CAPÍTULO 1

INTRODUÇÃO

1

1.1 OBJECTIVOS

TRANSMISSÃO SÉRIE

CAMADA FÍSICA

SINAIS

MEIOS DE TRANSMISSÃO:

TÉCNICAS DE MODULAÇÃO

TÉCNICAS DE DETECÇÃO DE ERRO E CORRECÇÃO

CAMADA LÓGICA

SUB-CAMADA MAC (MEDIUM ACCESS CONTROL)

SUB-CAMADA LLC (LINK LAYER CONTROL)

CAMADA DE REDE

ENDEREÇAMENTO

ENCAMINHAMENTO

1.2 ORGANIZAÇÃO DOS CAPÍTULOS SEGUINTES

Nos capítulos seguintes serão apresentadas várias redes de comunicação de dados.

Cada rede pode utilizar um ou vários protocolos de comunicação, actuando cada um deles ao nível de uma ou de várias camadas do modelo OSI.

A apresentação de cada rede está organizada, de acordo com o modelo de referência OSI, camada a camada, e a apresentação começará sempre pela sua camada física (sempre que existir).

Na camada física serão apresentadas as formas de onda dos sinais, a sua frequência e amplitude. Serão apresentadas as ligações eléctricas, rádio eléctricas ou ópticas entre os vários dispositivos. Seguidamente serão apresentadas as mensagens (tramas) e posteriormente será explicada a forma como os dispositivos acedem e partilham o meio de transmissão.

CAPÍTULO 2

EIA 232

José Santos
2008/2009

2

2.1 INTRODUÇÃO

Este protocolo foi proposto por …. No ano de xx/xx/xx… e tinha por objectivo…

2.2 CAMADA FÍSICA

TOPOLOGIA DAS LIGAÇÕES

Figura 2.1 - Rs232 – Topologia das ligações eléctricas
Figura 2.2 – Rs232 – Sinais eléctricos

ESTRUTURA DA PALAVRA SÉRIE

Figura 2.3 – Rs232 – Estrutura da mensagem (Trama)

INTERFACE FÍSICA

Figura 2.4 – Rs232 – Ficha DB9 e DB25

2.3 HARDWARE UTILIZADO

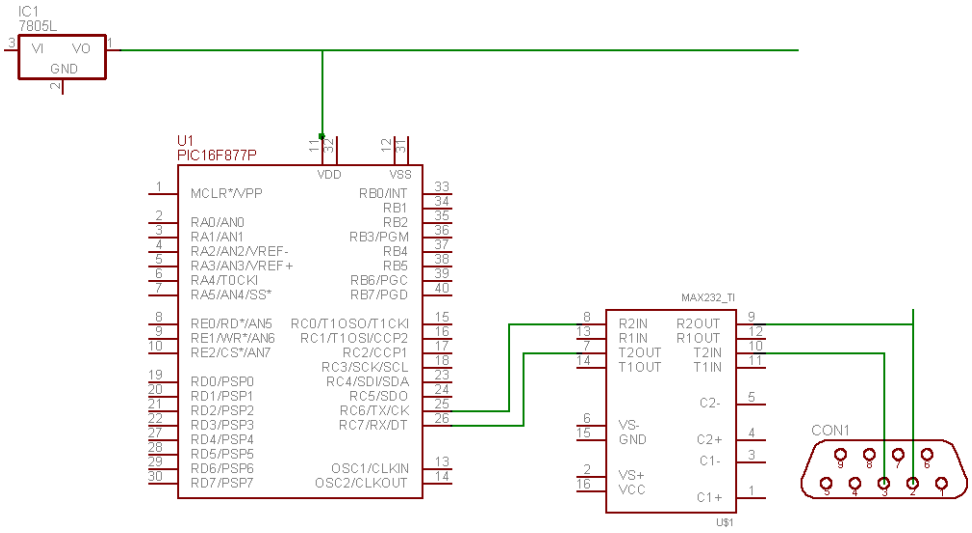

Figura 2.5 – Esquema eléctrico

2.4 EXEMPLOS DE CÓDIGO

Figura 2.6 – Diagrama de sequências – UML

PROGRAMA DE CONFIGURAÇÃO

PROGRAMA DE ENVIO

PROGRAMA DE RECEPÇÃO

CAPÍTULO 3

IRDA

Pedro Gouveia
2008/2009

3

3.1 INTRODUÇÃO

3.2 CAMADA FÍSICA

LIGAÇÕES ELÉCTRICAS DOS DISPOSITIVOS AO MEIO

SINAIS ELÉCTRICOS E O MEIO DE TRANSMISSÃO

Figura 3.1 – IRDA – Topologia das ligações Físicas.

Figura 3.2 – IRDA – Sinais eléctricos

Figura 3.3 – IRDA – Trama

3.3 CAMADA LÓGICA

3.4 HARDWARE UTILIZADO

Figura 3.4 – Esquema eléctrico

3.5 EXEMPLOS DE CÓDIGO

Figura 3.5 – Diagrama de sequências - UML

PROGRAMA DE CONFIGURAÇÃO

PROGRAMA DE ENVIO

PROGRAMA DE RECEPÇÃO

CAPÍTULO 4

ZIGBEE

Hugo Tavares
2008/2009

4

4.1 INTRODUÇÃO

4.1.1 Breve descrição do ZigBee

O mercado das telecomunicações tem vindo a evoluir cada vez mais nos últimos anos, no sentido de poder satisfazer as, cada vez mais, exigentes necessidades da "sociedade de informação". Isto originou a criação de várias redes wireless (sem fios), cada uma orientada para uma certa aplicação, tais como o **Bluetooth**, **WiFi**, **GSM**, **UMTS**, entre outras. Ora, foi neste contexto que surgiu muito recentemente o protocolo de comunicações wireless ZigBee.

O nome ZigBee vem da junção de ziguezague (Zig) com abelha (Bee). Este nome surgiu pelo facto de, na rede em malha apresentar vários percursos entre dispositivos, andando a informação em ziguezague pela rede. Tal como as abelhas, além de viverem numa colmeia (a rede), também voam em ziguezague, o que lhes permite informar aos outros membros da comunidade a distância, a direcção e localização de certos objectos que encontraram.

Criado pela **ZigBee™ Alliance**, o ZigBee surgiu por volta de Dezembro de 2004, no entanto a sua primeira versão só foi apresentada ao público a 27 de Junho de 2005, desde então o ZigBee tem vindo a despertar, cada vez mais, interesse em várias áreas, desde a indústria até à domótica. A **ZigBee™ Alliance** é uma aliança constituída por mais de 200 empresas, como a **Microchip**, **Siemens**, **Philips**, **Samsung**, entre outras, na qual estão, também, incluídos especialistas na área das telecomunicações e membros do **IEEE** (*Institute of Electrical and Electronics Engineers*).

Antes do aparecimento do ZigBee não existia no mercado nenhum protocolo de redes sem fios que fosse globalmente aceite na área de sensores e dispositivos de controlo. O ZigBee pretende aliar à transmissão de dados sem fios um consumo energético reduzido e uma elevada fiabilidade, destacando-se, assim, de outras redes sem fios, como o **Bluetooth** ou **WiFi**.

De facto o ZigBee possui uma grande potencialidade na sua aplicação em redes de baixo custo, baixo consumo e baixas taxas de transmissão. A rede ZigBee apresenta, assim, as seguintes características:

- **Baixo consumo energético** – possibilitando a utilização de pequenas unidades remotas, autónomas durante longos períodos de tempo sem a necessidade de manutenção.
- **Baixo custo** – devido à reduzida exigência do protocolo é possível implementa-lo em unidades com pouca memória e processamento.
- **Elevada densidade de dispositivos** – possibilita ligar numa só rede um máximo de 65535 dispositivos a um coordenador.
- **Diferentes tipologias de rede** – admite ligações em estrela (*Star*), malha (*Mesh*) ou árvore (*cluster tree*), permitindo o estabelecimento de redes de nós *ad-hoc*.
- **Operação em três bandas** – é possível operar em três bandas de rádio: 2.4 GHz (globalmente), 915 MHz (Estados Unidos) e 868 MHz (Europa). Cada uma apresenta, respectivamente, uma taxa de transmissão de 250 kbps, com 16 canais, 40 kbps, com 10 canais, e 20 kbps, com um só canal. Utilizando, no caso de 2.4 GHz, a modulação O-QPSK (*Offset Quadrature Phase Shift Keying*) e nos outros casos BPSK (*Binary Phase Shift Keying*).
- **Tempos de ligação reduzidos** – apresenta maior rapidez na ligação e na passagem do modo *standby* para activo e também uma baixa latência.
- **Dois estados de operação** – activo, quando envia ou recebe dados, e *sleep*, não sendo necessária a preocupação da aplicação na selecção destes estados.
- **Encaminhamento** – o protocolo ZigBee, quando não exista uma ligação directa com o dispositivo de destino, permite que as mensagens sejam reencaminhadas por outros dispositivos da rede.
- **Segurança** – utilizando um algoritmo de encriptação de mensagens.

4.1.2 Dispositivos da rede ZigBee

Os dispositivos presentes numa rede ZigBee podem ser categorizados em dois grupos distintos: os FFD (*Full Function Devices*) e os RFD (*Reduced Function Devices*), definidos na norma **IEEE 802.15.4**. Os FFD são os dispositivos que implementam todas as camadas protocolares (*stack*), sendo os dispositivos mais complexos. No entanto são, também, os mais versáteis, podendo funcionar como coordenador, *router* ou, até mesmo, RFD. Por outro lado, os RFD são os mais simples, pelo que, ao implementarem apenas parte do protocolo apresentam menores exigências

por parte do hardware, o que implica uma redução significativa nos custos totais da rede, visto que numa rede ZigBee, existem mais dispositivos RFD que FFD.

Ainda dentro dos dispositivos da rede ZigBee, estes podem ter várias funções:

- ***Coordenador*** – é o dispositivo que cria e gere a rede toda, mantendo a tabela de encaminhamento das mensagens. Em cada rede ZigBee existe unicamente um dispositivo coordenador, podendo este comunicar directa, ou indirectamente, com qualquer outro dispositivo da rede.
- ***Router*** – é o dispositivo que, normalmente, faz a ligação entre o coordenador e os RFD's, sendo o seu objectivo encaminhar as mensagens entre estes. Para isso cada router pode comunicar com o coordenador, outros *router's* e com os RFD's ligados a este, implementado toda a *stack* ZigBee.
- ***Endpoint*** – é o dispositivo mais simples, que tem apenas como função monitorar e enviar o estado das vaiáveis a ele associadas, podendo apenas comunicar com o FFD a ele directamente ligado.

