ZigBee

uma abordagem prática

Mário Saleiro e Emanuel Ey

Este documento surge como um trabalho para a cadeira de Redes de Comunicação do 3ºAno do curso de Engenharia Eléctrica e Electrónica no ramo de Tecnologias de Informação e Telecomunicações e tem como objectivo elucidar os interessados sobre as características e modos de funcionamento do sistema comunicação sem fios ZigBee. Para tal far-se-á uma abordagem prática do sistema de comunicação utilizando os módulos XBee da Maxstream.

Docente: Jânio Monteiro

Universidade do Algarve
Escola Superior de Tecnologia
Engenharia Eléctrica e Electrónica

Índice

Introdução

Porquê o ZigBee? 3

Como surgiu o ZigBee? 4

Quais as suas potencialidades? 4

Protocolo IEEE 802.15.4 e ZigBee

Os protocolos 6

IEEE 802.15.4 6

ZigBee 8

Topologias e modos de operação das Redes ZigBee 10

Arquitectura do protocolo 15

Nota conclusiva 23

XBee

O que são? 24

Características 25

Aplicação demonstrativa

Objectivos 26

Descrição da aplicação 26

Hardware e software 28

Configurar o XBee 39

Conclusão - o futuro 51

Referências 52

Introdução

- Porquê o zigbee?

Até há pouco tempo não existia no mercado uma solução normalizada para redes sem fios para aplicações de controlo e telemetria. Assim, para redes com requisitos como baixo consumo e elevada fiabilidade, cada fabricante tinha duas opções: desenvolver a sua própria tecnologia de comunicação proprietária ou adquirir uma licença para uma tecnologia de outro fabricante, tornando complexa e cara a tarefa de desenvolver este tipo de aplicações.

Embora a comunicação sem fios já fosse comum, não havia uma solução desenvolvida especificamente para aplicações de controlo de sistemas de ar condicionado, electrodomésticos, iluminação residencial, sistemas de segurança e vigilância, brinquedos, aquisição de dados de sensores de temperatura, humidade, precipitação, luminosidade, pressão, etc.

Figura 1- Aplicações do ZigBee

Com o crescimento do mercado e o aumento de complexidade da automação industrial e residencial tornou-se então necessário o desenvolvimento

de uma tecnologia que permitisse interoperabilidade entre dispositivos de fabricantes diferentes.

Eis então que surgiu no mercado o ZigBee. Com uma filosofia diferente da das redes sem fios já existentes como o *WiFi* ou o *Bluetooth*, o ZigBee, aposta não em elevada largura de banda para transmissão de grandes quantidades de dados, mas sim na comunicação fiável em combinação com consumo extremamente baixo.

- Como surgiu o zigbee?

As bases da tecnologia denominada hoje por ZigBee foram estabelecidas no protocolo *Home RFLite* criado pela Philips. A tecnologia foi pela primeira vez apresentada ao público com o nome de ZigBee em Julho de 2005. O nome ZigBee veio da analogia entre o funcionamento de uma *mesh network* e a maneira como as abelhas trabalham e se deslocam. As que vivem numa comeia voam em *zig zag*, de modo que quando voam em busca de néctar comunicam com outras abelhas da mesma colmeia, dando informações sobre a distância, direcção e localização de alimentos. Juntando o *zig zag* com abelha em inglês, Bee, temos hoje o ZigBee!

- Quais as suas potencialidades?

Uma das grandes vantagens do ZigBee é o facto de suportar redes em malha, onde, havendo vários caminhos possíveis, é possível eliminar falhas de comunicação no caso de falha de um nó de rede.

Embora tenha sido originalmente criado a pensar em redes em malha (*mesh*), suporta também topologias em estrela (*star*) e em árvore (*cluster tree*), permitindo o estabelecimento de redes de nós "*ad-hoc*". Independentemente do tipo de rede implementada, o protocolo permite até 65535 dispositivos por cada nó coordenado (*ZigBee Coordinator*).

Na topologia em malha (*mesh*), a rede auto-organiza-se de forma a optimizar o tráfego de dados, podendo abranger áreas geográficas relativamente extensas como por exemplo um prédio de grandes dimensões.

A topologia em estrela (*star*) é a rede de implementação mais simples, sendo composta por um nó coordenador e até 65535 nós terminais. Dado que toda a comunicação é gerida pelo nó coordenador, esta topologia é deve ser implementada em locais que não ofereçam muitos obstáculos à transmissão e recepção.

Com algumas semelhanças à topologia em malha, as redes em árvore (*cluster tree*) têm uma estrutura muito hierarquizada em que o nó coordenador assume um papel de nó mestre para a troca de informação entre os nós *router* e os nós terminais (*End Devices*)

Devido ao protocolo relativamente simples o desenvolvimento do código é simplificado, levando a custos reduzidos no desenvolvimento de aplicações. Outro factor que simplifica o desenvolvimento de aplicações é o facto de não haver inúmeros modos de funcionamento à escolha, mas sim apenas dois estados tanto para envio como para recepção - *active* e *sleep*.

Para além disto o protocolo possui também um reduzido tempo de ligação à rede e uma rápida transição entre modos de funcionamento, fazendo com que o ZigBee apresente também um baixa latência.

Operando na gama livre dos 2,4GHz, isenta de licenciamento, o protocolo ZigBee permite comunicações com excelente imunidade a interferências e taxas de transferência de dados entre os 20Kbps e os 250Kbps.

Se assim forem configurados, os módulos entram em modo *sleep* quando não estão a transmitir ou receber dados, levando a um consumo de energia muito reduzido. Este consumo reduzido - que é um dos grandes objectivos deste protocolo - permite a criação de dispositivos que funcionem durante meses ou anos alimentados apenas por pilhas comuns.

Protocolo IEEE 802.15.4 e ZigBee

- Os protocolos

A criação de redes sem fios pode ser feita utilizando uma grande variedade de protocolos de radiofrequência (RF). Alguns desses protocolos são propriedade de entidades independentes e outros são protocolos base, ou seja, *standard* da indústria. Nesta parte deste documento iremos analisar o protocolo ZigBee, que é um *standard* da indústria para a transmissão de dados, e o IEEE 802.15.4, que é o protocolo base sobre o qual foi desenvolvido o ZigBee. Iremos abordar as frequências usadas, as larguras de banda necessárias e as capacidades únicas de cada um dos protocolos no que diz respeito ao estabelecimento de redes de comunicação. Iremos também abordar os objectivos que se tiveram em conta quando se desenvolveram estes protocolos.

- IEEE 802.15.4

O IEEE 802.15.4 é um *standard* para o estabelecimento de comunicações *wireless* desenvolvido pelo IEEE (Instituto de Engenheiros Eléctricos e Electrónicos). O IEEE é uma associação técnica e profissional que já desenvolveu uma grande quantidade de *standards* para promover o crescimento e a compatibilidade entre tecnologias recentes e tecnologias já existentes. O IEEE publicou os *standards* que definem a comunicação em áreas como a Internet, os periféricos dos computadores (Firewire – IEEE 1394), comunicações industriais e comunicações wireless (wireless LANs – IEEE 802.11,wireless MANs – 802.16, Wi-Fi, Bluetooth).

Enquanto esses *standards* foram desenvolvidos com a preocupação numa grande largura de banda para serem utilizados em aplicações de acesso à Internet, o IEEE 802.15.4 foi desenvolvido com uma menor taxa de transmissão, fácil conectividade e mínimo consumo de energia. Este protocolo especifica que a comunicação pode ocorrer em 3 bandas diferentes, destinadas a aplicações científicas, industriais e médicas (ISM).:

- 868 -868,8 MHz

- 902-928 MHz

- 2.400 - 2.4835 MHz

Apesar de qualquer uma destas bandas poder ser utilizada para os dispositivos 802.15.4, a banda dos 2.4 GHz é a mais utilizada, uma vez que é uma banda livre na maioria dos países do mundo. A banda dos 868 MHz é específica para utilizações na Europa, e a banda dos 902-928 MHz só pode ser utilizada nos Estados Unidos, Canadá e outros países que permitam a utilização destas bandas

O *standard* 802.15.4 especifica que a comunicação deve ocorrer em canais de 5 MHz que podem ir deste os 2.405 GHz aos 2.280 GHz. Na banda dos 2.4 GHz o ritmo de transmissão máximo especificado é de 250kbps, com 16 canais disponíveis. No entanto, devido à complexidade acrescida pelos mecanismos de segurança e encriptação dos dados, o ritmo de transmissão é metade do especificado. Por sua vez, nas bandas dos 915 MHz e 868 MHz estão disponíveis taxas de transmissão de 40 Kbps com 10 canais de comunicação e 20 Kbps com um canal de comunicação, respectivamente. Além disso, enquanto o *standard* especifica canais de 5 MHz, apenas aproximadamente 2 MHz de cada canal é que são ocupados. Enquanto que nas bandas dos 868 MHz e 915 MHz se utiliza a modulação BPSK (Binary Phase Shift Keying), na banda dos 2.4 GHz o protocolo IEEE 80215.4 utiliza o O-QPSK (Offset Quadrature Phase Shift Keying) com forma de meia sinusóide para modular a portadora de radiofrequência . No gráfico seguinte podemos visualizar os diversos canais com o devido espaçamento regulamentado pelo IEEE 802.15.4

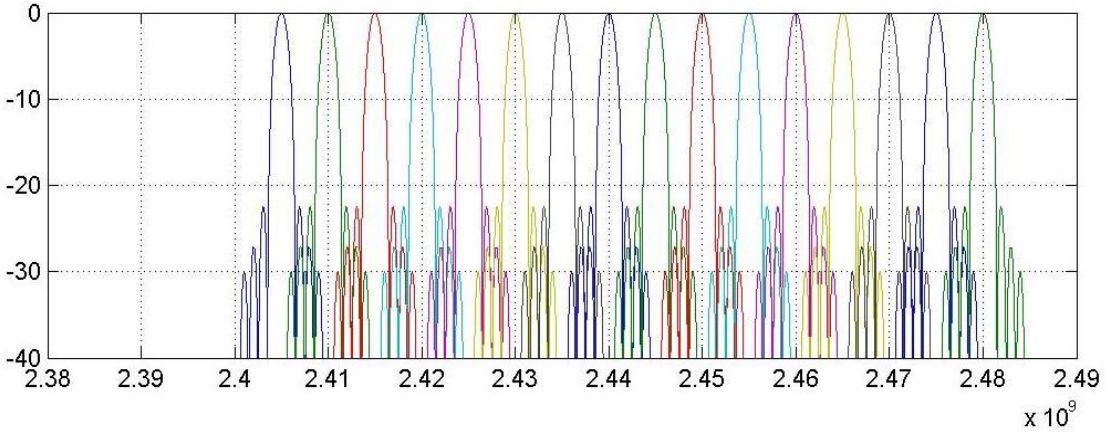

Figura 2 - Espectro de frequências mostrando os diversos canais da banda dos 2.4 GHz

O protocolo 802.15.4 permite que seja estabelecida uma comunicação ponto a ponto ou uma comunicação ponto a multiponto. Uma aplicação típica deste protocolo envolve a existência de um coordenador central ao qual estão ligados diversos nós que comunicam directamente com o coordenador, tal como está exemplificado na figura seguinte:

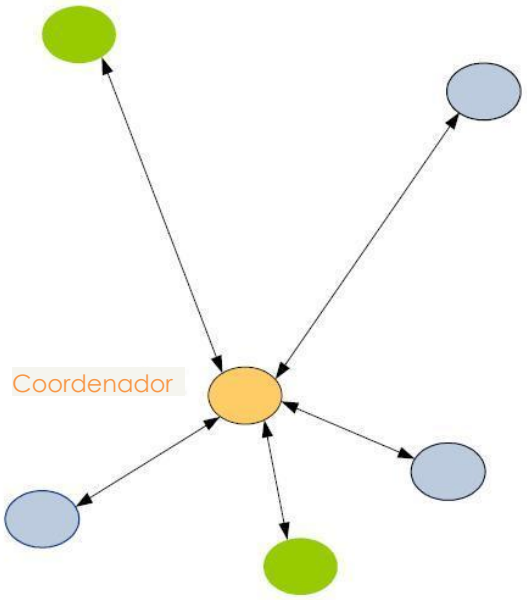

Figura 3 – Aplicação típica do 802.15.4

- ZigBee

O ZigBee é um protocolo que utiliza o standard IEEE 802.15.4 como base e acrescenta uma funcionalidade muito útil: a capacidade de estabelecer redes e de fazer *routing*. O protocolo ZigBee foi desenvolvido pela *ZigBee Aliance*. A *ZigBee Aliance* é um grupo de empresas que trabalharam em conjunto para desenvolver um protocolo para o estabelecimento de redes que pudessem ser utilizadas em diversos ambientes, como por exemplo o comércio e a indústria, em que não se exigem taxas de transmissão elevadas. Deste modo, o ZigBee foi concebido de modo acrescentar a implementação de *mesh networking* ao conjunto de funcionalidades ao IEEE 802.15.4. O tipo de rede em malha (*mesh networking*) é principalmente utilizado em aplicações em que se pretende efectuar a transmissão de dados entre dois nós que não que estão fora do alcance um do outro. Deste modo, os dados são transmitidos para outros nós intermédios que fazem o redireccionamento da informação até que esta chegue ao destinatário.

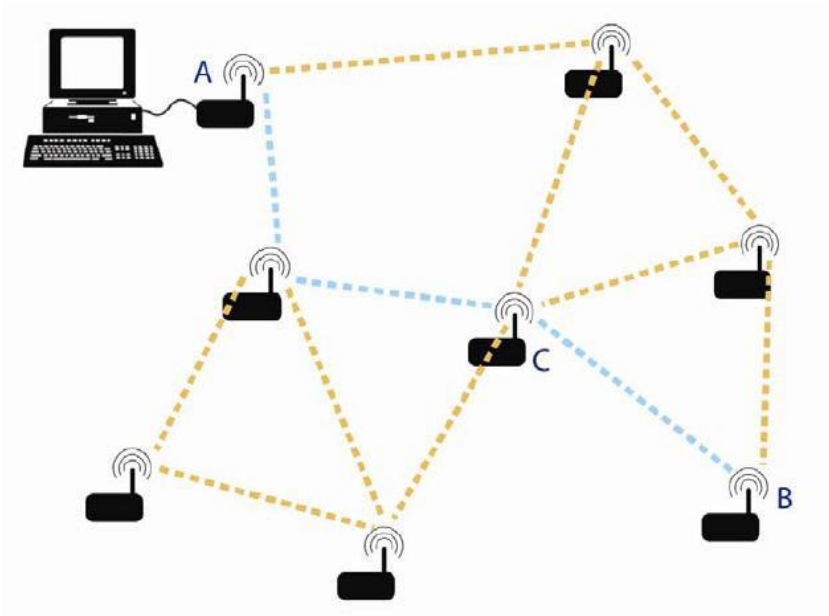

Figura 4 – Exemplo de *mesh networking*

Como exemplo, temos a situação da Figura 3. Supondo que queremos transmitir informação do ponto A para o ponto B mas a distância era grande demais entre os dois pontos a informação poderia ser transmitida passando por outros pontos tais como o ponto C.

