Comando e Monitorização de Sistemas de Actuação Via Redes Wireless – ZigBee

Carlos Manuel Oliveira Azevedo

Dissertação

Orientadores:

Prof. Manuel Rodrigues Quintas

Prof. Paulo Augusto Ferreira de Abreu

Mestrado Integrado em Engenharia Mecânica

Opção de Automação

Setembro de 2010

Aos meus pais,
Joaquim e Idalina.

Resumo

A comunicação Wireless apresenta uma importância cada vez mais activa nos dias de hoje. Diariamente milhões de pessoas em todo o mundo utilizam este meio de comunicação como processo de transferência de dados. Esta tecnologia pode ser implementada segundo vários protocolos, entre eles o protocolo ZigBee que se apresenta como uma forte aposta em aplicações industriais onde o controlo e a monitorização de sistemas aparecem como necessidades básicas.

O trabalho realizado teve como objectivo a criação e desenvolvimento de uma plataforma que permite o comando e a monitorização de um sistema de actuação utilizando o protocolo ZigBee como meio de comunicação Wireless entre o utilizador e o sistema de actuação.

Inicialmente é apresentada e justificada toda a estrutura de comunicação e comando implementada, relacionando os vários componentes de hardware com os meios de comunicação utilizados assim como os softwares envolvidos.

De seguida este trabalho aborda o protocolo utilizado na comunicação Wireless, descrevendo de forma sucinta algumas das suas características mais relevantes, como por exemplo as topologias por ele abrangidas e a estrutura das mensagens transmitidas.

Posteriormente é descrita e analisada a interface computacional criada, assim como os meios de comunicação cablados USB e USART utilizados.

Por fim é apresentada toda a estrutura de comando do sistema de actuação concluindo com a apresentação e discussão dos resultados obtidos nos diferentes ensaios realizados.

Abstract

Command and Monitoring of Actuation Systems Through Wireless Webs – ZigBee

Wireless communication represents, nowadays, a means of growing active importance. Daily, and world widely, millions of people use this means of communication as a process of transferring data. This technology can be implemented according to several protocols, among which the ZigBee protocol, which presents itself as a strong bet in industrial applications where systems control and monitoring appear as basic needs.

The work done aimed the creation and development of a platform which allows the command and monitoring of a performance system using the ZigBee protocol as a means of Wireless communication between the user and the actuation system.

To begin with, the whole structure of communication and command implemented is presented and justified, relating the several hardware components with the used means of communication, as well as the software used.

Then, this work approaches the protocol used in Wireless communication, describing some of its most relevant characteristics, such as the topologies it includes and the structure of the transmitted messages.

The computational interface created is described and analyzed later, such as the wired USB and USART used means of communication.

Finally, the whole command structure of the performance system is presented and the presentation and discussion of the results obtained in the different accomplished tests concludes the work.

Agradecimentos

Durante a realização deste projecto pude contar com o apoio e contributo de professores, amigos e familiares sem os quais este trabalho não teria sido realizado de forma tão positiva.

Gostaria de agradecer em primeiro lugar aos meus orientadores, Professor Manuel Quintas e Professor Paulo Abreu por toda a disponibilidade e apoio demonstrado nas muitas horas que despenderam durante a realização deste projecto. Pela partilha de experiências e conhecimentos que em muito contribuíram para a construção deste trabalho. Por toda a paciência e tolerância demonstrada nas horas de maior dificuldade, muito obrigado.

Ao coordenador da opção de Automação, Professor Francisco Freitas agradeço pelo acompanhamento e conselhos em tudo construtivos, facultados durante este Período.

Quero também agradecer a todos os meus colegas e amigos que de alguma forma contribuíram com o seu apoio, em particular ao Tiago Faustino pela disponibilidade e auxílio na percepção de vários assuntos abordados neste trabalho, aos meus colegas da opção de Automação com quem partilhei durante meses momentos de alegria, desespero e muita boa disposição, à Brigitte e ao Nuno pelo apoio e ajuda nas traduções.

À minha namorada Patrícia, quero agradecer com especial carinho, pelo seu apoio, amizade e compreensão, tendo-se tornado uma ajuda fundamental para a realização deste projecto.

Por último, mas não menos importante, quero agradecer à minha família, em especial aos meus pais que são para mim um exemplo de coragem e perseverança, pois sem eles nunca conseguiria chegar tão longe, aos meus irmãos por todo o incentivo que me deram durante todo o meu percurso académico.

Índice de Conteúdo

Índice de Figuras

Índice de Tabelas

Capítulo 1

Introdução

Neste capítulo são abordadas as razões que levaram à realização e desenvolvimento deste trabalho. É também feita a contextualização da tecnologia Wireless no domínio da engenharia em geral e da automação em particular. Para isso são apresentadas e comparadas algumas formas que esta tecnologia pode assumir assim como as suas classes de aplicações. Por fim são descritos os objectivos propostos para o trabalho.

1.1 Motivação

Uma rede Wireless, como o próprio nome indica, possibilita a comunicação entre dois ou mais dispositivos sem o uso de cablagem, seja esta telefónica, coaxial ou óptica. É comum empregar este tipo de tecnologia em redes de computadores, normalmente utilizada como forma de acesso à Internet, podendo no entanto ser utilizada em muitas outras aplicações. As redes sem fios surgem em várias dimensões físicas, podendo mesmo alcançar áreas de elevada extensão.

A utilização de meios de comunicação ausentes de cablagem torna qualquer sistema muito mais interessante do ponto de vista da sua portabilidade.

O protocolo ZigBee é um protocolo de comunicação Wireless concebido para taxas de transmissão de dados baixas e um hardware de suporte relativamente barato. Este protocolo é baseado na norma IEEE 802.15.4, permitindo uma comunicação global e confiável. O

protocolo ZigBee suporta diferentes topologias de rede Wireless, sendo uma boa alternativa a outros protocolos para aplicações de baixo alcance e baixo consumo energético.

Actualmente existem áreas, como a domótica, interessadas em aplicar as tecnologias acima referidas no controlo e comando de mecanismos. Desta forma, acções como a abertura e fecho de portões, o controlo de sistemas de rega avançados, o controlo em posição de um painel solar ou simplesmente a abertura e fecho de um estore tornaram-se realizáveis de forma automática.

Com tudo isto, é então necessário responder positivamente às exigências do avanço tecnológico, não esquecendo a competitividade do mercado e a essencialidade de minimização do custo que este avanço requer.

1.2 Contextualização

A cada dia que passa o controlo Wireless assume uma importância maior na vida comunitária. São inúmeras as aplicações às quais esta tecnologia pode responder. Tarefas simples como abrir um portão, fechar uma janela, tornam-se cada vez mais rápidas e cómodas com o desenvolvimento tecnológico.

O desenvolvimento desta tecnologia levou a que muitas das operações, que antes eram suportadas à base de cablagem, se tornassem agora fisicamente mais independentes e expansíveis. Na figura 1.1 são apresentados alguns exemplos de aplicações industriais de controlo Wireless.

Figura 1.1 - Exemplos de aplicações industriais de controlo Wireless

Algumas aplicações industriais exigem uma comunicação bidireccional entre o sistema e o supervisor; são disso exemplo o controlo da pressão de uma instalação hidráulica ou de uma conduta de gás. A utilização de uma rede sem fios torna possível não só o controlo como a

monitorização de sistemas. Desta forma o utilizador de uma rede Wireless pode controlar uma determinada aplicação e observar/analisar a sua resposta temporal.

A comunicação Wireless não está presente apenas no âmbito industrial. Existem actualmente brinquedos equipados com pequenas redes Wireless de forma a suportar toda a sua comunicação. Na figura 1.2 é possível observar alguns brinquedos que utilizam a tecnologia Wireless.

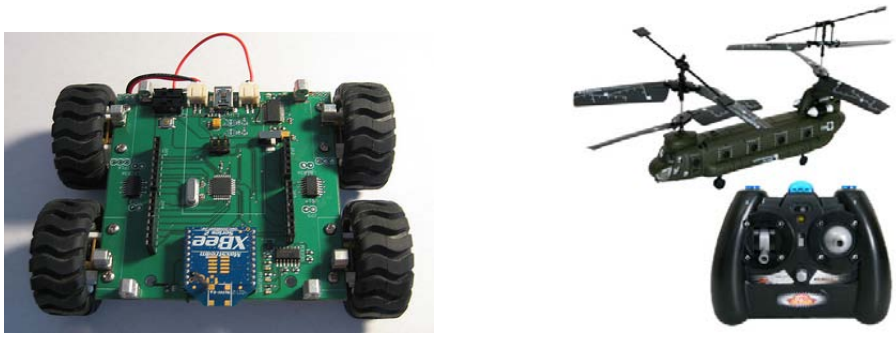


Figura 1.2 - Exemplos de brinquedos que utilizam a tecnologia Wireless

O controlo Wireless apresenta assim soluções robustas e com uma taxa eficiência/custo elevada para uma vasta gama de aplicações automáticas.

Wireless é um conceito um pouco vasto quando se fala de comunicação. Por este motivo existe a necessidade de abordar as diferentes tecnologias que ele pode possuir quando se introduz este tema.

Diariamente, cada pessoa utiliza inúmeras tecnologias de comunicação sem fios, cada uma delas com características próprias, utilizadas em aplicações bem distintas.

1.3 Tecnologias Wireless

ZigBee

ZigBee é um protocolo com o *layer* físico baseado na norma IEEE 802.15.4. Opera em várias frequências incluindo a banda não licenciada dos 2.4 GHz, que é usada pela maioria dos dispositivos Wi-Fi e Bluetooth, oferecendo um comparável ou ligeiramente maior alcance (10m - 100 m), embora com uma mais baixa taxa de transmissão de dados (20-250 Kbps).

A principal vantagem deste protocolo é o baixo consumo. Esta característica deve-se à capacidade que os dispositivos ZigBee têm para permanecerem em modo “adormecido” a maior parte do tempo. O consumo de energia é reduzido fazendo com que seja sempre possível que os aparelhos consigam trabalhar com a mesma bateria durante anos [1].

Este protocolo é principalmente implementado em aplicações industriais de controlo e monitorização de sistemas.

Bluetooh

O Bluetooh é um protocolo com o *layer* físico baseado na norma IEEE 802.15.1. O Bluetooh 2.0 suporta velocidades de comunicação até 2,1 Mbps. As mais comuns implementações são as de baixo consumo cujo alcance pode chegar a 1m ou aos 10 m dependendo da classe de potência [2]. Os dispositivos equipados com esta tecnologia podem também operar na banda não licenciada dos 2.4 GHz.

Os dispositivos Bluetooth requerem muito menos potência que os Wi-Fi, mas a área por eles alcançada e as taxas de transmissão de dados são igualmente muito baixas. Este alcance limitado, que alguns podem considerar uma desvantagem, torna-se uma vantagem uma vez que a mesma frequência pode ser reutilizada em áreas próximas.

Actualmente o Bluetooth é fortemente utilizado como meio de transferência de dados entre computadores ou telemóveis.

UWB

Ultra-Wideband (UWB) é uma tecnologia onde a comunicação é enviada por ondas magnéticas através de pulsações de curta duração, em vez da usual modelação dos transportadores de onda sinusoidal [3].

O UWB pode atingir taxas até 500 Mbps num alcance de 2 m ou 110 Mbps num alcance de 10 m. Esta tecnologia opera na mesma banda de frequência que outros sistemas de comunicação sem interferência significativa.

O UWB é utilizado numa variedade elevada de componentes electrónicos como televisões de alta definição, MP3's e LCD's. É esperado que o futuro USB Wireless seja baseado no UWB.

Wi-Fi

O Wi-Fi é uma tecnologia baseada nas especificações da norma IEEE 802.11b,g. Esta tecnologia suporta velocidades que podem chegar aos 11 Mbps para comunicações interiores até 30 m ou podendo chegar aos 54 Mbps para comunicações exteriores até 90 m.

Os dispositivos que utilizam esta comunicação operam na banda não licenciada de 2,4 GHz de modo que poderá existir elevada interferência entre estes dispositivos, bem como com satélites, fornos de microondas e telemóveis. Poderá também ser utilizada a banda de 5 GHz

que reduz significativamente as interferências [4] podendo no entanto aumentar a dificuldade de comunicar através de paredes, diminuindo assim o seu alcance em ambientes interiores.

O Wi-Fi é sobretudo utilizado como forma de acesso à internet a partir de computadores pessoais, contudo já existem operadoras telefónicas a investir nesta tecnologia.

WiMax

WiMax é uma tecnologia Wireless recente e está actualmente a ser discutida no grupo IEEE 801.16. Esta tecnologia é caracterizada pelo seu longo alcance, que pode mesmo chegar aos 50 km suportando uma velocidade de transmissão de dados de 75 Mbps por canal. Combinando vários canais, o WiMax pode fornecer larguras de banda até 350 Mbps.

Originalmente esta tecnologia utilizava as bandas de 10 a 66 GHz mas no futuro, através da norma IEEE 801.16a, poderá operar nas bandas licenciadas e não licenciadas de 2 a 11 GHz.

A utilização das bandas de baixas de frequência permite ultrapassar com maior facilidade obstáculos não metálicos como paredes, permitindo uma comunicação de maior alcance [5].

Na tabela 1.1 é possível observar algumas características de base das diferentes tecnologias abordadas neste capítulo.

Tecnologia	ZigBee	Bluetooh	UWB	Wi-Fi	WiMax
Norma	IEEE 802.15.4	IEEE 802.15.1	IEEE 802.15.3a	IEEE 802.11b,g	IEEE 801.16
Banda de Frequência	868/915 MHz, 2,4 GHz	2,4 GHz	3,1 GHz-10,6 GHz	2,4 GHz	10 a 66 GHz
Velocidade de transmissão	250 Kbps	2,1 Mbps	500 Mbps	54 Mbps	350 Mbps
Alcance	100 m	10/100 m	10 m	90 m	50 Km
Duração da Bateria	Anos	Dias	Horas	Horas	-
Aplicações	Controlo e monitorização	Telemóveis e Computadores	Televisões e LCD's	Lan Wireless e Internet	Wan Wireless

Tabela 1.1 - Comparação entre ZigBee, Bluetooh, UWB e Wi-Fi e WiMax

1.4 Classes de aplicações

A automação fornece um conjunto de soluções cada vez mas vasto mesmo para aplicações Wireless de dimensões físicas surpreendentes. Cada sistema automático possui requisitos específicos da aplicação para a qual foi concebido.

Quando se pretende automatizar uma unidade terminal remota cujo objectivo é monitorizar e comandar a longa distância uma quantidade elevada de variáveis em cada minuto, não se pode ter presente os mesmos padrões que quando se pretende controlar um servomotor DC utilizando sinais PWM. Os requisitos para estas duas aplicações são muito diferentes.

Dois dos maiores factores que influenciam a tecnologia a utilizar num sistema a automatizar é a distância relativa a que se encontram os órgãos terminais da rede e a largura de banda necessária para satisfazer os requisitos da aplicação. O aumento da distância relativa induz por vezes o aparecimento de barreiras físicas que têm de ser ultrapassadas. É então necessário, quando se pretende automatizar um sistema, enumerar os requisitos de comunicação. Estes requisitos são normalmente relacionados com aplicações típicas agrupadas nas seguintes classes [2]:

***Very loosely coupled* – Distâncias ultra-elevadas**

Nesta classe estão agrupadas as aplicações onde as tecnologias Wireless são necessariamente utilizadas devido à elevada distância que separa os dispositivos implementados para realizar a comunicação. Tipicamente podem fazer parte desta classe, entre outras, aplicações de monitorização e controle de parques eólicos, bombas de abastecimento de água e sistemas de alarme a elevada distância.

***Loosely coupled* – Distâncias elevadas**

Esta classe corresponde a aplicações onde existe uma grande quantidade de dispositivos, tipicamente sensores. Estes equipamentos destinam-se normalmente à monitorização e controlo de variáveis.

Podem ser englobadas nesta classe aplicações como a ventilação ou aquecimento de edifícios. Normalmente as aplicações desta classe envolvem quantidades e cadências de actualização de dados baixas.

As grandes vantagens destes sistemas Wireless são: a redução significativa da cablagem necessária e consequente redução dos custos de implementação, assim como a flexibilidade de posicionamento ou reposicionamento dos vários elementos da rede.

Estes sistemas devem possuir um baixo consumo energético capaz de alimentar todo o sistema durante anos utilizando pequenas baterias.

***Normally coupled* – Distâncias Intermédias**

Esta classe corresponde a aplicações onde a largura de banda necessária é maior do que na classe anterior, aplicações estas onde o baixo consumo de energia não é uma prioridade e a distância de comunicação varia entre os 10 m e os 100 m. Pode-se englobar nesta classe aplicações onde não exista um compromisso rigoroso na comunicação em tempo real, por exemplo quando se pretende carregar um novo código para uma máquina CNC.

***Closely coupled* – Pequenas distâncias**

Nesta classe são agrupadas as aplicações onde é necessário realizar comunicações entre dispositivos fisicamente próximos. O alcance destas aplicações é tipicamente 10 metros e a troca de dados é realizada com restrições de tempo pequenas. Nesta classe encontram-se aplicações de supervisionamento ou comando, como por exemplo um PDA equipado com monitores de supervisão e de diagnóstico do estado de um determinado equipamento ou um software configurado para definir os valores de um controlador de motor.

***Tightly coupled* – Curtas distâncias**

Esta classe corresponde a aplicações onde, apesar da distância entre os dispositivos ser curta, é necessária uma largura de banda elevada e de alta qualidade. Nesta classe consideram-se redes que funcionam normalmente dentro de uma máquina, por exemplo, dentro de uma máquina CNC ou de um robô móvel onde existem várias malhas de controlo de motores. O alcance destas aplicações é pequeno, normalmente inferior a 1 m. Para evitar interferências externas, o equipamento que contém os dispositivos Wireless pode operar, se necessário, dentro de uma gaiola de Faraday.

Na figura 1.3 é possível observar graficamente a relação que existe entre as várias tecnologias abordadas neste capítulo no que diz respeito ao tipo de classe e ao seu alcance.

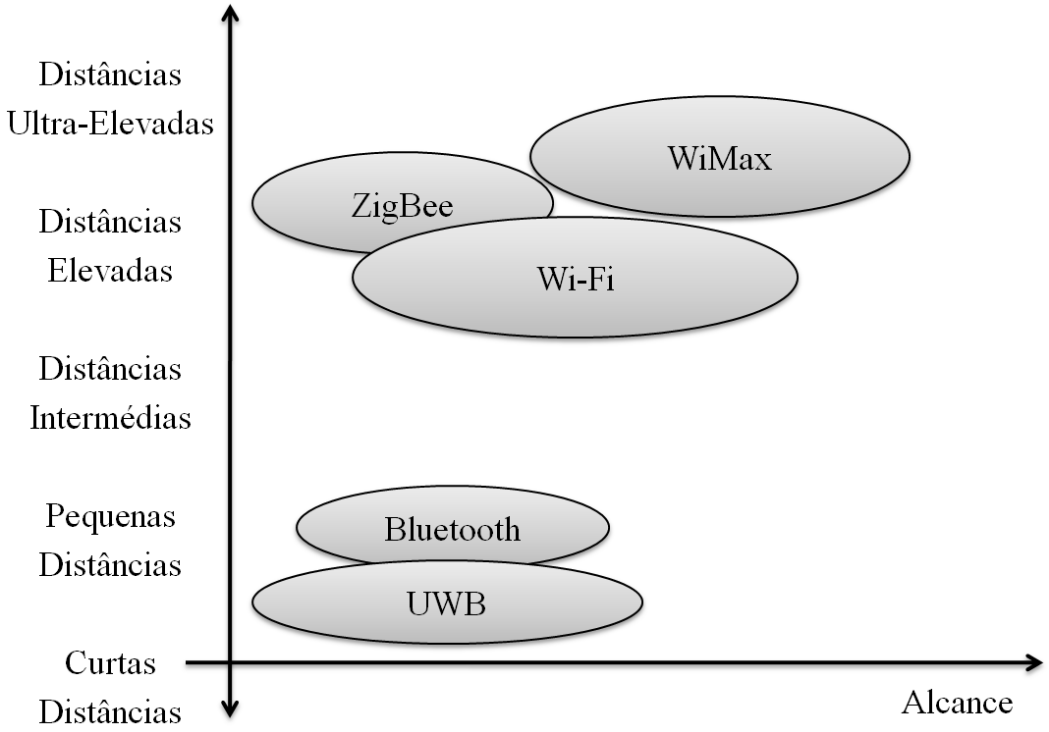


Figura 1.3 - Gráfico comparativo de soluções Wireless

O Wireless é então uma tecnologia com forte aplicação na área da comunicação. Uma tecnologia embora dominada, ainda em constante desenvolvimento. Desta forma, o Wireless torna-se algo cada vez mais presente e fundamental no quotidiano de milhões de pessoas.

São muitas as tecnologias com suporte na comunicação sem fios. Cada tecnologia possui uma gama de aplicações bem definida. Desta forma, a escolha de qual tecnologia a implementar em determinada aplicação passa pela definição correcta dos seus requisitos e classificação face a uma classe de aplicações.

1.5 Objectivos

O principal objectivo deste trabalho é a criação e desenvolvimento de uma plataforma que permita o comando e a monitorização de um sistema de actuação utilizando a tecnologia Wireless, suportada pelo protocolo ZigBee, como meio de transmissão de dados.

A plataforma que sustentará toda esta comunicação bidireccional pode ser dividida em duas componentes: uma componente física, englobando todo o hardware essencial à comunicação e o sistema de actuação criado e desenvolvido no laboratório, e uma componente virtual, na qual estará implícita a programação de componentes e a análise de redes, de forma a sustentar e implementar os protocolos de comunicação.

A interface entre o utilizador e o sistema é efectuada por meio de uma aplicação informática desenvolvida para o efeito. Esta plataforma deverá permitir o comando e a monitorização das acções do sistema de actuação.

Resultados esperados:

- Criação de um hardware, com suporte USB, para comunicação entre um computador e o microcontrolador;
- Criação de um hardware com suporte para comunicação Wireless;
- Comando e monitorização das acções do sistema de actuação;
- Programação em MicroC e MPLAB C18 dos diferentes dispositivos integrados no sistema;
- Criação de uma interface computacional para comando e monitorização das acções do sistema de actuação.

Capítulo 2

Arquitectura Geral Implementada

A arquitectura a implementar num projecto de comunicação, comando e monitorização de sistemas de actuação merece sempre bastante cuidado e atenção. Ela define quais os tipos de comunicação a utilizar assim como as técnicas de comando e os componentes electrónicos necessários à sua implementação.

Este capítulo é dedicado a toda a arquitectura de comunicação e comando implementada neste projecto. Pretende-se então evidenciar toda a estrutura de comunicação assim como o hardware e software utilizado.

2.1 Estrutura de comunicação

A estrutura utilizada para o transporte da informação desde a origem até ao destino varia consoante a sua aplicação. Neste caso em particular, os dados utilizados no comando e monitorização das acções do sistema de actuação são transportados através de diferentes tipos de comunicação. Na figura 2.1 está representado um diagrama onde é possível visualizar de forma simplificada a estrutura de comunicação base levada a cabo neste projecto.

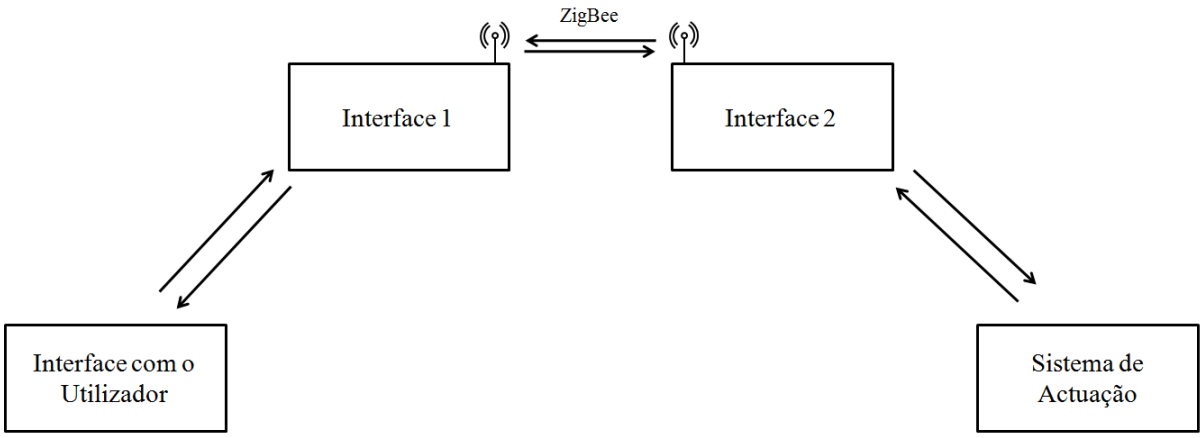


Figura 2.1 - Esquema base da estrutura de comunicação implementada

O utilizador interage com o sistema por meio de uma interface terminal. Esta interface pode adquirir vários formatos, passando por simples consolas lógicas até aos sistemas mais complexos do domínio computacional. Toda a informação de comando é transportada por vários meios de comunicação até ao sistema de actuação. Por sua vez, os dispositivos sensoriais associados ao sistema permitem a aquisição da resposta do mesmo face às ordens de comando, executando o envio da informação através da rede de comunicação e permitindo deste modo ao utilizador monitorizar as consequências das acções por ele impostas.

Substituindo os órgãos terminais pelos dispositivos objectivados inicialmente para este projecto, é possível esquematizar a estrutura anterior na forma apresentada na figura 2.2:

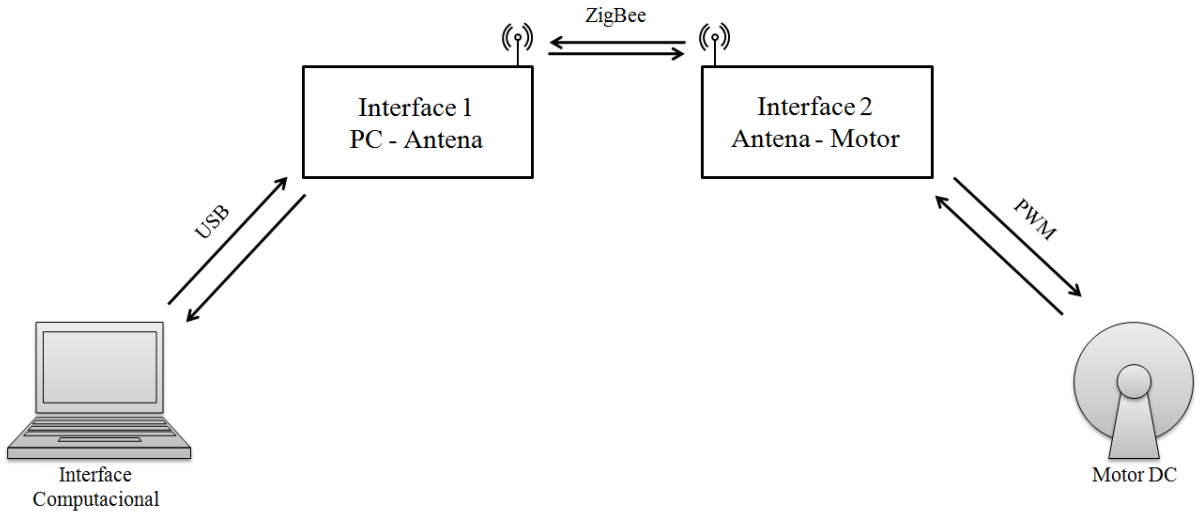


Figura 2.2 - Estrutura de comunicação implementada

O sistema de actuação utilizado consiste num motor DC de íman permanente equipado com um encoder incremental. Devido à inexistência de catálogo do motor fornecido pelo

fabricante (Computer Optical Products, Inc.), algumas das seguintes características foram obtidas de forma experimental:

- Motor DC de ímanes permanentes
- Tensão nominal: 24V
- Corrente em vazio: 120 mA
- Velocidade em vazio: 4500 rpm
- Resolução do encoder: 400 pps

Na figura 2.3 é apresentado o motor utilizado na implementação do sistema de actuação.

Figura 2.3 - Motor DC de ímanes permanentes utilizado

O comando e a monitorização das acções do motor são realizados através de uma interface computacional onde o operador pode actuar sobre o sistema e visualizar o resultado das suas acções.

É possível visualizar ainda na figura 2.2 a existência de duas interfaces. A primeira é responsável pela comunicação entre a antena e o computador sendo, por sua vez, a segunda responsável por receber e enviar os dados de ordem e monitorização do sistema entre a antena e o motor DC. A comunicação entre as duas interfaces, fisicamente separadas, é realizada por antenas transmissoras utilizando o protocolo ZigBee.

O comando do motor é executado com um sinal em PWM disponibilizado pela interface 2. Esta interface possui um microcontrolador que controla as acções do motor. O sinal do encoder associado ao eixo do motor, que mede a sua posição angular e a sua velocidade, é adquirido pelo microcontrolador ficando assim disponível para ser enviado (via ZigBee) para a interface computacional.

Descendo a um nível mais pormenorizado, as interfaces representadas na figura 2.2 podem ser esquematizadas de acordo com a figura 2.4:

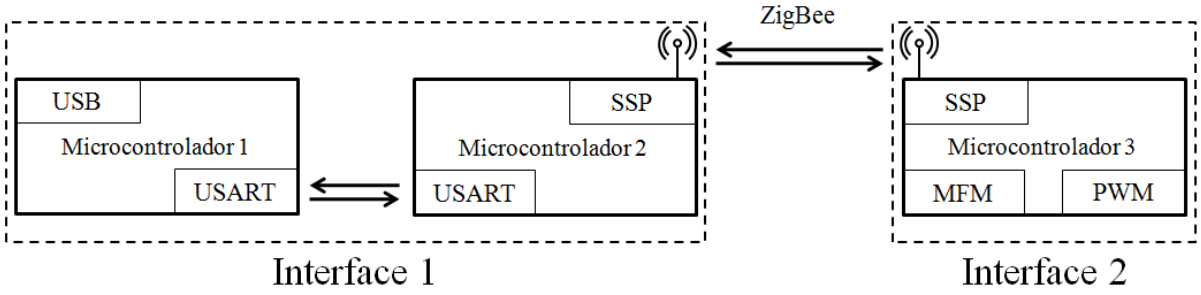


Figura 2.4 - Esquema pormenorizado das interfaces 1 e 2

A interface 1 é composta fundamentalmente por dois microcontroladores. O primeiro microcontrolador tem como principal função estabelecer a comunicação com o computador, utilizando o periférico USB, e realizar a troca de informação com o microcontrolador 2 efectuada através da comunicação USART. Este último é utilizado como intermediário entre o microcontrolador 1 e a antena transmissora Wireless. A permuta de dados entre a antena transmissora e o microcontrolador é realizada recorrendo à comunicação *Synchronous Serial Port* (SSP).

A interface 2 é composta apenas pelo microcontrolador 3. Este microcontrolador realiza a troca de dados com a antena transmissora através das portas dedicadas à comunicação SSP. O sinal de comando do motor DC é composto por dois sinais em PWM criados e emitidos pelo microcontrolador para a ponte de potência que irá alimentar o motor.

Observando agora a figura 2.5 é possível visualizar de forma mais detalhada toda a estrutura global de comunicação e potência implementada neste projecto.

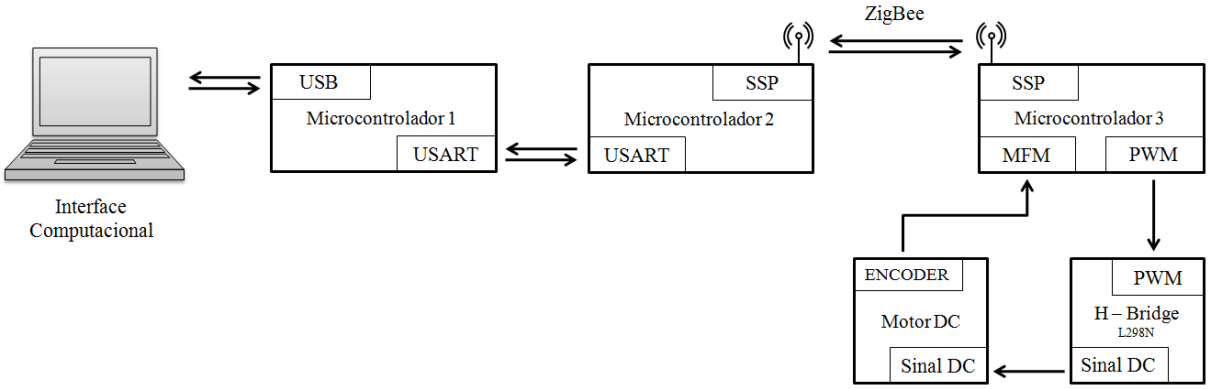


Figura 2.5 - Estrutura de comunicação e potência implementada

2.2 Hardware utilizado

A implementação da estrutura de comunicação apresentada requer um suporte físico com alguma complexidade. Nas figuras 2.6 e 2.7 estão representados respectivamente o esquema eléctrico da interface 1 e o aspecto físico da mesma.