4.1.3 Tipologias de ligação

Sendo uma das grandes vantagens desta rede a versatilidade, existem, como já fora mencionado anteriormente, três tipologias diferentes para a estruturação duma rede ZigBee: Estrela, Malha ou Árvore.

Perspectivando este protocolo uma rede de uma forma *ad-hoc*, não existe uma topologia predeterminada, nem um controlo centralizado. No entanto, em qualquer das três tipologias disponíveis, existe um factor comum: o coordenador é o dispositivo responsável pela iniciação e controle da rede.

ESTRELA (*STAR*):

Neste tipo de configuração, como é possível ver na figura 4.1, todos os dispositivos são *endpoint's*, podendo estes ser FFD ou RFD, e possuem uma ligação directa com o coordenador, dispensando-se assim dos dispositivos de reencaminhamento na rede, os *router's*. Esta tipologia é bastante utilizada nas redes com poucos dispositivos e próximos, devido à sua simplicidade, quer da sua configuração, quer dos algoritmos necessários para a sua criação, o que conduz a uma menor sobrecarga de processamento do coordenador.

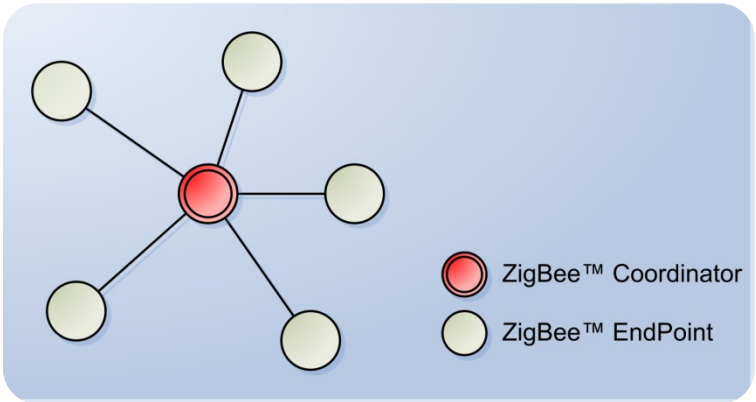

Figura 4.1 - Configuração em Estrela (Star)

MALHA (*MESH*):

Na configuração em malha, representada na figura 4.2, os dispositivos FFD são livres de comunicarem com qualquer tipo de dispositivos, FFD ou RFD. O que torna esta, tipologia, uma das mais vantajosas, já que permite redes de longo alcance e ultrapassar obstáculos que enfraqueçam o sinal. Ao contrário da situação anterior, o controlador continua a registar a entrada e saída de dispositivos, mas já não assume um papel tão preponderante no fluxo de informação.

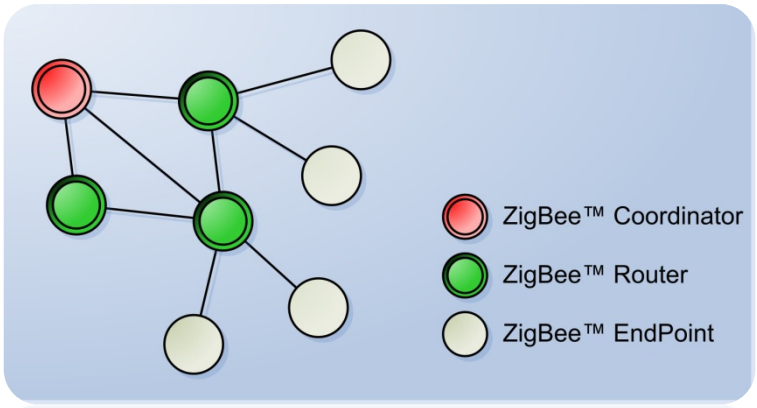

Figura 4.2 - Configuração em Malha (Mesh)

No caso da perda de conectividade de um *endpoint* com um *router* originam-se duas situações distintas. Primeiro é necessário reconfigurar as rotas de acesso de toda a rede, seguidamente, e se existirem dispositivos RFD que tenham perdido a ligação com a rede é necessário recuperar essa conectividade. Porém, este tipo de acção só é possível na presença de *router's* perto do router, que perdeu a ligação, e que possam aceitar esses *endpoint's* na sua rede interna.

ÁRVORE (*CLUSTER TREE*):

Esquematizada na figura 4.3, esta tipologia apresenta semelhanças com a estrutura em malha, usando também dispositivos *router's*. No entanto estes apresentam uma função organizativa de rede, sendo-lhe conferido níveis de hierarquização. Nessa estrutura hierárquica é o coordenador que assume o papel de nó nuclear da rede, como acontecia no caso da estrutura em estrela.

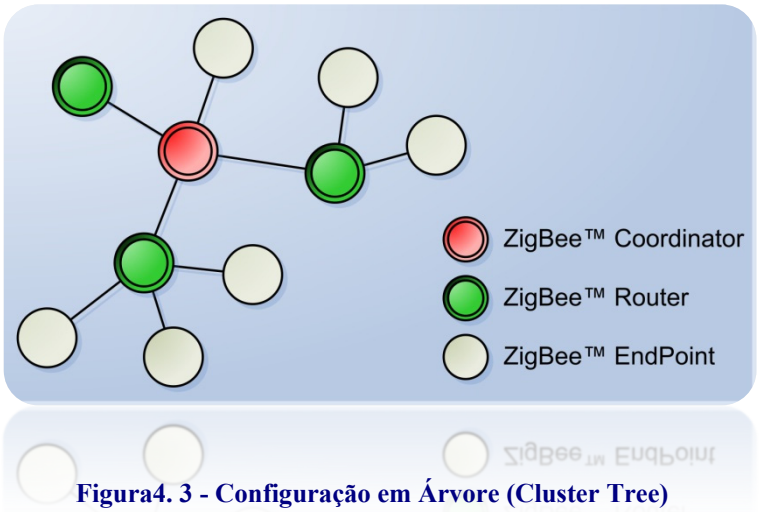

Figura4. 3 - Configuração em Árvore (Cluster Tree)

4.2 Arquitectura protocolar do ZigBee

Baseado no modelo OSI (*Open Systems Interconnection*) de sete camadas, o protocolo ZigBee, ao contrário do modelo OSI, apenas define as camadas de interesse para atingir as suas funcionalidades.

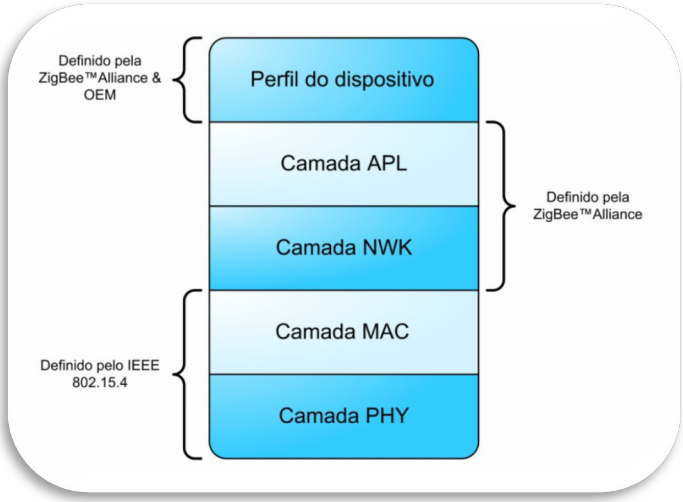

Figura 4.4 - Camadas da arquitectura protocolar ZigBee

Cada camada executa serviços específicos disponibilizados pela camada superior. A entidade de dados disponibiliza dados para o serviço de transmissão e a entidade de gestão disponibiliza dados para todos os outros serviços. Cada entidade de serviço expõe uma interface para a camada superior através do SAP (*Service Access Point*) e cada SAP suporta um número de primitivas de serviço (figura 4.6) para activar a funcionalidade que se pretende solicitar.

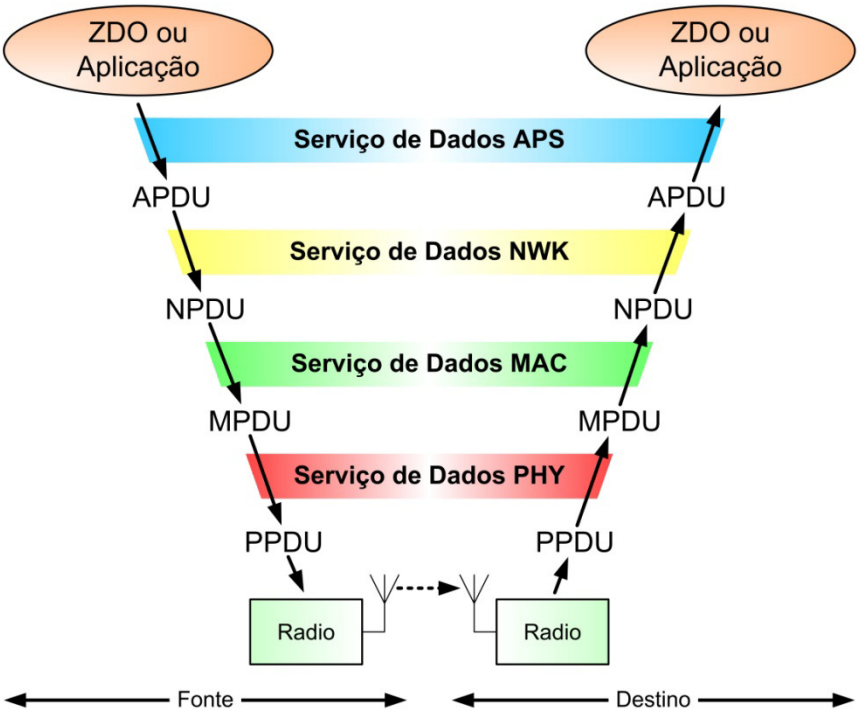

Figura 4.5 – Serviços de transferência de dados entre dois dispositivos

O protocolo ZigBee é definido pela ZigBee™ Alliance, sendo esta responsável pelas camadas de rede (NWK) e de aplicações (APL). Estas camadas assentam nas camadas física (PHY) e de acesso ao meio (MAC), as quais são definidas pela norma IEEE 802.15.4.