O protocolo ZigBee foi concebido de maneira que independentemente da localização e disposição dos pontos de envio e recepção de dados, a rede se formasse automaticamente sem necessitar da intervenção do utilizador na configuração da mesma. Deste modo, o protocolo encarrega-se de todo o processo de reenvio, confirmações de recepção e *routing* das mensagens. No caso de algum dos pontos de envio e recepção de dados for removido ou desligado, uma nova rede será gerada automaticamente. A esta última capacidade do ZigBee podemos chamar de *Self-Healing*. Se o ponto C for removido do sistema por alguma razão, um novo caminho seria utilizado para transmitir dados de A para B.

Qualquer dispositivo que se encontre dentro das normas d ZigBee pode ser utilizado como dispositivos de envio, recepção, ou ambos. Neste último caso, podemos ter simples dispositivos de comunicação bidireccional ou assumir ainda funções de *routing* e coordenação. Uma vez que o protocolo ZigBee utiliza o IEEE 802.15.4 como base para definir as camadas PHY e MAC, a frequência, a largura de banda do sinal e as técnicas de modulação são semelhantes.

Como o ZigBee foi desenhado para ter um consumo muito reduzido de potência, encaixa perfeitamente em aplicações com sistemas embebidos e em todas as áreas em que as principais exigências sejam uma fácil implementação e uma grande versatilidade, em vez de uma grande largura de banda. Na tabela seguinte pode-se ver uma comparação entre o ZigBee e outras tecnologias de comunicação *wireless*, tendo em conta as suas principais aplicações.

	Zigbee e 802.15.4	GSM/ GPRS CDMA	802.11	Bluetooth
Aplicação Principal	**Monitorização de processos e controlo**	Transmissão de dados e voz em grandes áreas	Internet de alta velocidade	Conectividade entre dispositivos
Autonomia	**Anos**	1 semana	1 semana	Semanas
Largura de Banda	**250kbps**	Até 128k	11Mbps	720kbps
Alcance Típico	**mais de 100 m**	Alguns km	50 – 100 m	10 – 100 m
Vantagens	**Baixo consumo de potência e custo reduzido**	Infraestruturas já existentes	Altas velocidades	Versatilidade na ligação entre dispositivos

Tabela 1 – Comparação entre tecnologias de comunicação wireless

A baixa taxa de transmissão dos dispositivos ZigBee dá lugar a uma melhor sensibilidade a alcance, mas implica também um *troughput* mais baixo, ou seja, taxas de transferência mais reduzidas. As principais qualidades do ZigBee são o baixo consumo de potência, a grande autonomia e a possibilidade de estabelecimento de redes segundo várias topologias com grande versatilidade e *self-healing*.

- Topologias e modos de operação das Redes ZigBee

Apesar de já terem sido referidas as topologias de rede que é possível implementar segundo o protocolo ZigBee convém analisar detalhadamente as mesmas.

Uma vez que este *standard* perspectiva a rede de uma forma *ad hoc*, não existe uma topologia pré-definida nem um controlo obrigatoriamente centralizado para a implementação da rede. Deste modo, existem várias topologias possíveis para a rede a ser implementada, havendo, assim, uma configuração da rede de forma dinâmica, sendo esta uma característica muito importante do ZigBee.

No entanto, antes de partir para o estudo das topologias de rede que é possível implementar com o protocolo ZigBee, é conveniente fazer uma breve abordagem sobre os tipos de dispositivos que podem coexistir numa rede ZigBee. Enquanto que no protocolo IEEE 802.15.4 apenas se faz a distinção entre dispositivos FFD e RFD, o protocolo ZigBee faz a distinção entre três tipos de dispositivos lógicos, tal como se pode ver na tabela abaixo. No entanto, há que notar que nos dispositivos ZigBee o tipo de dispositivo não é definido por hardware, mas sim por software, dependendo da configuração da rede em que estão inseridos. A nível de hardware todos os dispositivos ZigBee são iguais.

Dispositivo	Tipo de dispositivo físico associado (IEEE)	Função
Coordinator	FFD	Forma a rede, atribui endereços, faz a manutenção da rede, suporta *binding table*. Existe apenas um por rede, mas pode servir de ponte entre várias redes.
Router	FFD	Permite que mais nós se juntem à rede aumentando o seu alcance físico. Pode também efectuar funções de controlo ou monitorização, para além do reencaminhamento de dados. A sua existência é opcional.
Endpoint	RFD ou FFD	Efectua uma acção de controlo ou monitorização através de dispositivos que lhe esteja associado (sensores, microcontrolador, actuador, etc.). É o que consome menos energia, pois muitas vezes está em modo *sleep*.

Tabela 2 – Tipos de dispositivos

No que diz respeito aos tipos de dispositivos físicos associados presentes na tabela é também conveniente perceber a distinção feita pelo IEEE nos tipos de dispositivos *RFD* e *FFD*:

- Os *FFD* (*Full Function Device*) são dispositivos mais complexos, necessitando assim de um hardware mais potente para a implementação da pilha de protocolos. Uma vez que são mais potentes, têm também maior consumo de energia. Como se viu na tabela, na tecnologia ZigBee estes dispositivos podem ser ZigBee *Coordinators*, ZigBee *Routers*, ou até mesmos um ZigBee Endpoint. Os dispositivos FFD têm a capacidade de comunicar com quaisquer outros membros da rede. A nível de hardware, são implementados em microcontroladores com um mínimo de 32KB de memória de programa e uma determinada quantidade de memória RAM para implementação de tabelas de rotas e configurações de parâmetros.
- Os *RFD* (*Reduced Function Device*) são dispositivos mais simples, utilizando os mínimos recursos de hardware para implementar a sua pilha de protocolo. Podem ser implementados com microcontroladores de 8 bits com uma memória de programa de 6KB. No entanto, só podem comunicar com dispositivos *FFD* (Coordinator ou *Router*). Do ponto de vista de uma rede ZigBee estes só podem assumir o papel de ZigBee *Endpoint*. Na prática, são os sensores, interruptores, controladores de relés, etc.

Como se viu na descrição dos tipos de dispositivos do IEEE 802.15.4, cada tipo de dispositivo tem requisitos diferentes de hardware, o que é uma clara desvantagem comparativamente aos dispositivos ZigBee, pois nestes o tipo de dispositivos é definido por software quando a rede é configurada.

Agora, que já foi feita a distinção entre os tipos de dispositivos que podem coexistir numa rede, serão abordadas sucintamente as diversas topologias de rede, dando atenção aos principais conceitos inerentes a cada uma delas. Contudo, há que realçar que a implementação de uma rede não implica que apenas uma das topologias seja usada. Pelo contrário, dentro de uma rede podem coexistir todas as topologias.

- Topologia em estrela (star)

Nesta topologia de rede o ZigBee *Coordinator* tem toda a responsabilidade no controlo da rede, assumindo assim um papel central e fazendo a comunicação directa com todos os dispositivos Endpoint (dispositivos situados nos limites da rede). Deste modo, o *Coordinator* tem a função de iniciar toda a rede e manter todos os dispositivos associados dentro da rede. Nesta topologia toda a informação transmitida tem que passar pelo nó central, ou seja, o *Coordinator*.

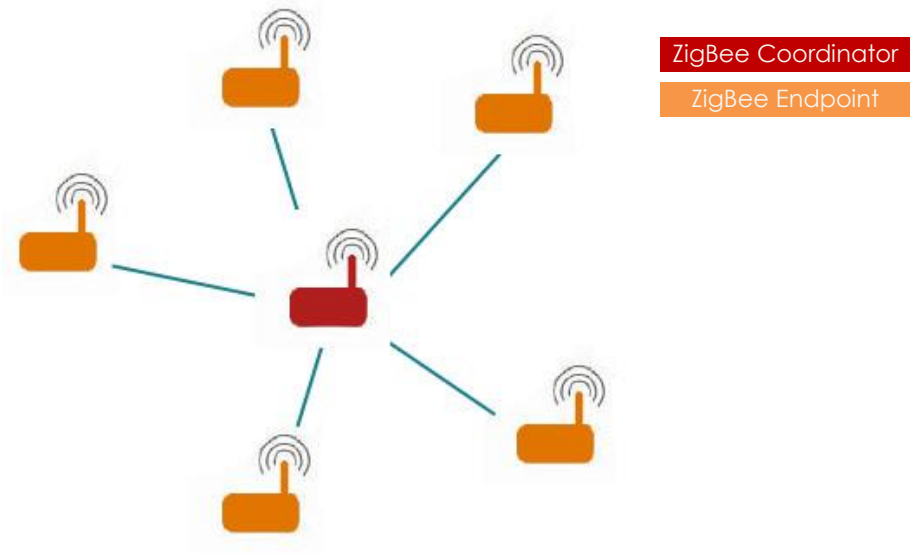

Figura 5 – Topologia em estrela

- Topologia em malha (*mesh*)

Neste tipo de topologia os dispositivos coordenadores ou *routers* são livres de enviar informação para qualquer outro dispositivo da rede, ou seja, não existe uma centralização da rede tão profunda como na topologia em estrela. Neste caso, o *Coordinator* apenas regista a entrada e saída de dispositivos na rede, assumindo um papel passivo no que diz respeito ao fluxo de informação como acontecia na topologia anterior. Esta topologia é muito útil, principalmente porque permite a fácil expansão física da rede, permitindo que seja estabelecida uma rede com grande capaz de abranger uma área relativamente grande. Neste tipo de topologia pode verificar-se o *self-healing* da rede, pois mesmo que um dos dispositivos desapareça, apenas em casos excepcionais a comunicação entre os restantes será afectada.

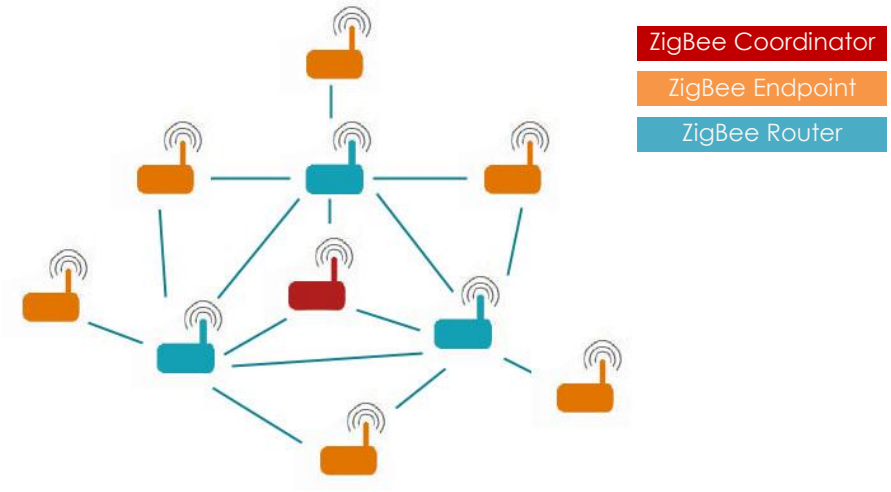

Figura 6 – Topologia em malha

- Topologia em árvore (*cluster tree*)

Esta topologia tem algumas semelhanças com a topologia em malha, sendo também utilizados os dispositivos *Router*. No entanto, nesta topologia é estabelecida um estrutura hierárquica segundo a qual é feita a distribuição de dados e mensagens de controlo. No topo da hierarquia temos o ZigBee Coordinator, que assume mais uma vez o papel de coordenador, sendo o núcleo da rede. Deste modo, do Coordinator surgem diversas ramificações primárias de onde saem ramificações secundárias. No entanto, enquanto as ramificações secundárias podem comunicar entre si passando a informação por um *ZigBee Router*, que encaminha a informação para o destinatário correcto, as ramificações primárias só podem comunicar entre si passando a informação para o *ZigBee Coordinator* que se encarregará de as encaminhar.

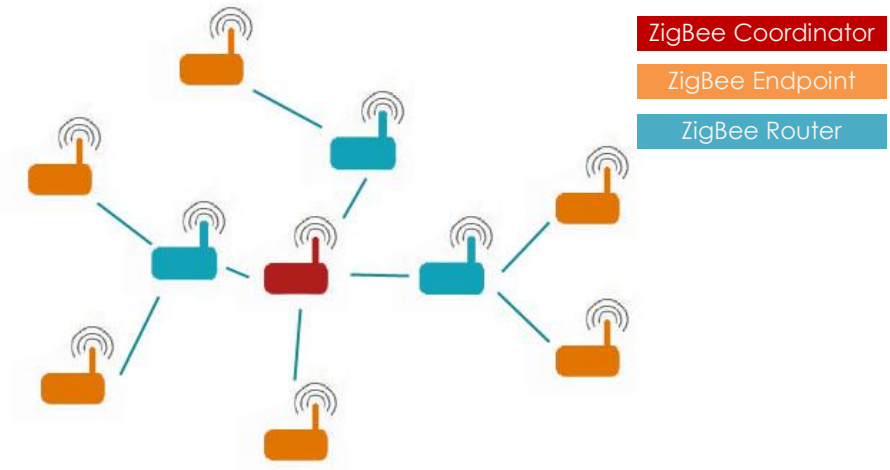

Figura 7 – Topologia em árvore

Como acabamos de ver, o protocolo ZigBee não se limita a uma topologia de rede, permitindo a implementação de várias topologias, dotando os dispositivos ZigBee de uma grande versatilidade. No entanto, a versatilidade não fica por aí. Esta é ainda aumentada devido aos dois modos de operação das redes.

O primeiro modo que abordaremos é o modo *beaconing*. Neste modo os *ZigBee Routers* transmitirão periodicamente mensagens de sinalização, ou seja, *beacons*, informando os outros nós da sua presença, que apenas precisam de estar activos no momento da sinalização. Deste modo, os dispositivos ZigBee podem manter-se no modo *sleep* entre sinalizações, reduzindo bastante o consumo energético. Este modo *sleep* consiste numa redução do *duty cycle* dos dispositivos ZigBee, o que resulta no prolongamento da autonomia da bateria que esteja a alimentar o dispositivo. O intervalo de tempo entre o envio sucessivo de dois *beacons* pode variar entre os 15,36 ms e os 251,65 ms para uma taxa de 250kbit/s. No entanto, há que realçar o facto de a redução do *duty cycle* implicar a existência de uma temporização de elevada precisão, o que poderá vir a colidir com o interesse em produzir um dispositivo de custos reduzidos.

