagens informativas dos respetivos pais, são enviadas para a raiz DODAG.
- No modo ***storing***, as mensagens DAO são enviadas de um *router* filho para um *router* pai.

Os outros dois tipos de mensagens são a *DODAG Information Solicitation* (DIS) e a *DAO Acknowledgment* (DAO-ACK). Esta primeira é enviada pelos nós com o intuito de solicitar mensagens DIO aos routers mais próximos. Por outro lado, as DAO-ACK atuam como resposta da receção das mensagens DAO [45].

Neighbor Discovery

O protocolo de *Neighbor Discovery* (ND) do IPv6, para além de descobrir a vizinhança, também é capaz de manter informações de acessibilidade e de configurar os caminhos base. Este processo utiliza mensagens *Internet Control Message Protocol* (ICMP) e endereços *multicast*, para obter os endereços da camada de enlace dos dispositivos vizinhos. Este protocolo recorre a 4 tipos de mensagens [46]:

- *Router Solicitation* (RS): Um *host* ou nó sensor envia este tipo de mensagem para obter os endereços dos *routers* na vizinhança. Aquando da receção por parte de um router, este envia um RA.
- *Router Advertisement* (RA): O *router* anuncia a sua presença juntamente com alguns parâmetros, tais como, de configuração de endereços e o valor máximo de saltos. Os RA podem ser enviados periodicamente ou apenas em resposta às solicitações dos nós.
- *Neighbor Solicitation* (NS): Enviado por um nó com o objetivo de determinar o endereço de um vizinho, ou apenas para verificar a sua acessibilidade.
- *Neighbor Advertisement* (NA): Utilizado na resposta a um NS ou aquando de mudanças do endereço da camada de enlace. Nesta primeira situação, o NA é enviado em *unicast*,

apenas para o dispositivo que o solicitou, enquanto que no segundo cenário, o destino é *multicast*.

2.1.3 ZigBee

O *standard* IEEE 802.15.4 especifica as redes *LoWPAN*, e tem vindo a ser utilizado em aplicações cujo requisito principal é o baixo consumo energético. Geralmente, estas aplicações requerem que os nós sensores possuam capacidades para comunicar com nós longínquos, através de "saltos", em que as mensagens são reencaminhadas por diversos nós, até alcançar o dispositivo final [35].

Tal como foi visto secção 2.1.1, este *standard* apresenta um tamanho da trama reduzido, bem como uma baixa taxa de transmissão, estando a ser usado em dispositivos com recursos limitados. A soma destas restrições levou a que vários vendedores criassem os seus próprios protocolos, como é o caso do ZigBee.

Desenvolvido pela *ZigBee Alliance*, o protocolo ZigBee surge da necessidade de criar um *standard* de redes em malha, que suportassem vários dispositivos [47]. ZigBee e IEEE 802.15.4 são diferentes. ZigBee é um protocolo de rede, suportado apenas pelo seu desenvolvedor, que utiliza serviços de transporte especificados pelo IEEE 802.15.4. A pilha protocolar, apresentada na Figura 2.10, evidencia bem essa diferença [48]. As primeiras duas camadas, física e MAC, são da responsabilidade do *standard* IEEE 802.15.4. Já as restantes são a cargo da *ZigBee Alliance* [49].

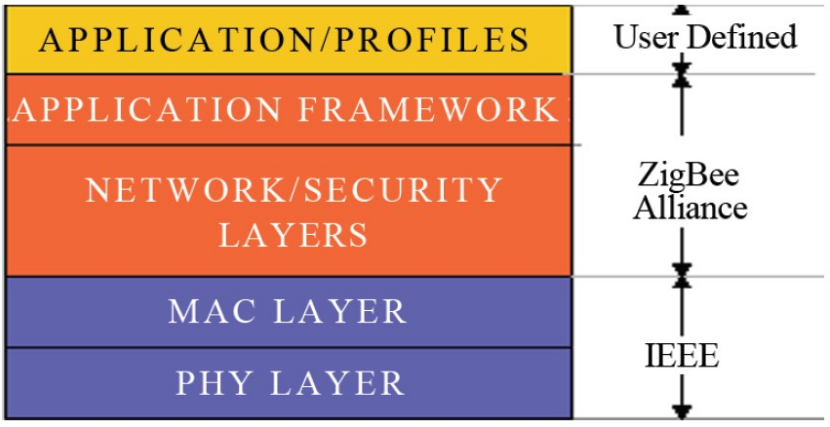


Figura 2.10: Pilha protocolar ZigBee

Existem 3 tipos de categorias de dispositivos num sistema ZigBee: coordenador, *router* e *end*

device. Este primeiro é o que irá configurar toda a rede, informando os outros dispositivos sobre diversos parâmetros, tais como, a topologia a utilizar ou o tamanho dos pacotes. Para além disso, estabelece ligação com um *gateway*, permitindo a comunicação com o exterior da rede ZigBee. Salienta-se o facto de o coordenador ser único numa rede. Um *router* ZigBee tem como função principal, reencaminhar os pacotes recebidos, o que permite aumentar a cobertura da rede. Por fim, o *end device* medirá as variáveis de ambiente. Normalmente, este tipo de dispositivo é *low power* e alimentado por baterias. Não possui a capacidade de retransmitir mensagens, e como tal, não terá necessidade de se manter acordado por longos períodos de tempo, contribuindo para um consumo energético reduzido [49].

São suportadas 3 topologias por uma rede ZigBee: estrela, árvore e em malha. Esta primeira é a mais simples, mas apresenta uma grande desvantagem. Caso o coordenador deixe de funcionar, toda a rede fica comprometida. Numa topologia em árvore, o coordenador pode ter *routers* e *end devices* associados, e a cada *router*, é possível associar outros dispositivos. A informação é passada dos filhos para os pais (*routers*) até chegar ao coordenador, e vice versa. Assim, se um dos *routers* parar de funcionar, apenas os filhos deste perdem a conexão à rede. Por fim, a topologia em malha é a mais flexível das três, pois existem diversos caminhos para a informação ir de um *end device* até ao coordenador [50].

No que se refere a *routing*, geralmente são utilizadas duas técnicas: *tree routing* e *Ad hoc On-Demand Distance Vector Routing* (AODV) [51]. Esta primeira agrupa os dispositivos em pequenos *clusters* e reencaminha as mensagens pelo router responsável de determinado grupo. O método é adequado para dispositivos estacionários, mas o caminho para as mensagens pode não ser o mais apropriado. Por outro lado, o protocolo de *routing* AODV possui um mecanismo de reparação do trajeto dos pacotes. Caso a comunicação entre dois dispositivos falhe, este protocolo utiliza o *broadcasting* para determinar um novo caminho [52].

Um *gateway* ZigBee possui uma interface abstrata entre os protocolos ZigBee e IP. Este traduz endereços e comandos entre os dois lados da rede, de forma a possibilitar a comunicação com o exterior da rede [37]. É de salientar que os detalhes de ambos os protocolos são ocultados durante a tradução das informações. Comparativamente a um *gateway* 6LoWPAN, o da ZigBee é um pouco mais complexo.

6LoWPAN e ZigBee partilham o mesmo protocolo nas camadas física e MAC, e tentam ambos obter uma comunicação *low power*, mas a estratégia utilizada é diferente. No protocolo ZigBee,

o *duty-cycle* de receção de dados é fixa. As mensagens são enviadas assincronamente, e como tal, estas podem ser transmitidas nos intervalos de tempo cujo recetor se encontra inativo, e desta forma, provocar atrasos na comunicação. Já no 6LoWPAN, o *duty-cycle* dos nós sensores é dinâmico, variando consoante a quantidade de tráfego de mensagens [53].

2.1.4 Thread

Um dos mais recentes protocolos de rede é o Thread. Criado pelo *Thread Group* no ano de 2015, este protocolo pretende criar a melhor forma de conectar e controlar dispositivos em casa [54]. Baseado em *open standards*, Thread é uma tecnologia de rede *mesh, low power*, destinada a produtos IoT. Baseada em IPv6, é projetada para ser segura e *future-proof*. A Figura 2.11 apresenta a *stack* deste protocolo [55] [56]. Como é possível verificar, as primeiras 2 camadas utilizam o IEEE 802.15.4, sendo que as restantes correspondem ao protocolo em discussão.

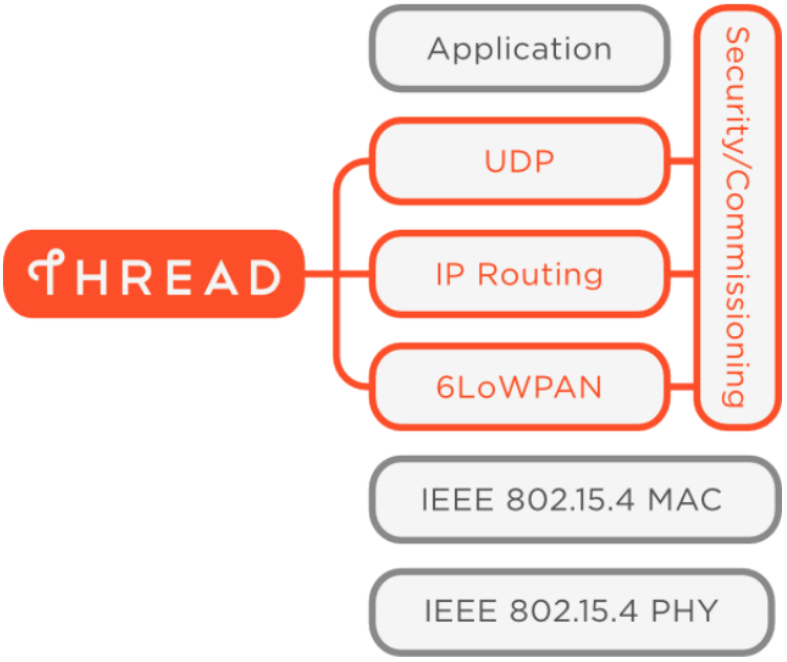


Figura 2.11: Pilha protocolar Thread

Thread tem como base o 6LoWPAN, e como tal, suporta as populares camadas de aplicação existentes, bem como plataformas IoT. Para além disso, uma rede Thread proporciona as seguintes características [57]:

- **Início e operação simples**: Os protocolos de conexão e gestão da rede permitem que esta se auto-configure e repare as conexões entre os dispositivos;

- **Segura**: Apenas os dispositivos permitidos é que se podem conectar. Por outro lado, as comunicações são encriptadas;

- **Elevado alcance**: Os dispositivos, quando interligados numa rede em malha, possibilitam a cobertura de um grande àrea, como por exemplo, uma casa;

- **Baixo consumo energético**: Os dispositivos *host* conseguem ser alimentados por pilhas comuns durante anos.

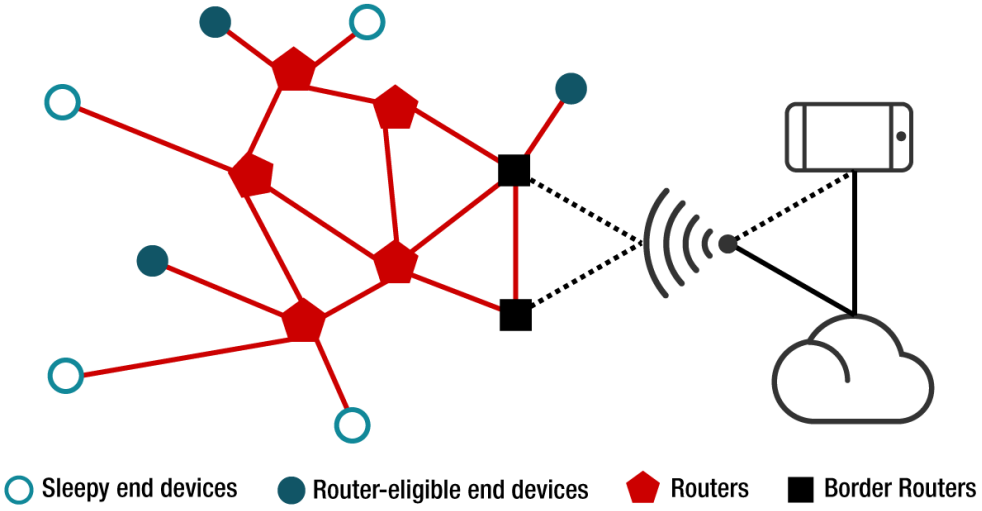


Figura 2.12: Exemplo de uma rede Thread

A topologia em malha é a que permite uma maior flexibilidade nas conexões com os dispositivos, sendo esta a utilizada por uma rede Thread. Na Figura 2.12 está representado um exemplo desta rede, bem como os vários tipos de dispositivos [58]. Ao contrário do protocolo ZigBee, Thread possui 4 tipos de dispositivos [55, 58]:

- ***Border router***: Efetua a conexão entre a rede IEEE 802.15.4 e a rede adjacente, normalmente, *Wi-Fi* ou *Ethernet*. Podem existir um mais *border routers* na mesma rede.

- ***Routers***: Fornecem serviços de roteamento dos pacotes bem como serviços de conexão e segurança na rede. Normalmente, este tipo de dispositivos permanecem sempre acordados.

- ***Router-eligible End Devices* (REED)**: Possuem capacidade para ser *routers*, mas não atuam como estes. Normalmente, são utilizados pela rede como *routers* aquando da necessidade da própria rede, sendo os mesmos geridos por esta.

- ***Sleepy End devices***: Também denominados de *hosts*. Apenas comunicam com o seu router pai e não reencaminham qualquer mensagem. Geralmente são operados a baterias.

2.1.5 Resumo

No decorrer deste subcapítulo foram abordados diversos protocolos de comunicação sem fios, tais como, o IEEE 802.15.4, e três protocolos que recorrem aos serviços de transporte deste, mais concretamente, o 6LoWPAN, ZigBee e Thread. Poderiam ser discutidos outros *standards*, como por exemplo, o Wi-Fi ou o *Bluetooth Low Energy* (BLE), mas estes provam não ser adequados para o âmbito da dissertação. O Wi-Fi possui elevado *throughput* mas o seu consumo energético torna-o demasiado dispendioso. Por outro lado, a versão 4 do Bluetooth, apesar de ser *low power*, apresenta um alcance de transmissão de dados reduzido [59].

É de notar que o IEEE 802.15.4 oferece dois modos de operação, *beacon-enable* e *nonbeacon-enable*. Ao contrário de uma rede ZigBee, que pode optar por um destes modos, o 6LoWPAN recorre sempre ao modo *nonbeacon-enable*. Os protocolos IP assumem que o rádio encontra se sempre ligado, apesar de este ficar a maior parte do tempo desligado [53]. Em relação ao *routing*, o protocolo RPL do 6LoWPAN apresenta um eficiência de 99,9% no reencaminhamento de pacotes. No protocolo ZigBee, o AODV apenas conseguiu uma percentagem de 37,3% [52],

No que se refere à conectividade, um *gateway* ZigBee é mais complexo que um *gateway* 6LoWPAN. Na rede ZigBee, o *gateway* necessita de aceder às camadas superiores para extrair informações dos pacotes provenientes da sua rede, e convertê-las para o protocolo IP. Já numa rede 6LoWPAN, a transição de mensagens é praticamente direta, visto que o protocolo entre as duas redes é o mesmo [37].

2.2 Sistemas ciber-físicos

Um sistema ciber-fisico, do inglês *Cyber-Physical System* (CPS), resulta da integração da computação com os processos físicos [60]. Por outro lado, há quem defenda que se trata de um sistema que combina os elementos ambientais com a parte computacional [61]. Dados adquiridos do meio ambiente, bem como atuação no meio em que se inserem, correspondem aos elementos ambientais. A partir do momento que existe uma tradução dos dados do meio ambiente para o mundo digital, é da responsabilidade da computação tratar esses mesmos dados [61].

Os CPS monitorizam e controlam o mundo físico, havendo a possibilidade de existirem redes de sensores, bem como, atuadores associados [62]. Desta forma, este tipo de sistemas ficam dependentes da sinergia entre os componentes físicos e computacionais. Por outro lado, e ao contrário dos sistemas embebidos tradicionais, os CPS realçam um visão holística do sistema, ou seja, este é visto como um todo, e não apenas como vários módulos isolados [63]. A Figura 2.13 apresenta uma arquitetura básica de um CPS [64].

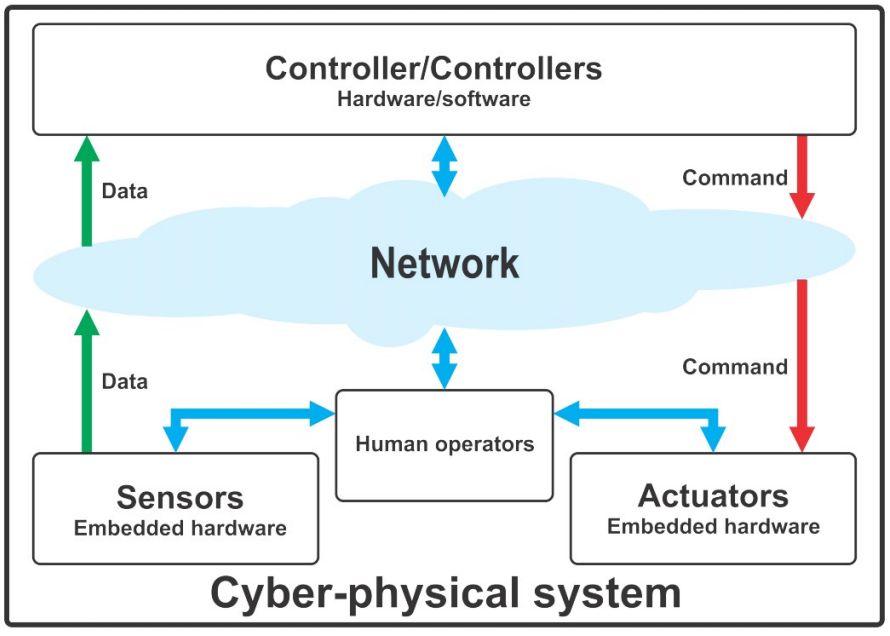


Figura 2.13: Arquitetura de um CPS

O termo CPS surgiu em 2006 na Fundação Nacional da Ciência, nos Estados Unidos [65]. Este termo tem como base, a cibernética, criada por Wiener durante a segunda guerra mundial. São aplicados nas mais diversas áreas, nomeadamente, na Indústria 4.0, *smart grids*, veículos computorizados, redes de sensores sem fios, medicinal e praticamente em todos os dispositivos

que utilizem IoT [66].

2.2.1 *Wireless sensor networks*

Uma *Wireless Sensor Network* (WSN), ou rede de sensores sem fios, consiste num conjunto de dispositivos com capacidade de adquirir dados do meio ambiente, e enviá-los para um dispositivo central. Este, por sua vez, terá de gerir essa informação, podendo-a armazenar localmente ou enviá-la para a camada superior, a Internet. Os dispositivos que obtêm os dados do ambiente no qual estão inseridos são denominados de nós sensores. Normalmente, estes são de tamanho reduzido com baixa/média capacidade de processamento [67].

O dispositivo central ou coordenador, para além de receber os dados provenientes dos nós sensores, utilizando uma tecnologia sem fios, também é responsável por formar e coordenar a rede. Inicialmente, os nós são considerados órfãos, estando constantemente a enviar pequenas mensagens de associação, esperando a resposta de um coordenador [68]. Por sua vez, este transmite uma resposta atribuindo-lhe uma identificação na rede.

Uma WSN simples é composta por apenas dois tipos de dispositivos, nó sensor e coordenador, mas também poderá conter *routers*. Estes possuem a capacidade de retransmitir tramas, podendo também desempenhar a mesma função dos nós sensores [68].

Este tipo de redes é aplicado em diversas àreas, como é exemplo, àreas da saúde [69], agricultura [70], monitorização de ecopontos [71], entre outras. Uma outra àrea de aplicação é para deteção de colisões, cuja qual será apresentada em mais detalhe no seguinte subcapítulo.

2.2.2 Aplicações das WSNs

2.2.2.1 WSN para deteção de colisões

Ao longo dos últimos anos foram desenvolvidas várias pesquisas e implementações de sistemas que detetam colisões de veículos com as guardas de segurança. Em [72] foi desenvolvido um sistema que utiliza a vibração causada pelo embate do veículo nas guardas de segurança, para iniciar a gravação de imagens. Esta rede de sensores sem fios é composta por quatro unidades essenciais. A primeira é representada pelos nós sensores, em que utilizam um acelerómetro para detetar a colisão do veículo com a guarda de segurança. A aceleração é medida a uma frequência

fixa, e quando ultrapassa um determinado *threshold*, é enviado um sinal para a segunda unidade recorrendo ao protocolo IEEE 802.15.4.

A segunda unidade é essencialmente composta por um câmara fotográfica imóvel, que após receber o sinal de deteção do acidente inicia a captura de várias imagens. Posteriormente, estas são enviadas para uma estação central através de uma terceira unidade, responsável pela transmissão dessas imagens. Por usa vez, utiliza o protocolo de comunicação sem fios IEEE 802.15.4 para o envio das imagens.

Finalmente, a quarta e última unidade é a estação central que recebe as imagens capturadas pela segunda unidade. A duração da receção total das imagens é de 3 a 5 minutos. A qualquer momento, a estação central pode enviar um *request* para a captura de imagens. Após este pedido, as imagens são capturadas pela câmera imóvel durante 3 segundos. Na Figura 2.14 está apresentado o sistema descrito [72].