A camada física (PHY) suporta três tipos de frequência distintas, já referidas anteriormente, proporcionando ao programador a escolha da frequência que introduzam menor interferência no meio onde a rede será implementada. A banda mais comum para as aplicações utiliza a banda destinada a aplicações industriais, científicas e médicas conhecida por ISM e disponibilizando 16 canais diferentes para a sua utilização à frequência de 2.4 GHz. No entanto utiliza a mesma gama de frequência do WiFi o que pode originar, em espaços com redes muito sobrecarregadas, certos problemas de conectividade. Depois existem também as frequências de 915 MHz, utilizada nos Estados Unidos, e a de 868 MHz, utilizada em toda a Europa.

A camada PHY é responsável por transmitir e receber mensagens na rede, através de um canal, seleccionado pela própria camada, de radiofrequência (RF). Além disso, é também sua função, controlar o *transceiver*, detectar o nível de energia (ED – *Energy Detection*), indicar a qualidade de ligação (LQI – *Link Quality Indication*).

A camada de acesso ao meio (MAC) define os, já referidos anteriormente, dispositivos FFD e RFD. Nesta camada é bem visível a diferença entre estes dispositivos, visto que, enquanto o FFD apresenta todas as funções, o RFD apenas dispõe das funções básicas que lhe permitam enviar mensagens para a rede e controlar a suas variáveis. Cabe, também a esta camada controlar o acesso aos canais RF, utilizando mecanismos de detecção de prevenção de colisão CSMA-CA (*Carrier Sense Multiple Access – Collision Avoidance*), comunicando à camada inferior PHY.

Ao invés da camada de ligação à rede (PHY) que apenas identifica dois tipos de dispositivos, a camada de rede (NWK) cria mais uma subdivisão nos FFD, podendo ser distinguidos entre coordenador e router. A camada de rede é também responsável pela criação da rede, criação dos endereços dos nós, gestão da rede e da descoberta de novas rotas entre dispositivos. Esta é hierarquicamente a primeira camada que é definida pela norma ZigBee.

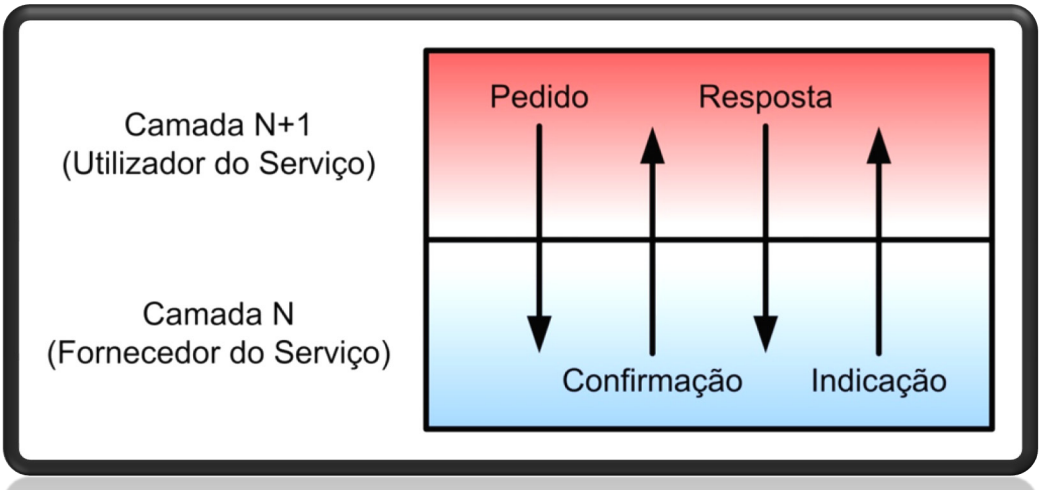

Figura 4.6 - Primitivas de serviço entre camadas

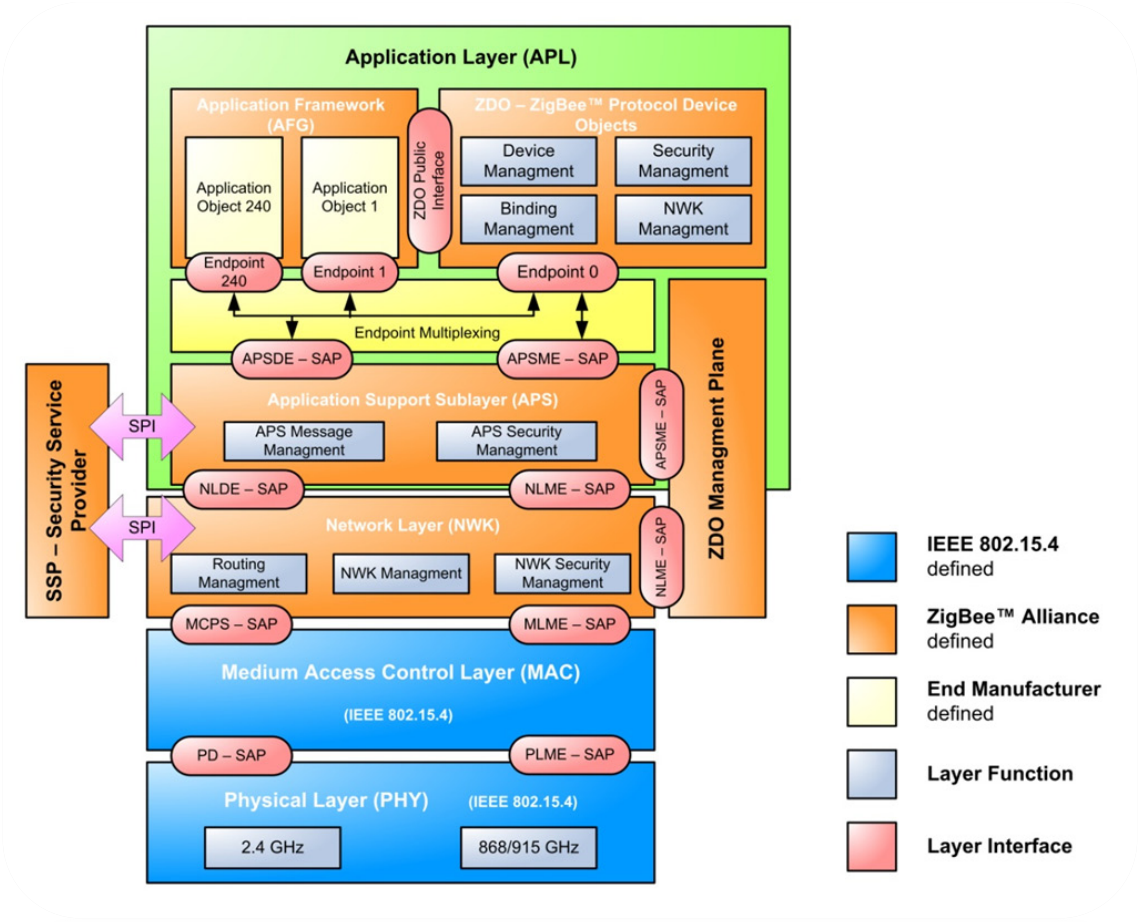

Figura 4.7 - Arquitectura Protocolar ZigBee

Na camada de aplicação (APL) são incluídas as subcamadas de suporte aplicacional (APS), o *ZigBee Device Object* (ZDO) e a *Application Framework* (AF). Esta última apresenta um conjunto de *Application Objects* (APO's) que criam um mapa dos nós vizinhos, facilitando, assim, a interacção entre eles. O ZDO tem o objectivo a descoberta de novos dispositivos na rede e os serviços por eles implementados. A subcamada de suporte aplicacional é unicamente responsável pela comunicação com a camada inferior, a camada de rede (NWK), quer seja a enviar dados das outras subcamadas da APL, quer a receber dados da camada NWK.

A camada APL pretende assegurar uma correcta gestão e suporte para as diversas aplicações.

Tanto a camada APS como a NWK podem ser encriptadas, para segurança da rede, através de um módulo de segurança (SSP – *Security Service Provider*).

A representação detalhada das camadas protocolares do ZigBee pode ser vista na figura 4.7.

4.3 Camada física

A camada PHY é responsável por controlar a interface entre a camada MAC com o meio de transmissão, activando e desactivando os *transcievers*.

A camada PHY é encarregue de detectar a energia, avaliar o estado do meio, seleccionar os canais pelos dispositivos, assim como, modular e desmodular os dados que são enviados e recebidos, respectivamente, sob a forma de sinais. Podendo operar em 27 canais diferentes, a camada PHY opera num canal determinado pelas camadas superiores.

Uma vez que nas ligações *wireless* o meio está a ser compartilhado por várias redes, a camada PHY tem de avaliar o estado dos canais e detectar o nível de energia quando um certo canal de transmissão estiver ocupado. A detecção de energia possibilita que os dispositivos tenham picos de energia, permitindo assim que a escolha do canal.

Os seus serviços de dados e administração às camadas superiores dirigidas pela PLME (*Physical Layer Management Entity*) com comandos vindo das camadas superiores para os AP's (Access Points).

Para a camada NWK, o pedido de detecção de energia é realizado pela notificação da camada MAC, que por sua vez, pede a detecção na sua secção PLME, da qual devera sair um byte que representa um valor associado que corresponde à linearidade de valores em decibéis (dB) da sensibilidade do receptor.

A norma IEEE 802.15.4 requer que os dispositivos tenham a capacidade de avaliar o estado do canal através de um processo designado por CCA (*Clear Channel Assessment*).

O processo CCA é dividido em três modos:

- Modo I – Modo de detecção de energia. Define níveis de energia acima dos limites em que o meio deve ser condicionado ou classificado como ocupado. Neste modo funcionam apenas as frequências de 2.4 GHz com sensibilidade de 75 dBm e 915 MHz com 82 dBm de sensibilidade.
- Modo II – Modo sensorial de portador. O CCA sinaliza o estado ocupado de um canal através de um sinal com a modulação e difusão do IEEE 802.15.4 e que pode ser maior ou menor que o nível ED.
- Modo III – Modo sensorial de portador com detecção de energia. É uma combinação dos modos anteriores. O CCA sinaliza o estado ocupado de um canal através de um sinal com a modulação e difusão do IEEE 802.15.4 e com energia abaixo do nível ED.

