

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



FEUP

Emulador em Hardware de Floppy Disk Drive com acesso sem fios

José Pedro Patrício Gonçalves de Sá

Dissertação submetida no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e
de Computadores

Orientador: Hédio Sousa Mendonça (Professor Doutor)

2011

A Dissertação intitulada

“Emulador em Hardware de Floppy Disk Drive com Acesso sem Fios”

foi aprovada em provas realizadas em 19-10-2011

o júri



Presidente Professor Doutor José Carlos dos Santos Alves
Professor Associado do Departamento de Engenharia Electrotécnica e de
Computadores da Faculdade de Engenharia da Universidade do Porto



Professor Doutor Manuel Gradim de Oliveira Gericota
Equiparado a Professor Adjunto do Departamento de Engenharia Electrotécnica do
Instituto Superior de Engenharia de Porto



Professor Doutor Hélio Mendes de Sousa Mendonça
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.



Autor - José Pedro Patrício Gonçalves Sá

Resumo

O uso de disquetes em equipamento antigo mas funcionalmente relevante torna-se cada vez menos fiável e de difícil manutenção. Nesta tese propõe-se a substituição das drives de disquetes por dispositivos emuladores que, para além de dispensarem o uso das disquetes, acrescentam valor ao permitir a transferência da informação através de comunicação sem-fios.

Com o avanço tecnológico sempre em evolução, é apenas natural que suportes de informação, um dia considerados universais, sejam preteridos para dar lugar a outros mais fiáveis, fáceis de utilizar e de maior capacidade. As disquetes que no passado eram facilmente acessíveis, hoje em dia, desapareceram de circulação. No entanto, não é incomum descobrir equipamentos, nomeadamente industriais, ainda úteis no dia de hoje que, por diversos motivos, ainda utilizam disquetes na sua operação. Devido à sua utilidade não se pretende a sua substituição completa e a sua adaptação através de um emulador é preferida.

Neste trabalho é efetuado o estudo relacionado com a comunicação entre uma drive de disquetes e o seu controlador de forma a permitir que a drive possa ser substituída de forma totalmente transparente e não intrusiva. Nesse sentido é implementado um emulador com o auxílio de um microcontrolador e de uma imagem de disco que replica os dados presentes nas disquetes.

O projeto apresenta ainda o valor acrescentado de permitir que a transmissão dos dados a serem emulados sejam feitos através de uma ligação sem fios, solução inovadora em emuladores de drive de disquetes. Na tese descreve-se a implementação do emulador e da comunicação sem fios. De forma a auxiliar estas funcionalidades, são ainda criadas ferramentas de apoio a estes sistemas nomeadamente software de gestão e transmissão das imagens de dados.

Durante o desenvolvimento da solução implementada, e devido à especificidade da sua comunicação, são construídas ferramentas de análise e teste apropriadas para a comunicação entre o emulador e o o controlador.

São efetuados ensaios experimentais de emulação e da transmissão através da ligação sem fios que demonstram a validade da solução proposta.

Abstract

The use of floppy disks in old but still relevant equipment, has become less reliable and of difficult maintenance. In this thesis we propose the replacement of floppy disk drives with an emulator device which, other than not requiring the use of floppy disks, also adds value by allowing the data transfer to occur by a wireless communication link.

With constant technological evolution, it is only natural that older storage mediums, once considered universal, are now deprecated and being substituted by more reliable, easy to use and of larger capacity equipment. Floppy disks that once were easily accessible, today have almost disappeared. Meanwhile, it is not uncommon to find useful equipment, especially in industry, that, for different reasons, still uses floppy disks in their operation. Considering the usefulness of these equipments, their substitution is not desirable and its conversion through an emulator is preferred.

We performed the study of the communication protocol between the floppy disk drive and its controller to achieve the replacement of the floppy drive in a transparent and non intrusive way. An emulator is built with the aid of a microcontroller and a disk image that copies all the data present in a floppy disk.

Together with the advantage of no longer requiring the usage of floppy disk, the project also presents the added value accomplished by the wireless communication, an innovative solution in floppy disk emulators. The thesis describes the implementation of the emulator and wireless communication. To help these features we also created tools to support these systems including software that manages and transmit the disk images.

During the development of the project, and because of the uniqueness of their communication, auxiliary tools are built to analyze and test the communication between the emulator and the floppy disk controller.

Experimental tests of the emulation and transmission through the wireless link were performed to demonstrate the validity of the proposed solution.

Agradecimentos

Este tese nunca poderia ter sido possível sem o apoio de inúmeras pessoas:

Gostaria de agradecer ao Professor Hélio Mendonça pela oportunidade e paciência que demonstrou durante a realização deste projeto.

Ao Professor José Martins Ferreira, pelos ensinamentos transmitidos (incluindo os das aulas de Microcontroladores há muitos anos atrás) e por me ter dado a motivação final que me permitiu concluir este trabalho.

Uma palavra de agradecimento muito especial aos meus Pais que sempre acreditaram em mim e, muito para além de pais, são amigos e verdadeiros heróis. A eles, um Muito Obrigado.

À minha irmã pela sua sabedoria e suas palavras de apoio incondicional. Cinco mil quilómetros, que durante tanto tempo nos separaram, são insignificantes.

À minha namorada mais do que um agradecimento, devo também um pedido de desculpa por tanto que se gostaria de ser feito e que teve de ser adiado. Obrigado por acreditares em mim.

Agradeço a todos os meus amigos que me acompanharam durante todos estes longos anos do curso. Não tenho dúvida que ter-vos conhecido fez de mim uma pessoa melhor.

Conteúdo

Índice	viii
Lista de Figuras	x
Lista de Tabelas	xi
Abreviaturas e Glossário	xiii
1 Introdução	1
1.1 Motivação	1
1.2 Objetivos	2
1.3 Estrutura do Documento	2
2 Enquadramento	5
2.1 Uso de disquetes em equipamento industrial	5
2.2 Uso de disquetes em equipamentos não industriais	8
2.3 Soluções Existentes ou Alternativas	9
3 Discos Magnéticos Flexíveis	13
3.1 História	13
3.2 Disquete e o Disco Magnético	16
3.3 Interface Controlador	18
3.3.1 Codificação MFM	21
3.4 Formato das Faixas e Sectores	23
3.4.1 CRC	28
3.4.2 Campo de Dados	28
4 Interface Sem Fios	29
4.1 Tecnologias de Comunicação Sem Fios	29
4.2 Norma ZigBee	31
4.3 Z-Stack	34
4.4 Mensagens ZigBee	34
5 Desenvolvimento e Implementação	37
5.1 Plataforma e Ferramentas de Desenvolvimento	37
5.1.1 RC2300	38
5.1.2 Kit de Desenvolvimento CC2431DK	41
5.1.3 Equipamento Auxiliar	42
5.1.4 Software Auxiliar	43

5.2	Gestor remoto	48
5.2.1	Software Gestor	48
5.2.2	Comunicação via porta série	50
5.3	Emulador	52
5.3.1	Modo Emulador	53
5.3.2	Modo Carregar Imagem	55
5.3.3	Hardware	56
5.4	Formato da imagem	60
5.5	Memória de armazenamento	65
5.6	Transmissão da imagem	65
5.7	Implementação Protocolo MFM	68
5.7.1	Hardware dedicado	68
5.7.2	Sinal auxiliado por hardware	69
5.7.3	<i>Bit-Banging</i>	71
5.7.4	Solução em <i>Assembly</i>	73
6	Resultados	77
6.1	Emulação	77
6.2	Transmissão sem fios	79
7	Conclusões e Trabalho Futuro	81
7.1	Conclusões	81
7.2	Trabalho Futuro	82
A	Esquemático e desenho da PCB do Emulador	85
	Referências	94

Lista de Figuras

2.1	Tear mecânico com cartões perfurados.	6
2.2	Atari 1040ST com drive de disquetes.	8
3.1	Disquetes de 8, 5 ¼ e 3 ½ polegadas.	15
3.2	Diagrama de um disco magnético e cabeça de leitura/escrita.	18
3.3	Cabo de dados com inversão de sinais de seleção de drive e conector IDC.	20
3.4	Captura de comunicação entre a drive e o controlador de disquetes.	21
3.5	Codificação MFM	23
3.6	Byte 0x59 nas codificações FM e MFM	23
3.7	Byte 0xC2 quando codificado no campo de dados ou usado com sinal de sincronismo.	26
4.1	Esquema simplificado da arquitetura da pilha ZigBee criada em cima da norma IEEE 802.15.4	32
4.2	Exemplo de configuração de uma rede ZigBee com Coordenador, Encaminhadores e Ponto Terminal	33
4.3	Esquema de transporte de dados dos pacotes nas diversas camadas que compõem o protocolo ZigBee	35
5.1	Diagrama geral.	38
5.2	Módulo RC2300 da Radiocrafts.	39
5.3	Software do analisador lógico USBeeSX em modo de demonstração.	44
5.4	Interface da aplicação de análise do analisador <i>MFM Log Parser</i>	46
5.5	Interface (a) e Programador (b) da memória externa com recurso ao Bus Pirate.	47
5.6	Interface da aplicação Gestor.	49
5.7	Diagrama de funcionamento geral do Emulador.	52
5.8	Diagrama de funcionamento do Emulador ao processar os sinais do controlador.	54
5.9	Diagrama de funcionamento do Emulador no modo Carregar Imagem	55
5.10	Conector de energia de 4 pinos.	56
5.11	Sinais /DRVSx e /MOTEx ativados simultaneamente pelo controlador de drive de disquetes.	58
5.12	Esquema lógico do circuito de ativação do Emulador.	58
5.13	Circuito de deteção de degrau nos sinais /DRVS e /HEADS	60
5.14	Envio de imagem: Diagrama de sequência de mensagens entre Gestor, Coordenador, Emulador e Memória	66
5.15	Pino de saída em resposta aos comandos efetuados na Listagem 5.1	72
6.1	Listagem do conteúdo da imagem emulada.	78
6.2	Medição de tempos na transmissão sem fios.	79

A.1	Esquemático da placa de extensão ao módulo SoC_BB do dispositivo Emulador. .	86
A.2	PCB da placa de extensão ao módulo SoC_BB do dispositivo Emulador.	87
A.3	Protótipo do emulador ligado ao módulo SoC_BB	88

Lista de Tabelas

3.1	Diversos formatos de disquetes e suas características.	16
3.2	Pinos do cabo de comunicação entre drive de disquetes e controlador.	19
3.3	Organização dos campos numa faixa.	24
3.4	Descrição das características dos campos das faixas.	25
3.5	Campo preâmbulo da faixa.	25
3.6	Campo de identificação do sector.	26
3.7	Campo de dados do sector.	27
3.8	Campo de enchimento.	27
3.9	Comprimento total de uma faixa.	27
3.10	Campo de Identificação de um sector	28
4.1	Comparação de sistemas de comunicação sem fios	31
5.1	<i>Pinout</i> RC2300.	40
5.2	Formato e tamanho dos campos do pacote utilizado na comunicação série.	50
5.3	Comandos utilizados na comunicação.	51
5.4	Sinais de entrada da drive de disquetes.	57
5.5	Sinais de saída da drive de disquetes	59
5.6	Formas de onda MFM e a sua representação binária em ficheiros de imagem.	64
5.7	Octetos SPI a simular bits na modulação MFM	70
6.1	Valores temporais médios registados na rotina de transmissão de dados.	79

Abreviaturas e Glossário

Abreviaturas

CI	Circuito Integrado
CCITT	<i>Comité Consultatif International Téléphonique et Télégraphique</i>
CRC	<i>Cyclic Redundancy Check</i> (Verificação de Redundância Cíclica)
CPLD	<i>Complex Programmable Logic Device</i> (Dispositivo Lógico Complexo Programável)
DD	<i>Double Density</i> (Dupla Densidade)
DEEC	Departamento de Engenharia Eletrotécnica e de Computadores
DJNZ	<i>Decrease and Jump if Not Zero</i> (Decrementa e salta caso não seja zero)
DMA	<i>Direct Memory Access</i> (Acesso Direto à Memória)
ED	<i>Extended Density</i> (Densidade Estendida)
E/S	Entrada/Saída
IDC	<i>Insulation Displacement Connector</i>
I/O	<i>Input/Output</i> (Entrada/Saída)
FAT	<i>File Allocation Table</i> (Tabela de Alocação de Ficheiros)
FCS	<i>Frame Check Sequence</i> (Verificador da Trama)
FDC	<i>Floppy Disk Controller</i> (Controlador de drive de disquete)
FDD	<i>Floppy Disk Drive</i> (Drive de disquete)
FEUP	Faculdade de Engenharia da Universidade do Porto
FM	<i>Frequency Modulation</i> (Frequência Modulada)
HD	<i>High Density</i> (Alta Densidade)
JTAG	<i>Joint Test Action Group - Standard Test Access Port and Boundary-Scan Architecture</i>
kbps	<i>Kilobits per second</i> (Kilobits por segundo)
Mbps	<i>Megabits per second</i> (Megabits por segundo)
MFM	<i>Modified Frequency Modulation</i> (Frequência Modulada Modificada)
OSAL	<i>Operating System Abstraction Layer</i> (Camada de Abstração de Sistema Operativo)
PCI	<i>Peripheral Component Interconnect</i> (Interconetor de Componentes Periféricos)
PLL	<i>Phase-Lock loop</i> (Malha de Captura de Fase)
PWM	<i>Pulse-Width Modulation</i> (Modulação por largura de pulso)
RAM	<i>Random Access Memory</i> (Memória de Acesso Aleatório)
RF	Radiofrequência
rpm	Rotações por minuto
RLL	<i>Run-Length Limited</i>
SD	<i>Single Density</i> (Densidade simples)
SoC	<i>System-On-Chip</i> (Sistema em circuito integrado)
SOP	<i>Start Of Packet</i> (Início de Pacote)
SPI	<i>Serial Peripheral Interface</i> (Interface Periféricos Série)
tpi	<i>tracks per inch</i> (Faixas por polegada)

UART	<i>Universal Asynchronous Receiver/Transmitter</i> (Recetor/Transmissor Assíncrono Universal)
USB	<i>Universal Serial Bus</i> (Barramento Série Universal)
WPAN	<i>Wireless Personal Area Network</i>
ZDO	<i>Zigbee Device Object</i> (Objeto de Dispositivo ZigBee)

Glossário

Controlador	Dispositivo responsável pela interligação entre a aplicação Gestor e o dispositivo Emulador.
Emulador	Dispositivo construído no âmbito deste projeto que é responsável pela emulação de uma drive de disquetes.
Firmware	Programa carregado e executado por um microcontrolador.
Gestor	Aplicação informática responsável pela gestão e envio das imagens das disquetes para o Emulador.
Imagem	Ficheiro binário que replica o conteúdo de um suporte de dados como um disco rígido ou disquete.
Octeto	Conjunto de 8 bits. O mesmo que um <i>byte</i> .
8051	Família de microcontroladores de 8 bits desenvolvidos pela Intel.

Capítulo 1

Introdução

A inovação tecnológica, imparável, tráz evolução aos diversos processos produtivos. Máquinas e tecnologias mais rápidas, fiáveis e potentes aparecem para substituir outras mais antigas. No entanto, estes avanços são por vezes disruptivos o que os pode tornar indesejáveis e a implementação destas novas tecnologias, em substituição das antigas, poderá ter um custo demasiado elevado para a aplicação em causa. A utilização de disquetes magnéticas em equipamentos industriais é uma das aplicações onde o uso de uma tecnologia, entretanto obsoleta, diminui o valor geral do equipamento. Da necessidade de adaptar este processo tecnológico de forma a poder ser aplicado também a equipamentos mais antigos, surge a possibilidade de criar pontes ente as tecnologias mais antigas e as mais recentes.

1.1 Motivação

A motivação por detrás deste trabalho está relacionada com a possibilidade de acrescentar valor a equipamentos mais antigos. A utilização de disquetes contribui para o decréscimo de fiabilidade e facilidade de uso nos equipamentos que recorrem a este suporte de dados.

De forma a valorizar instrumentos, que de outra forma ficariam, obsoletos recorre-se à adaptação destes equipamentos que utilizam tecnologias mais antigas por outras mais modernas.

No caso de equipamentos que utilizam disquetes, pretende-se libertá-los da sua dependência de um suporte descontinuado e dota-los de tecnologias que tragam mais valias ao seu funcionamento como é a fornecida pela comunicação sem fios. A solução de um emulador de uma tecnologia

obsoleta com um módulo de comunicação sem fios permitem fazer a ponte entre o passado e o presente.

1.2 Objetivos

No âmbito desta tese pretende-se substituir um módulo de disquetes por uma tecnologia mais atual e fiável. Propusemo-nos a:

- Efetuar um levantamento exemplificativo de equipamentos existentes que podem ser alvo desta adaptação através da troca da drive de disquetes;
- Estudar o funcionamento básico das disquetes magnéticas, das suas drives e da forma como comunicam com o controlador.
- Implementar em hardware um módulo que consiga comunicar de forma transparente com um controlador de drive de disquetes;
- Criar software dedicado que permita a gestão das imagens a serem emuladas;
- Utilizar tecnologias sem fios ZigBee como meio de comunicação e transporte de dados;
- Implementar ferramentas de testes e análise que auxiliem o desenvolvimento da emulação.

1.3 Estrutura do Documento

Este documento encontra-se dividido em 7 Capítulos que expõem o problema e descrevem as soluções testadas e utilizadas na resolução da dissertação.

Este texto de introdução ao trabalho compõe o primeiro Capítulo.

O segundo Capítulo apresenta o meio onde se pretende que o emulador criado seja utilizado. Descreve a evolução dos teares mecânicos e de outros equipamentos industriais que eventualmente levou ao uso de disquetes. Analisa ainda uma variedade de equipamentos e cenários que utilizam esta tecnologia dando relevo ao interesse deste tipo de emuladores. Na parte final do Capítulo são listados alguns emuladores já existentes.

No terceiro Capítulo é feito um estudo mais aprofundado da tecnologia associada aos discos magnéticos flexíveis. O Capítulo inicia a abordagem com uma introdução histórica passando de

seguida para uma análise das propriedades físicas e dos formatos existentes. É ainda analisada a comunicação entre a drive de disquetes e o respetivo controlador com especial relevo dado ao protocolo MFM onde são modulados os dados. O Capítulo termina com uma análise ao formato de armazenamento dos dados nas faixas que compõem a disquete.

O quarto Capítulo é referente à comunicação sem fios. Analisa diversas tecnologias e apresenta as tecnologias utilizadas: o protocolo ZigBee e a plataforma de programação Z-Stack, que o implementa.

No quinto Capítulo é descrito todo o processo de implementação do emulador. Começa por introduzir as plataformas que são usadas para implementar o emulador e as ferramentas utilizadas no desenvolvimento do mesmo. Segue-se a descrição dos diversos elementos que irão comunicar com o emulador e a forma como essa comunicação é realizada. É ainda descrito o emulador e o respetivo funcionamento. São endereçados os diversos componentes necessários ao funcionamento do emulador e descritos os desafios encontrados durante o seu desenvolvimento.

No sexto Capítulo encontram-se os resultados obtidos na implementação do dispositivo emulador onde o seu desempenho é analisado, tanto na componente de emulação como de transmissão de dados.

O sétimo e último Capítulo apresenta as conclusões gerais do trabalho realizado e ainda algumas considerações sobre desenvolvimentos futuros.

Capítulo 2

Enquadramento

A relevância e utilidade de um trabalho académico em engenharia podem ser medidas, entre outras métricas, pelo seu impacto no mundo industrial ou doméstico. Será tanto mais útil quanto mais prática for a sua aplicação em outros ambientes. Não é por isso de estranhar, que muitos projetos realizados em âmbito académico, sejam consequência de problemas encontrados no seio industrial.

No caso em questão, descrevem-se contextos de aparelhos que utilizam, na sua sequência de trabalho, dados em suporte magnético. No entanto, este suporte poderá ser substituído de forma a obter a mesma funcionalidade.

Nesta secção, descrevem-se alguns tipos de equipamentos que se enquadram nesta descrição e na qual um emulador de drive de disquetes poderá ser utilizado. São ainda analisadas soluções já existentes.

2.1 Uso de disquetes em equipamento industrial

A par da penetração das disquetes magnéticas no mercado doméstico, estas tiveram também um papel relevante numa gama de equipamentos destinados à indústria. O recurso à mecanização para facilitar a execução de operações complexas de forma rápida permitiu uma rápida evolução na própria complexidade das ações. Para compensar esta evolução, ferramentas mais simples

foram surgindo de forma a codificar as instruções, não necessitando de serem feitas diretamente na máquina, no meio de uma fábrica, mas sim, por exemplo, no conforto de um escritório. Passa apenas a ser necessário transferir a informação do local onde foi criada para o equipamento que a vai executar. Uma solução é precisamente o uso das disquetes magnéticas, um suporte fácil de utilizar e extremamente portátil que respondia a este desafio de forma ideal. Surgem assim equipamentos industriais com drives de disquetes de 3 ½ ou 5 ¼ polegadas nas suas consolas de programação e controlo.

Uma das atividades que utilizou (e ainda utiliza) esta técnica é a indústria ligada à confeção têxtil.

Na evolução dos teares manuais para os teares mecânicos, no início do Séc XIX, Joseph Marie Jacquard implementou um tear cuja operação e desenho do tecido final eram controlados através de uma sequência de sinais em cartões perfurados como o representado na Figura 2.1. Desta forma, conseguia-se uma vasta variedade de padrões fabricados de forma automatizada e facilmente replicável.



Figura 2.1: Tear mecânico com cartões perfurados. [1]

Com o aumento da complexidade dos teares mecânicos e eletrónicos e da informação necessária para os operar, os cartões perfurados são abandonados e chega-se, eventualmente, ao uso das disquetes.

Os controladores eletrónicos dos teares são então equipados com drives de disquetes que carregam os padrões a tecer. Apesar de usarem drives de disquetes, as semelhanças com o computador

peçoal ficam-se por aqui. Estes controladores não possuem outros métodos de Entrada e Saída alternativos como ranhuras PCI que possam ser usados. Estes sistemas também não possuem um sistema operativo que possa ser alterado no sentido de modificar a forma como a informação é carregada.

Paralelamente aos teares descritos, outros equipamentos industriais utilizam este método para introdução de dados. De forma a melhor entender a penetração deste uso particular, procuraram-se alguns exemplos práticos destes equipamentos e segue-se uma lista não exaustiva, mas exemplificativa, da influência que as disquetes tiveram, e em alguns casos ainda têm, no meio industrial.

Ainda no âmbito da indústria têxtil, e paralelamente à tecelagem, também se encontra o mesmo tipo de controladores em máquinas de bordar ou de malha. Alguns exemplos são:

- SWF Dual Function Embroidery Machines SWF/B-902DF – Máquina de bordar automática [2];
- Brother BE-1206B-BC StandAlone Type – Máquina de bordar automática com 6 cabeças [3];
- Brother BE-1201BAC – Máquina de bordar comercial (2007) [4];
- HAPPY HGC-Series Multi-Head Machine – Máquinas de bordar com múltiplas cabeças [5];
- Tajima TMEG-G915 – Máquina de bordar industrial [6];
- Brother Pacesetter ULT2001 – Máquina de bordar orientada ao mercado doméstico [7].

Outro tipo de equipamentos que utilizam instruções geradas por computador são as máquinas de CNC (*Computer Numerical Control*) que utilizam informação processada em ferramentas de manipulação CAD (*Computer Assisted Design*) como os sistemas:

- VKM4 - 3 Axis CNC The Knee Mill with MillSlide – Fresadora de 3 eixos [8];
- MILLPWR CNC Control Systems – Controlador para máquinas CNC [9];
- ABB IEB 1400 – Braço Robótico [10].

Finalmente, são listadas algumas ferramentas que usam as disquetes como suporte temporário para exportar informação que pode ser processada noutros ambientes:

- Tektronix TDS340A - Osciloscópio Digital [11];
- Agilent 35670A FFT Dynamic Signal Analyzer - Analisador de sinal [12].

Atentos à evolução e necessidades dos seus clientes, os fabricantes destas ferramentas também os equiparam com outros métodos de transmissão de dados tal como comunicação série ou de rede. Equipamentos mais moderno já apresentam o uso de portas USB que permitem a leitura de memórias que carregam os dados pretendidos. De notar que estas opções alternativas vão coexistindo com o uso de disquetes e que em alguns casos é dada a opção ao cliente sobre qual o sistema a usar. Um exemplo desta opção pode ser consultada nas especificações da máquina de CNC da Milltronics VKM4 que oferece a opção de uso de um controlador alternativo que utiliza memórias USB em vez de disquetes [8].

2.2 Uso de disquetes em equipamentos não industriais

Para além dos sistemas industriais, o uso de disquetes também deixou a sua marca nos computadores pessoais e consolas de jogos lançados desde os anos 70 até aos meados dos anos 90. Fabricantes como a Atari, Commodore, Sinclair [13, 14], etc, lançaram diversos equipamentos orientados ao mercado doméstico, que, se nas primeiras versões dos seus produtos utilizaram cassetes para carregar e guardar programas. Tal como se pode ver na Figura 2.2, estas soluções acabaram por evoluir para o uso de disquetes devido à sua maior fiabilidade, capacidade e sobretudo a possibilidade de acesso não sequencial.



Figura 2.2: Atari 1040ST com drive de disquetes. [15]

É no entanto necessário registar que alguns destes sistemas usavam formatos e codificações fechadas numa altura em que cada fabricante tentava implementar a sua norma de forma a maximizar a capacidade disponível ou simplesmente tentar evitar as tentativas de cópia pirata [16, 17]. Apesar

destas modificações no protocolo não serem compatíveis com a norma seguida neste trabalho não deixa de ser interessante observar a existência destas soluções.

Finalmente, e a par de equipamento fabril e doméstico, regista-se mais uma utilização de equipamentos de alto valor para os seus proprietários mas que utilizam novamente disquetes no seu funcionamento. Alguns exemplos são os equipamentos ligados à indústria da luz e som que utilizam disquetes de forma a guardar informação referente a sequências de notas que podem ser mais tarde recuperadas ou utilizadas em outros equipamentos como:

- Yamaha PSR-620 – Teclado Digital lançado em 1995 [18];
- Jands Hog 500 – Consola de controlo de luz [19].

2.3 Soluções Existentes ou Alternativas

Com o progressivo abandono do uso de disquetes foram desaparecendo dos equipamentos as respetivas drives. Isto criou problemas em duas frentes principais: 1) como fazer a interface entre equipamentos antigos e novos; 2) como armazenar a informação. No mercado doméstico, por exemplo, este problema foi resolvido com o aparecimento de diversas drives de disquetes que interagem com o sistema anfitrião através da interface USB. Não era incomum ver na lista de acessórios de computadores portáteis a drive de disquetes externa, que permitia ao utilizador aceder aos seus dados [20]. No entanto esta solução não é resposta ao problema do equipamento industrial por três fatores: 1) não dispensa o uso de disquetes; 2) o equipamento industrial em foco raramente é equipado de interface USB; 3) a lógica interna teria de procurar a informação noutro dispositivo que não o esperado.

A solução, passa pela substituição da drive de disquetes por um emulador. O emulador transforma os dados antes presentes numa disquete na linguagem utilizada entre a drive de disquetes e o controlador.

São então encontradas algumas soluções que procuram resolver o problema apresentado. Para além de produtos disponíveis comercialmente, encontrou-se também alguma informação sobre conceitos relacionados, por exemplo, uma patente da 3Com Corporation, datada de 1995m que procurava implementar um “aparelho para usar memória flash como emulador de disquete num computador” [21]. Este sistema, no entanto, apenas apresenta como ponto de interface o próprio emulador, não prevendo que funcione como ponte de ligação com outros sistemas ou protocolos.