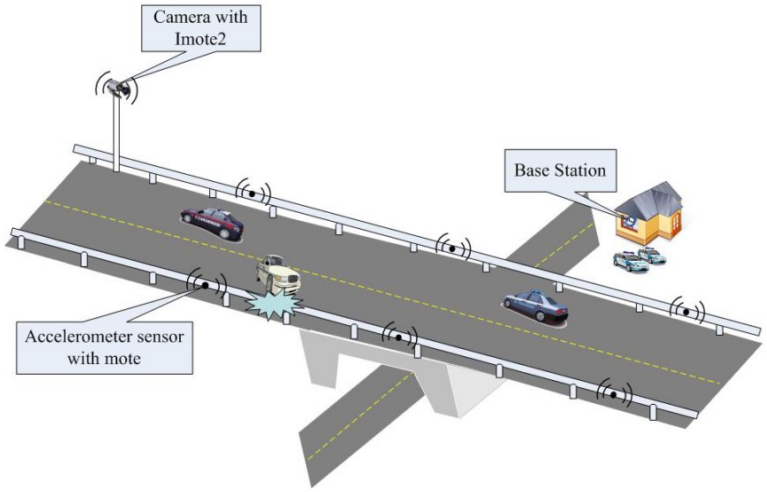


Figura 2.14: Sistema de monitorização de acidentes

2.2.2.2 Projeto *Barriera Attiva*

Barriera Ativa foi um projeto financiado pelo Ministro da Educação, Universidade e Investigação Italiano, e o seu principal objetivo visava a criação de um sistema a colocar nas guardas de segurança, para medição de diversas variáveis do ambiente em que se insere [73]. Uma das funções consistia na deteção de colisões dos veículos com a guarda de segurança. Foi desenvolvido um *Wireless Active Guardrail System* (WAGS) propondo uma rede de sensores sem fios que, para além de detetar colisões, também contava o número de veículos, calculava a sua velocidade e

detetava a proximidade dos mesmos [74].

O sistema proposto em [74] é dividido em 4 camadas. A primeira englobava os nós sensores, que mediam todas as variáveis supracitadas, e as enviava para o *gateway* (segunda camada). Ambos os dispositivos estão atracados a uma guarda de segurança, sendo que cada *gateway* formava uma sub-rede com vários nós sensores. A terceira camada era formada pelos concentradores, dispositivos que recebiam as informações dos *gateways* e as colocavam num servidor na Internet. Finalmente, na quarta camada, o supervisor acedia aos dados recolhidos através do servidor.

2.2.2.3 Projeto SustIMS

SustIMS é um projeto que se desenvolveu de 2012 a 2015 e que visava o desenvolvimento de uma plataforma de gestão sustentável para infraestruturas rodoviárias. Um dos objetivos consistia em gerir os principais elementos de uma infraestrutura incluindo obras de arte, pavimentos, taludes, muros, entre outros [6]. De forma a concretizar esse objetivo, foi desenvolvida uma WSN com o intuito de detetar colisões com as guardas de segurança das autoestradas.

A rede de sensores sem fios é composta pelos nós sensores, coordenador e gateway. Estes primeiros enviam dados para o coordenador, sendo assim reencaminhados para o gateway. Este recolhe a informação e envia-a para uma *cloud*, para posteriormente ser acedido pelos utilizadores via *browser* ou aplicação Android [5]. O acelerómetro, que irá medir a aceleração da vibração provocada pelo choque do veículo com a guarda de segurança, encontra-se grande parte do tempo no modo de baixo consumo energético e quando deteta uma aceleração acima de um determinado valor, despoleta uma interrupção no microcontrolador. Desta forma, é reduzida a energia consumida pelo nó sensor, dispositivo ao qual está ligado o acelerómetro.

A WSN deste projeto tem vindo a ser alvo de diversas revisões. A revisão de 2013 utilizava o *System-on-Chip* (SoC) CC2530 para os nós sensores e coordenador, desenvolveu uma aplicação para *Desktop*, bem como um *Website* para interação com a WSN [75]. A revisão de 2014 [68] utilizava o *System-on-Chip* (SoC) CC2530 com arquitetura 8051 de 8 *bits*. Em termos de *software*, utilizava a Z-Stack da *Texas Instruments*, uma solução totalmente compatível com ZigBee. Para além de detetar colisões, também estava preparado para suportar diversos sensores de medição do meio ambiente, como por exemplo, temperatura e humidade do ar, temperatura do solo, pressão atmosférica e luminosidade. Este sistema autodenomina-se de *HighWayMon* e a sua grande vantagem era a capacidade de *plug-and-play*. Era composto por diversos módulos

que se encaixavam formando uma pilha. Existiam módulos para o CC2530, circuitos de potência, sensores, acelerómetro e aquisição de energia. Realça-se o facto de este sistema possuir capacidade para adquirir energia através de um painel solar.

A reisão de 2018 [76] propôs uma alteração no *hardware*, mais concretamente, alterou para um microcontrolador com arquitetura ARM de 32 *bits*, o SoC CC2538. Para além disso, reduziu o suporte de alguns sensores, restando apenas os de temperatura e humidade do ar, e de luminosidade. Não possui capacidade de adquirir energia, mas provou ser mais eficiente energeticamente. Em termos de *software*, recorreu a uma versão mais recente da Z-Stack.

2.2.2.4 Comparação entre as várias WSNs

Na Tabela 2.4 encontra-se representada uma comparação entre as quatro redes de sensores sem fios acima mencionadas. Nesta tabela, pode-se comparar o acelerómetro e o método de leitura deste utilizados, a plataforma ou microcontrolador, o sistema operativo utilizado, e a existência da capacidade de obtenção de energia elétrica dos nós sensores.

Tabela 2.4: Comparação entre as três WSNs

	WSN para deteção de colisões [72]	***Barriera Attiva* [74]**	**SustIMS versão 2014 [68]**	**SustIMS versão 2018 [76]**
Acelerómetro	CXL10GP3	ADXL-326	LIS331DLH	LIS331DLH
Método de leitura do acelerómetro	Frequência fixa	—	Interrupção	Interrupção
Plataforma	IRIS mote	ATmega1281	CC2530	CC2538
Sistema operativo	TinyOS	TinyOS	Z-Stack	Z-Stack
Aquisição de energia	Não	Sim	Sim	Não

2.3 Sistema Operativo Contiki

Contiki é um sistema operativo *open-source* para utilização na *Internet of Things* (IoT) [77]. É bastante leve e portátil, suportando vários dispositivos de recursos limitados, desde microcontro-

ladores de 8 *bits* até plataformas ARM Cortex M3 de 32 *bits*. Escrito em linguagem C, é orientado a eventos com escalonamento cooperativo, mas também suporta *multi-threading*. Desde o seu lançamento, este sistema operativo tem vindo constantemente a ser atualizado até à sua mais recente versão 3.0 lançada em agosto de 2015 [78] .

Contiki foi um projeto que se iniciou em 2002 com Adam Dunkels, mas a primeira versão apenas foi lançada em 2003, sendo o primeiro sistema operativo a fornecer conectividade IP para redes de sensores [79]. Devido a isso, em 2008 foi criado um novo *standard Internet Engineering Task Force* (IETF) e fundada a aliança *Internet Protocol for Smart Objects* (IPSO), que promovia o uso do IP nas redes IoT. Nesse mesmo ano, foi-lhes atribuído o prémio de 30ª inovação mais importante do ano pela revista TIME. Em 2013, IPSO lança um desafio onde inovadores teriam de mostrar projetos que recorriam à tecnologia IP [80]. O primeiro prémio foi atribuído a um projeto que media a corrente que passava num *power cable* [81]. A informação sobre essa corrente era transmitida para um servidor na internet onde poderia ser acedida em qualquer lugar. Utilizava o sistema operativo Contiki, sendo que as grandes vantagens deste projeto, é a não utilização de baterias e, dada a sua forma de clipe, a sua colocação era fácil.

2.3.1 Arquitetura

No que se refere à arquitetura, existem discordâncias entre alguns autores. Uns defendem que segue uma arquitetura monolítica em que o próprio sistema e o conjunto de processos são compilados numa só imagem [15]. Outros autores seguem uma arquitetura modular, que possibilita a configuração dinâmica dos módulos [82]. Efetivamente existe um *loader* no Contiki onde é possível alterar os módulos que serão executados em tempo de execução, concluindo assim, que se trata de uma arquitetura modular.

2.3.2 *Photothreads*

Os processos no Contiki são *threads* com uma pilha reduzida desenhadas para sistemas com recursos limitados. A essas *threads* dá-se o nome de *photothreads*, sendo que a grande vantagem destas sobre as *threads* normais, é que não requerem uma *stack* separada. Desta forma, o *overhead* provocado pelas múltiplas alocações na pilha é menor.

Dado que a *stack* é a mesma para as várias *photothreads*, a mudança de contexto é apenas

composta por um deslocamento na pilha. Como as variáveis locais de cada processo não são guardadas, o conteúdo destas pode ser perdido, se não forem usadas com precaução. Neste cenário, os processos devem informar o escalonador quando pretendem terminar ou bloquear a sua execução, mas só o podem fazer dentro da própria função. Os seja, as *photothreads* podem chamar outras funções, mas não é permitido invocar o escalonador, ou bloquear o processo, sem ser no interior deste.

2.3.3 Escalonamento

Todas as tarefas neste sistema operativo são processos, e como tal, devem ser criadas e executadas. O Contiki apresenta um *kernel* orientado a eventos o que, de grosso modo, equivale a um *loop* infinito que responde a todos os eventos. Um processo é executado de duas formas. O primeiro é em modo cooperativo, em que um processo é iniciado e não é interrompido pelo sistema. Quando termina a sua execução informa o escalonador para este poder atribuir a execução a outro processo. A segunda forma é o modo preemptivo, em que no decorrer da execução de determinado processo, este é interrompido por uma interrupção externa ou por um temporizador síncrono.

Existem dois tipos de eventos suportados pelo Contiki, os eventos síncronos e os assíncronos [83]. Os eventos síncronos, por requerem períodos de tempo fixos, são atendidos imediatamente. Por outro lado, os eventos assíncronos são colocados numa fila, para posteriormente serem enviados para o seu processo destino. É de salientar que, nenhum evento poderá interromper outro, e apenas uma interrupção ou um *real timer* o pode fazer. O *kernel* proporciona um escalonador de eventos responsável por direcionar eventos para os seus respetivos processos. Em adição, o *kernel* também possui um mecanismo de *polling* utilizado pelos processos que monitorizam o *hardware*.

2.4 Resumo

No decorrer deste capitulo foram abordadas algumas tecnologias para comunicações de dados sem fios, nomeadamente o IEEE 802.15.4 e 6LoWPAN, utilizados nesta dissertação, o ZigBee, *standard* utilizado nas revisões anteriores do projeto, e Thread, um protocolo de rede que recorre ao 6LoWPAN. Posteriormente, referiu-se temas como sistemas ciber-físicos e redes de sensores sem fios, e discutiu-se algumas WSNs semelhantes à que foi implementada. Ambas as 4 redes apresentadas detetavam possíveis colisões.

Por fim, falou-se sobre o sistema operativo Contiki, que possibilita a utilização do protocolo IP nas comunicações entre dispositivos. O Contiki tem como principais vantagens ser *open source*, extremamente portátil e possibilita o uso de *multithreading*. Para além disso, o seu código é compacto, sendo um dos sistemas operativos mais leves.

Capítulo 3

Especificação do sistema

O presente capítulo pretende descrever todo o processo funcional do sistema implementado. Primeiramente serão apresentados os requisitos a que este projeto se submete. Posteriormente, é apresentada uma visão geral do sistema, e especificado o *hardware* e *software* utilizado.

3.1 Requisitos do Sistema

Os requisitos podem ser divididos em funcionais e não funcionais. Estes primeiros visam apresentar as funcionalidades e os serviços do sistema. Para além disso, também descreve como o sistema deve reagir a determinadas entradas, bem como, o que este não deve fazer. Por outro lado, os requisitos não funcionais definem as propriedades e restrições do sistema, como por exemplo, o desempenho ou o espaço ocupado. Muitas das vezes, este tipo de requisitos é mais crítico que os requisitos funcionais.

3.1.1 Requisitos Funcionais

Pode-se então referir que os requisitos funcionais são os seguintes:

- Detetar acelerações acima de um nível estipulado. O acelerómetro conectado ao nó sensor deve alertar este, de que ocorreu uma aceleração superior à pré-definida;

- Informar o coordenador de que detetou uma aceleração superior ao *threshold* definido. O nó sensor deve enviar uma mensagem ao coordenador informando-o do ocorrido.

- Enviar informações periodicamente para o coordenador, indicando que o nó sensor se encontra ativo. Recorrendo ao envio de mensagens periódicas será possível determinar se o nó sensor ainda se encontra em funcionamento;

- Guardar o valor RSSI do último *acknowledge* enviado pelo coordenador. O valor RSSI servirá como indicador do estado de comunicação da rede;

- Selecionar entre o modos *deep sleep* e normal. Os nós sensores devem consumir o mínimo possível de energia, e como tal, quando não estão a realizar nenhuma tarefa, deverão estar em *deep sleep*;

- O coordenador deve reencaminhar as mensagens provenientes do nó sensor para o *gateway*. É o coordenador que estabelece a comunicação entre os nós sensores e o *gateway*, para além de efetuar a gestão da rede interna;

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais deste sistema são os seguintes:

- O nó sensor deve ter baixo consumo energético e possuir um ano de autonomia;

- Deve possuir um tamanho reduzido, pois necessita de ser o menos intrusivo possível, tornando-o mais sensível a acelerações e facilmente colocável em qualquer ponto da guarda de segurança.

- Deve ser robusto de forma a resistir às intempéries. Dado que o sistema estará exposto ao tempo, este deverá ser resistente;

- Utilizar o Contiki como sistema operativo;

- Utilizar o SoC CC2538;

- O coordenador deve possuir interface UART para comunicação com o *gateway*;

- O coordenador deve ser alimentado com energia proveniente da rede.

3.2 Visão geral

O sistema desenvolvido para esta dissertação tem como objetivo implementar uma nova revisão da WSN do projeto SustIMS. Assim sendo, este sistema integra-se na terceira parte deste projeto, nos sistemas de monitorização baseados em redes de sensores sem fios.

A Figura 3.1 apresenta uma visão geral do sistema. Os nós sensores estão atracados às guardas de segurança das autoestradas. É da responsabilidade destes, detetar as perturbações causadas pela colisão dos veículos. Aquando do acontecimento, o nó sensor deve informar ao coordenador o ocorrido. Este, por sua vez, deve reencaminhar as mensagens provenientes do nó sensor para o *gateway*. Estando as informações no dispositivo que faz a interface entre as duas redes, o *gateway* deve enviá-las para uma *cloud*, através de uma ligação Wi-Fi ou Ethernet, que posteriormente serão armazenadas numa base de dados. A partir do momento que as informações estejam acessíveis *online*, o utilizador pode aceder a um local *Web* e visualizar os dados obtidos pela WSN.

Posto isto, é de referir que esta dissertação apenas se irá focar nos nós sensores e no coordenador. Estes dispositivos estão orientados numa topologia em estrela. Desta forma ter-se-á uma comunicação mais viável, e com menos perdas de informação entre os nós sensores e o coordenador.

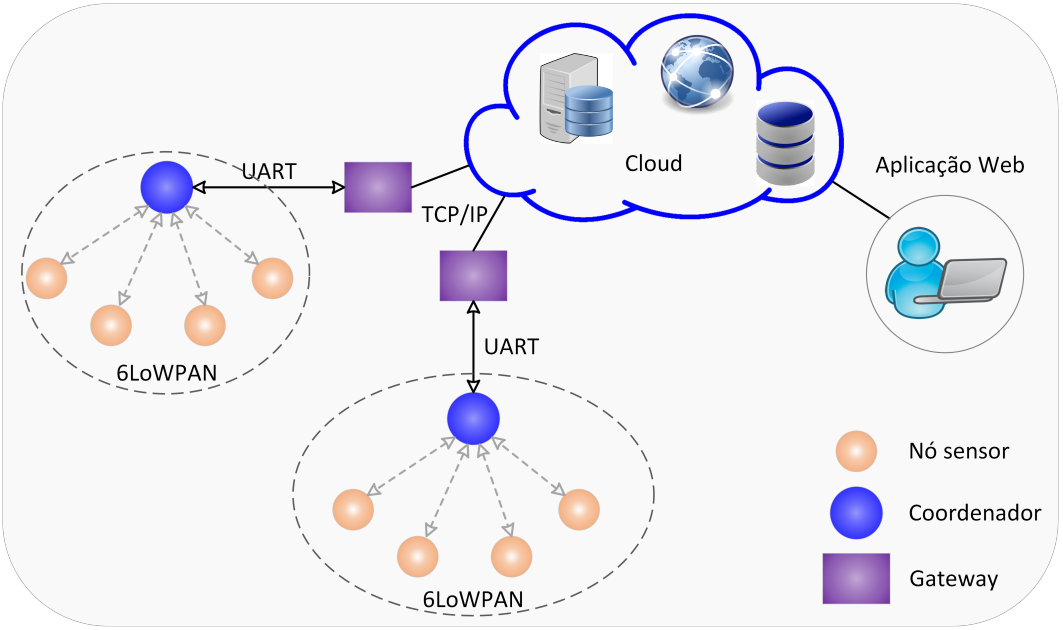


Figura 3.1: Visão geral do sistema

3.3 Arquitetura do sistema

A rede de sensores sem fios é composta pelos nós sensores e pelo coordenador. Na Figura 3.2 encontra-se um diagrama de blocos, no qual estão apresentados estes dois tipos de dispositivos, bem como as suas conexões.

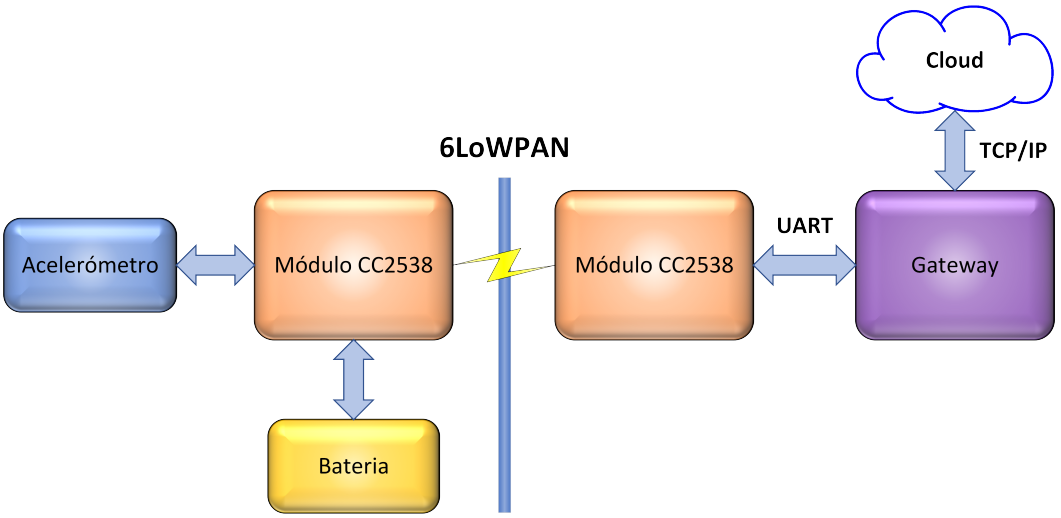


Figura 3.2: Diagrama do blocos dos sistema

Cada nó sensor está equipado com um acelerómetro e alimentado por uma bateria. Implementa um modelo de gestão energética, de modo a que não haja desperdício de energia, e desta forma, prolongar a vida útil da bateria. A comunicação entre os nós sensores e o coordenador é realizada via *wireless*, com recurso ao protocolo IEEE 802.15.4.

É de salientar que, devido à topologia em estrela, não há necessidade de *routers*. Dado que os dados são enviados diretamente para o coordenador, o reencaminhamento de mensagens dentro da WSN não é preciso.

Na Figura 3.3 está apresentado um caso de uso relativo ao sistema. A colisão com as guardas de segurança irá provocar uma vibração nesta, que será detetada pelo acelerómetro. Após este acontecimento, o nó sensor deve ler os dados recolhidos pelo sensor de aceleração, e enviá-los para o coordenador. Por sua vez, este deve converter a mensagem recebida num formato *JavaScript Object Notation* (JSON) e reencaminhá-los, por porta série, para o *gateway*.

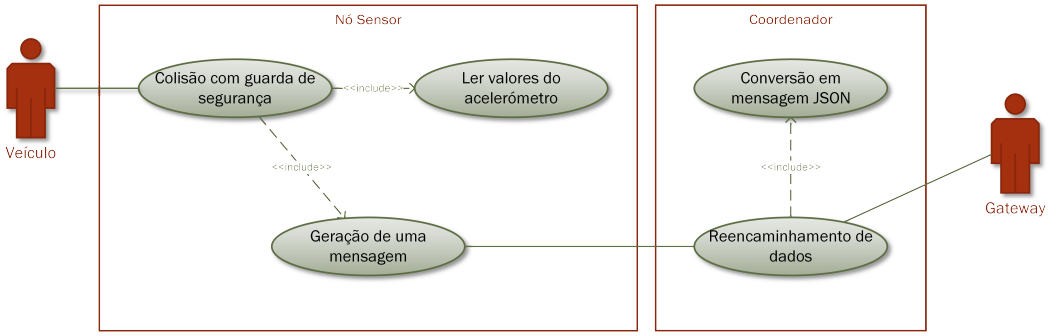


Figura 3.3: Caso de uso do sistema

3.4 Especificação do Hardware

Nesta secção são especificados os componentes *hardware* utilizados, nomeadamente, o microcontrolador, o acelerómetro e a bateria. É de salientar que apenas foram desenvolvidos o nó sensor e o coordenador. Apesar desta dissertação ser um *refactoring*, as principais alterações regem-se a nível de *software*.