Na camada PHY, a transmissão de dados é feita com recurso à modulação de sinais O-QPSK nos canais de 2.4 GHz e BPSK nas outras frequências, em que os sinais são transmitidos através do modelo DSSS (*Direct Sequence Spread Spectrum*), no qual, o sinal a ser codificado deve ser multiplicado por um sinal pseudo-aliatório com uma frequência superior ao sinal original.

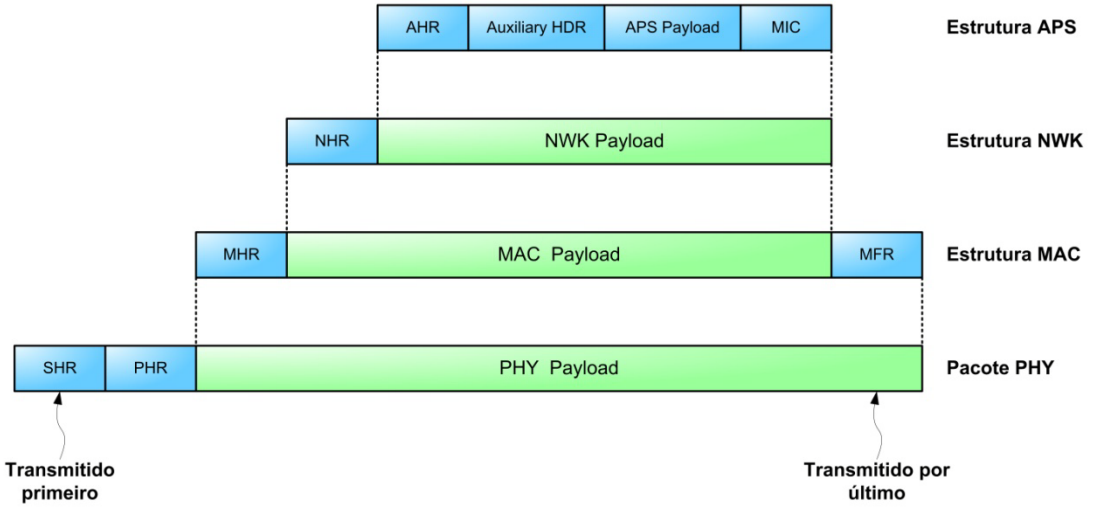

Figura 4.8 – Transformação da mensagem ao longo das várias camadas

Na específica implementação do DSSS na norma IEEE 802.15.4, a cada quatro bits de cada octeto do PPDU (PHY *Protocol Data Unit*), estes são agrupados formando um símbolo. Depois é usada uma tabela de procura que vai corresponder esse símbolo a uma sequência única de 32 bits, designada por sequência *chip* ou sequência *pseudorandom noise* (PN), a fim de reduzir os erros de transmissão.

A transmissão de dados é feita transmitindo, primeiro, o símbolo menos significativo e depois o mais significativo e transmitindo do primeiro para o último byte.

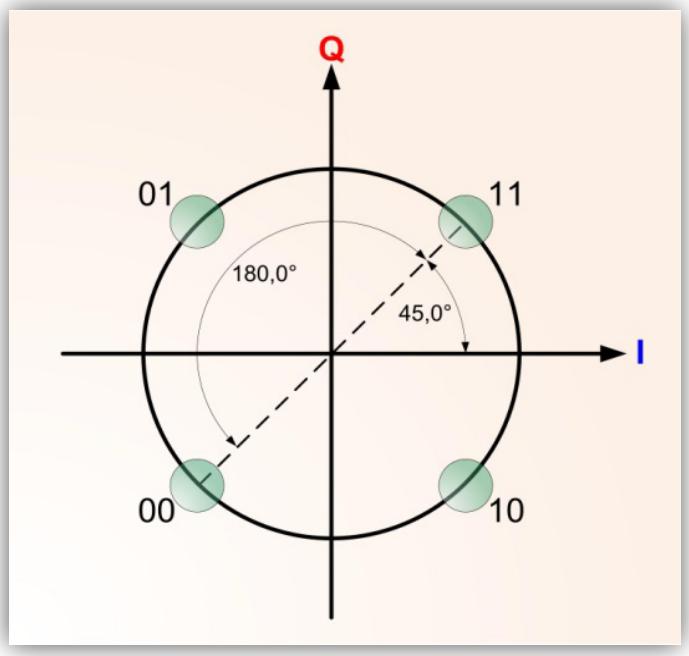

Figura 4.9 - Diagrama de constelação do QPSK

A modulação QPSK (*Quadrature Phase Shift Keying*) agrupa dois bits em blocos, isto é, 00, 01, 10 e 11. Cada bloco apresenta uma fase e ângulo, distribuídos igualmente entre si, que são transmitidos sinusoidalmente com frequências e amplitudes constantes. Como se tratasse de uma dupla transmissão em canais separados, o QPSK garante duplicidade da largura de banda, isto é, pode enviar dois bits de cada vez.

Utilizada nas frequências de 2.4 GHz a modulação O-QPSK é semelhante à QPSK, apresentando uma forma de meia onda sinusoidal, no entanto a diferença reside no facto de a fase ser dividida por dois, isto é, a onda é atrasada pelo período de um bit para evitar que as mudanças de fase sejam superiores a 90º. No fundo, o sinal é parecido com sinais binários para amplificar eficazmente o sinal, que resulta num baixo consumo energético.

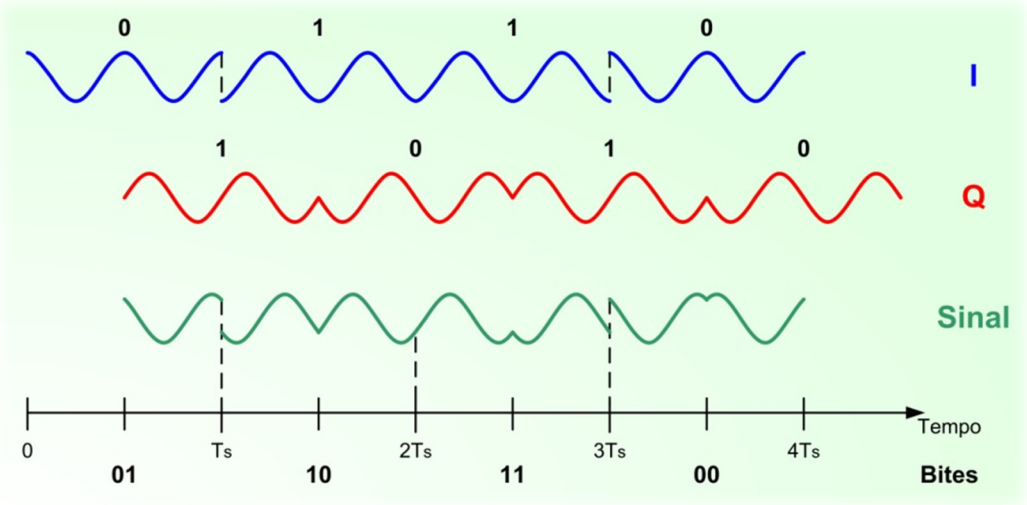

Figura 4.10 - Transmissão de sinais na modulação O-QPSK

4.4 Tipos de trama

Existem quatro tipos de tramas definidas pelo protocolo IEEE 802.15.4 para a camada MAC:

- Trama de *beacon.*
- Trama de dados.
- Trama de *acknowledgment*.
- Trama de comando MAC.

A trama *beacon* é usada pelo coordenador e router para transmitir *beacons*. As tramas de dados e *Acknowledge* são usadas para transmitir dados e reconhecer o sucesso da recepção de uma mensagem, respectivamente. Os comandos MAC, por sua vez, são transmitidos usando uma trama de comando MAC.

4.4.1 Trama de *Beacon*

A estrutura desta trama é esquematizada na figura 4.6. Aqui é possível verificar que toda a trama MAC é carregada no pacote PHY (PHY *Payload*). O conteúdo deste carregamento é referido como sendo o PSDU (PHY *Service Data Unit*).

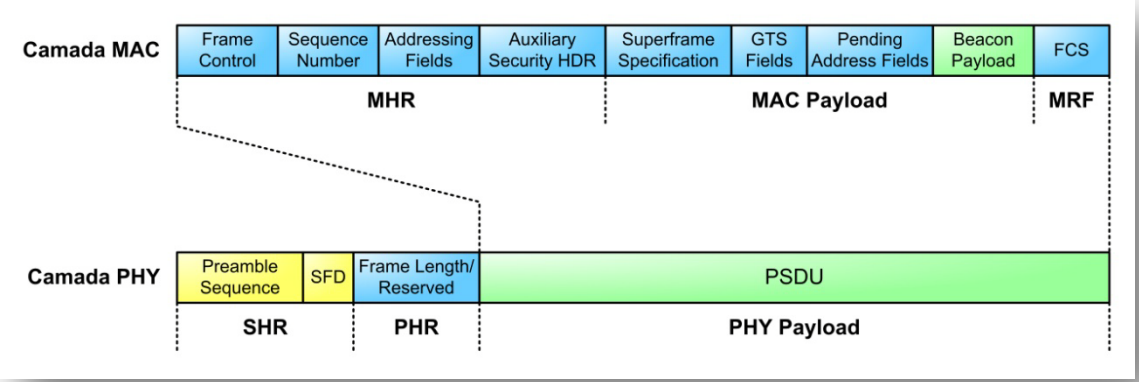

Figura 4.11 - Estrutura da trama MAC de *beacon*

No pacote PHY, o primeiro campo é utilizado pelo receptor para sincronização. O delimitador do inicio da trama (SFD) indica o fim da SHR (cabeçalho de sincronização) e o inicio da PHR (cabeçalho do PHY). O comprimento do PHR especifica o número total de octetos no PSDU.

A trama MAC é constituída por três secções distintas: o cabeçalho (MHR), o carregamento MAC (MAC *Payload*) e o rodapé (MRF). O campo de controlo (*Frame Control*) no MHR contém informação que define o tipo de trama, o campo de endereço e outras *flags* de controlo. O campo de número de sequência (*Sequence Number*) especifica o número de sequência da *beacon* (BSN). O campo de endereço fornece os endereços da fonte e do destino. O cabeçalho auxiliar de segurança (*Auxiliary Security* HDR) é opcional e contém informação requerida para o processo de segurança.