Em relação ao segundo problema, o da preservação da informação armazenada em disquetes magnéticas, regista-se também interesse e consequente investigação [22–24]. Parte do estudo realizado sobre discos magnéticos flexíveis é relativo à recuperação e manutenção de informação, no seu sentido lato e não no protocolo de comunicação em si, com vista à sua emulação e substituição direta [24].

Como fonte de informação registam-se algumas folhas de características dos controladores de disquetes [25–28] que entretanto foram absorvidos por *superchips* de E/S que equipam as placas mãe e lidam com diversos protocolos e diferentes interfaces. Parte destas fontes de informação são bastante datadas relativamente aos controladores atuais mas possuem ainda informação útil no que diz respeito ao seu funcionamento básico. Por outro lado, encontrou-se também informação relevante nas folhas de características das drive de disquetes como a Teac FD-235HF [29] ou Shoreline X1DE-31U [30].

Outras fontes de informação utilizadas na realização deste projeto foram compiladas por uma comunidade de curiosos e hobbyistas de jogadores de videojogos em computadores pessoais. Entusiastas documentaram relativamente bem as interfaces utilizadas nestas plataformas numa tentativa de não perderem os jogos da sua juventude e de os poderem continuar a jogar no ambiente que tão bem conheciam. Como o suporte magnético é perecível, gastando-se com o tempo e o número de leituras, era necessário copiar a informação guardada nos discos para um suporte não degradável. Apesar do objeto alvo deste projeto ser diferente, o meio usado e tecnologias usadas são as mesmas sobre as quais estes projeto se debruça. Alguns destes projetos são analisados.

O “*Emulateur de lecteur de disquete*” [31] executado por Jean Francois del Nero consiste num emulador da drive de disquetes. Este projeto usa imagens de discos anteriormente copiados das disquetes originais eliminando da equação os problemas que caracterizam os discos flexíveis. As imagens dos discos são carregadas para o emulador via USB desde um computador anfitrião, ou através de um cartão de memória *Secure Digital* que pode alojar várias imagens.

O “*Amiga Floppy Emulator, Mark II*” [32], é outro projeto realizado por Tim Tashpulatov, um entusiasta dos computadores da década de 80 que realizou uma interface e emulador para comunicar com a sua consola e poder dispensar as disquetes.

Por último surge o projeto do emulador “*Cumulus*” [33]. Os seus sistemas alvo *Oric-I* e *Oric Atmos* não possuem o controlador de drive de disquetes integrado recorrendo assim a acessórios que ligam a portas de extensão. Este projeto tem a particularidade de emular não só a disquete mas também o controlador associado usando assim um ponto de contacto diferente.

Entre projetos levados a bom termo e ainda outros em curso, verifica-se que existe elevado interesse neste tipo de soluções que não interfiram com o equipamento anfitrião. Mais, a existência

deste tipo de projetos não industriais, mostra que é possível a sua criação através do recurso a equipamento de relativo baixo custo e a elementos facilmente acessíveis.

Para além destes projetos, são conhecidos alguns equipamentos emuladores criados especificamente para a substituição das drives de disquetes em equipamentos industriais.

O *USB Floppy Emulator* [34], lançado pela empresa alemã ipcas GmbH, apresenta na sua face posterior os mesmos conectores de energia e de disquetes de uma drive normal, tendo na sua face frontal uma porta USB à qual se pode ligar uma vulgar memória USB. Botões e um pequeno visor numérico permitem escolher a disquete virtual que se pretende utilizar uma vez que a memória USB pode conter dados com a informação de 100 disquetes.

Outra empresa, PLR Electronics, especializada na reparação de equipamento de bordados, criou um emulador de drive de disquetes como uma mais valia que permite ainda o acesso via rede Ethernet, para além do acesso via memória USB [35].

Datex, uma empresa orientada à recuperação de dados e reparação de hardware, possui também uma linha de produtos de emulação, no qual se insere o *DTX 200 - Emulation de Floppy* [36]. Como os anteriores, o DTX 200, ocupa o espaço equivalente de uma drive de disquetes de 3 ½ polegadas e garante suporte para uma variedade de drives e controladores, incluindo os mais antigos 8 polegadas.

Seguindo o mesmo formato dos emuladores anteriores, o emulador EMUFDD [37, 38] apresenta como características diferenciadoras o uso de software livre na geração das imagens e a especial atenção dada à sensibilidade a erros.

Finalmente, apresenta-se uma solução alternativa que se revela interessante ao prescindir da intrusão no equipamento que se pretende intervencionar. Esta solução recorre a um outro ponto de contacto para efetuar a emulação: as cabeças de leitura e escrita da drive de disquetes.

Tanto a SONY como a JVC produziram disquetes emuladoras que aceitam cartões de memória e fazem de ponte entre os pinos do cartão e a cabeça de leitura magnética da drive de disquetes [39–41]. Esta solução dispensa totalmente a alteração de equipamento no sistema anfitrião ao utilizar apenas disquetes modificadas. Estes discos foram lançados para o grande público para que os utilizadores de câmaras digitais conseguissem aceder aos cartões de memória de forma prática. Para além de não intrusiva, esta solução apresenta-se bastante interessante também do ponto de vista tecnológico: o emulador tem todos os sinais de controlo disponíveis que lhe permitem efetuar as operações de leitura e escrita. A alimentação tem de ser externa e sentir os sinais através da rotação do disco bem como o movimento da cabeça de leitura. Finalmente, o espaço físico disponível para alojar a toda a lógica de emulação é muito reduzido.

No entanto, ao analisar estas soluções, verifica-se que requer a utilização de software especial que corre em Sistemas Operativos de alto nível (Windows e Macintosh). Esta limitação é impeditiva nos sistemas industriais que usam sistemas operativos específicos.

Apesar destas limitações e da acrescida complexidade, poderá ser interessante pesquisar a viabilidade deste tipo de soluções num outro trabalho.

Das alternativas encontradas nenhuma recorre a tecnologia sem fios para carregar a informação no emulador como é proposto neste trabalho.

Capítulo 3

Discos Magnéticos Flexíveis

"A problem clearly stated is a problem half solved."

Dorothea Brande (1893 - 1948)

De forma a apresentar uma implementação funcional, é necessário conhecer o funcionamento – da teoria à prática – do alvo: a drive de disquetes. A tecnologia associada a este equipamento é assim então analisada e interpretada para que o emulador possa substituir a drive de forma totalmente transparente. O funcionamento de uma drive de disquetes, a própria disquete e a comunicação com o controlador são explorados e analisados neste capítulo.

3.1 História

Na evolução das máquinas computacionais, é inevitável ignorar os suportes de gravação de dados de forma permanente ou a longo prazo, especialmente os que se pretendem portáteis [42]. E se hoje em dia é ubíquo o transporte de memórias USB no porta-chaves, houve uma época em que eram discos magnéticos que ocupavam os bolsos das camisas. Apresenta-se nesta secção, uma breve história do aparecimento e evolução dos discos magnéticos de forma a melhor entender o seu funcionamento.

Desde a existência de suportes que codificavam a informação usando propriedades físicas, procurou-se encontrar outras soluções que pudessem, de forma mais eficiente, comunicar com os computadores. Na segunda metade do século XIX, o uso de cartões perfurados foi um dos métodos preferenciais no armazenamento e transferência de informação. Com eles, era possível programar e configurar máquinas tais como teares mecânicos. Foi então de forma quase natural que passaram a ser também usados, como método de introdução de dados, em computadores no advento da sua evolução [43]. No entanto, esta tecnologia era manifestamente limitada e inconveniente face ao crescente volume de dados pois requeria uma interface física ou ótica de grande dimensão para extração de dados que eram limitados pela área física do cartão. Gradualmente, foram adotadas outras soluções nomeadamente o registo em suporte magnético: fio magnético, bobines, cassetes, discos rígidos e disquetes para mencionar apenas alguns. Destes suportes, os representados pelas disquetes ou discos magnéticos flexíveis têm um papel de especial relevo.

Os discos flexíveis foram introduzidos nos finais dos anos 60 como um meio de carregar o código de programas de arranque ou de diagnóstico em servidores. Os primeiros discos tinham grandes dimensões, aproximadamente 20 cm (8 polegadas). Apesar disso a sua capacidade de armazenamento esgotava-se abaixo dos 100 kB e permitiam apenas a operação de leitura uma vez que eram programados durante a sua fabricação. Quando a IBM introduziu o modelo 33FD em 1973, um disco que permitia uma maior densidade e consequente capacidade, introduziu também a novidade de permitir tanto a leitura como escrita [44, 45]. Foi só em 1976 que apareceram as disquetes de 5.25 polegadas chamadas minifloppy [46]. Em comparação com as de 8 polegadas, as de 5.25 polegadas apresentavam como característica inovadora o seu tamanho mais reduzido. De registar que estes discos efetivamente representavam uma regressão em termos de capacidade bruta. Em 1976 os discos de 8 polegadas já apresentavam duas cabeças de leitura para usufruir dos dois lados do disco. Estes discos permitem uma maior capacidade de armazenamento de 568 kB face aos 322 kB dos discos de 5.25 polegadas [44].

A procura por um formato mais prático foi sempre uma preocupação, demonstrada pelo aparecimento em 1984 de um disco ainda mais pequeno de 3 polegadas e meia (3 ½ polegadas). A Figura 3.1 mostra a relação de tamanhos entre as disquetes de 8, 5 ¼ e 3 ½ polegadas. As disquetes de 3 ½ polegadas mantiveram-se como o suporte universal até meados dos anos 90 [47]. Este disco 3 ½ continha um invólucro em plástico rijo de forma a proteger o disco magnético no seu interior. A evolução tecnológica encarregou-se de trazer inovação aos novos discos que facilmente cabiam num bolso: Elevada densidade de escrita, taxa da transferência mais rápida associada a mais altas rotações com a mesma fiabilidade e o uso da modulação MFM (3.3.1). O uso de MFM veio substituir a até então utilizada FM ao permitir uma maior densidade de bits [46, 48, 49].

Com a evolução das disquetes, diversos sistemas operativos poderiam utilizar diferentes formatos de organização de dados, por vezes incompatíveis entre si, apesar de usarem o mesmo suporte físico. Estas diferenças pretendiam aumentar a capacidade do disco através do uso de



Figura 3.1: Disquetes de 8, 5 $\frac{1}{4}$ e 3 $\frac{1}{2}$ polegadas. [50]

formatos e números de faixas diferentes. Outras alterações surgiam da pretensão de dificultar as cópias ilegais tornando os processos de leitura destes discos limitados às aplicações e equipamentos para quais eram desenhados [16, 17].

Sendo um suporte magnético, flexível e exposto ao ambiente os discos são afetados, e degradados, por estímulos externos como campos magnéticos, temperatura, humidade e partículas de pó [24, 51]. Se em condições ótimas de armazenamento as disquetes têm uma vida útil prevista de 5 anos, em utilização em ambientes adversos este valor pode diminuir consideravelmente e exigir a sua substituição numa altura em que estas são descontinuadas [47].

A par dos discos magnéticos flexíveis, existe também outra família de produtos baseados em discos magnéticos rígidos que hoje em dia são utilizados no armazenamento de elevados volumes de informação nos computadores pessoais. Estes discos, por serem rígidos em vez de flexíveis, permitem atingir elevadas velocidades na ordem das 10.000 rpm. Associando a velocidade ao facto de a cabeça de leitura/escrita não tocar na superfície do prato rotativo torna-se possível obter uma maior longevidade. Verifica-se assim, que a tecnologia associada aos discos magnéticos não caiu em desuso, mas sim evoluiu noutras direções. Apesar de hoje em dia o disco flexível estar praticamente abandonado, o disco rígido continua como um dos principais formatos de gravação de dados no mercado doméstico e empresarial.

Este trabalho irá debruçar-se sobre o formato mais comum: as disquetes de 3 $\frac{1}{2}$ polegadas de 720 kB e 1.44 MB. Como alguma da informação sobre as tecnologias usadas nas disquetes de 3 $\frac{1}{2}$ não estava disponível foi, em alguns casos, necessário adaptar a informação extraída de documentos relativos a outros formatos.

Devido à forma como a informação é armazenada e transmitida ser relevante para a solução

proposta neste documento, apresenta-se de seguida uma breve descrição do funcionamento de um disco magnético.

3.2 Disquete e o Disco Magnético

A convencional disquete de 3 1/2 polegadas sobre a qual este trabalho se debruça utiliza, tal como outros suportes magnéticos flexíveis, um disco de plástico boPET (biaxialmente orientado Politereftalato de Etileno) revestido por óxido de ferro ou uma liga de cobalto que lhe atribui propriedades magnéticas [49, 52]. Em comparação com os outros formatos de 5 1/4 e 8 polegadas, este formato apresenta, para além da menor dimensão, e como medida de proteção, uma capa de plástico mais rígida e uma proteção de metal que, ao deslizar, permite que a cabeça de leitura e escrita possa aceder ao disco magnetizado. Este invólucro rígido garante uma maior durabilidade e maneabilidade. No interior da disquete, encontra-se ainda um fino filme de fibra que limpa a superfície do disco contra resíduos e partículas que possam eventualmente ter entrado no interior da disquete.

Considerando que a disquete é um meio portátil, é necessário garantir que possa ser lido e escrito em diferentes equipamentos. De forma a garantir esta propriedade, é necessário formatar a informação nos discos para que possa ser recuperada em diferentes sistemas. Isto é conseguido através da utilização de formatos pré definidos, que definem a forma como a informação é guardada nos discos. Um disco é dividido em faixas (*track* ou *cylinder*). Cada faixa é por sua vez dividida em sectores (*sector*). Diferentes formatos de discos diferem, para além do tamanho, no número de faixas, na quantidade de sectores por faixa e finalmente no tamanho do sector [49, 53]. Um disco pode ainda utilizar as duas faces da sua superfície e assim duplicar o número de faixas disponíveis. Cada face é acedida por uma cabeça de leitura/escrita diferente. A Tabela 3.1 lista alguns destes formatos.

Tabela 3.1: Diversos formatos de disquetes e suas características.

Tamanho	Capacidade	Cabeças	Faixas	Densidade	Sectores	Bytes/Sector
5 1/4	160K	1	40	48 tpi	8	512
	180K	1	40	48 tpi	9	512
	320K	2	40	48 tpi	8	1024
	360K	2	40	48 tpi	9	1024
	1.2M	2	80	96 tpi	15	512
3 1/2	720K (DD)	2	80	135 tpi	9	512
	1.4M (HD)	2	80	135 tpi	18	512
	2.8M (ED)	2	80	135 tpi	36	512

Cada material magnético dependendo da forma como é construído, apresenta diversas propriedades associadas ao campo magnético sendo uma destas características a densidade magnética.

Melhor lógica e maior densidade magnética permitem maiores variações no campo magnético na mesma área física, logo maior densidade de informação. Com a evolução tecnológica, os diversos formatos de disquetes foram progressivamente adaptados de forma a obter uma maior densidade de informação e consequente capacidade e fiabilidade. O número de faixas por polegada suportado foi aumentando ou seja, no mesmo espaço físico, passa a ser possível codificar mais informação. Esta evolução foi acompanhada com a melhoria na sensibilidade e velocidade da lógica utilizada nas operações de manipulação do campo magnético. Do mesmo modo, ao usar um componente magnético com diferentes características, consegue-se uma maior taxa de Bits por Polegada (*bpi*) que permite variações no campo magnético mais próximas umas das outras. Esta propriedade verifica-se facilmente nas disquetes de 4MBytes (2.8MBytes após formatação) que, ao usar um composto de Bário como material magnético, atinge débitos de 1 Mbps [45, 54].

Como apresentado na Figura 3.2, uma faixa é disposta concentricamente ao longo do disco, estando a faixa 0 no rebordo exterior. Separadas por uma distância mínima de forma a evitar interferência entre faixas, estas dispõem-se de forma a ocuparem a área disponível. Em cada faixa, a informação é organizada em sectores. Dependendo da densidade com que os dados podem ser codificados na superfície magnética, o número de sectores e o tamanho destes variam. De notar que estes parâmetros são constantes ao longo do disco apesar da área linear disponível ser menor no interior do disco do que no seu rebordo. Ainda que tenham existido discos que utilizavam velocidades de rotação variáveis, os discos de $3\frac{1}{2}$ polegadas utilizam uma velocidade de rotação constante de 300 rpm. Sendo o número de sectores fixo por faixa, a capacidade do disco é limitada pela máxima quantidade de informação que se consegue guardar na faixa mais interior. Poder-se-ia aproveitar melhor a área disponível ao utilizar velocidades variáveis. No entanto, as drives de disquetes iriam necessitar de uma lógica mais complexa, tanto a nível do motor, como no tratamento de dados; este aumento de complexidade provavelmente não compensaria o ganho de capacidade e assim foi adotada a solução existente, com velocidade de rotação fixa.

Para aceder à informação contida nas disquetes, o controlador indica à drive de disquetes a faixa sobre a qual deve colocar a cabeça de leitura. A drive envia o conteúdo dessa faixa assim que a cabeça de leitura deteta as transições do campo magnético. Os sinais de controlo e sincronismo presentes na faixa, permitem ao controlador detetar o sector que deseja escrever/ler. Os sectores são definidos na formatação do disco. Dependendo do formato, o controlador de disquetes do sistema manipula os sectores através de campos de cabeçalho. Estes campos ajudam a saber onde guardar a informação e se esta foi gravada corretamente através de campos de verificação. O formato dos sectores é discutido na secção 3.4 referente ao formato de dados.

O resto do capítulo irá concentrar-se nas disquetes de $3\frac{1}{2}$ polegadas de Dupla Densidade (DD) de 720 kB e nas de Alta Densidade (HD), 1.44 MB de capacidade IBM format [30]. Em algumas situações, outros sistemas poderão ser referidos para que sirvam de comparação.

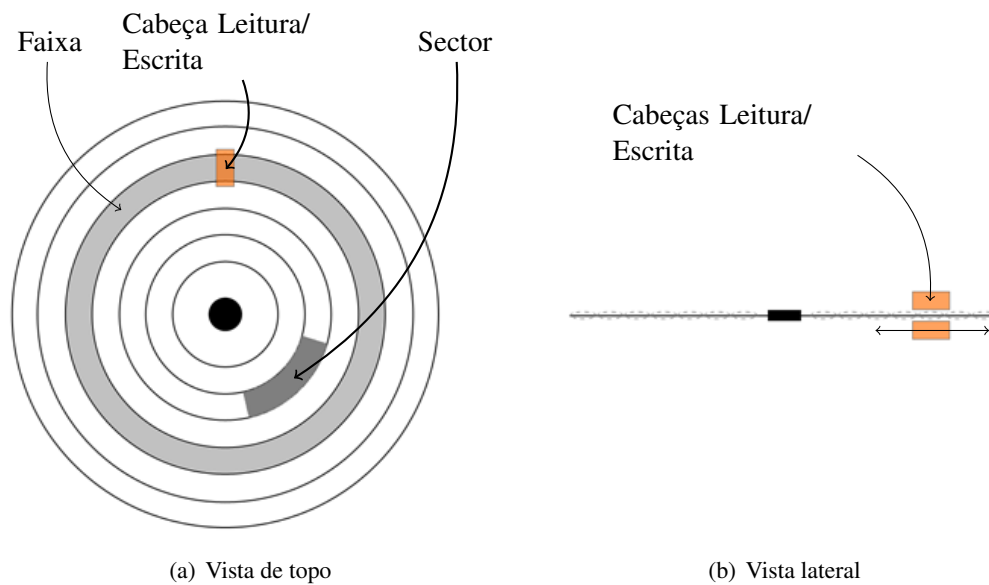


Figura 3.2: Diagrama de um disco magnético e cabeça de leitura/escrita.

3.3 Interface Controlador

A comunicação entre a drive de disquetes de 3 1/2 polegadas e o respetivo controlador é feita por diversos sinais transmitidos num cabo plano de 34 pinos como representado na Figura 3.3 e caracterizado na Tabela 3.2.

Apesar de a informação de dados ser transmitida num formato série, o controlo é realizado através de sinais paralelos e independentes. A lógica que rege o funcionamento de uma drive de disquetes (FDD) é relativamente simples devido à sua natureza principalmente mecânica. Os motores são controlados a partir de instruções dadas pelo controlador no equipamento anfitrião. A drive de disquetes também não executa qualquer interpretação dos sinais de dados. Durante a leitura, deteta as alterações no campo magnético presente nas disquetes e sinaliza-as através de um impulso no sinal de leitura. Efetua a operação inversa, manipulando o campo magnético, quando deteta um impulso no sinal de escrita, na operação de escrita.

O elevado número de pinos presente no cabo é explicado pela presença de sinais de terra em metade dos sinais. A terra presente nos pinos ímpares ajuda a diminuir a interferência por *cross-talking* nos outros sinais. Os pinos de sinal são assim os pinos com numeração par reduzindo de forma considerável a quantidade de sinais que são necessários para interagir. Apesar do envio de dados recorrer a um único sinal em formato série, todos os outros sinais necessitam de ser considerados.

Tabela 3.2: Pinos do cabo de comunicação entre drive de disquetes e controlador (FDC: Controlador de disquetes, FDD: Drive de disquetes)

Pino	Sinais	Direcção FDC – FDD	Descrição
Impares	Terra		
2	/DENSEL	————→	Seleção Densidade
4	N/C		Reservado
6	N/C		Reservado
8	/INDEX	←————	Índice
10	/MOTEA	————→	Ativar Motor Drive A
12	/DRVSB	————→	Selecionar Drive B
14	/DRVSA	————→	Selecionar Drive A
16	/MOTEB	————→	Ativar Motor Drive B
18	/DIR	————→	Selecionar direcção
20	/STEP	————→	Passo cabeça leitura/escrita
22	/WDATA	————→	Escrita de Dados
24	/WGATE	————→	Ativação de modo escrita
26	/TRK00	←————	Faixa 0
28	/WPT	←————	Proteção Escrita
30	/RDATA	←————	Leitura de Dados
32	/SIDE	————→	Seleção de cabeça
34	/DSKCHG	←————	Disco ejetado

Os sinais digitais têm lógica negativa e são ativos a 0 Volts e desativos a 5V. O sinal *Density Select* (/DENSEL) sinaliza a drive de disquetes sobre qual a densidade e consequente débito de dados deve operar. 5V desativa o sinal e habilita a transmissão de dados de alta densidade (500 kbps e 1 Mbps) e 0V ativa o sinal que sinaliza a baixa densidade (300 kbps ou 250 kbps). De notar que o equipamento evoluiu de forma a ter esta informação diretamente na disquete através da realização de furos no invólucro de plástico nas disquetes de 3 ½ polegadas [55]. Juntamente com a evolução da tecnologia que permite a deteção automática destas características, o sinal foi perdendo utilidade e a sua própria função foi sendo alterada dependendo do fabricante da drive de disquetes.

Através de sinais de controlo, o mesmo cabo pode suportar duas drives de disquetes (tipicamente referidas por A e B). Normalmente utiliza-se um cabo que possui alguns dos sinais de controlo (pinos 10 a 16) invertidos para que cada drive seja controlada sem conflitos ou necessidade de configurações adicionais. Cada drive vem de fábrica configurada como sendo uma drive B e assim duas drives, cada uma conectada ao cabo em pontos diferentes, vêm os sinais na sua porta de comunicação como se o controlador tivesse sempre a comunicar com a drive B apesar de este usar sinais diferentes.

Estes sinais, *Motor Enable* (/MOTEx) e *Drive Select* (/DRVSx) indicam à drive de disquetes que deve ligar o motor principal e estar preparada para receber comandos. A drive necessita



Figura 3.3: Cabo de dados com inversão de sinais de seleção de drive e conetor IDC.

de algum tempo para que a rotação do disco estabilize nas 300 rotações por segundo, assim, o controlador pode acionar o sinal de motor antes e só ativar a drive assim que estiver também preparado para receber e enviar informação. A utilização dos dois sinais permite ainda a utilização de duas drives de disquetes de forma simultânea em que os motores são mantidos ativos enquanto que o sinal de seleção de drive é alternado.

O controlador de drive de disquetes (FDC) controla a cabeça de leitura da drive através de 3 sinais que servem para seleccionar qual a faixa pretendida: *Head Step (/STEP)*, *Head Direction (/DIR)* e *Head Select (/SIDE)*. A localização da cabeça de leitura/escrita não é transmitida da drive de disquetes ao controlador a não ser quando está sobre a faixa 0. Neste caso o sinal *Track 0 (/TRK00)* é ativado informando o controlador. A partir daqui, o controlador deverá contar o número de impulsos que envia à drive de disquetes de forma a seleccionar a faixa pretendida. A drive de disquetes move a cabeça de leitura ao longo do raio da superfície da disquete a cada impulso do sinal */STEP* na direção definida pelo sinal */DIR*. Se */DIR* não estiver ativo a cabeça move-se na direção do rebordo do disco em direção à faixa 0. Caso contrário, sobe nas faixas até à faixa 79 no interior do disco. Na Figura 3.4 é visível uma captura da comunicação entre o controlador e a drive de disquetes onde se mostra a sinalização relativa ao movimento da cabeça. Com o sinal */DIR* não ativo, o sinal de */STEP* é repetidamente impulsionado até que o sinal */TRK00* seja ativado.

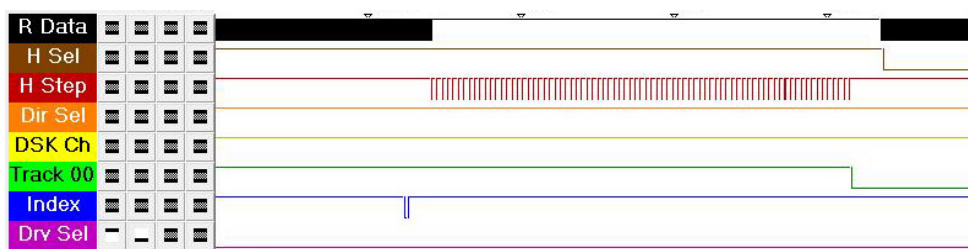


Figura 3.4: Captura de comunicação entre a drive e o controlador de disquetes.

Dependendo do sinal */SIDE*, a cabeça do respetivo lado é ativada e as operações de leitura ou escrita são efetuadas com essa cabeça. Na verdade, a ação de leitura não necessita de qualquer comando por parte do controlador. A cabeça de leitura, ao encostar à superfície do disco, envia a informação que encontra de forma contínua. A cada inversão no campo magnético, é emitido um impulso no pino *Read Data (/RDATA)*. É relegada para o controlador a tarefa de interpretar a informação quando este precisar e estiver disponível. Devido à velocidade de rotação do disco, a informação presente nas faixas é repetida a cada 200 milissegundos. Para ajudar o controlador a sincronizar esta informação, um sinal *Index (/INDEX)* é ativado quando a cabeça de leitura passa sobre o preâmbulo da faixa que precede o primeiro sector.

Caso o controlador de drive da disquetes pretenda ler ou gravar dados mas a drive de disquetes não tenha nenhuma disquete inserida, a drive sinaliza o controlador deste facto através da ativação do sinal *Disk Change (/DSKCHG)*. Se o controlador tentar escrever dados numa disquete que tenha ativada a proteção contra escrita, a drive também sinaliza o controlador através do sinal *Write Protect (/WPT)*.