Na revisão anterior do projeto [76], para além do microcontrolador e do acelerómetro, eram utilizados sensores para medir temperatura, humidade relativa e luminosidade. Estes sensores recorriam a uma interface *Inter-Integrated Circuit* (I^2C) para se comunicarem com o microcontrolador. Como o objetivo desta dissertação é apenas detetar colisões, foram retirados todos estes sensores, restando apenas o acelerómetro. Também é de referir que, todos os componentes *hardware* já se encontravam predefinidos desde o início deste projeto. O microcontrolador já se encontrava selecionado na revisão anterior, e os restantes elementos foram escolhidos com base em estudos anteriores, realizados no laboratório *Embedded System Research Group* (ESRG) da Universidade do Minho.

3.4.1 *System-on-Chip* (SoC)

O *System-on-Chip* (SoC) utilizado é o CC2538 da *Texas Instruments* (TI), tal como ilustra a Figura 3.4. É considerado SoC por possuir capacidade de processamento, aliada a um *transceiver* RF num só *chip*. Este *transceiver* RF é compatível com a norma IEEE 802.15.4, funciona na gama dos 2.4 GHz e pode atingir taxas de transferências de 250 kbps.

As aplicações para este SoC são diversas. Pode ser aplicado na *Internet of Things* (IoT), em sistemas de luminosidade inteligentes, entre outras aplicações. Nesta dissertação será aplicado

Figura 3.4: *System-on-Chip* CC2538 da *Texas Instruments* no *package* QFN56

numa rede de sensores sem fios, e dado o seu tamanho reduzido (8mm * 8mm QFN56 *package*), a sua integração num sistema com limitações de espaço é facilitada.

O SoC CC2538 é adequado para sistemas de baixo consumo energético, e como tal, permite diversos modos de funcionamento. A tabela 3.1 apresenta os consumos energéticos de cada modo, quando alimentado a 3V com a frequência de relógio do sistema a 8 MHz.

Tabela 3.1: Consumos dos vários modos do CC2538

Características de consumo CC2538	
Modo Ativo Rx	20 mA
Modo Ativo Tx	24 mA
Power Mode 1 (4 μs *wake-up*)	0,6 mA
Power Mode 2 (*Sleep Timer*)	1,3 μA
Power Mode 3 (Interrupções Externas)	0,4 μA

O CC2538 é alimentado por uma tensão que pode variar de 2V a 3,6V, e apresenta as seguintes características [84]:

- Microcontrolador de 32 bits ARM Cortex-M3;
- Frequência de relógio máxima de 32 MHz;
- Memória *Flash* de 128, 256 ou 512 kB;
- Memória *Random Access Memory* (RAM) de 32 kB;
- Suporta atualizações *Over-the-Air*;

- *Debugging* por cJTAG e JTAG;
- Interfaces disponíveis: 2 * SPI, 2 * UART, I^2C, USB;
- ADC de 12 bits com 8 canais;
- 32 *General Purpose Input/Output Pins* (GPIO);
- 4 *timers* de propósito geral, 1 *sleep timer* e 1 *watchdog timer*;
- *Transceiver* RF de 2,4 GHz IEEE 802.15.4.

O módulo utilizado, incorpora também o *range extender* CC2592, que irá permitir obter maiores distâncias de comunicação. Na Figura 3.5 está apresentado o referido módulo. Este *range extender* é *low power*, e adequado para aplicações *wireless* 2,4 GHz. A existência de um *Power Amplifier* (PA) e de um *Low Noise Amplifier* (LNA) permitem aumentar a potência de saída e a sensibilidade da receção de dados, respetivamente. Este módulo tem como referência o código ZB3219PAI.

O CC2592 foi criado para ser utilizado com os todos os *transceivers* CC25XX, transmissores e SoCs da TI. O seu tamanho é reduzido, 4mm * 4mm no *package* QFN16, e apresenta as seguintes características [85].

- Frequência de operação limitada de 2400 a 2483 MHz;
- Com o uso deste *range extender*, em conjunto com o SoC, a potência de saída acresce até 22 dBm e melhora a sensibilidade em mais 3 dBm.
- Consumo de 100 nA quando desligado;
- Consumo de 155 mA quando em transmissão para +22 dBm;
- Consumo em recepção: 4 mA em alto ganho e 1,9 mA em baixo ganho;
- Operação de 2V a 3,7V.

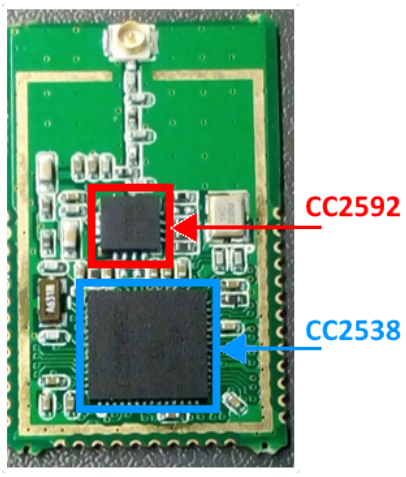


Figura 3.5: Módulo com o SoC CC2538 e o *range extender* CC2592

3.4.2 Acelerómetro

O acelerómetro tem como principal função, detetar e medir a aceleração provocada pela colisão. A escolha deste componente baseou-se no estudo efetuado pelo laboratório ESRG, sendo também o mesmo utilizado na revisão anterior do projeto. O acelerómetro em questão é o LIS331DLH da STMicroeletronics. Na tabela 3.2 é possível visualizar os consumos do acelerómetro. Como se pode verificar, este dispositivo apenas consome 10 μA no modo *low power*.

Tabela 3.2: Consumos do acelerómetro LIS331DLH

Características de consumo do acelerómetro	
Modo Normal	250 μA
Modo *low power*	10 μA
Desligado	1 μA

Alimentado por tensões entre 2,16V e 3,6V, este acelerómetro permite configurar um valor de *threshold* da aceleração, no qual, quando for detetada uma aceleração acima deste valor, o dispositivo alerta o microcontrolador através de uma interrupção. O seu tamanho reduzido de 3mm * 3mm permite a integração em sistemas com pouco espaço. A Figura 3.6 mostra o acelerómetro em questão.

O LIS331DLH apresenta as seguintes características [86]:

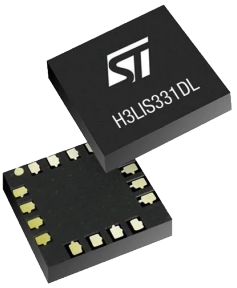


Figura 3.6: Acelerómetro LIS331DLH no *package* LGA 16

- *Full scales* dinamicamente selecionáveis: $\pm$2g, $\pm$4g, $\pm$8g;
- Frequência de medição de 0,5Hz a 1kHz;
- Interface digital de saída SPI/I^2C;
- 3 eixos de medição;
- 2 geradores de interrupção programáveis independentes;
- Frequência de relógio do SPI até 10 MHz.

A interface utilizada para comunicar com o acelerómetro foi o SPI. Utiliza as 4 ligações requeridas por este protocolo, nomeadamente *Clock* (CLK), *Slave Select* (SS), *Master Input Slave Output* (MISO) e *Master Output Slave Input* (MOSI), conectados ao pinos PA2, PA3, PA4 e PA5, respetivamente. Para além destes, são também ligados os 2 pinos correspondentes aos dois geradores de interrupção do acelerómetro, ao PD0 e PD1.

3.4.3 Bateria

A bateria escolhida para o nó sensor é a LS17500 da SAFT Batteries. O seu nome advém da sua composição e das suas dimensões. É composta por *Lithium Thionyl Chloride* (Li-$SOCl_2$), apresenta um diâmetro de 16,85 mm (aproximadamente 17 mm) e um comprimento de 50,3 mm. Este tamanho é o mesmo das baterias tipo A. Na Figura 3.7 está apresentada uma imagem da bateria. Para além disso, apresenta as seguintes características:

- Tensão Nominal: 3,6 V;
- Capacidade: 3400 mAh;

- Temperaturas de funcionamento: -60°C a +85°C;
- Peso: 21,9 g;

Figura 3.7: Bateria LS17500

A sua grande vantagem é a capacidade de fornecer picos de corrente sem que a tensão desça drasticamente. Por outro lado, a tensão desta bateria é praticamente constante ao longo da sua vida útil e não é recarregável.

3.5 *System Stack*

O sistema pode ser dividido em 3 camadas essenciais: *hardware*, *software* e aplicação, tal como é mostrado na Figura 3.8. Na camada mais baixa, a de *hardware*, localiza-se o SoC utilizado, bem como os sensores conectados, que neste caso é apenas o acelerómetro. O SoC CC2538 tem incorporado o rádio, ou *transceiver*, e o *range extender*.

O *software* encontra-se dividido em 2 camadas: a do sistema operativo, que neste caso é o Contiki, e a da aplicação. Nesta última camada, foram implementadas técnicas de gestão do consumo energético, de leitura dos valores do acelerómetro, e de comunicação entre os vários dispositivos.

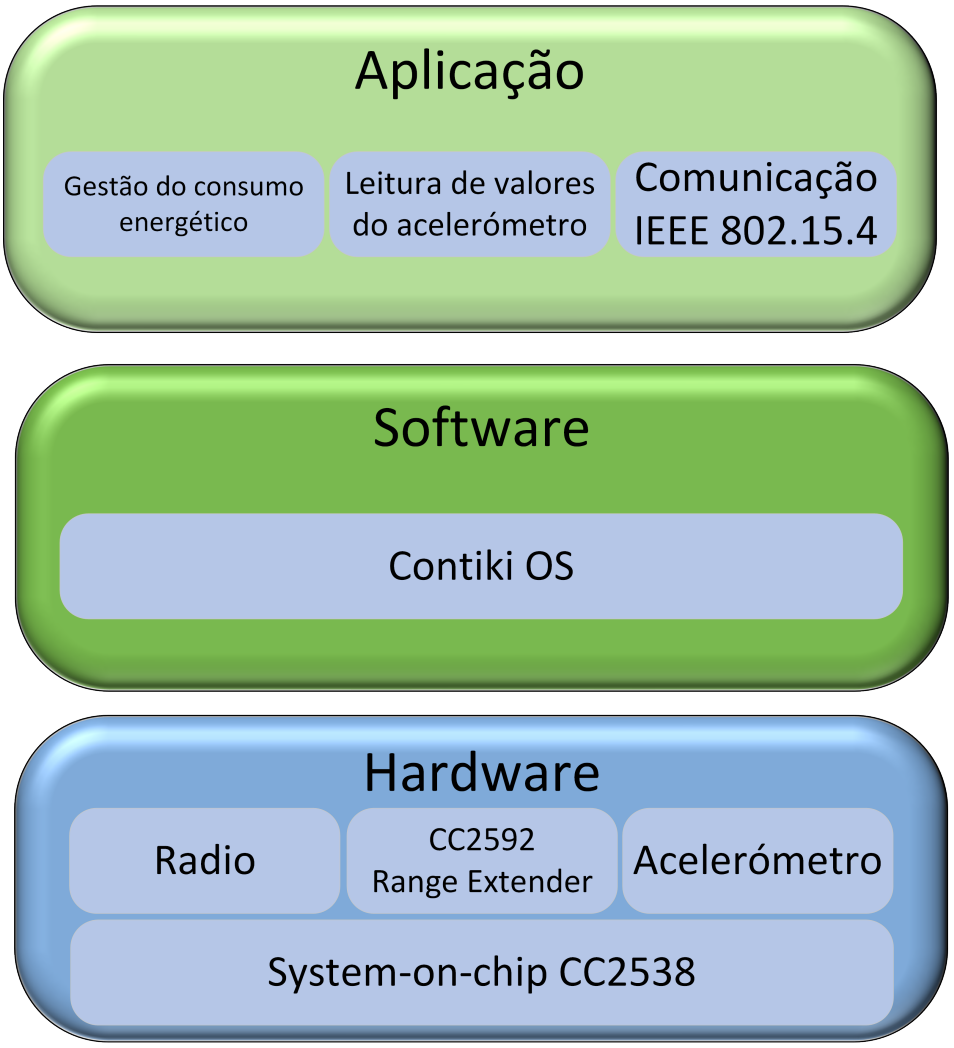


Figura 3.8: *System stack*

3.6 Especificação do Software

Nesta secção serão abordados alguns detalhes sobre o funcionamento do sistema. Inicialmente será explicado algumas notações necessárias, como é o caso do JSON e posteriormente, as funções de ambos os dispositivos.

3.6.1 *JavaScript Object Notation*

Criado por Douglas Crockford, JSON é um formato de troca de dados leve e de fácil interpretação, tanto para pessoas, como para máquinas. Apesar desta notação ser independente da linguagem, recorre a muitas convenções de linguagens como C, C++, Javascript, entre outras [87].

A base deste formato assenta em duas estruturas: num agrupamento de um par nome e valor, e numa lista ordenada de valores. Um texto JSON é uma sequência de *tokens*, dos quais seis estruturais e três literais. As chavetas ({ }) são para iniciar e encerrar objetos, os parênteses retos ([ ]) possuem a mesma função, mas para os *arrays*, os dois pontos (:) para separar entre o nome e o valor, e finalmente, a vírgula (,) para separar valores. Os *tokens* literais são usados em valores, e correspondem às palavras *true*, *false* e *null* [88].

3.6.2 Aplicação do nó sensor

3.6.2.1 Conexão ao coordenador

A conexão dos nós sensores ao coordenador é realizada através de várias etapas. A Figura 3.9 apresenta todas as fases para que um nó sensor se conecte a uma rede. Na inicialização do nó sensor é configurado o acelerómetro e inicializada uma conexão UDP. O segundo estado é da responsabilidade da camada de rede, em que dois protocolos, *Routing Protocol for low power and Lossy networks* (RPL) e *Neighbor Discovery* (ND), tentam obter uma resposta do coordenador. Após a receção desta, considera-se que foi encontrado o responsável da rede. No terceiro estado, a camada da aplicação envia mensagens de associação. No quarto estado aguarda-se pela mensagem de resposta à associação. Quando esta for recebida é enviada uma mensagem de configuração do nó sensor. Os detalhes destes dois tipos de mensagens estão no anexo A. Assim, no quinto e último estado, o dispositivo fica associado à rede.

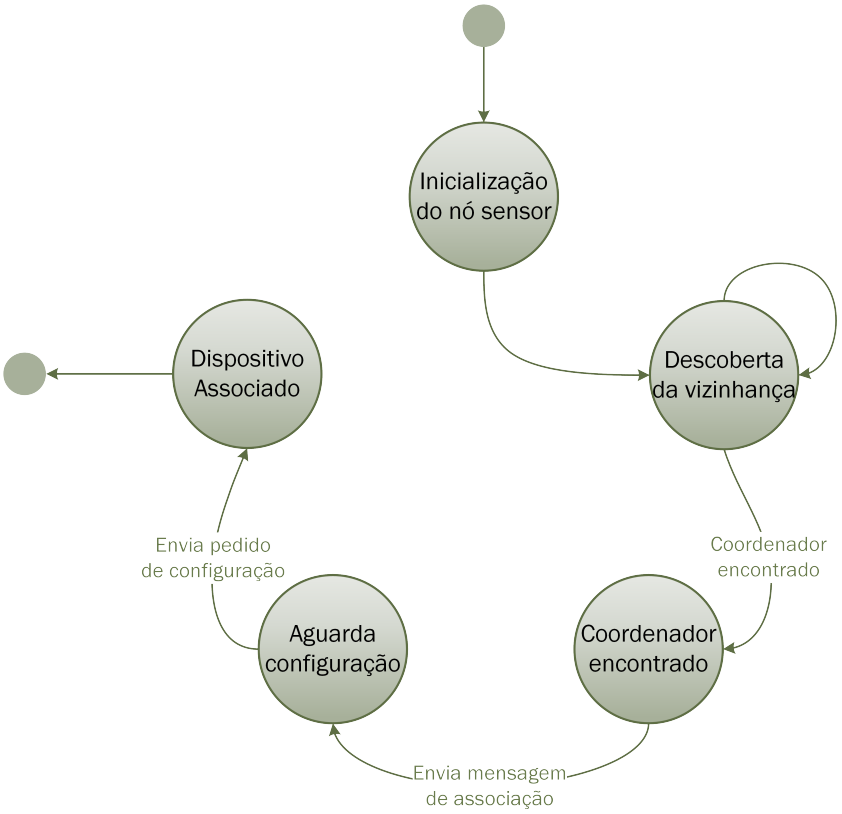


Figura 3.9: Diagrama de estado da conexão do nó sensor ao coordenador

3.6.2.2 Funcionamento geral

O nó sensor passa a maior parte do tempo no modo de *deep sleep*. Apenas existem duas formas de este acordar: por interrupção externa provocada pelo acelerómetro, ou pelo *timeout* do *sleep timer*. Na Figura 3.10 está apresentado um diagrama de estado, representativo das ações responsáveis por acordar o microcontrolador. No estado *Sleep*, o nó sensor está no modo de baixo consumo energético. Quando ocorre a colisão de um veículo com as guardas de segurança, o acelerómetro deteta essa aceleração, despoleta uma interrupção no microcontrolador e entra no estado de enviar uma mensagem de colisão.

Por outro lado, há um temporizador programado para acordar o dispositivo em cada intervalo de tempo. Quando ocorre o *timeout* deste temporizador, o nó sensor entra no estado de enviar uma mensagem *keep alive*, e comunica ao coordenador da rede. O objetivo deste mecanismo é informar o coordenador que o nó sensor continua em funcionamento, e caso ocorra alguma colisão, o responsável da rede é informado.

Para entender o funcionamento do nó sensor são apresentados alguns diagramas de sequên-

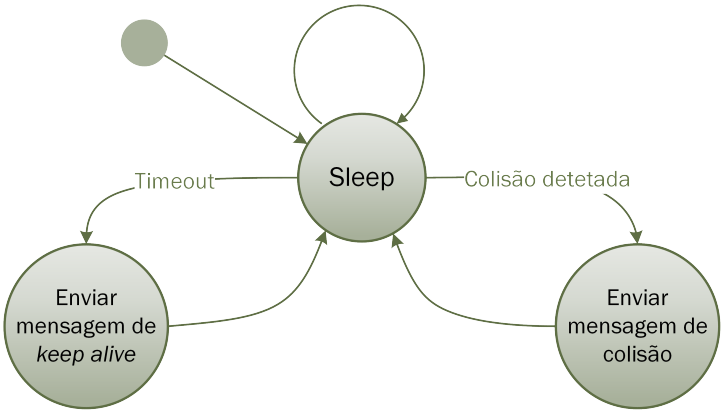


Figura 3.10: Diagrama de estado das possíveis ações depois do *wake-up* do nó sensor

cia das ações que ocorrem depois do microcontrolador acordar. Na Figura 3.11 está apresentado um diagrama de sequência alusivo à colisão com as guardas de segurança. Como é possível verificar, após o embate de um veículo, o acelerómetro deteta essa mesma colisão. Seguidamente, é despoletada uma interrupção que fará acordar o dispositivo, procedendo-se ao cálculo da aceleração. Por sua vez, esta informação é enviada para o coordenador. A mensagem é convertida no formato JSON, um protocolo de serialização de dados, e reencaminhada, por porta série, para o *gateway*. Como o *gateway* possui ligação à Internet, os dados recebidos são colocados numa *cloud*. Após algum tempo, na ordem dos segundos, é enviada uma 2ª mensagem de colisão. Esta contém o valor da maior aceleração registada no intervalo de tempo entre as duas mensagens. No capítulo seguinte será explicado o processo de geração da 2ª mensagem.

A Figura 3.12 apresenta um diagrama de sequência que ilustra as ações que decorrem para a geração da mensagem *keep alive*. O *sleep timer* do microcontrolador está configurado para acordar o dispositivo, num tempo predefinido pelo coordenador. Quando ocorre o *timeout* deste temporizador, e se estiver habilitada esta opção, o nó sensor adquire o valor RSSI do último *acknowledge* recebido, e envia uma mensagem ao coordenador. Este por sua vez, converte-a no formato JSON e reencaminha-a para o *gateway*.