O carregamento MAC é fornecido pela camada NWK. A *superframe* é a trama delimitada por duas tramas *beacon*. A *superframe* é opcionalmente usada numa rede com *beacon* activado e ajuda a definir GTS's (*Guaranteed Time Slot*). O campo GTS, no carregamento MAC, determina quando um GTS é usado para receber ou transmitir.

A trama *beacon* não é apenas usada para sincronizar os dispositivos. Esta é, também, utilizada pelo coordenador para permitir que um determinado dispositivo na rede, saiba que existem dados pendentes para ele no coordenador. O dispositivo, a seu critério, irá contactar o coordenador e pedir-lhe que lhe transmita dados. Isto é designado por transmissão indirecta. O campo de endereço pendente (*Pending Address Fields*), no carregamento MAC, contém o endereço dos dispositivos que têm dados pendentes no coordenador. Sempre que um dispositivo recebe um *beacon*, ele vai verificar nesse campo se existem dados pendentes para ele.

O campo de carga *beacon* (*Beacon Payload*) é um acampo opcional, que pode ser usado pela camada NWK e é transmitido juntamente com a trama *beacon*. O receptor usa o campo de verificação da sequência da trama (FCS) para verificar quaisquer erros possíveis na trama recebida.

4.4.2 Trama de Dados

O carregamento de dados (*Data Payload*) é fornecido pela camada NWK. Os dados no carregamento MAC é designado MSDU (MAC *Service Data Unit*). Os campos da trama, como é possível verificar na figura 4.7, são similares aos da trama *beacon*, excepto na *superframe*, os campos GTS e de endereços pendentes (*Pending Address fields*) não estão presentes neste tipo de trama. A trama de dados MAC é designada por MPDU (MAC *Protocol Data Unit*) e é introduzida no carregamento PHY (PHY *Payload*).

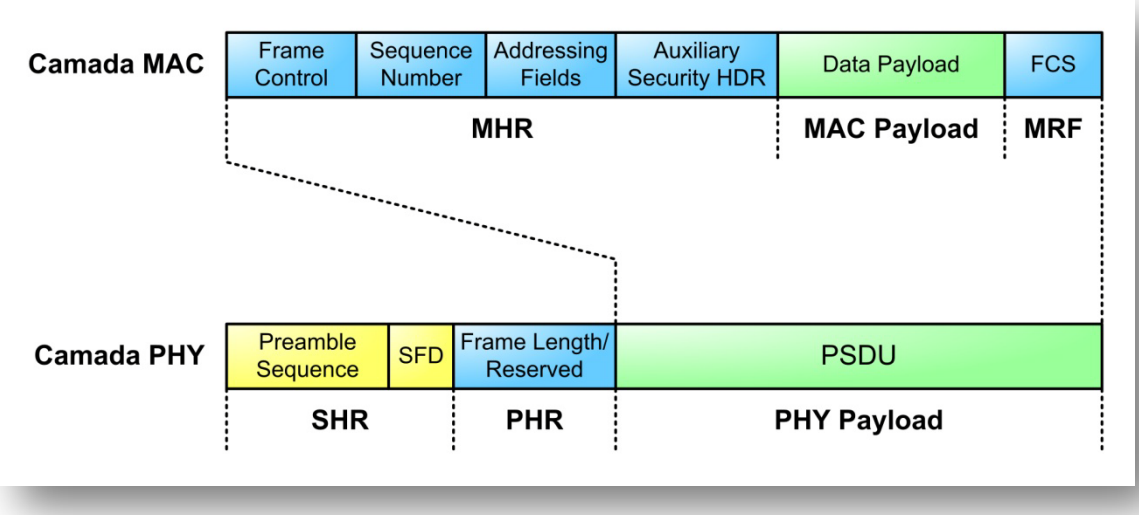

Figura 4.9 - Estrutura da trama MAC de dados

4.4.3 Trama de *Acknowledgment*

A trama MAC de *acknowledgment*, esquematizada na figura 4.8, é a trama MAC mais simples e não possui qualquer carregamento MAC (MAC *Payload*). Esta trama é enviada de entre dispositivos, para confirmar o sucesso da recepção de um pacote de dados.

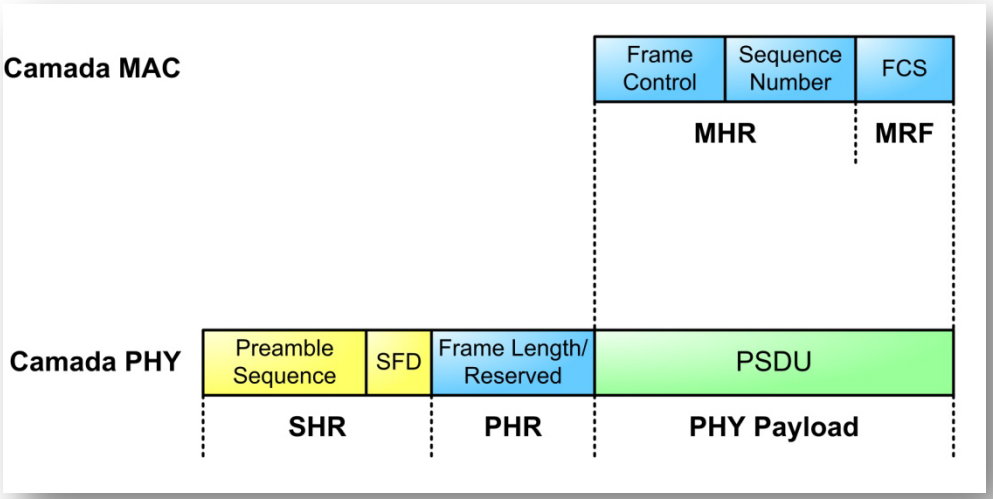

Figura 4.10 - Estrutura da trama MAC de acknowledgment

4.4.4 Trama de Comando

Os comandos MAC, tais como o pedido de associação ou desassociação com a rede, são transmitidos usando a trama MAC de comando, que pode ser vista na figura 4.9. O campo do tipo de comando (*Command Type*) determina o tipo de comando enviado, por exemplo pedido de associação ou pedido de dados. O carregamento de comando (*Command Payload*) contém o próprio comando. Toda a trama de comando MAC é passada para o PSDU.

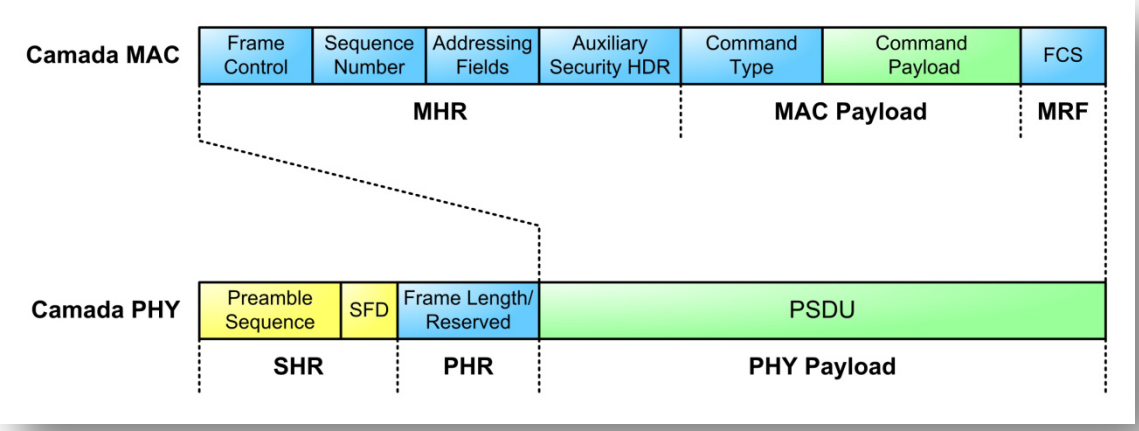

Figura 4.11 - Estrutura da trama MAC de comando

4.5 HARDWARE UTILIZADO

Para a implementação de uma ligação ZigBee foram montados dois circuitos (ver figura 4.13) em placa branca, um para o coordenador e outro para um *end device*. Utilizou-se o microprocessador PIC18LF4620 da Microchip para implementar a stack ZigBee, também da Microchip, e um pequeno programa que testa a comunicação entre os dois dispositivos. Para a transmissão por rádio é necessário um *transceiver* que trata da camada MAC e PHY. Neste caso utilizou-se o MRF24J40MA, que é uma plaquinha que já vem com o *transceiver* e toda a parte electrónica, por este, necessária, que comunica com o PIC por SPI (*Serial Port Interface*) e é controlada pelo PIC. Depois o transceiver é encarregue de enviar os dados, na modulação O-QPSK, pela antena.