3.3.1 Codificação MFM

Através da manipulação de corrente elétrica na cabeça de escrita é alterado o campo magnético associado ao disco e assim guardados os dados que compõem a faixa e que irão descrever a informação pretendida. A codificação dos sinais digitais é então descrita.

De notar que é difícil detetar (ler) um campo magnético estático, assim são utilizadas as alterações da polarização no campo magnético presente no disco de forma a identificar sinais lógicos. Um controlador especial consegue assim, a partir das variações detetadas, interpretar uma sequência de sinais binários lógicos. De forma a melhor contornar as limitações que o suporte apresenta, são utilizadas codificações otimizadas. Sendo que os dados são guardados de forma sequencial, a sua interpretação necessita de um sinal de relógio associado. Uma vez que a disquete é um suporte de gravação portátil, esta pode ser usada em diferentes sistemas que geram relógios diferentes. Não se pode então recorrer a um sinal de relógio único e universal para garantir que a

interpretação dos dados, feita na operação de leitura, corresponda à operação de escrita num outro equipamento.

Assim, um dos desafios que a gravação em suporte magnético apresenta é a dificuldade da manutenção de um relógio de sincronização. Apesar da velocidade de rotação do disco se pretender constante, pequenas variações serão inevitáveis e assim este pode ser considerado como fonte de relógio. Longas sequências de sinais iguais apresentam o risco de perder sincronismo. No caso dos discos, este problema é resolvido com a introdução de sinais de sincronização no meio da cadeia eliminando assim o risco de sequências perigosas que levem à interpretação errada da informação. Códigos que apresentam esta característica são geralmente apelidados de RLL (*Run Length Limited*) [56]. Da mesma forma, o material magnético pode apresentar características que dificultem a gravação de mais do que um determinado número de alterações de polaridade por unidade de espaço e, conseqüentemente tempo, devido à rotação do disco. Este facto limita a velocidade e densidade de informação. Um código cuidadosamente planeado consegue contornar e minimizar estas limitações. Existem diversos métodos e códigos que permitem este tipo de manipulação da sequência de informação a gravar nos discos. No caso dos discos flexíveis e de alguns discos rígidos mais antigos utilizavam-se códigos FM (*Frequency Modulation*) na codificação da informação. A codificação FM foi entretanto abandonada em favor da MFM (*Modified FM*) que apresenta maior eficiência [49,57].

Na codificação FM um sinal lógico 1 é codificado com duas alterações no fluxo magnético, e um sinal 0 é codificado com uma alteração seguida de uma não alteração no fluxo. Verifica-se assim que um sinal 1 tem o dobro da frequência que um sinal 0 e daí o nome da codificação – Modulação de Frequência.

Referente às disquetes de 5 ¼ polegadas e 360 kB de capacidade, a velocidade de transmissão (e também de armazenamento) de dados é de 250 kbps. Ou seja, cada bit demora aproximadamente 4 µs a ser transmitido. No caso de um sinal lógico 1, em que são necessárias duas inversões de sinal, implica que não haja mais do que duas inversões de sinal em menos de 2 µs. Como descrito anteriormente, este valor está também associado à frequência máxima de variações no fluxo magnético que o equipamento e o disco conseguem suportar. Com uma codificação mais eficiente consegue-se, respeitando a mesma distância mínima entre variações de fluxo, um maior débito de informação. Surge então a codificação MFM.

Na codificação MFM, cada bit ocupa um espaço temporal fixo: uma célula. Cada célula possui um sinal de relógio e outro de dados. O sinal de relógio apresenta-se no início da célula e o de dados no centro desta. Um sinal lógico 1 é traduzido por uma inflexão de sinal no centro da célula reservada ao bit. Um sinal lógico 0 apresenta duas codificações dependendo do bit que foi transmitido na célula anterior. Caso tenha sido um 1, então não é feita qualquer inversão no fluxo do sinal. Caso tenha sido um 0, então é feita uma inversão de sinal no início da célula. As formas

de onda associadas a estes sinais são desenhadas na Figura 3.5. O sinal de relógio (impulso no início da célula) não é transmitido a não ser que no bit anterior não tenha ocorrido um impulso no sinal de dados.

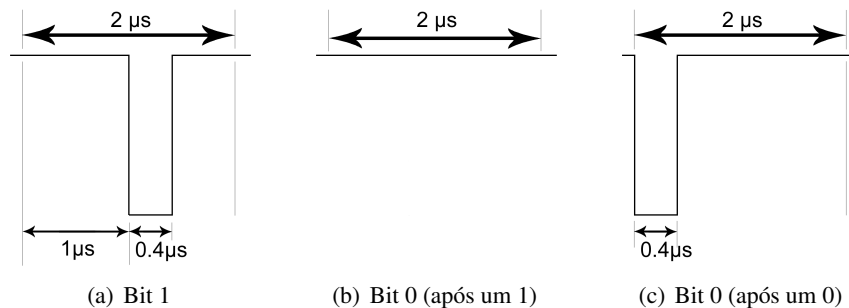


Figura 3.5: Codificação MFM

Com a nova codificação, é possível diminuir o tamanho da célula que cada bit utiliza para metade ($2 \mu s$) e mesmo assim garantir que a distância mínima entre as inversões no material magnético não se altera. O sistema MFM apresenta-se assim mais eficiente ao codificar a mesma informação em metade do tempo, como pode ser verificado na Figura 3.6, onde uma alteração do fluxo é representada através de um impulso de sinal. De notar que a distância mínima entre dois impulsos consecutivos é mantida. Esta codificação garante ainda a existência de um impulso a pelo menos cada 2 bits que dispensa assim a necessidade de um sinal de relógio externo.

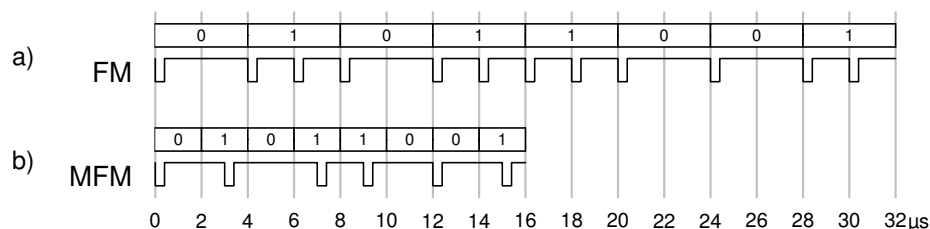


Figura 3.6: Byte 0x59 nas codificações FM e MFM

O uso da codificação MFM foi introduzido pela IMB no standard IBM System 34 e é o utilizado nas disquetes de $3\frac{1}{2}$ de dupla densidade (DD). Este formato é uma evolução do sistema 3740 usado nas disquetes de baixa densidade que ainda recorriam à codificação FM (SD) [27, 30, 58].

3.4 Formato das Faixas e Sectores

Para além da definição do formato de codificação da informação no seu nível mais baixo, também precisa de ser definida a forma como os dados estão organizados para garantir a interoperabilidade das disquetes em diversos meios. Esta informação permite, ao controlador que irá

processar a informação, saber exatamente onde estão os dados do utilizador e se estes se apresentam sem erros.

Em cada faixa, a informação codificada é dividida em várias secções (*sectors*) que são essenciais no controlo e sincronização. Estas secções são definidas nos formatos utilizados. No caso das disquetes de dupla e alta densidade é utilizado o formato IBM System 34 [59, 60].

O emulador implementado não distingue se a informação que é transmitida corresponde a conteúdos de dados ou de controlo, no entanto esta informação é valiosa na deteção de erros e é essencial no conhecimento do funcionamento dos discos magnéticos flexíveis.

Cada face do disco possui então diversas faixas (que no formato em análise são 80) colocadas concentricamente, e cada faixa é composta por diversos campos onde são guardados os dados. Como caracterizado na Tabela 3.3, o primeiro campo é o preâmbulo, aos quais se seguem uma sequência de 18 sectores compostos pelos respetivos campos de identificação (ID) e de Dados. A faixa é terminada com um campo de enchimento (*Gap*).

Preâmbulo	Sector 1		Sector 2		Sector 3		...	Sector 17		Sector 18		Gap
	ID	Dados	ID	Dados	ID	Dados		ID	Dados	ID	Dados	

Tabela 3.3: Organização dos campos numa faixa.

No final do 18º e último sector segue-se novamente o preâmbulo. Devido ao formato circular da disquete e à sua velocidade de rotação, o preâmbulo e os dados são reenviados a cada 200 ms.

Com a inclusão de sinal de relógio no próprio sinal, é necessário detetar o sincronismo e a extração dos dados do sinal. Nos controladores de drives de disquete isto é conseguido através do recurso a *PLL Data Separator* (Separador de Dados PLL) que deteta e extrai os dados relevantes [28, 61, 62].

De forma a facilitar certas operações de escrita e leitura, algumas disquetes podem apresentar a ordem dos sectores alterada. Se, por exemplo, numa operação de leitura o computador anfitrião pretender ler os sectores 1 e 2 mas não tiver capacidade para processar a informação recebida de forma célere, pode pedir para ler o sector 1 e só depois o sector 2. Devido à rotação constante do disco magnético, o computador teria de esperar o tempo correspondente a uma rotação inteira de forma a poder ler o sector seguinte. Com a ordem dos sectores alterada para, por exemplo, 1 8 6 4 2 9 7 5 3, o tempo de espera entre o sector 1 e 2 é diminuído. Esta técnica é apelidada de *interleaving* e entretanto caiu em desuso com o aumento da capacidade de processamento e armazenamento dos computadores. Estes podem assim ler e guardar diversos sectores numa operação de leitura e processar os dados apenas quando lhes for mais conveniente [28].

O formato dos campos que constituem uma faixa será descrito com a ajuda de alguns quadros. Como a maioria dos campos consiste em repetição de dados, os quadros descrevem estas características tal como apresentado na Tabela 3.4. Neste quadro de exemplo, o Campo 1 é composto pela repetição do octeto “Byte” tantas vezes como especificado em “Comprimento”. Segue-se imediatamente o Campo 2 que é composto por outras sequências de bytes com determinadas repetições.

Nome do Campo 1	Comprimento Byte	
Nome do Campo 2	Comprimento Byte	Comprimento Byte

Tabela 3.4: Descrição das características dos campos das faixas.

Segue-se uma descrição dos diversos campos que constituem o formato da faixa, começando pelo Preâmbulo. Os bytes são apresentados na notação hexadecimal sinalizada pelo prefixo *0x*.

Preâmbulo A drive de disquetes envia o conteúdo da faixa de forma cíclica e o controlador necessita então de processar essa informação de forma a poder extrair os dados de utilizador. Isto é conseguido através do reconhecimento de uma série de sinais pré-formatados que indicam o início da faixa. O controlador efetua uma pesquisa por estes sinais e, após obter a sincronização, poderá prosseguir com a extração e verificação.

O preâmbulo é o primeiro destes campos de sinais de controlo e sincronismo que antecede o primeiro sector e marca o início da faixa, sendo constituído por pelos sinais indicados na Tabela 3.5.

GAP4a	80 0x4E	
SYNC	12 0x00	
INDEX MARK	3 0xC2 *	1 0xFC
GAP	50 0x4E	

Tabela 3.5: Campo preâmbulo da faixa.

De forma a permitir que o controlador encontre os dados que pretende, necessita de sincronizar os diversos campos de controlo e de dados. Para diferenciar a sequência de dados do preâmbulo dos dados de utilizador, o INDEX MARK possui um sinal que não respeita a codificação MFM normal. Os sinais de relógio usados para gerar o sinal MFM são modificados gerando assim uma sequência de impulsos que não pode ser encontrada durante o envio do campo de dados [61, 63].

A Figura 3.7 apresenta o octeto 0xC2 quando modulado normalmente no campo de dados e com o sinal de relógio alterado quando utilizado como sinal de sincronismo. De notar a falta do impulso no bit 5, violando assim a modulação normal garantindo que não será confundido com um octeto de dados. Nas tabelas 3.5, 3.6 e 3.7 esta alteração é indicada pela presença do símbolo *.

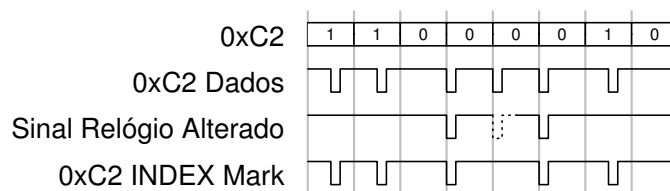


Figura 3.7: Byte 0xC2 quando codificado no campo de dados ou usado com sinal de sincronismo.

Cada sector é então composto por dois campos: Campo de identificação (ID) e campo de Dados.

ID A informação que descreve o sector, como o número do cilindro (ou faixa), a cabeça e o número identificativo deste é guardada no cabeçalho ID. A estrutura destes campos pode ser encontrada na Tabela 3.6. De notar ainda a existência do comprimento do sector. Este valor é calculado através da expressão (3.1).

$$\text{Comprimento Sector} = \log_2(\text{comprimento do sector em bytes}) - 7 \quad (3.1)$$

Num sector com 512 bytes de dados, o campo "Comprimento Sector" é 2.

SYNC	12 0x00	
ID ADD Mark	3 0xA1 *	1 0xFE
Cilindro	1 0xXX	
Cabeça	1 0xXX	
Nr Sector	1 0xXX	
Comprimento Sector	1 0xXX	
CRC	2 0xXX	
GAP2	22 0x4E	

Tabela 3.6: Campo de identificação do sector.

Neste campo, mais propriamente no parâmetro *ID Address Mark*, também é utilizada a violação no sinal de relógio MFM de forma a que este possa ser distinguido dos octetos de dados.

Dados É neste campo que são guardados os dados de utilizador. É precedida de campo de sinalização que também contém um sinal que viola a codificação MFM e é finalizada por um campo de controlo de erros (CRC) como pode ser consultado na Tabela 3.7.

SYNC	12 0x00	
Data Mark	3 0xA1 *	1 0xFB
Data	512 0xXX	
CRC	2 0xXX	
GAP3	84 0x4E	

Tabela 3.7: Campo de dados do sector.

Enchimento (GAP) Após os 18 sectores o espaço existente até ao final da faixa é ocupado por um campo de enchimento com 510 octetos de comprimento representado na Tabela 3.8.

GAP4b	510 0x4E
-------	-------------

Tabela 3.8: Campo de enchimento.

Devido à propriedade cíclica do disco e, consequentemente, do envio dos dados, o campo GAP4a do preâmbulo surge após o GAP4b do campo de enchimento.

No total, cada faixa, alberga mais do que 18 campos de dados com 512 bytes cada. Com o preâmbulo e os sinais de controlo dos sectores, uma faixa totaliza 12500 bytes tal como pode ser consultado na Tabela 3.9. Como cada byte demora 16 µs a ser transmitido, o ciclo repete-se a cada 200 ms.

	18 ×			
Preâmbulo	Campo Identificação	Campo Dados	Preenchimento	Total
146	44	614	510	12500

Tabela 3.9: Comprimento total de uma faixa.

3.4.1 CRC

De forma a garantir a validade da informação contra erros de escrita ou leitura, são utilizados campos de verificação de erros CRC (*Cyclic Redundancy Check*). Estes campos estão presentes tanto nos campos de identificação do sector como nos campos de dados. O algoritmo utilizado para cálculo de CRC é conhecido como CRC-16-CCITT. O polinómio gerador $g(x)$ é apresentado na Equação (3.2) e utiliza como semente a palavra 0xFFFF [64, 65].

$$g(x) = 1 + x^5 + x^{12} + x^{16} \quad (3.2)$$

A sequência de dados a serem protegidos é seguida imediatamente pelo código CRC. Deste modo, o cálculo polinomial dos dados mais a sequência de CRC resultará num resultado composto por zeros. Caso este resultado não seja zero, pode então concluir-se que existiu um erro na transmissão dos dados. Como exemplo, o código de verificação de um Campo de Identificação de um sector pode consistir da seguinte sequência representada na Tabela 3.10, constituída pelos campos de sincronismo, identificação e correção de erros.

ID Add Mark				Cilindro	Cabeça	Sector	Comprimento	CRC	
0xA1	0xA1	0xA1	0xFE	0x00	0x00	0x01	0x02	0xCA	0x6F

Tabela 3.10: Campo de Identificação de um sector

3.4.2 Campo de Dados

Para que a informação seja útil e facilmente acessível ao utilizador, esta tem de ser também entendida pelo sistema operativo do computador que opera a disquete. Para isso, a disquete é formatada com um sistema de ficheiros de dados. O sistema de ficheiros utilizados depende do computador e ambiente em que as disquetes são utilizadas. Os dados que perfazem esta formatação são incluídos nas secções do campo de dados fazendo com que a capacidade da disquete seja diminuída. Devido ao tamanho diminuto das disquetes, são também utilizados sistemas de ficheiros que não ocupam um volume grande de dados. No caso dos computadores pessoais com os sistemas operativos DOS ou Windows, é utilizado o sistema de formatação FAT12. Este sistema de ficheiros, apesar de ter um tamanho pequeno, ocupa alguns dos sectores de dados reduzindo a capacidade efetiva para ficheiros de utilizador [66]. No total, dos 2 MB de dados escritos na disquetes, apenas 1.44 MB estão disponíveis.

Capítulo 4

Interface Sem Fios

Uma das características do emulador que o diferencia dos restantes produtos existentes, é a capacidade de carregar os dados necessários à emulação da disquete através de um canal de comunicação sem fios. Foram analisadas algumas das tecnologias existentes e descreve-se a utilizada: ZigBee, implementada através da plataforma Z-Stack.

O uso de tecnologias sem fios está fortemente disseminado nos dias de hoje. É uma das áreas com maior desenvolvimento nos últimos anos e a sua penetração no dia-a-dia de forma transparente é fascinante. Pretende-se trazer algumas das suas vantagens para esta aplicação, reconvertendo sistemas que utilizam tecnologia datada e obsoleta em tecnologia moderna.

Existem várias vantagens na utilização de uma rede sem fios para transmissão dos dados. De forma imediata, deixa de ser necessária a presença física de um operador para transferir a informação até ao terminal.

4.1 Tecnologias de Comunicação Sem Fios

Existe um vasto leque de tecnologias de comunicação sem fios que podem ser utilizadas, desde módulos Bluetooth e WiFi passando por transceptores RF de mais baixo nível.

Ao considerar o tipo de aplicação que se deseja obter, um dos fatores a ter em consideração é a escolha da tecnologia mais apropriada. Não havendo necessidade de configurações ou características especiais, é recomendável o uso de tecnologias já testadas e com ferramentas de desenvolvimento facilmente acessíveis.

São relevantes duas famílias de comunicações sem fio: as redes WPAN e as redes Wi-Fi. No âmbito de redes WPAN (*Wireless Personal Area Network*) são exemplos os protocolos Bluetooth, ZigBee e MiWi. Estas redes são utilizadas na comunicação entre equipamentos que formam uma rede próxima e fechada. Não requerem infraestruturas exteriores para formar a rede uma vez que cada elemento pode também ter um papel na construção da rede. Aconselhadas para sistemas de baixo custo, baixo consumo e potencialmente portáteis, têm curto alcance e são orientadas para a interligação de equipamentos periféricos [67].

Uma das vantagens na utilização destas redes está relacionada com a existência de módulos com controladores e transmissores integrados. A utilização destes módulos, juntamente com uma API apropriada, facilita o desenvolvimento da aplicação ao dispensar a transmissão da informação entre o controlador e o módulo rádio, que pode diminuir a velocidade de transmissão e aumentar a complexidade e custo da solução.

Como referência, dão-se como exemplo os módulos da Texas Instruments que implementam num único circuito integrado (CI) um microcontrolador, memória, acesso a periféricos e até circuitos dedicados à comunicação rádio. As famílias SoC CC2430/CC2530 são exemplos destes módulos nos protocolos ZigBee enquanto que a família CC2540 é dedicada à comunicação Bluetooth [68].

A existência de módulos num único CI, que também integra a gestão de protocolo, faz diminuir as áreas de código e memória disponíveis para a aplicação do utilizador e acrescenta dados adicionais à transmissão. Dependendo das funcionalidades pretendidas, diferentes arquiteturas que tenham um impacto menor podem ser utilizadas.

Criada pelo Bluetooth SIG, a especificação Bluetooth é orientada para comunicação sem fios entre periféricos a curto alcance. Indicada sobretudo para a interligação de equipamentos pessoais, apresenta uma série de perfis no sentido de facilitar a interoperabilidade entre equipamentos de diferentes fabricantes. Nas suas sucessivas revisões, menor consumos e maiores débitos são consecutivamente conseguidos. Na revisão 2.1+EDR publicada em 2007, débitos na ordem dos 2 Mbps são obtidos [69]. A partir da revisão 3.0, com o auxílio de ligações baseadas em tecnologia 802.11, são obtidos débitos ainda mais elevados na ordem dos 24 Mbps. Apesar de orientada para a comunicação de curto alcance, a especificação Bluetooth também prevê a instalação em ambientes industriais com um alcance máximo de 100 metros [70].

Alternativamente, aparecem as redes Wi-Fi 802.11 que, nas suas diversas revisões, apresentam débitos mais elevados quando comparadas com as WPAN. O maior alcance associado à maior potência tornam o uso destas redes mais apetecível. No entanto, estas redes necessitam de equipamentos dedicados na sua formação como encaminhadores e pontos de acesso. Apesar de serem redes com manifesta penetração, a sua implementação de raiz pode requerer esforços extra na construção de uma rede complexa que poderá ficar sobre-dimensionada.

Na Tabela 4.1, resumem-se as principais características das redes mencionadas.

Tabela 4.1: Comparação de sistemas de comunicação sem fios

	ZigBee	Wi-Fi	Bluetooth
Bandas	2.4 GHz	2.4 GHz	2.4 GHz
Débito	250 kbps	Até 54 Mbps	Até 2 Mbps
Topologias de Rede mais comuns	Teia, Estrela, Ponto a Ponto	Estrela, com recurso a distribuidores	Ponto a Ponto
Utilizações Típicas	Controlo, Sensores, Automação	Internet, Vídeo	Substituição de cabos em curto alcance

4.2 Norma ZigBee

A norma ZigBee, criada pelo consórcio ZigBee Alliance, define um protocolo de comunicações sem fios que apresenta como fatores diferenciadores o baixo consumo e baixo custo. Principalmente destinada a interligar redes de sensores e controlo, estas redes encontram-se implementadas numa vasta gama de aplicações desde automação e segurança doméstica a equipamento médico.

Baseada no standard rádio IEEE 802.15.4 [71], a norma ZigBee pode utilizar as diversas frequências não licenciadas nas gamas 2.4GHz (global), 915Mhz (Américas) e 868Mhz (Europa). Se se usar uma potência baixa, o uso destas gamas de frequências não requer licenciamento por se situar na banda ISM (*Industrial, Scientific and Medical*). Aliando uma solução económica à existência de equipamentos com transdutores integrados com linguagens de alto nível, é relativamente fácil e acessível a sua implementação.

A especificação ZigBee é então construída em cima das especificações criadas pelo grupo 802.15 do IEEE que publica as duas camadas de mais baixo nível no protocolo, enquanto que a ZigBee Standards Organization especifica as três camadas superiores da arquitetura como apresentado no esquema da Figura 4.1.

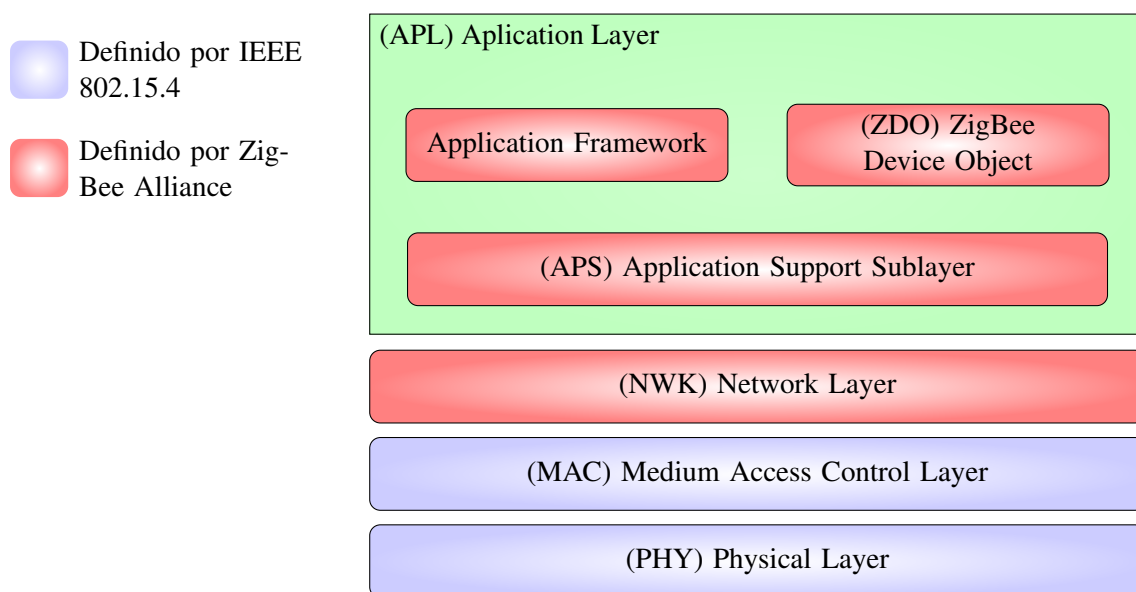


Figura 4.1: Esquema simplificado da arquitetura da pilha ZigBee criada em cima da norma IEEE 802.15.4

As duas camadas mais baixas do protocolo, a camada física e a camada de controlo de acesso ao meio fornecem a base da comunicação rádio. A camada física, PHY (*Physical*) é a responsável pelo envio e receção dos dados através da modulação dos sinais rádio. O protocolo pode utilizar duas bandas de frequências: 2.4 GHz e sub-GHz. A primeira é mais rápida, com um débito de 250 kbps, e está dividida em 16 canais que podem ser utilizados de forma a minimizar as interferências. Nas frequências sub-GHz, 915 MHz (Estados Unidos e Austrália) ou 868 MHz (Europa) os débitos são mais baixos não atingindo os 50 kbps. Naturalmente, o uso dos 2.4 GHz pode ser mais apetecível pela sua rapidez e universalidade, mas esta frequência também é utilizada por outros protocolos rádio como Wi-Fi ou Bluetooth o que pode gerar interferências. Por cima desta camada de baixo nível é especificada a camada MAC (*Medium Access Control*), responsável pela sincronização rádio e por garantir que a transmissão rádio ocorre sem erros. A descrição técnica destas camadas pode ser consultada em documentação publicada pelo IEEE [67].

As especificações da norma ZigBee acrescentam diversas camadas no topo do standard referido e já implementam muitas das funcionalidades de gestão das mensagens, incluindo a utilização de *endpoints* e *clusters*, que são usados como identificadores dos comandos e serviços que suportam. Estas funcionalidades de gestão, implementadas na NWK (*Network Layer*) [71], libertam o programador para se concentrar na sua aplicação e não em detalhes técnicos de gestão da rede ou mesmo da interface rádio e de gestão de dispositivos.

Uma das características das redes ZigBee é a possibilidade de configuração em malha, possibilitando assim a comunicação entre dois dispositivos que não estejam ligados entre si diretamente.

Esta solução permite a instalação destas redes em locais onde os terminais extremos de uma rede estão a uma distancia que não permita a comunicação direta entre eles como demonstrado na Figura 4.2.