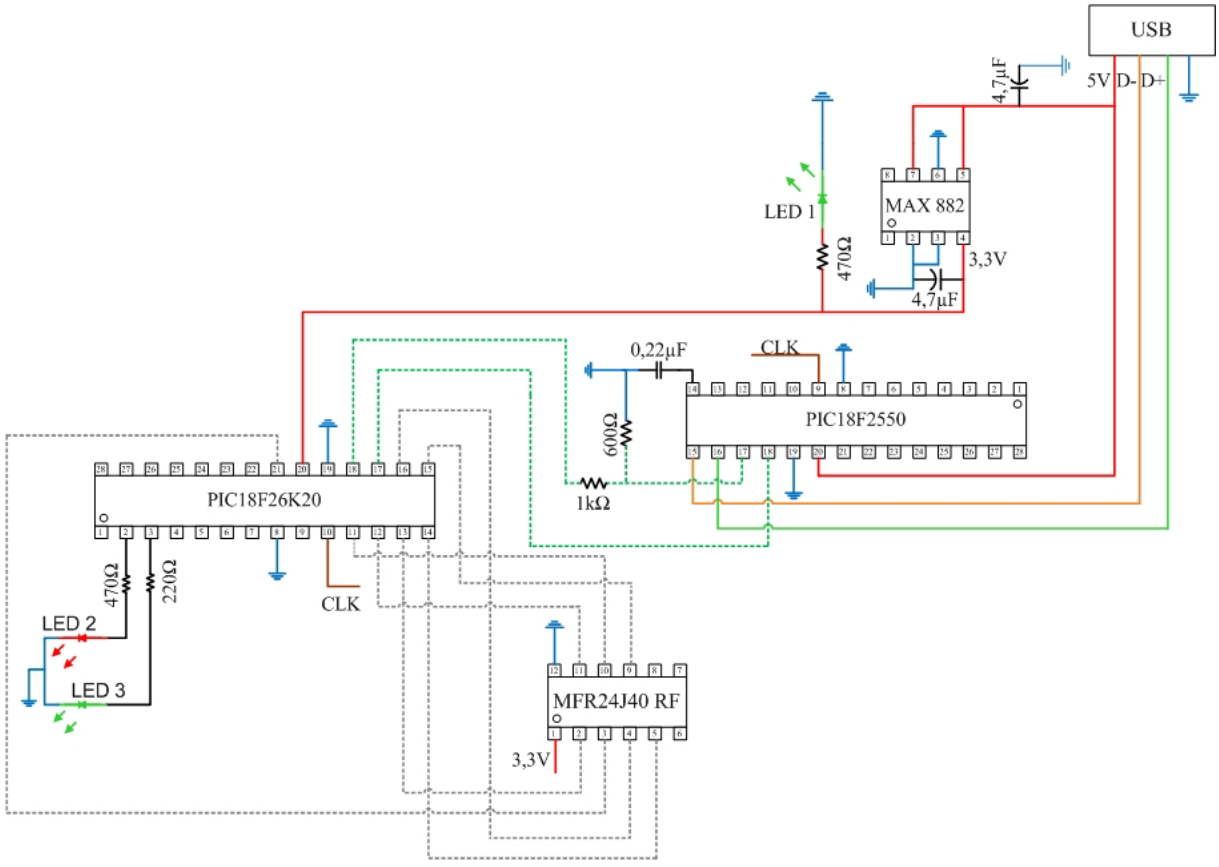


Figura 2.6 - Esquema eléctrico da interface 1

O led 1 indica o estado da alimentação do circuito, isto é, quando o circuito é alimentado o led 1 acende. Os led's 2 e 3 dizem respeito ao estado da comunicação, acendendo respectivamente quando um pacote de dados é enviado ou recebido pela antena transmissora.

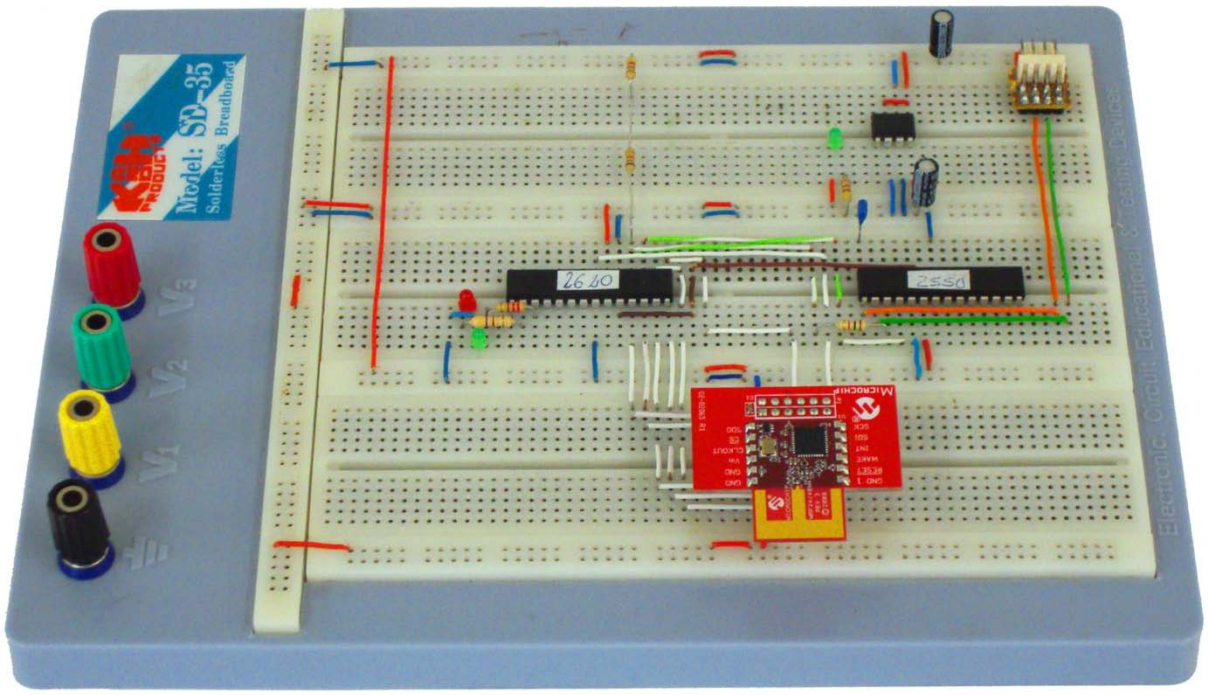


Figura 2.7 - Aspecto real da interface 1

A interface 2 é composta por um número considerável de componentes electrónicos. Este circuito é composto por dois reguladores de tensão, um microcontrolador, uma antena transmissora, um integrado lógico “NAND” e uma ponte H dedicada à alimentação do motor.

Nas figuras 2.8 e 2.9 é possível visualizar respectivamente o esquema eléctrico da interface 2 e o seu aspecto físico.

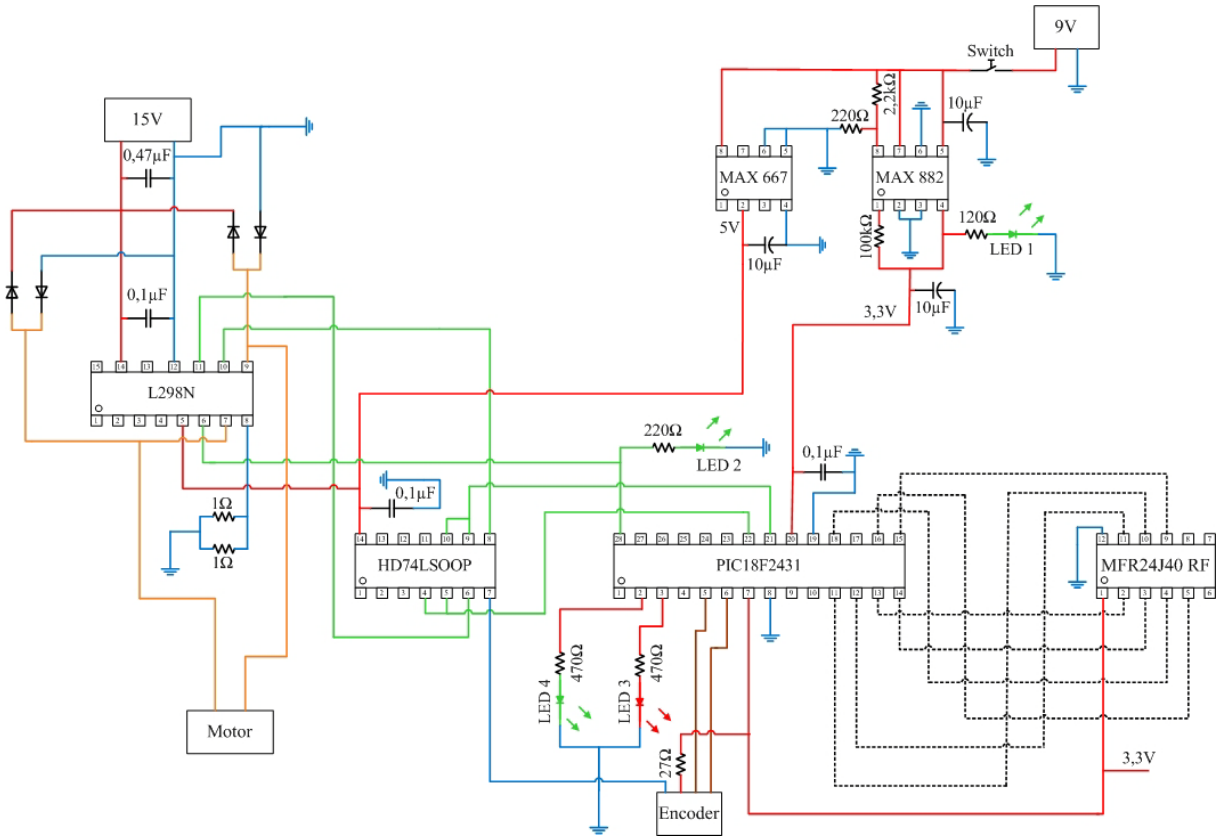


Figura 2.8 - Esquema eléctrico da interface 2

O led 1, à semelhança da interface anterior, indica o estado de alimentação do circuito electrónico. Por sua vez o led 2 indica o estado de alimentação do motor. Os leds 3 e 4 informam o utilizador sobre o estado da comunicação, acendendo respectivamente quando um pacote de dados é recebido ou enviado pela antena transmissora.

De forma a perceber melhor os esquemas eléctricos e a simplificar a sua visualização foi criado um código de cores próprio apresentado na tabela 2.1:

Cor	Interface 1	Interface 2
Azul	V_{SS}	V_{SS}
Vermelho	V_{DD}	V_{DD}
Verde	USB	PWM
Branco	SSP	SSP
Verde + Branco	USART	-
Laranja	USB	Alimentação do Motor
Castanho	CLOCK	Sinal do Encoder

Tabela 2.1 - Código de cores implementado nos circuitos reais

Por motivos de compatibilidade visual, a cablagem representada nas figuras 2.6 e 2.8 pelas cores verde-tracejado e preto-tracejado correspondem respectivamente às cores verde + branco e branco nos circuitos reais implementados.

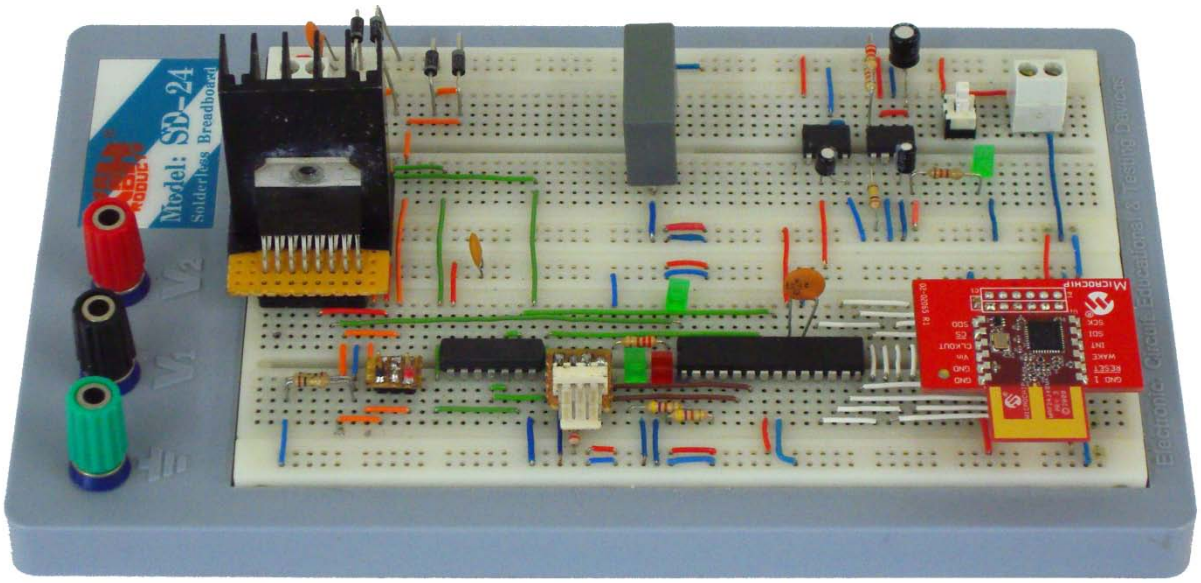


Figura 2.9 - Aspecto real da interface 2

A escolha atenta dos microcontroladores simplificou de forma significativa a implementação de todo o hardware. Os microcontroladores escolhidos pertencem à família PIC18LF e PIC18F e são produzidos pela empresa Microchip. Possuem periféricos dedicados para vários tipos de funções, deste modo foi possível, sem recorrer a periféricos externos, realizar toda a comunicação e comando do motor.

Na figura 2.10 está representado o diagrama de pinos do microcontrolador 1 [6]. Observando atentamente a figura é possível reparar que cada pino possui diferentes funções, estando cada função associada a um registo que pode ser configurado pelo utilizador.

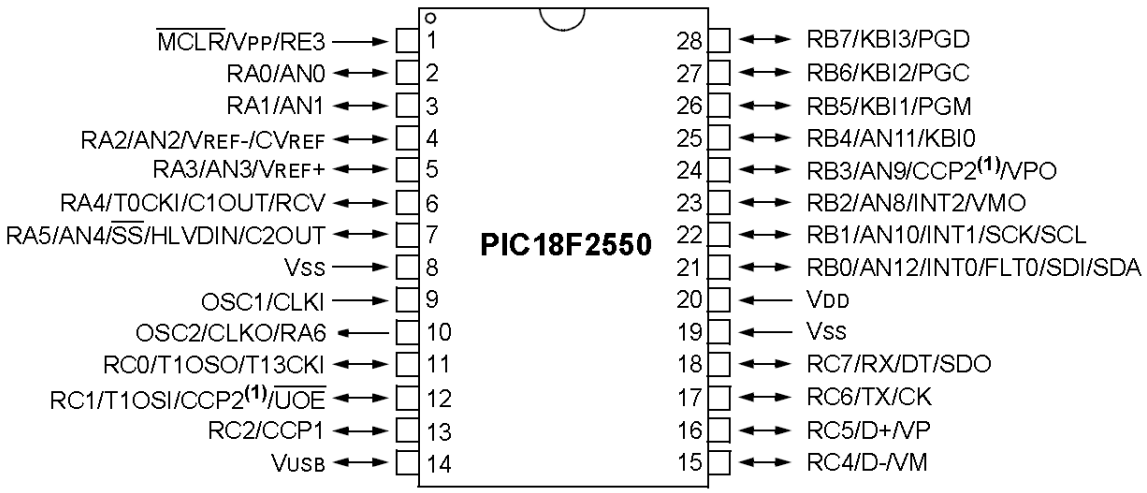


Figura 2.10 - Diagrama de pinos do microcontrolador 1 - PIC18F2550

Este microcontrolador apresenta algumas particularidades importantes para este projecto, destacando-se nomeadamente o facto de possuir um periférico dedicado à comunicação USB. Na tabela 2.2 podem ser visualizadas as características mais relevantes deste PIC.

PIC18F2550	
Universal Serial Bus	
Compatível com USB V2.0	Comunicação entre 1.5 Mb/s e 12 Mb/s
Alimentação	
3 Modos distintos de funcionamento	Tensão de alimentação de 5V
Oscilador	
Oscilador interno até 8 MHz	Possibilidade de oscilador externo até 48 MHz
Periféricos	
Comunicação USB	Comunicação USART
Saídas analógicas e digitais	Entradas analógicas e digitais

Tabela 2.2 - Características mais relevantes do microcontrolador 1 - PIC18F2550

Para desempenhar as funções do microcontrolador 2 foi escolhido o PIC18F26K20. A figura 2.11 mostra o diagrama de pinos relativo a este microcontrolador [7].

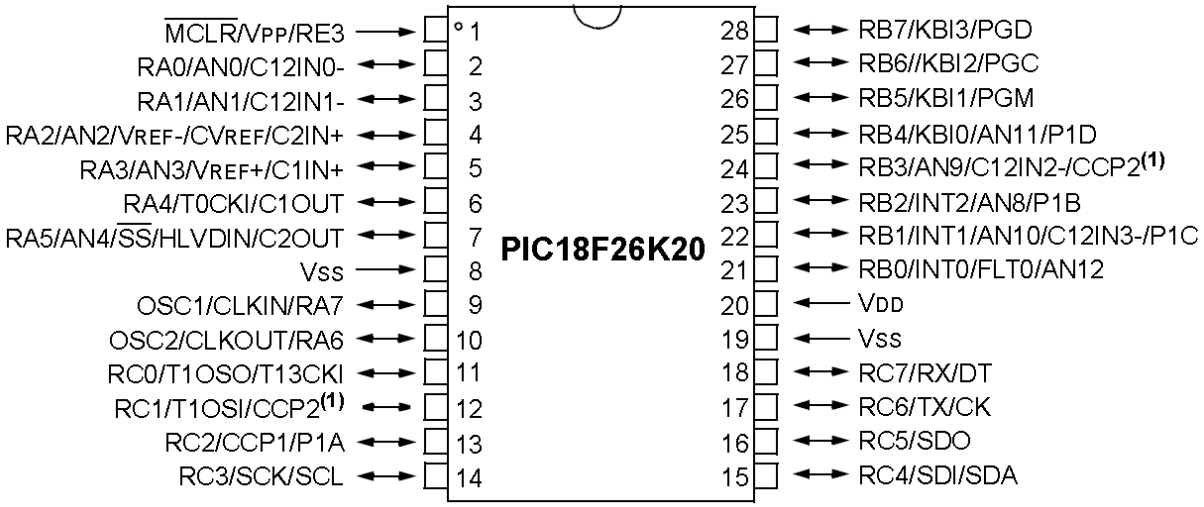


Figura 2.11 - Diagrama de pinos do microcontrolador 2 - PIC18F26K20

Os principais requisitos que influenciaram a escolha deste microcontrolador foram a capacidade de comunicação USART e a presença de barramentos *Serial Peripheral Interface* (SPI) que possibilitam a comunicação com o transmissor rádio.

A tabela 2.3 evidencia as suas características mais importantes para esta aplicação.

PIC18F26K20	
Alimentação	
3 Modos distintos de funcionamento	Tensão de alimentação entre 1,8V e 3,6V
Oscilador	
Oscilador interno até 16 MHz	Possibilidade de oscilador externo até 48 MHz
Periféricos	
Comunicação USART	Comunicação SSP
Saídas analógicas e digitais	Entradas analógicas e digitais

Tabela 2.3 - Características mais relevantes do microcontrolador 2 - PIC18F26K20

O PIC18LF2431 foi o microcontrolador escolhido para desempenhar as funções do microcontrolador 3. Este PIC é frequentemente utilizado no comando e controlo de servomotores DC e assíncronos em aplicações como esta.

O diagrama de pinos deste microcontrolador pode ser visualizado na figura 2.12 [8].

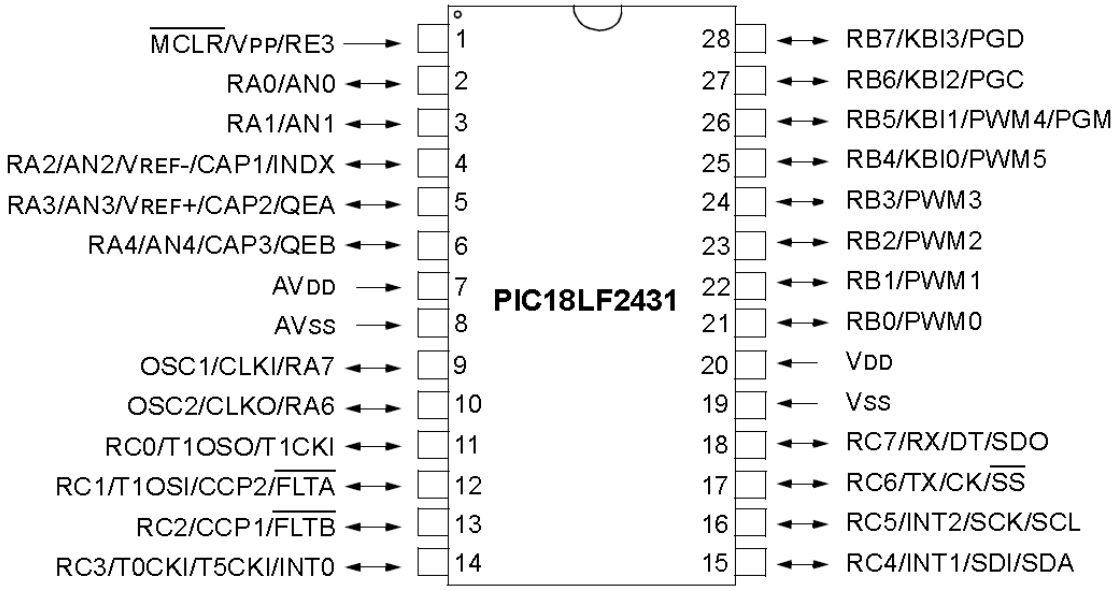


Figura 2.12 - Diagrama de pinos do microcontrolador 3 - PIC18LF2431

O PIC18LF2431 possui seis saídas PWM que, quando utilizadas em modo complementar, permitem comandar até três motores DC. A tabela 2.13 evidencia algumas das características deste microcontrolador.

PIC18LF2431	
PWM	
Até 3 saídas complementares	Frequência com resolução até 10 bits
Duty Cycle com resolução até 14 bits	Período com resolução até 14 bits
Alimentação	
3 Modos distintos de funcionamento	Tensão de alimentação entre 1,8V e 3,6V
Oscilador	
Oscilador interno até 8 MHz	Possibilidade de oscilador externo até 40 MHz
Periféricos	
PWM	Comunicação SSP
Saídas analógicas e digitais	Entradas analógicas e digitais

Tabela 2.4 - Características mais relevantes do microcontrolador 3 - PIC18LF2431

As antenas transmissoras rádio utilizadas são também fabricadas pela empresa Microchip. Estes componentes são os principais responsáveis pela implementação de todo o protocolo ZigBee utilizado neste projecto. Na figura 2.13 é possível visualizar o diagrama dos pinos utilizados na comunicação SSP realizada com o microcontrolador 2 e 3 [9].

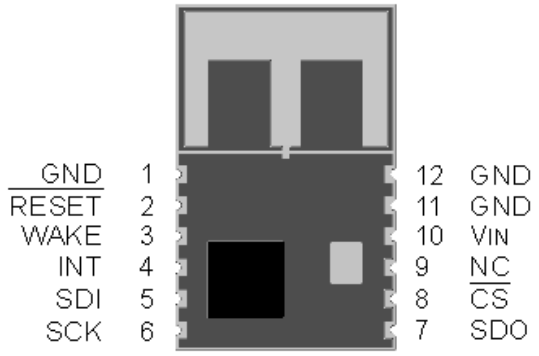


Figura 2.13 - Diagrama de pinos da antena Wireless

Esta antena possui particularidades próprias para implementação do protocolo ZigBee. A tabela 2.5 mostra as principais características de funcionamento deste transmissor rádio MRF24J40MA.

MRF24J40MA	
Tensão de alimentação típica de 3,3V	Frequência de operação: 868 MHz, 915 MHz, 2.4 MHz
Velocidade de transferência de dados: 250 Kbps	Alcance até 120 metros
Compatível com a família de micro controladores PIC18F	
Comunicação SSP	Baixo consumo de corrente

Tabela 2.5 - Características mais relevantes da antena transmissora – MRF24J40MA

Os circuitos integrados MAX667 e MAX882 são reguladores de tensão que foram configurados para 5V e 3,3V respectivamente. São utilizados para regular a tensão de alimentação dos circuitos electrónicos [10-11]. De seguida são apresentados os diagramas de pinos (figura 2.14) e algumas características destes componentes (tabela 2.7).

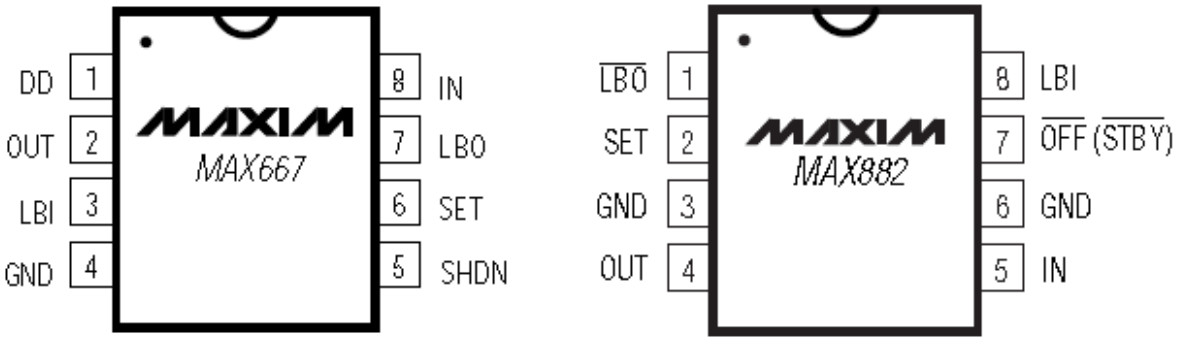


Figura 2.14 - Diagrama de pinos do MAX667 e MAX882

MAX667	MAX882
Tensão de saída ajustável até 5V	Tensão de saída ajustável entre 1,25V e 11,5V
Tensão de entrada entre 3,5V e 16.5V	Tensão de entrada entre 2,7V e 11,5V
Corrente de saída até 250 mA	Corrente de saída até 250 mA
Detecção de bateria de alimentação fraca	Detecção de bateria de alimentação fraca

Tabela 2.6 - Características mais relevantes do MAX667 e do MAX882

Na interface 2 é possível encontrar, para além dos componentes já referidos, mais dois circuitos integrados. O HD74LSOOP refere-se a um circuito lógico "NAND" e é utilizado para filtrar o ruído e forçar a 5V o sinal PWM enviado pelo microcontrolador antes de entrar na ponte de potência. O circuito lógico implementado neste integrado pode ser visível na figura 2.15.

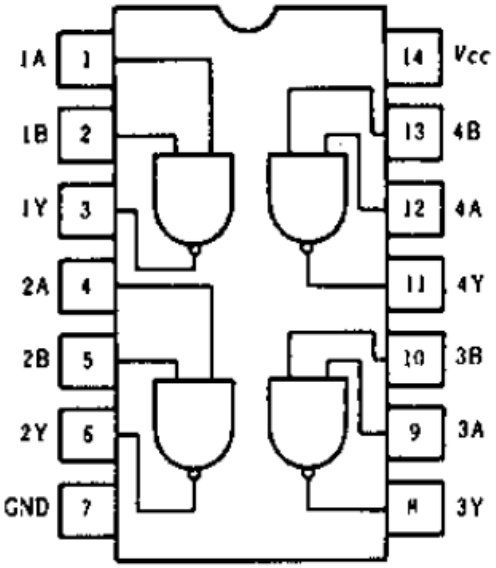


Figura 2.15 - Circuito lógico implementado no integrado HD74LSOOP

A ponte H L298N é utilizada para fornecer toda a potência ao sistema de actuação. Na figura 2.16 está representado o circuito lógico deste integrado.

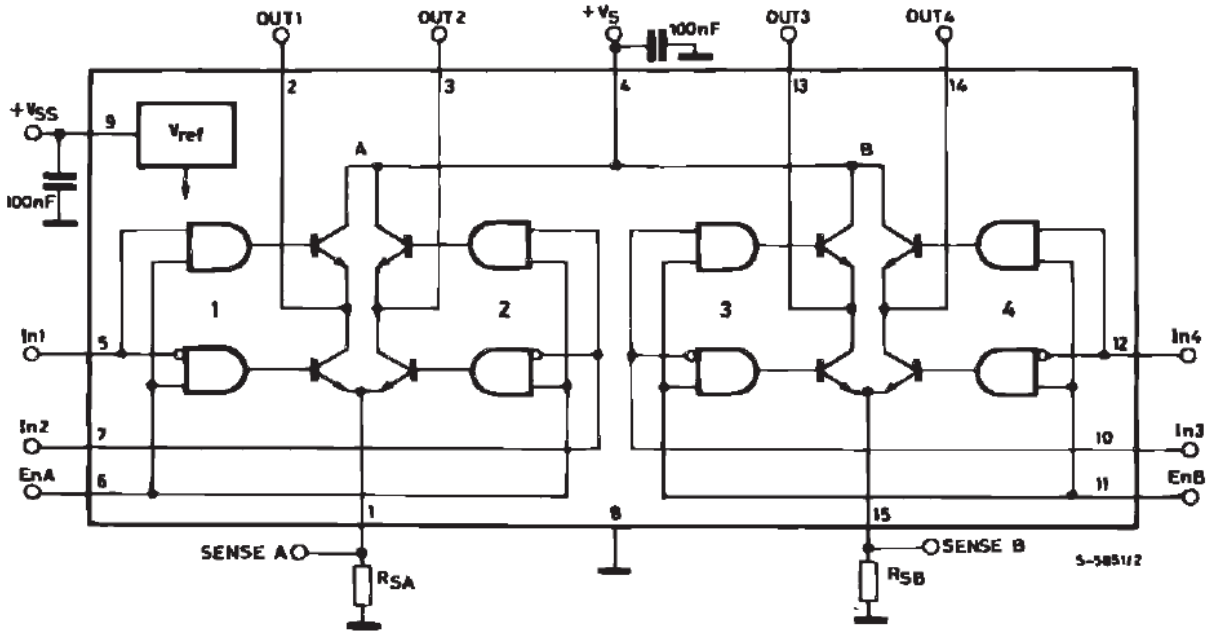


Figura 2.16 - Circuito lógico implementado no integrado L298N

Este integrado possui no seu interior duas pontes o que permite fornecer potência a dois motores DC. Na tabela 2.7 é possível visualizar algumas das características sobre esta ponte de potência.

Parâmetro	Valor Mínimo	Valor Máximo
Tensão de entrada - V_S	V_{SS} + 2,5 V	46 V
Tensão de referência - V_{SS}	4,5 V	7 V
Corrente de saída	-	2 A
Potência dissipada	-	25 W

Tabela 2.7 - Características mais relevantes da Ponte H - L298N

2.3 Software utilizados

Desde a programação dos microcontroladores até à supervisão da rede de comunicação foram vários os softwares utilizados durante este projecto. De seguida são apresentados os softwares utilizados com maior relevância para este projecto.

ZENA™ Wireless Network Analyzer Overview

Para auxiliar a desenvolver as aplicações do protocolo ZigBee, protocolo MiWi e protocolo MiWi P2P, a Microchip disponibiliza um software chamado Zena. Este software é fornecido, em versão demo, gratuitamente como parte do conjunto de ferramentas *Microchip Stack* [12]. Este conjunto de ferramentas foi concebido pela empresa Microchip de forma a auxiliar o utilizador na criação de novos projectos.

Zena é um software de baixo custo particularmente desenvolvido para a análise de redes. Este software contém também ferramentas destinadas à criação de ficheiros específicos de configurações e rotinas de ligação para aplicações dos três protocolos referidos anteriormente. Estes ficheiros configurativos são criados de forma a apoiar o *Microchip Stack* e analisar previamente o tráfego da rede Wireless, assim como conhecer em tempo real as actividade que estão a ser executadas.

O analisador Zena dispões de três principais ferramentas para desenvolver, através da norma IEEE 802.15.4, soluções de forma rápida e eficiente utilizando o *Microchip Stack*. Com este software é possível modificar e adaptar o *Stack* de modo a suportar as necessidades da aplicação, assim como identificar a topologia da rede tal como ela é constituída, permitindo aos utilizadores observar e analisar a transmissão de dados entre os vários constituintes da mesma.

Na figura 2.17 é possível visualizar a interface gráfica do software Zena.

Figura 2.17 - Interface gráfica do software Zena Network Analyzer Overview

MikroC Integrated Development Environment

MikroC IDE é uma ferramenta poderosa para microcontroladores da família PIC12, PIC16, e PIC18.

Esta ferramenta proporciona uma boa correspondência com *Ambiente de Desenvolvimento Integrado* (IDE) altamente avançado. Compatível com o compilador ANSI, dispões de um amplo conjunto de bibliotecas de hardware, documentação completa e uma grande quantidade de programas prontos a funcionar.

Este software inclui bibliotecas que permitem o desenvolvimento de funções aplicativas, destacando-se, a aquisição de dados, gestão de memória, *displays* e comunicações.

A empresa MikroElektronika fornece, juntamente com este software, uma série de exemplos prontos a serem analisados e desenvolvidos de forma a serem utilizados na construção de novos projectos e aplicações.

MPLAB Integrated Development Environment

O MPLAB Integrated Development Environment (IDE) fornece um conjunto de ferramentas integradas para o desenvolvimento de aplicações utilizando microcontroladores Microchip.

É fácil de utilizar e inclui uma série de componentes de software para o desenvolvimento rápido e depuração de todo o programa. MPLAB IDE é também utilizado como interface entre o utilizador e softwares/ferramentas adicionais de desenvolvimento de hardware.

Este software utiliza, entre outros, o MPLAB C18 Compilers, compiladores altamente optimizados para uma série de microcontroladores Microchip.

Esta plataforma de programação é utilizada quando existe a necessidade efectuar alterações de mais baixo nível não permitidas pelo MicroC IDE.

Capítulo 3

Comunicação Wireless

A comunicação Wireless permite a comunicação entre dois ou mais dispositivos sem a utilização de cablagem, sendo utilizada neste projecto com o objectivo de transferir informação entre as duas antenas situadas junto às extremidades (físicas) da rede sem fios.

Este capítulo apresenta uma descrição simplificada dos protocolos implementados na comunicação Wireless assim como do software utilizado para a análise da rede sem fios.

3.1 Protocolo ZigBee

O protocolo ZigBee é um protocolo de comunicação Wireless concebido para baixas taxas de transmissão de dados e hardware relativamente barato, baseado na norma IEEE 802.15.4, sendo uma comunicação global e confiável [13].

Este protocolo suporta topologias de rede em *Star*, *Cluster* ou *Mesh*, tornando-o adequado para aplicações de baixo alcance e baixo consumo energético. O protocolo Microchip ZigBee – 2006 é certificado para funcionar nas famílias dos microcontroladores PIC 18 e PIC 24 fabricados pela Microchip e transmissores MRF24J40 de 868 MHz, 915 MHz e 2.4 GHz.

O *Microchip Stack* para o protocolo ZigBee oferece uma biblioteca modular, e de fácil utilização, de aplicações especificamente concebidas para, com alterações mínimas de software, suportar mais que um transmissor RF. O *Microchip Stack* é indicado para ser implementado nos softwares MPLAB C Compile para PIC18 MCUs, MPLAB C Compile

para PIC24 MCUs e dsPIC DSCs, mas pode facilmente ser modificado para suportar outros compiladores.

3.1.1 Limitações

O protocolo ZigBee deixa muitas decisões de nível superior ao projectista. Deste modo o *Microchip Stack* não fornece nenhum suporte explícito para algumas funções:

- *Beacon networks* não são suportadas por esta versão do protocolo ZigBee;
- Os endereços dos nós que deixaram a rede não podem ser transferidos;
- Não é suportada a fragmentação do pacote enviado;
- Não é suportada grande agilidade frequencial;
- Um coordenador *Personal Area Network* (PAN) alternativo não é suportado em redes de protocolo ZigBee.