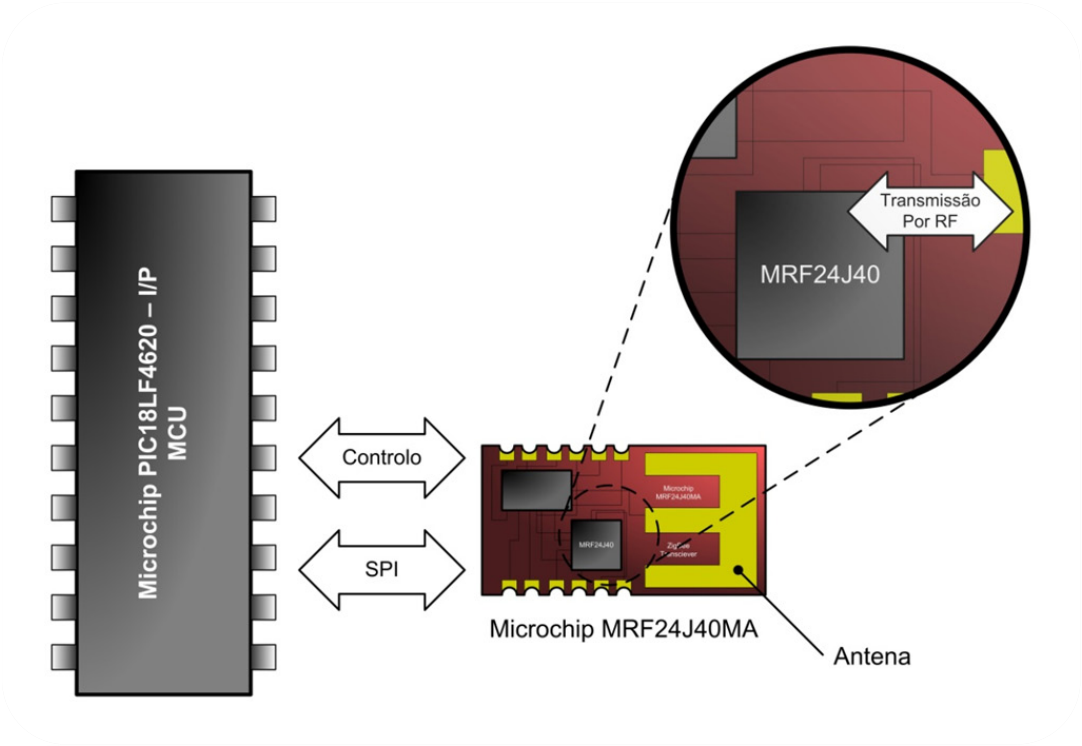

Figura 4.12 – Interface do PIC com o *transceiver*

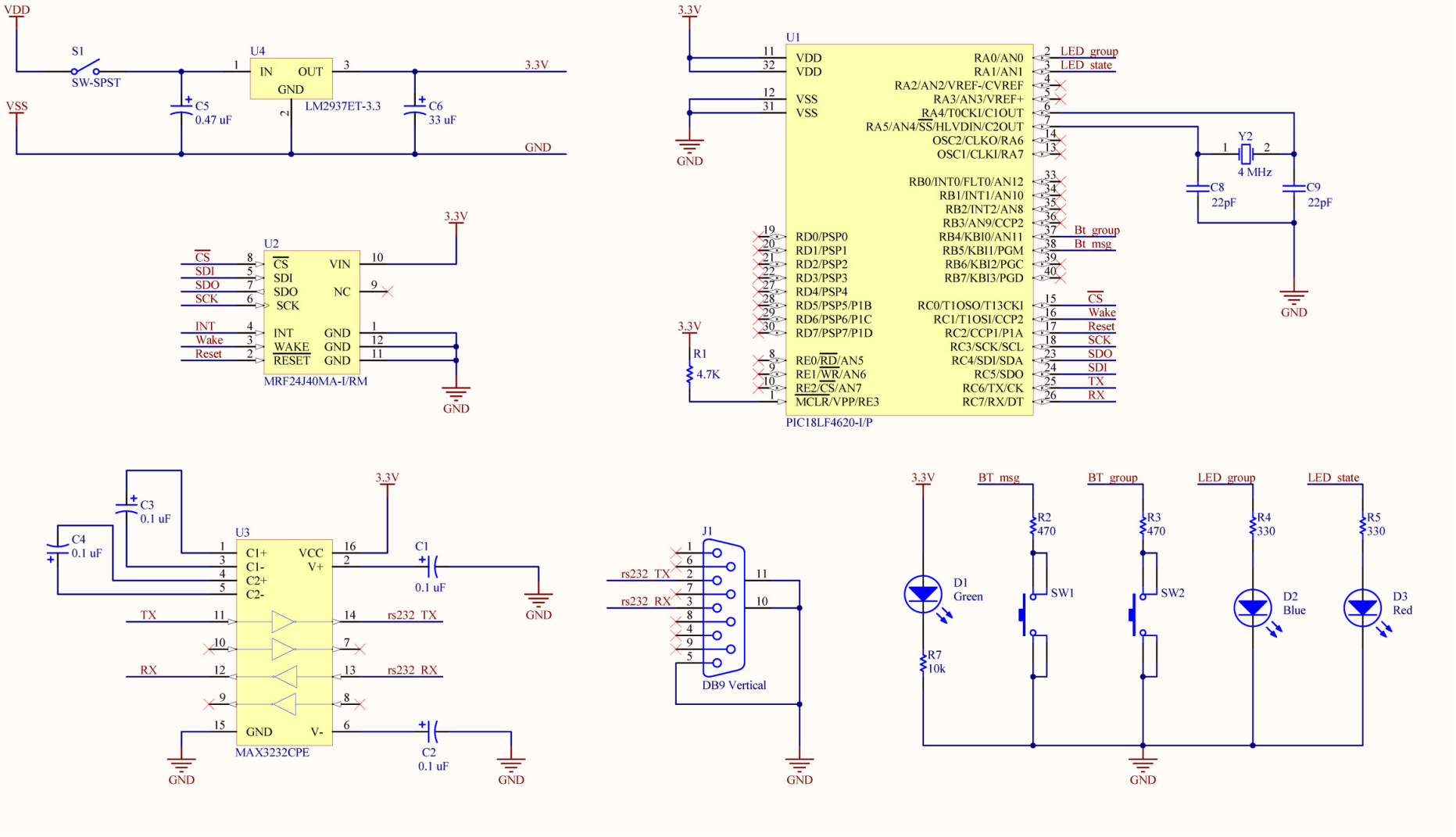

Figura 4.13 – Esquema eléctrico das placas

4.6 Programa

O programa foi realizado no MPLAB IDE e compilado pelo MPLAB C18 da Microchip.

4.6.1 Ficheiros

Como já foi referido, foi implementado a stack protocolar ZigBee da Microchip com algumas alterações. A *stack* é composta por vinte e quatro ficheiros:

- **SymbolTime.c, .h** – Contém funções de tempos para uso da *stack* ZigBee.
- **zAPL.h** – Cabeçalho da camada de aplicação da *stack*.
- **zAPS.c, .h** – Camada APS do protocolo ZigBee.
- **zTest.h** – Informação do perfil ZigBee ZCP. Estas alterações dependem do perfil.
- **zigbee.h** – Constantes genéricas do protocolo ZigBee.
- **ZigBeeTasks.c, .h** – Direcciona o programa pelas diversas camadas da *stack*.
- **zMAC.h** – Cabeçalho da camada MAC.
- **zMAC_MRF24J40.c, .h** – Camada MAC para o *transceiver* MRF24J40.
- **zNVM.c, .h** – Contém funções que escrevem e lêem na memória não volátil EEPROM (*Electrically-Erasable Programmable Read-Only Memory*).
- **zNWK.c, .h** – Camada NWK do protocolo ZigBee.
- **zPHY.h** – Cabeçalho da camada PHY.
- **zPHY_MRF24J40.c, .h** - Camada PHY para o *transceiver* MRF24J40.
- **zSecurity.h** – Cabeçalho da camada de segurança do protocolo ZigBee. (não utilizado)
- **zSecurity_MRF24J40.c, .h** - Camada de segurança para o *transceiver* MRF24J40. (não utilizado)
- **zZDO.c, .h** – Camada ZDO (ZDP) do protocolo ZigBee.

Alem destes ainda são incluídos mais nove ficheiros:

- **Compiler.h** – Contém definições para o compilador.
- **Console.c, .h** – Funções que permitem ler e escrever na porta série.
- **Generic.h** – Definições de constantes e tipos de variáveis e estruturas.
- **MSPI.c, .h** – Funções para a interface SPI.
- **sralloc.c, .h** – Funções de alocação dinâmica de memória.
- **Main.c** – Aplicação principal.
- **myZigBee.c** – Contém informação específica da aplicação.
- **zigbee.def** – Contém informação específica da aplicação.
- **zLink.lkr** – Script de ligação do projecto.

A aplicação principal começa por fazer a inicialização da USART, do *hardware* e da *stack* ZigBee. Depois inicia um ciclo infinito, onde, primeiro, faz um reset ao *watchdog*, depois determina qual a próxima primitiva ZigBee, seguidamente Processa a primitiva actual e por fim processa outras tarefas que não estejam relacionadas com o ZigBee.

Na rotina que processa a primitiva actual, se a primitiva actual não for nenhuma (NO_PRIMITIVE), vai, primeiro, verificar se estamos na sequência de inicio. Se for caso disso e for o coordenador, então vai tentar iniciar uma rede, se for um *end device*, vai tentar se juntar a uma rede. Se o coordenador já criou uma rede, então vai processar o menu pela porta série e seguidamente vai verificar o estado dos botões. No caso do *end device*, apenas verifica o estado do botões.

No caso do botão de adesão ao grupo, neste caso ao grupo 4, se o dispositivo já

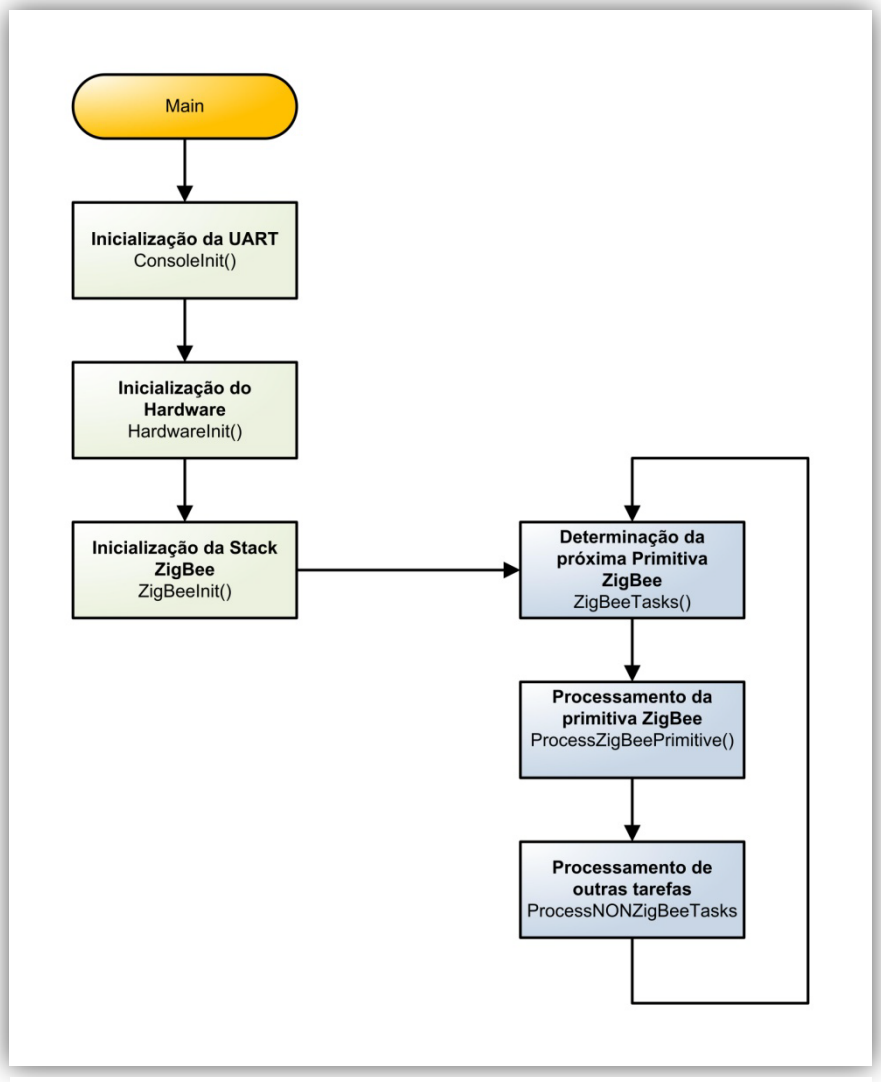

Figura 4.14 – Rotina do processo principal

pertencer ao grupo este é removido do mesmo, caso contrário o dispositivo é adicionado ao grupo. Um led respectivo indica o estado do dispositivo em relação ao grupo, isto é, se o dispositivo pertencer ao grupo o led está aceso, caso contrário está apagado. Notar, no entanto, que inicialmente o led está aceso, no entanto o dispositivo ainda não está no grupo.