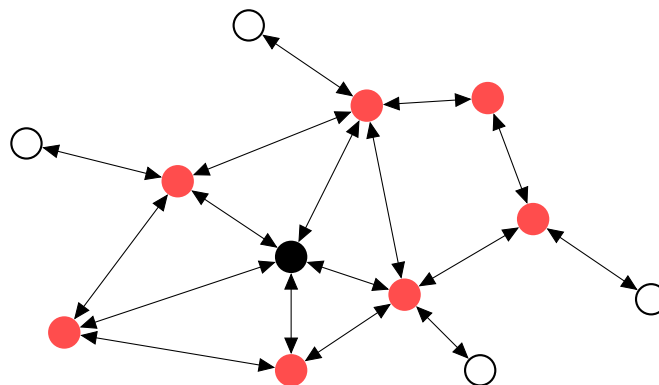


Figura 4.2: Exemplo de configuração de uma rede ZigBee com Coordenador ●, Encaminhadores ● e Ponto Terminal ○.

A tecnologia ZigBee permite estas configurações devido à sua estrutura dinâmica e a existência de um dispositivo *ZigBee Coordinator* (Coordenador) que coordena toda a rede. Este Coordenador inicia a rede e permite, consoante a política definida pelo gestor da rede, que outros dispositivos se liguem a ele. Existe apenas um Coordenador em cada rede.

A rede pode ser expandida através de outros dois tipos de módulos: *ZigBee Router* (Encaminhador) e *ZigBee EndPoint* (Ponto Terminal). O primeiro, auxilia o coordenador na expansão da rede, atribuindo identificadores a novos nodos e pode ainda decidir qual a rota mais apropriada para a comunicação entre dois dispositivos. O outro tipo de dispositivo, *EndPoint* (Ponto Terminal), não tem responsabilidades de gestão de rede e assim prescinde da sua ligação constante. O *EndPoint* é adequado a equipamento que use como fonte de alimentação baterias pois permite entrar em estados de baixo consumo e acordar periodicamente para comunicar com os outros elementos da rede. De notar que um equipamento *EndPoint* apenas se liga à rede por um ponto, ao contrário dos *Routers* que podem estabelecer várias pontes de comunicação.

Assim, se um dos pontos de comunicação falhar, o sistema procura um caminho alternativo para encaminhar as mensagens.

Apesar de ser necessária a existência de um Coordenador para a formação da rede, não existe limitação, a nível físico, que diferencie os tipos de dispositivos. Assim, se um Ponto Terminal não conseguir ligar-se a uma rede, poderá ele próprio formar uma nova rede e tornar-se um *ZigBee Coordinator* pois esta seleção é feita por software. Sendo os próprios equipamentos responsáveis pela gestão da rede, dispensa-se a instalação de equipamento dedicado a esta função.

De forma a facilitar o desenvolvimento destas aplicações, existem plataformas de software sobre as quais se podem criar as aplicações do utilizador. Estas plataformas de programação são responsáveis pelas funções de gestão de rede e ainda fornecem ao utilizador um conjunto de interfaces de acesso aos periféricos. Uma destas plataformas que implementam as especificações ZigBee é a Z-Stack, que é utilizada pelos SoC CC2430 usados neste projeto.

4.3 Z-Stack

A *Z-Stack™* (*ZigBee Protocol Stack*) é uma plataforma de software, baseada na norma IEEE 802.15.4 que segue as especificações indicadas no protocolo ZigBee, fornecida pela Texas Instruments para a sua família de circuitos integrados CC2430 e similares. Esta plataforma, ao seguir as normas ZigBee, implementa diversas configurações dos perfis necessárias à comunicação. Estas configurações, recomendadas pela *ZigBee Alliance*, promovem a interoperabilidade e possibilitam assim a utilização de equipamentos de outros fabricantes desde que sigam a mesma norma e configurações.

A *Z-Stack* implementa toda a lógica necessária para gestão das mensagens e dos papéis de coordenador e de *router* dos elementos da rede nas camadas NWK e ZDO como representadas na Figura 4.1. Implementa ainda, para além das interfaces de comunicação entre os módulos, também ferramentas de gestão da aplicação e de acesso aos periféricos como portas UART, memória não-volátil ou temporizadores através de um *OS Abstraction Layer* [72, 73]. Usando estas ferramentas, o desenvolvimento da aplicação fica então bastante simplificado

4.4 Mensagens ZigBee

A comunicação entre os diferentes módulos que compõe a rede é realizada através de mensagens rádio trocadas pelos dispositivos. É através destas mensagens que é feita tanto a gestão da rede, como a troca de dados do utilizador. Com o uso de plataformas que fazem a gestão da rede, como a referida Z-Stack, a ocorrência destas mensagens de controlo é totalmente transparente para o utilizador.

Sendo o protocolo usado para controlo de dispositivos distintos entre si, a ZigBee Alliance procurou fomentar a interoperabilidade dos equipamentos através da definição de conjuntos de mensagens que transmitem as mesmas intenções. Isto é feito através de perfis e de *clusters* que podem ser usados de forma a adaptar-se às diversas utilizações. Com esta funcionalidade, um programador pode facilmente transmitir comandos entre os dispositivos sem ter que desenhar e

implementar um protocolo de comunicação de raiz e que ficaria limitado à sua aplicação. No caso da Z-Stack, estas funcionalidade estão acessíveis ao programador através das interfaces de abstração [72, 74]. Estas camadas de abstração expõem funções de gestão de rede dos ZDO (*ZigBee Device Objects*) que são os elementos base dos dispositivos de rede ZigBee. As funções de gestão de rede permitem obter e manipular a informação relativa aos dados da rede tal como pesquisar por dispositivos, obter os endereços pelos quais podem ser referenciados e iniciar ligações entre estes.

Estas operações efetuadas pela Z-Stack de forma a implementar os ZDO são feitas em cima das especificações dos protocolos ZigBee e consequentemente também do IEEE 802.15.4. Cada uma das camadas apoia-se na anterior e aproveita os campos de dados de cada uma das interfaces para transportar os dados que necessita. O campo de transporte da trama, definida pela camada física PHY, é então ocupado pela trama definida pela camada MAC como representado na Figura 4.3.

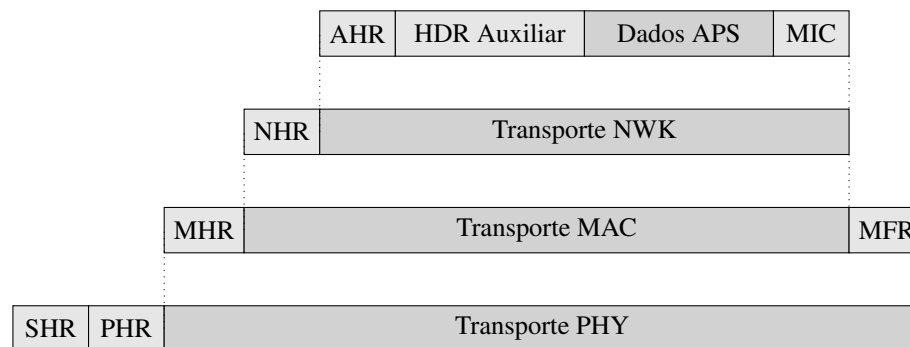


Figura 4.3: Esquema de transporte de dados dos pacotes nas diversas camadas que compõem o protocolo ZigBee [75]

A diversidade de camadas e sinais de controlo usados faz diminuir o número de bytes disponíveis para as camadas superiores como, por exemplo, a de aplicação do utilizador. No caso da plataforma Z-Stack utilizada, o tamanho do campo de dados disponível é de 85 bytes em cada mensagem.

Capítulo 5

Desenvolvimento e Implementação

Após a análise do funcionamento e dos requisitos necessários para a implementação do projeto, prossegue-se ao seu planeamento e implementação em protótipo funcional. Não havendo, naturalmente, uma solução única, são analisadas as tecnologias utilizadas e consideradas algumas das suas características na forma como se adequam a este projeto. Neste capítulo, serão expostos os desafios encontrados durante o desenvolvimento e as soluções adotadas para os superar.

De forma a poder substituir a drive de disquetes, o projeto propõe a construção de um dispositivo emulador que interaja, por um lado, com o controlador de drive de disquetes em equipamentos compatíveis, por outro, com dispositivos que façam a gestão e envio da informação necessária para a emulação. O diagrama geral pode ser consultado na Figura 5.1. Neste diagrama um Gestor remoto envia informação para a memória de armazenamento do Emulador através de um Coordenador. A informação é transmitida num formato de imagem de disco dedicado à emulação que será diretamente usada pelo Emulador na comunicação com o equipamento anfitrião. É objetivo do projeto o desenho, construção e programação destes dispositivos e das respetivas ferramentas de teste e análise.

5.1 Plataforma e Ferramentas de Desenvolvimento

Na implementação deste projeto, dois fatores chave necessitam de ser analisados: a tecnologia usada na transmissão sem fios e o microcontrolador utilizado para efetuar a emulação e controlo

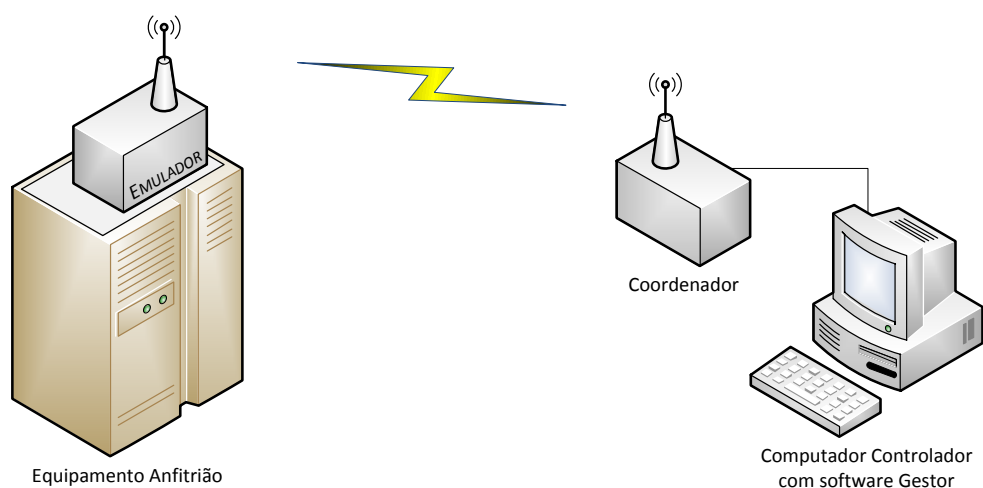


Figura 5.1: Diagrama geral.

geral da aplicação. De forma a efetuar esta escolha, diversos elementos têm de ser tomados em consideração de forma a conseguir realizar o projeto com sucesso. Estes elementos, nomeadamente custo, disponibilidade, facilidade de utilização, etc, irão condicionar a escolha dos fatores chave mencionados. Como referido em 4.1, existe uma variedade de microcontroladores que integram componentes de RF e que possuem plataformas de desenvolvimento de software de forma a facilitar o desenvolvimento. Assim, mais do que a escolha de um microcontrolador e uma tecnologia sem fios, é necessário também ter em consideração todo o ambiente de desenvolvimento necessário para a sua programação e desenvolvimento. Estes ambientes de desenvolvimento podem ser complexos e/ou caros pelo que é importante considerar o seu custo e tempo de aquisição no desenvolvimento de qualquer projeto. A própria experiência do executante do projeto com o ambiente de desenvolvimento é fundamental para evitar que demasiado tempo seja desperdiçado na sua ambientação. Face às diversas possibilidades, deu-se preferência a plataformas já conhecidas no seio do DEEC (Departamento de Engenharia Eletrotécnica e de Computadores) da Faculdade de Engenharia da Universidade do Porto. Foi então escolhida como plataforma de desenvolvimento a família RC230x da Radiocrafts, mais precisamente o RC2300 que integra um microcontrolador baseado em 8051 e um módulo rádio com interface ZigBee. De salientar que a escolha, apesar de conveniente, não condiciona o desenvolvimento, nem sequer apresenta desvantagens significativas face às alternativas consideradas.

5.1.1 RC2300

O módulo RC2300 [76], apresentado na Figura 5.2, pertence a um família de módulos compactos equipados com um microcontrolador e emissor/recetor de radiofrequência que respeita a norma ZigBee. É um módulo que encapsula toda a lógica necessária e permite criar soluções com um muito reduzido número de componentes extra. Baseado no *System-on-Chip* CC2430 da

Texas Instruments, complementa este microcontrolador com uma implementação que reduz a necessidade de material e conceção extra ao já estar equipado com cristais de relógio e uma antena embutida. O desenvolvimento do projeto, nomeadamente no desenho dos circuitos eletrónicos, é assim simplificado.



Figura 5.2: Módulo RC2300 da Radiocrafts [77].

De forma a verificar que o microcontrolador é adequado à aplicação em causa, são analisadas as suas propriedades, das quais as mais relevantes são destacadas [76]:

- Baseado no IC CC2430 da Texas Instruments;
- Antena Integrada;
- Tamanho reduzido: 12.7 x 25.4 x 2.5 mm;
- 128 kB Memória Flash, 8kB SRAM;
- 19 Pinos I/O Analógico/Digital;
- Interface UART, SPI e Debug;
- Relógio 32.768Mhz;
- Baixos requisitos de potência (< 100 mW).

Apesar destes módulos possuírem uma antena integrada, existe a opção da utilização de uma antena externa que aumentará o alcance das transmissões. O módulo RC2300 é baseado na solução integrada CC2430 da Texas Instruments CC2430 que reúne num único integrado um microcontrolador baseado em 8051 e um transceptor ZigBee.

No sentido de verificar a adequação do microcontrolador à manipulação da variedade de sinais necessários à emulação, foi consultada a folha de características fornecida pelo fabricante e analisado o seu *pinout*. Nesta análise, feita através da Tabela 5.1, regista-se que alguns dos pinos possuem múltiplas funcionalidades e assim nem todos estarão disponíveis para a interface. Verifica-se assim a existência de duas portas P0 e P1 de 8 pinos cada e ainda de 3 pinos na porta P2 disponíveis para acesso direto. Regista-se ainda a existência de portas UART e SPI.

No caso do Emulador, como referido na secção 3.3, existe um requisito associado ao número de sinais que são operados pelo controlador e pela drive de disquetes. Estes sinais requerem

Tabela 5.1: *Pinout* RC2300 [76]

Pino	Nome	Descrição
1	GND	Terra
2	VCC	Alimentação
3	P2_2	Relógio Depuração, P2.2. Canal de depuração usado na programação.
4	P2_1	Dados Depuração, P2.1. Canal de depuração usado na programação.
5	P2_0	E/S Digital, P2.0
6	RESET_N	Reset. Ativo baixo com pull-up interno.
7	GND	Terra
8	P0_0	E/S Digital ou analógica, P0.0
9	P0_1	E/S Digital ou analógica, P0.1
10	P0_2	E/S Digital ou analógica, P0.2, I2C SDA
11	P0_3	E/S Digital ou analógica, P0.3, I2C SCL
12	P0_4	E/S Digital ou analógica, P0.4
13	P0_5	E/S Digital ou analógica, P0.5
14	P0_6	E/S Digital ou analógica, P0.6
15	P0_7	E/S Digital ou analógica, P0.7
16	RXTX	Saída Digital, sinal de controlo RX/TX.
17	NC	Não Ligado. Reservado para uso futuro.
18	GND	Terra
19	GND	Terra
20	32kHz_Q2	Oscilador 32 kHz interno. Não ligar. (P2.4)
21	32kHz_Q1	Oscilador 32 kHz interno. Não ligar. (P2.3)
22	P1_7	E/S Digital, P1.7 / SPI
23	P1_6	E/S Digital, P1.6 / SPI
24	P1_5	E/S Digital, P1.5 / SPI
25	P1_4	E/S Digital, P1.4
26	P1_3	E/S Digital, P1.3
27	P1_2	E/S Digital, P1.2
28	P1_1	E/S Digital, P1.1
29	P1_0	E/S Digital, P1.0
30	GND	Terra
31	RF	Ligação E/S a antena externa, 50 Ohm. Não ligar se usar antena interna.
32	GND	Terra

então que o microprocessador apresente pinos de I/O controláveis em número suficiente tanto para processar os pedidos enviados pelo controlador de disquetes como para fornecer os dados da disquete que se pretende emular. A existência das diversas portas de I/O, juntamente com um planeamento cuidadoso dos sinais a processar, permitem que este microcontrolador interaja com os sinais pretendidos sem necessitar de lógica mais complexa. A presença de portas de comunicação SPI e UART permitem a comunicação com memórias e outros periféricos.

De forma a auxiliar o desenvolvimento de aplicações que usem estes módulos, pode-se recorrer a um conjunto de desenvolvimento CC2431DK [78]. Esta ferramenta, fornecida pela Texas Instruments, torna a tarefa de programação, depuração, prototipagem e teste das soluções muito mais simples ao fornecer placas de desenvolvimento e de acesso direto aos pinos (*breakout*) do microcontrolador. Sendo o RC2300 baseado no CC2430, é totalmente compatível com as ferramentas de desenvolvimento que também se encontram disponíveis no seio do DEEC.

A simplicidade das ferramentas e disponibilidade do equipamento, associado às características técnicas que o integram, fazem com que a combinação RC2300/CC2431DK seja relevante como solução para o projeto. Este microcontrolador será a base tanto do Controlador/gestor remoto como do Emulador.

5.1.2 Kit de Desenvolvimento CC2431DK

De forma a facilitar o desenvolvimento de software e iniciar a sua implementação (antes de ser necessária a criação dos módulos físicos baseados em RC2300), utilizou-se, como referido, o kit de desenvolvimento da Texas Instruments CC2431DK. Este kit é composto por duas placas de desenvolvimento SmartRF® 04EB, 10 módulos de extensão SoC_BB, 10 módulos CC2431EM e 2 módulos CC2430EM com as respetivas antenas. As placas SmartRF04EB são ligadas ao computador programador via USB e dão acesso direto a uma série de periféricos, como um visor de cristais líquidos, porta série, botões e leds de interface entre outros [78]. Estas funcionalidades, juntamente com a depuração no ambiente de desenvolvimento, permitem acelerar a escrita do código e a deteção de erros de forma significativa.

As placas SoC_BB fornecem uma interface aos módulos EM de forma móvel. Esta placa fornece a alimentação e fácil acesso a pinos de expansão que conectam às portas e sinais dos módulos rádio CC2430EM e CC2431EM. As placas contêm ainda um interruptor para ligar a alimentação, 2 conetores IDC que permitem total acesso aos pinos do microcontrolador, um botão de entrada, um indicador LED e um conector de interface de programação JTAG. A alimentação é fornecida através de duas pilhas AA, o que as torna no sistema ideal para testar a solução em mobilidade. Também é possível alimentar os sistemas baseados nas placas SoC_BB através de programadores USB preparados para o efeito.

A maior parte do desenvolvimento do Emulador foi efetuado recorrendo às placas SoC_BB com os módulos CC2431EM e CC2430M. O ambiente de desenvolvimento utilizado para a programação e compilação do código *firmware* dos microcontroladores é o IAR Embedded Workbench IDE 7.30A for 8051¹.

5.1.3 Equipamento Auxiliar

Para efetuar a programação dos microcontroladores nas placas SoC_BB, recorreu-se a um programador dedicado. O FlashPro-CC² da Elprotronic Inc. é um programador USB que suporta diversos microcontroladores, entre eles estão os da família CCx da Chipcon/Texas Instruments. Através de um adaptador e software dedicados, o programador é ligado ao conector de programação da placa SoC_BB e é descarregado o novo software para a memória interna do módulo. O programador pode ainda realizar outras funcionalidades como a programação do endereço IEEE do módulo, a operação de apagar os dados da memória interna e realizar a verificação de *Check-Sum* da memória. Esta comunicação e configurações é realizada através dos pinos de programação e depuração existentes nos módulos.

O FlashPro-CC permite também a ligação a um software de desenvolvimento como o IAR Embedded Workbench e efetuar a programação e depuração do microcontrolador diretamente a partir do ambiente de desenvolvimento. No entanto, algumas dificuldades foram encontradas neste modo: o software parava de responder e o programador deixava de ser reconhecido pelo computador. Esta situação levava à necessidade de reiniciar o computador de modo a que o programador fosse novamente reconhecido e, conseqüentemente, reatado o desenvolvimento. Face a estas contrariedades, optou-se por apenas compilar o código num ficheiro de programação e envia-lo via programador usando o software fornecido com o equipamento. Procedimento mais lento e trabalhoso mas que se verificou mais fiável.

No desenvolvimento do projeto, foram ainda utilizadas mais duas ferramentas: Um analisador lógico USBee SX da CWAV, Inc. e um adaptador Bus Pirate da Dangerous Prototypes.

O USBee SX [79] é um analisador lógico USB. Este tipo de analisadores lógicos necessita de estar ligados a um computador para efetuar a captura dos dados uma vez que não apresentam, geralmente, memória interna para os guardar nem instrumentos autónomos para os visualizar. Por outro lado, a interligação com um computador permite uma maior facilidade na manipulação e interpretação dos dados obtidos.

¹Mais informações sobre este ambiente de desenvolvimento pode ser encontrado no website da IAR Systems: <http://www.iar.com>.

²Mais informação sobre o programador FlashPro-CC pode ser obtida no website do seu fabricante em <http://www.elprotronic.com/flashproCC.html>.

As imagens das capturas presentes neste documento, foram realizadas usando o USBee SX que apresenta as seguintes características técnicas [79]:

- Amostragem máxima: 24 Msps;
- 8 Canais;
- Comunicação via USB 2.0;
- Disparo por software ou sinal externo;
- Compacto (5.3 x 3.3 x 1,27 cm);
- Gerador de Sinais.

Para este equipamento existem dois programas que podem ser utilizados na captura e interpretação dos sinais lógicos. Ambos permitem a análise das formas de onda dos sinais capturados e a decodificação dos protocolos SPI e Série. O *USBee SX Logic Analyzer* é uma aplicação mais antiga e com menos funcionalidades mas que permite a captura de sessões mais longas que o *USBee Suite*. No entanto, este último apresenta uma maior variedade de sinais que podem ser decodificados e tem uma interface mais amigável que permite visualizar os dados decodificados diretamente na forma de onda.

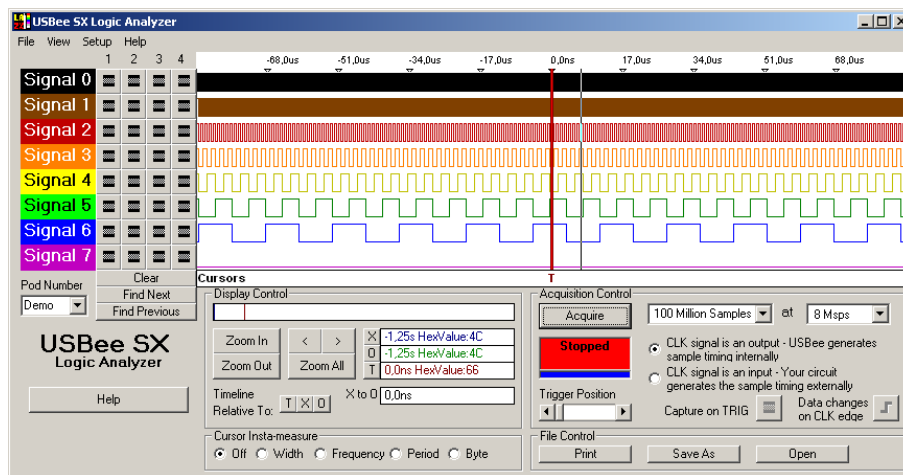
Foi ainda utilizada a funcionalidade de exportação das capturas do *USBee SX Logic Analyzer* para um formato próprio denominado de ULB [80]. Estas capturas são utilizadas na análise e verificação dos testes de transmissão sem haver a necessidade de os fazer em tempo real.

Por último, outro equipamento utilizado no desenvolvimento do projeto foi o *Bus Pirate* [81]. Uma ferramenta livre pois tanto o código fonte como o hardware estão licenciados sob a licença CC0 *Creative Commons No Copyright - Public Domain*. O *Bus Pirate* serve de interface de comunicação com diversos protocolos como I2C, SPI, UART entre outros. Pode também ser utilizado na programação de microcontroladores e memórias. O equipamento é totalmente auto-suficiente ao apresentar uma interface de consola acedida através de uma porta série virtual, mas possui também um modo de acesso binário orientado a *scripts* e programas dedicados. Tanto um modo como o outro criam uma ponte de fácil acesso entre o programador e o equipamento em teste.

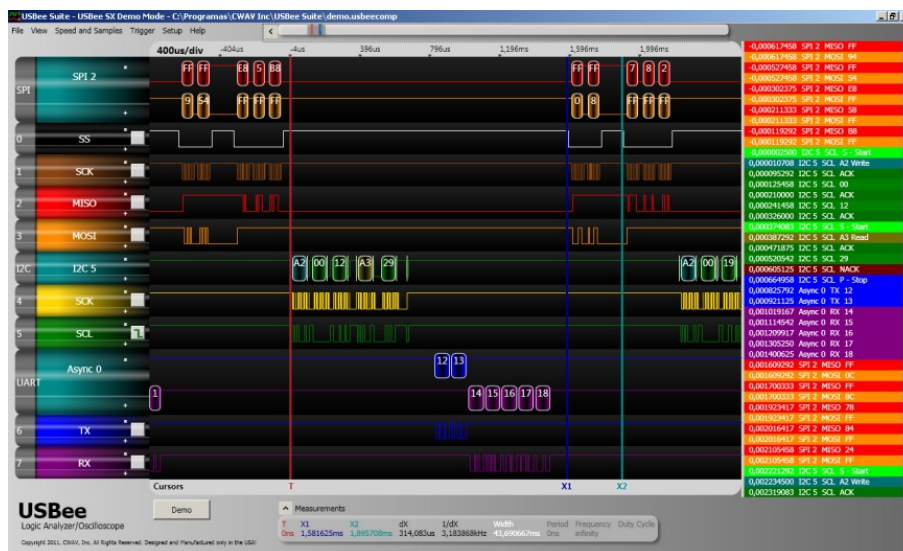
No contexto deste projeto, o *Bus Pirate* foi utilizado para efetuar a escrita e leitura de dados da memória externa.

5.1.4 Software Auxiliar

Não tendo o sistema operativo de um computador acesso à comunicação que é realizada entre a disquete e o controlador, o processo de deteção e correção de erros torna-se mais difícil. O controlador, ao não conseguir extrair a informação da disquete, informa o sistema operativo da ocorrência do erro através de um registo do controlador. O Sistema Operativo interpreta então



(a) USBee SX Logic Analyzer



(b) USBee Suite

Figura 5.3: Software do analisador lógico USBeeSX em modo de demonstração.

os erros e informa o utilizador através de uma mensagem mais amigável. Se estas mensagens de erro geradas pelo controlador de drives de disquetes são úteis, são também insuficientes no teste ao emulador, uma vez que não é possível extrair destes erros as circunstâncias exatas de como os erros ocorreram.

Não passa pelo objetivo deste trabalho a criação de uma interface entre o sistema operativo e o controlador de drive de disquetes. No entanto, são utilizados alguns programas já existentes que transmitem estas informações da forma mais útil para este projeto. Apesar de não orientados à verificação de sinais MFM, fornecem informação sobre os dados extraídos pelo controlador e, consequentemente, sobre erros que tenham ocorrido. Estas aplicações utilizam *drivers* (Controladores de software) do controlador da drive de disquetes alternativos que permitem a extração de

mais alguma informação do que a acessível através do sistema operativo Microsoft Windows XP utilizado no desenvolvimento e teste.