3.1.2 Norma IEEE 802.15.4-2003

O protocolo ZigBee utiliza as especificações *Medium Access Layer* (MAC) e *Physical Layer* (PHY) da norma IEEE 802.15.4-2003.

Esta norma define três bandas de frequência operacionais: 868 MHz, 915 MHz e 2.4 GHz. Cada uma destas bandas tem um número fixo de canais: a banda de frequência 868 MHz disponibiliza um único canal (canal 0), a banda 915 MHz disponibiliza 10 canais (canais 1-10) e a banda 2.4 GHz disponibiliza 16 canais (canais 11-26).

A velocidade de transmissão de bits depende da frequência de operação. A banda de 868 MHz fornece até 20 kbps, a banda de 915 MHz até 40 kbps e a banda de frequência de 2.4 GHz disponibiliza uma transmissão de dados ate 250 kbps. Como é de esperar, a velocidade de transmissão de dados real é inferior à velocidade especificada devido aos atrasos de processamento e sobrecarga do sistema.

O tamanho máximo de um tramo MAC da norma IEEE 802.15.4-2003 é de 127 bytes, incluindo o valor do bit 16 *cyclic redundancy check* (CRC). O valor do bit 16 CRC verifica a validade do pacote transmitido. Além disso, a norma IEEE 802.15.4-2003 pode utilizar um sistema de reconhecimento de transferência de dados. Com este método todos os pacotes que possuírem um pedido especial de reconhecimento são reconhecidos pelo receptor, assegurando assim que de facto o pacote é entregue. Se um determinado pacote é enviado com o pedido especial de reconhecimento e a confirmação não é recebida num período de tempo específico, o transmissor repete a transmissão durante um número definido de vezes

antes de declarar a existência de um erro. É de notar que este aviso indica simplesmente que o pacote foi correctamente recebido pela *layer* MAC, não sendo no entanto detectável se o pacote é ou não processado correctamente. É possível que a *layer* MAC do receptor receba e reconheça um pacote correctamente, no entanto, devido à falta de recursos de processamento, um pacote pode ser recusado pelas camadas superiores. Desta forma, as *layers* superiores podem necessitar de um pedido de reconhecimento adicional.

3.1.3 Tipo de dispositivos

A norma IEEE 802.15.4-2003 define dois tipos de dispositivos. Na tabela 3.1 pode-se visualizar estes dois tipos e algumas das suas características.

Tipo de Dispositivo	Serviços Disponíveis	Fonte de Alimentação	Configuração de Recepção
Full Function Device (FFD)	A maioria ou todos	Rede pública	Quando desocupado
Reduce Function Device (RFD)	Limitado	Bateria	Quando ocupado

Tabela 3.1 - IEEE 802.15.4 - Tipos de dispositivos

Por sua vez o protocolo ZigBee utiliza não dois, mas três tipos distintos de dispositivos. A tabela 3.2 mostra esses dispositivos e a forma como eles se relacionam com os dispositivos da norma IEEE.

Dispositivo (Protocolo ZigBee)	Dispositivo (Norma IEEE)	Função
Coordenador	FFD	É necessário um por rede. Atribui endereços de rede aos dispositivos e permite que outros se juntem à rede.
Router	FFD	Opcional. Amplia o alcance físico da rede. Permite que mais nós se juntem à rede. Também pode realizar monitorização e/ou funções de controlo.
Final	FFD ou RFD	Realiza monitorização e/ou funções de controlo.

Tabela 3.2 - Protocolo ZigBee - Tipos de dispositivos

3.1.4 Topologias da rede

As redes Wireless que utilizam o protocolo ZigBee podem assumir várias topologias. Em todas elas existem pelo menos dois componentes principais:

- Coordenador;
- Dispositivo Final.

Um coordenador é uma variante específica do *Full Function Device* (FFD) que implementa um conjunto maior de serviços do protocolo ZigBee. Um dispositivo final pode ser um FFD, como por exemplo um *Router*, ou um *Reduce Function Device* (RFD). Um RFD é o mais pequeno e simples nó do protocolo ZigBee e que, por sua vez, apresenta um conjunto mínimo de serviços. As redes Wireless que utilizam este protocolo podem ainda, opcionalmente, incluir um *Router* como terceiro elemento.

Uma rede de protocolo ZigBee é uma rede de multi-acesso, o que significa que todos os nós da rede têm igual acesso aos meios de comunicação.

Configuração em Star

A configuração topológica em *Star*, referida na figura 3.1, consiste num único nó coordenador e num ou mais dispositivos finais. Neste tipo de configuração todos os dispositivos finais comunicam somente com o coordenador. Se um dispositivo final necessitar de enviar dados para outro dispositivo final, este envia para o coordenador, que por sua vez os encaminha para o destinatário.

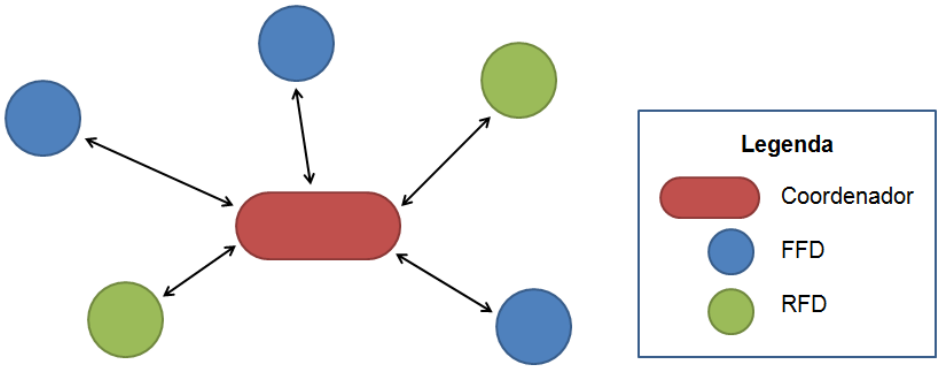


Figura 3.1 - Configuração Topológica em *Star*

Configuração em Cluster Tree

Outro tipo de topologia que uma rede Wireless configurada com o protocolo ZigBee pode assumir é a *Cluster Tree*, apresentada na figura 3.2. Neste tipo de configuração os dispositivos finais podem estar ligados tanto ao coordenador da rede como a um *Router*. Na configuração *Cluster Tree* os *Routers* assumem dois tipos de funções: a de aumentar o número de nós que podem estar numa rede e a de estender o alcance físico da mesma. Com a adição de um *Router*, o dispositivo final não necessita de estar no alcance físico do coordenador, pois todas as mensagens são encaminhadas ao longo dos vários ramos da rede.

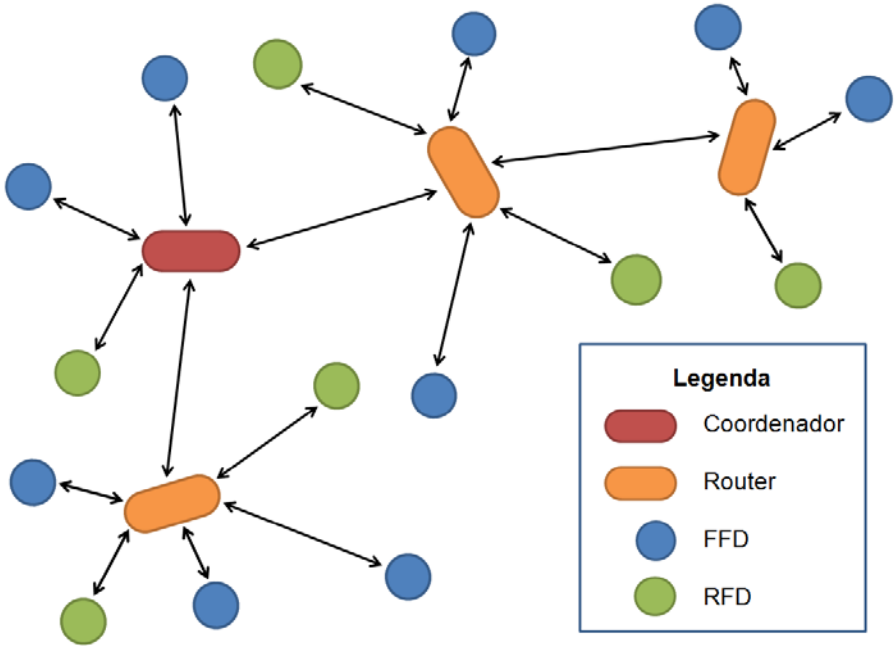


Figura 3.2 - Configuração topológica em *Cluster Tree*

Configuração em Mesh

A configuração em *Mesh* é bastante similar à configuração anterior. A diferença, visível na figura 3.3, surge da possibilidade de um dispositivo FFD enviar dados directamente para outro dispositivo do mesmo tipo sem passar pelo coordenador ou um *Router*. A grande vantagem desta topologia é a redução da latência da mensagem aumentando a sua robustez e fiabilidade.

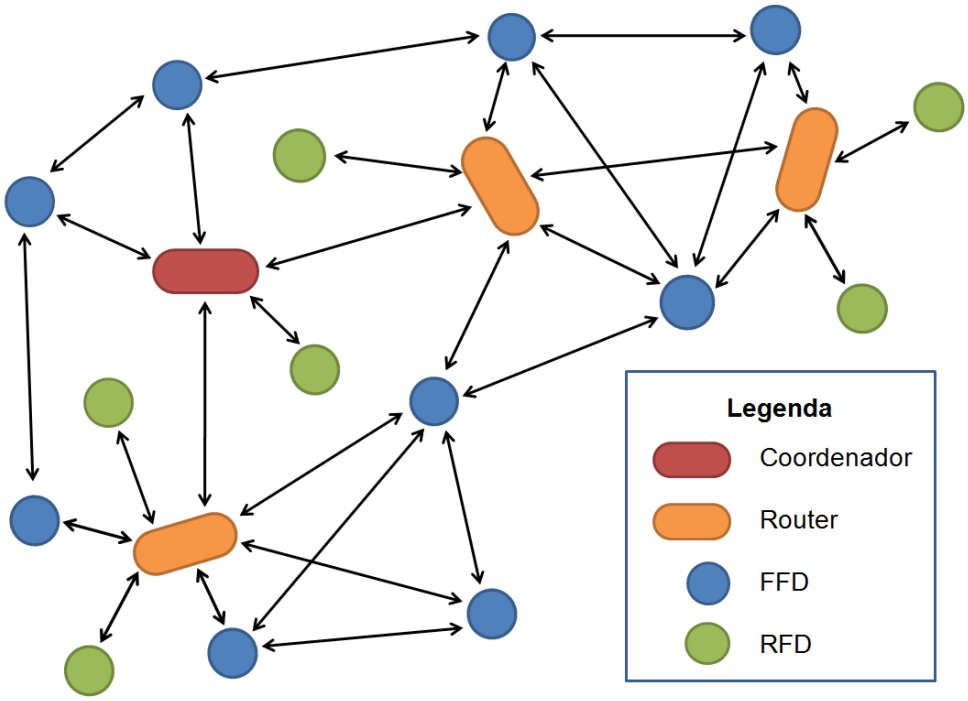


Figura 3.3 - Configuração Topológica em Mesh

3.1.5 Mecanismos de acesso

A comunicação entre os vários dispositivos exige regras. Desta forma é necessário criar mecanismos de acesso aos canais da rede de forma a controlar o tráfego de dados.

Existem dois tipos de mecanismos de multi-acesso: *beacon* e *non-beacon*. O mecanismo de acesso do tipo *non-beacon* possibilita realizar em simultâneo a transmissão de dados entre diferentes nós, desde que os canais estejam desocupados. No caso do mecanismo de acesso do tipo *beacon*, a transmissão de dados é realizada em períodos de tempo bem definidos. O coordenador transmite periodicamente um super pacote, designado como *beacon frame*, e todos os nós sincronizam-se com esse mesmo pacote. A cada nó é atribuída uma parte específica do super pacote, utilizada então para transmitir e receber os seus dados. O super pacote pode ainda conter uma parte comum. O acesso a esta parte é então disputado por todos os nós.

A versão actual do *Microship Stack* apenas suporta redes com mecanismos de acesso do tipo *non-beacon*.

3.1.6 Tipos de endereço

Cada dispositivo que utilize o protocolo ZigBee tem de possuir um endereço MAC de 64 bits exclusivo. Este endereço é constituído por 24 bits criados segundo a *Organizationally Unique*

Identifier (OUI), acrescidos de 40 bits atribuídos pelo fabricante. Os bits OUI devem ser comprados à IEEE de forma a garantir a exclusividade global do endereço.

Quando um dispositivo é adicionado a uma rede de protocolo ZigBee é lhe atribuído um código de 16 bits, de um leque pré-definido de endereços, que lhe permite comunicar com os outros dispositivos da rede.

Cada nó possui então dois endereços: um endereço MAC de 64-bit e um endereço da rede de 16-bit.

3.1.7 Hardware necessário

Para criar um nó com o protocolo ZigBee utilizando o *Microchip Stack* é necessário, no mínimo, os seguintes componentes:

- Um microcontrolador da Microchip com interface SSP;
- Um transmissor Microchip MFR24J40 RF com os componentes externos necessários;
- Uma antena;
- Um *External serial EEPROM* (opcional);

A comunicação entre o microcontrolador e o transmissor MFR24J40 RF baseia-se numa relação do tipo *Master – Slave*. Nesta relação o microcontrolador comporta-se como *Master*, implementando o modelo IEEE 802.15.4 *Medium Access Control* (MAC) e o protocolo ZigBee, e as acções executadas pelo transmissor assume-se como *Slave*. Esta comunicação é realizada através do barramento SSP e alguns sinais de controlo discreto, esquematizada na figura 3.4.

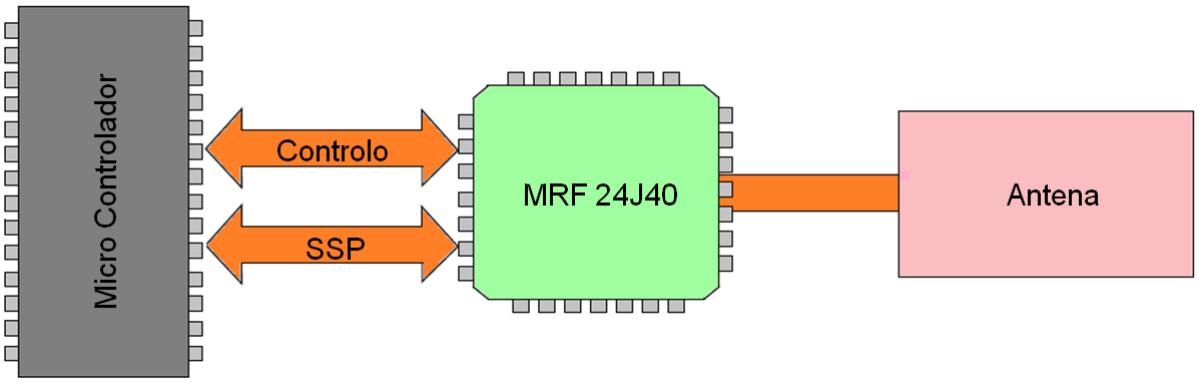


Figura 3.4 - Hardware de um nó (Protocolo ZigBee)

3.2 Protocolo MiWi P2P

Este protocolo resulta de uma simplificação do protocolo ZigBee, aplicável apenas quando a rede é constituída somente por dispositivos Microchip.

A grande diferença do protocolo MiWi P2P relativamente à norma IEEE 802.15.4 recai sobre o processo de associação de um dispositivo à rede [14].

Na figura 3.5 estão representadas, de forma esquemática, as várias etapas do processo de associação típicas da norma IEEE 802.15.4. Estas etapas são desenvolvidas da seguinte forma:

1 - Inicialmente o dispositivo que se pretende ligar à rede envia um pedido de atenção na forma *Broadcast*;
2 - Todos os dispositivos capazes de se ligar a outros respondem a esse pedido;
3 - O dispositivo recebe todas as respostas, analisa-as e decide através de qual pretende estabilizar a sua ligação, enviando de seguida o pedido de associação;
4 - Após um tempo pré-definido, o dispositivo emite um comando de solicitação de dados para que possa receber a resposta ao pedido de associação;
5 - O dispositivo situado do outro lado da conexão emite a resposta de associação finalizando o processo.

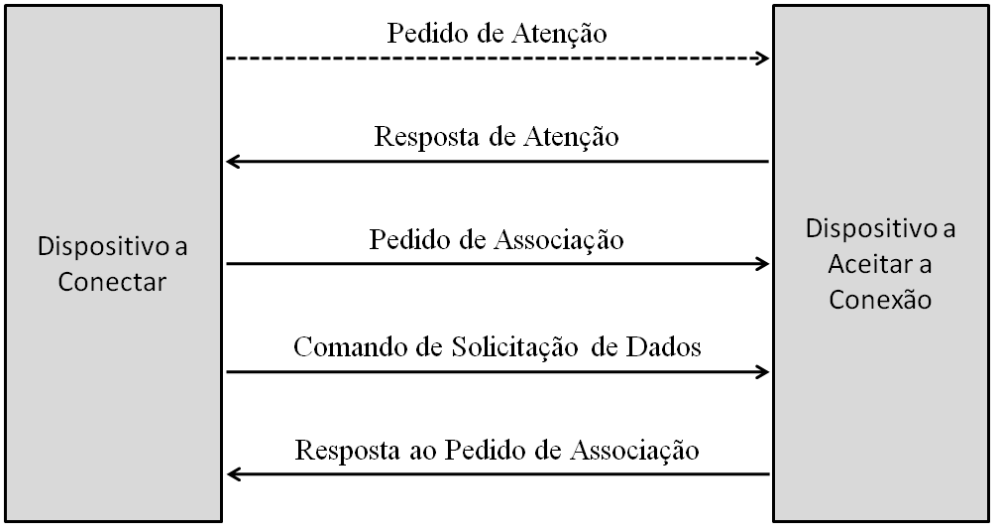


Figura 3.5 - Processo de associação no IEEE 802.15.4

O protocolo MiWi P2P é concebido para actuar de forma directa e simples no processo de conexão em comunicações com topologias do tipo star e P2P. Para isso este protocolo utiliza conexões múltiplas em vez das conexões simples utilizadas pela norma IEEE 802.15.4. Por

esta razão o processo de associação executado pelo protocolo MiWi P2P requer unicamente duas etapas. Estas etapas podem ser esquematicamente observadas na figura 3.6.

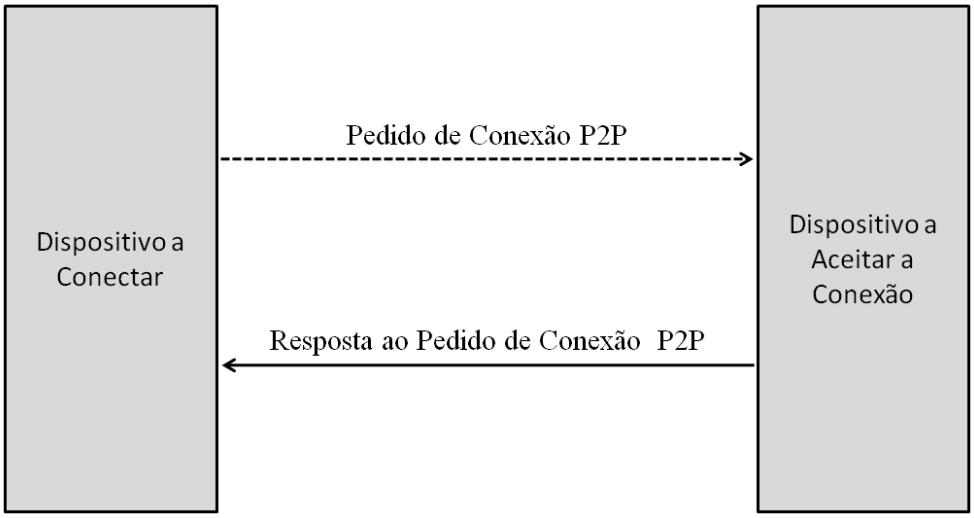


Figura 3.6 - Processo de associação no MiWi P2P

As duas etapas são as seguintes:

1 - O dispositivo que se pretende ligar à rede envia um pedido de conexão P2P na forma *Broadcast*;
2 - Os dispositivos que estão no seu alcance físico respondem ao pedido finalizando a conexão P2P.

Este processo pode então criar múltiplas ligações estabelecendo uma topologia P2P. Desde que o processo de associação utilize a *layer* MAC a probabilidade de colisão entre as mensagens enviadas é largamente diminuída.

Os dispositivos RFD podem receber vários pedidos de associação enviados pelos vários dispositivos FFD da rede, mas apenas se podem conectar a um. Esta escolha recai sobre o primeiro dispositivo FFD a solicitar a conexão.

3.2.1 Composição dos pacotes transmitidos

As mensagens são enviadas de dispositivo para dispositivo através de pacotes. Cada pacote é composto por diferentes sectores, onde cada sector possui uma função bem definida.

Na figura 3.7 é possível visualizar a estrutura do pacote enviado para o pedido de conexão quando utilizado o protocolo MiWi P2P.

15/25 Bytes	1 Byte	1 Byte	1 Byte	1 Byte (Opcional)
MAC Header	Identificador de Comando	Canal de Operação	Capacidade de Informação	Informação de identificação do nó

Figura 3.7 - Estrutura do pacote enviado para pedido de conexão - MiWi P2P

O sector reservado ao MAC *Header* é constituído pelos campos MAC *Frame Control*, *Sequence Number* e pelos endereços das mensagens que estão a ser transmitidas.

O Identificador de comando armazena a informação relativa ao tipo de comando, isto é, se o pacote diz respeito a um pedido de conexão ou resposta a um pedido de conexão.

Na figura 3.8 está representado o pacote real relativo ao pedido de conexão enviado pelo transmissor quando este se tenta associar à rede.

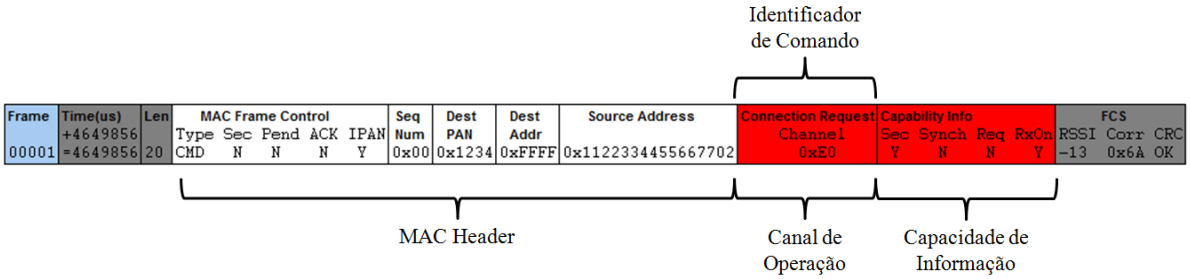


Figura 3.8 - Pacote real relativo ao pedido de conexão – MiWi P2P

Na figura 3.8 é também visível a divisão do sector *MAC Header* nos campos referidos anteriormente. A tabela 3.3 pretende mostrar de forma mais pormenorizada o conteúdo do campo *MAC Frame Control*.

Código	Nome	Opções
Type	*Frame Type*	CMD / ACK / DATA
Sec	*Security Enable*	Y / N
Pend	*Frame Pending*	Y / N
ACK	*Acknowledgement Request*	Y / N
IPAN	Intra PAN	Y / N

Tabela 3.3 - Conteúdo do campo *MAC Frame Control*

O pacote pode variar entre três tipos, *CMD* (*Command Frame*) no caso de se tratar de um pedido de associação ou a resposta a um pedido de associação, *ACK* (*Acknowledgement*)

quando o pacote corresponde a um pedido especial de reconhecimento ou *DATA* (*Data Frame*) quando o pacote transferido possui informação relevante ao sistema.

A activação do bit relativo à segurança significa que o pacote actual está encriptado, indicando que existe um *header* adicional de segurança que será detalhado em secções posteriores.

O bit correspondente ao *Frame Pending* é utilizado unicamente no pacote de reconhecimento e manuseado pelo transmissor rádio. Este bit indica se algum pacote adicional será enviado a seguir ao reconhecimento, depois do comando de solicitação de dados ser recebido pelo dispositivo RFD.

O bit associado ao Intra PAN indica se a mensagem está contida no PAN actual. Se este bit tomar o valor '1', o campo relativo ao PAN ID de origem será ocultado dos campos destinados aos endereços. Este bit aparece normalmente definido com o valor '1' podendo no entanto ser definido como '0' actuando a comunicação Inter-PAN.

Na figura 3.9 é mostrada a estrutura do pacote enviado em resposta ao pedido de conexão P2P.

21 Bytes	1 Byte	1 Byte	1 Byte	Vários (Opcional)
MAC Header	Identificador de Comando	Status (0x00 = Sucesso)	Capacidade de Informação	Informação de identificação do nó

Figura 3.9 - Estrutura do pacote enviado em resposta ao pedido de conexão - MiWi P2P

Este pacote apresenta uma estrutura em muito idêntica ao anterior variando unicamente na quantidade de bytes associados ao sector *MAC Header* e na substituição do byte relativo ao canal de operação pela informação do estado da conexão.

O byte correspondente ao sector *Status* pode variar entre uma quantidade elevada de valores, contudo o valor '0x00' é o único que define o sucesso da conexão, todos os outros são assumidos como erro.

Na figura 3.10 é possível visualizar o pacote real de resposta ao pedido de associação P2P.

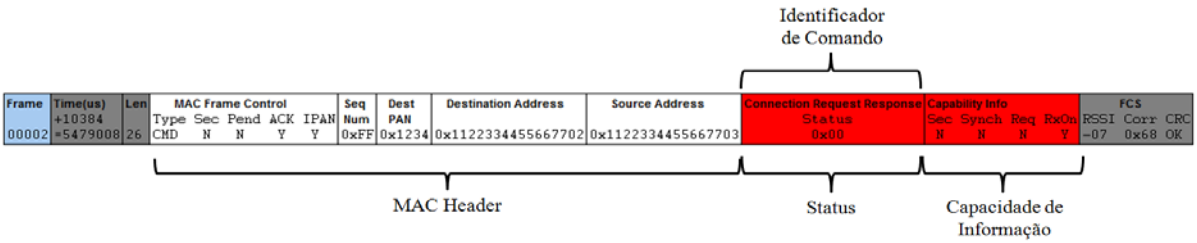


Figura 3.10 - Pacote real relativo à resposta ao pedido de conexão – MiWi P2P

Após a conclusão da conexão, a rede está pronta para executar a transferência dos dados solicitados pelo utilizador. Na figura 3.11 está ilustrada a estrutura do pacote onde são transportados esses mesmos dados.

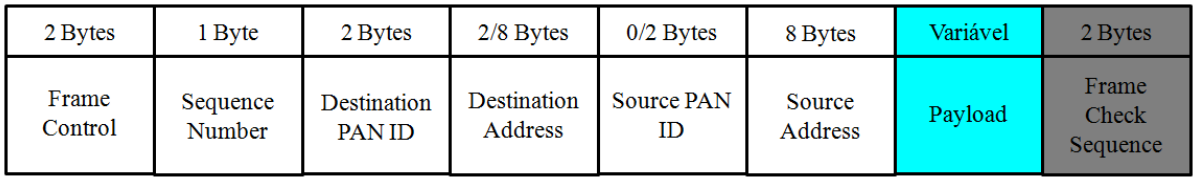


Figura 3.11 - Estrutura do pacote de envio dos dados solicitados

O pacote real onde é realizado o transporte dos dados solicitados pelo utilizador pode ser visualizado na figura 3.12.

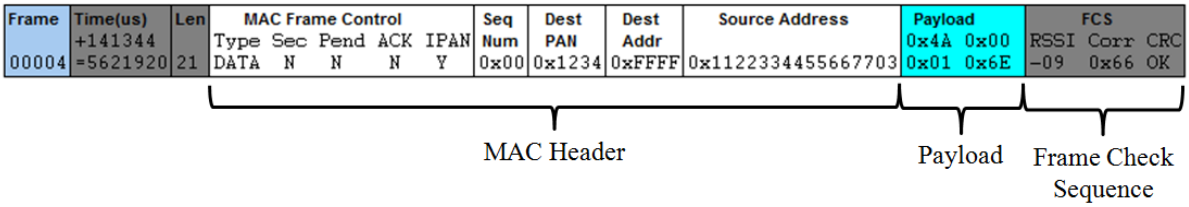


Figura 3.12 - Pacote real de transporte dos dados solicitados

Os campos preenchidos a branco correspondem mais uma vez ao sector *MAC Header*, o sector *Payload*, preenchido a azul, fica reservado para todo o pacote de informação solicitado pelo utilizador para ser enviado.

Observando agora a figura 3.13 é possível visualizar todos os pacotes trocados entre dois transmissores durante o processo de conexão e inicialização da rede sem fios.

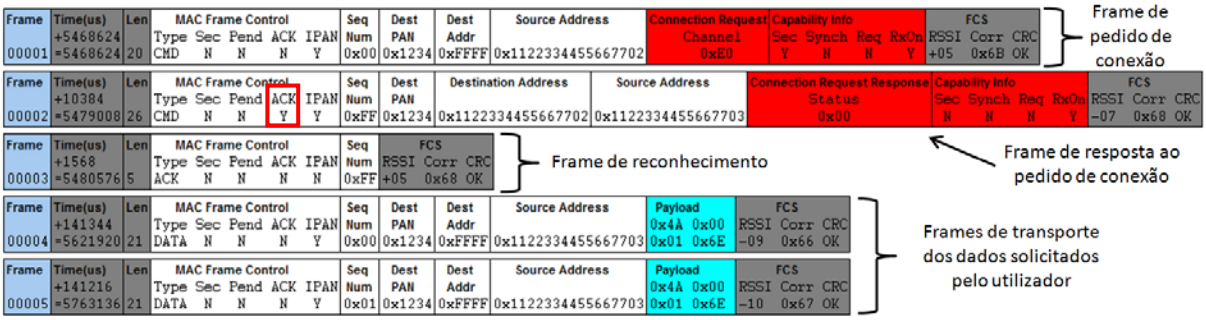


Figura 3.13 - Sequência de pacotes reais na inicialização da rede

Na figura 3.13 é também de realçar a activação do bit de reconhecimento, assinalado no pacote 2, solicitando um pedido especial de reconhecimento enviado no pacote 3.

3.2.2 Ferramenta de configuração Stack

O *Microchip MiWi P2P Stack* oferece ao utilizador a capacidade de se integrar em redes de topologia em *Star* ou *Pear-to-Pear* (P2P).

O software Zena possibilita o auxílio na configuração do *Microchip Stack* gerando automaticamente ficheiros *header* para aplicações de protocolo MiWi P2P. Utilizando as várias janelas de dialogo é possível seleccionar opções, que estão configuradas no *MiWi P2P Stack*, de forma a activar ou desactivar determinados parâmetros de acordo com as opções seleccionadas. Todas as opções activas adquirem valores pré-definidos [12]. De seguida são representados alguns detalhes sobre este processo de configuração.

Especificação do dispositivo

Na figura 3.14 é apresentada a janela de diálogo para a especificação do dispositivo utilizado. Nesta janela são disponibilizadas opções de forma a definir, entre outras, informações como o tipo de dispositivo e o seu endereço.

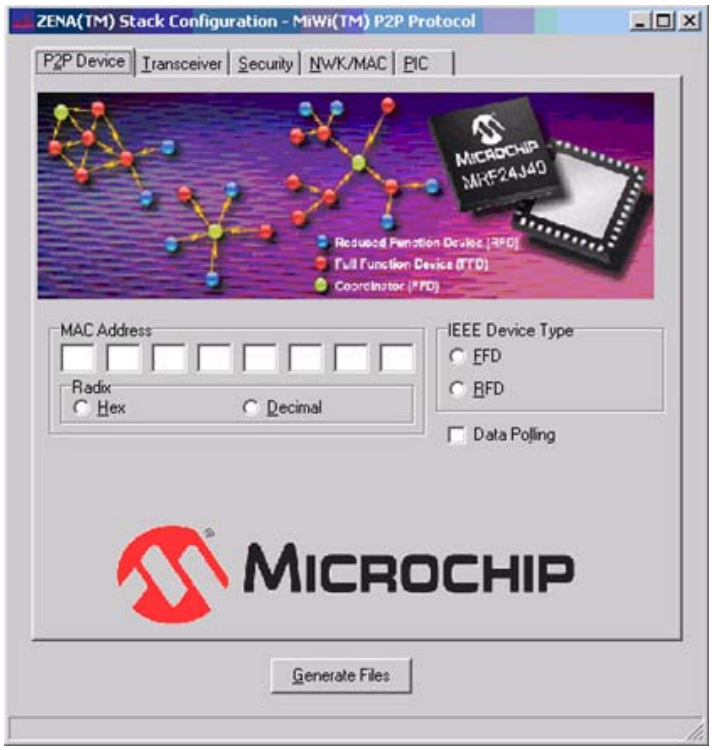


Figura 3.14 - Janela de diálogo - Especificação do dispositivo

Na tabela 3.4 estão sumariamente descritas algumas opções disponíveis na janela de diálogo para especificação do dispositivo utilizado.

Configuração	Descrição da Opção
MAC Address	Cada dispositivo tem de ter o seu próprio endereço exclusivo. O Microchip OUI é fornecido por defeito unicamente para desenvolvimento.
IEEE Device Type	Escolher se o dispositivo é do tipo FFD ou RFD. Um dispositivo RFD entra periodicamente em modo *sleep*, atribuindo-lhe assim a possibilidade de ser alimentado por uma bateria, sendo esta a principal característica que o distingue de um dispositivo FFD no protocolo MiWi P2P.
Data Polling	Seleccionar esta opção para habilitar o dispositivo RFD a pesquisar dados nos FFD associados. Nesta opção é necessário verificar se o RFD não está à espera de receber nenhuma mensagem de outro dispositivo.

Tabela 3.4 - Especificação do dispositivo

Especificação do transmissor

Na figura 3.15 é possível visualizar a janela de diálogo para especificar o transmissor Wireless. Esta janela permite ao utilizador configurar o transmissor a implementar na rede sem fios e algumas características de funcionamento da rede.