No caso do botão de mensagem, este envia um pedido de dados de 10 bytes ao grupo 4, isto é em *broadcast*. Depois vai receber dados que correspondem ao estado do

outro dispositivo. Entretanto o outro dispositivo ao receber o pedido de dados vai alterar o seu estado indicando no led respectivo, se este estiver ligado, desliga-o e vice-versa.

4.6.2 Iniciação da rede

O coordenador quando é ligado, vai criar uma rede ZigBee, para isso vai ter que procurar em cada canal, dos que está permitido aceder, verificando se o canal está inactivo executando o CCA (*Clear Channel Assessment*), para poder criar uma rede nesse canal. Para o acesso aos canais o protocolo IEEE 802.15.4 utiliza o CSMA-CA (*Carrier Sense Multiple Access with Collision Avoidance*), existem dois tipos de CSMA-CA: *slotted* e *unslotted*. O *slotted* CSMA-CA executa o CSMA-CA enquanto houver uma estrutura *superframe*. A *superframe* divide o período activo em 16 aberturas de tempo iguais (*slots*). O período de *backoff* do algoritmo CSMA-CA precisa de ser alinhado com as aberturas de tempo. O unslotted CSMA-CA é utilizado quando não existe uma estrutura *superframe*, e por conseguinte, não é necessário o alinhamento do período de *backoff*. Uma rede *nonbeacon* utiliza sempre o algoritmo *unslotted* CSMA-CA para o acesso aos canais.

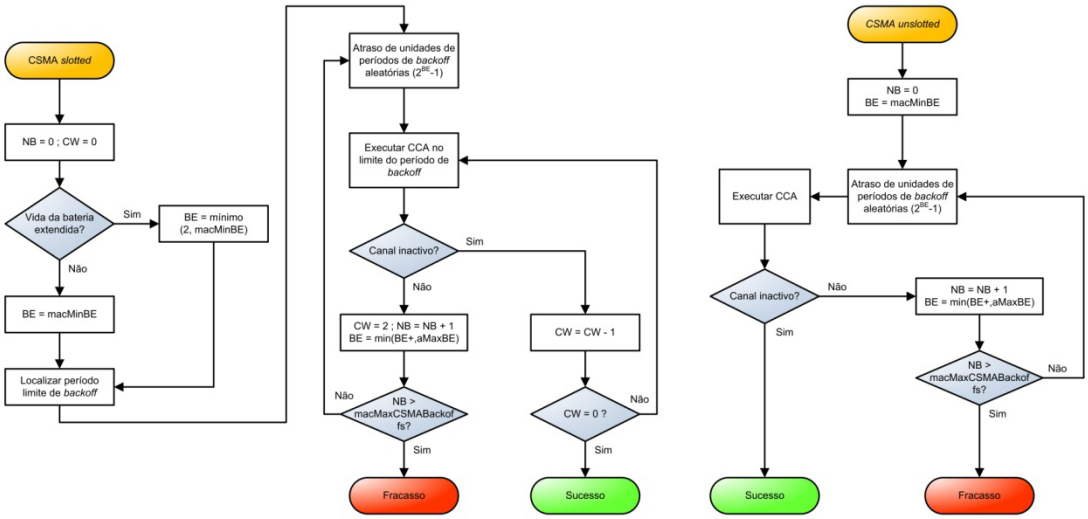

Figura 4.15 – Rotina CSMA-CA

Se o CCA indicar um canal ocupado, o dispositivo irá retirar-se por um período aleatório de tempo (*backoff period*) e depois tentará novamente. Este período aleatório de tempo é um inteiro múltiplo de unidades de períodos de *backoff*. A unidade de períodos de *backoff* é igual a uma constante da camada MAC **aUnitBackoffPeriod** (no caso da stack da Microchip equivale a 20).

4.6.3 Pedido de Associação

Para que um dispositivo, *end device* ou *router*, possa se juntar a uma rede necessita de realizar um pedido de associação ao coordenador. O dispositivo inicia o processo enviando uma mensagem de pedido de associação, depois de receber uma confirmação de que a sua mensagem chegou ao coordenador, espera um tempo de resposta enquanto o coordenador processa o pedido de associação. Depois vai pedir dados ao coordenador da resposta ao pedido de associação, depois de receber a resposta confirma ao coordenador que recebeu a resposta e vai processar essa resposta, ao mesmo tempo que o coordenador vai processar o estado resultante da acção de pedido de associação.

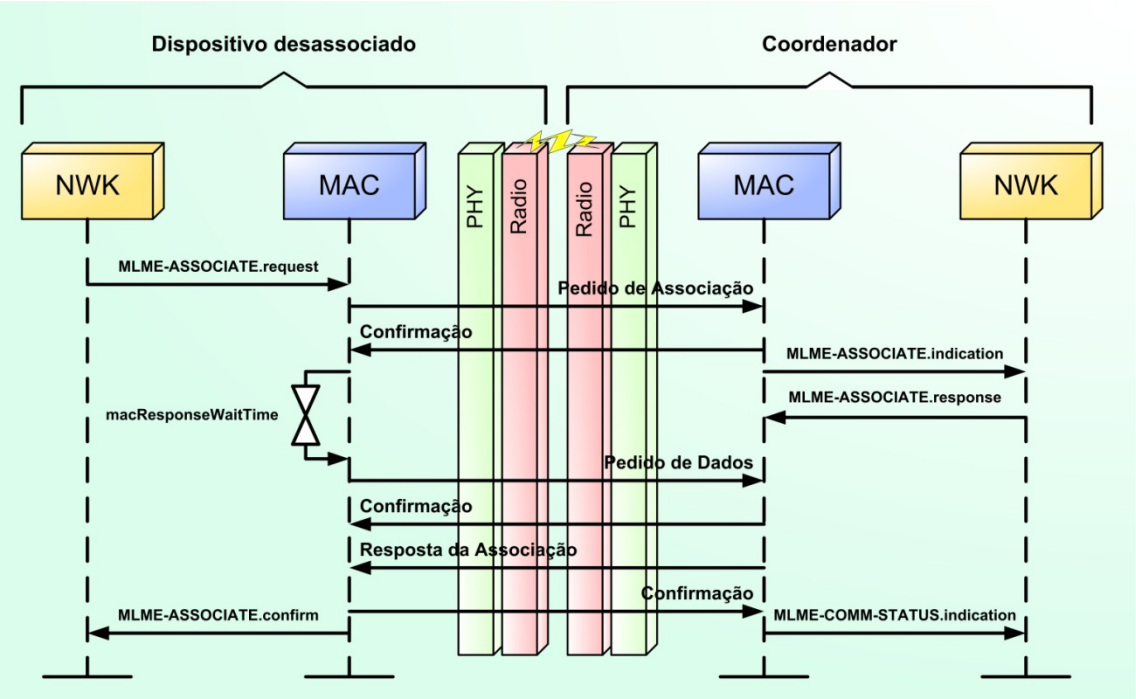

Figura 4.16 – Esquema do pedido de associação

No coordenador, o processamento ao pedido de associação faz-se da seguinte forma: Primeiro procura na tabela de vizinhança pelo endereço do dispositivo que pediu a associação, se estiver nessa tabela, a associação é bem sucedida. Caso contrário, vai verificar se a tabela está cheia, se estiver cheia a associação não é bem sucedida. Se não estiver cheia, Vai verificar que tipo de dispositivo fez o pedido de associação. Se for um *router* aumenta o número de *routers* na informação da tabela de vizinhança, seguidamente vai verificar se atingiu o limite máximo de *routers*, se atingir esse limite acciona uma *flag* que indica que já não podem ser associados mais routers. Se for um *end device* vai aumentar o número de *end devices* na informação da tabela de vizinhança, depois vai verificar se atingiu o limite máximo de *end devices*, se tal se verificar acciona uma *flag* que indica que já não podem ser associados mais *end devices*.

Depois disto vai adicionar estas informações todas à tabela de vizinhança e retorna ao processo principal.