No segundo modo de operação de uma rede ZigBee, ou seja, no modo *non-beaconing* a maioria dos dispositivos mantém os seus receptores permanentemente activos, havendo um maior consumo energético, podendo tornar-se necessário a utilização de fontes de alimentação com maiores capacidades.

- Arquitectura do protocolo

Tal como em outros protocolos já estudados, a arquitectura do protocolo ZigBee é composta por camadas, havendo uma estrutura hierárquica. Cada entidade de serviço fornece uma interface para a camada superior através do ponto de acesso ao serviço (*SAP – Service Access Point*) e cada *SAP* suporta um número de primitivas de serviço para activar a funcionalidade que será solicitada pela camada superior. Apesar de o protocolo ZigBee se basear no modelo *OSI* (*Open Systems Interconnection*)que tem sete camadas, a arquitectura do protocolo ZigBee apenas define as camadas necessárias para atingir um conjunto de funcionalidades desejadas. De uma forma muito sucinta,

as várias camadas da arquitectura protocolar podem ser esquematizadas da seguinte forma:

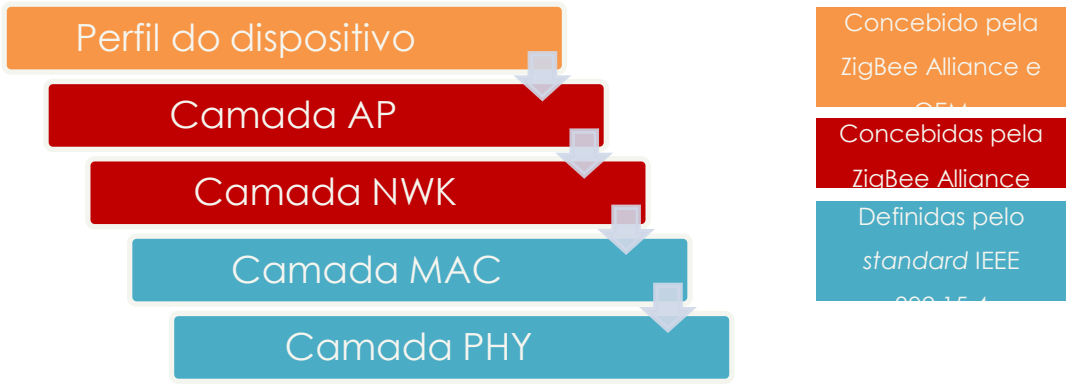

Figura 8 – Camadas da arquitectura do protocolo ZigBee

As duas camadas inferiores, a camada física (*PHY*) e a camada de controlo de acesso ao meio (*MAC*), foram definidas pelas normas do protocolo IEEE 802.15.4, pois, como já tinha sido referido, o protocolo ZigBee foi concebido a partir do protocolo do IEEE. As restantes camadas de rede foram concebidas especificamente para o protocolo ZigBee. Tais camadas são a camada de rede (*NWK*) e o *Framework* para a camada de aplicação (*AP*). Nesta camada estão incluídas a subcamada de suporte aplicacional (*APS*), o objecto de dispositivo ZigBee (*ZDO - ZigBee Device Object*) e os objectos de aplicação (*Aplication Objects*).

A **camada *PHY (IEEE 802.15.4)*** é responsável pelo controlo da transmissão e da recepção de mensagens através de um canal físico RF cujas características já foram referidas anteriormente. Algumas das suas funções são a activação e a desactivação do *transciever*, a detecção de energia (*ED – Receiver Energy Detection*), a indicação da qualidade da ligação (*LQI – Link Quality Indication*), a selecção do canal e ainda a transmissão e recepção de pacotes através do meio físico. Esta camada fornece dois serviços: o *PHY data service* e o *PHY management service* que faz o interface com o ponto de acesso(SAP – *Service Access Point*) da entidade de gestão da camada física (PLME – *Physical Layer Management Entity*) ste último serviço pode também ser designado de PLME-SAP, pela junção das duas siglas. O PHY *data service* activa a transmissão e a recepção das unidades de dados do protocolo PHY através do canal físico de radiofrequência. Deste modo, as principais funções da camada PHY são a activação e desactivação dos *transcievers*, detecção de energia, verificação

da qualidade do *link*, selecção de canais, acesso a canais livres, transmitir e receber pacotes de dados através do meio físico.

Sendo assim, a camada PHY cria um interface entre a camada MAC e o canal de rádio físico através do firmware e hardware de radiofrequência. A PLME, que forncece o serviço de gestão da camada através do qual as funções de gestão da camada podem ser invocadas. O PLME é também responsável por manter uma base de dados (PIB – *PHY Pan Information Base*)de objectos geridos pertencentes à camada PHY. No que diz respeito ao serviço de dados da camada PHY, este é responsável pela transferência de MPDUs (*MAC protocol data unit*) entre as várias entidades da camada MAC.

No que diz respeito à **camada MAC (IEEE 802.15.4)**, esta tem a principal função de controlar o acesso aos canais RF, utilizando para isso mecanismos de prevenção de colisão *CSMA-CA* (*Carrier Sense Multiple Access – Collision Avoidance*). Para efectuar esse controlo estabelece uma comunicação com a camada inferior, ou seja, com a camada física (*PHY*). Além disso, define também o tipo de dispositivos permitidos na rede e a estrutura de tramas admissível e faz o controlo do processo de sinalização, ou seja, faz a sincronização e transmissão de *beacons*, de modo a haver alguma fiabilidade no funcionamento da rede.

Esta camada, à semelhança da camada PHY, também fornece 2 serviços: o MAC *data service* e o MAC *management service* que faz o interface com o ponto de acesso(SAP – *Service Access Point*) da entidade de gestão da camada MAC (MLME – *MAC sublayer Management Entity*). Este último serviço pode também ser designado de MLME-SAP, pela junção das duas siglas. O serviço de dados da camada MAC activa a transmissão e recepção de unidades de dados do protocolo MAC através do serviço de dados da camada PHY. Deste modo, as principais funções da camada MAC são a gestão de *beacons*, acesso aos canais utilizando CSMA-CA, gestão do GTS (*Guaranteed Time Slot*), validação de tramas, envio de tramas de *Acknowledgement*, associação e desassociação PAN, segurança de dispositivos duportados e criação de *links* entre duas entidades MAC. Adicionalmente a estas funções, a camada MAC fornece ainda uma base para implementar mecanismos de segurança apropriados às aplicações.

O serviço de gestão incluí a entidade MLME que fornece interfaces através dos quais as funções do serviço de gestão da camada podem ser invocadas. À semelhança do PLME, o MLME também é responsável por manter uma base de dados dos objectos geridos que pertençam à camada MAC. Esta base de dados também tem o nome de PIB. No que diz respeito ao serviço de dados, este suporta o transporte de unidades de dados do protocolo SSCS (Service-Specific Convergence Sublayer) entre entidades SSCS.

Quanto à **camada NWK (ZigBee)**, que é, hierarquicamente, a primeira camada definida pela norma ZigBee, tem como responsabilidade a descoberta de novos dispositivos que possam passar a integrar a rede, armazenando as informações relativas aos mesmos, a atribuição dos endereços aos dispositivos membros da rede (apenas dos ZigBee *Coordinators*) e a monitorização das entradas e saídas de dispositivos da rede. É através desta camada que é feita a configuração de novos dispositivos e nela estão também definidos os mecanismos de descoberta de rotas e encaminhamento de informação (*routing*).

Esta camada de rede é necessária para fornecer funcionalidades que garantam o correcto funcionamento do MAC do IEEE 802.15.4 e também para fornecer um serviço adequado para fazer o interface com a camada de aplicação. Para interagir com a camada de aplicação a camada de rede contém na sua definição dois serviços que fornecem as funcionalidades necessárias. Estas entidades são o serviço de dados e o serviço de gestão. Tal como nas duas camadas anteriores o serviço de gestão contém uma entidade (NLME – Network Layer Management Entity) que através do seu SAP fornece os serviços necessários. Deste modo, o NLME é responsável por criar recursos que permitam à aplicação interagir com a pilha. Outros serviços próprios do NLME são:

- configuração de novos dispositivos – a capacidade de configurar a pilha de operação é necessária. As opções de configuração incluem o início de operação como um ZigBee *Coordinator* ou o processo de integrar uma rede já existente;

- criação de uma rede – estabelececimento de uma rede;

- juntar-se ou abandonar uma rede – é a capacidade de se juntar ou deixar uma rede e também a capacidade de um ZigBee *Coordinator* ou ZigBee *Router* requerer a um dispositivo que abandone a rede;

- Endereçamento – é capacidade dos ZigBee *Coordinators* e *Routers* atribuírem endereços a dispositivos que se juntem à rede;

- Descoberta da vizinhança – descoberta, registo e comunicação de informações relativas aos "vizinhos" directos de um dispositivo;

- Descoberta de rotas – descoberta e registo de rotas através da rede através das quais as mensagens podem ser devidamente redireccionadas.

No que diz respeito ao dos serviços de dados contém uma entidade (NLDE – *Network Layer Data Entity*) que fornece o serviço de transmissão de dados através do seu *Service Access Point* (SAP). O NLME utiliza o NLDE para conseguir levar a cabo algumas das suas tarefas de gestão e também mantém uma base de dados dos objectos geridos, conhecidos como NIBs (*Network Information Base*). O NLDE fornece um serviço de dados que permita a uma aplicação transportar APDUs (*Application Protocol Data Units*) entre dois ou mais dispositivos que façam parte da mesma rede. Deste modo, o NLDE fornece os seguintes serviços:

- Criação de NPDUs (*Network Protocol Data Units*) – o NLDE consegue gerar uma NPDU a partir de uma PDU da camada de Aplicação através da adição de um cabeçalho apropriado do protocolo;

- Topologia específica de Routing – o NLDE deve ser capaz de transmitir uma NPDU para o devido dispositivo quer este seja o destinatário final da comunicação ou o próximo passo para chegar ao destinatário final;

- Segurança – é a capacidade de garantir a autenticidade e confidencialidade de uma transmissão.

No que diz respeito à **camada de aplicação (ZigBee)**, como já foi referido, esta contém a sub-camada *Application Support Sublayer* (*APS*), o ZigBee *Device Object* (*ZDO*) e a *Application Framework* (*AF*). Esta camada tem a função de garantir uma gestão correcta e um suporte fiável para as diversas aplicações.

A APS fornece um interface entre a camada NWK e a camada de aplicação através de conjunto geral de serviços que são usados pela ZDO e pelas aplicações definidas pelo fabricante. Os serviços desta camada são fornecidos pelas seguintes entidades:

- APS *data entity* (APSDE), através do *Application Service Data Entity Access Point* (APSDE-SAP). Esta entidade torna possível a transmissão de dados para o transporte de PDUs de aplicação entre dois ou mais dispositivos localizados na mesma rede incluindo uma filtragem das mensagens endereçadas ao grupo. A APSDE suporta ainda a fragmentação e reconstrução dos pacotes maiores que o *payload* suportado pelas *Application Service Data Units* e garante ainda um transporte de dados viável.

- APS *management entity* através do Application Service Management Entity Access Point (APSME-SAP). Esta entidade fornece serviços de segurança, registo e remoção de endereços de grupo e ainda mantém uma base de dados dos dispositivos geridos que tem o nome de APS *information base* (AIB). O AIB suporta o mapeamento de endereçamentos entre endereços IEEE de 64 bits e endereços NWK de 16 bits.

A *Application Framework* é um ambiente em que os objectos de aplicação estão guardados em dispositivos ZigBee. Dentro desta *Framework* os objectos de aplicação enviam e recebem dados através do APSDE-SAP, realizando funções de controlo e manutenção das camadas de protocolo do dispositivo ZigBee e inicialização de funções de rede *standard*. O serviço de dados utilizado por estes objectos incluí funções de pedido, confirmação, resposta e primitivas de indicação para a transferência de dados, que são utilizadas para indicar a transferência de dados da APS para a aplicação ou entidade de destino. As funções de pedido suportam transferências de dados entre aplicações de entidades de objectos.

Os ZigBee *Device Objects* representam uma base de funcionalidades que fornece um interface entre os objectos de aplicação, o perfil do dispositivo e a *Application Support Sublayer*. Os ZDO situam-se entre a *Application Framework* e a *Application Support Sublayer*. Estes objectos têm o objective de satisfazer os

requisites de todas as aplicações que estejam a ser executadas na pilha de protocolo do ZigBee. Os ZDO são responsáveis por inicializar a APS, a camada NWK e o serviço de segurança e juntar informações de configuração das aplicações finais para determinar e implementar a descoberta e gestão de segurança e rede. Estes objectos têm interfaces publicas com os objectos da *Application Framework* para que estes possam fazer o controlo de funções de dispositivo e de rede.