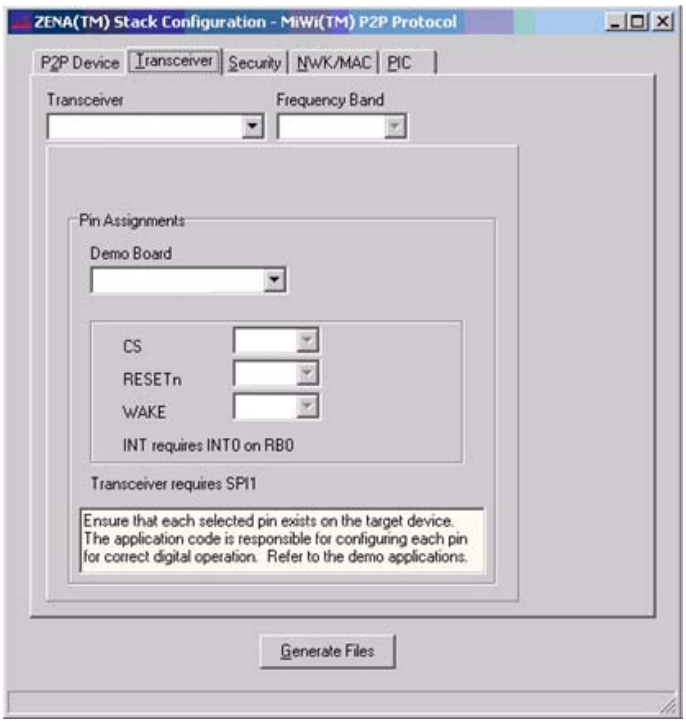


Figura 3.15 - Janela de diálogo - Especificação do transmissor

Na tabela 3.5 são descritas de forma simplificada os campos de selecção disponíveis na janela de configuração do transmissor Wireless.

Configuração	Descrição da Opção
Transceiver	Seleccionar um dos transmissores suportados pelo Stack.
Frequency Band	Seleccionar uma banda de frequência disponível para o transmissor escolhido.
Pin Assignments	O Stack necessita de determinados pinos I/O para interface com o transmissor. Se for utilizado o PICDEM Z(1) ou a placa de demonstração Explorer 16 deverá ser seleccionada a opção de configuração automática. No caso de um hardware personalizado a especificação correcta dos pinos I/O fica ao encargo do utilizador.

Tabela 3.5 - Especificação do transmissor

Especificação da segurança

Na figura 3.16 é visível a janela de diálogo para configurar a segurança da rede. Nesta janela é possível, preenchendo os campos associados, definir as configurações da segurança da comunicação sem fios.

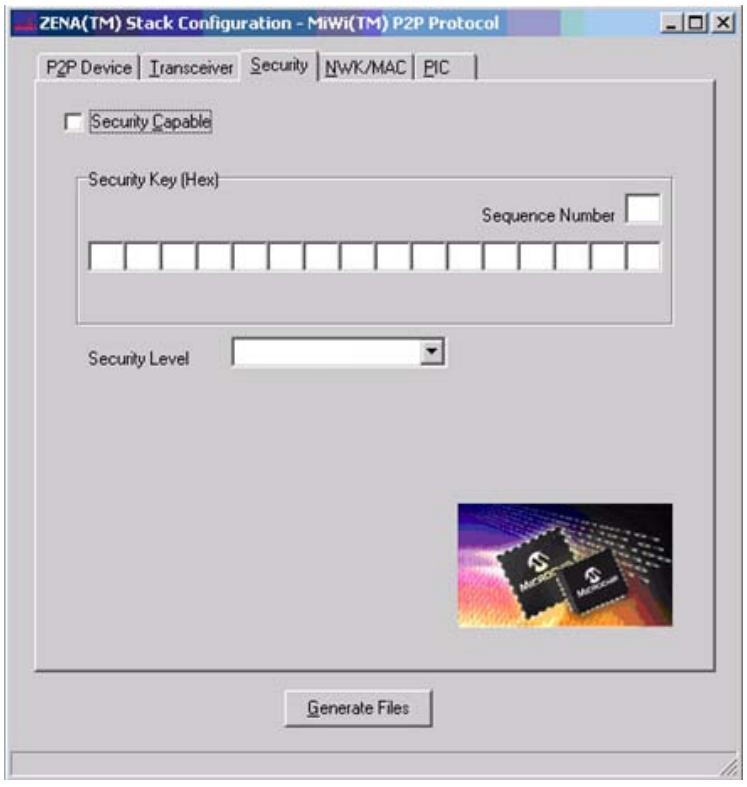


Figura 3.16 - Janela de diálogo - Especificação da segurança

Na tabela 3.6 é realizada uma breve descrição sobre os vários parâmetros configurativos da janela de especificação da segurança.

(1) – Kit de demonstração do protocolo

Configuração	Descrição da Opção
Security Capable	Activar esta opção para habilitar os recursos de segurança no MiWi P2P Stack.
Security Key	Esta é a chave de segurança de 16 bytes para o mecanismo de segurança AES, juntamente com o número de sequência da chave de segurança.
Security Level	Seleccionar o nível de segurança IEEE definida na especificação da norma IEEE 802.15.4.

Tabela 3.6 - Especificação da segurança

Especificação da Network e da layer MAC

A figura 3.17 apresenta a janela de diálogo para a especificação da *Network* (*NWK*) e da *layer MAC*. Nesta janela estão disponíveis campos para o preenchimento de várias características associadas às mensagens transmitidas. Através destes campos é possível definir, entre outras, variáveis como o tamanho das mensagens transmitidas durante a comunicação Wireless.

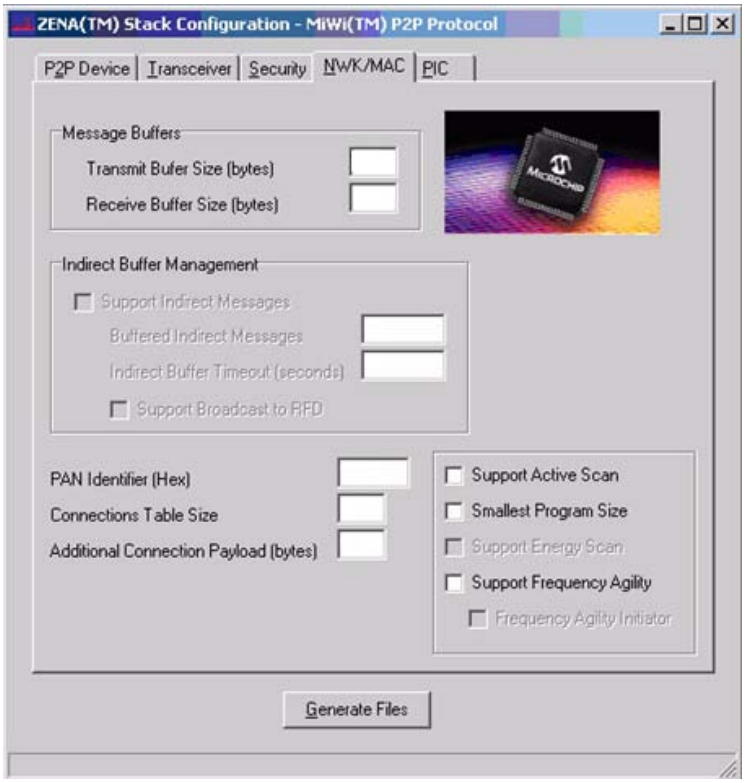


Figura 3.17 - Janela de diálogo - Especificação da NWK e da layer MAC

Na tabela 3.7 são descritos os campos disponíveis na janela de configuração da *NWK* e da *layer MAC*.

Configuração	Descrição da Opção
Transmit Buffer Size (bytes)	Inserir o número máximo de bytes de uma mensagem enviada. O maior número máximo possível é 127 bytes.
Receive Buffer Size (bytes)	Inserir o número máximo de bytes de uma mensagem recebida. O maior número máximo possível é 127 bytes.
Indirect Buffer Management	Esta opção apenas está disponível para dispositivos FFD. Se o sistema suporta mensagens indirectas, então é necessário definir o *Buffered Indirect Messages* e o *Indirect Buffer Timeout*.
PAN Identifier	*Personal Area Network* – Identificador utilizado no *MiWi P2P Stack*.
Connections Table Size	Define o número máximo de ligações P2P que um dispositivo pode manter.
Additional Connection Payload (bytes)	Esta é a informação adicional a ser transferida durante o estabelecimento da ligação para identificar o dispositivo. Esta definição é específica da aplicação e o *MiWi P2P Stack* não irá aumentar o *payload* da ligação.
Support Active Scan	A escolha desta opção irá habilitar o *MiWi P2P Stack* a realizar uma pesquisa activa de modo a adquirir a totalidade dos *MiWi P2P PANs*.
Smallest Program Size	Ao seleccionar esta opção será permitido ao *MiWi P2P Stack* diminuir o programa para o menor tamanho possível. Determina os recursos que serão desactivados, como a comunicação entre PANs e a verificação da existência de novos dados no modo de Segurança.
Support Energy Scan	Esta opção apenas está disponível para dispositivos FFD. Ao seleccionar esta opção será permitido ao *MiWi P2P Stack* realizar um exame de detecção de energia para conhecer os níveis de ruído em frequências diferentes. Esta função ajuda a determinar o canal ideal para utilizar.
Support Frequency Agility	Ao activar esta opção será permitido ao dispositivo alternar de canal durante uma operação de forma a se adaptar às variações de ruído na rede. Se for um dispositivo FFD e a opção *Support Energy Scan* estiver activa, a opção *Frequency Agility Initiator* poderá ser seleccionada permitindo a operação de agilidade frequencial.

Tabela 3.7 - Especificação da *NWK* e da *layer MAC*

Especificação do PIC MCU

A especificação do microcontrolador utilizado na implementação da comunicação é realizada através da janela de diálogo apresentada na figura 3.18. Através desta janela é possível definir

a família do microcontrolador e algumas características associadas à velocidade de comunicação entre o microcontrolador e a antena transmissora.

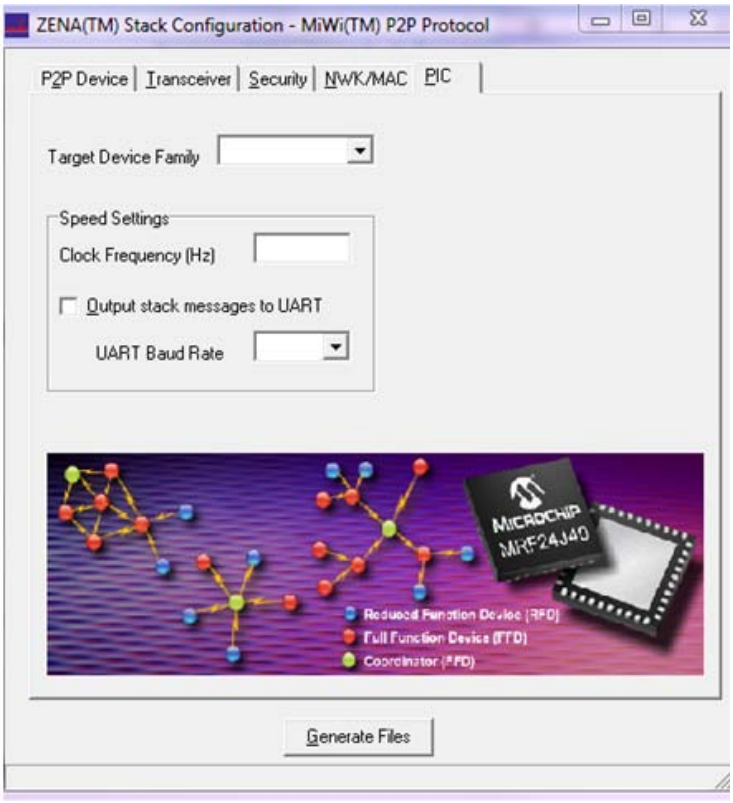


Figura 3.18 - Janela de diálogo - Especificação do PIC MCU

Na tabela 3.8 é apresentada uma breve descrição sobre os parâmetros possíveis de serem definidos na janela de configuração do microcontrolador.

Configuração	Descrição da Opção
Target Device Family	Seleccionar a família do dispositivo do processador da aplicação de destino.
Clock Frequency (Hz)	Especificar a frequência do relógio interno para o PIC MCU. Este valor é importante uma vez que todos os temporizadores internos do protocolo MiWi P2P serão colocados fora do mesmo.
Output Stack Messages to UART	Esta opção é direccionada tanto para o uso do PICDEM Z como da placa de demonstração Explorer 16. A mesma fará com que as mensagens relativas às operações do Stack sejam enviadas para o UART com o objectivo de serem exibidas num terminal. Esta opção necessita da selecção de uma taxa de transmissão de dados.

Tabela 3.8 - Especificação do PIC MCU

Geração dos ficheiros configurativos

Quando todas as opções são definidas de forma adequada, o software verifica se todos os campos obrigatórios adquirem valores adequados e se todas as especificações do protocolo são atendidas.

3.2.3 Monitorização da rede

A monitorização de uma rede de protocolo MiWi P2P é semelhante à de uma rede de protocolo ZigBee. Na Figura 3.19 é visível a janela de configuração da monitorização de uma rede de protocolo MiWi P2P.

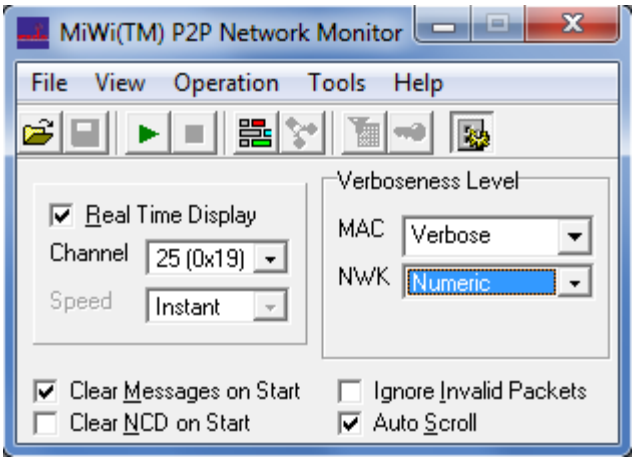


Figura 3.19 - Janela de configuração da monitorização de uma rede de protocolo MiWi P2P

A tabela 3.9 caracteriza de forma sucinta algumas opções disponíveis na janela de configuração apresentada na figura 3.19.

Configuração	Descrição da Opção
Real Time Display	Seleccionar esta opção para que sejam exibidas as mensagens que estão a ser recebidas pelo analisador.
Channel	Seleccionar o canal a monitorizar. Pode ser necessário seleccionar vários canais ate encontrar a rede. Esta selecção só pode ser alterada quando a monitorização é interrompida.
Ignore Invalid Packets	Esta opção permite filtrar volumes de mensagem inválidos, por exemplo ruídos indesejados.

Tabela 3.9 - Configuração da monitorização de uma rede de protocolo MiWi P2P

Cada mensagem contém uma grande quantidade de informação, tornando por vezes difícil a visualização no ecrã. O software Zena oferece então três níveis distintos para visualizar a

MAC, a *NWK*. Cada *layer* pode ser configurada separadamente na janela da figura 3.19, seleccionando no *Verboseness Level* um dos três níveis apresentados na tabela 3.10.

Configuração	Descrição da Opção
Verbose	Os *headers* são fornecidos com uma descrição do valor correspondente.
Numeric	Os *headers* são fornecidos com um valor numérico do respectivo campo.
Condensed	Nenhum *header* é fornecido. Todos os bytes são representados numericamente, com o byte menos significativo em primeiro.

Tabela 3.10 - Níveis de configuração do *Verboseness Level*

3.3 Protocolo de comunicação série – Protocolo SPI

O módulo *Synchronous Serial Port* (SSP) é uma interface série utilizada em comunicações entre o microcontrolador e outro dispositivo periférico ou microcontrolador [7-8]. O módulo SSP pode operar em dois modos distintos:

- *Serial Peripheral Interface* – SPI
- *Inter-Integrated Circuit* – I^2C

O modo SPI, utilizado neste projecto, permite enviar e receber 8 bits de dados de forma síncrona e simultânea.

Para realizar esta comunicação são tipicamente utilizados 3 pinos:

- *Serial Data Out* – SDO
- *Serial Data In* – SDI
- *Serial Clock* – SCK

Pode ainda ser utilizado no modo de operação *Slave* um quarto pino:

- *Slave Select* – SS

Quando a comunicação SPI é inicializada é necessário configurar o registo SSPCON do microcontrolador. Este registo está associado a várias especificações configurativas referentes ao modo de funcionamento (*Master/Slave*) e ao *clock* de comunicação.

Na figura 3.20 está representado o diagrama de blocos da comunicação SSP operando no modo SPI.

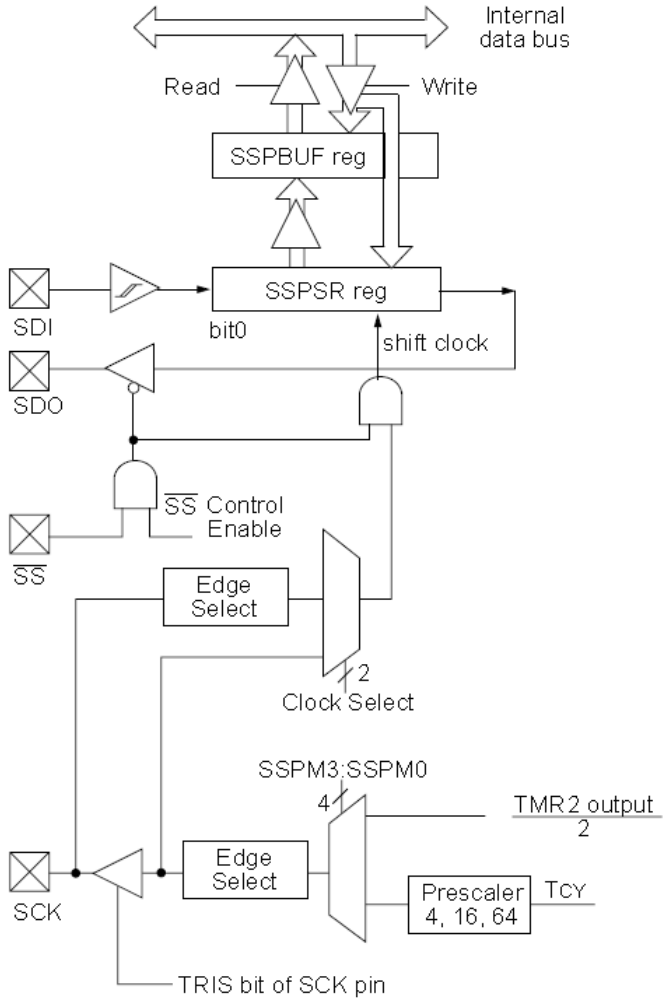


Figura 3.20 - Diagrama de blocos da comunicação SSP - Modo SPI

O SSP diz respeito ao registo de transmissão e recepção de dados (SSPSR) e ao registo do buffer da comunicação (SSPBUF). O registo SSPSR é responsável pela transferência de dados para dentro e fora do dispositivo, por sua vez o registo SSPBUF contém os dados escritos no SSPSR enquanto os dados recebidos são lidos. Uma vez que os 8 bits que contêm os dados da comunicação são bem recebidos, este byte é movido para o registo SSPBUF. De imediato é activado o bit SSPSTAT e o bit SSPIF, não representados no diagrama anterior, indicando respectivamente que a transmissão foi realizada com sucesso e gerada uma nova interrupção. Este *buffer* duplo dos dados recebidos (SSPBUF) permite iniciar a recepção do próximo byte antes de ler os dados que acabou de receber. Quaisquer outros dados escritos para o registo SSPBUF durante a transmissão/recepção de dados serão ignorados e o bit responsável pela detecção de colisão activado. Este bit deve ser inicializado a “0” pelo software para que possa determinar se o byte corrente foi recebido com êxito pelo SSPBUF. Enquanto o software aplicativo aguarda pela recepção de novos dados, o SSPBUF deve ser

limpo antes que um novo byte de dados inicie a transferência. Na figura 3.21 é possível visualizar a conexão SPI típica entre dois microcontroladores.

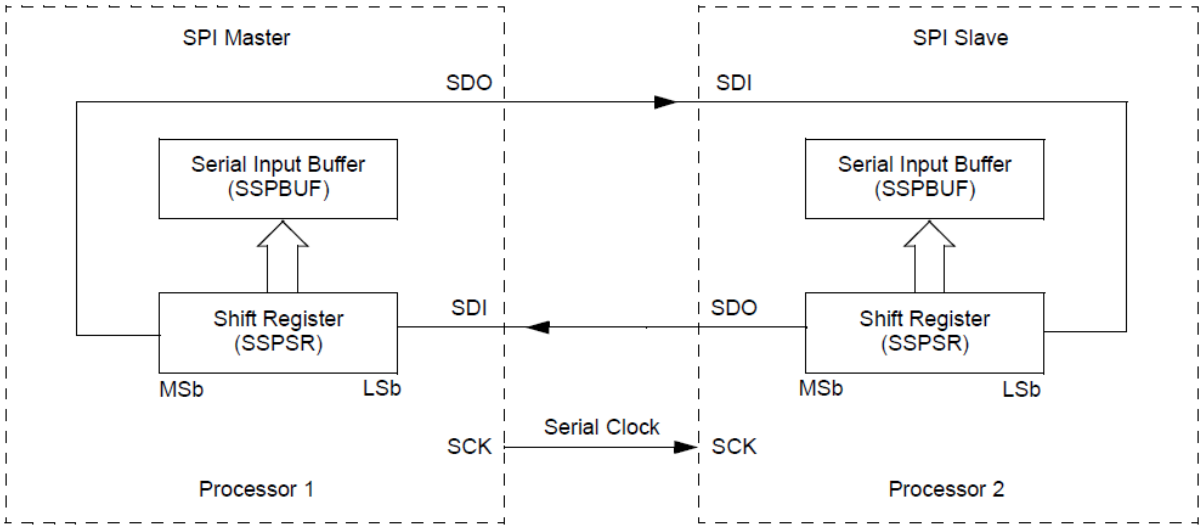


Figura 3.21 - Conexão SPI típica entre dois microcontroladores

O controlador *Master* inicia a comunicação enviando um sinal SCK *clock*. A informação é enviada por ambos os registos SSPSR à cadência programada. Ambos os controladores podem estar programados para a mesma polaridade do *clock* (*Clock Polarity* – CKP), significando isto que ambos os controladores enviam e recebem informação ao mesmo tempo. Se os dados são significativos (ou não significativos) depende da aplicação de software. Isso leva a três cenários possíveis para a transmissão de dados:

- *Master* envia dados significativos – *Slave* envia dados não significativos
- *Master* envia dados significativos – *Slave* envia dados significativos
- *Master* envia dados não significativos – *Slave* envia dados significativos

Os dois microcontroladores responsáveis pela comunicação SPI estão configurados no modo *Master*, o que significa que a informação é enviada/recebida logo que o registo SSPBUF recebe informação nova.

Nas figuras 3.22 e 3.23 é possível visualizar respectivamente o código em C18 associado ao envio e recepção de um byte na comunicação SPI.

```
void SPIPut(BYTE v)
{
    PIR1bits.SSPIF = 0;             // a interropção é apagada
    do
    {
        SSPCON1bits.WCOL = 0;       // a detecção de colisão é apagada
        SSPBUF = v;                 // o buffer é carregado com a informação do byte v
    } while( SSPCON1bits.WCOL );

    while( PIR1bits.SSPIF == 0 );
}
```

Figura 3.22 - Código associado ao envio de um byte na comunicação SPI

```
BYTE SPIGet(void)
{
    SPIPut(0x00);
    return SSPBUF;      // Carregar o valor do buffer
}
```

Figura 3.23 - Código associado à recepção de um byte na comunicação SPI

3.4 Resultados experimentais

Os primeiros ensaios realizados tiveram como objectivo visualizar e verificar todo o processo de criação da rede Wireless e o conteúdo das mensagens transmitidas.

Na figura 3.24 estão representadas as mensagens transmitidas, de forma bidireccional, durante o processo de criação da rede sem fios e uma mensagem de monitorização a ser posteriormente recebida pela interface computacional.

Frame	Time(us)	Len	MAC Frame Control					Seq	Dest	Dest	Source Address	Connection Request	Capability Info				FCS		
	+5801856		Type	Sec	Pend	ACK	IPAN	Num	PAN	Addr		Channel	Sec	Synch	Req	RxOn	RSSI	Corr	CRC
00001	=5801856	20	CMD	N	N	N	Y	0x88	0x1234	0xFFFF	0x1122334455667702	0xE0	Y	N	N	Y	+06	0x69	OK
Frame	Time(us)	Len	MAC Frame Control					Seq	Dest	Dest	Source Address	Connection Request	Capability Info				FCS		
	+1996976		Type	Sec	Pend	ACK	IPAN	Num	PAN	Addr		Channel	Sec	Synch	Req	RxOn	RSSI	Corr	CRC
00002	=7798832	20	CMD	N	N	N	Y	0x89	0x1234	0xFFFF	0x1122334455667702	0xE0	Y	N	N	Y	+06	0x68	OK
Frame	Time(us)	Len	MAC Frame Control					Seq	Dest	Destination Address	Source Address	Connection Request Response	Capability Info				FCS		
	+11024		Type	Sec	Pend	ACK	IPAN	Num	PAN			Status	Sec	Synch	Req	RxOn	RSSI	Corr	CRC
00003	=7809856	26	CMD	N	N	Y	Y	0xFF	0x1234	0x1122334455667702	0x1122334455667703	0x00	N	N	N	Y	-19	0x6B	OK
Frame	Time(us)	Len	MAC Frame Control					Seq	FCS										
	+1568		Type	Sec	Pend	ACK	IPAN	Num	RSSI	Corr	CRC								
00004	=7811424	5	ACK	N	N	N	N	0xFF	+06	0x69	OK								
Frame	Time(us)	Len	MAC Frame Control					Seq	Dest	Dest	Source Address	Payload	FCS						
	+143328		Type	Sec	Pend	ACK	IPAN	Num	PAN	Addr		0x4A 0xE0	RSSI	Corr	CRC				
00005	=7954752	21	DATA	N	N	N	Y	0x00	0x1234	0xFFFF	0x1122334455667703	0x6A 0x6F	-19	0x6A	OK				

Figura 3.24 - Mensagens transmitidas durante o processo de criação da rede Wireless

Observando a figura 3.24 é visível a transferência de três pacotes durante o processo de criação da rede. Inicialmente o dispositivo envia um pacote correspondente ao pedido de conexão P2P, não sendo este respondido, após um período de tempo de aproximadamente 2 segundos, o mesmo dispositivo volta a enviar um novo pedido de conexão (segundo pacote). O terceiro pacote diz respeito à resposta do outro dispositivo e consequente criação da rede P2P. Como a resposta ao pedido de conexão P2P é enviada com um bit especial de

reconhecimento, é posteriormente enviado um pacote referente a esse reconhecimento. Após este processo de inicialização é gerada a troca de dados entre os dois dispositivos da rede.

Para verificar o alcance da comunicação Wireless foram efectuados alguns testes. Estes testes foram realizados num ambiente aberto sem obstáculos e num espaço interior composto por paredes e portas.

Os ensaios realizados no exterior tiveram lugar no campus da Faculdade de Engenharia da Universidade do Porto (figura 3.25).

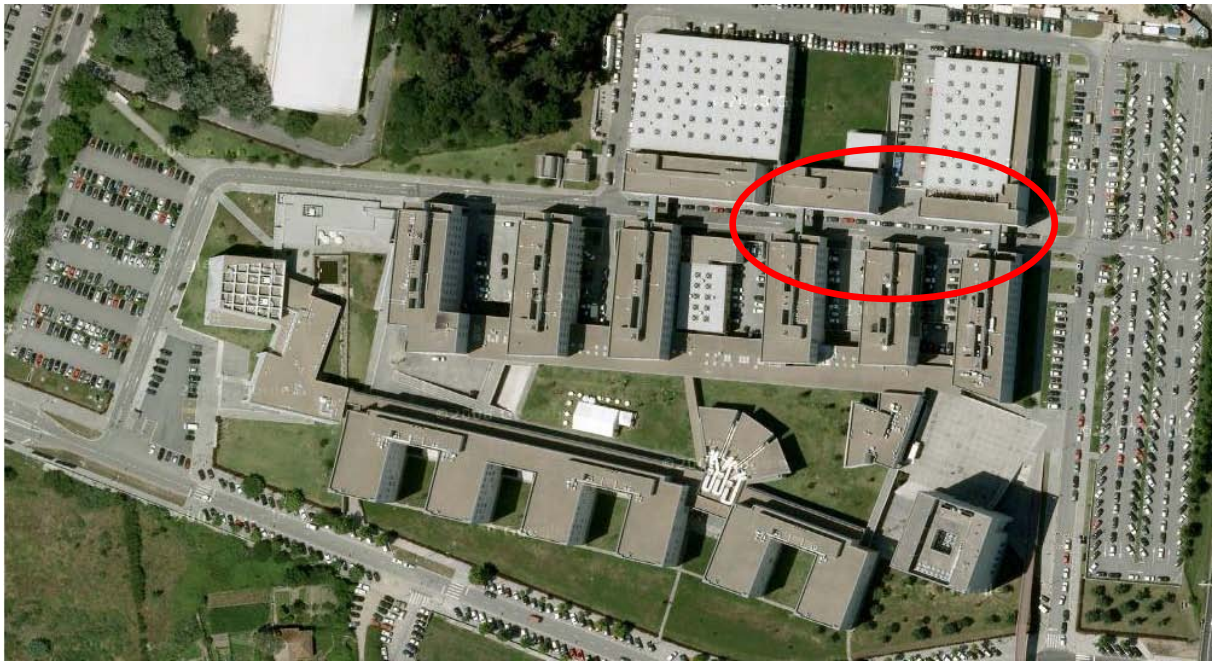


Figura 3.25 - Local exterior de ensaios da comunicação Wireless

Observando atentamente a zona assinalada, através da figura 3.26, é possível verificar a posição dos órgãos terminais da comunicação Wireless no momento em que foi realizado o ensaio que obteve melhores resultados.

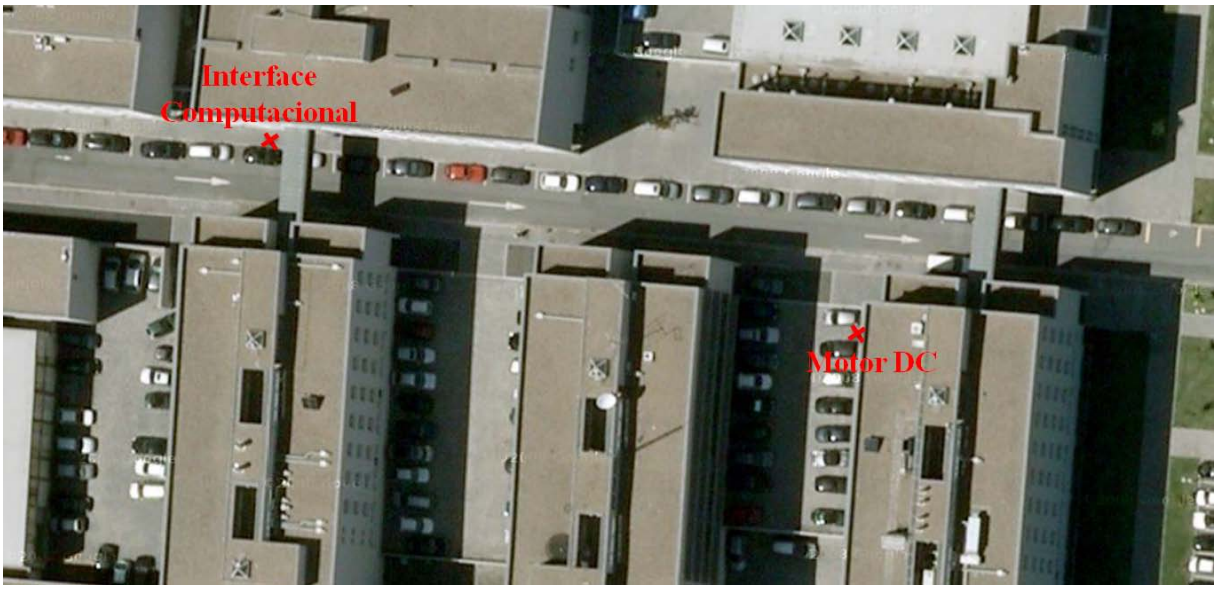


Figura 3.26 - Posicionamento dos órgãos terminais da comunicação Wireless no ensaio realizado no exterior

O posicionamento referido na figura 3.26 distancia os órgãos da comunicação Wireless em cerca de 50 m. Esta foi a distância máxima à qual foi possível estabelecer a comunicação cumprindo o comando e monitorização dos movimentos do motor.

Os ensaios realizados no interior do edifício tiveram como principal objectivo visualizar o comportamento da comunicação sem fios face a obstáculos como portas e paredes.

Na figura 3.27 está representado o posicionamento dos órgãos terminais da comunicação Wireless nos ensaios mais significativos realizados no interior do edifício.

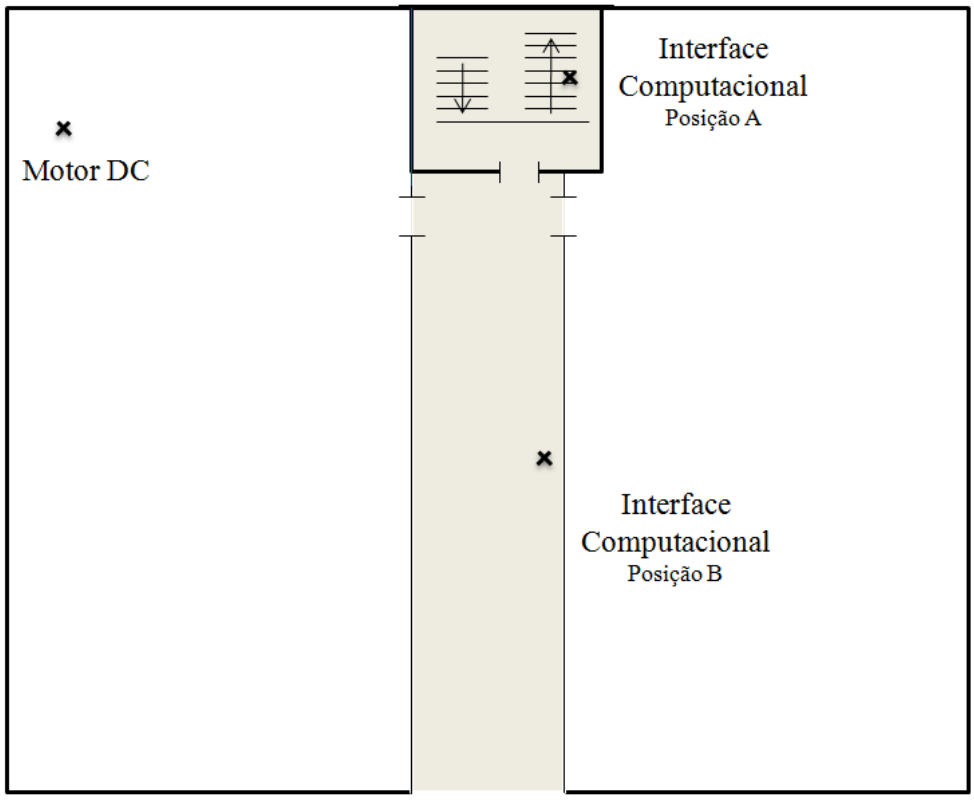


Figura 3.27 - Posicionamento dos órgãos terminais da comunicação Wireless nos ensaios realizados no interior

Estes ensaios foram realizados a uma distância aproximada a 7 m para a posição A e 10 m para a posição B.