3.6.2.3 Módulo do acelerómetro

O acelerómetro tem como principal função, detetar e medir as acelerações provocadas por uma colisão. Este módulo tem vindo a ser desenvolvido ao longo de vários projetos realizados no laboratório ESRG. Contém diversas funções que implementam o interface com o acelerómetro, como por exemplo, para ler e escrever em registos e de calcular a força da aceleração. Para

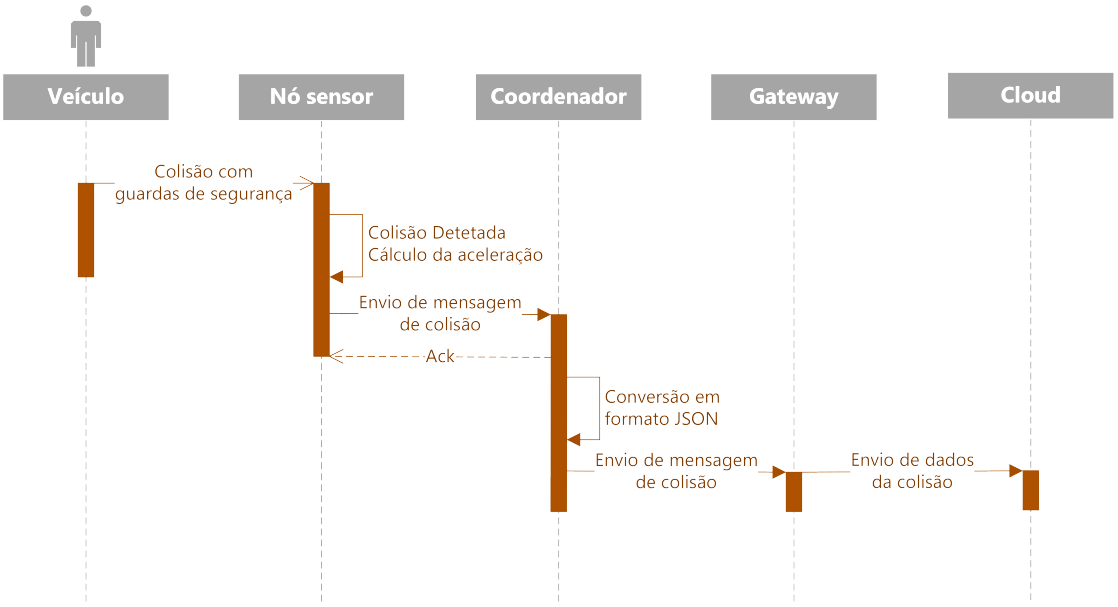


Figura 3.11: Diagrama de sequência das ações provocadas pela colisão de um veículo

além disso, possuía rotinas de inicialização do inclinómetro e de cálculo de ângulos. Estas, por sua vez, foram retiradas pois este sistema apenas se foca na aceleração.

O acelerómetro suporta 3 gamas de aceleração dinâmicas, nomeadamente, de $\pm$2g, $\pm$4g, $\pm$8g. O valor da escala referido na Equação 3.1 corresponde exatamente aos valores 2, 4 e 8, respetivamente. A resolução é de 12 bits e está configurado para medir acelerações nos três eixos. O acelerómetro está configurado para despoletar uma interrupção no microcontrolador, quando detetar acelerações superiores a 2g na gama de $\pm$8g. O LIS331DLH possui diversas frequências de leitura dentro do modo *low power*, variando entre 0,5 Hz e 10 Hz, sendo que foi selecionada a frequência de 5 Hz. No modo normal, as frequências variam de 50 Hz a 1 kHz. Quando este deteta uma aceleração de valor superior ao *treshold* é configurado no modo normal com uma frequência de leitura de 1 kHz. Após o envio da 2ª mensagem de colisão, o acelerómetro é novamente configurado no modo *low power*.

A Equação 3.1 apresenta o cálculo do valor da aceleração em mg. O valor lido de cada eixo é multiplicado pelo dobro da escala, e dividido pela resolução, ou seja, 2^{12}. Para converter o resultado de g para mg, multiplica-se por 1000.

$$Aceleração(mg) = Acc[N] * \left(\frac{Escala * 2}{2^{12}}\right) * 1000 \tag{3.1}$$

Equação 3.1: Equação de cálculo da força da aceleração em mg

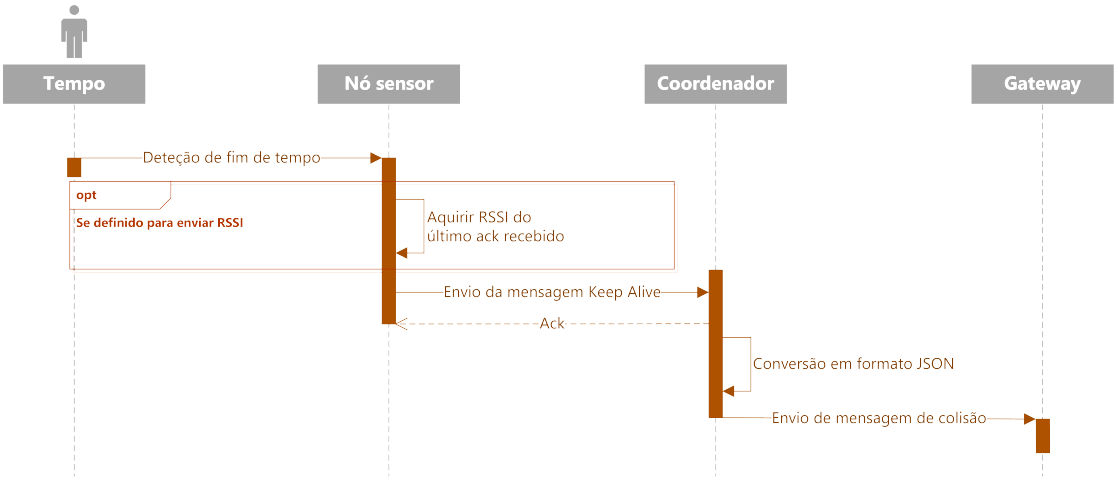


Figura 3.12: Diagrama de sequência das ações conseguintes do *timeout* do *timer* de *keep alive*

Na Equação 3.2 está apresentado o cálculo do módulo da aceleração. Este valor irá servir de comparação para outras acelerações, e assim conseguir determinar qual a vibração que obteve maior módulo.

$$|Aceleração| = \sqrt{AccX^2 + AccY^2 + AccZ^2} \tag{3.2}$$

Equação 3.2: Equação de cálculo do módulo da aceleração

3.6.3 Aplicação do coordenador

O coordenador é o responsável da rede. As suas funções regem-se a criar e gerir a rede, e reencaminhar informações para o *gateway*. Na Figura 3.13 pode-se observar um diagrama de estado representativo das funcionalidades do coordenador.

Na inicialização do coordenador é inicializada uma conexão UDP, bem como, configurada a UART. A criação da rede consiste na seleção de dois parâmetros essenciais, o canal de frequência a usar e o identificador da rede (PAN ID). Estes parâmetros são definidos previamente durante a compilação do código. Na inicialização do coordenador é iniciada uma conexão UDP com duas portas pré-definidas, uma para receber dados e outra para os enviar, e também configurada a *Universal Asynchronous Receiver / Transmitter* (UART). Depois de concluído este passo, o coordenador está preparado para receber novas conexões de nós sensores, ficando assim, no estado *Idle*.

A associação de novos dispositivos inicia-se com a receção de uma mensagem de associação.

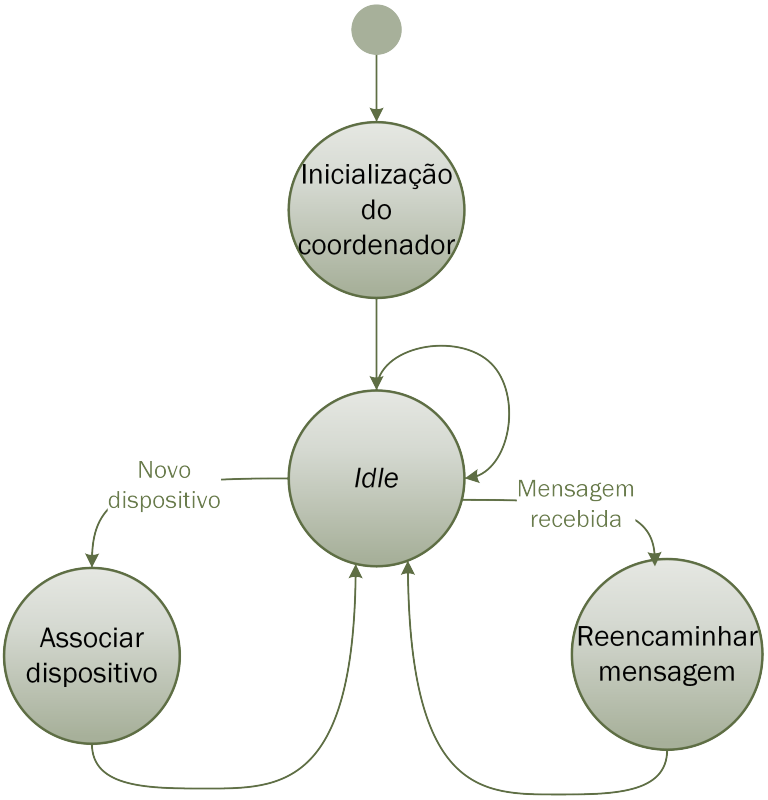


Figura 3.13: Diagrama de estado do funcionamento do coordenador

Desta forma, o coordenador entra no estado de associar um dispositivo, integrando-o na rede. Por outro lado, se for recebida uma mensagem de *keep alive* ou de colisão, o coordenador entra no estado de reencaminhar a mensagem, enviado-a para o *gateway*.

A funcionalidade de reencaminhamento de dados utiliza a porta série. O *gateway* apenas interpreta JSON, e como tal, a mensagem a enviar do coordenador para o *gateway* deve estar sempre nesse formato. A mensagem está dividida em dois grupos: as informações alusivas à rede e as relativas à aplicação. No que se refere à rede, as variáveis são as seguintes:

- WPAN ID;
- GATEWAY MAC;
- LQI;
- RSSI;
- SECURITY;
- SAMPLING RATE;

- SEQ NUMBER.

No que se refere às variáveis da aplicação podem variar consoante o tipo da mensagem. É de referir que duas dessas variáveis são fixas, nomeadamente:

- DEVICE MAC;
- DEVICE TYPE.

  Nas restantes mensagens, o conteúdo é variável, sendo o seguinte:

- Para mensagens de associação de novos dispositivos:
  - CONNECTION.
- Para mensagens *keep alive*:
  - RSSI.
- Para mensagens de colisão:
  - ACCELERATION;
  - TOTAL;
  - INDEX.

3.7 Resumo

Ao longo do capítulo foram especificados os requisitos do projeto, bem como uma apresentação da visão geral e funcionamento do sistema. Posteriormente, foi especificado o *hardware* utilizado e mostrada a *system stack*. No final foi especificado o *software* para o nó sensor e coordenador.

Capítulo 4

Desenvolvimento do sistema

Este capítulo contém o *software* e *hardware* desenvolvido. Primeiramente é feita uma referência ao *hardware* existente e ao PCB assemblado. Posteriormente, é explicado o *software* implementado. Para além disso, é mostrado o formato da mensagem utilizado na comunicação entre os nós sensores e o coordenador.

4.1 Hardware desenvolvido

4.1.1 Alterações efetuadas

Na revisão anterior do projeto [76], existiam sensores de luminosidade, temperatura e humidade. Estes comunicavam por I^2C, mas visto que foram retirados, a ligação a esta interface deixou de ser necessária. Para controlo da alimentação dos sensores, havia um mecanismo de corte de energia, realizado por um MOSFET, que também foi retirado. Por fim, existia um interruptor que, quando pressionado, enviava mensagens de *keep alive*. Como este tipo de mensagens eram apenas enviadas aquando do *timeout* do temporizador, este interruptor também foi eliminado. Salienta-se o facto de que nada foi adicionado ao *hardware*.

Relativamente à parte de potência, o *Low-Dropout Regulator* (LDO) TPS783 foi mantido, e em conjunto com a bateria, fornece uma tensão de 3 volts ao circuito. Visto que havia espaço na placa, para testar a alimentação do módulo foi mantido o interruptor de teste.

Na Figura 4.1 está apresentado o bloco de regulação da tensão de alimentação. Se pilha for colocada com a polaridade invertida, o diodo em série com a bateria não permite a passagem

de corrente, e consequentemente, não danifica o circuito.

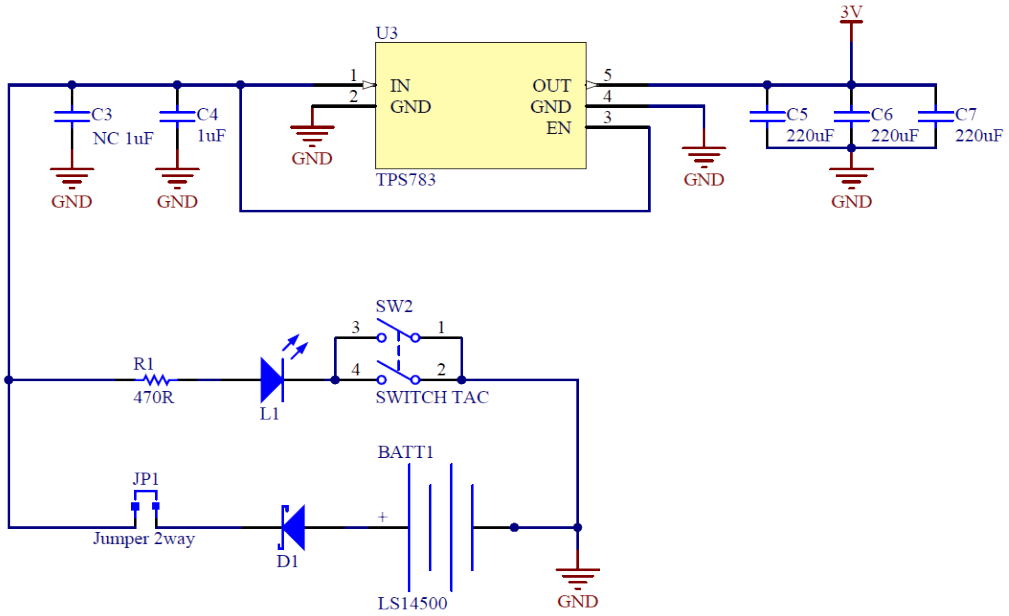


Figura 4.1: Esquemático da alimentação do circuito

A Figura 4.2 mostra as ligações do módulo CC2538/CC2592 às interfaces exteriores. Do lado esquerdo, para além da alimentação, os pinos TMS (*Test Mode Select*) e TCK (*Test Clock*) são sinais *Joint Test Action Group* (JTAG) utilizados para a programação do CC2538. Na parte inferior, ligados aos pinos PA0 e PA1 estão o Tx e Rx da UART. É de referir que, ao contrário do coordenador, no nó sensor estes pinos não são utilizados porque a porta série não é necessária. O pino de *reset* está conectado a um interruptor de pressão. Por fim, o SoC tem duas instâncias SPI. O acelerómetro utiliza a instância zero, e conecta-se ao módulo em 4 pinos, de PA2 a PA5. Possui também duas interrupções conectadas aos pinos PD0 e PD1, provenientes do acelerómetro. O restante esquemático pode ser visto no anexo B.

4.1.2 PCB

Após realizado o esquemático, foi desenvolvido um *Printed Circuit Board* (PCB) do nó sensor. Este *design* teve como foco principal, o modelo da caixa onde iria ser inserido. Na Figura 4.3 é possível visualizar o modelo 3D deste PCB. Devido à não utilização de alguns pinos do módulo CC2538 + CC2592, estes foram deixados desconectados. É de referir que a bateria corresponde ao modelo LS17500.

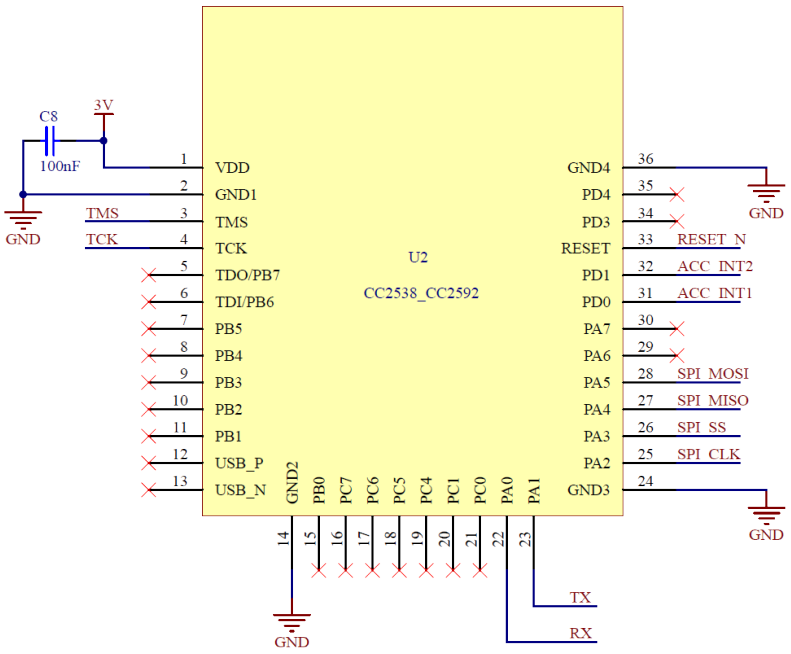


Figura 4.2: Esquemático do módulo CC2538/CC2592

Na Figura 4.4 estão imagens da caixa utilizada e do PCB assemblado. A caixa é uma Hammond com IP66, cujo formato interno é o apresentado na imagem. A fixação do PCB à caixa é efetuada por meio de um parafuso no centro da placa.

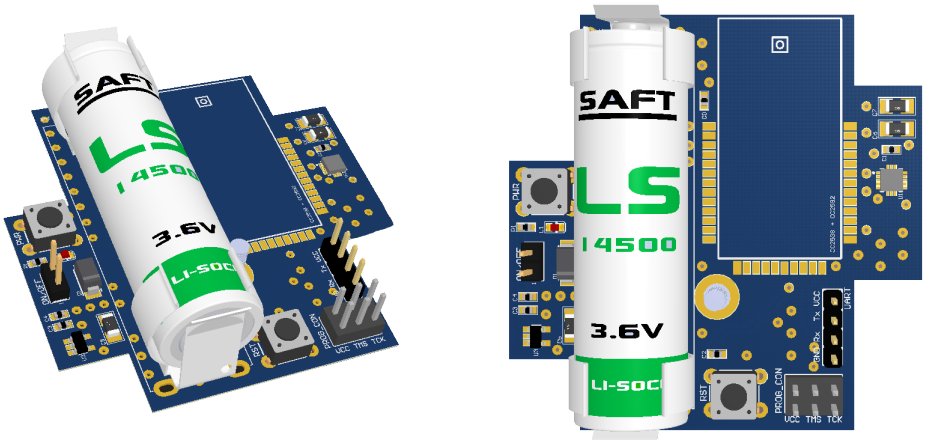


Figura 4.3: Visualização 3D do PCB

O *hardware* requerido pelo coordenador é apenas o módulo CC2538 + CC2592. Para este dispositivo foi utilizada a mesma placa, mas ignorado os restantes componentes. Em termos de alimentação foi ligada uma fonte de tensão em substituição à bateria. Refere-se também, que

ambos os dispositivos encontravam-se sempre conectados a uma antena.

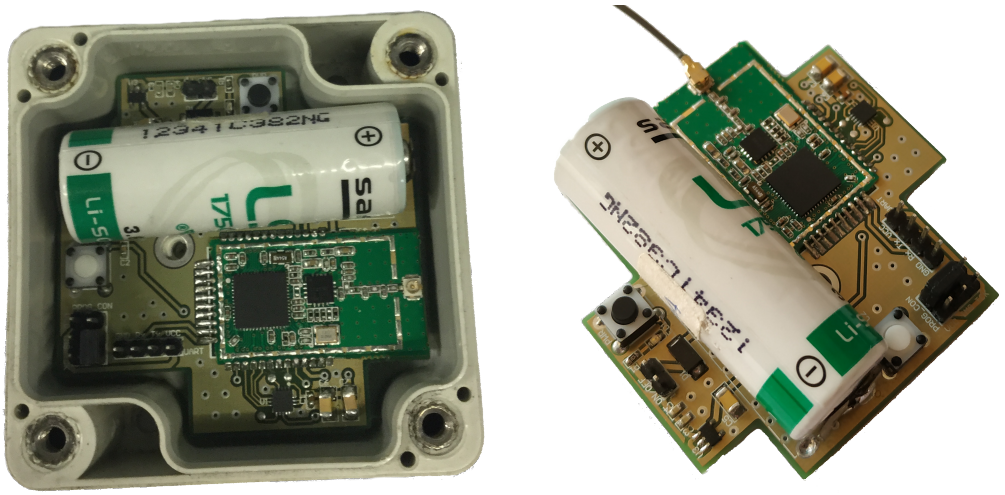


Figura 4.4: Visualização da caixa e do PCB assemblado

4.2 *Code Composer Studio*

O *Integrated Development Environment* (IDE) utilizado para o desenvolvimento do sistema foi o *Code Composer Studio* (CCS). Suporta a programação de microcontroladores da TI entre outros processadores embebidos. Inclui um compilador otimizado de C/C++, editor de código, ambiente de construção de projetos, *debugger*, entre outras funcionalidades. Baseado na *framework* Eclipse, e com adição de algumas capacidades de *debug* avançadas, o CCS resulta num ambiente de desenvolvimento completo, adequado para desenvolvedores de sistemas embebidos [89].

O codificador embutido conecta as ferramentas MATLAB, Simulink e Simulink Coder com o IDE em discussão. Desta forma, é possível desenvolver algoritmos, analisar, simular e gerar código. Os utilizadores podem ser desenvolvedores de algoritmos ou de *software*, *designers* de sistemas ou engenheiros.