Para uma análise geral do que foi descrito acima relativamente ao protocolo ZigBee pode-se observar o seguinte esquema:

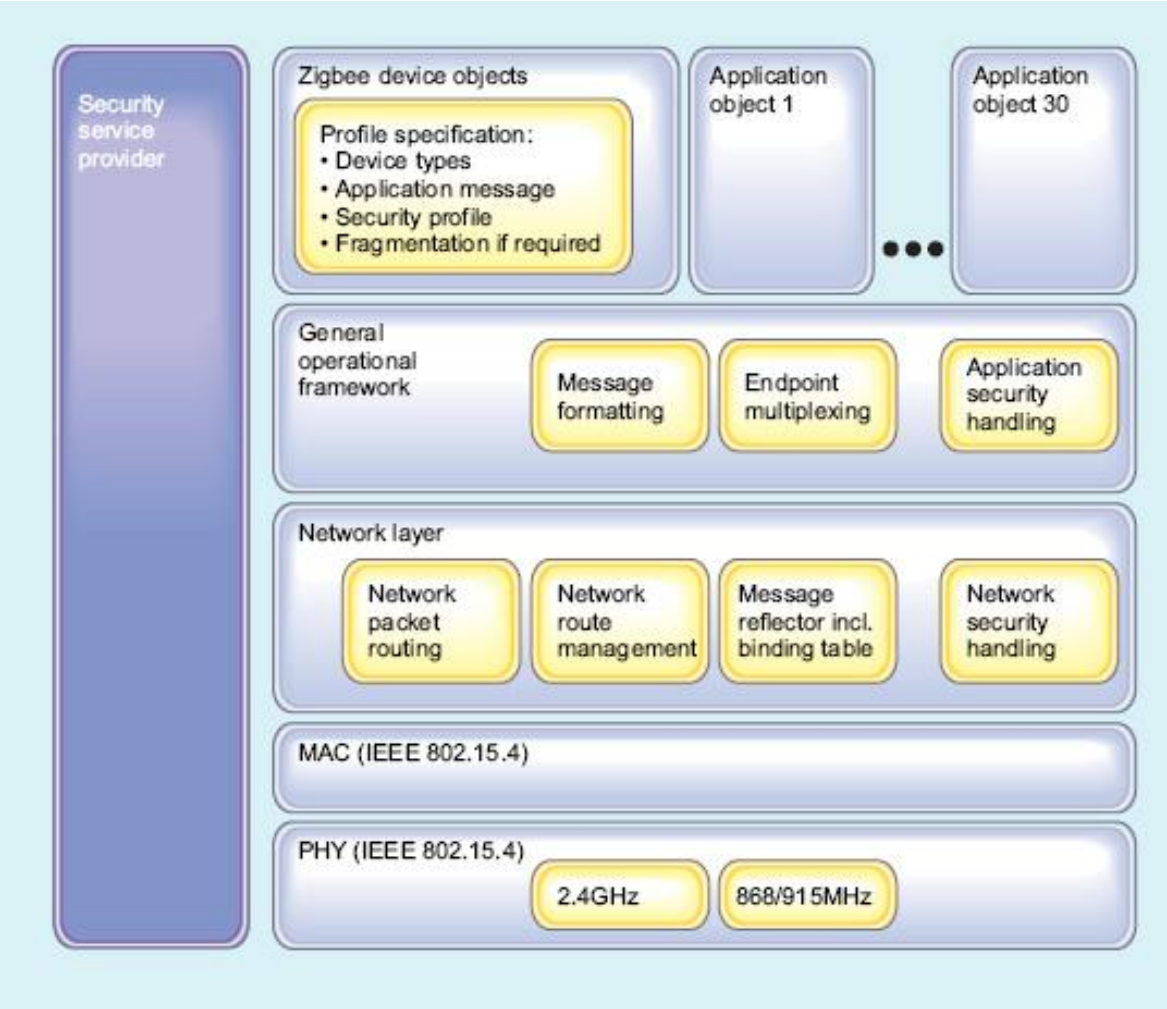

Figura 9 – Camadas detalhadas da arquitectura do protocolo ZigBee

Na figura 9 temos a arquitectura do protocolo ZigBee, correspondendo cada linha bloco horizontal a uma camada, excepto nos dois blocos superiores, que fazem ambos parte da camada de Aplicação. Na camada PHY temos as

três bandas utilizadas pelo ZigBee. De seguida temos a camada MAC, que faz a interface entre a camada de rede e a camada física. Como já tinha sido referido, as duas camadas anteriores são originárias do protocolo IEEE 802.15.4. A terceira camada é a camada de Rede, que se encarrega do *routing* de pacotes de rede, da gestão e segurança da rede e contém ainda as tabelas de endereços e outros dispositivos, necessários para efectuar o *routing*. De seguida encontramos a camada de aplicação, que na figura está dividida em três partes: a *Application Framework*, o ZigBee *Device* Object, e um conjunto de *Application* Objects. A *Application* Framework é responsável pela formatação das mensagens, multiplexagem dos ZigBee *Endpoint*, e também pela segurança das aplicações. Dentro da camada de aplicação temos ainda o ZigBee *Device Object* que é responsável pela gestão e manutenção das especificações do perfil de funcionamento do dispositivo. Dentro destas especificações temos os tipos de dispositivos, as mensagens de aplicação, o tipo de segurança implementado e também a fragmentação de mensagens, se os dispositivos forem configurados para tal. Por fim, dentro da camada de aplicação temos até 30 objectos de aplicação. Todas estas camadas encontram-se em paralelo com um serviço: o serviço de segurança do ZigBee.

Após esta análise da arquitectura protocolar do ZigBee é conveniente fazer também um estudo dos tipos de tramas utilizadas no protocolo:

- Tramas de comando MAC (*MAC command*) – estas tramas são utilizadas para efectuar o controlo dos nós clientes.
- Tramas de dados – estas tramas são as que mais interessam ao utilizador, pois são usadas para todo o tipo de transferência de dados. Podem ter até 104 bytes e estão numeradas, afim de manter alguma fiabilidade na comunicação, pois a existência de uma sequência de *frame-check* permite assegurar uma transmissão fiável e sem erros.
- Tramas de *acknowlegement* (*ACK*) –são utilizadas para confirmar a recepção bem sucedida de pacotes. O *acknowledgement* e feito no tempo livre de comnicações existente entre o envio de tramas.

- Tramas de *beacon* – são utilizadas pelos ZigBee *Coordinator* e *Router* para efectuar a transmissão de *beacons*.

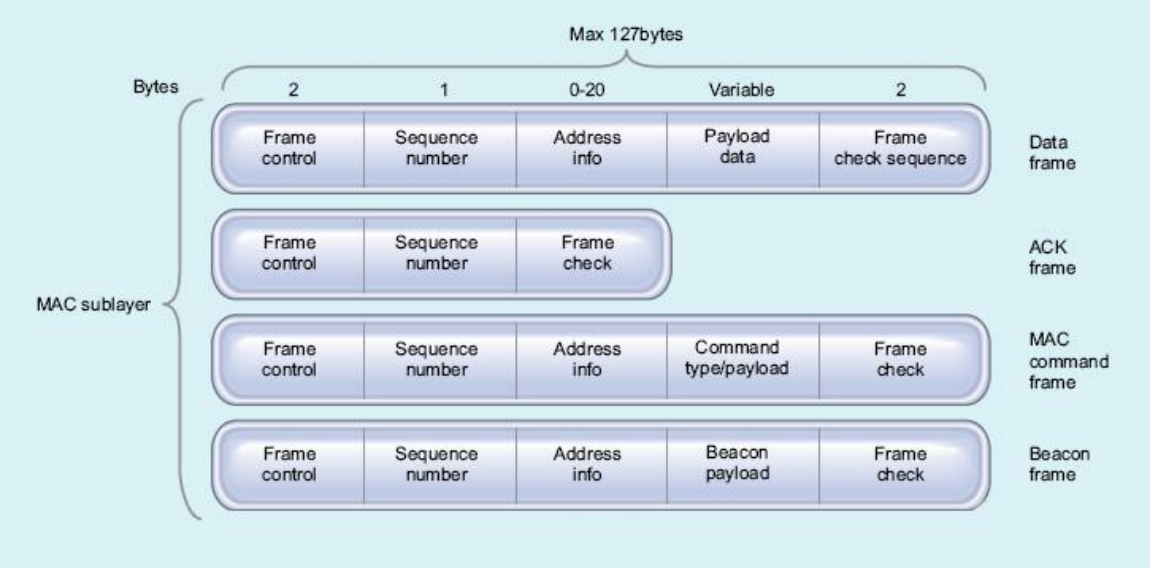

Figura 10 – Tramas de comando MAC

- Nota Conclusiva

Após a análise dos dois protocolos e depois de compreendidas as suas diferenças podemos concluir que a utilização do IEEE 802.15.4 ou do ZigBee depende do que se pretende fazer. No caso de se querer fazer uma comunicação ponto a ponto ou ponto a multiponto, o IEEE 802.15.4 conseguirá dar conta do recado, estabelecendo sem problemas a comunicação entre os dispositivos e será mais fácil de implementar do que utilizando o ZigBee para atingir o mesmo objectivo. Porém, se necessitarmos de implementar uma *mesh network* no sistema, o ZigBee é, sem qualquer dúvida, a escolha mais acertada. Após esta análise, verificamos que o IEEE 802.15.4 não são a mesma coisa, sendo o ZigBee uma "evolução" do standard IEEE 802.15.4.

XBee

- O que são?

Os módulos XBee, fabricados pela Maxstream, são módulos que incluem todo o hardware e a lógica necessária para implementar uma rede ZigBee. Pode-se dizer que um módulo Xbee está para o ZigBee tal como uma placa de rede está para uma rede Ethernet.

A Maxstream fabrica duas versões do XBee -XBee e Xbee-Pro- que se diferenciam sobretudo na potência de emissão e sensibilidade de recepção, logo no alcance máximo. Para além disto, ambas as versões estão disponíveis com três tipos de antena: chicote (whip), chip, e conector para antena externa.

Ambos os modelos dispõem de 16 canais seleccionáveis via software, com suporte até 65.000 endereços por canal e encriptação 128-bit AES.

De notar que embora o XBee-Pro tenha uma potência máxima de saída de 60mW, na Europa é necessário limita-lo a 10mW, de modo a não infringir a legislação em vigor.

Este tipo de configuração é possível através do software X-CTU da Maxstream.

Configurados de fábrica para a operação em modo broadcast, os módulos permitem a utilização sem necessidade de configuração prévia.

- Características

	XBee	XBee-Pro
Potência de saída	1 mW (0 dBm)	60 mW (18 dBm)
Alcance interior	até 30 m	até 100 m
Alcance Exterior	até 100 m	até 1600 m
Sensibilidade do receptor	-92 dBm	-100 dBm
Frequência de operação	ISM 2.4000 a 2.4835 GHz	
Taxa de transmissão	250 kbps	
Taxa de dados da Interface	115.2 kbps	
Tensão de alimentação	2.8V a 3.4V	
Corrente de transmissão (típica)	45 mA @ 3.3 V	215 mA @ 3.3 V
Corrente de Recepção (típica)	50 mA @ 3.3 V;	55 mA @ 3.3 V
Corrente em modo *Sleep*	<10 µA	
Dimensões	2.438cm x 2.761cm	2.438 cm x 3.294 cm
Peso	3 g	4 g
Temperatura de operação	-40 a 85º C (industrial)	

Tabela 3 – Características dos módulos XBee

Aplicação demonstrativa

- Objectivos

Como exemplo de implementação do XBee segue-se uma explicação detalhada de uma possível aplicação desenvolvida para fins demonstrativos e didácticos. Para compreender este exemplo na totalidade é necessário ter alguns conhecimentos prévios de programação de microprocessadores, electrónica digital e linguagem C. O objectivo é proporcionar ao leitor uma aprendizagem pela prática de como implementar um sistema com ZigBee utilizando os módulos XBee.

- Descrição da aplicação

Para efeitos de teste e demonstração dos módulos XBee optou-se por implementar uma aplicação de controlo de acessos a um parque de estacionamento a pessoal autorizado. Este sistema seria semelhante à conhecida Via Verde, ou seja, o condutor do veículo não teria que carregar em qualquer botão para abrir a cancela.

Para permitir o controlo e registo dos acessos ao parque de estacionamento cada veículo teria um número de identificação correspondente ao número de funcionário que seria enviado juntamente com um código de resposta para a cancela quando o dispositivo existente no veículo recebesse o código da cancela, indicando a sua proximidade. No caso desse número de identificação e resposta ao código serem válidos, a cancela abre, deixando passar o carro. Caso contrário, permaneceria fechada. Para evitar o "sniffing" de códigos da cancela, esta possui vários códigos diferentes que são enviados alternadamente. (neste caso, para fins demonstrativos, tem dois códigos). Contudo, para garantir maior segurança é possível definir chaves de encriptação no XBee, como veremos mais adiante.

A aplicação mencionada é constituída por dois circuitos designados por *base* e *móvel*, ambos baseados num microcontrolador 8051 (DS89C450),

dispondo ambos de um módulo XBee para a comunicação e de 4 leds amarelo (*LED0*:*LED3*) para sinalização de estado. O circuito *base* dispõe ainda de um led verde e um vermelho de modo sinalizar a abertura da cancela do estacionamento.

O circuito Base estaria integrado no circuito de controlo da cancela do parque de estacionamento enquanto que o circuito móvel estaria a bordo de um veículo.

Estabeleceu-se que a aplicação deveria seguir o seguinte procedimento:

1. *Base* envia periodicamente um byte, activando *LED0* após o envio;
2. Caso *móvel* esteja ao alcance deverá:
    1. Ligar *LED0* para indicar a recepção do byte enviado por *base*.
    2. Enviar o primeiro dos seus dois bytes de identificação (*IDhigh*) e seguidamente ligar *LED1*;
    3. Enviar o segundo dos seus dois bytes de identificação (*IDlow*) e seguidamente ligar *LED2*;
    4. Enviar um terceiro byte obtido a partir da operação: [*IDlow*] XOR [*IDhigh*] XOR [byte recebido].
3. *Base* deverá receber os três bytes enviados por *móvel* activando *LED1* após a recepção de *IDhigh*, *LED2* após a recepção de *IDlow*, e *LED3* após a recepção do byte de verificação.
4. *Base* deverá verificar se IDhigh e IDlow se encontram na sua lista de ID's com autorização de acesso e se o terceiro byte recebido é de facto válido.

    Verificando-se estas condições *Base* deverá desligar o led vermelho, ligar o led verde, esperar durante algum tempo para permitir a passagem do veículo e de seguida voltar a comutar o led verde e o vermelho.

De seguida será descrito todo o hardware e software utilizados nesta aplicação, de modo a fornecer toda a informação necessária para recriar esta aplicação com fins didácticos.

- Hardware e software

- O circuito Base

Como mencionado anteriormente, concebeu-se o circuito *base* em torno de um microcontrolador 8051, mais especificamente um DS89C450 da Dallas Maxim Semiconductor.

Em primeiro lugar atendeu-se à necessidade de incluir no circuito um cristal e respectivos condensadores para gerar o sinal de relógio do microcontrolador. Optou-se por recorrer a um cristal de 11,0592 MHz e a dois condensadores cerâmicos de 30 pF (C1, C2). Incluiu-se também no circuito um botão de pressão para permitir efectuar o reset do microcontrolador.

Dado o microcontrolador utilizado operar a 5V e o módulo XBee não suportar tensões superiores 3,3V recorreu-se novamente a um divisor de tensão para proteger o módulo XBee. Tal como anteriormente, os valores escolhidos para as resistências do divisor de tensão -R8 e R9- foram 3,3 kΩ e 1,8 kΩ, respectivamente.

Neste circuito estabeleceu-se uma comunicação série recorrendo apenas às linhas Din e Dout do módulo XBee, ligadas aos pins TXD e RXD do microcontrolador, respectivamente, e tendo apenas a primeira destas linhas um divisor de tensão.

Tal como nos circuitos anteriores, recorreu-se a fontes de tensão externas para a alimentação do circuito com as tensões de 5V e 3,3V. No entanto, no caso de só se dispor de uma fonte pode utilizar-se um circuito bastante simples utilizando um regulador de tensão variável como o LM317 e duas resistências, tal como é explicado no *datasheet* do LM317.

Na página seguinte, na figura 11 pode-se ver o esquemático do circuito completo. Na aplicação completa poderia ainda juntar-se uma conexão para a abertura e fecho da cancela.