O controlador de software *fdrawcmd.sys* [82] é escrito por Simon Owen e expõe às aplicações um conjunto de comandos, antes não acessíveis, do controlador da drive de disquetes. Desta forma, é possível a escrita e leitura de faixas com formatos diferentes dos suportados pelo Sistema Operativo.

Foram usadas duas aplicações que utilizam este controlador e que permitem operar a drive de disquetes num nível mais baixo e refinado, como pedir a leitura de apenas uma determinada faixa e de um determinado lado.

A aplicação Fdio [83], escrita por Ansgar Kückes, surge da impossibilidade de ler disquetes em formatos diferentes dos especificados pelo sistema operativo. Pretendendo aceder a outros formatos como os utilizados por equipamentos HP, 9845 e LIF, Kückes cria uma aplicação que permite a leitura e escrita de outros formatos. Esta ferramenta pode também efetuar análises às disquetes e reportar as características do seu formato.

A aplicação SAMDisk [84], do mesmo autor do controlador *frawcmd.sys* Simon Owen, é também uma aplicação de linha de comandos, que permite a leitura e escrita de disquetes em formatos especiais.

Para além das operações de leitura e escrita das imagens das disquetes, as aplicações apresentam ainda algumas opções que permitem um controlo mais refinado das operações. No caso da aplicação SAMDisk, a velocidade de rotação do disco pode ser alterada, a existência do fenómeno de *Interleaving* é controlada e até o tamanho do campo de dados dos sectores pode ser manipulado. O uso destas aplicações permite testar o emulador da drive de disquetes de forma mais prática e precisa.

De forma a compensar a dificuldade de aceder aos dados que chegam ao controlador da drive de disquetes através do computador, é necessário analisar os sinais diretamente no cabo de 34 pinos que interliga o controlador e a drive de disquetes. Enquanto que alguns sinais como, por exemplo o sinal /INDEX, são facilmente interpretados através da aplicação do analisador lógico, outros exigem a sua descodificação, nomeadamente o sinal MFM.

No sentido de verificar que os sinais gerados pelo emulador obedecem às temporizações definidas e que são corretamente transmitidos, procedeu-se à programação de uma aplicação que faz a análise dos sinais capturados. Para além da análise de sinais, a aplicação tenta adaptar os dados em informação de mais fácil análise como a interpretação dos estados da drive de disquetes e a descodificação dos dados MFM.

Foi então implementada a aplicação *MFM Log Parser*. Esta aplicação, escrita em C#, faz a análise aos sinais trocados entre o controlador e a drive de disquetes a partir dos dados exportados pela aplicação *USBee Logic Analyzer* no formato ULB [80] já referido na secção anterior. Neste formato, após um cabeçalho, cada bit de um byte de dados do ficheiro representa o estado do sinal lógico de cada canal quando foi registado. Cada byte representa então os 8 sinais observados a cada amostra. Consegue-se assim processar toda a comunicação efetuada entre os dois dispositivos e interpretar a sua informação.

A aplicação começa por analisar o cabeçalho onde os dados sobre a captura estão registados. O cabeçalho contém a temporização da taxa de amostragem utilizada e a localização do sinal de *trigger* que sinaliza o início da captura dos sinais relevantes. Inicia-se então um processo de análise de cada byte e bit registados e, caso ocorra uma alteração num destes sinais que seja relevante para o programa, um evento é criado. Este evento será processado por um módulo do programa que é responsável pela interpretação dos eventos associados a cada sinal do cabo de comunicação. No caso dos eventos associados ao sinal MFM, um outro módulo é responsável pela descodificação dos sinais MFM, e consequente interpretação e verificação.

Todos estes eventos podem ser visualizados através de uma lista na interface que dispõe a informação descodificada e que sinaliza eventuais erros que sejam encontrados. A interface gráfica da aplicação é apresentada na Figura 5.4.

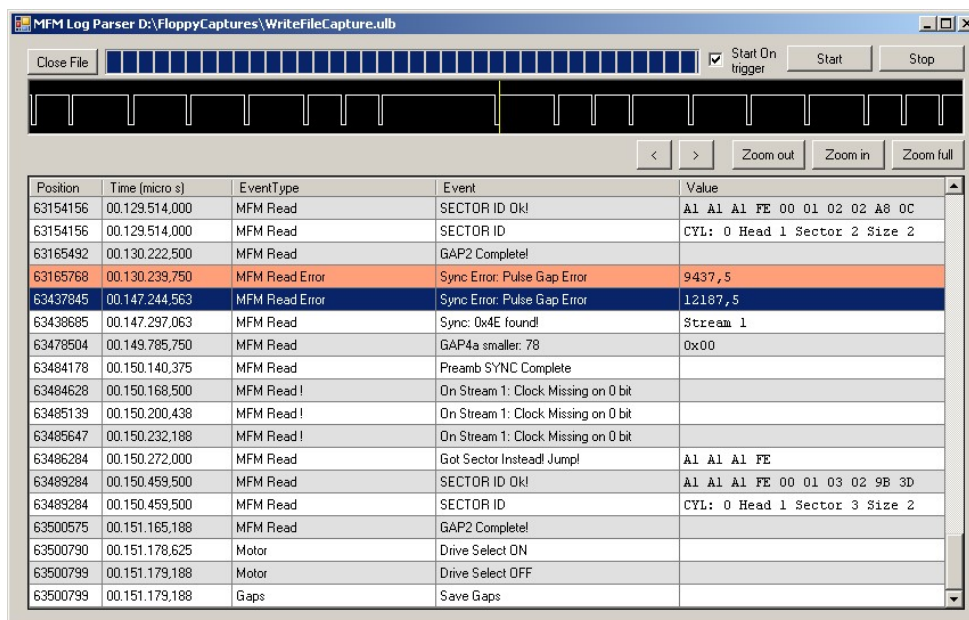
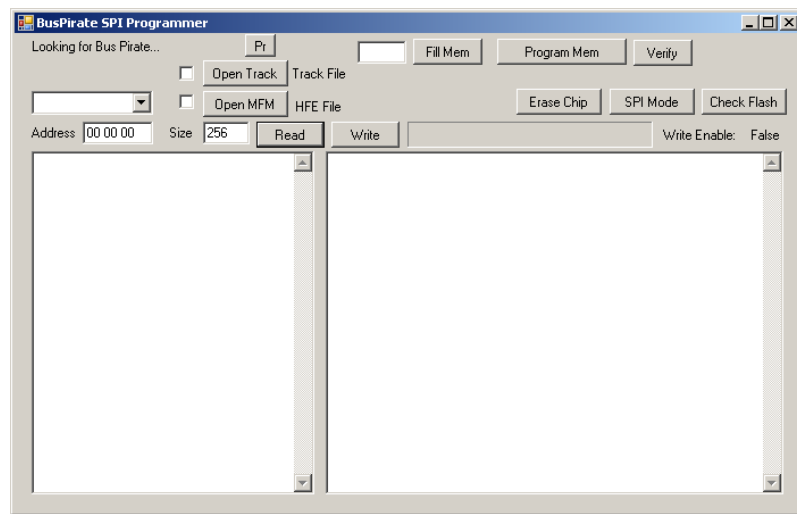


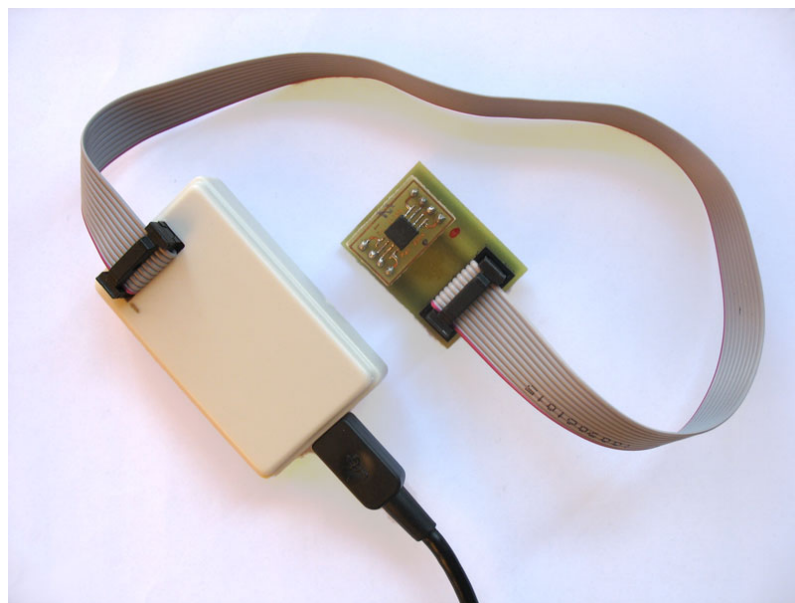
Figura 5.4: Interface da aplicação de análise do analisador *MFM Log Parser*.

Por último, de forma a auxiliar os testes de extração de dados da memória e de emulação, foi criada uma outra aplicação: *BusPirate SPI Programmer* que pretende facilitar a operação de leitura e escrita das memórias externas. Esta aplicação, também escrita em C# faz essa programação

através do adaptador *Bus Pirate*. Na Figura 5.5 é mostrado o programador e a placa de circuito impresso que foi fabricada para facilitar a programação das memórias.



(a)



(b)

Figura 5.5: Interface (a) e Programador (b) da memória externa com recurso ao Bus Pirate.

5.2 Gestor remoto

De forma a gerir e transmitir os dados para o Emulador através da rede ZigBee, é necessário recorrer a um dispositivo de gestão. Este gestor é composto por dois componentes: Um módulo em hardware e outro em software. O módulo em hardware (Coordenador) é composto por um Coordenador ZigBee compatível com o Emulador. O módulo em software (Gestor) interage com o módulo em hardware e corre num computador que irá gerir as imagens de disco e o envio de dados para o Emulador. A utilização de um computador para efetuar esta gestão diminui a complexidade do módulo físico e permite uma maior flexibilização das suas funcionalidades.

No sentido de minimizar o desenvolvimento do Coordenador, recorreu-se a um módulo já criado por outro aluno do MIEEC. O módulo Coordenador [85] criado por Décio Macedo permite a ligação de um SoC RC2300 a um computador via cabo USB e comunica com o computador via uma porta série virtual. Usando este módulo, é apenas necessário alterar o código *firmware* que o microcontrolador executa de modo a adequar-se às funcionalidades pretendidas.

O módulo Coordenador é responsável pela interpretação dos comandos recebidos do Gestor Remoto e do envio dos comandos para o Emulador. Do mesmo modo, recebe mensagens do Emulador e, caso seja necessário, envia-as para o programa Gestor.

Apesar do coordenador ser responsável pela gestão da rede ZigBee, tem apenas um papel de interligação entre o Emulador e o programa de gestão transferindo de forma quase transparente as mensagens entre estes dois atores. No entanto, possui também um papel de relevo na sequência de transmissão que é descrita na secção 5.6 Transmissão da imagem.

5.2.1 Software Gestor

O aplicação de gestão *Floppy Manager* (Gestor) é responsável por gerar novas imagens a partir de uma disquete física, gerir imagens existentes, verificar o estado da rede e dos emuladores e finalmente enviar a imagem para o emulador. A interface deste programa é replicada na Figura 5.6.

A aplicação Gestor foi desenvolvido para o sistema operativo Windows e escrito em C# usando Visual Studio C# 2010 Express Edition. A utilização de C# e da *framework* .NET na versão 4 permite utilizar um conjunto de funcionalidades que facilitam o desenvolvimento da aplicação como por exemplo o uso de controlos pré-existentes de interface com o utilizador, classes de gestão e comunicação entre processos e métodos de manipulação de ficheiros no disco.

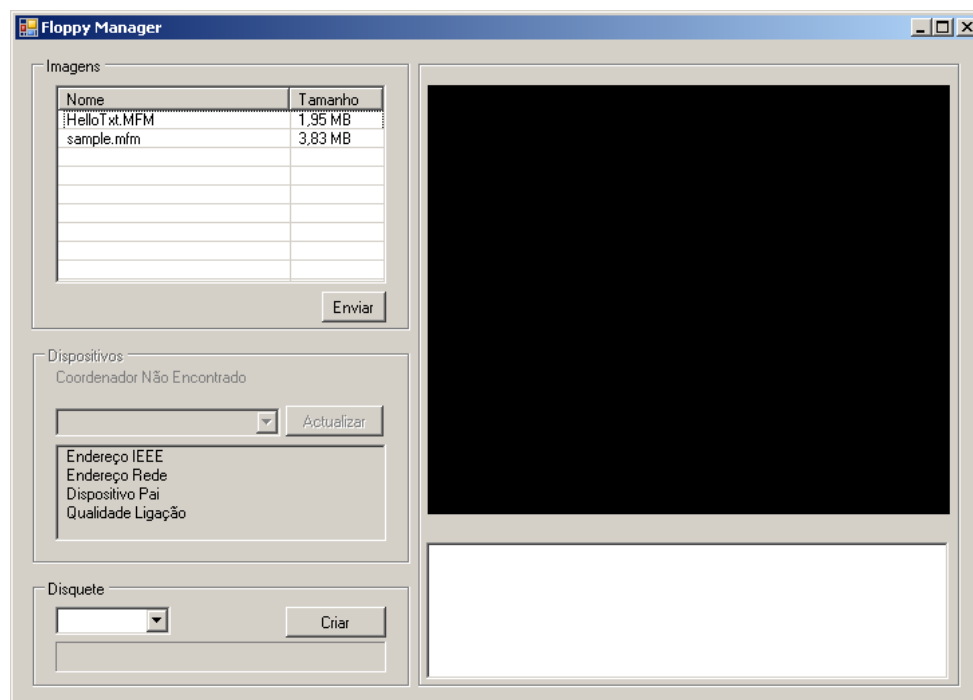


Figura 5.6: Interface da aplicação Gestor.

De notar que, com o progressivo desaparecimento das drives de disquetes dos computadores, também o suporte delas tem desaparecido das linguagens de programação mais recentes. De forma a obter as informações sobre a drive de disquetes e poder lê-la num formato compatível, foi necessário recorrer a algumas funções do sistema operativo que não estavam presentes na plataforma .NET.

Programou-se então a aplicação para, no caso de detetar a presença de uma drive de disquetes, disponibilizar a opção de criação de imagem a partir da drive. A imagem é criada em dois passos: através da leitura dos conteúdos dos campos de dados das disquetes para um ficheiro binário, seguida da conversão deste ficheiro para um novo em formato apropriado à emulação através de um programa auxiliar. O formato do ficheiro das imagens de disco utilizadas e como podem ser geradas é aprofundado na secção 5.4.

No arranque, o programa *Gestor Floppy Manager* questiona o Sistema Operativo pelos equipamentos ligados às portas USB e, caso detete o dispositivo Coordenador, abre a porta série virtual criada por este último de modo a iniciar a comunicação. Caso o Coordenador não esteja ligado, o programa monitoriza os eventos associados às portas USB e estabelece a ligação assim que o coordenador seja ligado. Para acautelar a possibilidade da existência de erros na comunicação entre o computador e o coordenador foi adaptado um protocolo que prevê a deteção de erros como descrito na secção seguinte 5.2.2.

Encontrado o coordenador, segue-se a pesquisa de emuladores disponíveis que é feita através de envio de mensagens de *broadcast*. Os emuladores existentes respondem a esta mensagem dando conhecimento do seu endereço e indicam a sua disponibilidade para receber os comandos como por exemplo o envio de uma nova imagem de disco.

A natureza assíncrona da comunicação série e da comunicação sem fios aumentou a complexidade do programa de gestão. A comunicação com o Coordenador é composta por comandos que podem requerer uma resposta. No entanto, esta resposta pode não chegar ou, entretanto, chegar outra resposta (possivelmente de erro) que não a esperada. Como a receção das mensagens série é efetuada num processo separado do processo principal do programa foi necessário implementar um sistema de eventos e de troca de mensagens entre os processos que correm na aplicação Gestor que permitisse o correto funcionamento nos seus diversos estados. Ao chegar uma mensagem do Coordenador, o programa verifica a validade dos dados contra erros de transmissão e coloca a mensagem numa lista que é depois consultada por outros processos do programa. Caso um processo esteja à espera de uma resposta, pode sinalizar esta intenção e ser sinalizado quando a resposta chegar.

Com esta funcionalidade, o processo que é responsável pelo envio de dados da imagem pode enviar uma trama e ficar parado à espera da resposta que autoriza o envio da trama seguinte sem interferir com outros processos da aplicação. No entanto, se for transmitida uma mensagem de erro do Coordenador, este pode ser verificado e corrigido, pois o processo deteta que a autorização para enviar a mensagem seguinte não foi recebida em tempo útil e verifica a existência de erros.

5.2.2 Comunicação via porta série

Na comunicação via porta série entre a aplicação Gestor e o Coordenador é utilizado um formato de tramas baseado no protocolo de comunicação Z-Tool Serial Port Interface [86], mas adaptado para comandos com apenas 1 byte e comprimentos de dados até 65535 bytes. Desta forma, é possível tanto o envio de comando de pesquisa de Emuladores que não necessita de dados, como o envio do comando de escrita de dados em memória em blocos de 256 bytes.

SOP	CMD	LEN	DATA	FCS
1	1	2	LEN	1

Tabela 5.2: Formato e tamanho dos campos do pacote utilizado na comunicação série.

Cada pacote é constituído por diversos campos como representado na Tabela 5.2. A transmissão é iniciada por uma campo de inicio de pacote (SOP) seguido do código do comando (CMD)

e de dois octetos de comprimento de dados (*LEN*). Caso este seja zero, então o campo de dados (*DATA*) não é transmitido e o pacote é finalizado com o campo de verificação de dados (*FCS*).

O campo de comprimento (*LEN*) e de dados (*DATA*) são enviados com o byte mais significativo primeiro. O campo *FCS* (*Frame Check Sequence*) contém o resultado da operação **XOR** de todos os bytes a partir do *CMD* até ao último byte de dados. O octeto *SOP* não é incluído nesta operação. Assim, ao se efetuar a operação **XOR** em toda a trama o resultado deverá ser 0. Caso o valor calculado seja diferente é indicador da existência de um erro na trama e esta é descartada.

De notar que a utilização da operação **XOR** pode não proteger contra erros que afetem um número múltiplo de bits e também não permite a reconstrução da trama em caso de erro. No entanto, deverá ser suficiente para a aplicação em causa uma vez que a quantidade de erros ocorridos durante os testes foi reduzida e foram todos detetados pela aplicação.

Os comandos utilizados nos canais de comunicação deste projeto são listados na Tabela 5.3. Alguns destes comandos são utilizados também na comunicação sem fios entre o Controlador e o Emulador.

STATUS_REQ	Pede aos dispositivos na rede dados sobre o seu estado.
STATUS_RSP	Resposta dos dispositivos na rede ao comando STATUS_REQ .
MODE_SET	Altera o estado de funcionamento do Emulador.
MODE_GET	Resposta do Emulador sobre o estado de funcionamento.
MODE_ERROR	Resposta do Emulador caso seja emitido o comando MODE_SET com um estado inválido.
MEM_WRITE	Comando de escrita de dados na memória externa do Emulador
MEM_STATUS	Pede ao Emulador dados sobre a memória externa.
MEM_RSP	Resposta do Emulador ao comando MEM_STATUS .
MEM_RESET	Reinicia o estado de processamento da rotina de gravação de dados na memória externa do Emulador.
MEM_ACK	Sinal de resposta ao comando MEM_WRITE que sinaliza envio dos dados com sucesso.
ZIGBEE_ERROR	Comando que sinaliza a ocorrência de um erro na comunicação sem fios.
UART_ERROR	Comando que sinaliza a ocorrência de um erro na comunicação via porta série.

Tabela 5.3: Comandos utilizados na comunicação.

Na comunicação sem fios, os comandos são transmitidos como *clusters* dos pacotes, libertando assim os campos de dados para informação extra. A definição dos *clusters*, já referida na secção 4.2, permite a interoperabilidade entre equipamentos diferentes mas que tenham o mesmo objetivo. Neste caso, são utilizados comandos exclusivos para esta aplicação uma vez que não é previsto este uso na especificações ZigBee.

5.3 Emulador

O Emulador é o dispositivo que implementa todas as tecnologias analisadas nos capítulos 3 e 4. Interage com um controlador de drive de disquetes via cabo de dados de 34 pinos e com o coordenador através da rede sem fios.

O comportamento do Emulador é definido através de uma máquina de estados. São definidos 3 estados gerais: Espera, Carregar Imagem e Emulação. O Diagrama geral da relação entre estes estados pode ser consultado na Figura 5.7.

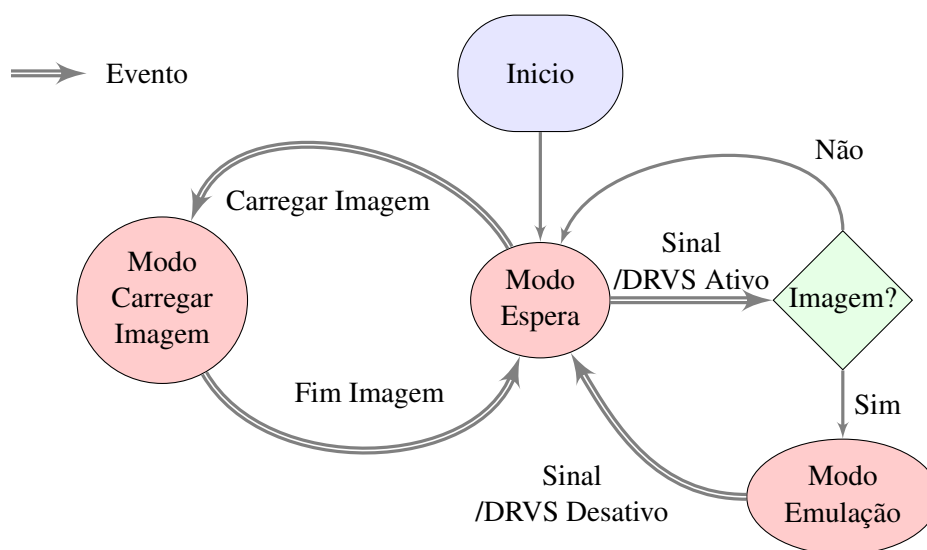


Figura 5.7: Diagrama de funcionamento geral do Emulador.

No arranque, o microcontrolador perfaz a configuração e inicialização do sistema e dos respetivos periféricos e interfaces sem fios. O arranque fica concluído assim que o Emulador inicia todos os periféricos necessários. Entretanto, tenta ligar-se a uma rede ZigBee criada pelo Coordenador mas esta não é condição necessária para o seu funcionamento como Emulador. Terminada esta sequência, O Emulador encontra-se no modo Espera e fica à espera de um estímulo externo, seja este uma mensagem do Gestor/Coordenador via interface sem fios, seja via sinais do controlador de drive de disquetes.

Ao receber um sinal – mais propriamente a sequência /MOTEx e /DRVSx do controlador de drive de disquetes, o Emulador verifica a existência de uma imagem de disco e muda para o modo Emulação. Ignora os outros sinais do controlador uma vez que, sem os sinais de seleção de drive ativos a ativação dos outros sinais indica que o controlador de disquetes pretender comunicar com outra drive de disquetes presente no mesmo cabo. Caso receba uma mensagem ZigBee do controlador contendo o comando de mudança de estado apropriado, passará para o modo Carregar Imagem.

Durante a emulação, de forma a não interromper a transmissão de dados entre o emulador e o controlador de drive de disquetes, as interrupções associadas à transmissão sem fios são desabilitadas. Assim, para o programa Gestor, o Emulador encontra-se desligado e não é possível passar do estado de Emulação para o estado Carregar Imagem sem passar pelo estado intermédio de Espera onde a comunicação é restabelecida.

De modo análogo, de forma a garantir a validade da informação armazenada, durante o modo de Carregar Imagem o Emulador sinaliza ao controlador de disquetes que não está disponível para emulação, ativando o sinal de mudança de disco /DSKCHG, isto é, indica que não possui nenhuma disquete (emulada) inserida.

Uma vez que não existem requisitos de poupança de energia relativos ao uso de baterias, este dispositivo é implementado a partir de um dispositivo *Router* em vez de um dispositivo *End Device* que permite desligar-se da rede e entrar em estados de baixo consumo para poupar energia. A implementação da comunicação sem fios é facilitada pela utilização de um dispositivo *Router* ao não diminuir a frequência com que é feita a comunicação entre dispositivos. No entanto, como o *Router* também tem um papel ativo na gestão de rede, durante o estado de Emulação em que a comunicação sem fios está desativada, pode provocar quebras na comunicação entre outros dispositivos.

5.3.1 Modo Emulador

No modo de Emulação, a comunicação com o controlador de disquetes necessita de ser processada tanto a nível físico, adaptando os sinais elétricos de forma a serem compatíveis com os equipamentos, como a nível de lógico, interpretando e gerando os sinais apropriados.

Como referido no Capítulo 3, a drive de disquetes possui pouca lógica para processar os comandos e dados. Esta limita-se a manipular os motores que possui a partir dos comandos recebidos do controlador de disquetes. Por exemplo, um sinal no pino /MOTEx, ativa o motor central que permite a rotação do disco magnético. No emulador, não existe este motor, mas a consequência desta rotação, tem que ser replicada neste: inicia o envio dos dados guardados na memória de forma cíclica. A implementação em hardware da deteção destes sinais é aprofundada na secção 5.3.3.

Para garantir o correto funcionamento enquanto no modo Emulação, ao entrar neste modo, o programa efetua alguns comandos de verificação e configuração. Caso o emulador não tenha nenhuma imagem carregada, sinalizará este facto ao controlador através do sinal de /DSKCHG que indica a ausência de uma disquete na drive emulada. Nesta situação, o emulador sairá deste

modo e entrará de novo no modo de Espera e ficará à espera que lhe sejam enviados dados para armazenar a imagem da disquete.

Caso exista uma imagem válida, o emulador só poderá entrar no estado de emulação após desligar o sistema rádio. Este passo é necessário de forma a garantir que durante o processamento de dados não haja qualquer interrupção, que poderia significar corrupção no envio dos dados. O sistema rádio é desligado através da desativação das interrupções do microcontrolador associadas à transmissão rádio.

Após a inicialização, o emulador calcula a faixa e o lado que o controlador pretende através dos sinais de /HEAD, /STEP e /DIR e envia os dados armazenados na faixa correspondente sendo que a faixa 0 é definida inicialmente de forma arbitrária. Como representado na figura 5.8, ao receber um dos sinais de controlo que mudam a faixa (/STEP, /SIDE), o Emulador recalcula a faixa a ser enviada e altera a posição de memória de onde os dados serão extraídos.