4.3 Software desenvolvido

4.3.1 Nó sensor

O nó sensor, ou *end device*, é o dispositivo que ficará atracado às guardas de segurança das autoestradas. Para facilitar a interpretação do *software* desenvolvido, este é explicado através de fluxogramas. Assim, de forma a entender o funcionamento deste dispositivo, a explicação do código desenvolvido será dividida nas seguintes partes:

1. **Módulo *Radio Duty Cycling* (RDC)**: O sistema operativo Contiki está dividido em várias camadas. Uma delas é a RDC, cuja função consiste em gerir a ativação e desativação do rádio.

2. **Associação ao coordenador**: A conexão do nó sensor é realizada por duas camadas. A de rede, cuja responsabilidade é do sistema operativo, e pela camada da aplicação.

3. **Envio das mensagens de *keep alive***: O nó sensor acorda de forma síncrona para enviar este tipo de mensagens.

4. **Módulo do acelerómetro**: Quando é detetada uma aceleração acima da estipulada, ocorrem uma sequência de ações que serão descritas nesta parte.

4.3.1.1 Módulo *Radio Duty Cycling* (RDC)

A pilha de rede do sistema operativo Contiki está organizada em 5 camadas. A primeira é a do rádio, cuja função consiste em interagir diretamente com o *transceiver* do SoC CC2538. A segunda camada é a *Radio Duty Cycling* (RDC) que trata de ligar e desligar o rádio, de forma a obter um menor consumo energético final. A versão 3.0 do Contiki possui vários protocolos RDC, como por exemplo, o *6lowMAC*, cuja poupança de energia é nula, ou o ContikiMAC, que mantém o rádio desligado na maior parte do tempo. Para esta dissertação, foi desenvolvido um novo protocolo RDC.

A camada *Medium Access Control* (MAC) é responsável pelo endereçamento e retransmissão do pacotes. O protocolo utilizado nesta camada é o *Carrier Sense Multiple Access with Collision Avoidance* (CSMA-CA). Em termos de segurança, o Contiki fornece o protocolo *Link Layer Security*

(LLSEC), mas, para esta dissertação, não foi utilizado qualquer tipo de segurança. Esta, por sua vez, localiza-se a seguir à camada MAC. A Figura 4.5 apresenta a pilha de rede do Contiki. As camadas, em que o seu conteúdo foi alterado/acrescentado, estão assinaladas a cor verde.

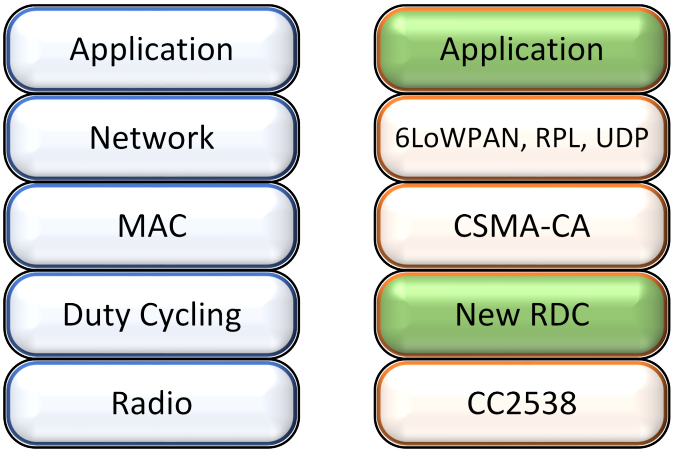


Figura 4.5: Pilha de rede utilizada no nó sensor

A camada de rede está internamente dividida em 3 partes: adaptação, *routing* e transporte. O 6LoWPAN faz a compressão e descompressão dos pacotes IPv6 trocados entre as camadas superiores e inferiores, atuando como uma adaptação entre estas. O *routing* é responsável por definir o caminho que as mensagens devem seguir, até chegarem ao coordenador. O protocolo utilizado é o *Routing Protocol for low power and Lossy networks* (RPL). Por fim, o protocolo de transporte utilizado é o *User Datagram Protocol* (UDP). Na inicialização do nó sensor, é criado um cliente UDP associado a uma porta, local por onde as mensagens passarão até chegar à última camada. Finalmente, na camada de aplicação está todo o código desenvolvido que contribuiu para o carácter funcional do sistema.

Tal como referido anteriormente, foi desenvolvida a camada RDC para o nó sensor. Enquanto este não estiver associado ao coordenador, o rádio permanecerá sempre ligado, de forma a receber todas as tramas enviadas pelo responsável da rede durante a associação. Após esta fase, o módulo RDC entra em modo *low power*. Neste modo, depois de enviada uma mensagem para o coordenador, o nó sensor aguarda cerca de 600 μs pelo *acknowledge*. Após este tempo, o rádio é desligado. Se este for recebido, é guardado o valor do RSSI, caso contrário, o valor de *acks* sequenciais perdidos incrementa. Se a mensagem enviada do nó sensor for um *Neighbor Solicitation* (NS), o tempo de espera até desligar o rádio é de 100 ms. O *range extender* CC2592 está configurado de modo a que esteja ligado sempre que o rádio se encontre no modo Tx ou Rx.

Assim, se o rádio estiver ligado, o *range extender* também estará.

4.3.1.2 Associação ao coordenador

A conexão de um nó sensor ao coordenador é realizada em 2 camadas, na de rede e na aplicação. A Figura 4.6 apresenta um diagrama sequência das tramas, geridas pelo sistema operativo, que irão realizar a conexão na camada de rede.

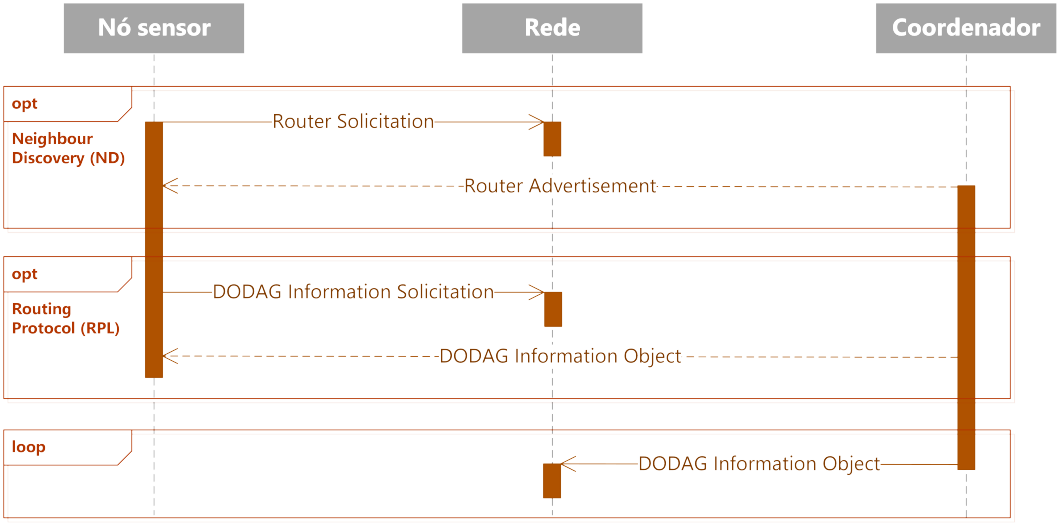


Figura 4.6: Diagrama de sequencia da conexão segundo os protocolos ND e RPL

A associação do dispositivo à rede, por parte do Contiki, pode ser efetuada por dois protocolos, *Neighbor Discovery* (ND) ou *Routing Protocol for low power and Lossy networks* (RPL). No caso do ND, é enviado um *Router Solicitation* esperando que o coordenador responda com um *Router Advertisement*. É de referir que, as mensagens de solicitação são enviadas sincronamente, pois este protocolo mantém uma lista atualizada dos dispositivos vizinhos.

Por outro lado, a associação pode ser realizada através da receção de mensagens *DODAG Information Object* (DIO). É possível pedir este tipo de mensagem enviando um *DODAG Information Solicitation* (DIS), ou esperando que o coordenador envie um DIO, visto estes serem enviados sincronamente.

Terminada a associação por parte da camada de rede, procede-se para a aplicação. Nesta fase, existem 2 mensagens que devem ser recebidas pelo nó sensor, derivadas de 2 solicitações. O *end device* envia um *Association Request*, esperando o coordenador enviar um *Association Response*. Neste momento, o nó sensor já se encontra associado, mas ainda necessita de ser

configurado. Assim, é enviado um *Configuration Request* e recebido um *Configuration Response*. Nesta última mensagem, os parâmetros que podem ser configurados pelo coordenador são a frequência de envio e o conteúdo dos *keep alive*. A Figura 4.7 apresenta um fluxograma representativo das ações que ocorrem durante a associação do nó sensor à rede.

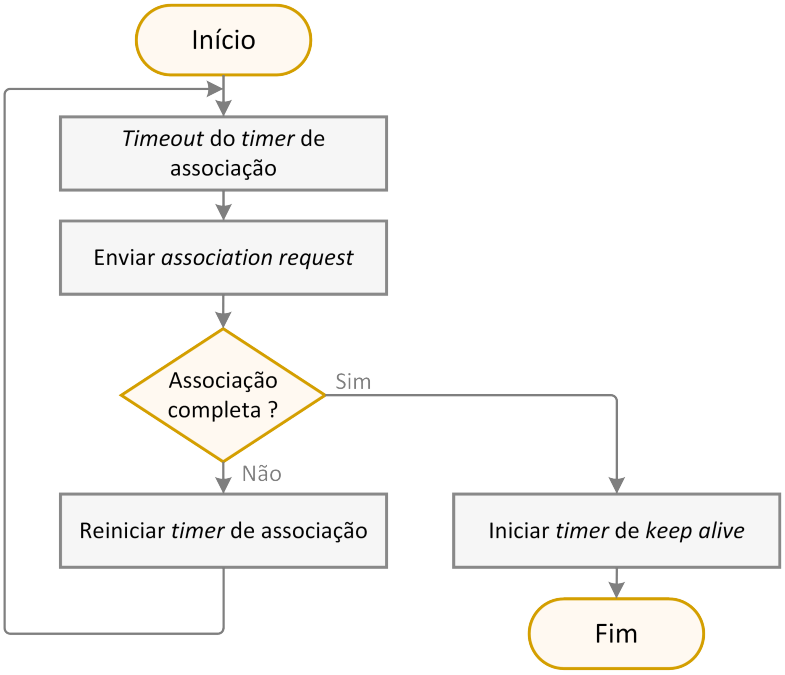


Figura 4.7: Fluxograma de associação do nó sensor

O *timer* de associação está configurado para enviar mensagens *Association Request* a cada 2 segundos, até obter resposta do coordenador. Posteriormente é efetuada a configuração dos parâmetros acima mencionados. De seguida, é iniciado o *timer* do *keep alive* com o *sampling rate* definido.

4.3.1.3 Envio das mensagens *keep alive*

A função das mensagens *keep alive* é informar o coordenador de que o nó sensor continua ativo. A cada mensagem enviada, ocorre a transmissão de um *acknowledge* por parte do coordenador. Caso a trama não chegue ao destino, o *ack* não é enviado, e é incrementada uma variável que contará o número de *acks* consecutivos perdidos. Se essa variável atingir o número 10, o nó sensor é reiniciado. Desta forma, o dispositivo poderá se conectar a outra rede.

A Figura 4.8 apresenta um fluxograma das ações que ocorrem, quando é despoletada a interrupção proveniente do *timer keep alive*. Após o *timeout* deste temporizador e acordado o

dispositivo, é verificado o número de *acknowledges* perdidos. Se este for maior que 10, predefinido como o máximo de *acks* consecutivos que podem ser perdidos, o nó sensor é reiniciado. Se isso não se verificar, é enviada uma mensagem de *keep alive*, e caso esteja definido, contendo o valor do último RSSI recebido.

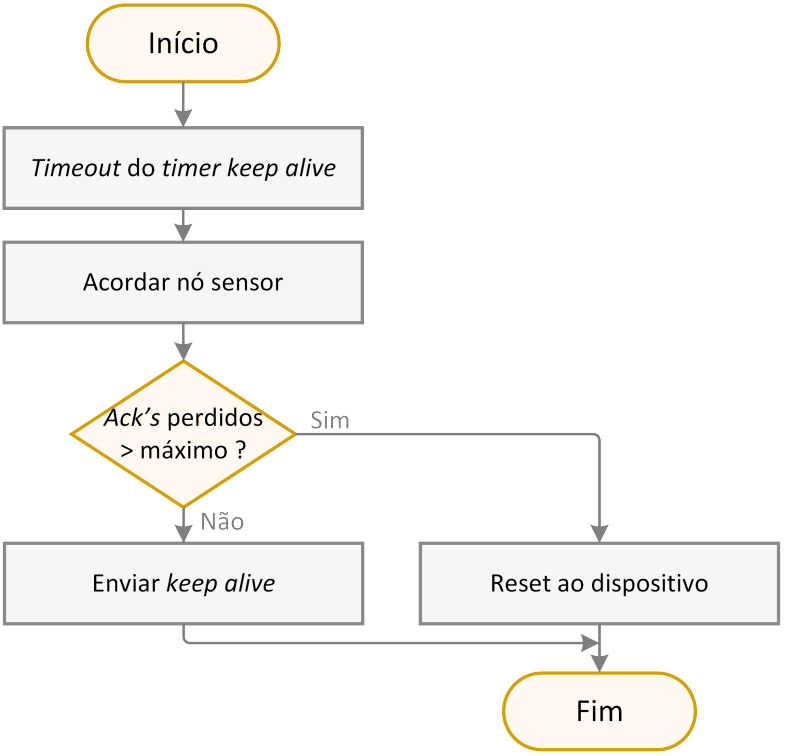


Figura 4.8: Fluxograma do nó sensor relativo ao envio do *keep alive*

O SoC CC2538 possui um *sleep timer* que permite acordar o dispositivo quando este se encontra no modo *deep sleep*. Este módulo apresenta 3 modos de *sleep*. O *power mode 3* (PM3) tem um consumo bastante baixo, cerca de 0,4 μA, mas o dispositivo só acorda por interrupções externas. No PM2, o consumo já é de 1,3 μA mas é possível ter o *sleep timer* a contar e a acordar o microcontrolador. No PM1, o tempo de *wake-up* é de 4 μs, mas apresenta um consumo de 0,6 mA. Assim, visto que é necessário um temporizador para o envio dos *keep alive* foi utilizado o PM2 para o modo *deep sleep*. O PM1 também era uma opção válida, mas o seu consumo energético teórico é superior ao PM2.

4.3.1.4 Módulo do acelerómetro

O acelerómetro tem como função, medir acelerações e alertar o microcontrolador se ocorrer uma aceleração superior ao *threshold* definido. A Figura 4.9 mostra dois fluxogramas que representam

o funcionamento deste módulo.

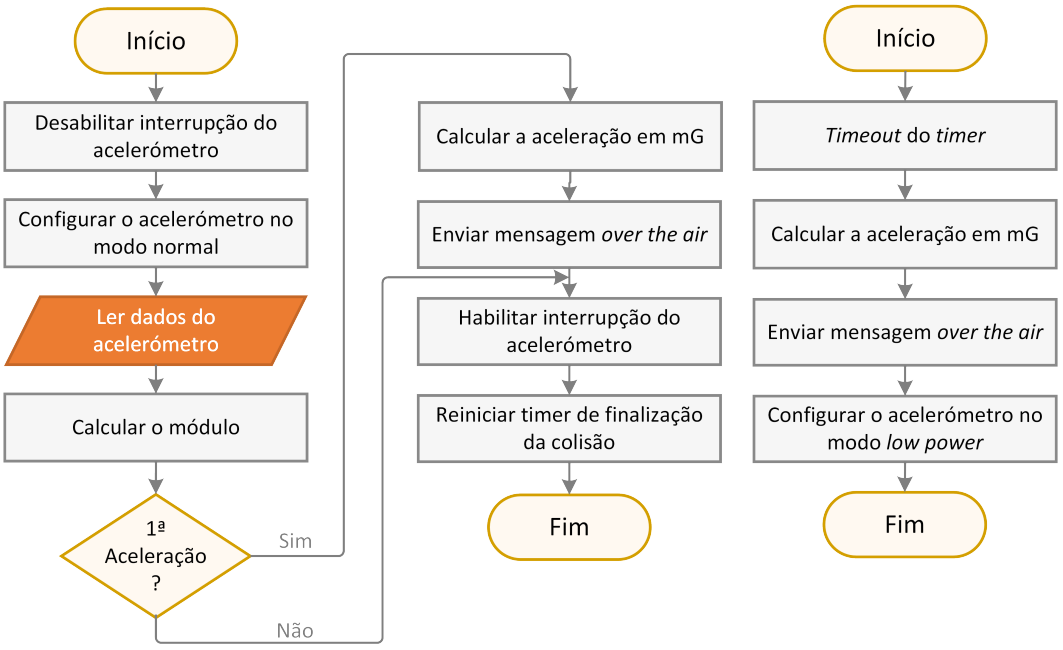


Figura 4.9: Fluxograma do acelerómetro

Quando é detetada uma aceleração superior à estipulada, o acelerómetro despoleta uma interrupção no SoC. Este, desabilita as interrupções, para não ser incomodado durante os cálculos, e configura o acelerómetro para o modo normal. Seguidamente, são lidos os valores da aceleração e calculado o seu módulo. Se este módulo for maior que o atualmente armazenado, os valores lidos da aceleração são guardados. Desta forma, ter-se-á sempre o maior módulo armazenado em memória. Caso esta seja a primeira aceleração, proveniente da possível colisão, os valores lidos são convertidos em mg, através da Equação 3.1 referida no capítulo 3, sendo enviada uma mensagem com esses valores. Seguidamente, as interrupções são novamente ativadas.

O *timer* de finalização da colisão é utilizado para finalizar a medição de uma vibração. Este é reiniciado cada vez que ocorre uma nova interrupção, e ao fim de 2 segundos sem nenhuma aceleração acima do normal, é chamada a função de fim de eventos. Nesta rotina, os valores da aceleração de maior módulo registado são convertidos em mg e enviados para o coordenador.

Em suma, é enviada uma primeira mensagem, para alertar o responsável da rede, que algo ocorreu no local do nó sensor. Na segunda mensagem, segue a maior aceleração proveniente da vibração causada pelo veículo, para depois ser analisada na *cloud*.

4.3.2 Coordenador

As funções do coordenador regem-se pela gestão da rede, e pelo reencaminhamento das tramas recebidas dos nós sensores. A Figura 4.10 apresenta um fluxograma representativo do funcionamento do coordenador.

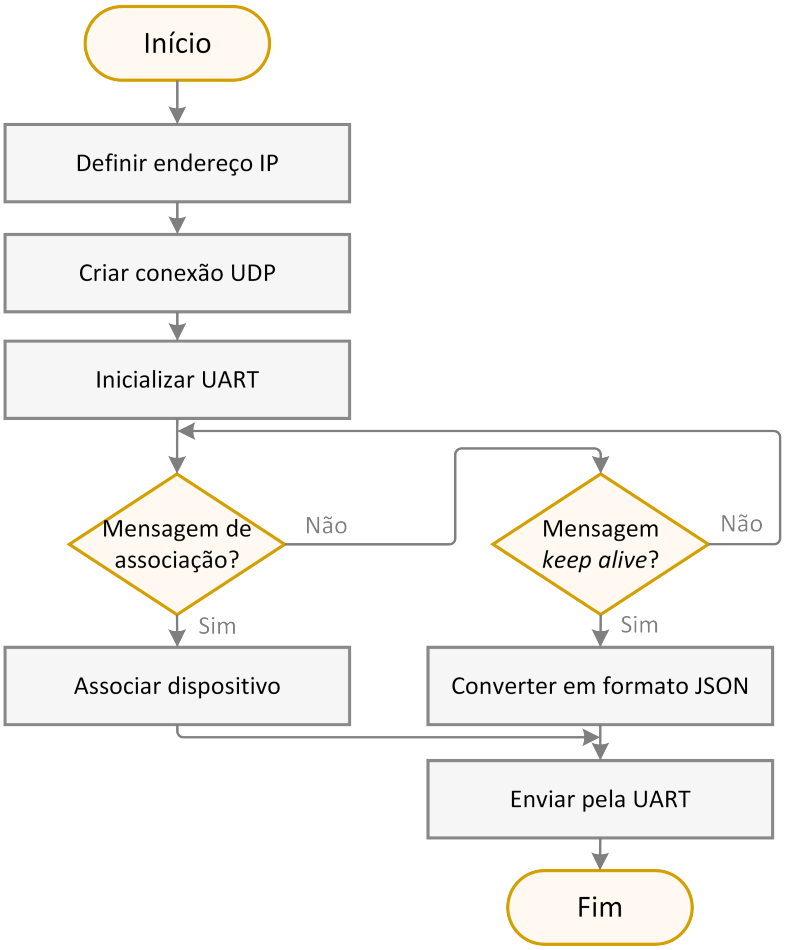


Figura 4.10: Fluxograma do coordenador

Os endereços IP no 6LoWPAN são definidos com base no endereço MAC do dispositivo. Tal como no nó sensor, o coordenador também inicia uma conexão UDP. Depois de inicializada a UART, o dispositivo está pronto para efetuar associações. Assim sendo, quando um nó sensor envia um pedido de associação, o coordenador associa-o à rede e configura-o. De seguida, envia ao *gateway* uma mensagem em formato JSON, informado-o que ocorreu uma nova conexão.

Por outro lado, quando o coordenador recebe uma mensagem *keep alive* ou de colisão envia uma mensagem JSON ao *gateway*, através da porta série.