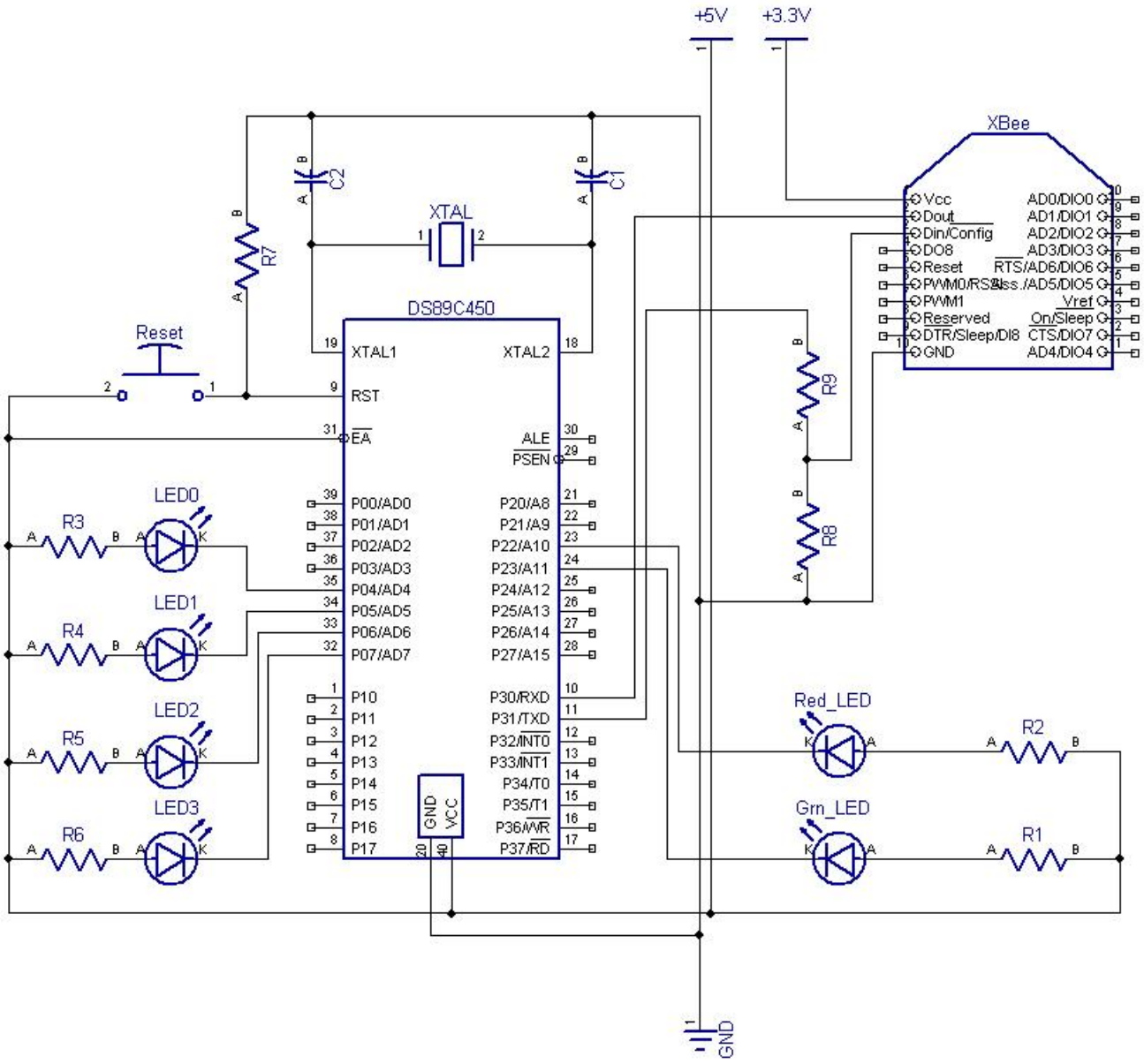

Figura 11 – Circuito da Base

Lista de Componentes:

R1 a R7	- 1,0 kΩ +/- 5%
R8	- 3,3 kΩ +/- 5%
R9	- 1,8 kΩ +/- 5%
C1, C2	- 30 pF
XTAL	- 11,0592 MHz

- O circuito Móvel

Este circuito é em quase tudo idêntico ao circuito *Base*, apresentado anteriormente. A única diferença é o facto de este circuito não necessitar nem do led verde, nem do vermelho.

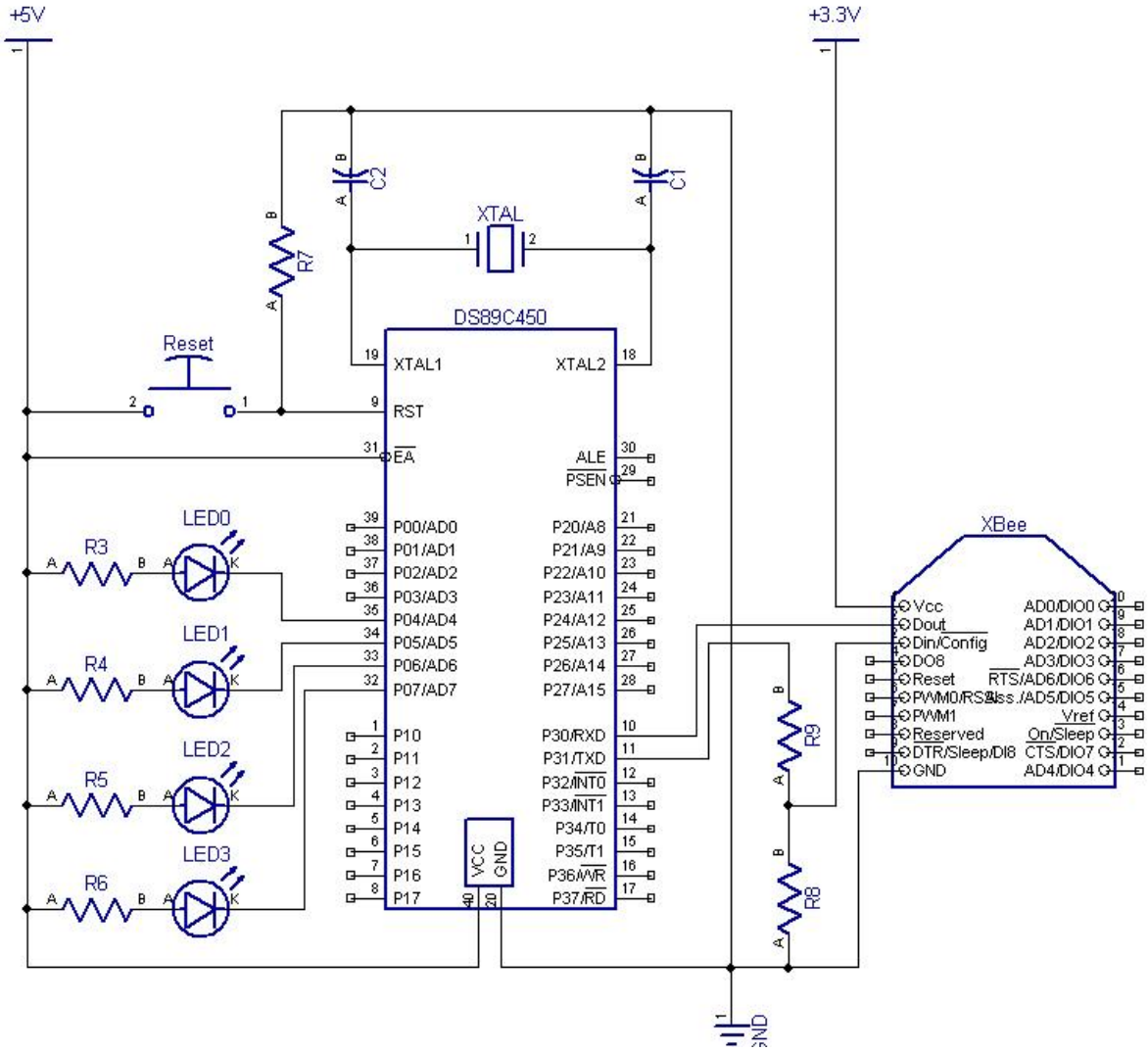

Figura 11 – Circuito da Base

Lista de Componentes:

R1 a R7	- 1,0 kΩ +/- 5%
R8	- 3,3 kΩ +/- 5%
R9	- 1,8 kΩ +/- 5%
C1, C2	- 30 pF
XTAL	- 11,0592 MHz

- O software

A configuração dos módulos XBee via porta série é efectuada mediante o uso de comandos AT enviados por meio de um programa de emulação de terminal. No entanto, por esta forma de configuração ser muito demorada e algo penosa, optou-se por recorrer ao software X-CTU fornecido pela própria MaxStream. Este software disponibiliza em ambiente gráfico, de maneira que facilmente se alteram todas as opções de configuração do módulo XBee.

Para além de permitir a configuração dos módulos, o software X-CTU permite também efectuar o update do firmware e diversos testes, como por exemplo um teste de alcance, uma vez que permite ver a potência do sinal em tempo real. Mais adiante veremos mais detalhadamente como se faz esta configuração.

De seguida veremos o código do programa do microcontrolador da Base. Para fazer a compilação e *debugging* do programa utilizou-se o software Keil µvision da Keil Software. Para programar os microcontroladores utilizou-se o software Microcontroller Tollkit, disponível na secção de downloads do site da Dallas Maxim Semiconductor. Para programar os microcontroladores foi necessário implementar o circuito da figura 12. Há que notar que este circuito é apenas programador, não permitindo *In-Circuit Programming*.

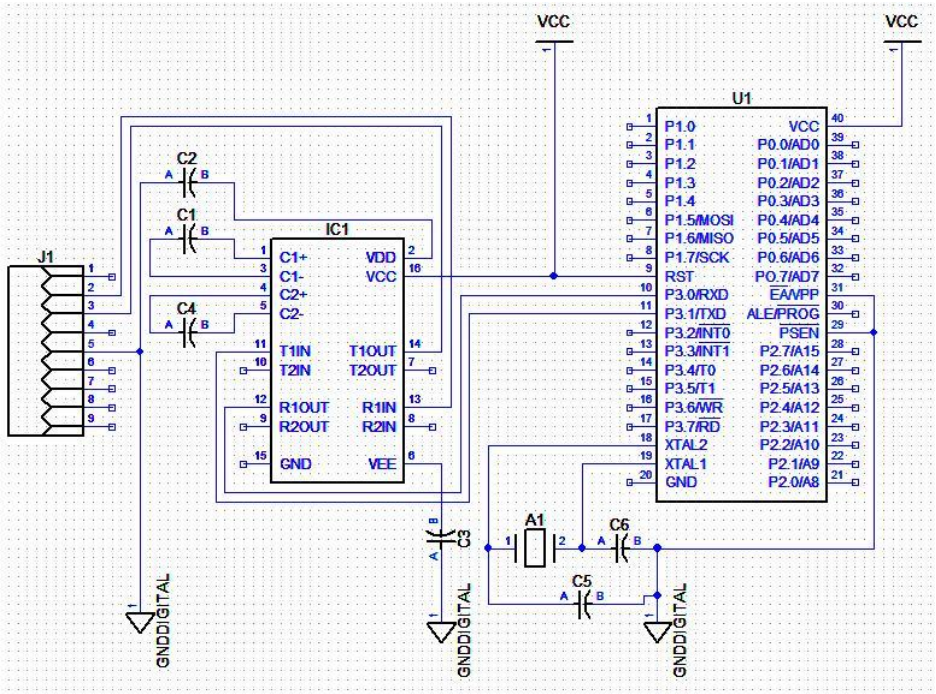

Figura 12 – Programador dos DS89C450

Lista de Componentes:

C1 a C4 - 2,2 µF

IC1 - MAX232

U1 - DS89C450 (DS89C430 também serve)

C5, C6 - 22 pF

A1(XTAL) - 11,0592 MHz

J1 - Conector DSUB9 fémea

Nota: Entre oVcc e o Vdd do IC1 pode ainda levar um condensador de 10 µF para estabilizar a tensão de alimentação