Como seria de esperar, os ensaios efectuados no exterior obtiveram um alcance significativamente maior. Estes resultados devem-se à reduzida existência de obstáculos, o que facilita a transferência das ondas rádio. No entanto estes resultados são muito inferiores aos 100 m indicados pelo protocolo.

Esta comunicação apresenta uma velocidade de transferência de dados inferior ao limite máximo imposto pelo protocolo. A velocidade de comunicação é influenciada por vários factores, entre eles a capacidade que o microcontrolador possui de enviar os dados para a antena e a quantidade de informação enviada por cada trama transmitida.

Na aplicação implementada é apenas necessário enviar 4 bytes no sector Payload em cada trama transmitida, valor este muito inferior aos 102 disponibilizados pelo protocolo.

Utilizando o software Zena foi possível estimar que a comunicação Wireless implementada possui uma frequência de transferência aproximada de 20 Hz, isto significa que em cada segundo a comunicação Wireless transfere bidireccionalmente 20 pacotes. Sendo este um valor satisfatório para a frequência de actualização do valor da velocidade monitorizada pelo utilizador.

Sabendo que cada pacote é constituído aproximadamente por 29 bytes, é possível obter um valor aproximado para a velocidade de comunicação Wireless implementada de 4,6 kbps.

Desta forma, se a frequência de transferência de pacotes não fosse limitada pela programação e se o tamanho de cada pacote permanecesse com 29 bytes, poder-se-ia afirmar que, no limite, a frequência máxima teórica para a transferência de pacotes seria aproximadamente 1kHz.

3.5 Conclusões

Os resultados obtidos nos vários testes realizados foram bastante satisfatórios. Durante a execução deste trabalho foi possível criar uma rede de comunicação Wireless bidireccional e de baixo consumo. Também a velocidade de transmissão de dados obtida mostrou-se perfeitamente adequada para a aplicação.

Desta forma, todos os objectivos referentes à comunicação sem fios propostos inicialmente foram cumpridos com sucesso.

Capítulo 4

Interface e Comunicações Cabladas

As interfaces entre utilizadores e sistemas de actuação surgem na indústria em diferentes formatos. Dependendo da aplicação podem ser utilizadas consolas ou até mesmo aplicações computacionais para comandar e monitorizar as acções de um sistema.

As comunicações cabladas aparecem neste projecto como meio de comunicação entre o computador e o microcontrolador associado à comunicação USB e entre os microcontroladores responsáveis pela comunicação USART.

Neste capítulo é apresentada a interface computacional criada e desenvolvida para esta aplicação, assim como descritos, de forma sucinta, os protocolos de comunicação cablada utilizados neste projecto.

4.1 Interface computacional

A interface entre o sistema e o utilizador deve ser realizada de modo a ser:

- Perceptível;
- Intuitivo;
- Simples;
- De agradável visualização / navegação;
- Funcional.

Foi com esta preocupação que toda a interface foi desenvolvida de forma a responder à vontade do utilizador com rapidez e eficácia no controlo e monitorização do sistema de actuação. Foi utilizado o Visual Studio 2008, programado em linguagem Visual Basic, para criar e desenvolver toda a interface computacional do sistema.

Na figura 4.1 é apresentada a página de apresentação da interface computacional.

Figura 4.1 - Página de apresentação da interface computacional

Ao clicar no botão "Entrar", visível na interface de apresentação, aparecerá a interface de comando e monitorização das acções do motor. Esta segunda interface é apresentada na figura 4.2.

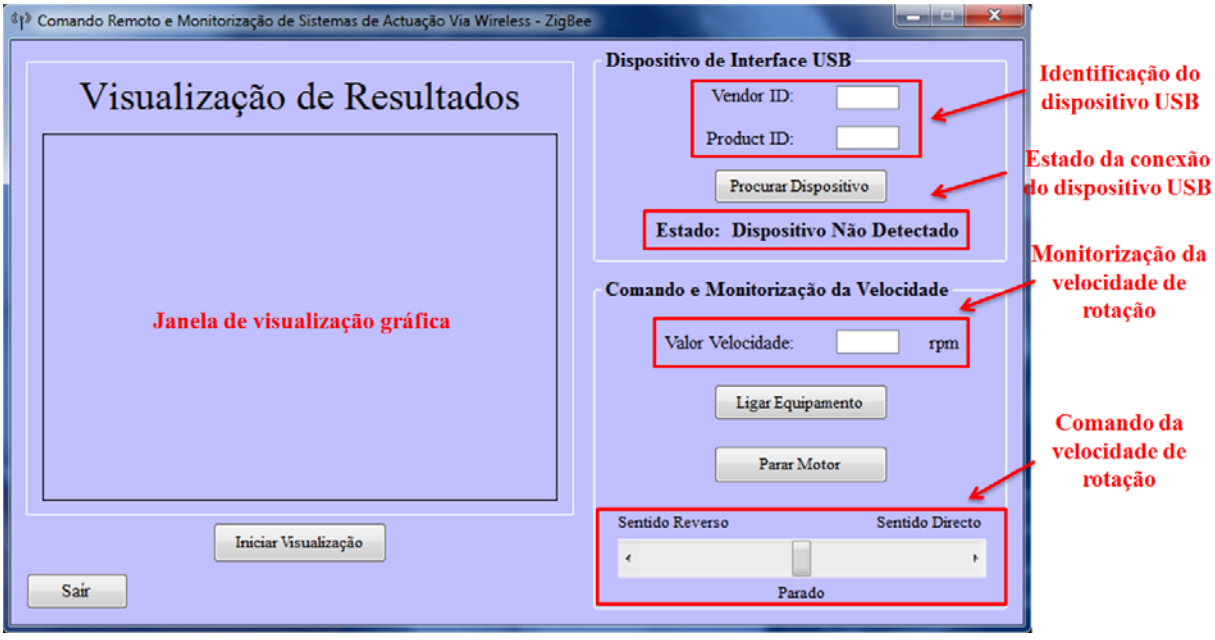


Figura 4.2 - Interface de comando e monitorização das acções do motor

Antes de iniciar o comando ou a monitorização das acções do motor é necessário verificar se o dispositivo USB se encontra ligado à porta USB do computador através do botão “Procurar Dispositivo”.

O comando da velocidade do motor é realizado através do cursor identificado na figura 4.2. O posicionamento do cursor na posição central corresponde a uma velocidade de comando nula. A velocidade aumenta a partir do momento em que o cursor se desloca para qualquer um dos lados, atingindo o seu valor máximo na posição extremo.

4.2 Comunicação USB

A comunicação USB é utilizada para estabelecer a ligação entre a interface computacional e o microcontrolador. Para implementar esta comunicação foi utilizado o PIC18F2550. A família deste microcontrolador possuir um módulo periférico interno dedicado à comunicação USB e compatível com comunicações de alta e baixa velocidade.

4.2.1 Protocolo USB

O protocolo USB é um dos mais utilizados no dia de hoje nomeadamente para a comunicação entre periféricos e computadores [15]. Vários dispositivos tais como, telemóveis, ratos, teclados, impressoras, máquinas fotográficas e armazenadores de memória utilizam este protocolo para comunicar com outro dispositivo, normalmente um computador.

Actualmente o protocolo USB suporta taxas de transferência de dados até os 480 Mbps, este valor é aplicado a comunicações USB 2.0.

Suporte Físico

Os cabos utilizados na comunicação USB possuem quatro fios, dois desses fios são utilizados na alimentação do dispositivo periférico, sendo os restantes reservados à comunicação. Na figura 4.3 é possível visualizar a estrutura física de um cabo USB.

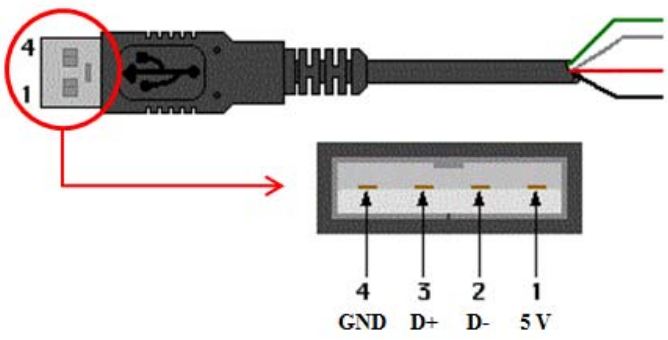


Figura 4.3 - Estrutura física de um cabo USB

Os fios reservados à transferência de dados transportam a informação sob a forma de um sinal diferencial, isto é, possuem a mesma magnitude e polaridade inversa. Esta forma de sinal reduz qualquer interferência que possa aparecer no fio. Na figura 4.4 é possível observar de forma simplificada o conceito de sinal diferencial.

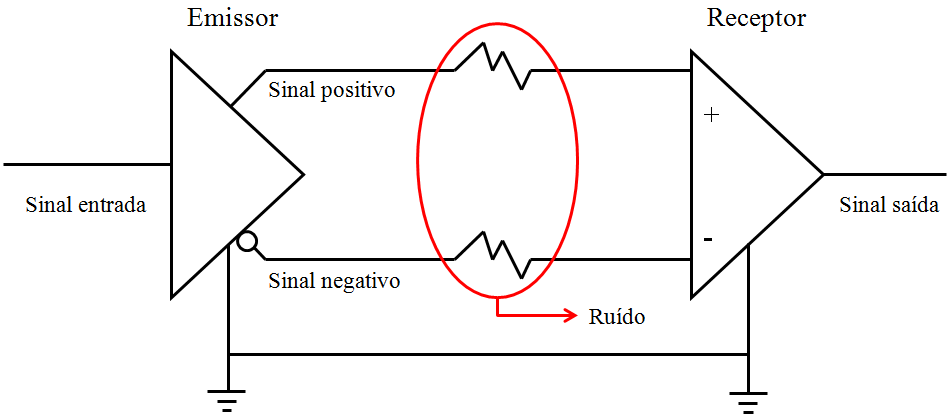


Figura 4.4 - Sinal diferencial

Utilizando um osciloscópio foi possível verificar a utilização de um sinal diferencial na comunicação USB. A figura 4.5 pretende retratar unicamente a inversão de polaridade nos fios da comunicação USB visível no monitor do osciloscópio.

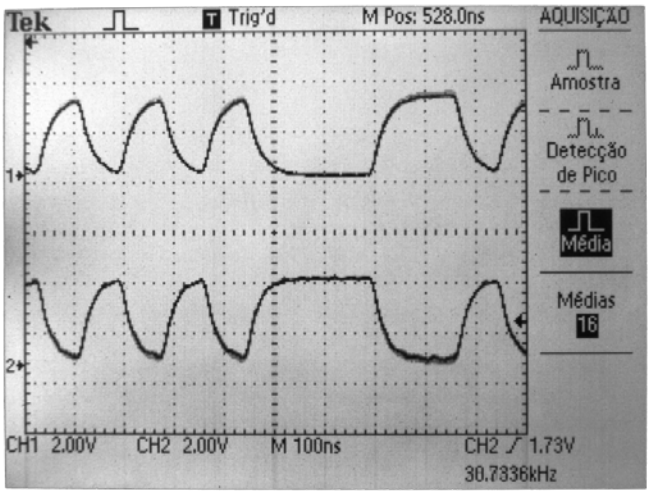


Figura 4.5 - Comunicação diferencial - USB

Composição dos pacotes transmitidos

As mensagens são enviadas no protocolo USB sob a forma de pacotes, cada pacote é composto por sectores e cada sector transporta informação bem definida.

Cada pacote possui um sector de endereço destinado à identificação do dispositivo receptor da mensagem.

Numa comunicação USB o computador associado a essa mesma comunicação é denominado *HOST*. Esse computador possui um software que efectua o controlo do barramento do sistema em comunicação.

Todas as mensagens enviadas pelo barramento USB necessitam essencialmente de 3 pacotes. Inicialmente o *HOST* envia um pacote *Token* com o endereço do dispositivo com o qual será realizada a comunicação. Este pacote é essencial à comunicação pois existe a possibilidade de vários dispositivos estarem ligados às portas USB do computador. De seguida o dispositivo ou o *HOST* envia um pacote com os dados a transmitir. A comunicação é finalizada com o envio de um pacote de reconhecimento, garantindo que o receptor recebeu a mensagem. Pode eventualmente existir um quarto pacote utilizado para funções adicionais.

Na figura 4.6 estão representadas de forma esquemática as etapas da comunicação USB.

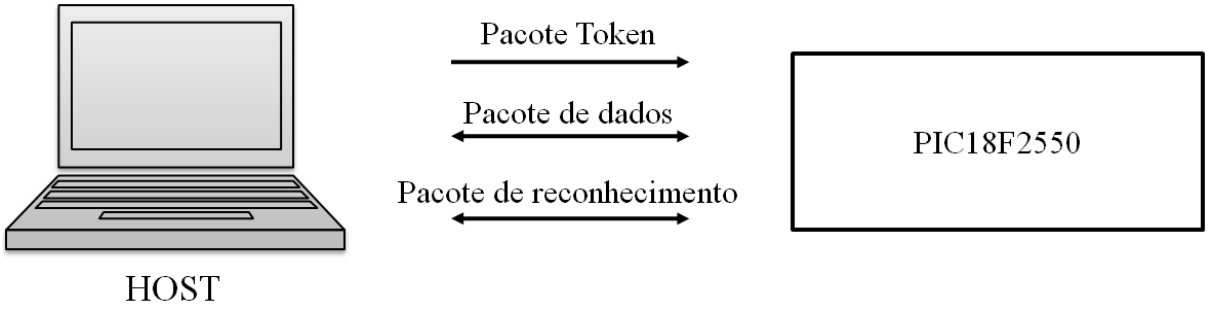


Figura 4.6 - Etapas da comunicação USB

Todos os pacotes transmitidos nesta comunicação iniciam com dois sectores de 8 bits. O primeiro sector é um campo de sincronismo e o segundo destina-se à identificação. O campo de sincronismo gera uma sequência de bits para que todos os dispositivos ligados ao barramento reiniciem o seu relógio sincronizando-o com o relógio do *HOST*, sendo deste modo garantido o sincronismo de toda a comunicação. O sector de identificação transporta quatro bits de identificação do tipo de pacote e quatro bits de identificação do seu sub-tipo. Na tabela 4.1 é apresentado os vários tipos de pacotes.

Tipos de pacotes	
Bits de identificação	**Tipo de pacote**
XX00XX11	Especial
XX01XX10	Token
XX10XX01	Reconhecimento
XX11XX00	Dados

Tabela 4.1 - Tipos de pacotes na comunicação USB

Os **pacotes do tipo *Token*** são unicamente enviados pelo HOST. Estes pacotes são constituídos por quatro bytes divididos da seguinte forma (figura 4.7):

8 Bits	8 Bits	7 Bits	4 Bits	5 Bits
Sector de Sincronização	Sector de Identificação	Sector de Endereço	Sector End-Point	Detecção de Erro

Figura 4.7 - Estrutura do pacote do tipo *Token* na comunicação USB

Na tabela 4.2 são apresentados os quatro sub-tipos de pacotes *Token* existentes na comunicação USB.

Bits de identificação	Sub-Tipo do pacote *Token*
00011110	Saída
01011010	Inicio de Pacote
10010110	Entrada
11010010	Setup

Tabela 4.2 - Sub-Tipos do pacote *Token*

O sub-tipo “Saída” indica ao dispositivo USB que o *HOST* deseja escrever-lhe uma mensagem. O subtipo “Entrada” por sua vez indica ao dispositivo USB que o *HOST* deseja ler uma mensagem sua. O sub-tipo “Inicio de Pacote” é utilizado para sincronizar com o dispositivo USB as mensagens subsequentes, este tipo de pacote é enviado periodicamente e em forma *broadcast*. O sub-tipo “*Setup*” é utilizado para configurar a comunicação com o dispositivo USB.

Os **pacotes do tipo Dados**, podem ser enviados tanto pelo *HOST* como pelo dispositivo USB e são utilizados para enviar a informação no sistema USB. Estes pacotes são constituídos da seguinte forma (figura 4.8):

8 Bits	8 Bits	De 0 a 1023 Bits	13 Bits
Sector de Sincronização	Sector de Identificação	Sector de dados	Detecção de Erro

Figura 4.8 - Estrutura do pacote do tipo Dados na comunicação USB

Existem dois sub-pacotes para pacotes do tipo Data. Estes sub-pacotes, Data 0 e Data 1, são utilizados alternadamente de forma a indicar que nenhuma mensagem de dados ou resposta de reconhecimento é perdida durante a comunicação. Na tabela 4.3 são representados os dois sub-tipos de pacotes dados utilizados na comunicação USB.

Bits de identificação	Sub-Tipo do pacote Dados
00111100	Dados 0
10110100	Dados 1

Tabela 4.3 - Sub-Tipos do pacote Dados

Os **pacotes do tipo Reconhecimento** utilizam unicamente os dois bytes comuns a todos os pacotes USB. Na figura 4.9 está representada a estrutura do pacote de reconhecimento.

8 Bits	8 Bits
Sector de Sincronização	Sector de Identificação

Figura 4.9 - Estrutura do pacote do tipo Reconhecimento na comunicação USB

Este pacote possui dois sub-tipos, *Acknowledgment* (*ACK*) e Non-*Acknowledgment* (*NAK*). O sub-tipo ACK é utilizado como resposta de reconhecimento a um pacote de dados, indicando o sucesso da sua recepção. O sub-tipo NAK indica que o dispositivo permaneceu temporariamente sem receber ou enviar qualquer mensagem. Na tabela 4.4 estão representados estes dois sub-tipos do pacote de reconhecimento.

Bits de identificação	Sub-Tipo do pacote Reconhecimento
00101101	ACK
01101001	NAK

Tabela 4.4 - Sub-Tipos do pacote Reconhecimento

Configuração do periférico USB

As operações do módulo USB são configuradas e geridas através de três registos de controlo. Existem ainda 19 registos utilizados para gerir as operações da comunicação USB [6]. Desta forma para configurar todo o módulo de comunicação USB é necessário definir os seguintes registos:

- *USB Control register* (UCON)
- *USB Configuration register* (UCFG)
- *USB Transfer Status register* (USTAT)
- *USB Device Address register* (UADDR)
- *Frame Number registers* (UFRMH:UFRML)
- *Endpoint Enable registers* (UEPn onde n pode tomar valores de 0 a 15)

Pode ser encontrada informação mais detalhada sobre cada um destes registos no *Data Sheet* do microcontrolador PIC18F2550 fornecido gratuitamente pela Microchip.

O módulo USB permite criar várias condições lógicas para gerar interrupções. De forma a ajustar todas as combinações, o módulo possui uma estrutura lógica própria e similar à do microcontrolador. Estas interrupções são criadas através da activação de bits próprios originários de registos de controlo e variáveis provenientes de registos de *flags*. Na figura 4.10 está representado o diagrama lógico do módulo USB para a geração de interrupções.

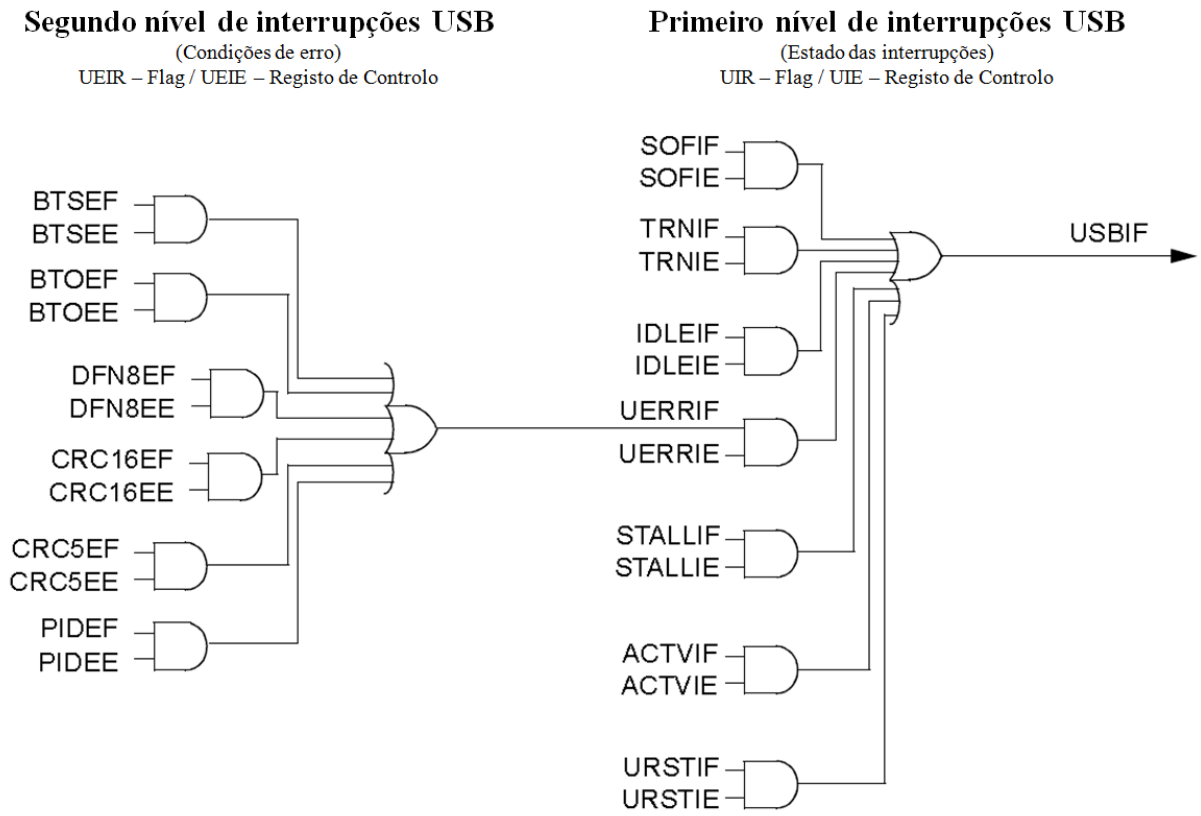


Figura 4.10 - Diagrama lógico do módulo USB para a geração de interrupções

De forma a se perceber melhor a correspondência entre cada bit e o registo/*flag* associado é apresentado na tabela 4.5 a constituição de cada registo de controlo e cada *flag* utilizada na geração de interrupções.

Nome	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
UEIR	BTSEF	-	-	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF
UEIE	BTSEE	-	-	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE
UIR	-	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF
UIE	-	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE

Tabela 4.5 - Constituição dos registos/flags associados às interrupções USB

Descritor do dispositivo USB

O descritor do dispositivo é um conjunto de informação de identificação do dispositivo USB. Uma espécie de bilhete de identidade que o *HOST* tenta decifrar quando o dispositivo é conectado.

Cada dispositivo USB possui apenas um descritor. Este proporciona informação geral como o fabricante, o número de série, o número do produto e a classe do dispositivo.

Na tabela 4.6 é realizada uma pequena descrição sobre cada informação presente num descritor de um dispositivo USB.

Nome	Descrição
bLength	Comprimento do descritor em bytes
bDescriptorType	Tipo de descritor
bcdUSB	Versão mais recente suportada (USB 1.1 ou USB 2.0)
bDeviceClass	Esta informação é fornecida pela organização USB e é utilizada pelo sistema para detectar a *classe driver* para o dispositivo
bDeviceSubClass	
bDeviceProtocol	
bMaxPacketSize0	Tamanho máximo para cada pacote transmitido.
idVendor	Fornecido pela organização USB e representa o *Vendor ID*
idProduct	Fornecido pela organização USB e representa o *Product ID*
bcdDevice	Número da versão do dispositivo
iManufacturer	Descrição do fabricante
iProduct	Índice da cadeia de produtos do descritor
iSerialNumber	Índice do número de série do descritor
bNumConfigurations	Número de configurações possíveis

Tabela 4.6 - Tabela de informações sobre o descritor do dispositivo USB

O software MicroC IDE possui uma ferramenta dedicada à criação do ficheiro identificativo do dispositivo USB. Esta ferramenta possibilitou criar de forma quase automática o descritor do dispositivo USB utilizado neste projecto. No Anexo B é apresentado o ficheiro identificativo do dispositivo USB gerado e aplicado na comunicação USB implementada neste projecto.

Na figura 4.11 é apresentada a janela disponibilizada pelo MicroC IDE para a criação deste ficheiro descritor do dispositivo.

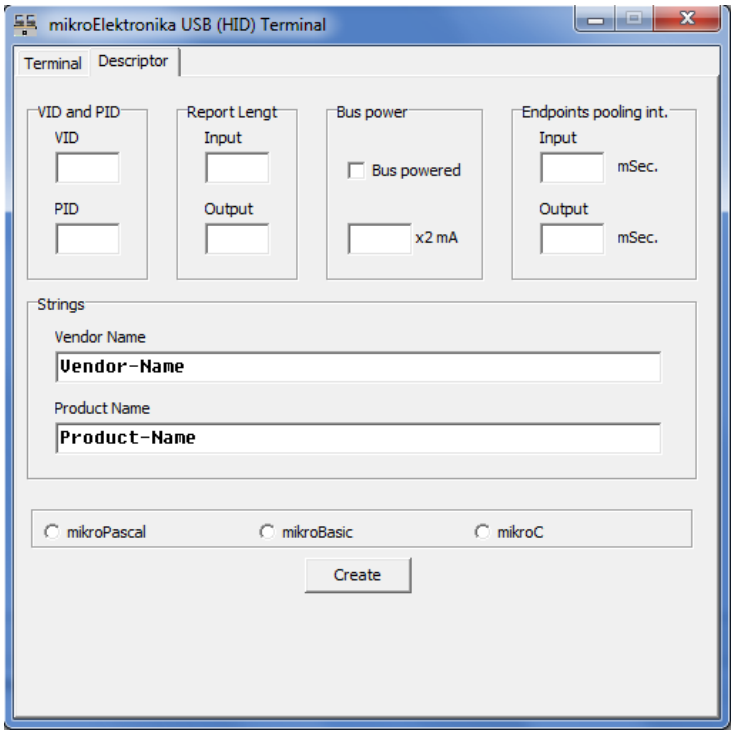


Figura 4.11 - Janela de criação do ficheiro descritor do dispositivo USB

4.3 Comunicação USART

O EUSART – *Enhanced Universal Synchronous Asynchronous Receiver Transmitter* é um módulo periférico de comunicação série Input/Output, também conhecido por *Serial Communication Interface* (*SCI*).

Este módulo pode ser configurado como um sistema assíncrono *Full-Duplex*, permitindo a comunicação com dispositivos periféricos tais como terminais CRT e computadores pessoais [6-7]. Este tipo de configuração implementa funcionalidades adicionais como o reconhecimento e calibração automática da taxa de transmissão e o *Auto-wake-up* sempre que existe um pedido de interrupção.

O USART pode ainda ser configurado como um sistema síncrono *Half-Duplex*, de forma a comunicar com circuitos integrados D/A ou A/D e *serial EEPROM*.

Neste projecto optou-se por configurar a comunicação USART como um sistema assíncrono. Esta escolha teve como principal influência a característica de auto detecção da velocidade de transmissão disponibilizada por este tipo de configuração.

4.3.1 Suporte físico

Na comunicação USART são utilizados unicamente 2 fios condutores. Cada um destes fios conduz a informação num determinado sentido. Desta forma, a comunicação USART não deve ser utilizada em grandes distâncias, pois é facilmente perturbada por ruídos externos.

4.3.2 Modo assíncrono

O envio e a recepção de mensagens na comunicação USART são implementados utilizando o formato standard *Non-Return-To-Zero* (NRZ). Este formato utiliza dois níveis de tensão: o nível V_{OH} define o nível de comunicação 1, e o nível V_{OL} define o nível de comunicação 0. Cada mensagem trocada está contida entre um bit de inicio e um bit de fim, desta forma o receptor sabe sempre qual o início e o fim da mensagem que está a receber. O bit de início é sempre assinalado através de um nível de tensão V_{OL}, por sua vez o bit de fim é sempre assinalado com um nível de tensão V_{OH}. Cada bit é transmitido durante um período de tempo bem definido. Esse intervalo de tempo pode ser calculado invertendo o valor do *Baud Rate*.

No modo assíncrono as mensagens transmitidas possuem normalmente entre 1 a 8 bits. A recepção e o envio de mensagens são funcionalidades independentes mas possuem o mesmo formato das mensagens e a mesma velocidade de comunicação. Na figura 4.12 é possível visualizar o formato de um pacote de dados enviado através da comunicação USART.

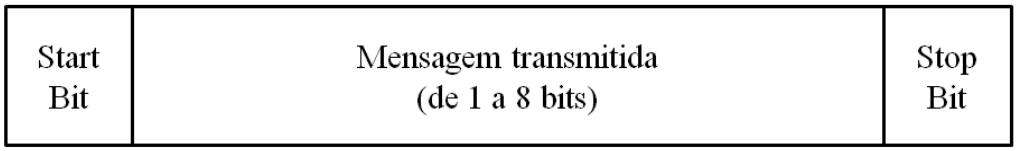


Figura 4.12 - Formato das mensagens na comunicação USART

Na figura 4.13 é possível visualizar o diagrama de blocos associado ao envio na comunicação USART.

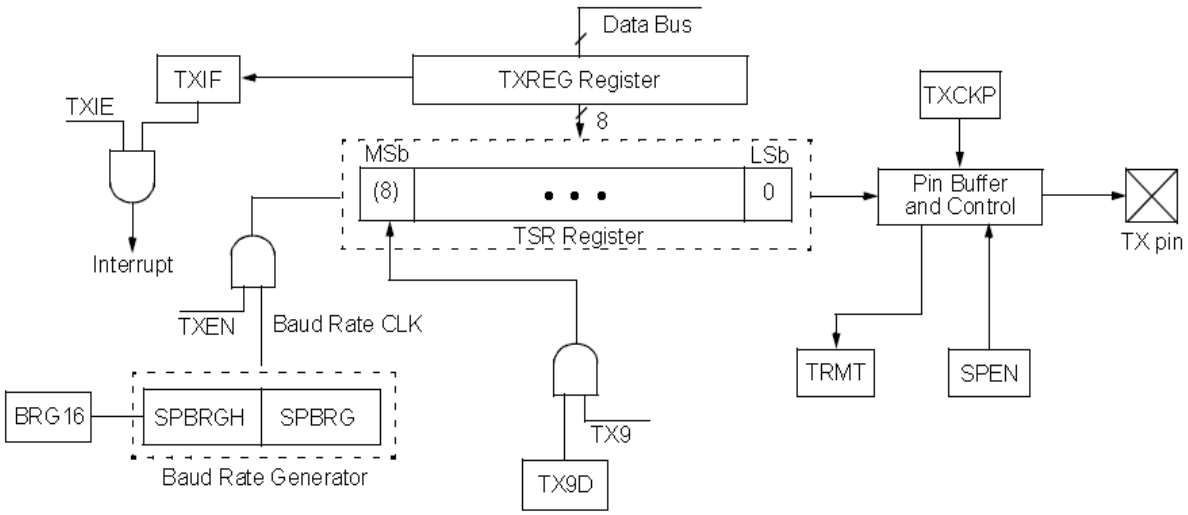


Figura 4.13 - Diagrama de blocos correspondente ao envio de mensagens

O registo TSR (*Transmit Shift Register*), como o próprio nome indica, é o registo associado ao conteúdo da mensagem a ser enviada. Este registo é carregado com informação proveniente do registo TXREG. Este carregamento é executado assim que o bit de stop da mensagem anterior é transmitido.

Assim que o conteúdo de informação é transferido do registo TXREG para o registo TSR, o bit TXIF é activado. Desta activação resulta um pedido de interrupção que irá permitir o envio da mensagem.

Enquanto o bit TXIF indica o estado do registo TXREG, um outro bit, TRMT, indica o estado do registo TSR. Este último bit é unicamente de leitura e não possui nenhuma lógica de interrupção, sendo utilizado unicamente para determinar se o registo TSR está vazio ou contem informação.

Na figura 4.14 está representado o diagrama de blocos correspondente à recepção de uma mensagem na comunicação USART.

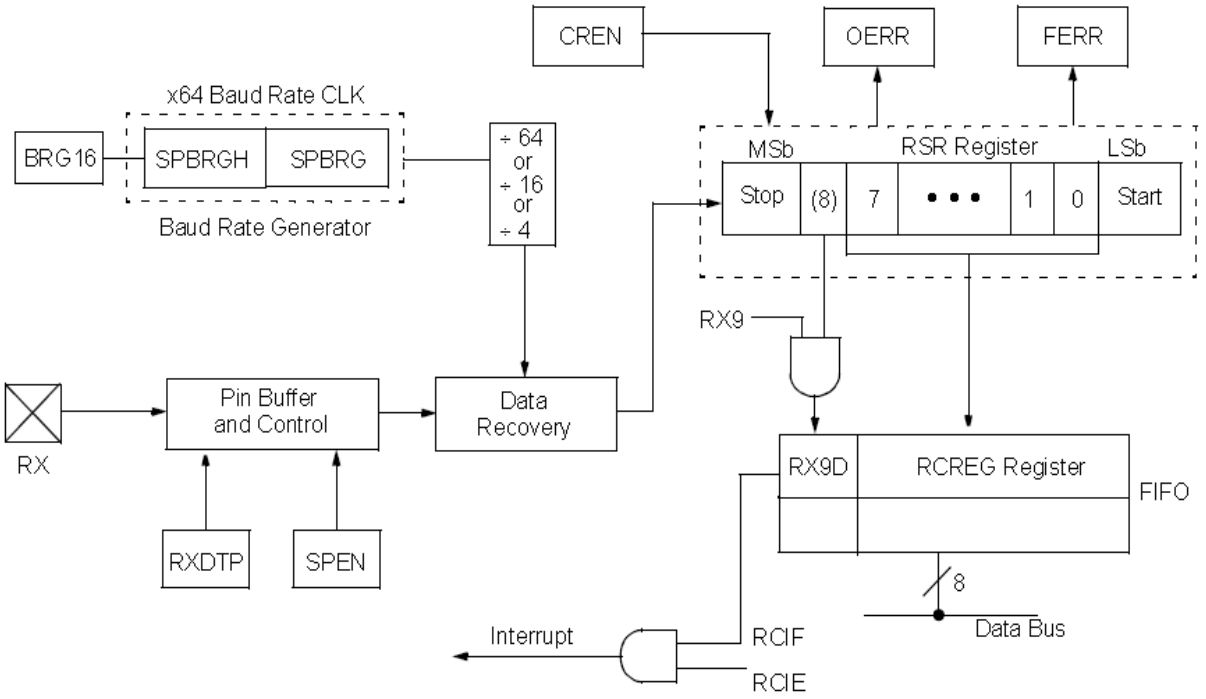


Figura 4.14 - Diagrama de blocos correspondente à recepção de mensagens

Na figura 4.14 é possível visualizar que no caso da recepção de uma mensagem, a interrupção é gerada assim que o registo RCREG é carregado com informação proveniente do registo RSR.