4.3.3 Formato da mensagem *Over-the-Air* (OTA)

As mensagens enviadas dentro da rede 6LoWPAN apresentam um formato, que se foca na redução do tamanho da trama. Na camada da aplicação existem 3 tipos de mensagens, nomeadamente:

- **Mensagem de associação**: Utilizada para efetuar a associação de um nó sensor à rede;
- **Mensagem de configuração**: Serve para configurar o nó sensor com os parâmetros definidos pelo coordenador;
- **Mensagem de *keep alive***: Utilizada nos *keep alive* e também nas mensagens de colisão.

Todas as mensagens têm um *package id* (PCK_ID) fixo que ocupa apenas 1 byte. Este encontra-se dividido em *MsgType* (2 bits), que identifica o tipo da mensagem, *DevType* (2 bits), indicando o tipo de dispositivo que enviou a mensagem, e *Mask* (4 bits), que define o conteúdo da restante trama (*payload*). É de referir que, apenas existem dois valores para o *DevType*, nomeadamente, o valor 1 para mensagens provenientes do coordenador e o valor 2 para as mensagens do nó sensor. Na *Mask*, cada bit pode representar a existência de determinado campo no *payload*, como também pode especificar a mensagem dentro de cada tipo. Na Figura 4.11 está apresentado o formato da mensagem de associação.

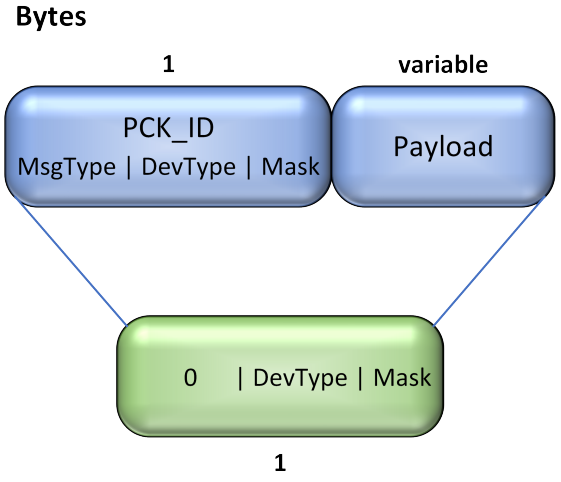


Figura 4.11: Formato das mensagens de associação

O número definido para as mensagens de associação é o zero. Apenas existem duas mensagens dentro deste tipo, *Association Request* e *Association Response*. É de notar que neste caso, o *payload* é nulo, e que cada bit da variável *Mask* especifica a mensagem.

Na Figura 4.12 está apresentado o formato das mensagens de configuração. Definido como o número um, apresenta duas possibilidades de mensagem. Pode ser *Configuration Request* ou *Configuration Response*. No *Request*, o *payload* é nulo. Já no *Response*, enviado apenas pelo coordenador, o *payload* pode conter o *sampling rate* e/ou os dados que uma mensagem *keep alive* deve conter. Em relação ao intervalo de envio das mensagens KA, o valor é em segundos. Já no conteúdo destas, apenas existe a opção de adicionar o valor RSSI do último *acknowledge* recebido pelo nó sensor. O valor da *mask* indicará a existência destes dois campos no *payload*.

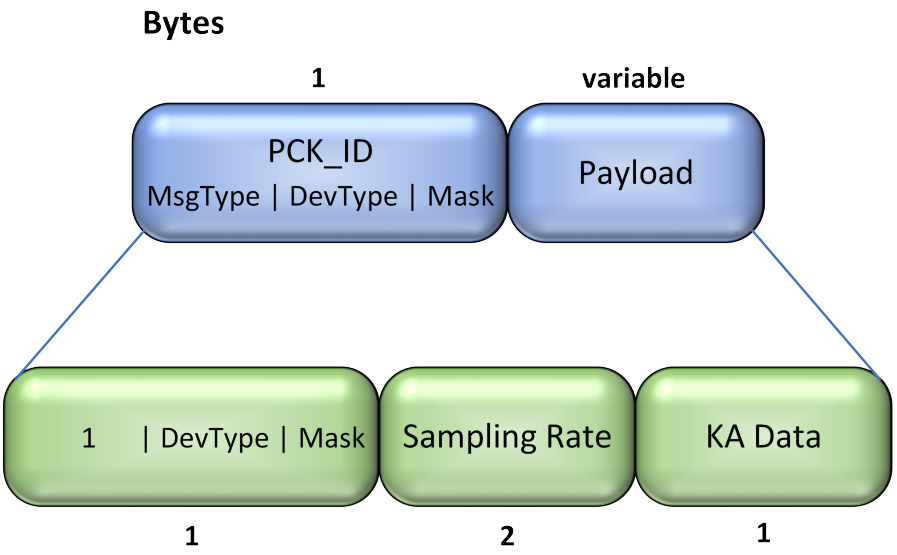


Figura 4.12: Formato das mensagens de configuração

Por último, as mensagens *keep alive* e de colisão apresentam o tipo número dois. Dado que estas mensagens são enviadas apenas pelos nós sensores, o *DevType* é sempre dois. Já no campo *Mask*, o seu valor irá depender do conteúdo do *payload*. O formato deste tipo de mensagens está apresentado na Figura 4.13. Para as mensagens *keep alive*, o *payload* apenas poderá conter o campo RSSI. Já nas mensagens de colisão conterá os dados relativos à aceleração. Por sua vez, esta é constituída por valores de aceleração em mg, total de acelerações superiores ao *threshold* recolhidas, e índice da aceleração de maior módulo.

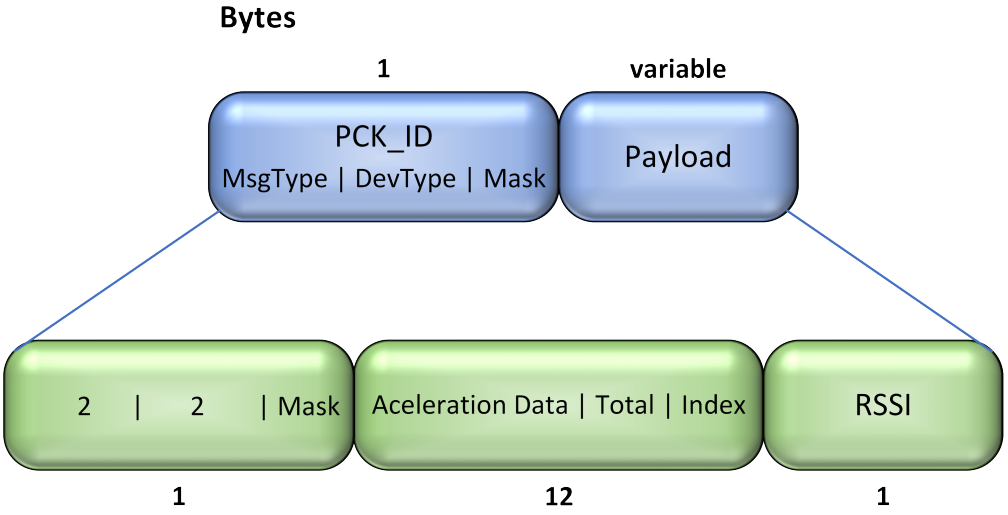


Figura 4.13: Formato das mensagens de *keep alive*

4.4 Resumo

Este capitulo descreveu a implementação do sistema. Inicialmente foram apresentadas as diferenças com a revisão anterior, bem como o *hardware* desenvolvido. Seguiu-se o *software*, mostrando algumas partes essenciais do projeto através de fluxogramas. Para além disso, recorreu-se a diagramas para descrever o formato das várias tramas utilizadas nas comunicações entre os dispositivos.

Capítulo 5

Avaliação do Sistema

No decorrer deste capítulo são apresentados os testes relativos ao sistema implementado. A validação do sistema é um dos objetivos da realização destes testes, bem como, a verificação dos consumos energéticos. Para além disso, também são apresentados alguns testes para verificação dos parâmetros RSSI e LQI.

5.1 Descrição dos testes realizados

A validação de todo o sistema recorrerá aos seguintes testes:

1. **Verificação** das mensagens reencaminhadas pelo coordenador, resultantes da comunicação com o nó sensor;
2. **Medição** do consumo do nó sensor quando se encontra em modo de *Deep Sleep*;
3. **Medição** do consumo do nó sensor quando se encontra a enviar uma mensagem de *keep alive*;
4. **Verificação** da força do sinal e da qualidade de ligação em comunicações entre o nó sensor e o coordenador.

No teste número 1 será verificado o carácter funcional do sistema. São mostradas as mensagens convertidas pelo coordenador, na qual, deverão ser a entrada para o *gateway*. Estas mensagens são geradas pelos nós sensores.

No segundo teste já é analisado o consumo do estado de *deep sleep*, no qual o nó sensor está a maior parte do tempo.

No teste seguinte também é analisado o consumo energético, mas com foco na parte em que o nó sensor se encontra a enviar uma mensagem *keep alive*. Com base nestes dois últimos testes, será possível efetuar uma previsão de duração da bateria.

Por fim, no 4º teste, é verificada a confiabilidade do sistema. Variando as distâncias, serão avaliados os valores de LQI e RSSI da comunicação entre um nó sensor e o coordenador.

5.2 Validação funcional

A validação funcional do sistema será realizada através da visualização da receção das mensagens no coordenador. Para isso, foram colocados um nó sensor e um coordenador a uma distância de 30 cm. O coordenador, conectado a um conversor de UART para USB, é ligado a um computador, e, para verificar as mensagens recebidas, recorreu-se à aplicação terminal da série *Bray*. Nesta aplicação, selecionou-se a porta correspondente, e definiu-se o *baud rate* para 115200, 8 bits de dados, sem paridade e 1 *stop bit*.

É de notar que, o nó sensor não recorre à sua porta série, e como tal, esta encontra-se sempre desligada. Com o *software* correspondente em cada um dos dispositivos, procedeu-se ao início do teste. A primeira fase é a associação do nó sensor com o coordenador. A Figura 5.1 evidencia a conclusão da associação do nó sensor à rede criada pelo coordenador. As mensagens enviadas pelo responsável da rede estarão sempre em formato JSON. Por sua vez, estas iniciarão com dados da camada de rede e depois com dados da camada de aplicação. É de reparar que a variável *DEVICETYPE* possui o valor 1, significando que é uma mensagem do coordenador. A variável *CONNECTION* contém a palavra "*new*" indicando que ocorreu uma nova conexão.


```
{"NTW" :
{"WPANID":"0xABCD","GATEWAYMAC":"00:12:4b:00:05:b3:c2:a6","LINKQUALITY":"108",
"RSSI":"-16","SECURITY":"0","SAMPLINGRATE":"60", "SEQNUMBER":"97"},"APP" :
{"DEVICEMAC":"fe80::212:4b00:5b3:c2cf","DEVICETYPE":"1","CONNECTION":"new"}}
```


Figura 5.1: Mensagem de conexão de um nó sensor

A mensagem que mais vezes é enviada é a de *keep alive* (KA) porque são geradas sincronamente, aquando da interrupção gerada pelo *sleep timer* do microcontrolador. A Figura 5.2

apresenta um exemplo de uma mensagem KA. Como é possível verificar, o *DEVICETYPE* é 2, o que significa que é uma mensagem gerada pelo nó sensor. Por outro lado, é de reparar que existem dois valores de RSSI. O valor localizado no campo de rede (*NTW*), é o RSSI do pacote enviado que gerou esta mensagem. O valor localizado no campo da aplicação (*APP*) é o RSSI do último *acknowledge*, enviado do coordenador para o nó sensor.


```
{"NTW" :
{"WPANID":"0xABCD","GATEWAYMAC":"00:12:4b:00:05:b3:c2:a6","LINKQUALITY":"108",
"RSSI":"-14","SECURITY":"0","SAMPLINGRATE":"60", "SEQNUMBER":"80"},"APP" :
{"DEVICEMAC":"fe80::212:4b00:5b3:c2cf","DEVICETYPE":"2","RSSI":"-15"}}
```


Figura 5.2: Mensagem de *keep alive* (KA)

Na Figura 5.3 estão apresentadas as duas mensagens geradas, aquando da colisão de um veículo com as guardas de segurança. Como é possível verificar, a primeira mensagem é gerada no momento em que é detetada a primeira vibração. A segunda mensagem é gerada após não serem detetadas mais acelerações. No conteúdo desta, segue os valores de aceleração, cujo módulo foi o maior. Neste caso, das 6 acelerações medidas, a que teve maior módulo foi a registada na posição 5.


```
{"NTW" :
{"WPANID":"0xABCD","GATEWAYMAC":"00:12:4b:00:05:b3:c2:a6","LINKQUALITY":"107",
"RSSI":"-16","SECURITY":"0","SAMPLINGRATE":"60", "SEQNUMBER":"102"},"APP" :
{"DEVICEMAC":"fe80::212:4b00:5b3:c2cf","DEVICETYPE":"2","ACCELERATION":{"X":"-
66.4062","Y":"-3.9062","Z":"1257.8125","TOTAL":"1","INDEX":"1"}}}

{"NTW" :
{"WPANID":"0xABCD","GATEWAYMAC":"00:12:4b:00:05:b3:c2:a6","LINKQUALITY":"108",
"RSSI":"-16","SECURITY":"0","SAMPLINGRATE":"60", "SEQNUMBER":"103"},"APP" :
{"DEVICEMAC":"fe80::212:4b00:5b3:c2cf","DEVICETYPE":"2","ACCELERATION":{"X":"-
70.3125","Y":"-58.5937","Z":"1332.0312","TOTAL":"6","INDEX":"5"}}}
```


Figura 5.3: Mensagens geradas pela colisão de um veículo

5.3 Consumos Energéticos

Neste secção são apresentados os testes realizados ao nó sensor, com o intuito de verificar o seu consumo. Este dispositivo apresenta 3 *standards* de consumo energético: quando em envio de mensagens *keep alive*, em modo de *deep sleep*, e ainda, quando deteta uma colisão. Foram realizados testes a estes *standards* à excepção deste último, pois a probabilidade de ocorrer uma

colisão é bastante reduzida. No final deste subcapítulo é também apresentada, uma estimativa de duração do nó sensor.

5.3.1 Consumo em *deep sleep*

O nó sensor estará a maior parte do tempo a dormir, mais concretamente, no estado de *deep sleep*. De forma a medir o seu consumo energético recorreu-se a uma multímetro, configurado como amperímetro. A Figura 5.4 apresenta o esquema de ligação utilizado. Como é possível verificar, o dispositivo de medição de corrente foi ligado em série, posicionando-se entre o positivo da fonte de alimentação e o pino *Vcc* do nó sensor.

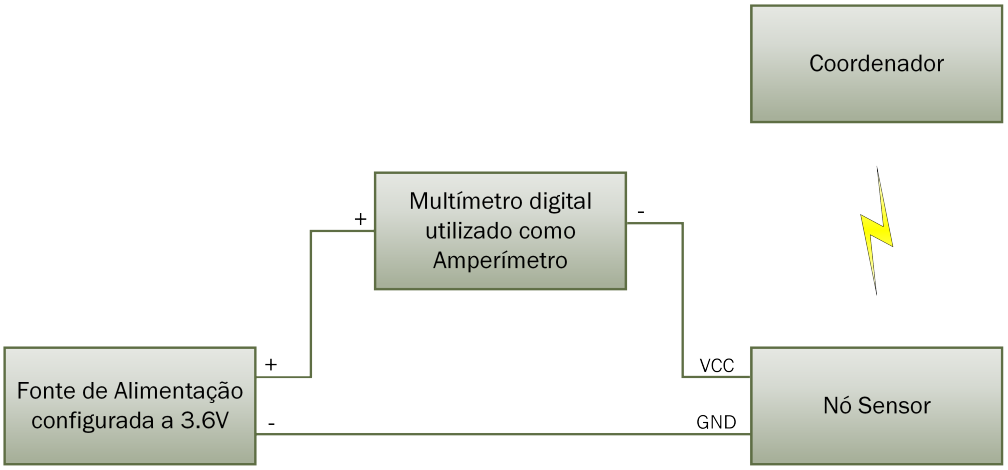


Figura 5.4: Esquema de ligação para medição do consumo energético no modo *deep sleep*

Depois de finalizada a conexão do nó sensor ao coordenador, este primeiro entra no modo de *deep sleep*. A partir deste momento, iniciou-se a medição do consumo energético e esperou-se pelo momento, imediatamente antes do envio do próximo *keep alive*, para parar a medição. Na tabela 5.1 estão apresentados os resultados obtidos, que por sua vez, foram lidos diretamente do multímetro utilizado. Estes consumos são do conjunto LDO, módulo CC2538 + CC2592 e acelerómetro, sendo que não foram realizados testes aos componentes individualmente. Para além disso, apenas foi utilizado um nó sensor assemblado para efetuar o respetivo teste.

Quando o nó sensor entra no modo de *deep sleep*, está configurado para ficar no *Power Mode* 2, pois é o modo de mais baixo consumo energético que permite acordar o dispositivo a partir de um *timer*. O acelerómetro está configurado no modo *low power* a funcionar a uma frequência de 5 Hz.

Tabela 5.1: Consumos do nó sensor no *Power Mode* 2

Valor (μA)			Número de
Mínimo	Médio	Máximo	Amostras
167,2	347,2	679,6	585

5.3.2 Consumo durante o envio do *keep alive*

Uma mensagem *keep alive* envolve, essencialmente, o acordar do dispositivo, a transmissão do pacote, a espera pelo *acknowledge* e o voltar ao modo *deep sleep*. Para realizar este teste, foi colocada uma fonte de alimentação variável, selecionada a 3,6 V, para simular a tensão da bateria usada no nó sensor. Como é possível verificar na Figura 5.5, o terminal da fonte de alimentação é conectado em série com uma resistência de 3,94 Ω e ligado ao GND do nó sensor. Em paralelo com a resistência, é colocado um osciloscópio. Com base no gráfico de tensão adquirido pelo osciloscópio, é possível aferir o consumo do dispositivo durante esta fase.

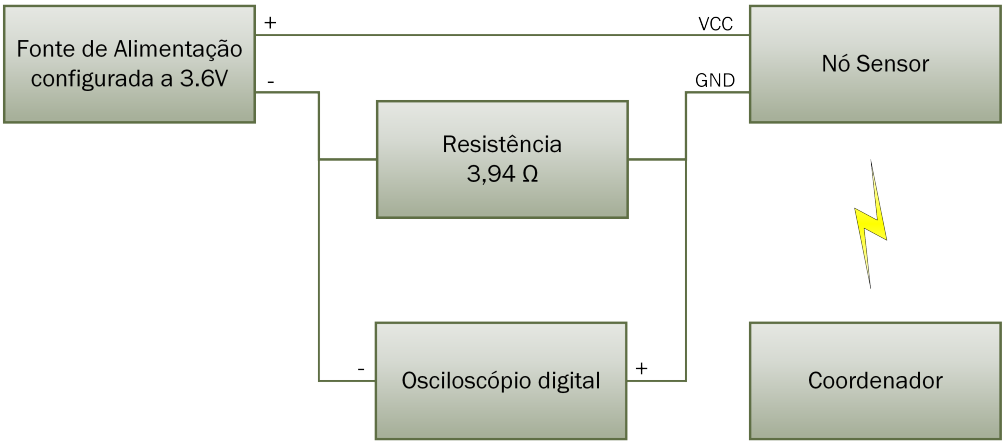


Figura 5.5: Ligação de teste para medir o consumo no modo ativo

O nó sensor e o coordenador foram colocados a uma distância de cerca de 30 cm, configurados a uma potência de transmissão de 19 dBm, sintonizados no canal 25 com o rádio a utilizar o *range extender* CC2592. De forma a visualizar a forma de onda do consumo durante o envio do *keep alive* foram retirados todos os condensadores situados entre o regulador de tensão e o módulo CC2538 + CC2592. Como o tempo entre o acordar e adormecer do dispositivo era demasiado elevado para uma visualização clara da forma de onda, dividiu-se em 2 figuras. Na Figura 5.6 está apresentada a parte correspondente ao envio da mensagem de *keep alive*, num

formato ampliado em relação ao tempo total. Já na Figura 5.7, encontra-se apresentada a forma de onda completa, começando após o acordar do dispositivo e terminando depois de este entrar em modo de *deep sleep*.

Com base nesta formas de onda, identificou-se alguns pontos de transição durante todo o processo de envio da mensagem. A tabela 5.2 apresenta alguns detalhes que ocorrem durante a fase ativa do nó sensor.