De seguida pode-se ver o programa implementado no microcontrolador da Base:

```
#include <DS89C4xx.h>

//Declaração de variáveis:
bit FlagEstadoStandby;       //flag indicadora de
                             //estado do programa
bit FlagEstadoValidate;      //          "
unsigned char authIDh[32]={"aaaabbbbccccdddd"};//vector que
                                               //contêm a lista
                                               //de bytes
                                               //superiores com
                                               //permissão de
                                               //entrada no
                                               //parque
unsigned char authIDl[32]={"abcdabcdabcdabcd"};//vector que
                                               //contêm a lista
                                               //de bytes
                                               //inferiores com
                                               //permissão de
                                               //entrada no
                                               //parque
unsigned char crc;           //byte ser enviado em broadcast
unsigned char crctest;       //variável para conter o ultimo byte
enviado pela base
unsigned char counter;       //variável de contagem
sbit led0 = P0^4;            //led de sinalização de estado
sbit led1 = P0^5;            //          "
sbit led2 = P0^6;            //          "
sbit led3 = P0^7;            //          "
unsigned char IDh;           //variável para conter o
                             //primeiro byte recebido
unsigned char IDl;           //variável para conter o segundo byte
                             //recebido
unsigned char crcanswer;     //variável para conter o terceiro
                             //byte recebido
sbit greenled = P2^3;        //led de sinalização de estado
sbit redled = P2^2;          //          "
int a;
int b;
```


```
//Função de inicialização:
init_all()
{
      TMOD = 0x21;          //configurar timer 1 em modo 8b-
                            //autorecarga, timer 0 em modo 16b
      TH0 = 0xdb;           //valor de carga do timer 0
      TL0 = 0xff;           //    "
      TR0 = 1;              //activação do timer 0
      SCON0 = 0x50;         //porta série no modo UART 8b,
                            //recepção habilitada
      PCON = PCON|0x80;     //activação flag GF1 -flag de uso
                            //geral
      REN_0 = 1;            //habilitar recepção na porta série0
      TH1 = 0xfa;           //valor de carga do timer 1
      TL1 = 0x00;           //valor de recarga do timer 1
      TR1 = 1 ;             //activação do timer 0
      RI_0 = 0;             //colocar flag de recepção de dados
                            //na porta série a zero
      TI_0 = 0;             //colocar flag de transmissão de
                            //dados na porta série a zero
      IE = 0x92;            //habilitação das interrupções de
                            //porta série e timer 0, bem como
                            //habilitação global de IRQs
      IP0 = 0x00;           //manter as prioridades das
                            //interrupções inalteradas
      FlagEstadoStandby = 1;
      FlagEstadoValidate = 0;
      crc=0x55;             //definir proximo crc a enviar
      IDh = '0';            //inicialização a zero de variáveis
      IDl = '0';            //    "
      crctest = '0';        //    "
      crcanswer = '0';      //    "
      counter = 150;        //contador para efectuar o timeout
                            //durante a recepção de dados
}

//Função para criar ciclos de espera temporizados:
void delay(int tempo)
{
     int i;
     for(i=0;i<tempo;i++)   //fazer n contagens de 10 ms
     {
          while(TF0!=1){}   //esperar até ao overflow do timer 0
          TF0 = 0;          //reset à flag de overflow do timer 0
          TH0 = 0xdb;       //colocar timer o a 0xdbff para uma
                            //temporização de 10 ms
          TL0 = 0xff;
     }
}

//Função para verificar se IDhigh e IDlow recebidos correspondem
//a um veículo com autorização:
int exists(void)
{
     int i = 0;
     for(i=0;i<10;i++){
          if(IDh==authIDh[i]&&IDl==authIDl[i])
          {
               return(1); //se o veiculo pertence à lista,
```

```
                              //devolver "1"
            }
      }
      return(0);              //se o veiculo não pertence à lista,
                              //devolver "0"
}

//Função para enviar o byte de crc:
enviacrc()
{
      SBUF0=crc;        //mover byte de crc para o buffer de saída
      crctest = crc;    //guardar o valor do ultimo byte de crc
                        //enviado para posterior processamento de
                        //bytes de resposta
      while(TI_0!=1){} //aguardar que seja terminado o envio
      TI_0 = 0;         //colocar a zero a flag de fim de envio
      crc=~crc;         //complementar o byte de crc
}


//Função de temporização para o timeout da recepção:
void Timer10ms(void)interrupt 1
{
      if(counter!=0)
      {
            counter = counter - 1; //enquanto counter diferente de
                                   //0, decrementar counter
      }
      else if(counter == 0){}      //se counter=0, não fazer nada
      TH0 = 0xdb;                  //valor de carga do timer 0
                                   //para uma temporização de 10
                                   //ms
      TL0 = 0xff;                  //    "
}

//Programa principal:
void main(void)
{
      init_all();//chamar função de inicialização
      ET0 = 0;    //desactivar IRQ do timer 0
      ES0 = 0;    //desactivar IRQ da porta série 0
      while(1)
      {
            while(FlagEstadoStandby)
            {
                  led0 = 1;   //desligar led0
                  led1 = 1;   //desligar led1
                  led2 = 1;   //desligar led2
                  led3 = 1;   //desligar led3
                  enviacrc();//chamar a função de envio do byte de
                              //crc
                  led0 = 0;   //ligar led0 para sinalizar o envio
                              //do byte de crc em broadcast
                  delay(50);  //esperar 500 ms
                  if(RI_0==1) //se tiver sido recebido um byte de
                              //resposta:
                  {
                        FlagEstadoStandby = 0; //mudar do estado de
                                               //standby,
```


```
                FlagEstadoValidate = 1;//para validate
                ET0 = 1;          //activar o IRQ do timer 0
                break;            //sair do ciclo while
            }
            delay(25);            //esperar 250 ms
            led0 = 1;             //desligar led0
            delay(25);            //esperar 250 ms
        }
        while(FlagEstadoValidate)
        {
            RI_0 = 0;         //colocar a "0" a flag de
                              //recepção da porta série 0
            IDh = SBUF0;      //guardar o byte no buffer de
                              //entrada em IDhigh
            led1 = 0;         //ligar led1 para indicar a
                              //recepção do primeiro byte de
                              //resposta
            while(RI_0!=1&&counter!=0){} //esperar até
                                         //receber um byte ou
                                         //até que haja um
                                         //timeout
            if(counter==0)    //se tiver ocorrido um timout:
            {
                FlagEstadoValidate = 0;//mudar o estado de
                                       //validate para
                FlagEstadoStandby = 1; //standby
                counter = 150;    //voltar a colocar o
                                  //temporizador de timeout
                                  //a 1500 ms
                break;
                              //sair do ciclo while
            }
            counter = 150;    //voltar a colocar o
                              //temporizador de timeout a
                              //1500 ms
            RI_0 = 0;         //colocar a "0" a flag de
                              //recepção da porta série
            IDl = SBUF0;      //guardar o conteúdo do buffer
                              //de entrada em IDlow
            led2 = 0;         //ligar led2 para sinalizar a
                              //recepção do segundo byte de
                              //resposta
            while(RI_0!=1&&counter!=0){} //esperar até
                                         //receber um byte ou
                                         //até que haja um
                                         //timeout
            if(counter==0)    //se tiver ocorrido um timout:
            {
                FlagEstadoValidate = 0;//mudar o estado de
                                       //validate para
                FlagEstadoStandby = 1; //standby
                counter = 150;         //voltar a colocar o
                                       //temporizador de
                                       //timeout a 1500 ms
                break;
                              //sair do ciclo while
            }
            counter = 150;    //voltar a colocar o
                              //temporizador de timeout a
```

```
                                    //1500 ms
                  ET0 = 0;          //desactivar IRQ do timer 0
                  RI_0 = 0;         //colocar a "0" a flag de
                                    //recepção da porta série
                  crcanswer = SBUF0;      //guardar o conteúdo do
                                          //buffer de entrada e
                                          //crcanswer
                  a = crcanswer;
                  b = crctest^IDh^IDl;    //fazer a mesma operaçação
                                          //que deverá ter sido
                                          //feita pelo circuito
                                          //móvel
                  led3 = 0;               //ligar o led3 para
                                          //sinalizar a recepção do
                                          //terceiro bye de resposta
                  if(exists())      //se os bytes recebidos
                                    //correspondem a uma entrada na
                                    //lista
                  {
                        if(a==b)    //o terceiro byte de resposta
                                    //recebido é válido:
                        {
                              greenled = 0;     //ligar o led verde
                              redled = 1;       //desligar o led
                                                //vermelho
                              delay(500);       //esperar 5000 ms
                              greenled = 1;     //desligar o led
                                                //verde
                              redled = 0;       //ligar o led
                                                //vermelho
                              led0 = 1;         //desligar led0
                              led1 = 1;         //desligar led1
                              led2 = 1;         //desligar led2
                              led3 = 1;         //desligar led3
                              FlagEstadoStandby = 1; //mudar o
                                                     //estado de
                                                     //standby para
                              FlagEstadoValidate = 0;//validate
                              break;      //sair do ciclo while
                        }
                  }
//NOTA: esta parte do código apenas é executada caso os dados não
correspondam a uma entrada na lista
                  led0 = 1;         //desligar led0
                  led1 = 1;         //desligar led1
                  led2 = 1;         //desligar led2
                  led3 = 1;         //desligar led3
                  FlagEstadoStandby = 1; //mudar o estado de
                                         //standby para validate
                  FlagEstadoValidate = 0;
            }
      }
}
```

Na página seguinte pode-se ver o código fonte do programa implementado no microcontrolador do Móvel:

```
#include <DS89C4xx.h>

//Declaração de variáveis:
bit FlagEstadoStandby; //flag indicadora de estado do programa
bit FlagEstadoValidate;//                "
unsigned char Validation[4]={'a','b','c'};
sbit led0 = P0^4;      //led de sinalização de estado
sbit led1 = P0^5;      //          "
sbit led2 = P0^6;      //          "
sbit led3 = P0^7;      //          "

//Função de inicialização:
init_all()
{
       TMOD = 0x21;          //configurar timer 1 em modo 8b-
                             //autorecarga, timer 0 em modo 16b
       TH0 = 0xdb;           //valor de carga do timer 0
       TL0 = 0xff;           //    "
       TR0 = 1;              //activação do timer 0
       SCON0 = 0x50;         //porta série no modo UART 8b,
                             //recepção habilitada
       PCON = PCON|0x80;     //activação flag GF1 -flag de uso
                             //geral
       REN_0 = 1;            //habilitar recepção na porta série0
       TH1 = 0xfa;           //valor de carga do timer 1
       TL1 = 0x00;           //valor de recarga do timer 1
       TR1 = 1 ;             //activação do timer 0
       RI_0 = 0;             //colocar flag de recepção de dados
                             //na porta série a zero
       TI_0 = 0;             //colocar flag de transmissão de
                             //dados na porta série a zero
       FlagEstadoStandby = 1;
       FlagEstadoValidate = 0;
       led0 = 1;             //desligar led0
       led1 = 1;             //desligar led1
       led2 = 1;             //desligar led2
       led3 = 1;             //desligar led3
}

//Função para criar ciclos de espera temporizados:
void delay(int tempo)
{
      int i;
      for(i=0;i<tempo;i++)   //fazer n contagens de 10 ms
      {
            while(TF0!=1){}  //esperar até ao overflow do timer 0
            TF0 = 0;         //reset à flag de overflow do timer 0
            TH0 = 0xdb;      //colocar timer o a 0xdbff para uma
                             //temporização de 10 ms
            TL0 = 0xff;
      }
}

//Função para enviar bytes:
enviaChar(unsigned char a)
{
      SBUF0=a;    //mover a para o buffer de saída da port série
      while(TI_0!=1){} //esperar pelo fim do envio
      TI_0 = 0;        //colocar a flag de envio a "0"
}
```

```
//Programa princioal:
void main(void)
{
      init_all();            //chamar a função de inicialização
      while(1)
      {
            while(FlagEstadoStandby)
            {
                  if(RI_0 == 1)     //se houve recepção pela porta
                                    //série:
                  {
                        led0 = 0;   //ligar led0 para sinalizar a
                                    //recepção do byte de crc
                        Validation[2] = SBUF0; //guardar o conteúdo
                                               //do buffer de
                                               //entrada
                        FlagEstadoStandby = 0; //mudar o estado de
                                               //standby para
                        FlagEstadoValidate = 1;//validate
                        RI_0 = 0;              //colocar a flag de
                                               //recepção da porta
                                               //série a "0"
                  }
            }
            while(FlagEstadoValidate)
            {

      Validation[2]=Validation[0]^Validation[1]^Validation[2];
      //gerar o byte de validação
                  enviaChar(Validation[0]);    //enviar IDhigh
                  led1 = 0;         //ligar led1 para indicar o
                                    //envio do 1o byte de resposta
                  delay(100);       //esperar 1000 ms
                  enviaChar(Validation[1]);    //enviar IDlow
                  led2 = 0;         //ligar led2 para sinalizar o
                                    //envio do 2o byte de resposta
                  delay(100);       //esperar 1000 ms
                  enviaChar(Validation[2]);    //enviar byte de
                                               //verificação
                  led3 = 0;   //ligar led3 para sinalizar o envio
                              //do 3o byte de resposta
                  delay(100);       //esperar 1000 ms
                  FlagEstadoStandby = 1;       //mudar o estado de
                                               //validate para
                  FlagEstadoValidate = 0;      //para standby
                  led0 = 1;   //desligar led
                  led1 = 1;   //desligar led
                  led2 = 1;   //desligar led
                  led3 = 1;   //desligar led
            }
      }
}
```

Para testar algumas das funcionalidades do XBee definimos endereços de 16 bits na configuração dos mesmos, colocando o endereço da base a 0000h e o endereço do móvel a 0001h. Por outro lado, definimos que o Destination

Address Low da Base seria FFFFh (Broadcast) e o Destination Address Low do Móvel seria 0000h (endereço da Base). Deste modo, na aplicação prática a Base comunicaria com qualquer veículo e cada veículo comunicaria apenas com a base.

De seguida está uma explicação detalhada destas configurações.

- Configurar o XBee

- O circuito de configuração do XBee

Para configurar os módulos XBee a empresa Maxstream fornece várias placas de desenvolvimento, no entanto, após a análise das mesmas e alguma pesquisa na internet, chegou-se à conclusão que se trataria de uma solução muito dispendiosa. Por este motivo, decidiu-se construir um circuito de configuração próprio que permitisse a configuração dos módulos. Para além de menos dispendiosa, esta solução mostrou-se também benéfica em termos de aquisição de conhecimentos.

Optou-se por configurar os módulos XBee via porta série, visto os módulos XBee disponibilizarem para este fim os pinos Dout, Din, DTR e RTS -o essencial para uma ligação ao PC.

Para adaptar os níveis de tensão RS232 aos níveis admitidos pelo XBee recorreu-se a um Hex-inverter SN74LS04N (IC1). Este integrado fornece tensões de 5V, no entanto o módulo XBee apenas suporta tensões até 3,3V. Para contornar este obstáculo recorreu-se a divisores de tensão em todas as entradas do XBee.

Para os divisores de tensão recorreu-se aos valores de 1,8kΩ (para R1, R3, e R5) e 3,3kΩ (para R2, R4, R6) por permitirem uma tensão aceitável nas entradas do XBee.

Cálculo da tensão aplicada aos módulos XBee:

$$\frac{5\ V}{\left(\frac{3{,}3+1{,}8}{3{,}3}\right)} = 3{,}24\ V$$

No pin Dout, por se tratar de uma saída, não foi necessário recorrer a um divisor de tensão.

A alimentação do circuito foi realizada por meio de duas fontes de laboratório, não havendo assim necessidade de implementar circuitos reguladores de tensão.

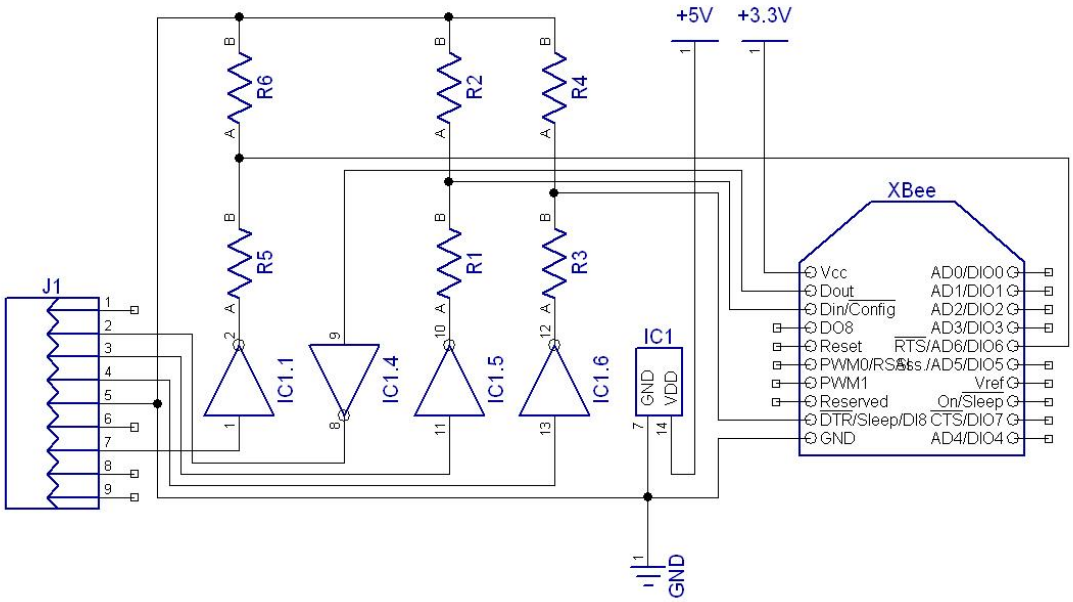

Figura 13 – Circuito de configuração do XBee

Ficha Dsub 9 pins	Nº do Pin	Módulo XBee	Nº do Pin
Receive Data	2	Dout	2
Transmit Data	3	Din	3
Data Terminal Ready	4	DTR	9
Ground	5	GND	10
Request to Send	7	RTS	16

Tabela 4 – Correspondência entre os pinos do XBee e a porta série

Lista de componentes:

R1, R3, R5 - 1,8 kΩ +/- 5%

R2, R4, R6 - 3,3 kΩ +/- 5%

C1 - SN74LS04N

J1 - Ficha Dsub 9 pins

XBee - Módulo XBee

- O software X-CTU

Este software é bastante útil pois permite configurar o XBee de um modo muito fácil utilizando o circuito descrito anteriormente. Quando o programa é executado abre a janela mostrada na figura 14, em que é possível configurar a velocidade da porta série, formato dos dados, de entre outras configurações.