4.4 Estrutura da programação

Os protocolos utilizados na transmissão de todas as mensagens de comando e monitorização do sistema de actuação são suportados, para além da estrutura física desenvolvida, por um elevado número de linhas de código programadas nos vários microcontroladores e na interface computacional. O fluxograma da comunicação realizada entre os vários dispositivos da rede é apresentado de forma muito simplificada na figura 4.15.

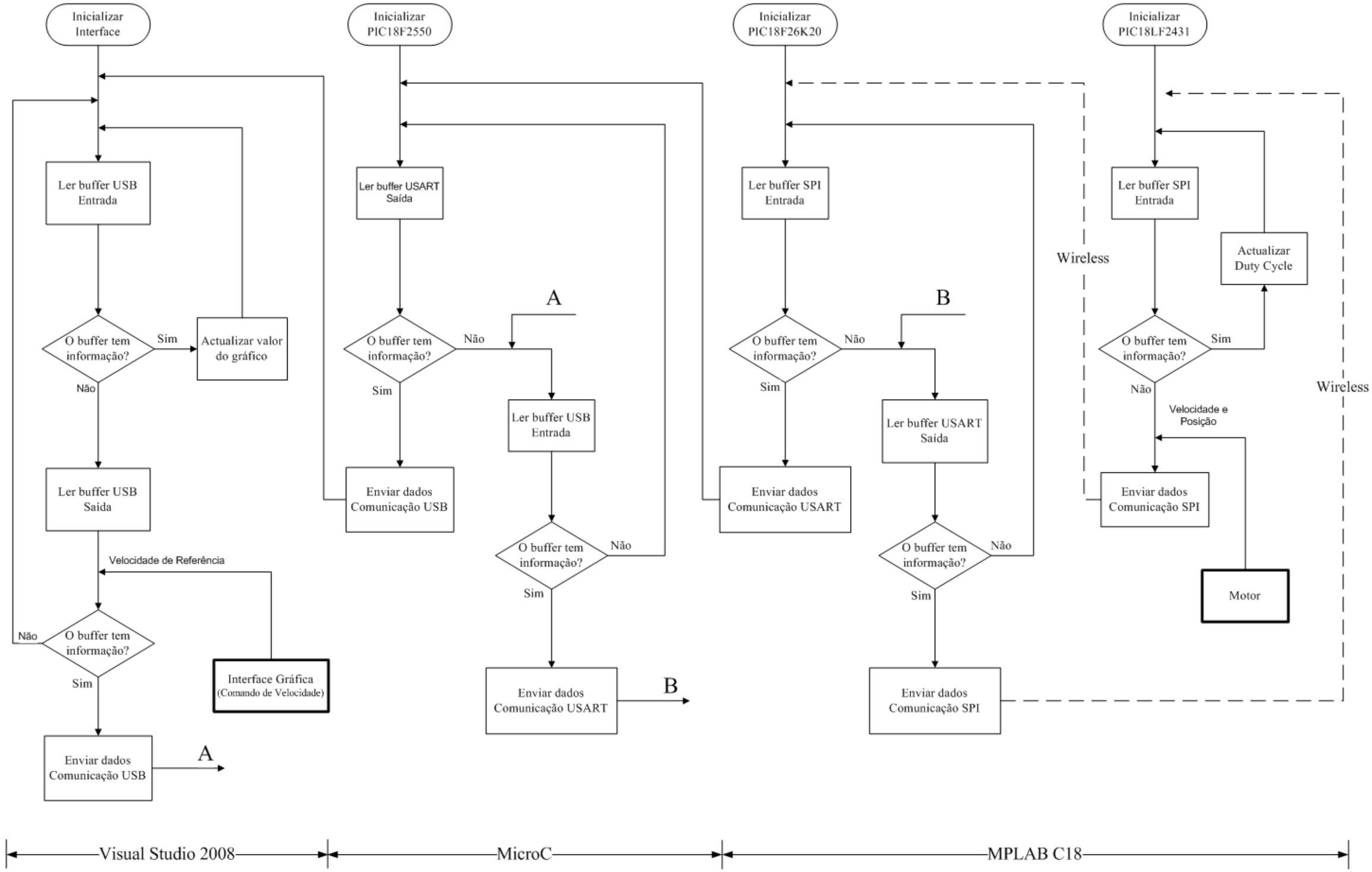


Figura 4.15 - Fluxograma simplificado da estrutura de programação

Toda a programação desenvolve-se com base numa estrutura de eventos, onde, cada evento, tem como origem um pedido de interrupção. Ao longo do caminho percorrido por cada mensagem são gerados vários pedidos de interrupção, possuindo cada um deles um nível diferente de prioridade. Estes vários níveis de prioridade permitem aos diferentes microcontroladores e à interface computacional obter uma ordem de resposta quando surgem duas ou mais solicitações em simultâneo.

Na figura 4.15 é também possível visualizar a linguagem de programação correspondente a cada conjunto de etapas da programação.

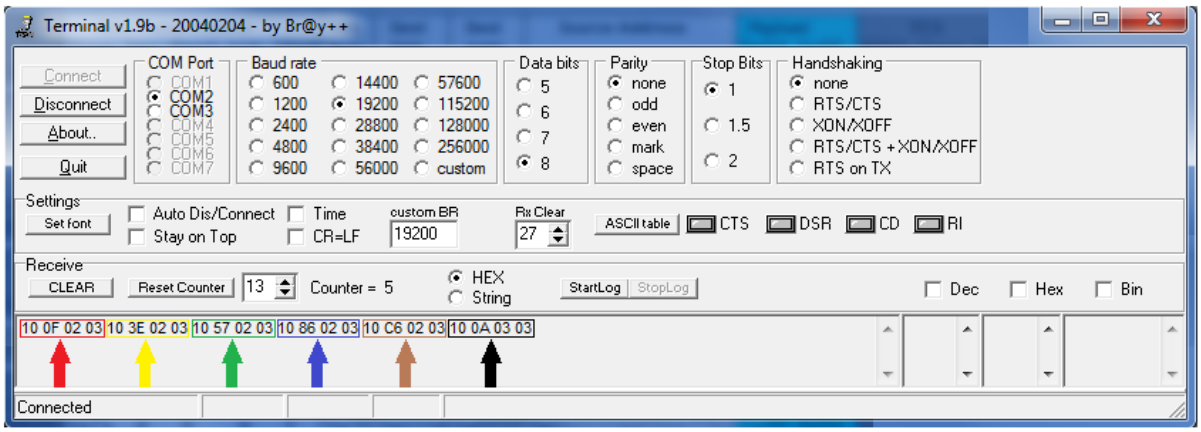
No Anexo A é possível visualizar de forma complementar o código realizado para a programação dos microcontroladores PIC18F26k20 e PIC18LF2431, por sua vez no Anexo B é apresentado o código associado à programação do microcontrolador o PIC18F2550. A programação correspondente à interface computacional é visível também de forma simplificada no Anexo C.

4.5 Resultados experimentais

Após verificar que os dados transmitidos na comunicação Wireless correspondiam ao previsto, surgiu a necessidade de verificar se esse mesmo conteúdo era transmitido de forma correcta pela comunicação USART. Deste modo foi utilizado um software, dedicado à monitorização de comunicações série, onde foi possível visualizar o conteúdo transmitido pela comunicação USART.

Observando a figura 4.16 é possível verificar que os dados enviados para o motor são correctamente transmitidos na comunicação USART e correctamente transferidos pela rede sem fios.

Para tornar mais perceptível a correspondência entre as duas comunicações, os dados homólogos foram assinalados com a mesma cor. Os pacotes captados na comunicação Wireless não assinalados dizem respeito a pacotes transmitidos no sentido contrário ao analisado neste teste.



ZENA(TM) Packet Sniffer - MiWi(TM) P2P Protocol

Frame	Time(us)	Len	MAC Frame Control					Seq	Dest	Dest	Source Address	Payload	FCS		
			Type	Sec	Pend	ACK	IPAN	Num	PAN	Addr		0x59 0xFF	RSSI	Corr	CRC
00020	+196944 =4122336	21	DATA	N	N	N	Y	0x7A	0x1234	0xFFFF	0x1122334455667703	0xFF 0x7B	-23	0x6A	OK
00021	+11856 =4134192	21	DATA	N	N	N	Y	0x03	0x1234	0xFFFF	0x1122334455667702	0x10 0x0F 0x02 0x03	-01	0x6B	OK
00022	+189664 =4323856	21	DATA	N	N	N	Y	0x7B	0x1234	0xFFFF	0x1122334455667703	0x59 0xFF 0xFF 0x7B	-22	0x6B	OK
00023	+11808 =4335664	21	DATA	N	N	N	Y	0x04	0x1234	0xFFFF	0x1122334455667702	0x10 0x3E 0x02 0x03	-02	0x6A	OK
00024	+191152 =4526816	21	DATA	N	N	N	Y	0x7C	0x1234	0xFFFF	0x1122334455667703	0x59 0xFF 0xCD 0x7B	-23	0x69	OK
00025	+11072 =4537888	21	DATA	N	N	N	Y	0x05	0x1234	0xFFFF	0x1122334455667702	0x10 0x57 0x02 0x03	-02	0x69	OK
00026	+217296 =4755184	21	DATA	N	N	N	Y	0x7D	0x1234	0xFFFF	0x1122334455667703	0x4C 0xFC 0x36 0x60	-22	0x6A	OK
00027	+11152 =4766336	21	DATA	N	N	N	Y	0x06	0x1234	0xFFFF	0x1122334455667702	0x10 0x86 0x02 0x03	-02	0x6B	OK
00028	+227040 =4993376	21	DATA	N	N	N	Y	0x7E	0x1234	0xFFFF	0x1122334455667703	0x1F 0xF0 0x61 0x5F	-22	0x69	OK
00029	+11120 =5004496	21	DATA	N	N	N	Y	0x07	0x1234	0xFFFF	0x1122334455667702	0x10 0xC6 0x02 0x03	-01	0x6B	OK
00030	+227392 =5231888	21	DATA	N	N	N	Y	0x7F	0x1234	0xFFFF	0x1122334455667703	0x11 0xD9 0x8D 0x94	-22	0x6B	OK
00031	+11184 =5243072	21	DATA	N	N	N	Y	0x08	0x1234	0xFFFF	0x1122334455667702	0x10 0x0A 0x03 0x03	-01	0x6B	OK
00032	+227600 =5470672	21	DATA	N	N	N	Y	0x80	0x1234	0xFFFF	0x1122334455667703	0x0B 0xB6 0x68 0xED	-22	0x6A	OK

Figura 4.16 - Visualização dos dados enviados para o motor na comunicação USART e Wireless

Os dados assinalados na figura 4.16 foram previamente enviados pela plataforma computacional. Estas mensagens contêm informação associada ao valor do *duty cycle* e, consequentemente ao valor da velocidade do motor pretendida. Desta forma está também verificada a transmissão de dados entre o microcontrolador e a interface computacional realizada pela comunicação USB. Todas as outras mensagens contêm informação enviada no

sentido oposto, ou seja, informação utilizada na monitorização do motor com destino à plataforma computacional.

Na figura 4.17 estão representados os dados enviados na comunicação Wireless e USART no sentido oposto, isto é, os dados de monitorização das acções do motor.

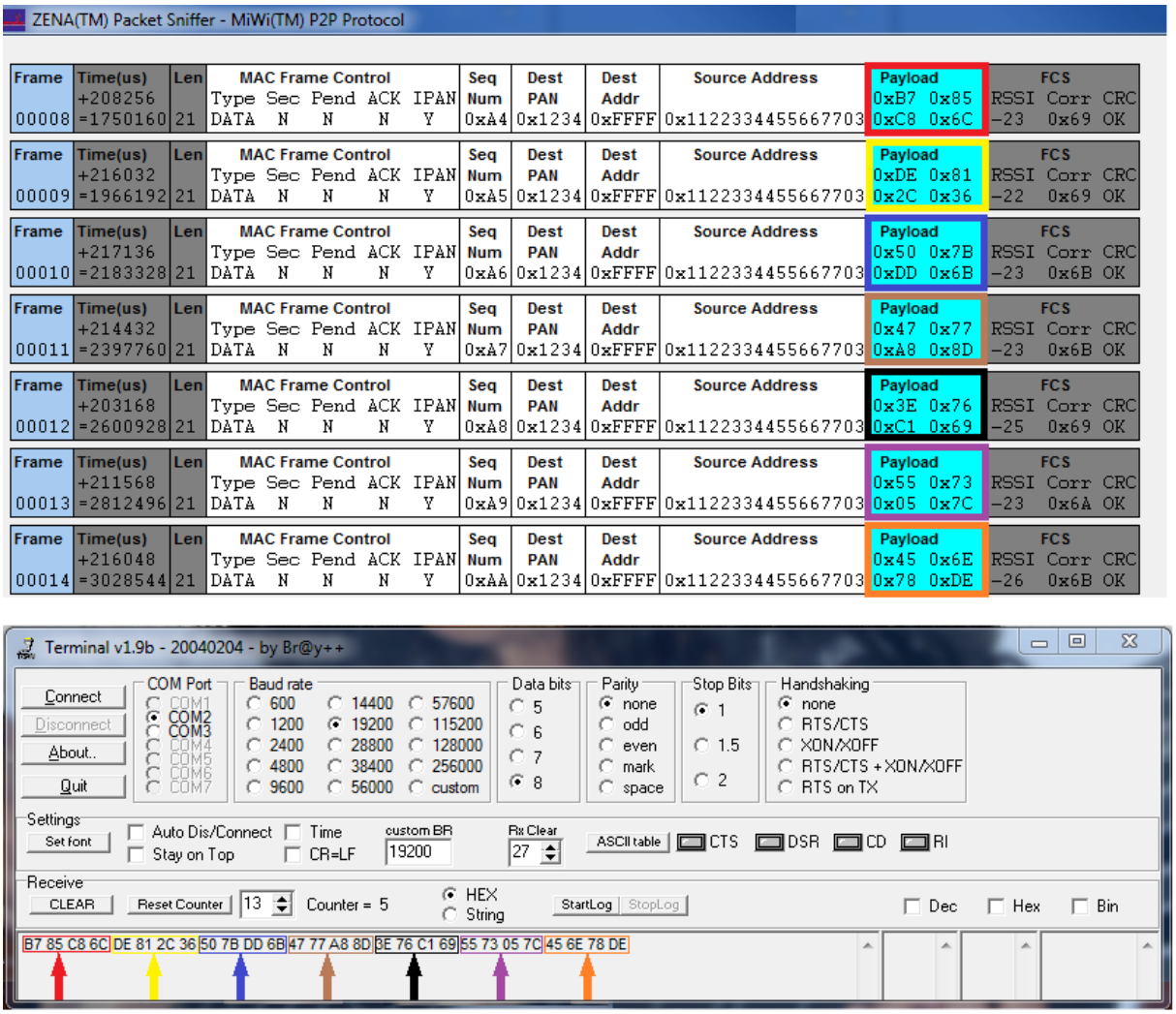


Figura 4.17 - Visualização dos dados provenientes do estado do motor enviados na comunicação Wireless e USART

Através da figura 4.17 é também possível verificar que os dados provenientes das acções do motor são transmitidos de forma correcta.

Após a monitorização da comunicação série e Wireless foi possível concluir que a transmissão dos dados de comando e de monitorização das acções do motor circulam por todo o sistema de comunicação de forma correcta e sem sofrerem alterações.

4.6 Conclusões

Durante a realização dos vários ensaios foram obtidos resultados considerados satisfatórios face aos objectivos propostos.

Foi possível obter um sistema capaz de executar uma comunicação bidireccional, desta forma o utilizador pode comandar e monitorizar o movimento do motor de forma simultânea.

O sistema criado possui uma velocidade de comunicação adequada para a aplicação, de modo a responder quase instantaneamente às ordens do utilizador.

Desta forma é seguro concluir que os vários objectivos propostos inicialmente para a comunicação USB e USART foram cumpridos.

Capitulo 5

Comando e Monitorização do Motor

Na indústria os motores DC são vulgarmente utilizados para accionamento de servomecanismos. Estes motores podem ser comandados de diversas formas. Neste projecto foi utilizada a tecnologia PWM – *Pulse Width Modulation*, em português *Modelação por Largura de Pulso*, para executar esta função.

O comando de um motor está normalmente associado à monitorização do seu movimento. Foi então utilizado um encoder acoplado ao eixo de rotação do motor de forma a tornar possível a aquisição da sua velocidade de rotação.

Neste capítulo são fundamentalmente abordadas as técnicas utilizadas neste projecto para realizar o comando do motor e a monitorização da sua velocidade.

5.1 PWM - Pulse Width Modulation

O PWM é uma poderosa técnica usada para controlar um circuito analógico utilizando as saídas digitais de um microcontrolador. Um dos motivos que leva à utilização desta técnica é a redução significativa da temperatura nos elementos condutores de corrente eléctrica associados à alimentação do sistema de actuação.

5.1.1 Princípio de funcionamento

O PWM envolve a geração de uma série de pulsos com uma frequência e período bem definidos. O *duty cycle* define a largura de cada pulso, isto é, define a percentagem do período

em que o sinal se encontra no estado activo. Se a largura do pulso ocupar metade do tempo total do período, a tensão de saída será metade do valor correspondente à tensão de entrada. Desta forma, é possível variar a tensão de saída variando o valor do *duty cycle*.

Na figura 5.1 é possível visualizar de forma gráfica os vários conceitos que esta técnica envolve.

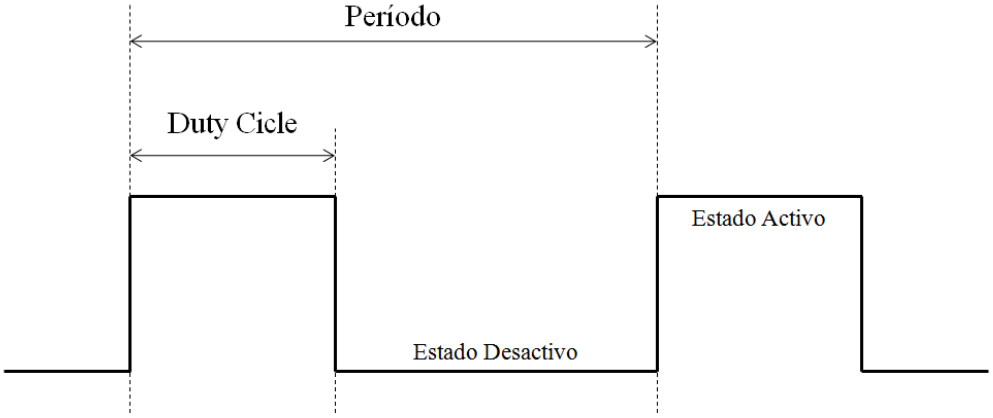


Figura 5.1 - Estrutura do sinal PWM

Quando se pretende comandar a velocidade de um motor DC utiliza-se normalmente não um, mas dois sinais PWM. Se estes sinais forem gerados de forma complementar possibilita o movimento do motor em diferentes sentidos e velocidades de rotação.

Na figura 5.2 estão representados os dois sinais complementares gerados pelo microcontrolador para o comando de um motor DC.

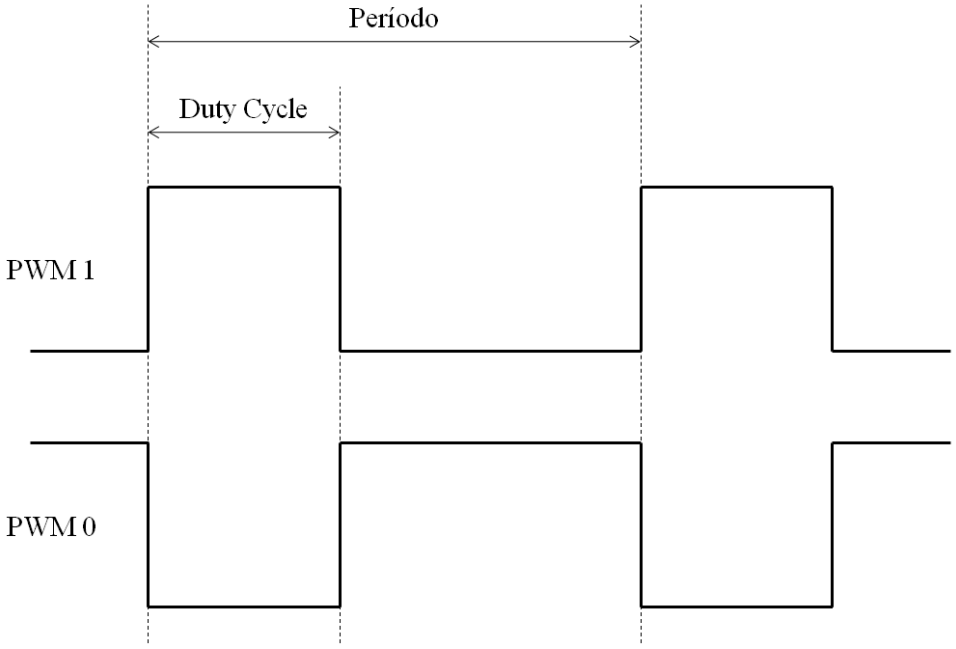


Figura 5.2 - PWM gerado em modo complementar

Por vezes a amplitude máxima do sinal PWM emitida pelo microcontrolador não é suficiente para alimentar o sistema de actuação, nesta situação é utilizada uma fonte de potência de modo a sustentar as necessidades potenciais do sistema. O microcontrolador PIC18LF2431, utilizado na geração do sinal PWM, fornece uma amplitude do sinal igual a 3,3 V, deste modo foi necessário utilizar a ponte H, referida no capítulo 2, para implementar o circuito de alimentação do motor. A conversão dos 3,3 V emitidos pelo microcontrolador para os 5 V requeridos pela ponte é realizada pelo dispositivo lógico HD74LSOOP.

Para que nenhum circuito integrado implementado na condução do sinal PWM entre num estado de indeterminação durante as transições ascendentes e descenderes dos sinais PWM e assumindo que nenhum sistema lógico consegue comutar de estado de forma instantânea, surge a necessidade de desfasar estes dois sinais. Para realizar este desfasamento é introduzido um tempo morto entre a transição realizada pelo sinal PWM 0 e a transição complementar realizada pelo sinal PWM 1. Desta forma, entre cada transição, existe sempre um curto intervalo de tempo em que os dois sinais se encontram inactivos.

Na figura 5.3 é visível a introdução do tempo morto entre os dois sinais PWM.

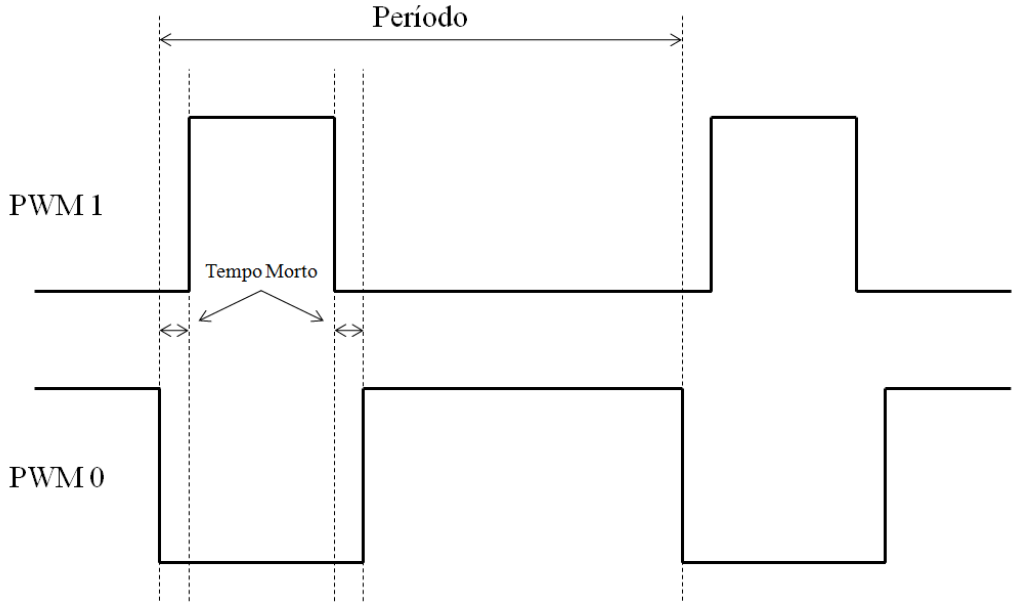


Figura 5.3 - Introdução do tempo morto entre os sinais PWM

5.1.2 Modelação dos sinais PWM

O módulo periférico de geração de sinais PWM, integrado no PIC18LF2431, facilita significativamente o processo de configuração e modelação destes sinais [8].

A geração dos sinais PWM é configurada através de 22 registos. Estes registos são utilizados para definir vários parâmetros, entre eles a base de tempo, o período, o *duty cycle* e o tempo morto.

Base de tempo PWM

A base de tempo PWM é responsável por impor a cadência de cálculo na geração dos sinais PWM. Ela é fornecida por um *timer* de 12 bits. Este *timer* pode ser fraccionado, utilizando funções de pré e *postscaler*, de modo a alterar a sua resolução. A base de tempo PWM é configurada através dos registos PTCON0 e PTCON1.

Esta base de tempo pode operar em quatro modos distintos seleccionados no registo PTCON0:

- *Free-Running*
- *Single-Shot*
- *Continuous Up/Down Count* (utilizado neste projecto)
- *Continuous Up/Down Count with interrupts for double updates*

Estes quatro modos produzem gráficos da base de tempos com duas formas distintas. O modo *Free-Running* e o modo *Single-Shot* produzem uma onda *Edge-Aligned*, por sua vez os dois modos de contagem *Up/Down* produzem uma onda *Center-Aligned*. Na figura 5.4 é possível observar estas duas formas de onda.

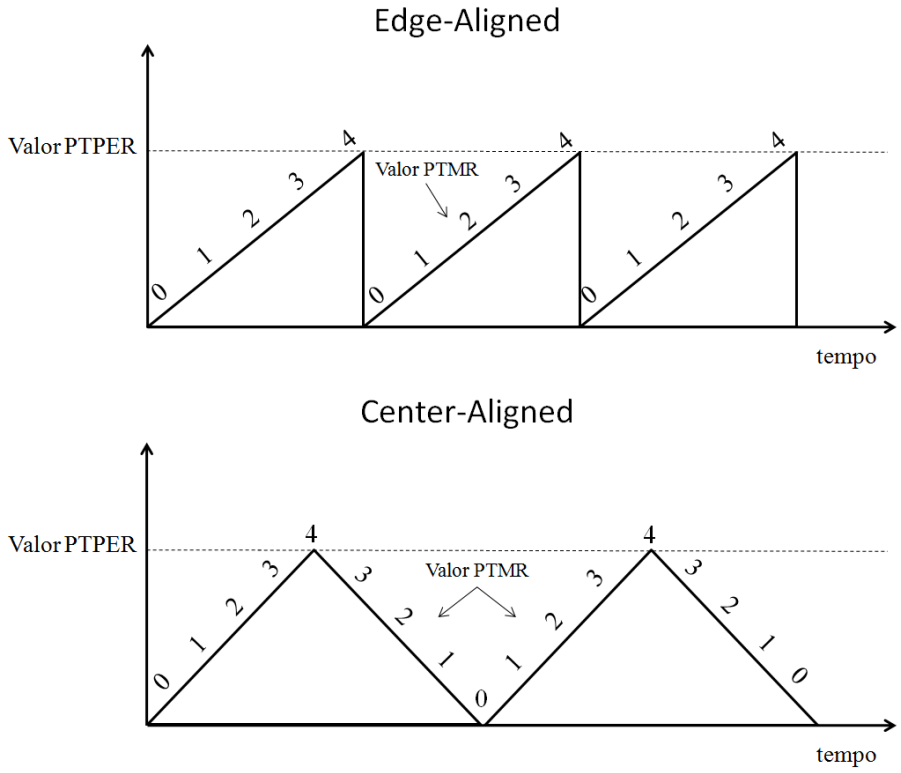


Figura 5.4 - Forma *Edge-Aligne* e forma *Certer-Aligned* de base de tempo

No modo ***Free-Running***, os registos PTMRL e PTMRH da base de tempo iniciam a sua contagem de forma ascendente até ao valor do período da base de tempo registada na variável PTPERL e PTPERH. Os registos PTMR são reiniciados no sinal seguinte enviado pelo *clock* iniciando novamente a sua contagem.

Quando a base de tempo PWM está configurada no modo *Free-Running*, é gerada uma interrupção sempre que o registo PTMR alcança o valor registado na variável PTPER. Utilizar uma razão diferente de 1:1 no valor do *postscale* reduz a frequência a que são geradas as interrupções.

No modo ***Single-Shot***, a base de tempo PWM inicia a contagem ascendente quando o bit PTEN, responsável pela sua activação, é activado. Quando o valor do registo PTMR coincide com o registo PTPER, o registo PTMR é reiniciado no sinal seguinte emitido pelo *clock*.

Quando a base de tempo PWM está configurada no modo *Single-Shot*, é gerada uma interrupção sempre que o registo PTMR assume o valor registado na variável PTPER. O valor da razão *postscaler* pode adquirir qualquer valor uma vez que não tem qualquer influência neste modo.

No modo ***Continuous Up/Down Count,*** implementado neste projecto, a base de tempo inicia a contagem ascendente até que o valor do registo PTMR coincida com o registo PTPER. No sinal seguinte emitido pelo *clock* a base de tempo retoma a contagem mas de forma

descendente. O bit PTDIR no registo PTCON1 é somente de leitura e indica o sentido de contagem.

Quando a base de tempo PWM está configurada para operar no modo *Continuous Up/Down Count*, a interrupção é gerada sempre que o valor PTMR é igual a zero e a base de tempo inicia a sua contagem ascendente. O valor da razão *postscale* pode ser utilizado neste modo para reduzir a frequência a que são geradas as interrupções.

No modo ***Continuous Up/Down Count with interrupts for double updates,*** a interrupção é gerada sempre que o registo PTMR adquire o valor zero e sempre que esta variável se iguala ao registo PTPER.

Período do sinal PWM

O período do PWM é definido pelo registo PTPER. Este registo é constituído por 12 bits distribuídos por dois bytes (PTPERL e PTPERH). O PTPER é então um buffer de duplo registo utilizado para definir o período da base de tempo PWM.

A fórmula de cálculo do período é realizada com base no registo PTPER no *prescale* atribuído à base de tempo e na frequência do oscilador utilizada. Nas expressões 1 e 2 estão representadas as fórmulas de cálculo do período para o modo *Free-Running* e *Continuous Up/Down Count* respectivamente.

$$T_{PWM} = \frac{(PTPER+1)\times PTMRPS}{FOSC/4} \quad (1)$$

$$T_{PWM} = \frac{(2\times PTPER)\times PTMRPS}{FOSC/4} \quad (2)$$

Deste modo, a frequência do PWM pode ser calculada da seguinte forma:

$$f_{PWM} = \frac{1}{T_{PWM}} \quad (3)$$

A resolução PWM máxima (em bits) pode ser calculada relacionando a frequência do oscilador com a frequência PWM utilizando a expressão 4:

$$Resolução = \frac{\log\left(\frac{FOSC}{f_{PWM}}\right)}{\log(2)} \quad (4)$$

Aplicando nas fórmulas de calculo 2 e 4 os valores utilizados neste projecto é possível obter o seguinte valor teórico para a frequência e resolução PWM respectivamente:

$$f_{PWM} = 1/\left[\frac{(2\times 256)\times 1}{8000000/4}\right] = 3900\ Hz \quad (5)$$

$$Resolução = \frac{\log\left(\frac{8000000}{3900}\right)}{\log(2)} = 11\ bits \quad (6)$$

O conteúdo do registo PTPER é carregado na base de tempo em momentos bem definidos. No modo *Free-Running* e *Single-Shot* o período é actualizado no momento em que o registo PTMR é reiniciado. Nos dois modos *Continuous Up/Down Count* a base de tempo carrega o valor do período quando o registo PTMR assume o valor zero. Na figura 5.5 está representado de forma gráfica o momento em que o período é actualizado em ambas as formas de onda.