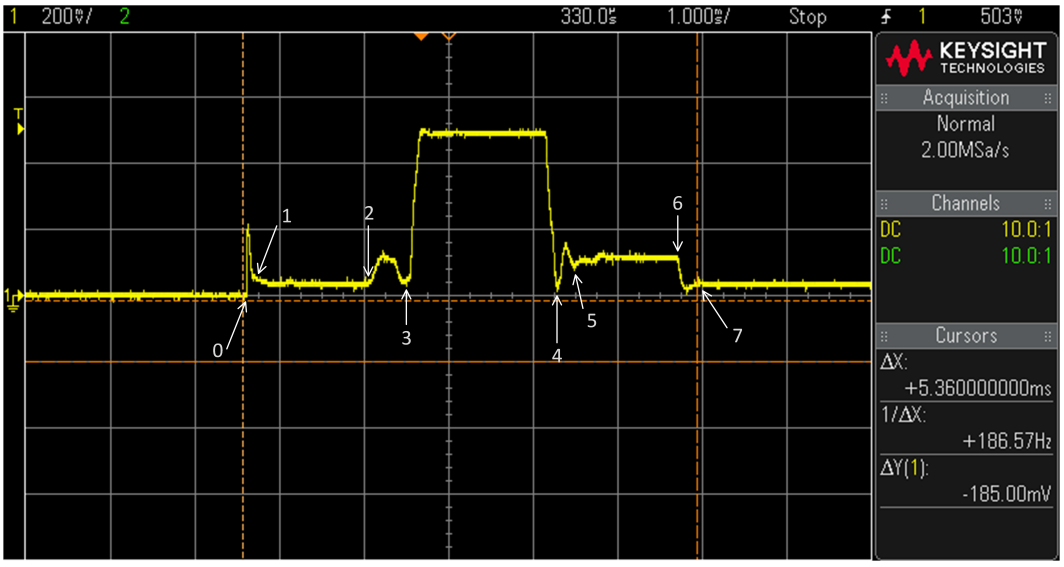


Figura 5.6: Gráfico da tensão durante o envio de uma mensagem de *keep alive* sem a utilização de condensadores

Nesta tabela, é apresentada a secção, representativa do espaço de tempo entre os momentos de transição, uma descrição da mesma, bem como alguns detalhes. Para a tensão, foi calculada a média de valores limitados por cada dois pontos de transição e medido o tempo entre estes. Com base no valor da resistência, calculou-se a corrente. Sabendo que a bateria fornece uma tensão constante de 3,6V, multiplicou-se esta tensão, pela corrente e pelo tempo, de forma a obter a quantidade de energia gasta.

Tabela 5.2: Detalhes da medição do consumo energético durante o envio do *keep alive*

Secção	Descrição	Tensão (mV)	Corrente (mA)	Tempo (ms)	Energia (μJ)
Antes de 0	*Power Mode* 2	0,09	0,02	0	0
Ponto 0 a 1	*Wake up from deep sleep*	92,39	23,45	0,135	11,40
Ponto 1 a 2	Microcontrolador funcionando a 32 MHz	36,28	9,21	1,295	42,93
Ponto 2 a 3	CSMA-CA observando o meio de transmissão	86,31	21,91	0,410	32,33
Ponto 3 a 4	CC2592 ligado, modo Tx e envio da mensagem KA	439,59	111,57	1,820	731,01
Ponto 4 a 5	CC2592 desligado, modo Rx	101,60	25,79	0,200	18,57
Ponto 5 a 6	Espera e recepção do ACK	113,57	28,83	1,220	126,60
Ponto 6 a 7	Rádio desligado	36,76	9,33	0,295	9,91
Ponto 7 a 8	Microcontrolador a processar	32,28	8,19	17	501,41
Depois de 8	*Power Mode* 2	1,78	0,45	0	0
Total				**22,4**	**1474,6**

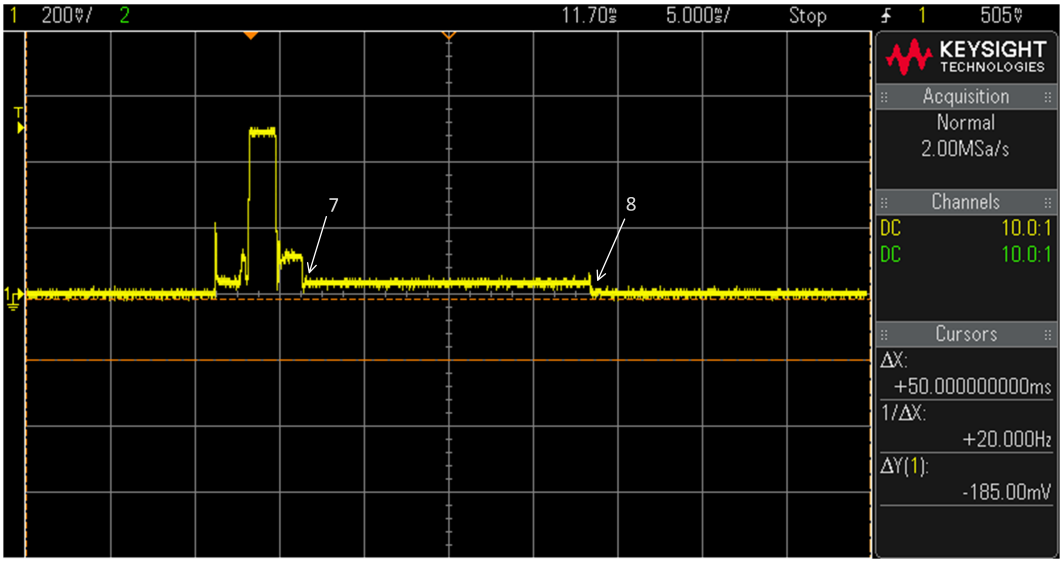


Figura 5.7: Gráfico da tensão durante o envio de uma mensagem de *keep alive* sem a utilização de condensadores

Após estes testes, recolocou-se todos os condensadores. A Figura 5.8 mostra a tensão na resistência de teste, quando o nó sensor se encontra a enviar uma mensagem de *keep alive*, mas desta vez com os respetivos condensadores. Como é possível verificar, ocorre uma ligeira

subida inicial da corrente, correspondente ao acordar do dispositivo e de algum processamento de informação. Posteriormente, o rápido incremento da corrente indica que o rádio e o *range extender* foram ligados. Por fim, o declínio corresponde a algum processamento adicional e a entrada no modo de *deep sleep*.

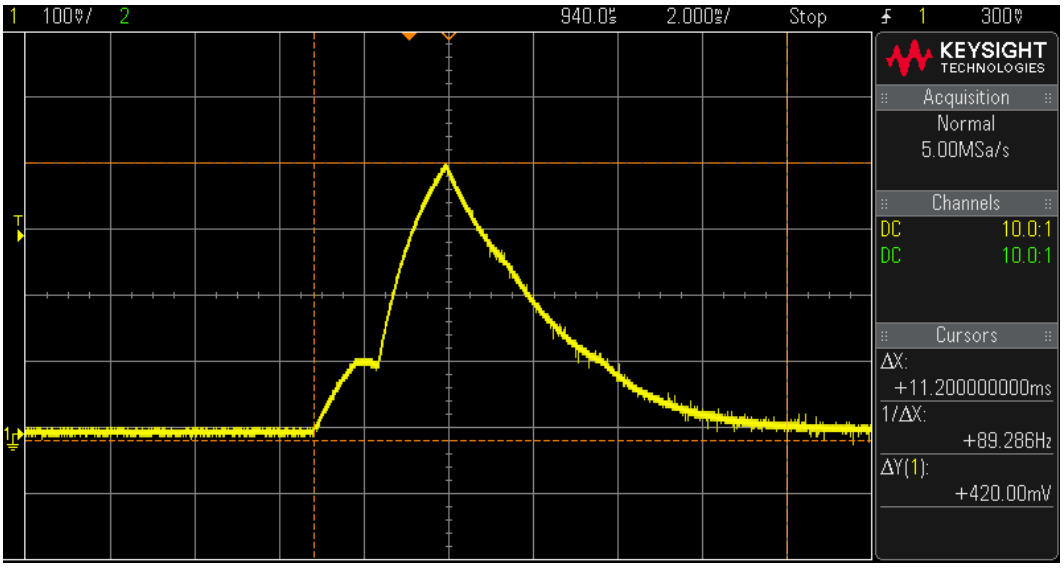


Figura 5.8: Gráfico da tensão durante o envio de uma mensagem de *keep alive*

Ao contrário das figuras 5.6 e 5.7, onde não é possível determinar os diversos estágios de consumo, a Figura 5.8 apresenta uma variação de corrente mais suave, não alcançando picos de corrente tão altos. Na versão sem condensadores, o pico máximo de corrente é de 128 mA, enquanto que na versão com condensadores é de apenas 101 mA.

O nó sensor foi configurado para enviar mensagens KA a cada 2 segundos. Durante esse intervalo de tempo, o dispositivo está em *deep sleep*, no modo PM2, com o acelerómetro no modo *low power*, verificando acelerações a uma frequência de 5 Hz. Na Figura 5.9 está representado o consumo do dispositivo durante o envio de vários *keep alive*.

5.3.3 Estimativa de duração da bateria

O nó sensor apresenta 3 *standards* de funcionamento: envio de KA, deteção de colisão e *deep sleep*. Para efeitos de estimativa de duração da bateria, e dada a baixa probabilidade de ocorrer uma colisão, apenas são contabilizados os eventos síncronos que geram as mensagens *keep alive* e o modo *deep sleep*.

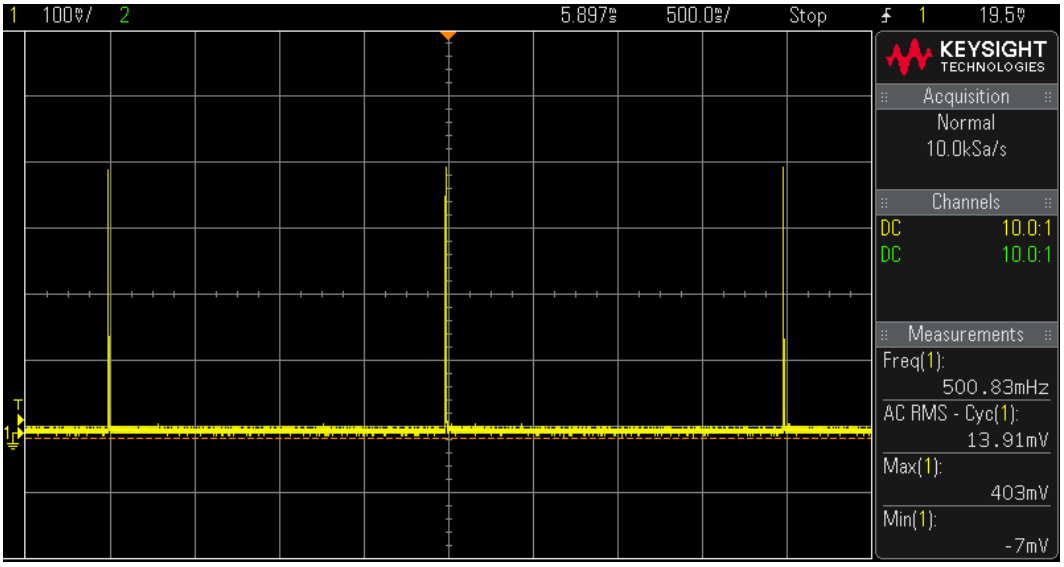


Figura 5.9: Gráfico da tensão durante o envio de três *keep alive*

A Equação 5.1 apresenta o cálculo do consumo energético médio que o sistema consome numa hora. Recorrendo á tabela 5.2, a quantidade de energia gasta durante o envio de uma mensagem *keep alive* é de 1474,6 μJ. Dividindo este valor pela tensão da bateria 3,6 V obtém-se 409,6 mA·ms. Substitui-se este valor pelo $I_{KeepAlive}$, e a variável $t_{KeepAlive}$ pelo número de *keep alives* enviados, multiplicado ao tempo gasto por cada um. Com base na tabela 5.1, a corrente no estado de *deep sleep* é de 347,2 μA. Assim sendo, com base na corrente média de cada fase de consumo, e no tempo gasto por estas, é possível afirmar o consumo médio do sistema.

$$Consumo_{médio}(mA) = \frac{I_{KeepAlive} * t_{KeepAlive} + I_{DeepSleep} * (3600 - t_{KeepAlive})}{3600} \quad (5.1)$$

Equação 5.1: Equação de cálculo do consumo energético médio por hora

A bateria utilizada, SAFT LS17500, apresenta uma capacidade nominal de 3400 mA·h. Sabendo o consumo horário e a capacidade da bateria, é possível prever a sua duração. A Equação 5.2 mostra o cálculo do tempo de vida da bateria.

$$Tempo_{vida}(h) = \frac{Capacidade_{Bateria}}{Consumo_{Sistema}} \quad (5.2)$$

Equação 5.2: Equação de cálculo para estimativa de duração da bateria

Com base nas equações anteriores, e variando o número de mensagens *keep alive* enviado por hora, obteve-se diferentes resultados de duração da bateria. A tabela 5.3 apresenta alguns desses resultados. Como é possível observar, se for enviado um KA por hora, ter-se-à uma duração de 408 dias, enquanto que se for enviado um KA por segundo, a redução é de apenas 2 dias.

Tabela 5.3: Estimativa de duração da bateria

KA por hora	Anos	Dias
1	1,12	408
60	1,12	408
1800	1,12	407
3600	1,11	406

5.4 Verificação do RSSI e LQI

O *Received Signal Strength Indication* (RSSI) e o *Link Quality Indicator* (LQI) são dois parâmetros que indicam a força do sinal e a qualidade da ligação, respetivamente. Com base nestes valores, é possível, por exemplo, determinar a distância entre dois dispositivos. Assim sendo, e tal como nos testes anteriores, o nó sensor e o coordenador foram configurados a uma potência de transmissão de 19 dBm, sintonizados no canal 25, e utilizam ambos o *range extender* CC2592.

Neste teste, colocou-se o nó sensor e o coordenador em duas distâncias diferentes. A posição geográfica dessas localizações são apresentadas na Figura 5.10. O ponto A é a localização fixa do coordenador, junto ao laboratório ESRG, enquanto que as restantes, são localizações do nó sensor. As posições B e C distam 75 e 200 metros do ponto A, respetivamente. Estas distâncias foram calculadas com recurso à ferramenta *Google Maps*.

Figura 5.10: Posição global dos dispositivos para realização dos testes

Num segundo teste, efetuou-se uma alteração na antena do coordenador, de forma a verificar se ocorreria alterações nos parâmetros acima descritos. Foi colocada uma base de alumínio

quadrada de 11 cm * 11 cm, em que no centro situava-se a mesma antena, criando um plano de terra na base desta. Os testes foram repetidos nas mesmas distâncias, obtendo-se os resultados apresentados na tabela 5.4. Nas mensagens *keep alive* recebidas pelo coordenador é possível extrair 2 valores RSSI: o que indica a força do sinal da própria mensagem e a do último *ack* recebido pelo nó sensor. Após vários testes verificou-se que os valores RSSI eram bastante semelhantes. Assim sendo, apenas se apresentou na tabela, valores relativos ao RSSI da própria trama.

Tabela 5.4: Resultados do teste aos parâmetros RSSI e LQI

Distância entre nó sensor e coordenador	Antena com base de alumínio	Valor médio LQI	Valor médio RSSI	Pacotes Entregues (%)
75 metros	Não	107	-65 dBm	100
	Sim	108	-62 dBm	77
200 metros	Não	108	-74 dBm	100
	Sim	108	-76 dBm	77

Em todos os testes, configurou-se o nó sensor a enviar uma mensagem *keep alive* por segundo durante 2 minutos. Após esse tempo, calculou-se a média de ambos os parâmetros e contabilizou-se o número de pacotes recebidos pelo responsável da rede. Posteriormente, efetuou-se o cálculo da percentagem de pacotes entregues ao coordenador.

5.5 Resumo

Ao longo deste capitulo foram descritos os teste efetuados ao sistema. Inicialmente realizou-se uma verificação funcional, do ponto de vista da ligação coordenador - *gateway*. Os testes de consumos energéticos foram divididos em 2 estágios, de *keep alive* e *deep sleep*, sendo posteriormente efetuada uma estimativa de duração da bateria. No final, apresentou-se os parâmetros RSSI e LQI.

Capítulo 6

Conclusões e trabalho futuro

Neste capitulo final são abordadas algumas conclusões sobre o trabalho desenvolvido. É efetuada uma breve recapitulação dos objetivos propostos, bem como uma pequena comparação dos resultados obtidos com os atingidos numa dissertação anterior [76]. Para além disso, também são sugeridos alguns melhoramentos para trabalho futuro.

6.1 Conclusões

A presente dissertação implementa uma WSN, enquadrada num sistema de deteção de colisões. Com base nos resultados obtidos, pode-se afirmar que é um sistema funcional, aplicável nas guardas de segurança das autoestradas.

Um dos focos desta nova revisão do sistema, consistia na remoção de componentes desnecessários, tanto a nível de *hardware* como de *software*. Desta forma, foram retirados todos os sensores, com excepção do acelerómetro, tal como descrito no subcapítulo 4.1. Em termos de *software*, foi modificado o formato da mensagem *Over-the-Air* (OTA), tornando-a mais compacta, criado um módulo *Radio Duty Cycling* (RDC) para gerir os consumos do *transceiver*, e adaptado o código já existente ao novo sistema operativo, tal como apresentado no subcapítulo 4.3. Foi também acrescentada à comunicação coordenador - *gateway*, a informação de que ocorreu nova conexão de um nó sensor.

O principal objetivo desta dissertação consistia em redesenhar do nó sensor e do coordenador, de forma a dar suporte ao Contiki OS, e conseguir implementar o protocolo 6LoWPAN nas comunicações entre os dispositivos. Este objetivo foi conseguido, resultando numa solução

genérica e facilmente adaptável a outras plataformas suportadas por este sistema operativo.

Em termos de consumo de energia no estado de *deep sleep*, e comparando com a revisão anterior [76], o nó sensor aumentou de 29,8 μA, no modo PM3, para 347,2 μA no modo PM2. É de salientar que, segundo o *datasheet* do SoC CC2538, o PM2 consome mais 0,9 μA que o PM3. Por outro lado, o tempo que o microcontrolador fica no modo ativo subiu de 10,92 ms para 22,4 ms. Contudo, a energia consumida durante o envio da mensagem *keep alive* desceu de 480 mA·ms para 409,5 mA·ms.

Em relação à autonomia, e com base nos dados apresentados anteriormente, a estimativa de duração da bateria é de 1,12 anos, com o envio de 1 *keep alive* por minuto. Na revisão anterior, para o mesmo intervalo de tempo, a duração da bateria era de 9,25 anos. A redução na autonomia é uma consequência do aumento do consumo energético no estado de *deep sleep*. Contudo, este resultado já era esperado, pois substituiu-se um sistema operativo específico, a Z-Stack da *Texas Instruments*, para um sistema operativo genérico, o Contiki OS.

Relativamente à distância de comunicação entre os dispositivos, é possível afirmar que esta aumentou ligeiramente. A utilização de um quadrado de alumínio na base da antena do coordenador revelou-se não ser vantajosa. Comparando apenas valores RSSI na situação do coordenador sem o plano de terra na antena, estes revelaram-se semelhantes para a distância de 75 metros, mas para os 200 metros, a força do sinal aumentou de -86 dBm (revisão anterior) para -74 dBm. Em relação ao parâmetro LQI, nada se pode concluir, visto que os valores obtidos para ambas as distâncias são semelhantes.

6.2 Trabalho futuro

Ainda que se tenha provado o funcionamento do sistema, bem como a sua viabilidade para utilização em autoestradas, existem diversas formas de melhoramento do mesmo. Assim sendo, como trabalho futuro enumera-se os seguintes pontos:

- Utilizar um *Real-Time Clock* (RTC) para a geração das interrupções dos *keep alive*. Assim, poder-se-ia utilizar o PM3 do SoC CC2538, invés do PM2, e possivelmente obter um menor consumo energético no nó sensor;
- Efetuar a ligação de um *gateway* ao coordenador, com o objetivo de colocar as informações provenientes desta WSN numa *cloud*;
- Realizar uma medição aos vários componentes do nó sensor a fim de determinar a origem do elevado consumo energético no modo *deep sleep*. Caso se trate do módulo CC2538, a sugestão é desligar individualmente as diversas interfaces do SoC.

Bibliografia

[1] Pordata, "Dados estatisticos sobre transportes rodoviários em Portugal." [Online]. Available: https://www.pordata.pt/Subtema/Portugal/Rodoviário-405

[2] Jornal Público, "Segurança Rodoviária." [Online]. Available: https://www.publico.pt/2017/07/03/sociedade/noticia/mais-sensibilizacao-e-fiscalizacao-para-reduzir-acidentes-nas-estradas-1777782/

[3] Trafficvision, "Traffic Intelligence from Video." [Online]. Available: http://www.trafficvision.com/

[4] K. SyedAliFathima and T. Sumitha, "To Enhance the Lifetime of WSN Network using PSO," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 1, 2014.

[5] J. Miranda, T. Gomes, R. Abrishambaf, F. Loureiro, J. Mendes, J. Cabral, and J. L. Monteiro, "A Wireless Sensor Network for collision detection on guardrails," *2014 IEEE 23rd International Symposium on Industrial Electronics (ISIE)*, pp. 1430–1435, 2014.

[6] C. Neves, L. Neves, and Lisboa, "INFRAESTRUTURAS RODOVIÁRIAS SUSTENTÁVEL," 2015. [Online]. Available: http://www.crp.pt/docs/A48S186-8_CRP_T5_074.pdf

[7] QREN, "Projetos Aprovados QREN," 2011. [Online]. Available: http://www.pofc.qren.pt/Projectos/Projectos-Aprovados-QREN?area=2&search=y&NumProjecto=23113

[8] Ascendi, "SustIMS." [Online]. Available: https://www.ascendi.pt/noticias/titulo-6-2-3/

[9] T. Gomes, F. Salgado, A. Tavares, and J. Cabral, “CUTE Mote, A CUstomizable and Trustable End-device for the Internet of Things,” *IEEE Sensors Journal*, vol. 17, no. 20, pp. 6816–6824, 2017.