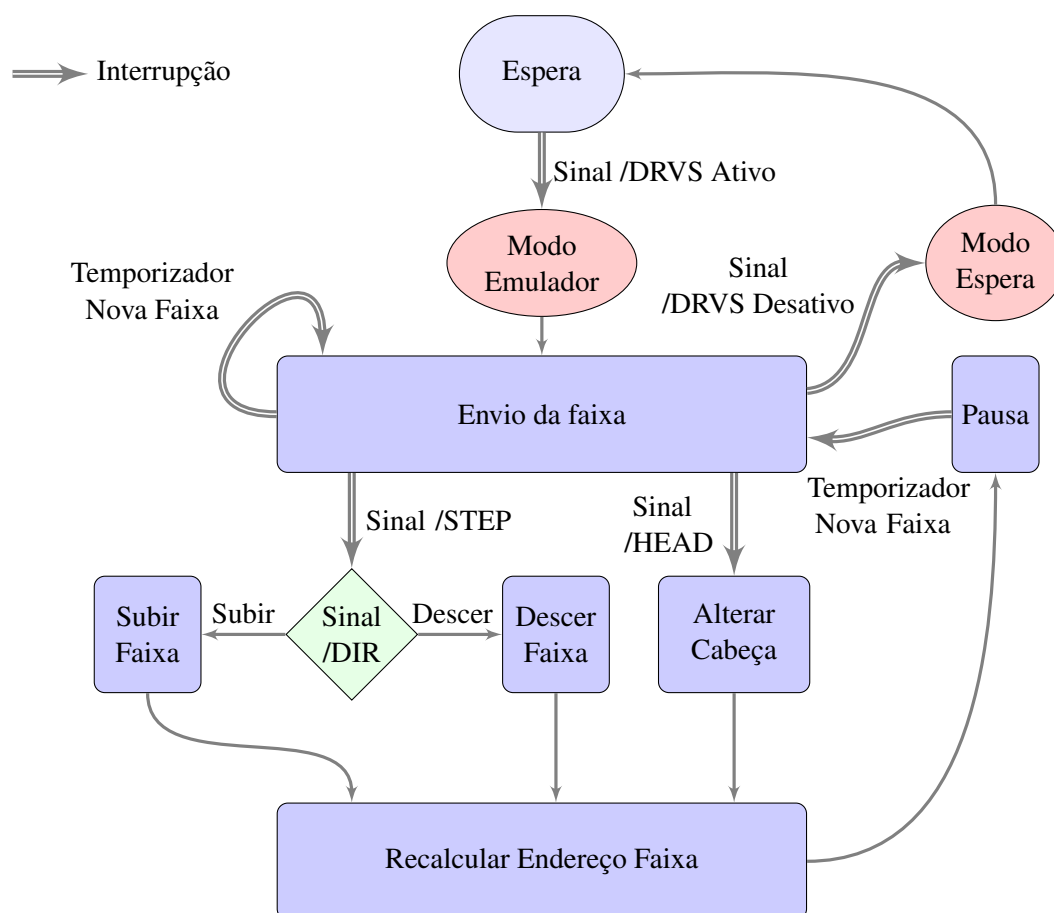


Figura 5.8: Diagrama de funcionamento do Emulador ao processar os sinais do controlador.

O sistema de envio dos dados MFM foi um dos principais desafios encontrados na realização deste trabalho. Este processo é descrito em maior pormenor na secção 5.7 Implementação

Protocolo MFM e o processo de "Envio da faixa" é descrito com maior pormenor na secção 5.7.4.

Dificuldades no processamento dos dados recebidos na operação de escrita das disquetes associada ao processamento extra necessário para efetuar a gravação de dados na memória externa não permitiram a implementação da funcionalidade de escrita. No entanto, é previsto no esquemático e foi colocado um *socket* na placa de circuito impresso para facilitar o uso de uma memória RAM que possibilite esta funcionalidade no futuro.

Quando são desativados os sinais de seleção e ativação de drive (/DRVS), o Emulador sai da rotina de processamento de envio de dados e volta a ativar as interrupções rádio. De seguida, regressa ao modo Espera ficando preparado para nova intervenção.

5.3.2 Modo Carregar Imagem

No modo Carregar Imagem, o Emulador recebe e armazena os dados enviados pelo programa Gestor. O protocolo ZigBee, orientado a mensagens de pequena dimensão, impõe que a transferência de dados seja realizada através de uma série de pacotes. O conjunto de pacotes enviados formará então uma imagem de disco (disquete).

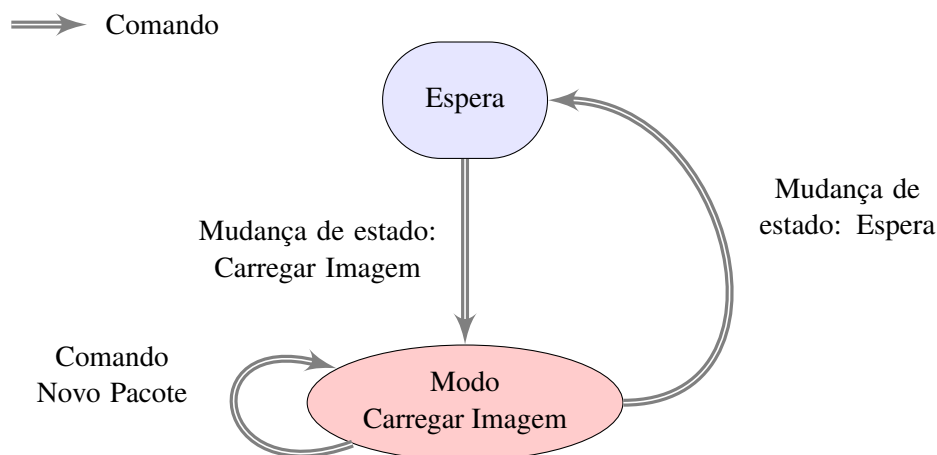


Figura 5.9: Diagrama de funcionamento do Emulador no modo Carregar Imagem

De forma similar ao modo Emulador, para não ser interrompido pelo computador anfitrião enquanto a imagem não tiver sido totalmente transmitida, o Emulador sinaliza o controlador de que está sem a "disquete" e ignora os pedidos deste.

Sequencialmente, o Emulador recebe cada pacote de dados e prepara-os de forma a poder escrever na memória externa. A sequência de envio de dados será descrita mais pormenorizadamente na secção 5.6.

Quando toda a informação estiver guardada, o Emulador sai do modo Carregar Imagem para o modo Espera e está pronto para interagir com o controlador quando este o requerer.

5.3.3 Hardware

O Emulador, interagindo tanto através da interface sem fios, como com o controlador de disquetes através do cabo de dados, necessita de hardware dedicado que lhe permita efetuar as suas funções. Como referido nas secções 4.1 e 5.1.1, os módulos utilizados já implementam o hardware necessário para a comunicação sem fios. No caso das placas SoC_BB com os módulos CC2430EM e CC2431EM é apenas necessária a colocação da antena. O protótipo do dispositivo Emulador é implementado como extensão da placa SoC_BB que permite um acesso fácil aos pinos do microcontrolador CC2430.

A alimentação do emulador é feita através do cabo de alimentação da drive de disquetes que se pretende substituir. Este cabo é terminado com um conector 4 pinos criado pela Berg Electronics Corporation e fornece a alimentação a 5 V e a 12 V como pode ser visto na Figura 5.10.

Pino	Cor do fio	Tipo
1	Vermelho	+5V
2	Preto	Terra
3	Preto	Terra
4	Amarelo	+12V

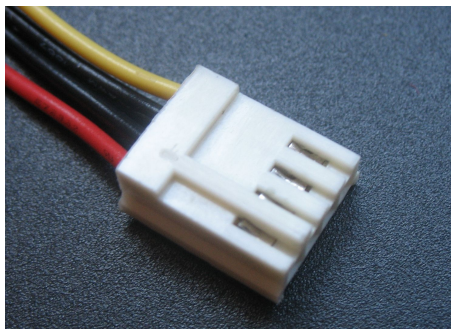


Figura 5.10: Conector de energia de 4 pinos.

Como a comunicação com o controlador da drive de disquetes funciona a 5 V TTL, a de alimentação 5 V é utilizada para alimentar a lógica associada à comunicação com o controlador da drive de disquetes. No entanto, o microcontrolador é alimentado a 3.3 V sendo assim necessários dois circuitos de alimentação distintos. Um regulador de corrente LM1117 é utilizado para converter os 5 V para os 3.3 V necessários para alimentar o módulo CC2430.

Para efetuar a emulação o microcontrolador necessita de responder a e manipular uma série de sinais que se interligam com o controlador de drive de disquetes. Os sinais relevantes de entrada,

de todos os apresentados na Tabela 3.2, podem ser resumidos aos da Tabela 5.4.

Tabela 5.4: Sinais de entrada da drive de disquetes.

Pino	Sinais
2	/DENSEL Seleção Densidade
10	/MOTEA Activar Motor Drive A
12	/DRVSB Seleccionar Drive B
14	/DRVSA Seleccionar Drive A
16	/MOTEB Activar Motor Drive B
18	/DIR Seleccionar direcção
20	/STEP Passo cabeça leitura/escrita
22	/WDATA Escrita de Dados
24	/WGATE Activação de modo escrita
32	/SIDE Seleção de cabeça

O Sinal /DENSEL como referido na secção 3.3 tem utilidade reduzida nesta aplicação uma vez que os modos de operação são predefinidos a influência deste sinal seria nula. Opta-se assim por não utilizar este sinal.

Os sinais /MOTEx e /DRVSx estão geralmente associados na sua função. O sinal /DRVSx é utilizado pelo controlador para indicar à drive de disquetes que pretende interagir com o equipamento. /MOTEx é o sinal que ativa o motor de rotação das disquetes. Afinal, numa drive de disquetes com motor, este demorará algum tempo até estabilizar a sua velocidade de rotação e consequentemente até a drive atingir um estado estável para comunicação [29, 87].

Assim, poderia ser de esperar que o sinal de /MOTEx fosse ativado momentos antes do sinal de /DRVSx de forma a que a velocidade de rotação estável seja atingida. Em controladores modernos, como o 82077, utilizados nos computadores de testes, estes sinais são controlados através de um registo próprio e a folha de características do controlador recomenda que os sinais sejam ativados simultaneamente [28].

De facto, ao se efetuar a análise dos sinais emitidos do controlador, registam-se as formas de onda representadas na Figura 5.11 que mostra que os dois sinais são ativados simultaneamente.

No caso do emulador, não existe um motor que necessite de ativação prévia e poderia usar-se apenas um dos sinais como controlo. Para aumentar a versatilidade do emulador face a outros controladores que podem ter comportamentos diferentes, optou-se por combinar os sinais através de uma porta lógica **OR** e sinalizar o emulador da intenção do controlador apenas quando os dois sinais estão ativos. Neste circuito, representado na Figura 5.12, a porta lógica **OR** é implementada através de 3 portas **NOR** e a seleção da drive, A ou B, sob o qual a emulador deve responder é realizada através de um seletor (*jumper*) manual. Normalmente é utilizada a configuração para a

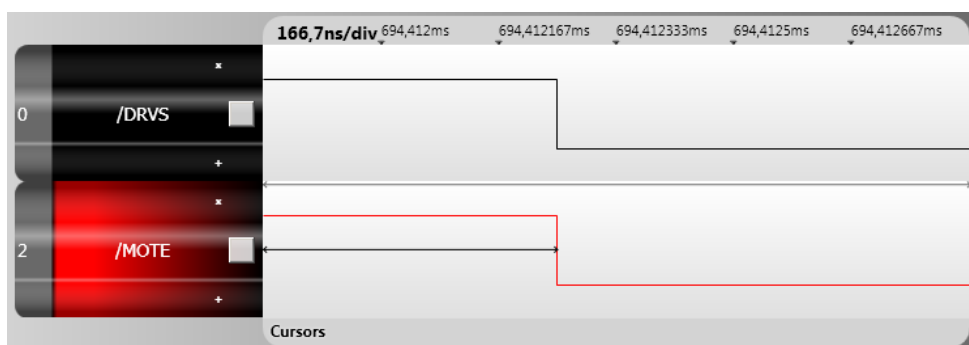


Figura 5.11: Sinais /DRVSx e /MOTEx ativados simultaneamente pelo controlador de drive de disquetes.

drive B devido ao uso de cabo de dados com troca de sinais como o representado na Figura 3.3. É utilizado o sinal de saída do circuito de ativação /DRVS como indicador de início de emulação.

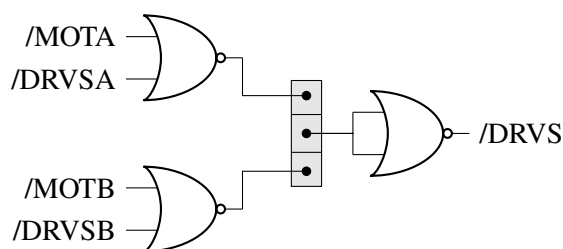


Figura 5.12: Esquema lógico do circuito de ativação do Emulador.

Os sinais /WDATA e /WGATE, associados à operação de escrita na drive de disquetes não são utilizados pois o Emulador não suporta a operação de escrita. De forma a esta funcionalidade ser suportada no futuro, estão ligados eletricamente ao microcontrolador.

São assim 6 os sinais que o microcontrolador necessita de ler na sua operação de emulação: /DVRs, /DIR, /STEP, /SIDE, /WDATA e /WGATE. É preciso ter em atenção que os módulos CC2430 não toleram os 5 V utilizados pelo controlador e, como tal, a lógica de conversão é essencial.

Para converter os sinais de entrada recorreu-se a um conversor de nível de 6 vias 74HC4050 da Philips. Este circuito integrado converte sinais até aos 15 V para níveis mais baixos, neste caso os 3.3 V suportados pelo CC2430. A alta tolerância à tensão de entrada (máx. 15 V) ajuda a proteger o microcontrolador contra tensões demasiado altas nos seus pinos.

Por outro lado, os sinais de saída do microcontrolador também necessitam de ser convertidos para os 5 V de forma a serem reconhecidos pelo controlador. Selecionando os sinais de saída da drive de disquetes, referidos na secção 3.3 elaborou-se a tabela 5.5 que mostram os 5 sinais que

necessitam da conversão de nível. Utiliza-se *Buffer/Driver* de 6 vias SN7407 da Texas Instruments que efetua a transição dos 3.3 V para os 5 V.

Tabela 5.5: Sinais de saída da drive de disquetes

Pino	Sinais
8	/INDEX Índice
26	/TRK00 Faixa 00
28	/WPT Protecção Escrita
30	/RDATA Leitura de Dados
34	/DSKCHG Mudança/Ejeção Disco

No modo de Emulação, o microcontrolador responde aos comandos do controlador ao detetar as transições nos sinais lógicos via interrupções. No entanto, o CC2430 apenas permite a configuração de interrupções à subida ou à descida do degrau enquanto que na aplicação em causa é necessário estar atento às duas transições em pelo menos dois sinais. 1) O sinal /DRVS quando ativo na descida indica o início da comunicação e quando desativo na subida sinaliza o fim da comunicação; e 2) /HEADS que indica qual o lado do disco que se pretende utilizar. Como a seleção de interrupção à subida ou à descida se aplica a toda a porta, não é possível alterar esta configuração após um dos eventos ocorrer, pois afetaria os outros sinais. Uma funcionalidade de *polling* também não é viável pois interferiria na transmissão de dados.

Para reagir a todos estes eventos, programou-se o CC2430 para disparar as interrupções nas descidas dos sinais lógicos e recorreu-se então a um circuito extra que deteta os eventos de subida dos sinais /HEADS e /DRVS e gera um impulso de descida numa das portas do microcontrolador disparando assim uma interrupção que verifica o estado dos sinais.

O circuito de deteção de degrau é realizado através do circuito integrado 74LS221 que possui dois multi-vibradores mono-estáveis que podem ser configurados para realizar a operação pretendida de geração de impulsos. Este circuito pode ser consultado na Figura 5.13 e recorre a um divisor resistivo para efetuar a conversão de níveis lógicos.

Para além do controlador de drive de disquetes, o Emulador ainda comunica com uma memória *flash* externa que é utilizada para armazenamento da imagem da disquete. A comunicação com a memória é realizada via protocolo SPI e usa as portas dedicadas USART1 SPI Alt.2 do CC2430 [88]. O uso destas portas facilita a comunicação com a memória pois funções dedicadas são fornecidas tanto pelo microprocessador como pela camada de abstração *OSAL* usada.

O Emulador, como extensão da placa SoC_BB, possui ainda um botão de *restart* e dois LED que fornecem *feedback* ao utilizador. Foi ainda instalado um adaptador de 8 pinos, também ligado à porta SPI, para instalação de uma memória RAM de forma a permitir a funcionalidade de escrita no futuro.

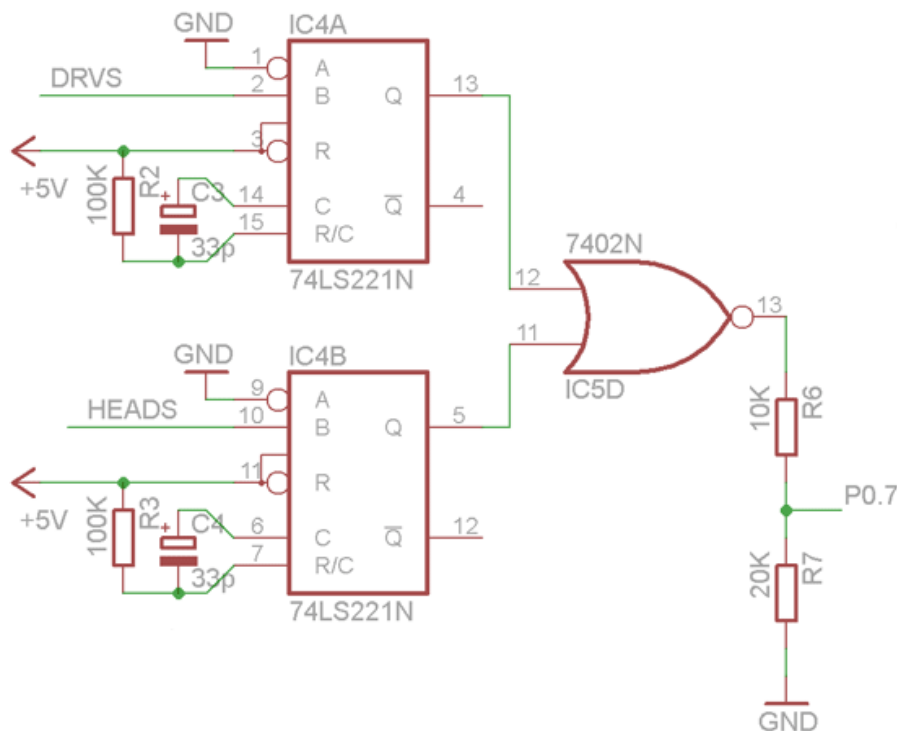


Figura 5.13: Circuito de detecção de degrau nos sinais /DRVS e /HEADS

O esquema elétrico e desenho da PCB do dispositivo Emulador, podem ser consultados no Anexo [A](#).

5.4 Formato da imagem

De forma a simplificar o processo de transmissão, armazenamento e processamento de dados, é recomendado o recurso de imagens de discos. Uma imagem de disco guarda toda a informação presente num disco num só ficheiro de computador. Desta forma, é possível abstrair-se de diversas camadas lógicas que não influenciam o funcionamento do emulador. Para este, não é essencial saber qual o tipo de ficheiro a ser transmitido nem o formato de sistema de ficheiros que é utilizado na disquete. Na ausência desta abstração, ficaria a cargo do emulador processar os ficheiros transmitidos e interpretar um sistema de ficheiros que seria utilizado para guardar os dados a serem enviados. Considerando que o controlador de disquetes do sistema anfitrião não faz os pedidos da informação da disquete por ficheiros mas sim por faixas no disco, é simplesmente necessário manter o registo e conteúdo das faixas que é totalmente independente do sistema de ficheiros. Simplifica-se assim o processo de forma significativa.

Como mencionado na secção 2.3, a utilização de imagens de discos não é inédita e são encontradas diversas utilizações para as mesmas, por exemplo a realização de cópias de segurança. Ao guardar toda a informação presente num disco, é possível que este seja reposto de forma rápida e direta sem qualquer processamento extra.

No caso do emulador, esta funcionalidade é fulcral uma vez que as temporizações do protocolo MFM como descritas na secção 3.3.1 dificultam o processamento extra dos dados em tempo real. O processo de envio de dados via Emulador para o controlador é aprofundado na secção 5.7.

No âmbito das imagens de discos, existem alguns formatos específicos para disquetes. De facto, verifica-se a existência de formatos não tanto associados ao tipo e tecnologia que equipam as disquetes mas sim à sua utilização. Surgem então uma variedade de formatos com diferentes características que os tornam mais ou menos apropriados aos diferentes usos.

A título de exemplo, referem-se dois destes formatos.

O formato VFD *Virtual Floppy Disk* é um formato criado pela Microsoft para utilização em ambientes de emulação virtuais como *Windows Virtual PC* e *Virtual Server* [89,90].

O formato LIF *Logical Interchange Format* da Hewlett-Packard é utilizado para armazenar informação usada numa série de calculadoras e outros equipamentos que utilizavam um sistema de armazenamento proprietário HP-IL. Os ficheiros LIF replicam estes sistemas de ficheiros e são necessárias ferramentas de conversão para utilização dos dados em ambientes diferentes [91,92].

Tentando ignorar os ambientes em que as disquetes são utilizadas, aparecem imagens genéricas que contêm dados que replicam o conteúdo completo das disquetes incluindo as tabelas de alocação de ficheiros e não apenas os ficheiros do utilizador. Com este formato consegue-se aceder a todos os sectores de dados presentes na disquete mas sem a informação de controlo descrita no Capítulo 3. Sendo que os ficheiros deste tipo contêm apenas os campos de dados, o tamanho do ficheiro corresponde ao tamanho da disquete.

A simplicidade e versatilidade que este tipo de formato encarna faz com que seja suportado por uma variedade de aplicações que lidam com ficheiros de imagens de disquetes como o *Virtual Floppy Drive*, WinImage, SAMdisk, Fdio, RawWriteWin [83, 84, 93–95]. De notar que estes ficheiros não possuem uma especificação ou conteúdos próprios pelo que apenas a extensão do ficheiro e o tamanho deste ajuda a identificar a tipo de disquete que representa. Assim, uma variedade de extensões podem ser usadas, mas as mais comuns são as *.img* ou *.raw*.

No caso da aplicação SAMdisk, a lista de formatos que suporta é extensa e é exemplificativa da variedade de formatos existentes [84]:

- EDSK - Extended disk image (Amstrad CPC, Spectrum +3, PC);
- MGT - MGT +D/Disciple/SAM (Sinclair Spectrum / SAM Coupé);
- SAD - SAm Disk (SAM Coupé);
- SBT - Sam BooTable disk (SAM Coupé);
- SDF - Sam Disk Format (SAM Coupé);
- CPM - Pro-DOS CP/M (SAM Coupé);
- TD0 - Sydex TeleDisk (various);
- IPF - Interchangeable Preservation Format [apenas MFM IBM-compatível];
- TRD - Beta128 disk for TR-DOS (Sinclair Spectrum);
- FDI - Full Disk Image (Sinclair Spectrum, e não Disk2FDI!);
- OPD - OPus Discovery (Sinclair Spectrum);
- MBD - MB-02+ Disk (Sinclair Spectrum);
- UDI - Ultra Disk Image (Sinclair Spectrum);
- SCL - Sinclair betadisk archive (Sinclair Spectrum);
- DSK - Disk image (Amstrad CPC);
- DSC - WinAPE disk image (Amstrad CPC);
- CFI - Compressed Floppy Image (Amstrad);
- BPB - FAT12/16 BIOS Parameter Block (MS-DOS, Atari ST);
- MSA - Magic Shadow Archive (Atari ST);
- D80 - Didaktik D80;
- DMK - David M Keil's disk format (TRS-80);
- IMD - ImageDisk utility image;
- D81 - Commodore 1581;
- D2M - Commodore CMD FD-2000;
- D88 - Toshiba Pasopia 7 disk (NEC PC-xx);
- LIF - Logical Interchange Format (Hewlett-Packard);
- S24 - Sega System 24 (Arcade, formatos 1.8M and 1.88M);
- RAW - Dados dos sectores, apenas identificado pelo tamanho do ficheiro.

Finalmente, referem-se formatos especificamente criados para a utilização num emulador. Estes formatos são criados de forma a minimizar a necessidade de processamento dos dados pelo equipamento. Uma disquete, não contém apenas campos de dados mas sim uma série de campos de controlo em cada sector e faixa (Secção 3.4). O Emulador, ao necessitar de enviar também esta informação pode gerá-la durante a emulação ou utilizar uma imagem de disco que a contenha.

No contexto dos emuladores referidos na secção 2.3, surgem formatos próprios dedicados à emulação. Salientam-se os formatos HFE e MFM criados por François del Nero. Estes formatos contêm a informação de baixo nível guardada num modo apropriado à emulação de sinais FM e MFM.

Como referido na secção 3.3.1, a comunicação MFM depende do histórico de bits enviados anteriormente para codificar o bit seguinte e, como tal, existe a necessidade de processar a informação a ser enviada. Este processamento pode ser efetuado em tempo real, durante o envio, pelo emulador ou ser feito previamente no processo que gera a imagem dos dados. Dependendo

da capacidade do microcontrolador responsável pela emulação, pode ser desejável dispensar esse processamento e utilizar dados previamente tratados.

Neste projeto, as imagens utilizadas na programação do Emulador são geradas a partir de ficheiros no formato MFM. Estes ficheiros, com extensão “.mfm”, podem ser criados através de aplicações dedicadas [31]. A aplicação *HxCFloppyEmulator*³ é uma aplicação Windows com ambiente gráfico que permite um conjunto de operações associadas ao emulador referido em 2.3. Uma destas operações é a criação de ficheiros MFM a partir de ficheiros presentes no computador ou de outros formatos de imagens. Um outro programa, utilizado na conversão para este formato, é o *HxC Floppy Emulator : Floppy image file converter*³ que funciona através da linha de comandos. Esta aplicação converte ficheiros de imagens presentes numa pasta para o formato especificado nos argumentos da linha de comandos. Esta aplicação é usada na criação de imagens MFM utilizadas no programa Gestor do Emulador 5.2.1. O programa de gestão extrai os dados da disquetes e guarda-os num ficheiro temporário. O programa de conversão é então executado e o ficheiro temporário é convertido para o formato MFM pretendido.

No entanto a utilização destes formatos implica a necessidade de memórias maiores. Um ficheiro “.mfm” que seja criado a partir de uma disquete de 1.44 MB ocupa cerca de 4 MB. Este aumento do tamanho é o resultado de diversos fatores:

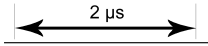
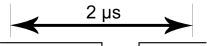
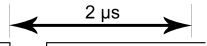
- A codificação dos sinais de controlo presentes nas disquetes para formatação das faixas e sectores utilizados pelos controladores de disquetes. No total, esta informação juntamente com os campos de dados, ocupam, numa disquete de 1.44 MB, cerca de 2 MB;
- A codificação dos sinais no formato compatível com a transmissão MFM requer a utilização de dois bits por cada bit de dados originais. Duplicam-se assim os dados a armazenar;
- O ficheiro “.mfm” possui um cabeçalho que contém dados sobre a disquete a ser emulada.

Na implementação do Emulador, o formato de dados utilizado é baseado neste formato MFM mas apresenta algumas modificações de forma a tornar mais fácil de manipular.

O pré-processamento dos dados de forma a facilitar a transmissão no protocolo MFM é efetuado para retirar lógica de controlo do emulador. Como descrito na secção 3.3.1, os sinais binários 1 e 0 na modulação MFM são representados por impulsos que podem ocorrer no início ou no meio da célula. Por sua vez, a existência e a posição destes impulsos, pode ser representada através dos sinais binários como representado na Tabela 5.6.

³As aplicações de conversão de formatos de imagem podem ser encontradas na página do projecto *HxC Floppy Emulator*: http://hxc2001.free.fr/floppy_drive_emulator/

Tabela 5.6: Formas de onda MFM e a sua representação binária em ficheiros de imagem.

Sinal MFM			
Codificação “.mfm”	0 0	0 1	1 0
Codificação Usada	1 1	1 0	0 1

No caso dos ficheiros “.mfm”, a existência de um impulso é indicada pelo bit 1. Assim, um sinal binário **1**, que na modulação MFM é indicado por um impulso no início da célula, é codificado pela sequência de sinais **1 0**.