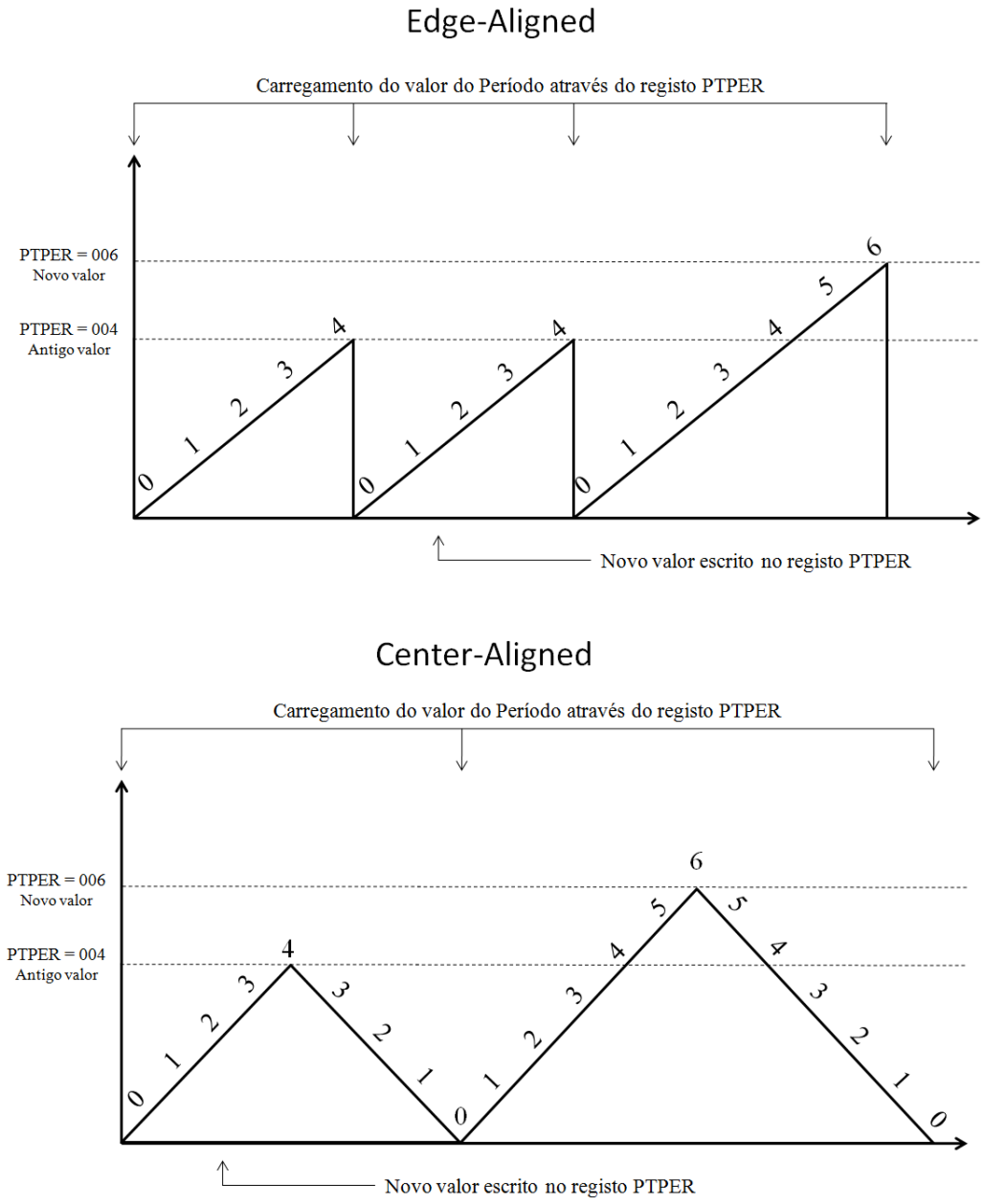


Figura 5.5 - Momento de actualização do periodo nas duas formas de base de tempo

Duty Cycle do sinal PWM

O *duty Cycle* do sinal PWM é definido pelo registo PDCx, onde x representa o número do par complementar de sinais PWM. Este registo pode conter até 14 bits, dependendo da resolução conseguida, distribuídos por dois bytes, (PDCxL e PDCxH).

Existe ainda uma variável interna à qual o programador não possui acesso. Esta variável adquire a comparação com o valor real utilizado no período actual.

À semelhança do registo PTPER também o PDCx é carregado para a base de tempo em momentos bem determinados.

Para uma base de tempo com forma *Edge-Aligned* o novo valor do *duty cycle* é actualizado sempre que a variável PTMR possui o valor do registo PTPER. Na figura 5.6 é possível visualizar de forma gráfica a actualização deste registo na base de tempo PWM.

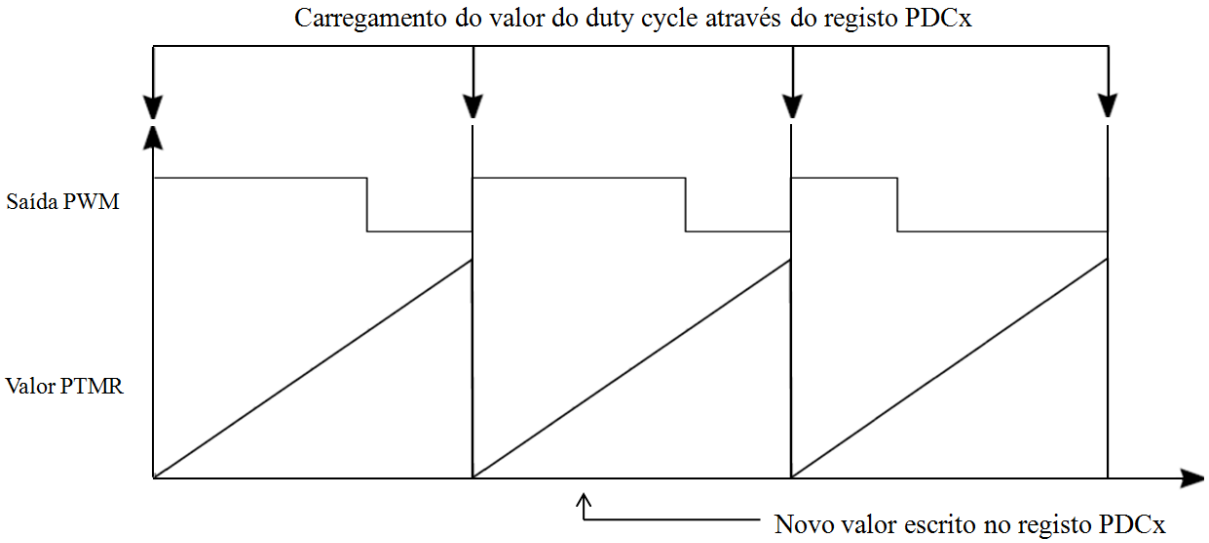


Figura 5.6 - Actualização do *duty cycle* na forma Edge-Aligned

Quando a base de tempo está configurada no modo *Continuous Up/Down Count*, o valor do novo *duty cycle* imposto é actualizado quando o valor do registo PTMR é nulo e a base de tempo inicia a sua contagem ascendente. O conteúdo do registo PDCx é carregado para a variável interna no momento em que a base de tempo PWM é desactivada, isto é, o bit PTEN assume o valor zero. Neste modo, todo o tempo de duração de um período está disponível para o cálculo e carregamento do novo *duty cycle* antes que as alterações sejam implementadas. Na figura 5.7 é apresentado de forma gráfica a actualização do valor do registo PDCx na base de tempo para este modo.

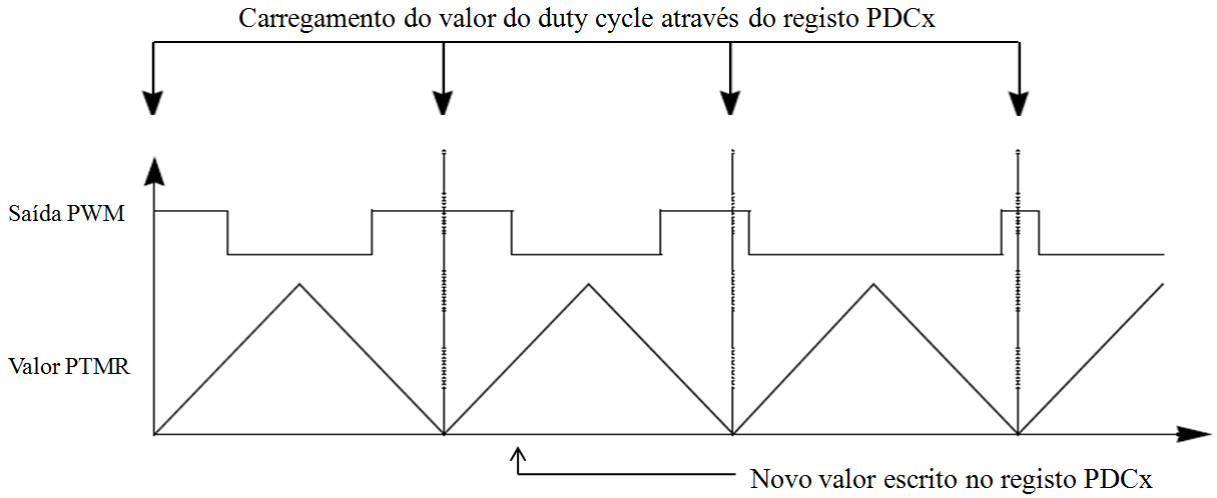


Figura 5.7 - Actualização do duty cycle no modo Continuous Up/Down Count

Para uma base de tempo configurada no modo *Continuous Up/Down Count with interrupts for double updates*, o valor do novo *duty cycle* é actualizado quando o valor do registo PTMR assume o valor zero. Neste mesmo instante o valor do conteúdo do registo PDCx é carregado para a variável interna. Neste modo apenas metade do tempo de duração de um período está disponível para o cálculo e carregamento do novo *duty cycle* antes que as alterações sejam realizadas. A figura 5.8 mostra o gráfico da actualização do novo valor do *duty cycle* na base de tempo PWM para este modo.

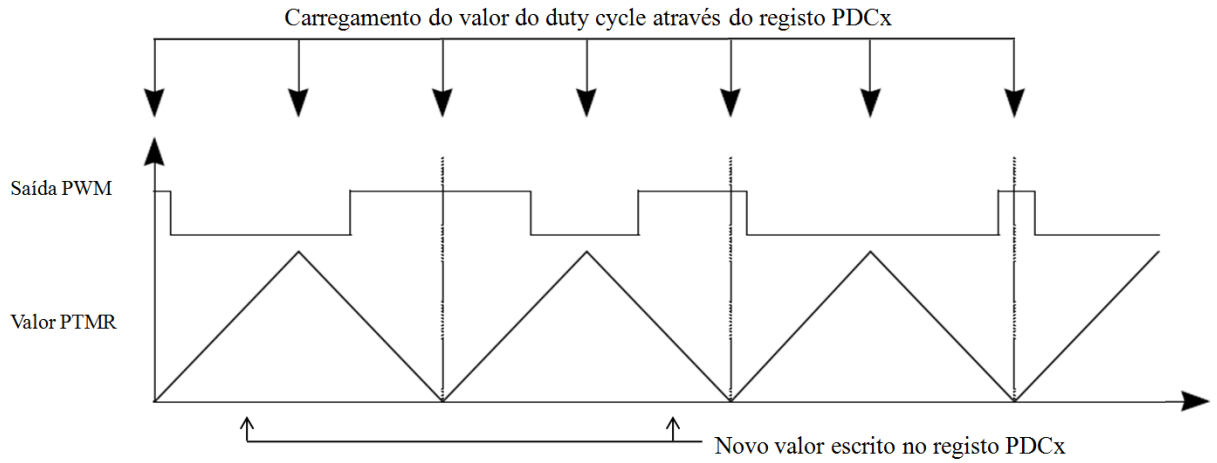


Figura 5.8 - Actualização do duty cycle no modo Continuous Up/Down Count with interrupts for double updates

5.2 Motion Feedback Module

O *Motion Feedback Module* - MFM é um módulo periférico integrado no PIC18LF2431 especialmente projectado para aplicações onde é necessária a monitorização de movimentos [8]. Este módulo utilizado juntamente com o de geração de sinais PWM fornece uma variedade de soluções para o controlo de uma vasta gama de motores eléctricos.

O MFM é composto essencialmente por dois sub-módulos distintos:

- IC – *Input Capture*
- QEI – *Quadrature Encoder Interface*

Estes dois sub-módulos, embora semelhantes, apresentam funções diferentes. Na tabela 5.1 são apresentadas algumas dessas funções.

Sub-Módulo	Funções
IC – Input Capture	Captura de largura de pulsos
	Medição de períodos
	Captura de mudanças
	Conversão A/D
QEI – Quadrature Encoder Interface	Medição de posição
	Medição de velocidade
	Identificação do sentido de rotação

Tabela 5.1 - Funções dos sub-módulos IC e QEI

Como um dos principais objectivos deste trabalho é a monitorização da velocidade de rotação do motor DC, a atenção recai essencialmente para o segundo sub-módulo apresentado na tabela 5.1.

Estes sub-módulos são mutuamente exclusivos, isto significa que apenas um deles pode estar em funcionamento. No caso da configuração dos registos habilitar os dois, o sub-módulo QEI prevalece, inibindo o *Input Capture*. A configuração dos sub-módulos QEI e IC é realizada pelos registos QEICON e CAPCON respectivamente.

5.2.1 QEI – Quadrature Encoder Interface

O sub-módulo QEI descodifica a informação proveniente de um sensor de movimento do tipo encoder. Esta ferramenta pode ser aplicada em qualquer sistema que utilize um encoder para o feedback de movimento. A tabela 5.2 enumera algumas características do *Quadrature Encoder Interface*.

Quadrature Encoder Interface	
Canais de entrada	
Dois para as fases do sinal – QEA e QEB	Uma entrada para o sinal índex - INDX
Medição de posição	
Identificação do sentido de rotação	Dois modos de actualização – x2 e x4
Medição de velocidade	
Postscale programável para altas velocidades	

Tabela 5.2 - Características do *Quadrature Encoder Interface*

Este sub-módulo é composto essencialmente por três componentes: controlo lógico QEI, contador da posição e *postscale* da velocidade. Na figura 5.9 é possível visualizar um diagrama de blocos simplificado da composição do sub-módulo QEI.

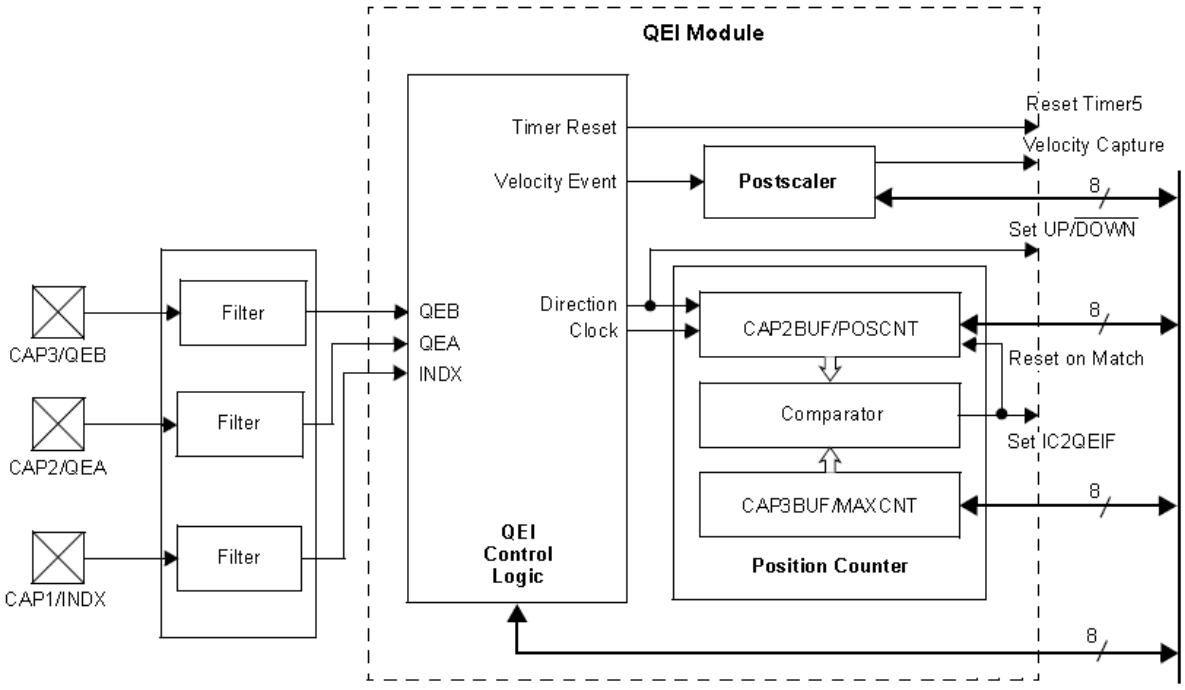


Figura 5.9 - Diagrama de blocos do sub-modulo QEI

O controlo lógico QEI detecta as transições existentes nas duas fases QEA e QEB que compõem o sinal. Estas transições geram pulsos contados internamente que são enviados para o contador da posição. Além disso, o controlo lógico recebe as transições do sinal índex (INDX) e gera também um sinal indicativo do sentido de rotação do motor.

O contador da posição actua como integrador para monitorizar a distância percorrida. O registo responsável pela contagem da posição (POSCNT) é incrementado em cada transição detectada pelo canal QEA ou pelos canais QEA e QEB, dependendo do modo de operação configurada. Este registo é reiniciado quando o seu valor ultrapassa o valor máximo admissível, isto é, iguala o valor definido no registo MAXCNT, ou quando é detectado um sinal na entrada INDX.

O *postscale* da velocidade é utilizado para diminuir o número de pulsos que são incrementados no contador da velocidade. Essencialmente este *postscale* é a razão entre o número de pulsos utilizados na medição da velocidade e o número de pulsos reais adquiridos.

5.2.2 Modos QEI

A resolução da medição da posição depende fundamentalmente de quantas vezes o registo POSCNT é incrementado para a mesma amplitude rotacional. Existem dois modos para actualizar este registo e medir a posição: QEI x2 e QEI x4.

No modo de actualização QEI x2 o controlo lógico QEI detecta unicamente as transições ascendentes e descendentes do sinal QEA. O modo de actualização QEI x4 prevê uma resolução mais fina da posição do rotor, uma vez que conta quer transições ascendeste que transições descendentes de ambos os sinais QEA e QEB.

Na tabela 5.3 são enumerados os casos possíveis relacionando o modo como o registo POSCNT é incrementado e a forma como é realizado o seu reset.

Modo / Reset	Descrição
QEI x2 / INDX	O sinal do encoder é lido sem quadratura (apenas o sinal QEA) / O reset é realizado pelo sinal INDX
QEI x2 / MAXCNT	O sinal do encoder é lido sem quadratura (apenas o sinal QEA) / O reset é realizado pela igualdade com o registo MAXCNT
QEI x4 / INDX	O sinal do encoder é lido em quadratura (sinal QEA e sinal QEB) / O reset é realizado pelo sinal INDX
QEI x4 / MAXCNT	O sinal do encoder é lido em quadratura (sinal QEA e sinal QEB) / O reset é realizado pela igualdade com o registo MAXCNT

Tabela 5.3 - Modos QEI

Na figura 5.10 e 5.11 é possível visualizar o número de transições por período nos sinais QEA e QEB nos dois modos de actualização do registo POSCNT.

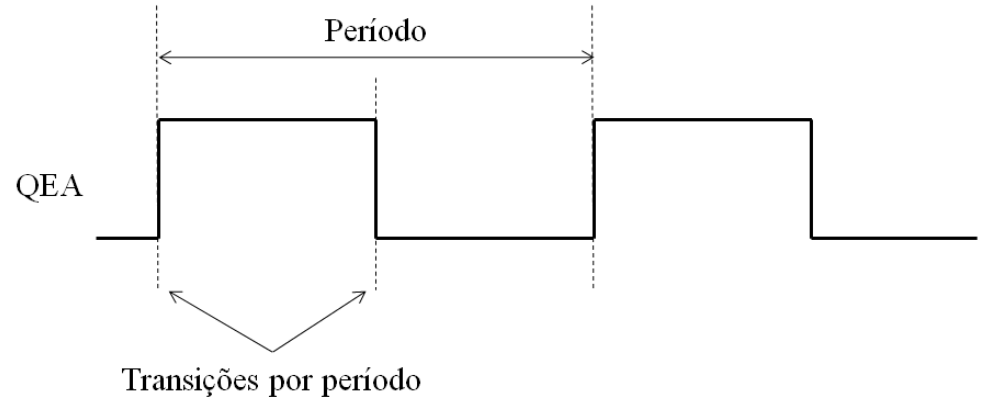


Figura 5.10 - Transições do sinal QEA no modo QEI x2

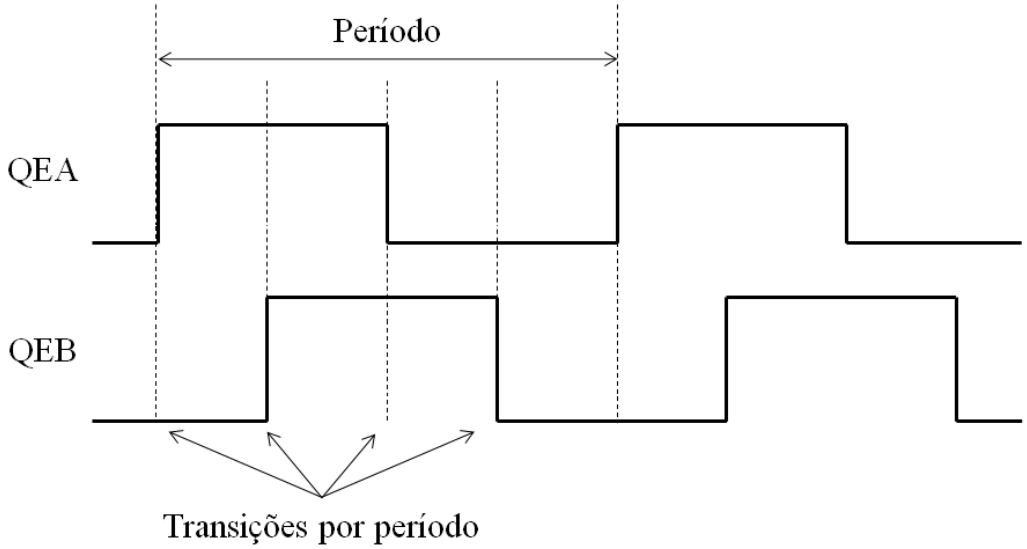


Figura 5.11 - Transições dos sinais QEA e QEB no modo QEI x4

5.2.3 Medição da velocidade

O *Input Capture Channel* 1 (IC1) é utilizado para capturar o valor de contagem do *Timer 5*, por sua vez o *Timer 5* é um *clock*/contador associado à medição da velocidade.

A geração de pulsos em conjunto com o IC1, a operar de forma síncrona com o *Timer 5*, fornece um método de alta precisão para medição de altas e baixas velocidades de um motor. O modo de medição de velocidade é activado quando o bit VELM do registo QEICON é definido com o valor zero e o modo QEI é configurado no modo QEI x2 ou QEI x4.

Quando se pretende medir velocidades altas, é utilizado um divisor de pulsos. Este divisor diminui a frequência de pulsos vista pelo contador, isto é, divide o número de pulsos gerados pelo sinal proveniente do encoder por um número inteiro. Deste modo, para o mesmo

intervalo de tempo, medido pelo *Timer 5*, o contador permite contar um maior número de pulsos reais gerados pelo encoder.

Na figura 5.12 é demonstrado o diagrama temporal da medição da velocidade.

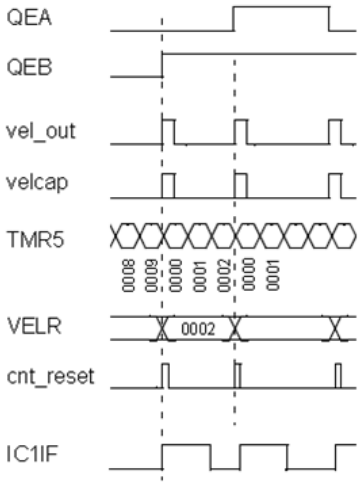


Figura 5.12 - Diagrama temporal da medição da velocidade

Na figura 5.12, o sinal *vel_out* representa a conjugação da quadratura entre o sinal QEA e QEB. O sinal *velcap* representa o sinal *vel_out* afectado da razão *postscale* (no caso representado esta razão é unitária). O sinal IC1IF representa as interrupções geradas para actualização do buffer VELR.

Observando a figura 5.12 é possível concluir que o valor do *Timer 5* é reiniciado sempre que surge um impulso no sinal *velcap*. O buffer VELR é carregado com o valor de pulsos contados pelo *Timer 5* em cada período do sinal *velcap*.

A velocidade de rotação do motor, monitorizada na interface computacional, é calculada utilizando a seguinte expressão:

$$V_{Rotação} = \frac{Frequência\ de\ operação/4}{PPR \times Racio\ de\ actualização\ da\ velocidade \times Racio\ de\ redução\ de\ pulsos \times Timer\ 5\ Prescale \times Valor} \quad (7)$$

Onde a *Frequência de operação* corresponde à frequência do oscilador programado no microcontrolador PIC18LF2431 afectada pelo *prescale* associado. A variável *PPR* representa o número de impulsos, por rotação e por canal, emitidos pelo encoder acoplado ao veio do motor. O *rácio de actualização da velocidade* está associado à quadratura dos sinais enviados pelo encoder e consequentemente aos modos de leitura QEI, por sua vez, o *rácio de redução de pulsos* corresponde ao factor multiplicativo necessário para a medição de velocidades elevadas. A variável *Timer 5 Prescale* corresponde ao valor *Prescale* afectante do *Timer 5*.

Por último a variável *valor* diz respeito ao valor enviado pelo microcontrolador PIC18LF2431 proveniente do sinal emitido pelo encoder e armazenado no buffer VELR.

Substituindo pelos valores correspondentes a cada variável obtém-se o seguinte resultado:

$$V_{Rotação} = \frac{8000000 \times 64/4}{400 \times 4 \times 1 \times 1 \times Valor} \times 60\ [rpm] \quad (8)$$

5.3 Resultados experimentais e conclusões

Com o auxílio de um osciloscópio foi possível visualizar e analisar o sinal PWM, gerado pelo microcontrolador e o sinal em quadratura enviado pelo encoder. Através deste instrumento de medida tornou-se mais fácil comprovar todos os fundamentos teóricos provenientes da geração destes dois sinais.

Na figura 5.13 é possível visualizar os sinais PWM gerados de forma complementar para diferentes valores do *duty cycle*, de forma a comandar a rotação do motor para diferentes estados.

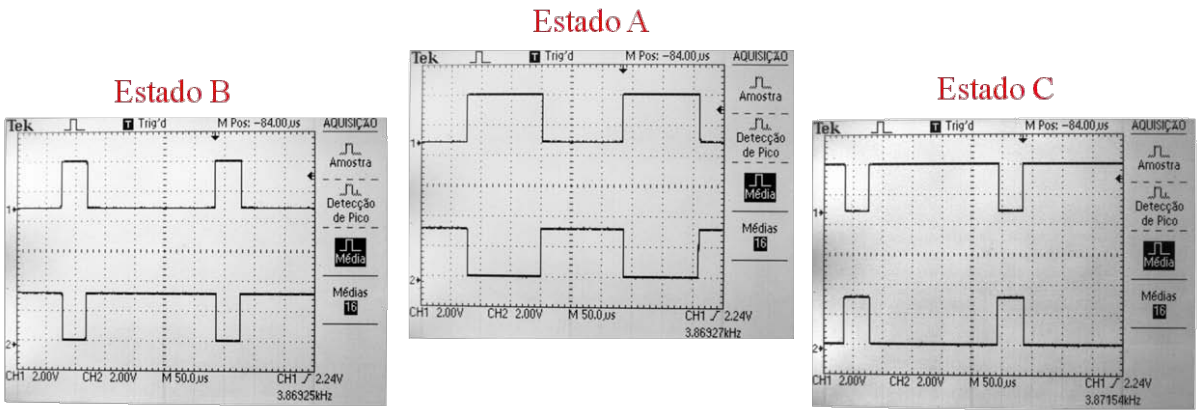


Figura 5.13 - Sinais PWM gerados com diferentes valores de duty cycle

No estado A, o valor do *duty cycle* é aproximadamente metade do seu valor máximo. Nestas condições o motor encontra-se parado, gerando ainda alguma oposição ao movimento provocado por perturbações externas.

No estado B, o valor do *duty cycle* é inferior a metade do seu valor máximo. Esta solicitação provoca a rotação do motor no sentido directo.

Quando o valor do *duty cycle* é superior a metade do seu valor máximo, estado C, a rotação do motor é efectuada no sentido reverso.

Observando mais atentamente a zona de transição dos sinais PWM, representada na figura 5.14, é possível visualizar para diferentes escalas de tempo, a existência do tempo morto entre os dois sinais complementares.

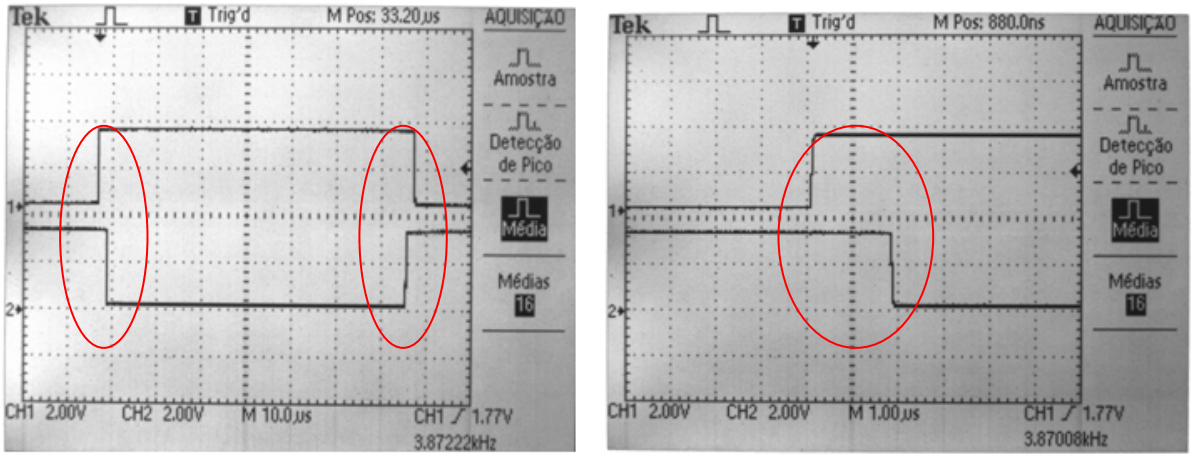


Figura 5.14 - Tempo morto entre os sinais PWM

Na figura 5.15 está representado o sinal composto enviado pelo encoder, onde, como seria de esperar, é visível a sua quadratura.

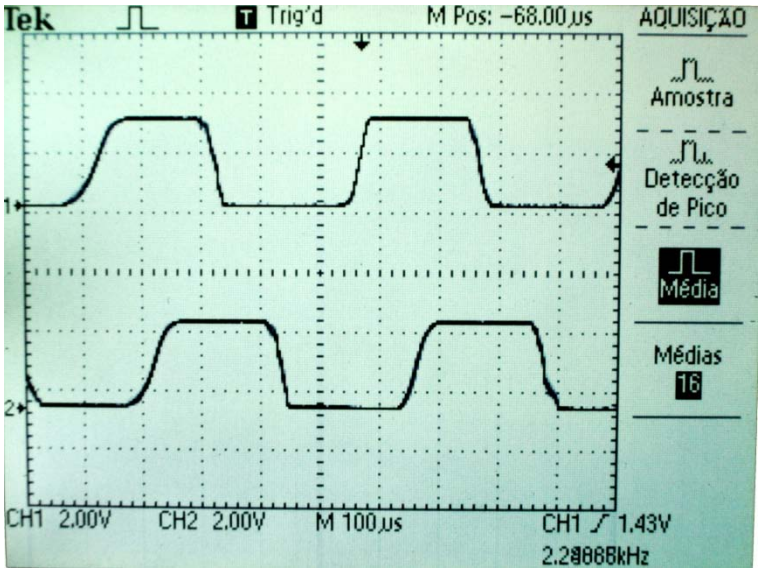


Figura 5.15 - Quadratura do sinal do encoder

Analisando então os vários resultados capturados pelo osciloscópio, conclui-se que através da utilização dos fundamentos teóricos associados à geração dos sinais PWM e dos sinais derivados do encoder, é possível obter no comando do motor resultados muito próximos dos esperados.

Os ensaios experimentais revelaram ainda a importância do tempo de *duty cycle* no comando da velocidade do motor, pois alterando unicamente esta variável é visível a variação da sua velocidade de rotação.

5.4 Conclusões

Durante a execução deste trabalho foi desenvolvida e implementada uma estrutura eficaz para o comando e monitorização do motor DC. É então possível efectuar a variação da velocidade do motor através de diferentes ordens de comando.

Os resultados obtidos nos ensaios realizados mostraram-se satisfatórios e próximos dos previstos inicialmente.

É então seguro afirmar que os resultados obtidos respondem de forma positiva aos objectivos propostos inicialmente.

Conclusões e Trabalhos Futuros

Conclusões

Este projecto baseou-se no desenvolvimento de uma plataforma capaz de comandar e monitorizar as acções de um motor DC de ímanes permanentes, suportada por uma comunicação sem fios implementada através do protocolo ZigBee.

Foi desenvolvida uma interface computacional onde o utilizador é capaz de fornecer ordens de comando ao sistema de actuação e monitorizar de forma simultânea as acções resultantes.

A implementação de toda a comunicação e parte de potência assenta sobre duas placas electrónicas criadas e desenvolvidas para o efeito.

Este sistema permite comandar e monitorizar a velocidade de um motor DC de ímanes permanentes, assim como alterar o estado On/Off do mesmo.

Foi desenvolvida uma estrutura de comunicação bidireccional composta por várias tecnologias/protocolos (USB, USART, SPI, MiWi P2P), capaz de atravessar distâncias não cabladas que poderão chegar aos 100 m. Esta comunicação suporta uma velocidade de transferência de dados considerada adequada para a aplicação.

Os resultados obtidos nos ensaios realizados apresentam valores satisfatórios, conseguindo assim alcançar de forma positiva todos os objectivos inicialmente propostos.

Sugestões de trabalhos futuros

Embora este projecto tenha alcançado todas as expectativas é possível, realizando algumas alterações, desenvolver e melhorar as suas potencialidades.

O sistema desenvolvido está preparado para que seja elaborada a programação de um controlador em tempo real de forma a fechar o anel de comando e garantir erro nulo a uma referência de velocidade.

É possível, recorrendo a alterações na programação, realizar o controlo da posição angular do motor DC, assim sendo, este poderia movimentar um eixo linear de forma a gerar trajectórias uniaxiais.

Os dados de monitorização enviados para a plataforma computacional transportam, para além do valor da velocidade de rotação do motor, a informação relativa à sua posição. Desta forma, será ainda possível obter a monitorização do valor da posição angular do motor.

Numa perspectiva mais inovadora, poderá pensar-se em duplicar e desenvolver este sistema de forma a criar uma mesa de posicionamento XY controlada e monitorizada segundo este protocolo de comunicação Wireless.

Referências

[1] Baker, N., ZigBee and Bluetooth, Strengths and Weaknesses for Industrial applications, ed. I.C.C. Engineering. Vol. 16. 2005.

[2] Carlos Cardeira, A.W.C., Ronald Schoop, Wireless solutions for automation requirements. IFAC-affiliated journal, ATP International – Automation Technology in Practice, Setembro 2006. 2: p. 51-58.

[3] Fontana, R.J., Recent System Applications of Short-Pulse Ultra-Wideband (UWB) Technology. IEEE Transactions on Microwave Theory and Techniques. Vol. 52. 2004.