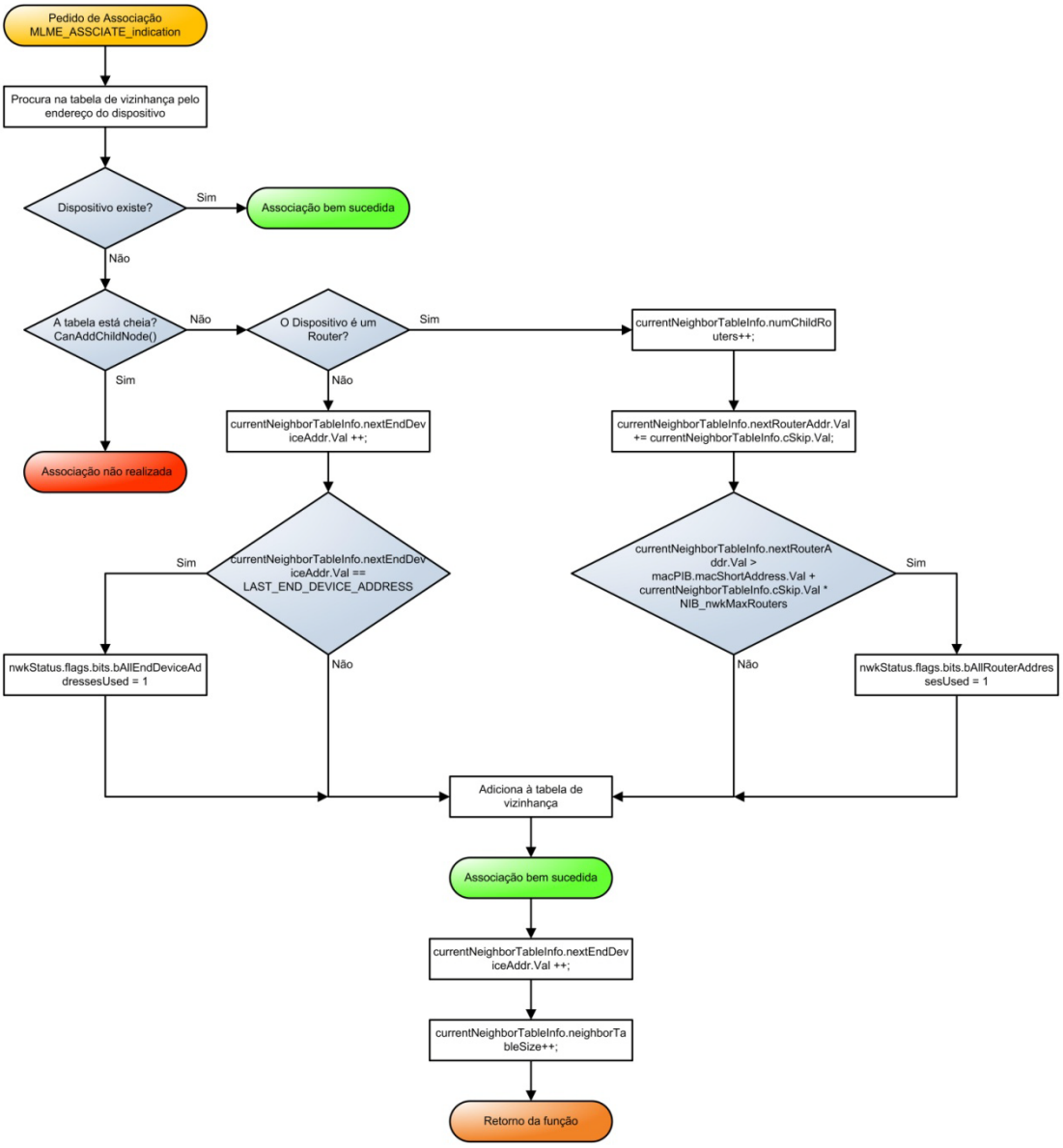

Figura 4.17 – Rotina de processamento do pedido de associação no coordenador

4.6.4 Exemplos do código

Não podendo escrever o código todo neste documento, devido a este ser muito extenso, escolhi alguns detalhes do código:

```
/* Envio de pedido de dados ao Grupo quando o butão é precionado */
if(PB_LEFT_SWITCH == 0)
{
    /* wait debounce time before taking any action again */
    /* espera pelo tempo de estabilização antes de realizar qualquer acção */
    if(PB_LEFT_pressed == FALSE)
    {
        PB_LEFT_pressed = TRUE;

        /* Envio de mensagem de Grupo para todos os Dispositivos */
        ZigBeeBlockTx();
        TxBuffer[TxData++] = 0x0a;      /* request 10-bytes */

        /* Modo de endereço de Grupo para pedido de envio */
        params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_GROUP;

        /* O GroupID MSB é zero neste exemplo */
        params.APSDE_DATA_request.DstAddress.ShortAddr.v[1] = 0x00;
        params.APSDE_DATA_request.DstAddress.ShortAddr.v[0] = GROUP_ID4;

        params.APSDE_DATA_request.SrcEndpoint               = GROUP_ID4;

        params.APSDE_DATA_request.RadiusCounter = DEFAULT_RADIUS;
        params.APSDE_DATA_request.DiscoverRoute = ROUTE_DISCOVERY_SUPPRESS;
        #ifdef I_SUPPORT_SECURITY
            params.APSDE_DATA_request.TxOptions.Val = 1;
        #else
        params.APSDE_DATA_request.TxOptions.Val = 0;
        #endif

        /* sem acknowledgment visto que isto é um pedido para todos (broadcast) */
        params.APSDE_DATA_request.TxOptions.bits.acknowledged = 0;
        params.APSDE_DATA_request.ProfileId.Val = 0x7f01;
        params.APSDE_DATA_request.ClusterId.Val = BUFFER_TEST_REQUEST_CLUSTER;
        currentPrimitive = APSDE_DATA_request;

        /* Inicio do tempo de estabilização */
        PB_LEFT_press_time = TickGet();

        break;
    }
}
else        /* Calculo do periodo do tempo de estabilização */
{
    TICK t = TickGet();
    tickDifference.Val = TickGetDiff(t,PB_LEFT_press_time);

    if(tickDifference.Val > DEBOUNCE_TIME)
    {
        PB_LEFT_pressed = FALSE;
    }

}
```

Figura 4.18 - Envio de mensagem de pedido de dados

```
/* Envio de pedido de dados ao Grupo quando o butão é precionado */
if(PB_LEFT_SWITCH == 0)
{
    /* wait debounce time before taking any action again */
    /* espera pelo tempo de estabilização antes de realizar qualquer acção */
    if(PB_LEFT_pressed == FALSE)
    {
        PB_LEFT_pressed = TRUE;

        /* Envio de mensagem de Grupo para todos os Dispositivos */
        ZigBeeBlockTx();
        TxBuffer[TxData++] = 0x0a;      /* request 10-bytes */

        /* Modo de endereço de Grupo para pedido de envio */
        params.APSDE_DATA_request.DstAddrMode = APS_ADDRESS_GROUP;

        /* O GroupID MSB é zero neste exemplo */
        params.APSDE_DATA_request.DstAddress.ShortAddr.v[1] = 0x00;
        params.APSDE_DATA_request.DstAddress.ShortAddr.v[0] = GROUP_ID4;

        params.APSDE_DATA_request.SrcEndpoint               = GROUP_ID4;

        params.APSDE_DATA_request.RadiusCounter = DEFAULT_RADIUS;
        params.APSDE_DATA_request.DiscoverRoute = ROUTE_DISCOVERY_SUPPRESS;
        #ifdef I_SUPPORT_SECURITY
            params.APSDE_DATA_request.TxOptions.Val = 1;
        #else
        params.APSDE_DATA_request.TxOptions.Val = 0;
        #endif

        /* sem acknowledgment visto que isto é um pedido para todos (broadcast) */
        params.APSDE_DATA_request.TxOptions.bits.acknowledged = 0;
        params.APSDE_DATA_request.ProfileId.Val = 0x7f01;
        params.APSDE_DATA_request.ClusterId.Val = BUFFER_TEST_REQUEST_CLUSTER;
        currentPrimitive = APSDE_DATA_request;

        /* Inicio do tempo de estabilização */
        PB_LEFT_press_time = TickGet();

        break;
    }
}
else        /* Calculo do periodo do tempo de estabilização */
{
    TICK t = TickGet();
    tickDifference.Val = TickGetDiff(t,PB_LEFT_press_time);

    if(tickDifference.Val > DEBOUNCE_TIME)
    {
        PB_LEFT_pressed = FALSE;
    }

}
```

Figura 4.19 - Recepção de mensagens

```
    /* Processa qunado não existe nenhuma Primitiva */
    if (currentPrimitive == NO_PRIMITIVE)
    {
        if (!ZigBeeStatus.flags.bits.bDataRequestComplete)
        {
            /* Não foram recebidos todos os dados do Pai (parent). Se não estivermos
               à espera de uma resposta a um pedido de dados, enviar um pedido de dados */
            if (!ZigBeeStatus.flags.bits.bRequestingData)
            {
                if (ZigBeeReady())
                {
                    /* O Pai pode ter dados para nos enviar */
                    params.NLME_SYNC_request.Track = FALSE;
                    currentPrimitive = NLME_SYNC_request;
                }
            }
        }
        else
        {
            if (!ZigBeeStatus.flags.bits.bHasBackgroundTasks )
            {
                /* Não existe nenhuma Primitiva para executar, todas as mensagens do Pai
                 * foram recebidas, a Stack não tem mais tarefas, e todos os processos
                 * especificos da aplicação foram completos. Agora entrar no modo Sleep.
                   Verificar se a UART está terminada, desligar o transceiver */

#if defined(BOTH_MICRO_TRANSV_SLEEP)
                if(!ZigBeeStatus.flags.bits.bRadioIsSleeping)
                {
                    while (!ConsoleIsPutReady());
                    INTCONbits.RBIE = 1;
                    MRF24J40Sleep();
                    SLEEP();
                    NOP();

                    /* Acordado pelo WDT, para pedir dados do Pai */
                    MRF24J40Wake();
                    params.NLME_SYNC_request.Track = FALSE;
                    currentPrimitive = NLME_SYNC_request;

                    GROUP_INDICATION    = !GROUP_INDICATION;
                    MESSAGE_INDICATION  = !MESSAGE_INDICATION;
                 }
#endif
#if !defined(BOTH_MICRO_TRANSV_SLEEP)
                {
                    TICK currentTime;

                    currentTime = TickGet();
                    if(SendingEDBRequest == 0)
                    {
                        if( ( TickGetDiff( currentTime, startPollingTime ) ) > (RFD_POLL_RATE))
                        {
                             params.NLME_SYNC_request.Track = FALSE;
                            currentPrimitive                = NLME_SYNC_request;

                            startPollingTime = TickGet();
                        }

                    }
                    else
                    {
                        if( ( TickGetDiff( currentTime, startPollingTime ) ) > (ONE_SECOND)/4)
                        {
                            params.NLME_SYNC_request.Track = FALSE;
                            currentPrimitive = NLME_SYNC_request;

                            GROUP_INDICATION    = !GROUP_INDICATION;
                            MESSAGE_INDICATION  = !MESSAGE_INDICATION;

                            startPollingTime = TickGet();
                        }


                    }


                }
#endif


            }
        }
    }
```

Figura 4.20 – Pedindo e recebendo dados num RFD