Este tipo de codificação dos dados da disquete apresenta ainda o benefício de processar os sinais de sincronismo dos sectores de dados. Caso esta conversão fosse realizada em tempo real pelo emulador, este teria de distinguir os sinais de sincronismo dos sinais correspondentes ao campo de dados de forma a alterar o sinal gerado no primeiro caso.

Os ficheiros “.mfm” possuem cabeçalhos com informação referente à disquete a ser emulada o que faz com que o tamanho do ficheiro ultrapasse o espaço disponível na memória externa. Esta informação não é necessária na memória externa (Secção 5.5) e os dados relevantes (como a velocidade de transmissão) podem ser enviados através de outros comandos.

Como o tamanho das faixas é constante para cada formato, basta guardar a localização do início de cada faixa, na memória interna do emulador ficando assim os dados da memória externa guardados sequencialmente.

De forma a simplificar ainda mais o processamento efetuado pelo Emulador, a informação a ser armazenada no Emulador é negada e assim um impulso é representado pelo bit **0** em vez do bit **1** dos ficheiros “.mfm”. Isto permite que para os sinais MFM os impulsos sejam automaticamente gerados com o bit **0** uma vez que normalmente o sinal está no nível mais alto. No Emulador implementado, o sinal binário **1** é guardado como **0 1**. De notar que para cada bit do sinal MFM são necessários guardar 2 bits em memória.

5.5 Memória de armazenamento

De modo a guardar a informação necessária à emulação da disquete, referida na secção 5.4, é necessária recorrer a uma memória cuja capacidade mínima é calculada da seguinte forma: ao se utilizar disquetes de 1.44 MB, são necessários um total de 16 Megabits de dados dos quais apenas 11 Megabits são dados do utilizador. Os restantes são utilizados nos sinais de sincronização. Para armazenar todos os dados da disquete são então necessários 32 Megabits de armazenamento. Numa disquete de 720 kB, este espaço desce para metade: 16 Megabits.

Considerando que o módulo *SoC* utilizado tem uma memória Flash limitada a 128 kB, esta não é suficiente para alojar todos os dados referentes à imagem da disquete. Por outro lado, a tecnologia sem fios utilizada não permite que estes dados sejam enviados em tempo real do computador anfitrião. É assim necessário recorrer a uma memória externa para alojar os dados.

Na escolha da memória a utilizar, para além da capacidade, é necessário analisar um protocolo de comunicação. A dificuldade em utilizar memórias de grande capacidade, não se prende com a sua inexistência. Hoje em dia são facilmente acessíveis cartões de memória com vários GigaBytes de capacidade, no entanto o formato em que é feita a comunicação é, neste caso, limitador.

Foram consideradas memórias em cartões *Secure Digital* pelo seu baixo preço e facilidade de aquisição e utilização. Estes cartões além do protocolo proprietário SD podem também comunicar usando um protocolo simplificado via SPI [96]. Apesar de ser uma solução interessante, a sua utilização foi descartada pois o protocolo de comunicação inclui bytes de controlo de erros que interferem com o processo de envio de dados do Emulador para o controlador de drive de disquetes. Foram assim consideradas memórias *flash* que suportem o protocolo SPI.

Na implementação do protótipo foi utilizada a memória AT25DF161 da Atmel [97–99]. Estas memórias alojam 16 Mbits de dados mas existem versões de 32 e 64 Mbits. Mas outras memórias de outros fabricantes implementam protocolos muito similares e podem ser facilmente adaptadas. São exemplos as memórias Winbond Serial Flash W25Q32BV [100] ou as Microchip SST25VF032B [101].

5.6 Transmissão da imagem

A característica diferenciadora deste Emulador em relação aos existentes é a sua capacidade de receção dos dados através de uma interface sem fios. Esta funcionalidade é conseguida através

do envio de pacotes de dados do Gestor para a memória externa do Emulador através do dispositivo Coordenador. Todos estes dispositivos são chamados a intervir na sequência de comandos necessários à transmissão. O diagrama simplificado do processo de envio de uma página de dados é apresentado na Figura 5.14.

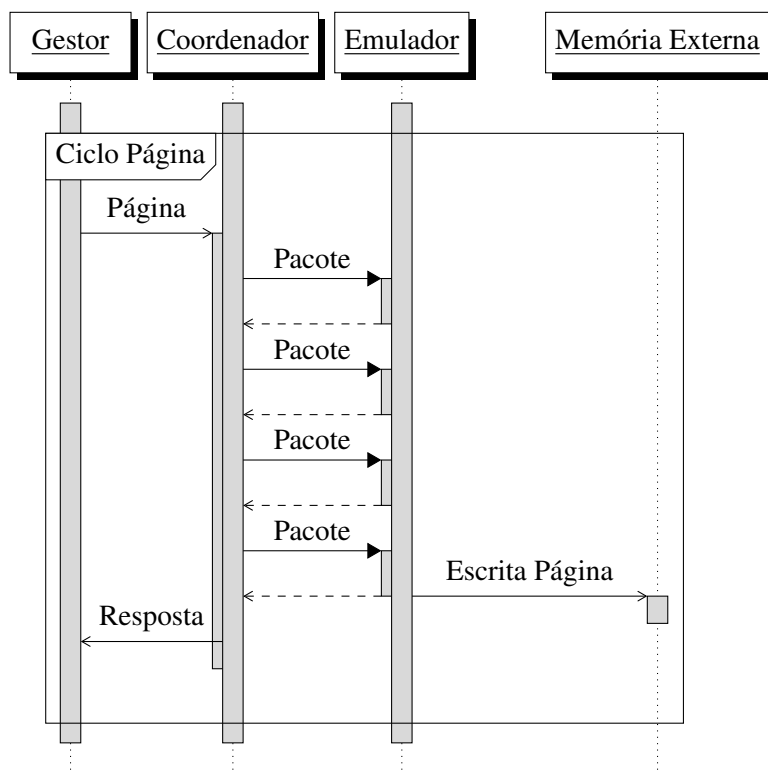


Figura 5.14: Envio de imagem: Diagrama de sequência de mensagens entre Gestor, Coordenador, Emulador e Memória

O processo de transmissão é iniciado no Gestor e este é responsável por toda a gestão do envio dos dados. Como referido na secção 5.4, apenas os dados relevantes são guardados na memória externa e como tal, é necessário o seu processamento prévio. O Gestor extrai a informação necessária do ficheiro “.mfm” descartando o cabeçalho e espaços entre as faixas. Prossegue também à negação dos octetos como referido na secção 5.4.

Como cada um dos atores envolvidos neste processo utiliza tecnologias diferentes, o envio da imagem tem que ser adaptado a cada uma das tecnologias.

No caso do processo da escrita na memória externa, ao se consultar a folha de características, verifica-se que a operação de escrita é efetuada em páginas de 256 bytes [98]. De forma a otimizar este processo, o Emulador necessita então de agrupar 256 bytes de dados antes de os enviar, via porta SPI, para memória. A escrita dos dados será então executada em blocos de 256 bytes referentes à página da memória a ser escrita.

Em cada ciclo de 256 bytes, o gestor envia para o Controlador o comando de escrita na memória, o endereço onde os dados devem ser escritos e os 256 bytes de dados num comando, através da porta série virtual. O Controlador ao receber os dados necessita de os adaptar de forma a serem enviados através do canal de comunicação sem fios.

Como referido na secção 4.4, a interface de programação fornecida pela Z-Stack, limita o campo de dados que a aplicação pode utilizar a 85 bytes. Desta forma, é necessário partir a trama de 256 bytes de dados em blocos que possam ser inseridos nos pacotes rádio. São então formados 4 pacotes que contêm o endereço dos dados e 64 bytes de dados. O Controlador envia o pacote para o Emulador e espera pelo evento gerado pela Z-Stack, que indica que a mensagem foi enviada com sucesso antes de enviar o pacote seguinte. Quando os 4 pacotes são enviados, o Controlador sinaliza ao Gestor que está pronto para o envio da página seguinte.

O processo do envio de pacotes entre o Controlador e o Emulador revelou ser o mais lento de toda a cadeia de comunicação. Entre o envio de um pacote e a ativação do evento que sinaliza o envio desse pacote com sucesso registaram-se valores na ordem dos 200 milissegundos. Assim, o intervalo entre o envio de duas páginas consecutivas pelo gestor demora, média, 850 ms.

Numa disquete de 720 kB, em que são transmitidos 2 milhões de bytes que compõe a imagem, o tempo de transmissão cifra-se em cerca de duas horas. No caso das disquetes de 1.44 MB, este valor duplica. Estes valores não são comportáveis com a aplicação pretendida e, como tal, foram feitas algumas tentativas no sentido de diminuir este tempo.

Uma simplificação da lógica de envio em conjunto com a geração dos eventos associados no dispositivo coordenador resultou na diminuição do tempo entre o envio de pacotes ZigBee para um valor mais próximo dos 150 ms. No entanto este valor é ainda alto e a transmissão de uma imagem de 2 milhões de bytes demora, então, aproximadamente 1 hora e 30 minutos.

Uma redução do número de pacotes a transmitir por cada página revelou-se impossível pois mesmo sem o envio da informação do endereço, 3 pacotes de 85 bytes não são suficientes para enviar 256 bytes.

Procedeu-se então à realização de testes que dispensam ou diminuem o tempo de espera pela confirmação da receção. No entanto, neste caso, o número de erros de transmissão aumentava de forma significativa o que exigia o reenvio dos dados tornando o processo ainda mais lento.

Considerando que a transmissão sem fios é afetada por outros sinais que podem ocupar o mesmo espectro radio-elétrico, torna-se difícil isolar o motivo destes atrasos. Tentativas de isolamento do Controlador e do Emulador de outras redes sem fios e a alteração do canal 802.14.5 utilizado não trouxeram quaisquer melhorias nestes tempos.

5.7 Implementação Protocolo MFM

O principal desafio na implementação do Emulador está associado ao protocolo MFM, utilizado na comunicação de dados entre o Emulador e o controlador de disquetes.

O protocolo MFM teve uma aplicação limitada no uso e no tempo pelo que não existem moduladores destes sinais em hardware como os existentes para SPI ou I2C do CC2430. Estes protocolos são usados em muitas aplicações diferentes o que justifica a existência de tais bibliotecas e circuitos dedicados. Não existindo esta facilidade, é então necessário recorrer a outras técnicas para criar as formas de onda que respeitem as especificações referidas na secção 3.3.1.

Um outro aspeto que é preciso analisar e que influencia de forma significativa a geração dos sinais MFM é a extração dos sinais a serem enviados da memória. Se o processo de aquisição de um novo dado for demasiado lento, a transmissão fica comprometida.

Nesta análise, são usadas como referência as disquetes de 3½ polegadas de alta densidade, que à velocidade de 300 rotações por minuto, debitam dados a 500 kbps, ou seja, 1 novo bit a cada 2 microssegundos. Como não é utilizado um sinal de sincronismo, mas sim os tempos entre impulsos sucessivos, regista-se a necessidade de garantir o envio sem atrasos, uma vez que estes que poderiam ser erroneamente interpretados como sinais legítimos, apesar de errados.

O desafio, é então, dividido em três partes:

- Extrair da memória os dados que se pretende enviar;
- Se necessário, processar a informação;
- Operar o sinal MFM na temporização e codificação certas.

Não existindo o hardware dedicado para a geração de sinais MFM, são analisadas soluções para a geração do sinal pretendido. A primeira é a utilização de hardware dedicado para a modulação do sinal; a segunda é a adaptação de uma modulação existente tais como o protocolo SPI ou a técnica PWM; a terceira é a manipulação direta dos pinos, uma técnica conhecida por *Bit-Banging*. Esta secção, irá descrever alguns testes realizados durante a implementação até à solução final.

5.7.1 Hardware dedicado

Foi contemplada a possibilidade de criação de um módulo em hardware que realizasse a modulação do sinal. Um módulo CPLD receberia na entrada um byte de dados e apresentaria na saída

o sinal MFM com a temporização e forma corretas. Esta implementação teria certamente algumas vantagens ao funcionar de forma independente do microcontrolador e poderia também controlar todo o formato de faixa, gerando os respetivos sinais de controlo, sendo apenas necessário fornecer os campos de dados. No entanto esta solução foi posta de parte pois implicaria a aprendizagem de uma nova tecnologia e aumentaria o custo da solução. Nas fases iniciais do projeto, ainda não eram conhecidas totalmente as capacidades do microcontrolador e das dificuldades que iriam surgir na implementação da modulação do sinal.

5.7.2 Sinal auxiliado por hardware

As formas de onda pretendidas podem ser conseguidas através da manipulação de sinais gerados pelas funções em hardware do microcontrolador CC2430, como o canal UART na configuração SPI e funções de geração de sinais PWM.

O SPI é um protocolo de comunicação síncrono que recorre a um sinal de relógio para transmitir informação para equipamentos periféricos de forma série. Neste projeto, o protocolo SPI é utilizado na comunicação entre o microcontrolador e a memória externa. Noutros contextos, ao manipular os dados a serem enviados, manipula-se a forma de onda do sinal gerado. Escolhendo a temporização e os octetos de dados cuidadosamente, este sinal pode replicar o sinal MFM pretendido.

Consultando a folha de especificações do microprocessador CC2430, verifica-se que a frequência máxima possível é dada pela Equação (5.1) onde F é a frequência de relógio.

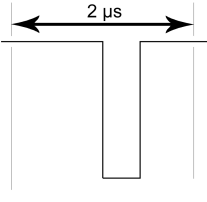
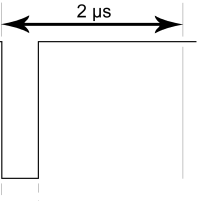
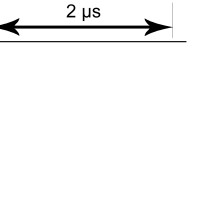
$$F_{SPI} = \frac{1}{8} * F \quad (5.1)$$

No caso dos módulos utilizados essa frequência é de 32 MHz. Assim, a frequência máxima do canal SPI é de 4 MHz que é, justamente, a frequência necessária para transmitir cada bit de informação como calculado na Equação (5.2).

$$\frac{1}{4MHz} * 8bits = 2\mu s \quad (5.2)$$

A duração de cada byte da transmissão SPI corresponde ao tamanho da célula que cada bit MFM a 500 kbps ocupa. Assim, a célula poderá ser modulada através de um byte SPI que replique a forma de onda desejada, tal como os apresentados na Tabela 5.7.

Tabela 5.7: Octetos SPI a simular bits na modulação MFM

Bit MFM	1	0 (Após um 0)	0 (Após um 1)
Forma de onda MFM			
Octeto dados SPI	0xF3	0x3F	0xFF

Se o *buffer* de envio foi preenchido de forma célere o suficiente, então é possível o envio dos sinais dentro das temporizações requeridas. Isto pode ser conseguido através da manipulação direta do registo associado à porta UART do microprocessador ou através do uso de uma DMA (*Direct Memory Access*) que alimenta de forma automática este registo.

A utilização da porta SPI apresenta no entanto algumas desvantagens: apesar de só interessar um pino de saída MOSI (*Master Out, Slave In*) da porta SPI, o microprocessador usa também mais dois pinos das suas portas de Entrada/Saída, um de relógio e outro de entrada MISO (*Master In, Slave Out*), inutilizando assim alguns pinos necessários para a comunicação com o controlador. Imediatamente se verifica que para cada bit que se pretende enviar, é necessário enviar um *byte* de dados multiplicando assim por 8 a quantidade de dados a serem armazenados.

Neste caso, o uso da DMA é preterido uma vez que obriga ao armazenamento de um volume muito maior de dados. O uso da DMA também se verifica problemático uma vez que esta limita a fonte de informação à memória interna do microcontrolador. O envio da sequência dos dados MFM é ininterrupto pelo que seria necessário guardar toda a sequência de dados referentes a uma faixa de dados inteira na memória interna, como referido na secção 3.4. A memória interna do microcontrolador, não é, portanto adequada ao volume de informação que se pretende processar.

Os módulos CC2430 (e o módulo alvo deste projeto, RC2300) possuem 128 kb de memória flash que é partilhada pelo código e dados de utilizador. Esta memória pode ser acedida através de funções da Z-Stack ou diretamente através da DMA. É, no entanto, de notar que não é aconselhável a escrita repetida nesta memória devido à degradação provocada pelas sucessivas escritas [88] para além do facto da operação ser morosa. Neste módulo, devido ao modo de funcionamento do canal DMA, não foi possível criar um canal de comunicação direto entre a porta SPI da memória externa e a porta SPI que simula o sinal MFM. Abandonou-se assim a solução com a tecnologia DMA.

Com a porta SPI como geradora de sinais testou-se ainda a possibilidade de alimentar o registo da porta de forma manual. Esta solução permitiria o processamento dos dados em tempo real sem necessidade de armazenar um volume de dados muito maior.

Seria então necessário o processamento dos dados e o envio do byte correspondente para a porta UART a cada 2 μ s. Apesar da referida permissividade a desvios de alguns nanossegundos, não foi possível escrever na porta SPI com a cadência desejada devido a atrasos que ocorriam entre bits consecutivos: de forma a garantir a escrita na porta sem erros, é necessário esperar que o indicador UxACTIVE do CC2430, que sinaliza o processo de escrita na porta SPI, seja limpo. No entanto, se se esperar que este indicador seja limpo pelo processador surge um atraso entre dois bytes consecutivos. Este atraso viola as temporizações impostas pelo protocolo MFM. Se se ignorar o indicador UxACTIVE e efetuar a escrita antes que o envio seja concluído, podem ocorrer perdas de dados.

De forma similar à utilização da porta SPI para gerar as formas de onda, foi também testada uma solução que recorre ao gerador de sinais PWM (*Pulse Width Modulation*) para desenhar os impulsos do protocolo MFM. Programando o contador PWM para desenhar impulsos de 400 ns de largura a cada 1 μ s, pode sobrepor-se a este sinal um outro, que filtra os impulsos que resultam na forma de onda pretendida. Este filtro seria gerado por um outro pino do microcontrolador e teria de ser atuado de forma a remover os impulsos que não se pretendem transmitir. Este sinal, operado a cada 1 μ s, seria conjugado com uma porta lógica **AND** para resultar no sinal MFM pretendido.

O uso do sinal PWM exige ao microprocessador que filtre o sinal a cada 1 μ s, o que apresenta apenas uma pequena vantagem face à operação direta do pino. Este sistema por sua vez, apresentou outros problemas nomeadamente na dificuldade em sincronizar o sinal PWM gerado com o sinal de filtragem.

Tendo as tentativas de utilização de sinais gerados por hardware não obtido os resultados pretendidos, o desenvolvimento da aplicação foi orientado à manipulação do pino MFM diretamente.

5.7.3 *Bit-Banging*

De forma a analisar a capacidade de comutação dos pinos do microcontrolador foi efetuado um teste à sua velocidade de comutação: Através do código da Listagem 5.1 é obtido o resultado registado na Figura 5.15. O teste revela que o CC2430 consegue comutar o pino em menos de 100 ns mas que este valor pode, em alguns casos, variar. Apesar da comunicação MFM depender do tempo entre sinais para descodificar os dados, este também é algo permissivo a pequenas variações se estas não influenciarem a forma de onda final. O impulso do sinal MFM tem normalmente 400 ns de largura, mas foram registados casos em disquetes reais com impulsos menores e maiores, bem como desvios entre os sinais de algumas centenas de nanossegundos, sem que tenham ocorridos erros. Existe assim alguma margem de manobra nesta operação.

Listagem 5.1: Operação direta do pino MFM

```
1 #define HIGH    1
2 #define LOW     0
3 #define Pin_MFM 0x90.0
4
5 int main( void ){
6     Pin_MFM = HIGH;
7     Pin_MFM = LOW;
8     Pin_MFM = HIGH;
9     Pin_MFM = LOW;
10    Pin_MFM = HIGH;
11    Pin_MFM = LOW;
12    Pin_MFM = HIGH;
13    Pin_MFM = LOW;
14 }
```

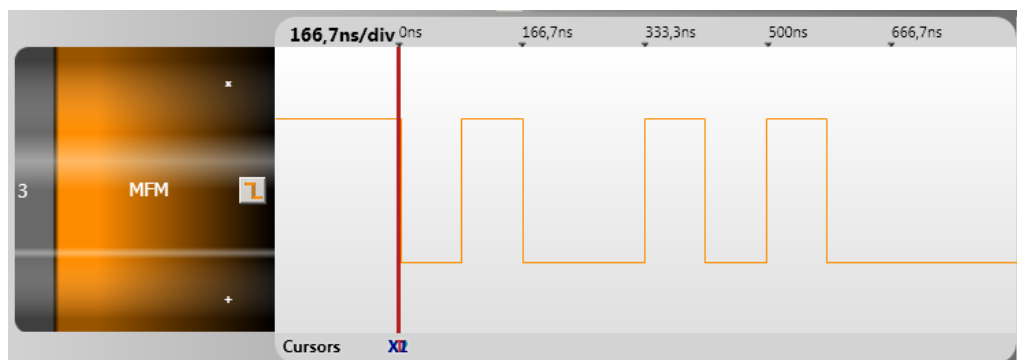


Figura 5.15: Pino de saída em resposta aos comandos efetuados na Listagem 5.1 .

No sentido de se obter tempos mais precisos, foi testada uma solução que recorre ao uso dos temporizadores do CC2430. O temporizador dispara a rotina de interrupção a cada 2 μ s (a duração de cada bit na codificação MFM) e esta rotina é responsável pelo envio do bit correspondente.

Esta solução também se verificou problemática pois surgiram dificuldades em colocar todo o processamento necessário dentro do tempo disponível na rotina de interrupção e começaram a surgir atrasos entre as interrupções. Verificou-se que o método de verificação de bit tinha um peso demasiado grande na rotina de interrupção uma vez que envolvia a extração do bit a ser processado, do byte recebido via SPI e a sua comparação com o bit enviado anteriormente.

Com as alterações ao formato “.mfm” que dispensam o processamento dos bits a enviar, o requisito de temporizações fica mais curto e deverá ocorrer a cada 1 μ s. O microcontrolador não se registou rápido o suficiente para efetuar as interrupções a este ritmo e assim esta solução foi também abandonada.

De forma a otimizar este processo, o projeto foi orientado de forma a implementar o envio dos dados em código *assembly* juntamente com o pré-processamento de dados como referido na

secção 5.4.

5.7.4 Solução em *Assembly*

A solução final do modulador MFM foi então concretizada em rotina separada, escrita em *assembly*, de forma a obter um maior controlo das temporizações do microcontrolador.

Ao entrar no processo de envio, a rotina responsável deverá correr continuamente, sendo apenas interrompida pelas interrupções geradas pelos eventos nos pinos de sinal que transmitem os comandos do controlador da drive de disquetes, e por um temporizador que reinicia o envio da faixa a cada 200 ms, ou seja, simula uma rotação do disco magnético a 300 rpm. A rotina descrita implementa o processo “Envio da faixa” referido na Figura 5.7.

Este controlo do Emulador sobre a rotina de envio de faixa é efetuado através de variáveis globais que a rotina em *assembly* consulta periodicamente:

- **spiAddr** Endereço da faixa a ser enviada;
- **stopMFM** Sinaliza que a rotina deve terminar;
- **restartMFM** Sinaliza que a rotina deve recomençar o envio de dados;
- **pauseMFM** Sinaliza que a rotina deve suspender o envio de dados.

Ao ser chamada, a rotina perfaz a inicialização das variáveis mencionadas. Inicia também o comunicação SPI através da ativação do sinal *Chip Select* da memória externa e o envio do comando para a leitura dos dados da mesma. O endereço da memória onde a faixa a ser enviada está presente é indicado pela variável **spiAddr**.

De forma a não interromper a sequência de envio de dados, a manipulação do sinal de /INDEX está também ao cargo deste código. O sinal é ativado no início da faixa e uma variável conta o numero de bytes que são enviados. Quando este valor atinge o valor programado, o sinal /INDEX é desativado. A largura do impulso é definida pela Equação (5.3). Inicializando o valor do contador a 256 é gerado um sinal /INDEX que fica ativo durante aproximadamente 4 ms.

$$\text{Contador} * 16\mu s = \text{Largura Impulso} \quad (5.3)$$

O controlo de todas as operações efetuadas pelo microcontrolador fica a cargo do código da rotina que deverá ter em conta a duração de cada comando, para respeitar a distancia entre bits do

sinal MFM. Quando se implementou o código *assembly* recorrendo a ciclos, registaram-se alguns desvios nas temporizações que impediram o seu uso.

O ciclos são implementados através de um contador num dos registo do processador que é inicializado com um valor que é, por sua vez, progressivamente decrementado. Quando o registo atinge o valor zero, uma ação diferente pode ser executada. No entanto, verificou-se que estas ações têm pesos temporais diferentes.

Estes contadores podem ser utilizados para implementar ciclos repetitivos como, por exemplo, o envio de 8 bits que compõem um byte. Neste caso, iniciar-se-ia o registo com o valor 8 e no final de cada bit o registo seria decrementado. No caso de valores maiores do que os suportados por um registo do contador, pode recorrer-se ao uso de 2 contadores, um dentro do outro, para executar a contagem de, por exemplo, os 25000 bytes de informação que perfazem a faixa a enviar.

Ao implementar um destes ciclos para o envio de cada bit, afinavam-se as temporizações de modo a que cada bit respeitasse a janela mas a diferença de tempo entre o último bit de um byte e o primeiro bit do byte seguinte registava-se superior às diferenças entre bits do mesmo byte. O uso de vários destes ciclos levou ao acumular de atrasos ao ponto de a distância entre bits consecutivos ser superior a uma janela temporal que se considerou segura (500 ns correspondentes 1/4 de bit).

A solução passou então por reduzir ao máximo a existência destes ciclos sendo apenas utilizados dois: o contador usado pelo sinal /INDEX, e um ciclo por cada byte a ser enviado. Cada bit deste ciclo é processado individualmente e não é contabilizado o numero de bytes a enviados em cada faixa.

O espalhamento do código criado por este problema permitiu, por outro lado, a execução de um código diferente no intervalo referente a cada um dos bits, o que aumentou a sua versatilidade e permitiu o teste às diferentes variáveis de controlo diferentes dentro do ciclo.

No início do ciclo do byte a ser enviado, este é lido do registo associado à porta SPI e o primeiro bit é enviado para o pino de saída. Se este bit tiver o valor 0, a escrita deste valor no pino de saída vai criar um impulso no sinal MFM. O valor da variável **restartMFM** é carregado no acumulador. De seguida, o impulso no pino MFM é terminado ao colocar o pino no nível lógico 1 e finalmente o valor da variável carregada (neste caso **restartMFM**) é testado antes de iniciar o envio do bit seguinte. Durante o envio do segundo bit, o bit 1, outra variável como a **stopMFM** é testada. Durante o bit 2 a variável **pauseMFM** é carregada e testada. Nos bits seguintes estas variáveis são novamente testadas de forma cíclica. Durante o envio do bit 4, a meio da sequência, é pedido um novo byte de dados da memória através da escrita de um byte na respetiva porta. Nos bits seguintes, as variáveis são continuamente testadas como nos primeiros bits. Se alguma das

variáveis de controlo for testada e tiver um resultado positivo a rotina altera o seu comportamento inviabilizando a sequência de dados que estava a ser enviada.