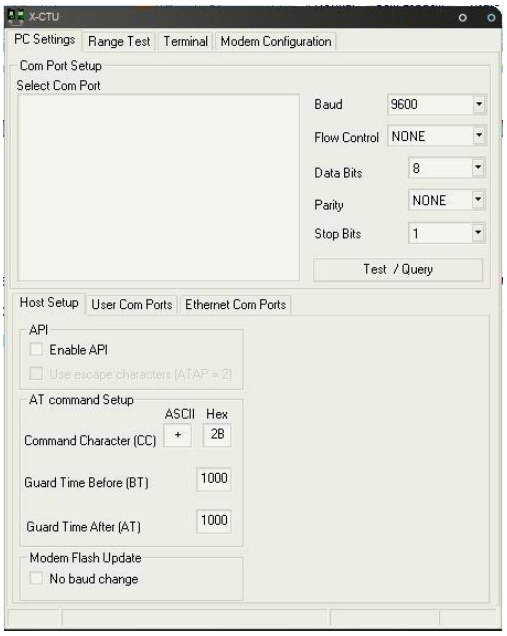

Figura 14 – Software de configuração dos módulos XBee X-CTU

Se se carregar no separador Range Test aparece vista apresentada na figura 15, em que é possível fazer um teste de alcance do módulo XBee:

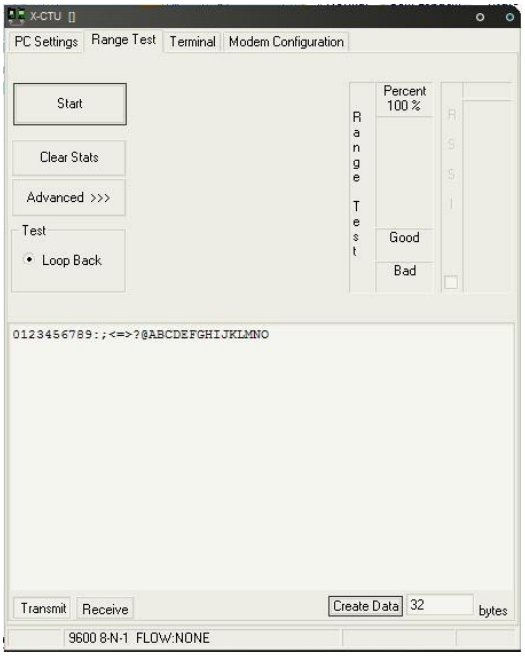

Figura 15 – Vista do ambiente de teste de alcance

Se se carregar no separador Terminal aparece a vista da figura 16 em que se podem inserir os comandos AT para configurar os módulos XBee. No entanto, este modo de configuração é díficil, trabalhoso e penoso.

Figura 16 – Vista do ambiente de terminal

Por fim, se se carregar no separador Modem Configuration obtemos a vista apresentada na figura 17, que é a que mais nos interessa e que facilita grande parte do trabalho de configuração.

Figura 17 – Vista do ambiente de configuração do modem XBee

Se o circuito estiver bem montado e devidamente ligado ao computador, ao carregar no botão “Read” o programa reconhecerá automaticamente que tipo de dispositivo está ligado e apresentará todas as opções disponíveis para fazer a configuração. Neste caso reconhecerá um dispositivo do tipo XB24. Quando as opções forem apresentadas, a janela terá o aspecto mostrado na figura 18.

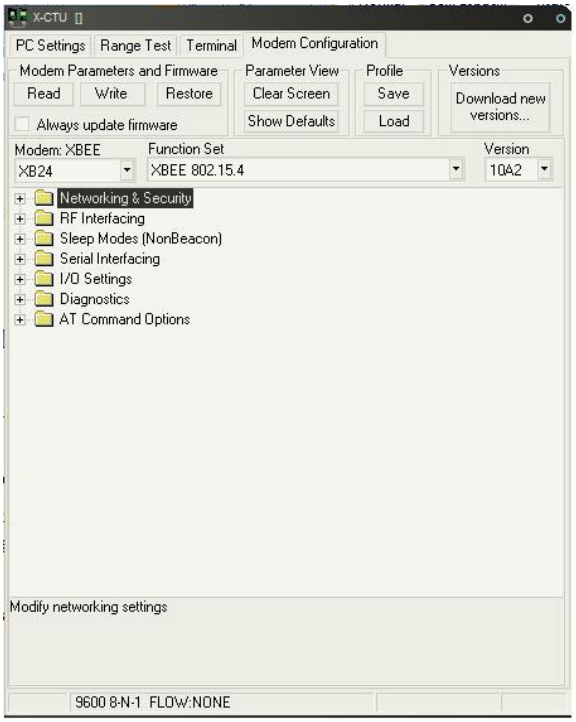

Figura 18 – Vista das pastas de opções de configuração

Como se pode ver, existem 7 conjuntos de configurações possíveis:

- Networking & Security;

- RF Interfacing;

- Sleep Modes (NonBeacon);

- Serial Interfacing;

- I/O settings;

- Diagnostics;

- AT Command Options.

De seguida analisaremos que tipo de opções se podem mudar dentro de cada um destes grupos, pela ordem apresentada acima.

- Networking & Security – neste grupo temos as opções mais importantes, ou seja, todas as opções relativas ao estabelecimento de redes e segurança das comunicações.

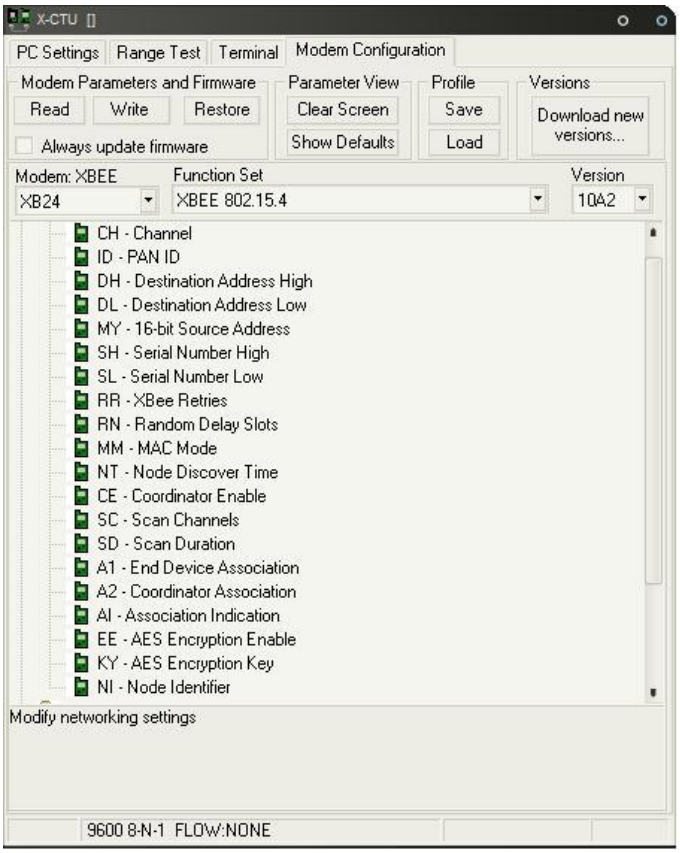

Figura 19 – Vista das opções de configuração do grupo de Networking & Security

Referindo apenas as configurações mais importantes e de cima para baixo começamos pelo Channel ID, em que podemos escolher o canal de comunicação utilizado. De seguida temos o Destination High(32 bits) e Destination Low(32 bits). Estes campos indicam o endereço de destino para onde devem ser enviados os dados. O facto de o endereço de destino ter no total 64 bits deve-se ao facto de os módulos poderem ter dois tipos de endereço: têm o seu número de série, de 64 bits (SH e SL), que não pode ser alterado; e têm um endereço de 16 bits configurável (MY). De seguida temos o endereço configurável de 16 bits do módulo que acabamos de referir (MY). Este endereço pode assumir qualquer valor de 0000h a FFFEh, pois o FFFF não deve ser utilizado uma vez que corresponde à transmissão em boradcast. De seguida temos as XBee Retries (RR) e o Node Discover Time (NT), que correspondem, respectivamente, ao número de tentativas máximo que o módulo faz para enviar um dado e ao tempo máximo que o módulo procura por outros nós na rede em que se insere. Podemos definir o módulo como ZigBee Coordinator manualmente activando o registo CE – Coordinator Enable. Os registos A1, A2 e AI referem-se também aconfigurações de rede. Nos registos AES Encryption Enable (EE) e AES Encryption Key (KY) podemos definir e activar uma chave de encriptação para as comunicações do módulo XBee, de

modo a garantir uma maior segurança nas comunicações. Para alterar qualquer um dos registos que seja alterável basta clicar sobre ele e preencher o respectivo campo.

- RF Interfacing – neste grupo temos as opções de radiofrequência.

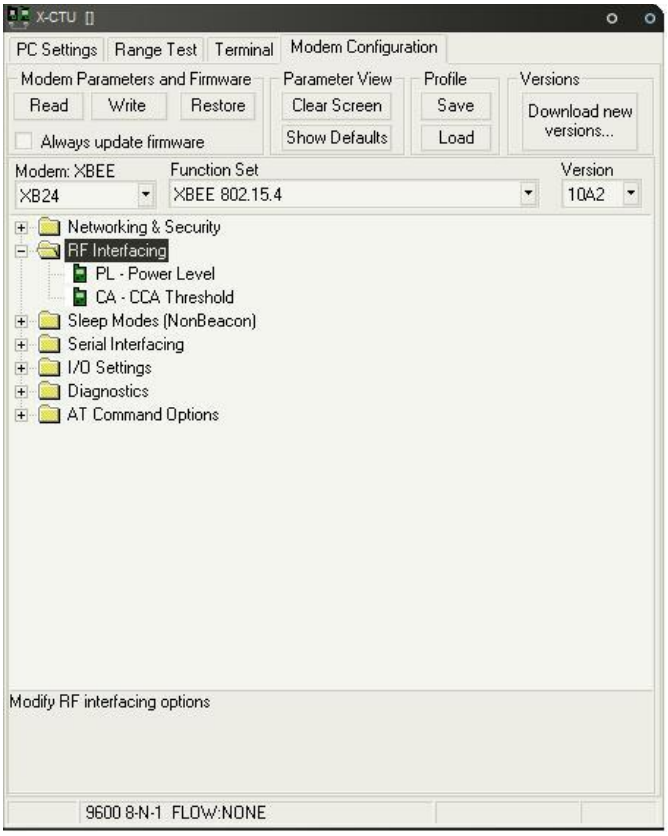

Figura 20 – Vista das opções de configuração do grupo de RF Interfacing

Neste grupo de configurações a opção mais importante é o registo Power Leve (PL) em que se pode escolher o grau de potência de emissão, de modo a garantir um maior ou menor alcance.

- Sleep Modes – neste grupo temos as opções de consumo energético

Neste grupo podem-se configurar os modos de Sleep do XBee de modo a minimizar o consumo energético. Além disso pode-se ainda configurar o tempo durante o qual o módulo fica activo antes de entrar no modo de Sleep seleccionado.

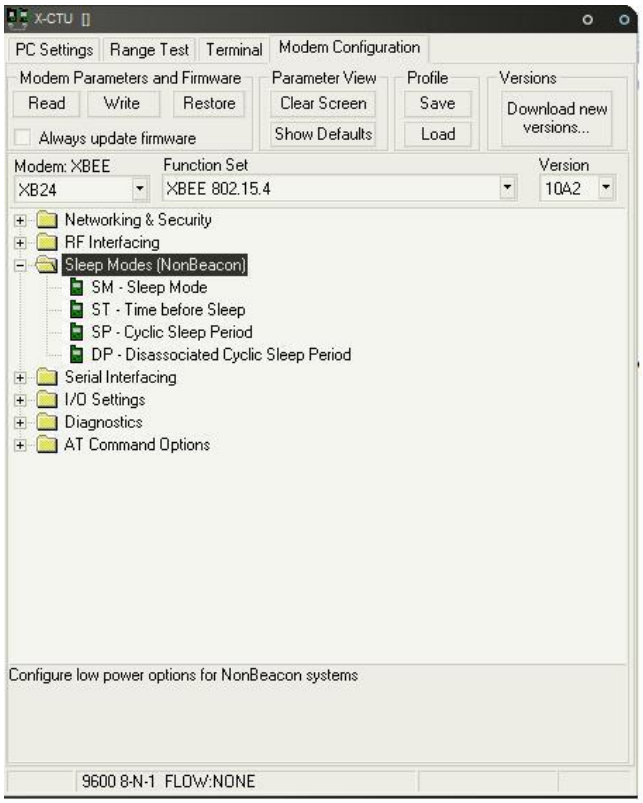

Figura 21 – Vista das opções de configuração do grupo de Sleep Modes

- Serial Interfacing – neste grupo temos as opções de interface série.