[4] Stallings, W., IEEE 8O2.11: wireless LANs from a to n. IT Professional. Vol. 6. 2004. 33-37.

[5] Vaughan-Nichols, S., Achieving wireless broadband with WiMax, in Industry trends. 2004. p. 10-13.

[6] Microchip, PIC18F2550 Data Sheet, Microchip, Editor. 2009.

[7] Microchip, PIC18F26K20 Data Sheet, Microchip, Editor. 2010.

[8] Microchip, PIC18F2431 Data Sheet, Microchip, Editor. 2007.

[9] Microchip, MRF24J40MA Data Sheet, Microchip, Editor. 2008.

[10] Maxim, +5V/Programmable Low-Dropout Voltage Regulator, in Data sheet, Maxim, Editor. 1994.

[11] Maxim, 5V/3.3V or Adjustable, Low-Dropout, Low IQ, 200mA Linear Regulators, in Data Sheet, Maxim, Editor. 1999.

[12] Microchip, ZENA™ Wireless Network Analyzer User's Guide, Microchip, Editor. 2008.

[13] Lattibeaudiere, D.P., Microchip ZigBee-2006 Residential Stack Protocol, in Application Note 1232, M.T. Inc., Editor. 2008.

[14] Yang, Y., Microchip MiWi™ P2P Wireless Protocol, M.T. Inc., Editor. 2008.

[15] Alzemiro Henrique Lucas, M.L., Ricardo Doth, Vinícius Emanuel Wobeto, Protocolo USB.

Anexo A

Programação em MPLAB C18

Devido à elevada extensão do código associado à programação em MPLAB C18 apenas é apresentado o seu conteúdo mais significativo.

Programação em MPLAB C18 do microcontrolador PIC18F26K20

```
void main(void)
{
    BYTE i, j;
    char cma[10];
    char pgf[10];


    BoardInit();
    ConsoleInit();
    P2PInit();
     #if defined(PICDEMZ)
        INTCONbits.GIEH = 1;

    LED_1 = 0;

    // Set default channel
    SetChannel(myChannel);

    EnableNewConnection();

    i = CreateNewConnection(2);



    // Turn on LED 1 to indicate P2P connection established
    LED_1 = 1;


   while(1)
   {

       if( ReceivedPacket() )
       {

           #ifdef ENABLE_SECURITY
               if( rxFrame.flags.bits.security )
               {
                   ConsolePutROMString((ROM char *)"Secured ");
               }
           #endif

           ConsolePut('I');
           for(i = 0; i < rxFrame.PayLoadSize; i++)
               {
                ConsolePut(rxFrame.PayLoad[i]);
               }

           // Toggle LED2 to indicate receiving a packet.
           LED_2 ^= 1;

           DiscardPacket();
```

```
      }
      else
      {
              default:

          if(DataRdyUSART())
          {
          FlushTx();
          getsUSART( cma, 4 );
              if(cma[1]!=pgf[1])
              {
                 for(i = 0; i < 4; i++)
                  {

                  WriteData(cma[i]);


                  LED_1 ^= 1;
                  }
                  BroadcastPacket(myPANID, FALSE, FALSE);
                  cma[1] = pgf[1];
              }
          }

              break;

      }
 }
```

Programação em MPLAB C18 do microcontrolador PIC18LF2431

```
void main(void)
{
   int a;

    BYTE i, cma, pgf, cmaant, pgfant, memoria;

    BoardInit();
    P2PInit();
    INTCONbits.GIEH = 1;
    LED_1 = 0;

    SetChannel(myChannel); // Set the device to operate on the optimal channel

    EnableNewConnection();

    i = CreateNewConnection(2);

    LED_1 = 1;  // Turn on LED 1 to indicate P2P connection established
    a=0;

    while(1)
    {
        if( ReceivedPacket() )
        {
```

```
                pgf = rxFrame.PayLoad[1];        //Low
                cma = rxFrame.PayLoad[2];        //High
                memoria = rxFrame.PayLoad[3];

            if (memoria == 0)
                {
                LATBbits.LATB7 = 0;
                }
            if (memoria == 1)
                {
                LATBbits.LATB7 = 1;
                }

            if (pgf != pgfant || cma != cmaant)
            {
            PWMCON1bits.UDIS = 1; //Parar a actualização    do Dutty Cicle
            cma &= 0b00111111;
            pgf &= 0b11111111;
            PDC0L = pgf;
            PDC0H = cma;
            pgfant = pgf;
            cmaant = cma;
            PWMCON1bits.UDIS = 0; //Realizar a actualização do Dutty Cicle
            }

             // Toggle LED2 to indicate receiving a packet.
            LED_2 = 1;

           DiscardPacket();
        }
        else
        {

                default:
                    a++;

                    if(a==950)//Controla a velocidade de envio dos dados.
                        {
                         FlushTx();
                         WriteData(VELRH);  //High velocidade
                         WriteData(POSCNTH);
                         WriteData(POSCNTL);
                         WriteData(VELRL);  //Low velocidade
                         BroadcastPacket(myPANID, FALSE, FALSE);
                         LED_1 ^= 1;
                         LED_2 = 0;
                         a=0;
                        }
                    break;
        }
    }
}
```

Anexo B

Programação em MicroC

Programação em MicroC do microcontrolador PIC18F2550

```
1: //ffbbb/Descrição: Este programa faz comunicação com o PC via USB. Recebe os
   dados via USART com uma
2: //           determinada taxa de comunicação e coloca-os na porta USB do PC.
3: //Autor:     Manuel Duarte Matos Lourenço.
4: //Data:      05-Out-2009.
5: //************************************************************************
   ********************
6: //     MCU:              PIC18F2550
7: //     Oscillator:       HS 8.000 MHz  (USB osc. is raised with PLL to 48.000MHz
8: //************************************************************************
   ********************
9: //     Parâmetros de configuração do terminal V1.9b:
10: //                  - Baud rate = 19200;
11: //                  - Data bits = 8;
12: //                  - Parity = none;
13: //                  - Stop Bits = 1;
14: //                  - Handshaking = none;
15: //                  - Colocar o Terminal a Enviar com uma taxa se 250ms cada um
   de 2 valores;
16: //     Nota: Atenção ao envio de dados do Terminal, para evitar a situação de
   envio simultâneo.
17: //************************************************************************
   ********************
18:
19: unsigned char k;
20: unsigned char userWR_buffer[8],userRD_buffer[8];
21:
22: //************************************************************************
   ********************
23: // Main Interrupt Routine
24: //************************************************************************
   ********************
25: void interrupt()
26: {
27:   HID_InterruptProc();
28: }
29:
30: //************************************************************************
   ********************
31: // Rotina de inicializações.
32: //************************************************************************
   ********************
33: void Init_Main()
34: {
35:   //--------------------------------------
36:   // Desabilita todas as Interrupções.
37:   //--------------------------------------
38:   OSCCON=0b11101110;                      //frequencia a 4Mz
39:   INTCON = 0;                             // Disable GIE, PEIE, TMR0IE,INT0IE,
   RBIE
40:   INTCON2 = 0xF5;
41:   INTCON3 = 0xC0;
42:   RCON.IPEN = 0;                          // Disable Priority Levels on
   interrupts
43:   PIE1 = 0;
44:   PIE2 = 0;
45:   PIR1 = 0;
46:   PIR2 = 0;
47:
48:   ADCON1 |= 0x0F;                         // Configure all ports with analog
   function as digital
49:   //--------------------------------------
50:   // C onfiguração dos Portos.
51:   //--------------------------------------
52:   TRISA = 0xFF;
53:   TRISB = 0xFF;
54:   TRISC = 0xFF;
55:
```

```
  56:   LATA = 0;
  57:   LATB = 0;
  58:   LATC = 0;
  59:
  60: }
  61:
  62: void Vals_Usart(unsigned char *Val_Us)
  63:    {
  64:     unsigned int Conf;
  65:     unsigned char a;
  66:     unsigned char Cincr;
  67:     a=1;
  68:     Conf=0;
  69:     Val_Us[0]=='0';
  70:     Val_Us[1]=='0';
  71:     Val_Us[2]=='0';
  72:     Val_Us[3]=='0';
  73:     Val_Us[4]=='0';
  74:
  75:      do
  76:         {
  77:          if (USART_Data_Ready())              // if data is received
  78:            {
  79:             Val_Us[a-1] = USART_Read();       // read the received data
  80:             if(Val_Us[0]=='I')
  81:                {
  82:                 a++;
  83:                }
  84:            }
  85:          Conf++;
  86:          if(Conf==10000)  //Se ao fim destas tentativas não receber valores
     envia sinal de erro.
  87:             {
  88:              Conf=0;
  89:              Val_Us[0]='I';
  90:              Val_Us[1]='0';
  91:              Val_Us[2]='v';
  92:              Val_Us[3]='2';
  93:              Val_Us[4]='w';
  94:              a=6;
  95:             }
  96:         }
  97:      while(a<6);
  98:
  99:    }
 100:
 101: unsigned char Send_Val_USB(unsigned char *Val_Us, unsigned char *Val_PC)
 102:    {
 103:     unsigned int Conf;
 104:     unsigned char i;
 105:
 106:     k=0;
 107:     Conf=0;
 108:        k = HID_Read();  //nº de byts do PC
 109:        i = 0;
 110:        while (i < k)
 111:             {
 112:              Val_PC [i] = userRD_buffer[i];
 113:               i++;
 114:             }
 115:         if (k)
 116:             {
 117:              k=0;
 118:              userWR_buffer[0] = Val_Us[0];
 119:              userWR_buffer[1] = Val_Us[1];
 120:              userWR_buffer[2] = Val_Us[2];
 121:              userWR_buffer[3] = Val_Us[3];
 122:              userWR_buffer[4] = Val_Us[4];
 123:              userWR_buffer[5] = Val_Us[5];
 124:              userWR_buffer[6] = Val_Us[6];
 125:              userWR_buffer[7] = Val_Us[7];
```

```
126:              userWR_buffer[8] = Val_Us[8];
127:              while (!HID_Write(&userWR_buffer, 8)) ;
128:             }
129:
130:
131:       return Val_PC;
132:
133:     }
134: //*****************************************************************************
     ********************
135: // Main Program Routine
136: //*****************************************************************************
     ********************
137:
138: void main() {
139:   unsigned char i, ch, Masc;
140:   unsigned char Val_PC [8]={0,0,0,0,0,0,0,0};
141:   unsigned char Val_Us[8]={0,0,0,0,0,0,0,0};
142:
143:   Init_Main();
144:
145:   HID_Enable(&userRD_buffer, &userWR_buffer);
146:   Delay_ms(2000); Delay_ms(2000);
147:
148:   USART_init(19200);          // Initializa o modulo da USART
149:
150:   ch = 0;
151:   Masc=1;
152:
153:
154:   while (1) {
155:
156:
157:      Vals_Usart(Val_Us);
158:      ch = Send_Val_USB(Val_Us, Val_PC);
159:
160:      if(Val_PC[1]==0x10)       //Alterado 1 - 0    10 - 70
161:         {
162:          USART_Write(Val_PC[1]);       //Byte vindo do PC.   Alterado 1 - 0
163:          USART_Write(Val_PC[2]);
164:          USART_Write(Val_PC[3]);
165:          USART_Write(Val_PC[4]);      //receber mais um byte
166:          Val_PC[1]=0x01;
167:         }
168:
169:     }
170:   HID_Disable();
171: }
```

Programa associado ao ficheiro identificativo do dispositivo USB

```
1: //*****************************************************************************
2: //
3: // File Version 1.01
4: //
5: //*****************************************************************************
6:
7: #include "Definit.h"
8: #include "VARs.h"
9:
10: //*****************************************************************************
11: // The number of bytes in each report,
12: // calculated from Report Size and Report Count in the report descriptor
13: //*****************************************************************************
14: unsigned char const HID_INPUT_REPORT_BYTES      = 8;
15: unsigned char const HID_OUTPUT_REPORT_BYTES     = 8;
16:
17: unsigned char const HID FEATURE REPORT BYTES    = 2;
18: //*****************************************************************************
19: // Byte constants
20: //*****************************************************************************
21: unsigned char const NUM ENDPOINTS               = 2;
22: unsigned char const ConfigDescr wTotalLength    = USB CONFIG DESCRIPTOR_LEN +
    USB INTERF DESCRIPTOR LEN + USB_HID_DESCRIPTOR_LEN + (NUM_ENDPOINTS *
    USB ENDP DESCRIPTOR LEN);
23: unsigned char const HID_ReportDesc_len          = 47;
24:
25: unsigned char const Low HID ReportDesc len      = HID ReportDesc len;
26: unsigned char const High_HID_ReportDesc_len     = HID_ReportDesc_len >> 8;
27:
28: unsigned char const Low HID PACKET SIZE         = HID PACKET SIZE;
29: unsigned char const High_HID_PACKET_SIZE        = HID_PACKET_SIZE >> 8;
30:
31:
32:
33: //*****************************************************************************
34: // Descriptor Tables
35: //*****************************************************************************
36: unsigned char const DescTables[USB_DEVICE_DESCRIPTOR_ALL_LEN*2] = {
37:
38: // Device Descriptor
39:     USB DEVICE DESCRIPTOR LEN, 0,           // bLength               - Length
    of Device descriptor (always 0x12)
40:     USB DEVICE DESCRIPTOR_TYPE, 0,          // bDescriptorType       - 1 =
    DEVICE descriptor
41:     0x00, 0,                                // bcdUSB                - USB
    revision 2.00 (low byte)
42:     0x02, 0,                                //
          (high byte)
43:     0x00, 0,                                // bDeviceClass          - Zero
    means each interface operates independently (class code in the interface
    descriptor)
44:     0x00, 0,                                  // bDeviceSubClass
45:     0x00, 0,                                  // bDeviceProtocol
46:     EP0 PACKET SIZE, 0,                       // bMaxPacketSize0       - maximum
    size of a data packet for a control transfer over EP0
47:     0x34, 0,                                  // idVendor              - Vendor
    ID (low byte)
48:     0x12, 0,                                  //
      (high byte)
49:     0x01, 0,                                  // idProduct             - Product
    ID (low byte)
50:     0x00, 0,                                  //
      (high byte)

51:     0x01, 0,                                // bcdDevice             - ( low
    byte)
52:     0x00, 0,                                //                         (high
    byte)
53:     0x01, 0,                                // iManufacturer         - String1
54:     0x02, 0,                                // iProduct              - String2
```

```
55:     0x00, 0,                                // iSerialNumber         - ( None )
56:     0x01, 0,                                // bNumConfigurations    - 1
57:
58: // Configuration Descriptor
59:     USB CONFIG DESCRIPTOR LEN, 0,           // bLength               - Length
    of Configuration descriptor (always 0x09)
60:     USB CONFIG DESCRIPTOR_TYPE, 0,          // bDescriptorType       - 2 =
    CONFIGURATION descriptor
61:     ConfigDescr wTotalLength, 0,            // wTotalLength          - Total
    length of this config. descriptor plus the interface and endpoint descriptors
    that are part of the configuration.
62:     0x00, 0,                                //                         ( high
    byte)
63:     0x01, 0,                                // bNumInterfaces        - Number
    of interfaces
64:     0x01, 0,                                // bConfigurationValue   -
    Configuration Value
65:     0x00, 0,                                // iConfiguration        - String
    Index for this configuration ( None )
66:     0xA0, 0,                                // bmAttributes          -
    attributes - "Bus powered" and "Remote wakeup"
67:     100, 0,                                 // MaxPower               - bus-
    powered draws 100*2 mA from the bus.
68:
69: // Interface Descriptor
70:     USB INTERF DESCRIPTOR LEN, 0,           // bLength               - Length
    of Interface descriptor (always 0x09)
71:     USB INTERFACE DESCRIPTOR_TYPE, 0,       // bDescriptorType       - 4 =
    INTERFACE descriptor
72:     0x00, 0,                                // bInterfaceNumber      - Number
    of interface, 0 based array
73:     0x00, 0,                                // bAlternateSetting     -
    Alternate setting
74:     NUM ENDPOINTS, 0,                       // bNumEndPoints         - Number
    of endpoints used in this interface
75:     0x03, 0,                                // bInterfaceClass       - assigned
    by the USB
76:     0x00, 0,                                // bInterfaceSubClass    - Not A
    boot device
77:     0x00, 0,                                // bInterfaceProtocol    - none
78:     0x00, 0,                                // iInterface            - Index to
    string descriptor that describes this interface ( None )
79:
80: // HID Descriptor
81:     USB HID DESCRIPTOR LEN, 0,              // bLength               - Length
    of HID descriptor (always 0x09)
82:     USB HID DESCRIPTOR_TYPE, 0,             // bDescriptorType       - 0x21 =
    HID descriptor
83:     0x01, 0,                                // HID class release number (1.01)
84:     0x01, 0,
85:     0x00, 0,                                // Localized country code (none)
86:     0x01, 0,                                // # of HID class descriptor to
    follow (1)
87:     0x22, 0,                                // Report descriptor type (HID)
88:     Low HID ReportDesc len, 0,
89:     High_HID_ReportDesc_len, 0,
90:
91: // EP1 RX Descriptor
92:     USB ENDP DESCRIPTOR LEN, 0,             // bLength               - length
    of descriptor (always 0x07)
93:     USB ENDPOINT DESCRIPTOR_TYPE, 0,        // bDescriptorType       - 5 =
    ENDPOINT descriptor
94:     0x81, 0,                                // bEndpointAddress      - In, EP1
95:     USB ENDPOINT_TYPE_INTERRUPT, 0,         // bmAttributes          - Endpoint
    Type - Interrupt
96:     Low HID PACKET SIZE, 0,                 // wMaxPacketSize        - max
    packet size - low order byte
97:     High HID PACKET SIZE, 0,                //                       - max
    packet size - high order byte
98:     1, 0,                                   // bInterval             - polling
    interval (1 ms)
99:
```

```
100: // EP1 TX Descriptor
101:     USB ENDP DESCRIPTOR LEN, 0,             // bLength               - length
     of descriptor (always 0x07)
102:     USB ENDPOINT DESCRIPTOR_TYPE, 0,        // bDescriptorType       - 5 =
     ENDPOINT descriptor
103:     0x01, 0,                                // bEndpointAddress      - Out, EP1
104:     USB ENDPOINT_TYPE_INTERRUPT, 0,         // bmAttributes          - Endpoint
     Type - Interrupt
105:     Low HID PACKET SIZE, 0,                 // wMaxPacketSize        - max
     packet size - low order byte
106:     High HID PACKET SIZE, 0,                //                       - max
     packet size - high order byte
107:     1, 0,                                  // bInterval             - polling
     interval (1 ms)
108:
109: // HID Report Descriptor
110:     0x06, 0,                                // USAGE_PAGE (Vendor Defined)
111:     0xA0, 0,
112:     0xFF, 0,
113:     0x09, 0,                                // USAGE ID (Vendor Usage 1)
114:     0x01, 0,
115:     0xA1, 0,                                // COLLECTION (Application)
116:     0x01, 0,
117: //  The Input report
118:     0x09, 0,                                // USAGE ID - Vendor defined
119:     0x03, 0,
120:     0x15, 0,                                //   LOGICAL_MINIMUM (0)
121:     0x00, 0,
122:     0x26, 0,                                //   LOGICAL_MAXIMUM (255)
123:     0x00, 0,
124:     0xFF, 0,
125:     0x75, 0,                                //   REPORT_SIZE (8)
126:     0x08, 0,
127:     0x95, 0,                                //   REPORT_COUNT (2)
128:     HID INPUT REPORT BYTES, 0,
129:      0x81, 0,                                    //   INPUT (Data,Var,Abs)
130:      0x02, 0,
131: //  The Output report
132:      0x09, 0,                                   // USAGE ID - Vendor defined
133:      0x04, 0,
134:      0x15, 0,                                   //   LOGICAL_MINIMUM (0)
135:      0x00, 0,
136:      0x26, 0,                                   //   LOGICAL_MAXIMUM (255)
137:      0x00, 0,
138:      0xFF, 0,
139:      0x75, 0,                                   //   REPORT_SIZE (8)

140:     0x08, 0,
141:     0x95, 0,                                //   REPORT_COUNT (2)
142:     HID OUTPUT REPORT BYTES, 0,
143:     0x91, 0,                                //   OUTPUT (Data,Var,Abs)
144:     0x02, 0,
145: //  The Feature report
146:     0x09, 0,                                // USAGE ID - Vendor defined
147:     0x05, 0,
148:     0x15, 0,                                //   LOGICAL_MINIMUM (0)
149:     0x00, 0,
150:     0x26, 0,                                //   LOGICAL_MAXIMUM (255)
151:     0x00, 0,
152:     0xFF, 0,
153:     0x75, 0,                                //   REPORT_SIZE (8)
154:     0x08, 0,
155:     0x95, 0,                                //   REPORT_COUNT (2)
156:     HID FEATURE REPORT BYTES, 0,
157:     0xB1, 0,                                //   FEATURE (Data,Var,Abs)
158:     0x02, 0,
159: //  End Collection
160:     0xC0, 0                                 // END_COLLECTION
161: };
162: //*****************************************************************************
```

```
163: unsigned char const LangIDDescr[8] = {
164:     0x04, 0,
165:     USB STRING DESCRIPTOR TYPE, 0,
166:     0x09, 0,                                    // LangID (0x0409) - Low
167:     0x04, 0                                     //                 - High
168: };
169: //*****************************************************************************
170: unsigned char const ManufacturerDescr[20] = {
171:     10, 0,
172:     USB STRING DESCRIPTOR_TYPE, 0,
173:     'F', 0, 0, 0,
174:     'E', 0, 0, 0,
175:     'U', 0, 0, 0,
176:     'P', 0, 0, 0
177: };
178: //*****************************************************************************
179: unsigned char const ProductDescr[56] = {
180:     28, 0,
181:     USB STRING DESCRIPTOR_TYPE, 0,
182:     'M', 0, 0, 0,
183:     'o', 0, 0, 0,
184:     't', 0, 0, 0,
185:     'o', 0, 0, 0,
186:     'r', 0, 0, 0,
187:     ' ', 0, 0, 0,
188:     'C', 0, 0, 0,
189:     'o', 0, 0, 0,
190:     'm', 0, 0, 0,
191:     'm', 0, 0, 0,
192:     'a', 0, 0, 0,
193:     'n ', 0, 0, 0,
194:     'd', 0, 0, 0
195: };
196: //*****************************************************************************
197: unsigned char const StrUnknownDescr[4] = {
198:     2, 0,
199:     USB_STRING_DESCRIPTOR_TYPE, 0
200: };
201: //*****************************************************************************
202:
203:
204:
205:
206:
207:
208:
209: //*****************************************************************************
210: // Initialization Function
211: //*****************************************************************************
212: void InitUSBdsc()
213: {
214: Byte tmp 0[0] = NUM ENDPOINTS;
215: Byte_tmp_0[0] = ConfigDescr_wTotalLength;
216: Byte tmp 0[0] = HID ReportDesc len;
217: Byte tmp 0[0] = Low HID ReportDesc len;
218: Byte tmp 0[0] = High HID ReportDesc len;
219: Byte tmp 0[0] = Low HID PACKET SIZE;
220: Byte_tmp_0[0] = High_HID_PACKET_SIZE;
221:
222:
223: DescTables;
224:
225: LangIDDescr;
226: ManufacturerDescr;
227: ProductDescr;
228: StrUnknownDescr;
229:
230: }
231: //*****************************************************************************
232:
```

Anexo C

Programação em Visual Studio 2008

Devido à elevada extensão do código associado à programação em Visual Studio 2008 apenas é apresentado o seu conteúdo mais significativo.

Programação da função de procura e identificação do dispositivo USB

```
Private Function FindTheHid() As Boolean

    Dim deviceFound As Boolean
    Dim devicePathName(127) As String
    Dim hidGuid As System.Guid
    Dim memberIndex As Int32
    Dim myProductID As Int16
    Dim myVendorID As Int16
    Dim success As Boolean


    myDeviceDetected = False

    GetVendorAndProductIDsFromTextBoxes(myVendorID, myProductID)

    HidD_GetHidGuid(hidGuid)

    Debug.WriteLine(MyDebugging.ResultOfAPICall("GetHidGuid"))
    Debug.WriteLine("  GUID for system HIDs: " & hidGuid.ToString)

    deviceFound = MyDeviceManagement.FindDeviceFromGuid(hidGuid, devicePathName)

    If deviceFound Then

      memberIndex = 0

      Do

        hidHandle = CreateFile _
          (devicePathName(memberIndex), 0, FILE_SHARE_READ Or FILE_SHARE_WRITE, IntPtr.Zero,
OPEN_EXISTING, 0, 0)

        Debug.WriteLine(MyDebugging.ResultOfAPICall("CreateFile"))
        Debug.WriteLine("  Returned handle: " & hidHandle.ToString)

        If Not (hidHandle.IsInvalid) Then

          ' O identificador retornado é válido,
          ' para descobrir se esse é o dispositivo que estamos à procura.

          MyHid.DeviceAttributes.Size = Marshal.SizeOf(MyHid.DeviceAttributes)

          success = HidD_GetAttributes(hidHandle, MyHid.DeviceAttributes)

          If success Then

            Debug.WriteLine("  HIDD_ATTRIBUTES structure filled without error.")
            Debug.WriteLine("  Structure size: " & MyHid.DeviceAttributes.Size)
            Debug.WriteLine("  Vendor ID: " & Hex(MyHid.DeviceAttributes.VendorID))
            Debug.WriteLine("  Product ID: " & Hex(MyHid.DeviceAttributes.ProductID))
            Debug.WriteLine("  Version Number: " & Hex(MyHid.DeviceAttributes.VersionNumber))

            ' Verifica se o dispositivo corresponde ao que nós estamos à procura.

            If (MyHid.DeviceAttributes.VendorID = myVendorID) And (MyHid.DeviceAttributes.ProductID =
myProductID) Then

              Debug.WriteLine("  My device detected")

              lstproduct.Items.Clear()   'Apagar a lstproduct
              lstvendor.Items.Clear()    'Apagar a lstvendor
```

```
                    lstvendor.Items.Add(Hex(MyHid.DeviceAttributes.VendorID)) 'Escrever o Vendor
                    lstproduct.Items.Add(Hex(MyHid.DeviceAttributes.ProductID)) 'Escrever o Product
                    lblstatus.Text = "Dispositivo Detectado"

                    ScrollToBottomOfListBox()

                    myDeviceDetected = True

                    ' Salva o DevicePathName para a função OnDeviceChange().

                    myDevicePathName = devicePathName(memberIndex)
                  Else

                    myDeviceDetected = False

                    hidHandle.Close()

                  End If

                Else

                  Debug.WriteLine("  Error in filling HIDD_ATTRIBUTES structure.")
                  myDeviceDetected = False
                  hidHandle.Close()
                End If

              End If

              ' Procurando até encontrar o dispositivo.

              memberIndex = memberIndex + 1

            Loop Until (myDeviceDetected Or (memberIndex = devicePathName.Length))

          End If

          If myDeviceDetected Then

            ' O dispositivo foi detectado.
            ' Registo para receber notificações se o dispositivo é removido ou ligado.

            success = MyDeviceManagement.RegisterForDeviceNotifications _
              (myDevicePathName, FrmMy.Handle, hidGuid, deviceNotificationHandle)

            Debug.WriteLine("RegisterForDeviceNotifications = " & success)

            ' Aprende as capacidades do dispositivo..

            MyHid.Capabilities = MyHid.GetDeviceCapabilities(hidHandle)

            If success Then

              ' Descobre se o dispositivo é um sistema de mouse ou teclado.

              hidUsage = MyHid.GetHidUsage(MyHid.Capabilities)

              ' Obter o tamanho do buffer de entrada.

              GetInputReportBufferSize()
              cmdInputReportBufferSize.Enabled = True

              readHandle = CreateFile(myDevicePathName, GENERIC_READ, FILE_SHARE_READ Or
FILE_SHARE_WRITE, IntPtr.Zero, OPEN_EXISTING, FILE_FLAG_OVERLAPPED, 0)

              Debug.WriteLine(MyDebugging.ResultOfAPICall("CreateFile, ReadHandle"))
              Debug.WriteLine("  Returned handle: " & readHandle.ToString)

              If readHandle.IsInvalid Then
```

```
            exclusiveAccess = True

          Else
            writeHandle = CreateFile(myDevicePathName, _
              GENERIC_WRITE, FILE_SHARE_READ Or FILE_SHARE_WRITE, IntPtr.Zero, OPEN_EXISTING, 0, 0)

            Debug.WriteLine(MyDebugging.ResultOfAPICall("CreateFile, WriteHandle"))
            Debug.WriteLine("  Returned handle: " & writeHandle.ToString)

            MyHid.FlushQueue(readHandle)

          End If
        End If
      Else
        ' O dispositivo não foi detectado.

        cmdInputReportBufferSize.Enabled = False
        cmdOnce.Enabled = True

        lstvendor.Items.Clear()     'Apagar Vendor
        lstproduct.Items.Clear()    'Apagar Product
        lblstatus.Text = "Dispositivo Não Detectado"


        Debug.WriteLine(" Device not found.")

        ScrollToBottomOfListBox()
      End If

      Return myDeviceDetected

    End Function
```

Programação do cursor de comando da velocidade

```
Private Sub HScrollBar1_Scroll(ByVal sender As System.Object, ByVal e As System.Windows.Forms.ScrollEventArgs)
Handles HScrollBar1.Scroll

    Dim byteValue As String
    Dim count As Int32
    Dim outputReportBuffer() As Byte
    Dim success As Boolean
    Dim a As Int16
    Dim b(2) As Byte

    If (myDeviceDetected = False) Then
       myDeviceDetected = FindTheHid()
    End If

    If (myDeviceDetected = True) Then

       success = False

       If (Not (readHandle.IsInvalid) And (Not writeHandle.IsInvalid)) Then

          ' Não tente enviar um relatório de saída se o HID não tem relatório de saída.

          If (MyHid.Capabilities.OutputReportByteLength > 0) Then

             ' Definir o limite superior do relatório de saída do buffer.
             ' Subtraia 1 a OutputReportByteLength porque a matriz começa no índice 0.

             ReDim outputReportBuffer(MyHid.Capabilities.OutputReportByteLength - 1)

             a = Convert.ToInt16(HScrollBar1.Value)
             b = BitConverter.GetBytes(a)

             outputReportBuffer(2) = 16
             outputReportBuffer(3) = b(0)
             outputReportBuffer(4) = b(1)
             outputReportBuffer(5) = 3

             If (chkUseControlTransfersOnly.Checked) = True Then

            ' Usa um controle de transferência para enviar o relatório, mesmo que o HID tenha um ponto de interrupção OUT.

                Dim myOutputReport As New Hid.OutputReportViaControlTransfer
                success = myOutputReport.Write(outputReportBuffer, writeHandle)
             Else

             ' Usa WriteFile para enviar o relatório.' Se o HID tem uma interrupção OUT terminal,

                Dim myOutputReport As New Hid.OutputReportViaInterruptTransfer
                success = myOutputReport.Write(outputReportBuffer, writeHandle)
             End If

             If success Then
                    For count = 1 To UBound(outputReportBuffer)
                    ' Mostra bytes com 2 caractéres Hex.
                    byteValue = String.Format("{0:X2} ", outputReportBuffer(count))
                     Next count
             Else
                lstResults.Items.Add("The attempt to write an Output report has failed.")
             End If
          Else
             lstResults.Items.Add("The HID doesn't have an Output report.")
          End If
       End If
    End If

End Sub
```

Programação da leitura dos dados recebidos relativos à monitorização da velocidade

```
Private Sub GetInputReportData(ByVal ar As IAsyncResult)
        Dim byteValue As String
       'Dim StrVal_2_Byts As String
       'Dim StrCodControl As String
       Dim valorl As Int16 'low byte velocidade
       Dim valorh As Int16 'high byte velocidade
       Dim valorhh As Int16
       Dim valor As Double
       Dim strvalor As String 'valor velocidade instantanea

       Dim count As Int32
       Dim Sinc As Integer = 0        'Conta o numero de ocorrências de byte iguais.
       Dim Val_Ant As Integer = 0     'Guarda o valor anterior.
       Dim inputReportBuffer As Byte() = Nothing
       Dim success As Boolean

       'Define um delegado usando o objeto IAsyncResult.

       Dim deleg As ReadInputReportDelegate = _
          DirectCast(ar.AsyncState, ReadInputReportDelegate)

       deleg.EndInvoke(myDeviceDetected, inputReportBuffer, success, ar)

       If (ar.IsCompleted And success) Then

          For count = 1 To UBound(inputReportBuffer)

             ' Mostra bytes com 2 caractéres Hex.

             byteValue = String.Format("{0:X2} ", inputReportBuffer(count))

             Select Case count

                Case 2

                   valorhh = inputReportBuffer(2)
                   valorh = valorhh << 8
                   MyMarshalToForm("AddItemTolst2", byteValue)

                Case 3

                   'Val_2_Byts = inputReportBuffer(3)
                   'Val_1 = Val_2_Byts
                   'Val_2_Byts = Val_2_Byts << 8     'Val_2_Byts * 256
                   'Escreve na listByteRecevido
                   'MyMarshalToForm("AddItemTolstByteRecevido", byteValue)

                Case 4

                   'Val_2 = inputReportBuffer(4)
                   'Val_2_Byts = Val_2_Byts + Val_2
                   'Para ficar com sinal. Esta operação é compensada no PIC2431.
                   'CodControl = Val_2_Byts
                   'Val_2_Byts = Val_2_Byts - 32768
                   'StrVal_2_Byts = CStr(Val_2_Byts)

                   'angulo = Val_2_Byts
                   'angulo = Val_2_Byts
                   'angulo = (Val_2_Byts * 360) \ 1500 '*'\=Divisão inteira.
                   'angulo = (Val_2_Byts * 360) \ 1500 '*'\=Divisão inteira.

                   'Actualisa a lstValInst.
                   'Actualisa o ponteiro.

                   'MyMarshalToForm("AddItemTolstValInst", StrVal_2_Byts)
                   'MyMarshalToForm("AddItemTolstByteRecebido1", byteValue)
```

```
            Case 5

                valorl = inputReportBuffer(5)

                valorh = valorh + valorl

                valor = (((8000000 * 64 / 4)) / (400 * 4 * 1 * valorh)) * 60

                Val_2_Byts = Convert.ToInt16(valor)

                strvalor = CStr(Val_2_Byts)


                MyMarshalToForm("AddItemTolstvel", strvalor)
                MyMarshalToForm("AddItemTolst5", byteValue)

            Case 6

        End Select

    Next count

Else

End If

' Habilitar solicitando outra transferência.

MyMarshalToForm("EnableCmdOnce", "")

End Sub
```