[10] IEEE, “Strategic Plan 2015-2020,” 2015.

[11] IEEE, “IEEE - Mission & Vision.” [Online]. Available: https://www.ieee.org/about/vision-mission.html

[12] IEEE, “IEEE Standarts Association.” [Online]. Available: http://grouper.ieee.org/groups

[13] B. M. Pires, “Seleção dinâmica de interface em redes IEEE 802.15.4,” 2017.

[14] C. M. E. Bormann and A. K. Ericsson, “Terminology for Constrained-Node Networks,” *RFC 7228 (Informational), Internet Engineering Task Force*, pp. 1–17, 2014.

[15] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, “Operating Systems for Low-End Devices in the Internet of Things: A Survey,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 720–734, 2016.

[16] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, “Wireless sensor networks: A survey on recent developments and potential synergies,” *Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014.

[17] R. Kennelly, *IEEE 802.15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2003, vol. 30, no. 2.

[18] LAN/MAN Standards Comitee, *Std. 802.15.4-2006*, 2006, vol. 2006, no. September.

[19] N. Salman, I. Rasool, and A. H. Kemp, “Overview of the IEEE 802.15.4 standards family for low rate wireless personal area networks,” *Proceedings of the 2010 7th International Symposium on Wireless Communication Systems, ISWCS’10*, pp. 701–705, 2010.

[20] I. Howitt and G. Jore A., *IEEE 802.15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*, 2011, vol. 2011, no. September.

[21] L. D. Nardis and M. D. Benedetto, "Overview of the IEEE 802.15. 4/4a standards for low data rate Wireless Personal Data Networks," *Positioning, Navigation and ...*, vol. 2007, pp. 285–289, 2007.

[22] E. KOUBAA, Anis; ALVES, Mário; TOVAR, "IEEE 802.15.4 for Wireless Sensor Networks: A Technical Overview," no. Stage 3, pp. 18–21, 2005.

[23] G. Mulligan, "The 6LoWPAN architecture," *Proceedings of the 4th workshop on Embedded networked sensors - EmNets '07*, p. 78, 2007.

[24] IETF, "IPv6 over Low power WPAN (6lowpan)," 2012. [Online]. Available: https://datatracker.ietf.org/wg/6lowpan/documents/

[25] N. Kushalnagar, G. Montenegro, and C. Schumacher, "6LoWPANs: Overview, Assumptions, Problem Statement, and Goals," *RFC 4919, IETF*, pp. 1–12, 2007.

[26] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," *RFC 4944, IETF*, pp. 1–30, 2007.

[27] J. Olsson, "6LoWPAN demystified," *Texas Instruments*, p. 13, 2014.

[28] T. Gomes, F. Salgado, S. Pinto, J. Cabral, and A. Tavares, "A 6lowpan accelerator for internet of things endpoint devices," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 371–377, Feb 2018.

[29] Z. Shelby and C. Bormann, *6LoWPAN :The wireless embedded internet*, 2009.

[30] S. Kent and K. Seo, "RFC 4301 Security Architecture for IP," *Ietf*, pp. 1–101, 2005.

[31] X. Ma and W. Luo, "The analysis of 6LowPAN technology," *Proceedings - 2008 Pacific-Asia Workshop on Computational Intelligence and Industrial Application, PACIIA 2008*, vol. 1, no. 3, pp. 963–966, 2008.

[32] N. H. A. Ismail, R. Hassan, and K. W. M. Ghazali, "A study on protocol stack in 6lowpan model," *Journal of Theoretical and Applied Information Technology*, vol. 41, no. 2, pp. 220–229, 2012.

[33] X. Liu and L. Xiao, "A survey of multihoming technology in stub networks: Current research and open issues," *IEEE Network*, vol. 21, no. 3, pp. 32–40, 2007.

[34] Z. Suryady, M. H. M. Shaharil, K. A. Bakar, R. Khoshdelniat, G. R. Sinniah, and U. Sarwar, "Performance evaluation of 6LoWPAN-based precision agriculture," *International Conference on Information Networking 2011, ICOIN 2011*, pp. 171–176, 2011.

[35] J. Hui and A. R. Corporation, "6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture," *Internet Protocol for Smart Objects (IPSO) Alliance White paper 3*, vol. 6, no. 59, pp. 1–17, 2009.

[36] E. Lehmann, "6LoWPAN Fundamentals Introduction into the "Internet of Things"," no. July, 2012.

[37] S. Hossen, A. F. M. S. Kabir, R. H. Khan, and A. Azfar, "Interconnection between 802 . 15 . 4 Devices and IPv6 : Implications and Existing Approaches," *Journal of Computer Science*, vol. 7, no. 1, pp. 19–31, 2010.

[38] M. R. Palattella, N. Accettura, X. Vilajosana, T. Watteyne, L. A. Grieco, G. Boggia, and M. Dohler, "Standardized Protocol Stack for the Internet of Important Things," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 3, pp. 1–32, 2013.

[39] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[40] P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks," pp. 1–24, 2011.

[41] A. Nasarre Ramírez, "Internet of things implementation with Raspberry Pi," *Mycotoxin research*, 2014.

[42] C. Lakshmi Devasena, "6LoWPAN for networking Internet of Things (IoT) - Analyzing its suitability for IoT," *Indian Journal of Science and Technology*, vol. 9, no. 30, 2016.

[43] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "Rfc 6550: Rpl," *RFC 6550, IETF*, pp. 1–157, 2012.

[44] T. Clausen, U. Herberg, and M. Philipp, "A critical evaluation of the IPv6 Routing Protocol for Low Power and Lossy Networks (RPL)," *International Conference on Wireless and Mobile Computing, Networking and Communications*, pp. 365–372, 2011.

[45] M. Richardson, "RPL- Routing over Low Power and Lossy Networks," *Ietf 94*, pp. 1–66, 2012.

[46] M. A. Seliem, K. M. Elsayed, and A. Khattab, "Optimized neighbor discovery for 6LoWPANs: Implementation and performance evaluation," *Computer Communications*, vol. 112, pp. 73–92, 2017.

[47] Z. Alliance, "Zigbee Alliance." [Online]. Available: https://www.zigbee.org/

[48] Y. Li, K. Zhang, and X. Zhang, "Research on application of ZigBee technology in flammable and explosive environment," *2009 1st International Conference on Information Science and Engineering, ICISE 2009*, vol. 2010, no. June, pp. 4074–4078, 2009.

[49] P. Dhillon and P. Malhotra, "A Review paper on History, Current trends and Future of Zigbee (IEEE 802.15.4) Standard Parneet," *Engineering Science and Technology*, vol. 3, pp. 293–298, 2013.

[50] B. Mihajlov and M. Bogdanoski, "Overview and Analysis of the Performances of ZigBee-based Wireless Sensor Networks," *International Journal of Computer Applications*, vol. 29, no. 12, pp. 975–8887, 2011.

[51] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," *RFC 3561*, 2003.

[52] C. W. Lu, S. C. Li, and Q. Wu, "Interconnecting ZigBee and 6LoWPAN wireless sensor networks for smart grid applications," *Proceedings of the International Conference on Sensing Technology, ICST*, pp. 267–272, 2011.

[53] E. Toscano and L. Lo Bello, "Comparative assessments of IEEE 802.15.4/ZigBee and 6LoWPAN for low-power industrial WSNs in realistic scenarios," *IEEE International Workshop on Factory Communication Systems - Proceedings, WFCS*, pp. 115–124, 2012.

[54] Thread Group, "What is Thread?" [Online]. Available: https://www.threadgroup.org/What-is-Thread/Overview

[55] Thread Group, “Thread Usage of 6LoWPAN,” 2015.

[56] Thread Group, “Thread in Commercial White Paper,” no. May, pp. 1–25, 2018.

[57] Thread Group, “Thread Stack Fundamentals,” pp. 1–21, 2015.

[58] Texas Instruments, “What is Thread?” [Online]. Available: http://www.ti.com/wireless-connectivity/simplelink-solutions/thread/overview.html

[59] T. Rault, A. Bouabdallah, and Y. Challal, “Energy efficiency in wireless sensor networks: A top-down survey,” *Computer Networks*, vol. 67, pp. 104–122, 2014.

[60] E. A. Lee, “Cyber physical systems: Design challenges,” in *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, vol. 00, 05 2008, pp. 363–369.

[61] R. Baheti and H. Gill, *Cyber-Physical Systems: From Theory to Practice*, 2011, no. 1.

[62] A. Humayed, J. Lin, F. Li, and B. Luo, “Cyber-Physical Systems Security – A Survey,” *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–1, 2017.

[63] Y. ZHANG, F. XIE, Y. DONG, G. YANG, and X. ZHOU, “HIGH FIDELITY VIRTUALIZATION OF CYBER-PHYSICAL SYSTEMS,” *International Journal of Modeling, Simulation, and Scientific Computing*, vol. 04, no. 02, p. 1340005, 2013.

[64] T. Sanislav, S. Zeadally, G. Mois, and H. Fouchal, “Multi-agent architecture for reliable cyber-physical systems (cps),” *9th IEEE International Workshop on Performance Evaluation of Communications in Distributed Systems and Web Based Service Architectures (PEDISWESA 2017)*, no. Pediswesa, pp. 2–7, 2017.

[65] E. A. Lee and S. A. Seshia, *Introduction to Embedded Systems - A Cyber-Physical Systems Approach*, 2nd ed. MIT Press, 2017.

[66] S. Zanero, “Cyber-Physical Systems,” *Computer*, vol. 50, no. 4, pp. 14–16, 2017.

[67] S. Schirrmacher, L. Overmeyer, and S. Lorisch, “Wireless condition monitoring of a marine gearbox,” *Ship Technology Research*, vol. 63, no. 1, pp. 38–49, 2016.

[68] H. C. Abreu, "Universidade do Minho Monotorização das Condições Atmosféricas nas Autoestradas utilizando WSNs," 2014.

[69] S. Pinto, J. Cabral, and T. Gomes, "We-care: An iot-based health care system for elderly people," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, March 2017, pp. 1378–1383.

[70] T. Gomes, J. Brito, H. Abreu, H. Gomes, and J. Cabral, "Greenmon: An efficient wireless sensor network monitoring solution for greenhouses," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, March 2015, pp. 2192–2197.

[71] T. Gomes, N. Brito, J. Mendes, J. Cabral, and A. Tavares, "Weco: A wireless platform for monitoring recycling point spots," in *2012 16th IEEE Mediterranean Electrotechnical Conference*, March 2012, pp. 468–472.

[72] J. Weiyun, W. Xiaojing, and Z. Li, "Monitoring system of car-guardrail accident based on wireless sensor networks," *Proceedings - 2008 8th International Conference on Intelligent Transport System Telecommunications, ITST 2008*, pp. 146–149, 2008.

[73] LESIM, "Progetto barriera attiva." [Online]. Available: http://lesim1.ing.unisannio.it/index.php/it/didattica/formazione/barriera-attiva

[74] P. Daponte, L. De Vito, F. Picariello, S. Rapuano, and I. Tudosa, "Wireless Sensor Network for traffic safety," *2012 IEEE Workshop on Environmental Energy and Structural Monitoring Systems (EESMS)*, no. September, pp. 42–49, 2012.

[75] Costa, André, "Deteção de Colisões com Guardas de Segurança," 2013.

[76] A. Ribeiro, "Implementação de nó sensor baseado em SoC ARM e ZigBee," 2018.

[77] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[78] Contiki, "Contiki: The Open Source OS for the Internet of Things." [Online]. Available: http://contiki-os.org/

[79] Adam Dunkels, “Contiki: Bringing IP to Sensor Networks,” 2013. [Online]. Available: https://ercim-news.ercim.eu/en76/rd/contiki-bringing-ip-to-sensor-networks

[80] IPSO, “IPSO Challenge 2013,” 2013. [Online]. Available: http://challenge.ipso-alliance.org/ipso-challenge-2013/

[81] IPSO, “Energy Harvest.” [Online]. Available: http://challenge.ipso-alliance.org/energy-harvest/

[82] M. O. Farooq and T. Kunz, “Operating systems for wireless sensor networks: A survey,” *Sensors*, vol. 11, no. 6, pp. 5900–5930, 2011.

[83] A. Dunkels, B. Grönvall, and T. Voigt, “Contiki - A lightweight and flexible operating system for tiny networked sensors,” *Proceedings - Conference on Local Computer Networks, LCN*, pp. 455–462, 2004.

[84] T. Instruments, “CC2538 Powerful Wireless Microcontroller System-On-Chip,” 2015.

[85] T. Instruments, “CC2592 2.4-GHz Range Extender CC2592,” 2014.

[86] STMicroelectronics, “LIS331DLH,” no. July, pp. 1–38, 2009.

[87] Douglas Crockford, “Introducing JSON.” [Online]. Available: https://www.json.org/

[88] T. Bray, “The javascript object notation (json) data interchange format,” pp. 1–16, 2017.

[89] T. Instruments, “Code Composer Studio (CCS) Integrated Development Environment (IDE).” [Online]. Available: http://www.ti.com/tool/CCSTUDIO

Anexos

Anexo A

Conexão do nó sensor ao coordenador

Na Figura A.1 estão representadas as mensagens que são trocadas entre o nó sensor e o coordenador, na camada de aplicação. É de salientar que a topologia utilizada neste projeto é em estrela, e apenas são utilizados nós sensores e um coordenador.

O nó sensor envia uma mensagem de *Association Request* de forma a que o coordenador devolva um *Association Response*. Feito isto, é necessário efetuar uma configuração por parte do coordenador. Assim, é enviado um *Configuration Request*, para que o coordenador envie um *Configuration Response*.

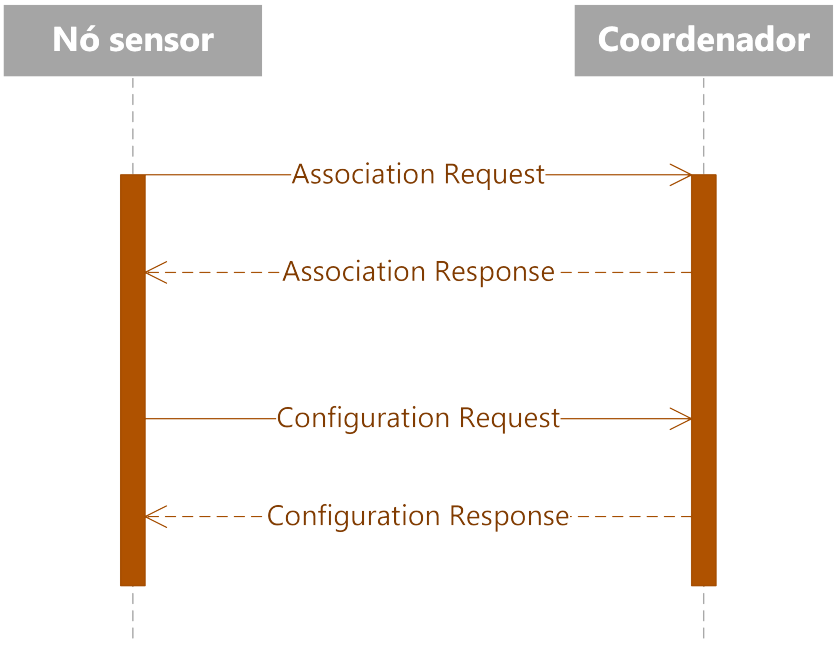
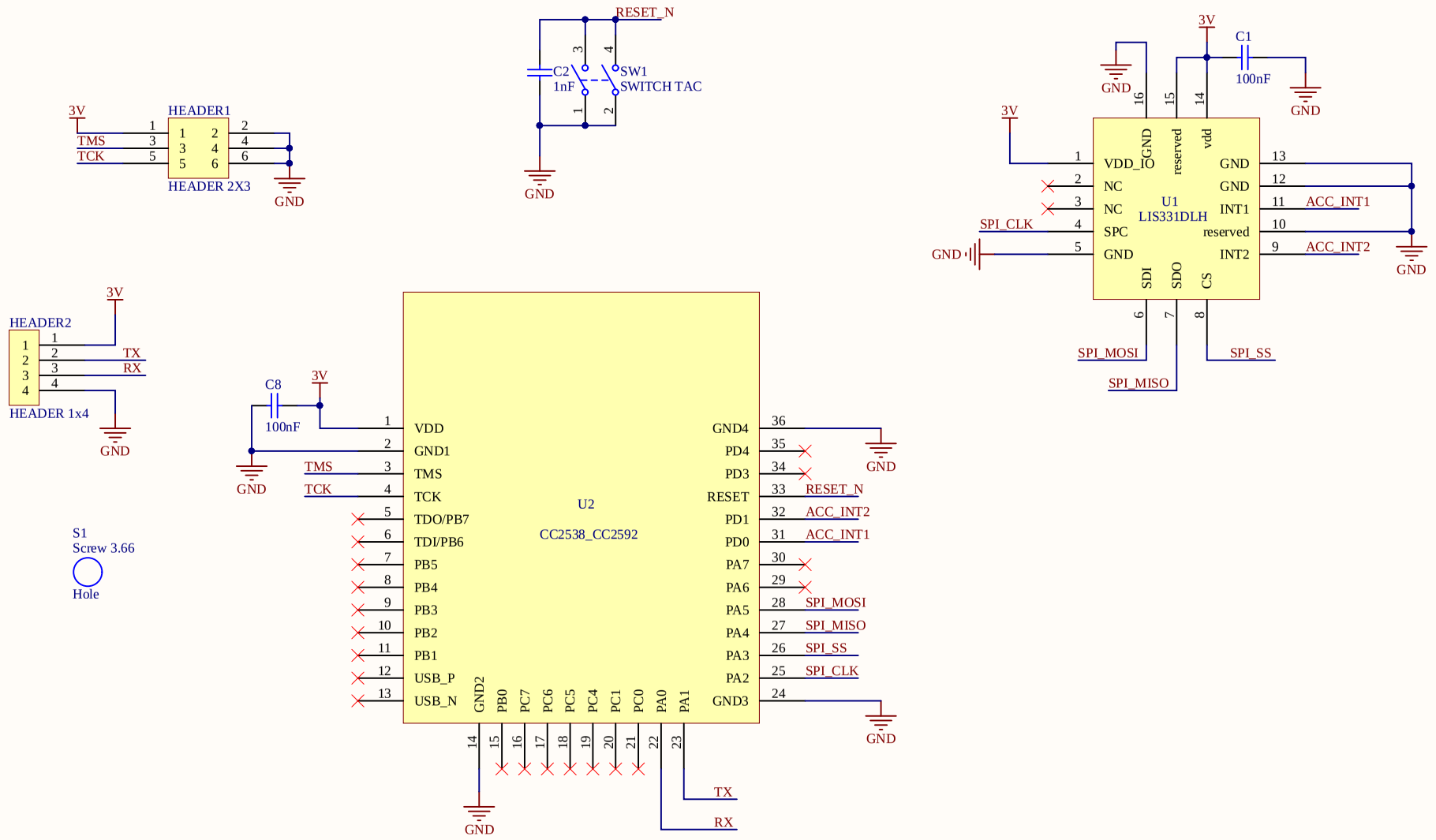


Figura A.1: Diagrama de sequência das mensagens de associação e configuração

Anexo B

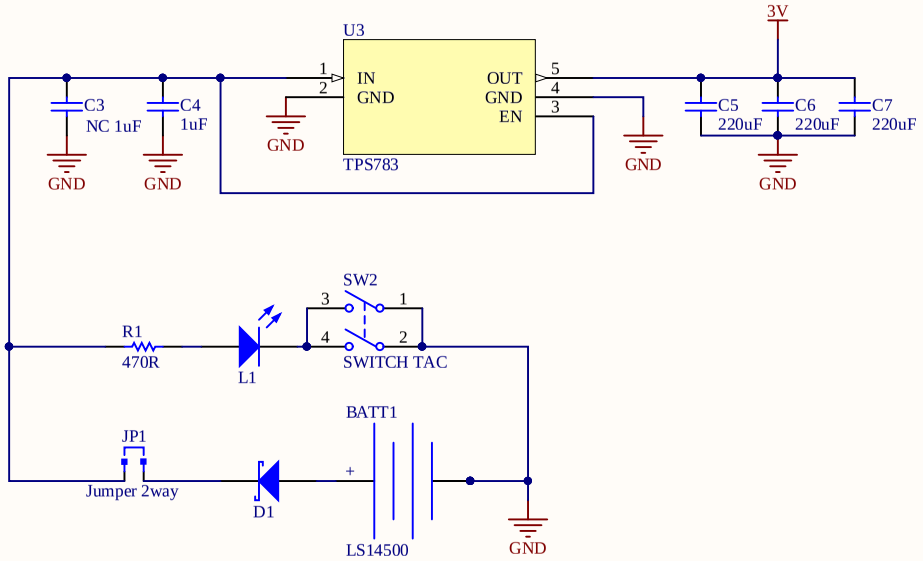
Esquemáticos do PCB realizado



Title	***CC2538 + CC2592***		
Size: A4	Number: 1	Revision: v1	
Date: 27/02/2019	Time: 15:10:36	Sheet 1 of 2	
File: E:\PCB_CC2538\Main.SchDoc			

Luís Pereira

Altium



Title	*CC2538 + CC2592*		
Size: A4	Number:2	Revision:v1	
Date: 27/02/2019	Time: 15:10:36	Sheet2 of 2	
File: E:\PCB_CC2538\Power.SchDoc			

Luís Pereira