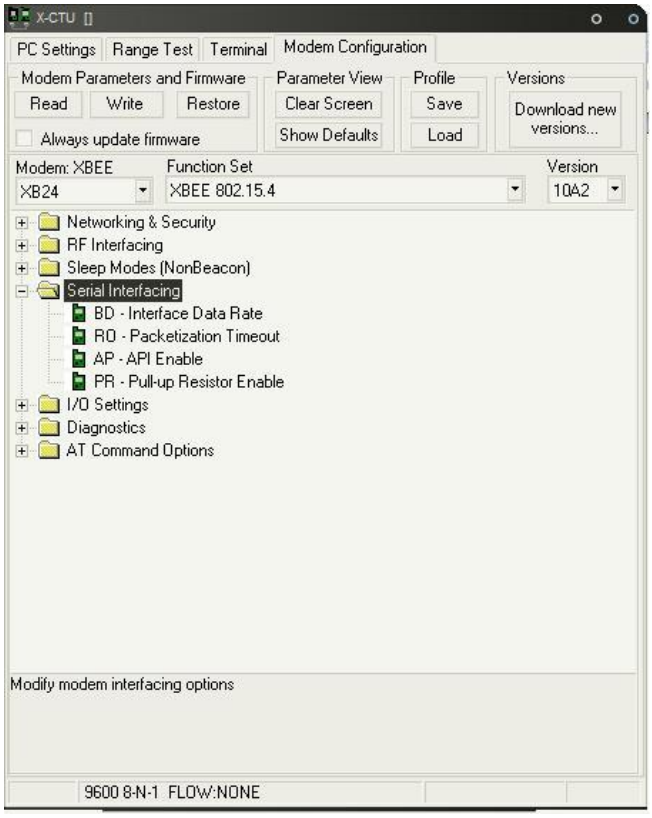

Figura 22 – Vista das opções de configuração do grupo de Serial Interfacing

Neste grupo de opções pode-se escolher a taxa de transferência de dados da interface série (BD), assim como a activação de resistências e Pul-up internas (PR).

- I/O Settings – neste grupo temos as opções de I/O.

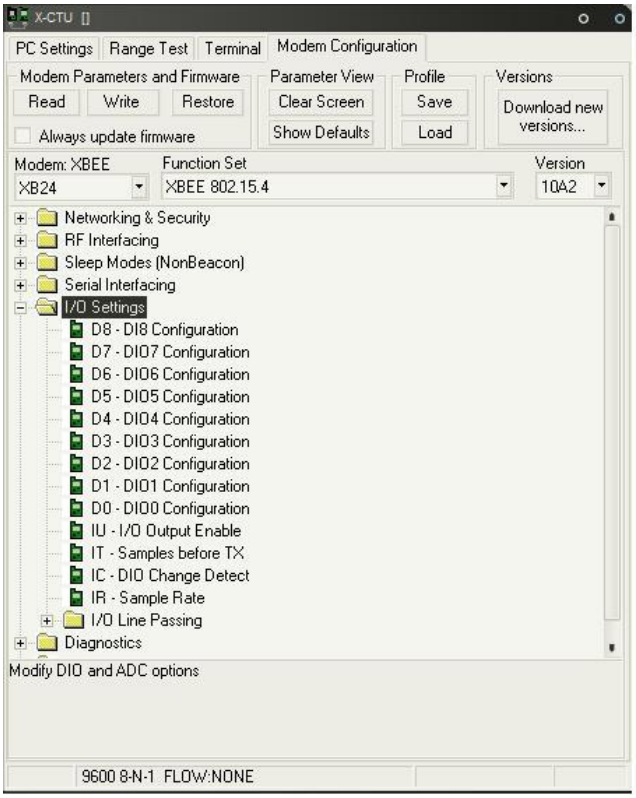

Figura 23 – Vista das opções de configuração do grupo de I/O Settings

Neste conjunto temos as opções relativas ao ADC interno do módulo. Como uma das principais aplicaçõesdo ZigBee são as redes de sensores, estes módulos trazem um ADC incluído de modo a poder fazer aquisição de dados sem necessitar do auxílio de um microcontrolador.

- Diagnostics – neste grupo temos as opções de diagnostico.

  Neste grupo pode-se ver qual é a versão do firmware (VR) e do hardware (HV) do módulo, assim como a potência do sinal recebido (DB). Pode-se ainda ver a quantidade de falhas ocorridas (EC e EA).

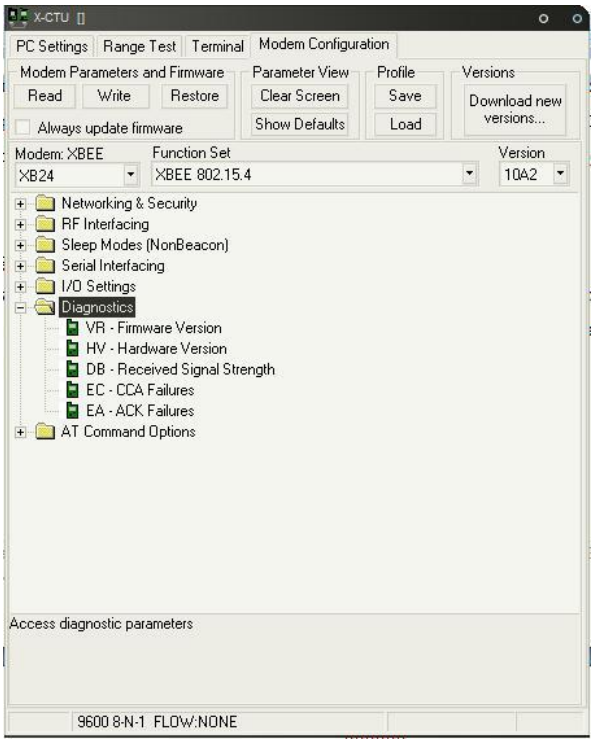

Figura 24 – Vista das opções de configuração do grupo de Diagnostics

- AT Command Options – neste grupo temos as opções de comandos AT.

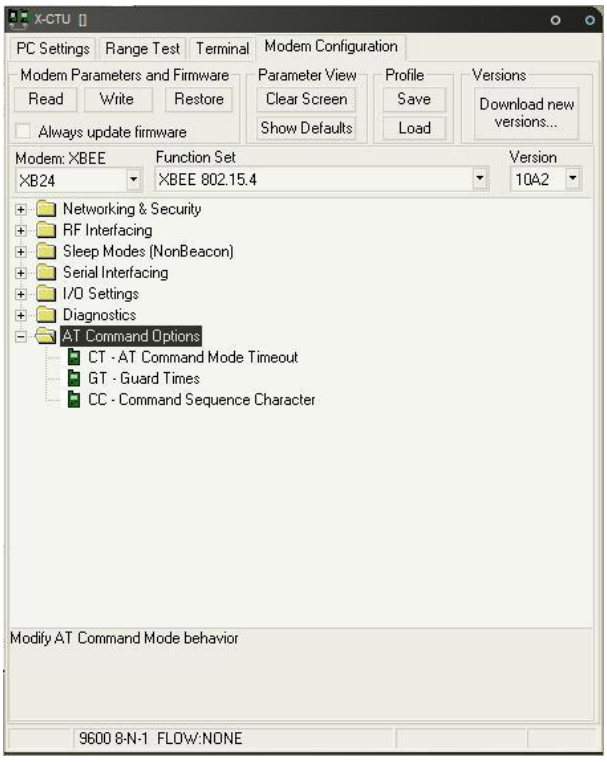

Figura 25 – Vista das opções de configuração do grupo de AT Command Options

Neste último conjunto de opções temos as opções de comandos AT, em que o principal registo é o AT Command Mode Timeout (CT) que corresponde ao tempo máximo para envio de comandos AT, no caso de optarmos por esse modo de configuração.

Após feitas as configurações, basta carregar no botão "Write" e o módulo passará a trabalhar com a nova configuração. Qualquer configuração que seja feita pode ser guardada para ser mais tarde restaurada.

De acordo com o que já foi dito durante esta explicação, de seguida seguem-se uma série de exemplos de configuração de redes utilizando módulos XBee, tendo sido o exemplo da figura 26 o utilizado na aplicação demonstrativa:

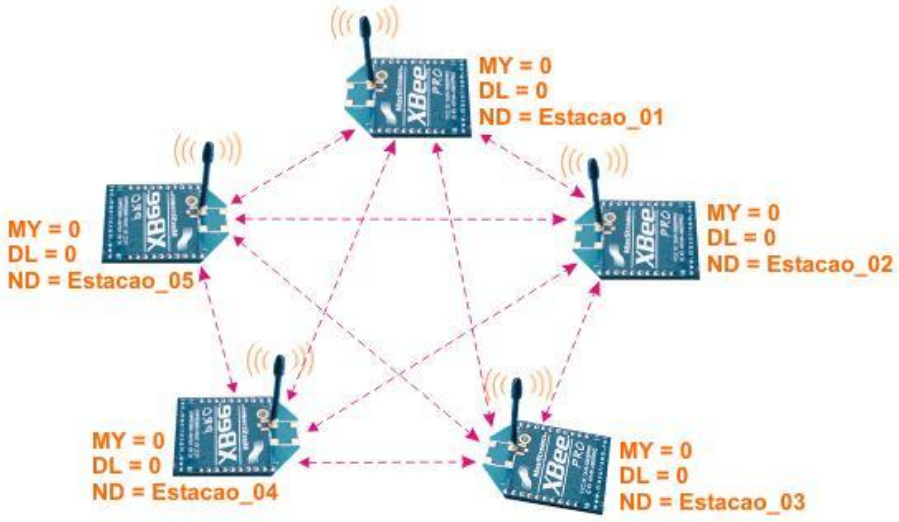

Figura 25 – Exemplo de Mesh Network

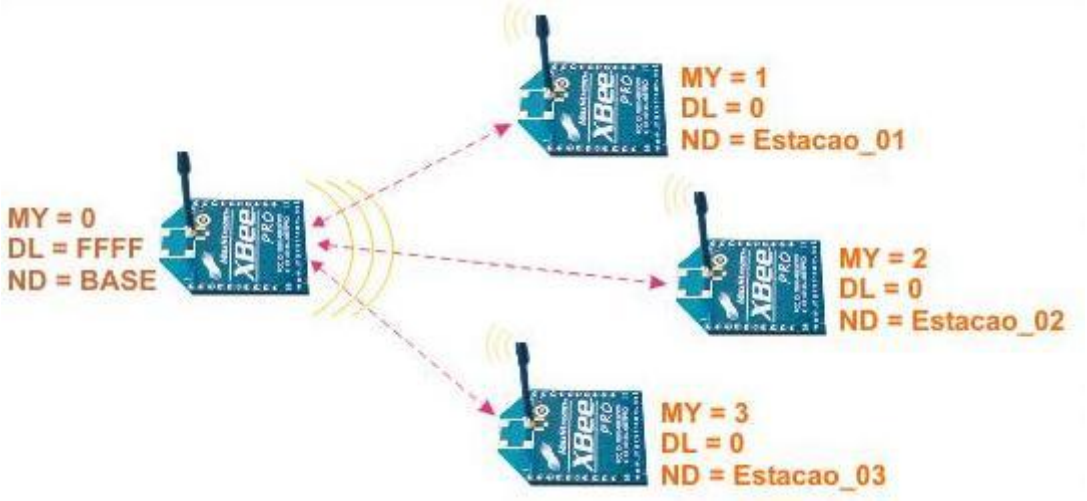

Figura 26 – Exemplo de Comunicação Ponto a Multiponto

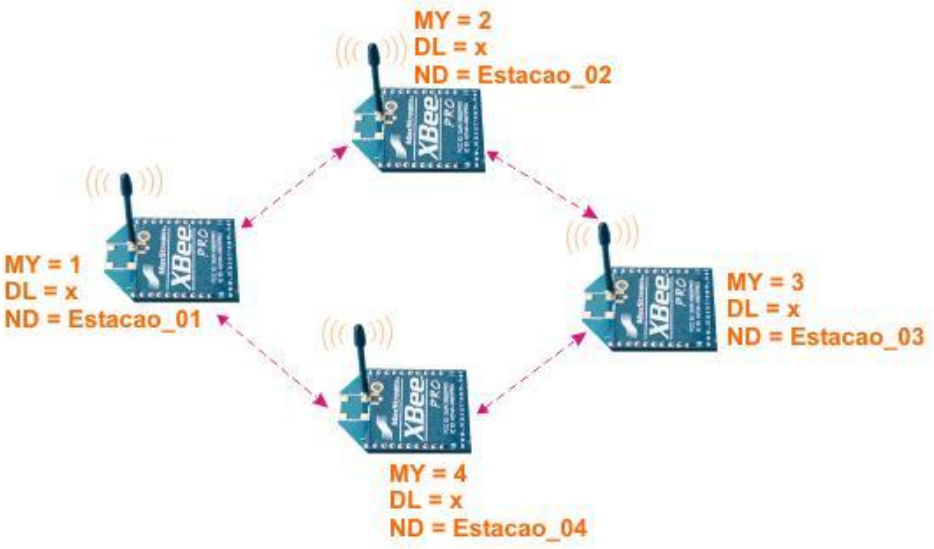

Figura 27 – Exemplo de Comunicação Ponto a Ponto

Conclusão

Após a realização deste trabalho podemos concluir que as redes sem fios ZigBee são bastante simples de implementar e funcionam na perfeição, confirmando-se o slogan da ZigBee Aliance: "Wireless Control that Simply Works". No entanto, nesta abordagem foram apenas focados os pontos principais desta nova tecnologia das comunicações sem fios. Uma análise prática mais aprofundada seria bastante útil para perceber melhor o estabelecimento de redes e a inserção \ remoção de novos dispositivos nessas mesmas redes. Contudo, com os recursos monetários que se puderam despender conseguiu-se fazer uma análise prática geral dos pontos mais importantes deste tema. É de referir que foi muito grande o enriquecimento a nível técnico obtido durante este estudo sobre esta nova tecnologia que se encontra ainda em fase de grande crescimento, sendo alvo de aperfeiçoamentos sucessivos por parte dos grandes fabricantes mundiais de semicondutores que estão constantemente a adicionar novas funcionalidades a estes dispositivos. Exemplo disso é a Texas Instruments que já comercializa módulos ZigBee que permitem a obtenção de uma localização relativa dos dispositivos ZigBee através de triangulações.

Deste modo, concluímos que o ZigBee permite a criação de sistemas bastante versáteis a adaptáveis a qualquer quase todas as circunstâncias, tendo apenas como falha o baixo ritmo de transferência de dados, apesar de ter sido concebido para aplicações com uma baixa exigência no que diz respeito à quantidade de dados transmitidos.

- O futuro

Depois deste estudo resta-nos manter o interesse por esta área, que certamente virá revolucionar extensas áreas tecnológicas em diversos ramos, sendo a domótica o perfeito exemplo disso, mas não só. Com a evolução que esta tecnologia tem tido, muitas novas funcionalidades e aperfeiçoamentos surgirão ao longo do tempo, sendo possível que um dia esta tecnologia esteja implementada de tal maneira que seja possível fazer todo e qualquer controlo com o ZigBee.

Referências

- www.zigbee.org/
- http://www.rogercom.com/ZigBee/ZigBeePag03.htm
- http://itp.nyu.edu/~raf275/meshnetworking/XBee/XBee_firmware_upgrade.html
- www.maxstream.net/
- http://www.gta.ufrj.br/ensino/CPE825/2006/resumos/TrabalhoZigbee.pdf
- http://paginas.fe.up.pt/~ee02055/info_zigbee.pdf