A variável **pauseMFM** coloca a rotina num estado de suspensão até que o comando de reinício seja emitido. Esta pausa é necessária para não enviar para o controlador de drives de disquetes informação que não corresponde ao estado que está à espera. Ocorre geralmente numa operação de mudança de faixa ou de cabeça de leitura.

No caso do reinício de faixa, cada vez que a interrupção do temporizador é ativada, a variável **restartMFM** é sinalizada com o valor lógico **TRUE** e, quando a interrupção termina e a rotina de envio de dados retoma a sua execução, encontrará este sinal ativo e procederá ao reinício da operação. Caso o endereço da faixa tiver sido alterado, este será automaticamente enviado para a memória externa quando o processo se iniciar.

Finalmente o comando de **stopMFM** instrui a rotina para terminar a sua operação e devolver o controlo do processador à Z-Stack.

Uma das dificuldades que ocorreu na implementação do código está associado ao aumento do tamanho do código que foi produzido por não se recorrer a ciclos. O código ficou mais longo e como tal, os comandos *DJNZ* do microprocessador, que efetua um salto entre dois pontos do código, não suportam que estes pontos estejam estão distantes. De forma a que este comando seja executado com sucesso, foi inserido código de ligação no início e no fim da rotina de envio de dados.

Uma drive com uma disquete de baixa densidade envia os dados a metade do débito em relação a uma disquete de alta densidade. A diferença de velocidades é conseguida através da redução da velocidade de relógio do microprocessador de 32 MHz para 16 MHz. Pequenas variações nas temporizações associadas à largura dos impulsos levaram à construção de duas rotinas separadas, uma para cada velocidade.

Capítulo 6

Resultados

Após a implementação do emulador, a sua validação é realizada através de testes funcionais. Testam-se as duas funcionalidades principais do dispositivo: A emulação da drive de disquetes; e a transmissão sem fios.

6.1 Emulação

A emulação da drive de disquetes é testada através do uso de um computador pessoal em que a drive de disquetes é substituída pelo emulador em teste. O computador de teste corre o sistema operativo Windows XP e possui um controlador de drive de disquetes embutida no controlador genérico da placa mãe que, de acordo com o programa SAMDisk, reporta ser o 82077.

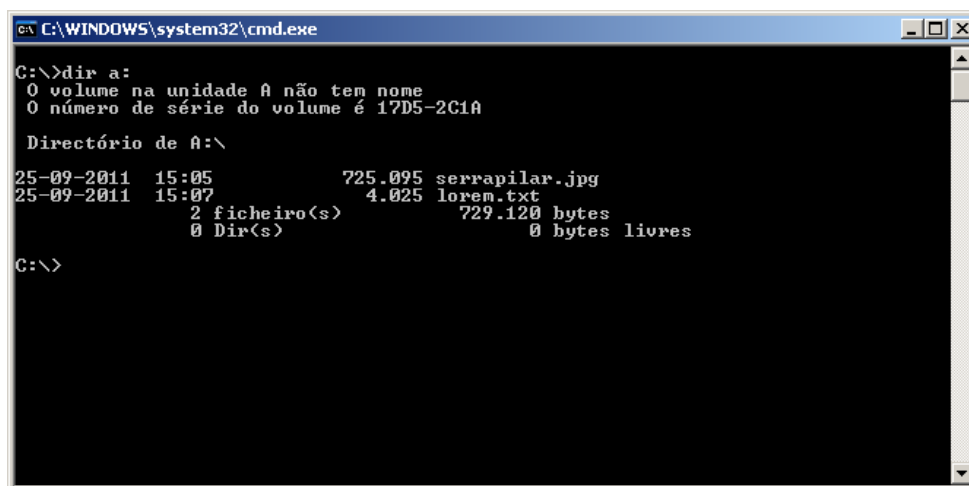
A imagem de testes foi criada com a aplicação WinImage e depois convertida para o formato MFM com as ferramentas referidas na Secção 5.4. A imagem de disco replica uma disquete de 720 kB com dois ficheiros: 1 ficheiro de texto com 4025 bytes e um ficheiro de imagem fotográfica com 725095 bytes. A capacidade da disquete fica assim praticamente completa exigindo que a leitura destes documentos, especialmente do ficheiro fotográfico, percorra toda a disquete.

São realizados dois testes funcionais: O primeiro teste envolve a listagem e leitura do ficheiro existente na disquete através das ferramentas fornecidas pelo sistema operativo.

Usando o programa Explorador que permite a listagem de ficheiros nos discos lógicos do computador, ao escolher a Drive de Disquetes, este lista os ficheiros presentes na disquete.

O ficheiro de texto é aberto na aplicação Bloco de Notas e o ficheiro de imagem é visualizado através de uma aplicação de visualização de ficheiros de imagem.

Também se recorre à aplicação Linha de Comandos para efetuar a operação de listagem.



```
C:\WINDOWS\system32\cmd.exe
C:\>dir a:
O volume na unidade A não tem nome
O número de série do volume é 17D5-2C1A

Directório de A:\
25-09-2011  15:05           725.095 serrapilar.jpg
25-09-2011  15:07           4.025 lorem.txt
                2 ficheiro(s)          729.120 bytes
                0 Dir(s)                  0 bytes livres

C:\>
```

Figura 6.1: Listagem do conteúdo da imagem emulada.

Nos testes realizados verificou-se a ocorrência ocasional de erros quando se tentava aceder à disquete de forma simultânea como tentar ler o ficheiro de texto enquanto o ficheiro de imagem carregava.

Num cenário mais simples, efetuaram-se os testes de leitura em que as operações de listagem e leitura dos ficheiros são efetuadas de forma sequencial. Estas operações foram repetidas 5 vezes e entre cada teste o computador foi reiniciado de forma a evitar que o Sistema Operativo guarde os ficheiros em memória e não recorra à leitura efetiva dos dados do emulador. Em cada um destes testes não foram registados erros.

O segundo teste envolve a operação de criação de uma imagem de dados a partir do emulador. Usando o comando `SAMDisk -c0-79 a: read.raw` obtém-se uma imagem dos dados emulados. Esta imagem, deverá corresponder exatamente à imagem usada para gerar os dados guardados na memória.

A leitura da imagem é efetuada 5 vezes, e nos 5 casos verifica-se que as imagens lidas correspondem à original indicando uma extração de dados sem erros. Para efetuar esta validação é utilizado um programa de comparação de ficheiros binários que sinaliza alguma possível diferença.

6.2 Transmissão sem fios

A aplicação Gestor utiliza um cronómetro no envio da imagem para o emulador que contabiliza o tempo que decorre entre o envio das tramas das 265 bytes para o Coordenador e calcula o seu valor médio. Outro cronómetro calcula o tempo total decorrido no envio da imagem completa.

Registando este valor, verifica-se que a transmissão da imagem de 720 kB de testes demora aproximadamente 1 hora e 25 minutos.

Uma análise aos tempos decorridos entre cada ação do Coordenador, permite verificar em que estados existem maiores maiores atrasos. Na Figura 6.2 e Tabela 6.1 são resumidos os tempos médios entre as ações associadas ao envio da página.

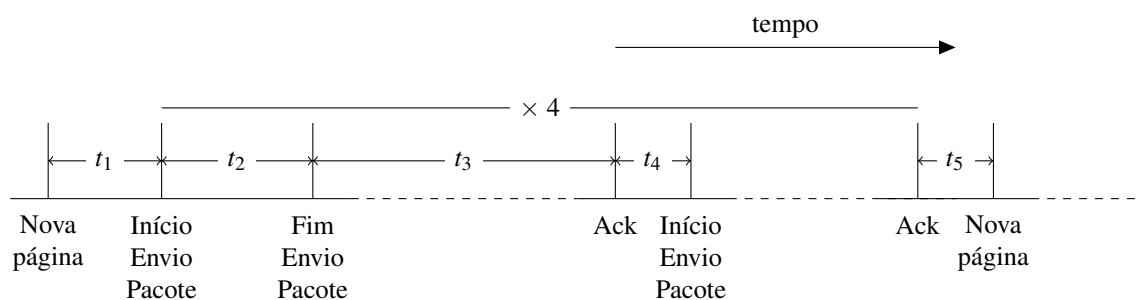


Figura 6.2: Medição de tempos na transmissão sem fios.

Intervalo	Tempo Médio
t_1	0.85 ms
t_2	2.87 ms
t_3	145.32 ms
t_4	0.30 ms
t_5	54.27 ms
Total Página	649.41 ms

Tabela 6.1: Valores temporais médios registados na rotina de transmissão de dados.

Analisando os resultados obtidos na Tabela 6.1, verifica-se que este ocupa a maior parte do seu tempo à espera da sinalização de mensagem enviada com sucesso. Sendo necessárias 4 destes tempos de espera, o tempo de espera totaliza cerca de 580 ms. No total, o envio da página demora 649 ms a ser enviada. Comparando este valor com o tempo total de envio da página médio calculado pelo Gestor, 650 ms, verifica-se que o tempo dedicado à lógica de processamento interno e comunicação série: 70 ms, é muito menor que o tempo despendido durante a espera: 580 ms.

De notar que estes testes são feitos entre um dispositivo Coordenador (*ZigBee Coordinator*) e Encaminhador (*ZigBee Router*) que formam a totalidade da rede. Caso existam outros dispositivos

na rede ou os dispositivos de origem e destino estejam ligados através de encaminhadores, os tempos de envio deverão aumentar ainda mais.

Apesar da comunicação ser efetuada com sucesso, o tempo necessário para a executar é excessivo face à utilização que se pretende do emulador e como tal não satisfaz, totalmente, os objetivos pois não trás uma mais valia face ao tempo requerido pela deslocação física.

Capítulo 7

Conclusões e Trabalho Futuro

A existência de equipamento que recorre a tecnologia obsoleta propõe a sua adaptação para tecnologia mais moderna. Neste trabalho, é implementada uma solução que propõe a substituição da drive de disquetes por um emulador que é equipado com um módulo de transmissão sem fios.

7.1 Conclusões

Foi estudado o funcionamento das drives de disquetes e a sua comunicação com o controlador de drive de disquetes de forma a implementar uma solução totalmente transparente para o equipamento onde é feita a substituição. Foi construído um protótipo do emulador que comunica com o controlador. Os testes realizados revelaram que o equipamento de testes não distingue o emulador de uma drive de disquetes real e que a emulação é realizada com sucesso. No sentido da implementação da emulação foram ainda estudados formatos de condicionamento de dados especialmente orientados à emulação e ao seu armazenamento em memórias. Ao ser dispensado o uso de disquetes e ao se recorrer a memórias flash, a solução apresenta uma melhoria na fiabilidade e longevidade do sistema, devido à tecnologia utilizada nas memórias quando comparada com as disquetes. Uma ferramenta de análise à comunicação entre o emulador e o controlador de drive de disquetes foi criada de forma a auxiliar o desenvolvimento do emulador. A sua estrutura modular permite a sua adaptação a outras ferramentas de captura e outros protocolos de comunicação.

Realizada uma prospeção a outros emuladores existentes, nenhum registou a funcionalidade da transmissão de dados via protocolos de comunicação sem-fios o que aumenta a relevância da solução proposta neste trabalho. O emulador é implementado com recurso a um microcontrolador com transceptor rádio integrado que suporta a comunicação via protocolo ZigBee. De forma a gerir

a comunicação e transmissão sem-fios dos dados para o emulador, é criado um programa de gestão e envio das imagens através de um módulo sem-fios ligado a uma porta USB do computador.

O envio dos pacotes entre o dispositivo controlador e emulador registou ser demasiado lento para a aplicação em causa. Tentativas de otimizar este recurso revelaram-se limitadas. Sendo assim, a implementação da transmissão dos dados via a comunicação ZigBee revelou-se limitadora para os dados que se pretendem transmitir.

Finalmente, considera-se que o projeto representou um desafio alargado que exigiu a aplicação e estudo de diversos campos que compõem o Curso em Engenharia Eletrotécnica. Para além da pesquisa de dados relevantes ao problema, envolveu a construção de programas em 3 linguagens de programação diferentes, desenho e construção de circuitos eletrónicos e análise de engenharia reversa de tecnologias existentes. O protótipo funcional resultante desta tese, embora lento na transmissão de dados, emula com sucesso uma drive de disquetes com dados recebidos por uma ligação sem fios.

7.2 Trabalho Futuro

No seguimento do trabalho realizado neste projeto são mencionados alguns pontos que se consideram relevantes em desenvolvimentos futuros.

Imediatamente se verifica a necessidade da utilização de outra tecnologia sem fios. Serão particularmente interessantes as soluções que recorram à tecnologia IEEE 802.11 que suportam um maior débito. Neste caso, terá de ser considerado cuidadosamente qual a tecnologia a utilizar pois apesar dos débitos elevados da rede, nem todos os equipamentos suportam estas velocidades. Esta alteração implicará também alteração do microcontrolador a utilizar. O novo controlador, caso tenha maior capacidade de processamento, poderá utilizar outra abordagem à emulação e utilizar uma das técnicas mencionadas nas secção 5.7.4 que dispense o tratamento prévio dos dados.

Alternativamente, ou de forma complementar, poderá otimizar-se a transmissão dos dados através do envio parcial dos dados tal como o envio de apenas uma faixa que tenha os dados relevantes. Caso os dados que se pretendam emular possuam um tamanho reduzido, não é necessário recorrer ao envio das faixas vazias. Outra solução, poderá recorrer à compressão dos dados a enviar.

Sugere-se ainda a substituição das memórias flash pela utilização de memórias *Secure Digital* ou de uma *pen* USB para armazenamento dos dados pela sua ubiquidade e versatilidade.

Não tendo sido implementada neste trabalho a possibilidade de escrita por parte do controlador de drive de disquetes, considera-se uma funcionalidade pertinente a ser implementada no futuro. De forma a suportar esta funcionalidade, o hardware do protótipo desenvolvido foi planeado e construído de modo a utilizar uma memória RAM, deixando essa possibilidade em aberto.

Registam-se também algumas alterações a serem incluídas no programa Gestor. Para além da criação de imagens a partir da drive de disquetes real, propõe-se a criação das imagens diretamente a partir de ficheiros existentes no disco, ou da criação de uma drive de disquetes virtual que permita uma utilização mais parecida com o funcionamento real (sem emuladores).

A conversão dos dados para o formato utilizado na emulação deverá ser implementada internamente de forma a não depender de programas externos. Torna-se possível a conversão de dados parciais, tal como de uma faixa, e não só de imagens de disco integrais.

Será ainda interessante a implementação de um sistema que permita a gestão de múltiplos emuladores e do envio parcial dos dados, como por exemplo, o envio de apenas uma faixa de dados.

Finalmente, referem-se melhorias a efetuar nas aplicações auxiliares que poderão simplificar e acelerar consideravelmente o desenvolvimento de novos emuladores.

A aplicação usada na análise da comunicação entre coordenador e emulador *MFM Log Parser* poderá ser adaptada para funcionar com outros equipamentos de captura e com outros formatos de ficheiros. A sua estrutura modular deverá facilitar esta transformação. De forma a melhor facilitar a depuração dos dados, a total descodificação dos sinais MFM pode ser implementada, dando assim acesso ao conteúdo dos sectores de dados.

Referente à aplicação de programação das memórias, poderão ser implementadas ferramentas que permitam um controlo mais refinado sobre os dados presentes na memória. A alteração do conteúdo da imagem na memória seria feita diretamente através do programa sem necessidade de criação de um ficheiro de imagem e a sua conversão para o formato MFM.

Anexo A

Esquemático e desenho da PCB do Emulador

Apresentam-se os diagramas de esquemático e o desenho da placa de circuito impressa criada para realização da emulador. Esta placa funciona como extensão à placa SoC_BB através da interligação de 2 Conectores IDC-PCB 20 pinos.

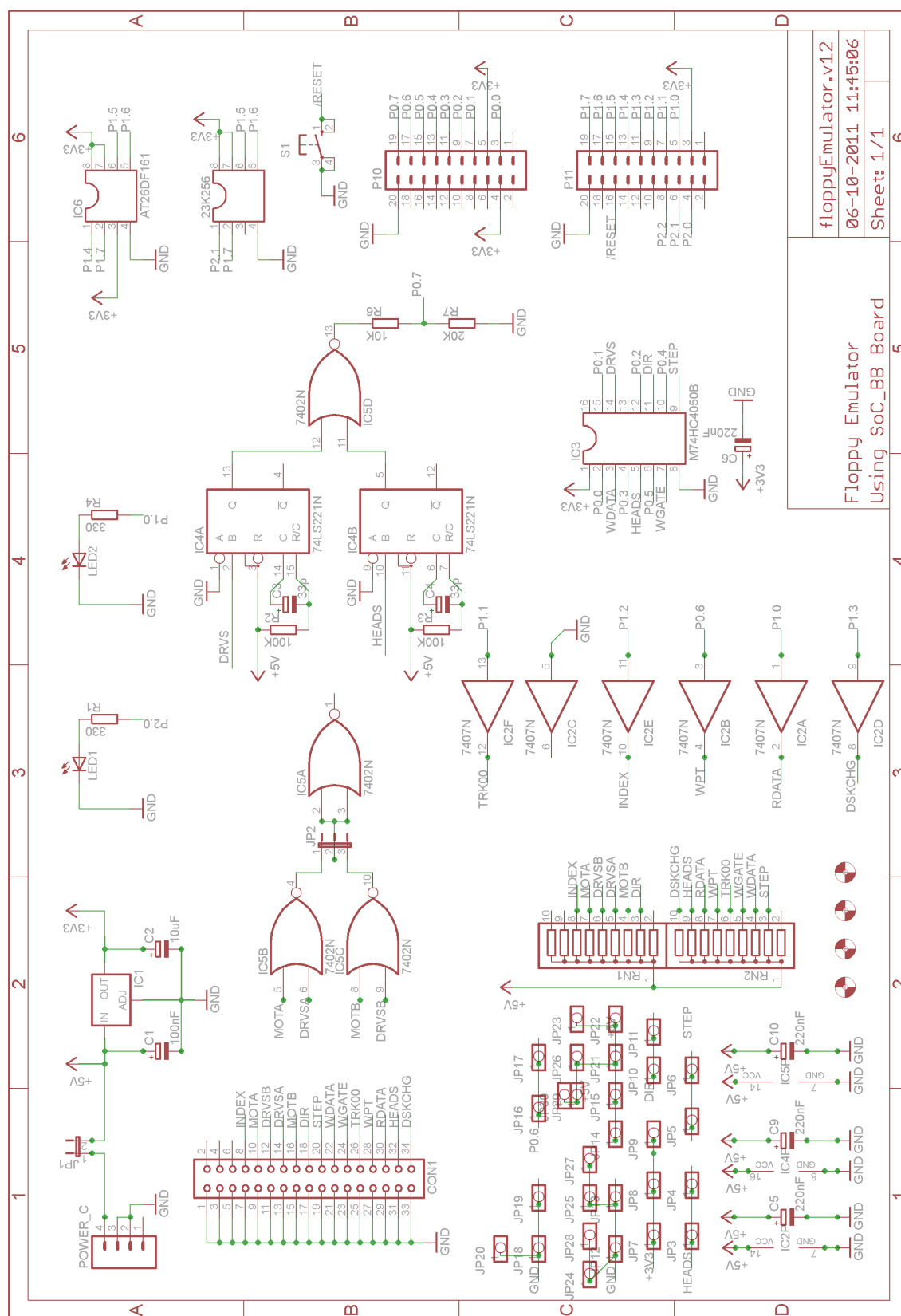


Figura A.1: Esquemático da placa de extensão ao módulo SoC_BB do dispositivo Emulador.



Figura A.3: Protótipo do emulador ligado ao módulo SoC_BB

Referências

- [1] George P. Landow. Jacquard Punch Card Loom. Disponível em WWW: <URL: <http://www.victorianweb.org/technology/textiles/jacquard2.html>>.
- [2] Inc SWF East. SWF/E-902DF Dual Function Two-head, Nine Needles. Disponível em WWW: <URL: http://www.swfeast.com/en/pdf/SWF_E902DF.pdf>.
- [3] LTD Brother Industries. BE-1204B-BC / BE-1206B-BC Instructions Manual. Disponível em WWW: <URL: http://www.brother-usa.com/ModelDocuments/Consumer/Instruction%20Manual/IM_BE_1204B_1204C_1206B_PC_EN_2077.PDF>.
- [4] BE-1201B-AC [Em Linha], Brother International Corporation. Disponível em WWW: <URL: <http://www.brother-usa.com/IndustSewEmb/ModelDetail.aspx?ProductID=BE1201BAC>>.
- [5] HAPPY HCG-Series Multi-Head Machines [Em Linha], TEXMAC. [Consult. Setembro 2011]. Disponível em WWW: <URL: http://www.happyemb.com/infopage_multi_hcg.htm>.
- [6] Multi-head Automatic Embroidery Machines. Disponível em WWW: <URL: <http://www.theembroiderywarehouse.com/brochures/Tajima%20TMEG.pdf>>, 1988.
- [7] LTD Brother Industries. UM ULT 2001 Operation Manual. Disponível em WWW: <URL: http://www.brother-usa.com/ModelDocuments/Consumer/Users%20Manual/UM_ULT_2001_EN_200.PDF>.
- [8] Milltronics Inc. *VKM4 - 3 Axis CNC The Knee Mill with MillSlide*, 2011.
- [9] HEIDENHAIN Corporation. *MILLPWR 3 Axes Operation Manual*.
- [10] ABB Robotics Products AB. *ABB Product On-line Manual IRB 1400*.
- [11] Digital Real-Time™ Oscilloscope TDS340S [Em Linha], Tektronix. [Consult. Fevereiro 2010]. Disponível em WWW: <URL: <http://www2.tek.com/cmswpt/psdetails.lottr?ct=PS&cs=Data+Sheets&ci=14415&lc=EN>>.
- [12] Agilent 35670A Dynamic Signal Analyzer Data Sheet [Em Linha], Agilent, 2009. [Consult. 8 Fevereiro 2010]. Disponível em WWW: <URL: <http://cp.literature.agilent.com/litweb/pdf/5966-3064E.pdf>>.

- [13] A history of the Atari ST and Commodore Amiga [Em Linha], Mark Miller, Maio 2008. Disponível em WWW: <URL: <http://tekkie.wordpress.com/2008/05/16/a-history-of-the-atari-st-and-commodore-amiga/>>.
- [14] Planet Sinclair - Sinclair Computers [Em Linha], Chris Owen. Disponível em WWW: <URL: <http://www.nvg.ntnu.no/sinclair/computers/computers.htm>>.
- [15] Right side view of 1040ST [Em Linha], Dave Dunfield. [Consult. Setembro 2011]. Disponível em WWW: <URL: <http://www.classiccmp.org/dunfield/atarist/index.htm>>, Imagem cedida por Dave Dunfield.
- [16] John C. Dvorak. Major Apple II Upgrade, Finally! *InfoWorld*, 7(25), 24 Jun 1985.
- [17] Central Point Software Incorporated, Portland, OR. *COPY II 64/128*, 1981.
- [18] PSR620 61-Key Full-Size MIDI Keyboard with Touch Sensitive Keys [Em Linha], Yamaha. [Consult. Fevereiro 2010]. Disponível em WWW: <URL: <http://www.yamaha.com/yamahavn/CDA/ContentDetail/ModelSeriesDetail/0,,CNTID%253D838%2526CTID%253D205100%2526CNTYP%253DPRODUCT,00.html?CNTID=838&CTID=205100&CNTYP=PRODUCT>>.
- [19] Jands Electronics Pty ltd. *Jands Hog 500 - Technical Specification Sheet*.
- [20] Inc TEAC America. *External USB Floppy Drive*, 2003.
- [21] David O. Savey e Donald M. Beelenger et al James H. Gibbons. Apparatus for using flash memory as a floppy disk emulator in a computer system, 1995.
- [22] Alan Heminger e Steven Robertson. Digital Rosetta Stone: A conceptual Model for a Maintaining Long-term Access do Digital Documents, 1998.
- [23] Paul Conway. Preservation in the Digital World. Technical report, Yale University Library, 1996.
- [24] Adrian Brown. *Digital Preservation Guidance Note 3: Care, Handling and Storage of Removable media*. The National Archives, Agosto 2008.
- [25] Intel. Interfacing a Floppy Disk Drive to an 80C186EX Family Processor, 1993.
- [26] UMC. *UM8272A/A-4 Floppy Disk Controller*.
- [27] Hynix. *GM82C765B Floppy Disk Subsystem Controller*.
- [28] Intel. *82077AA CMOS Single-Chip Floppy Disk Controller*, Maio 1994.
- [29] Inc TEAC America. *Teac FD-235HF-C891 Micro Floppy Disk Drive Specification*.
- [30] Shoreline Electronics. *XIDE-31U USB 3.5"Micro Floppy Disk Drive*.
- [31] Emulateur de lecteur de disquette [Em Linha], Jean F. NERO, 2010. [Consult. 13 Maio 2010]. Disponível em WWW: <URL: http://hxc2001.free.fr/floppy_drive_emulator/index.html>.
- [32] Amiga Floppy Emulator, mark II [Em Linha], Tim Tashpulatov, 2008. [Consult. Setembro 2011]. Disponível em WWW: <URL: <http://www.sensi.org/~tnt23/megadrive/>>.

- [33] Cumulus [Em Linha]. [Consult. Setembro 2011]. Disponível em WWW: <URL: <http://retromaster.wordpress.com/cumulus/>>.
- [34] ipcas GMBH. USB Disketten Emulator. Disponível em WWW: <URL: <http://www.ipcas.de/download/prospekt-datenblatt/fdd2udd-usb-disketten-emulator.pdf>>.
- [35] 3 ½ floppy drive to USB flash drive reader upgrade [Em Linha], PLRElectronics. Disponível em WWW: <URL: http://www.plrelectronics.com/floppy_to_usb.php>.
- [36] Datex. DTX-200 FLOPPY DISK DRIVE EMULATOR. Disponível em WWW: <URL: <http://datex.fr/emulateur/DTX200En.pdf>>.
- [37] MILO. *EMUFDD v2.00*, 2010.
- [38] EMUFDD 3.0 [Em Linha], Cosimo Oliboni. [Consult. Setembro 2011]. Disponível em WWW: <URL: http://embeddedsdsw.net/EMUFDD_Floppy_Hardware_Emulator_Home.html>.
- [39] Sony Corporation. *MSAC-FD2M*, 2000.
- [40] JVC. *VU-V101 FLOPPY DISK ADAPTER INSTRUCTIONS*.
- [41] CU-VFM40 MultiMedia Card Floppy Disk Adapter Driver Update for Windows® 2000 Professional [Em Linha], JVC. Disponível em WWW: <URL: https://www.jvc.ca/en/consumer/service/s_downloads/downloads_FP_UD_E.asp>.
- [42] Andrei Khurshudov. *The Essential Guide to Computer Data Storage: From Floppy to DVD*. Prentice Hall, 2001.
- [43] Lubar Steven. Do Not Fold, Spindle Or Mutilate: A Cultural History Of The Punch Card. *Journal of American Culture*, 1992.
- [44] Louis D. Stevens. The Evolution of Magnetic Storage. *IBM Journal of Research and Development*, 25(5):663–76, 1981.
- [45] C. Mee e Eric Daniel. *Magnetic Storage Handbook*. McGraw-Hill Professional, 1996.
- [46] Shugart. *SA450 Double-sided Minifloppy Diskette Storage Drive*, 1980.
- [47] R.I.P. Floppy Disk [Em Linha], BBC News, 2003. [Consult. 8 Fevereiro 2010]. Disponível em WWW: <URL: http://news.bbc.co.uk/2/hi/uk_news/2905953.stm>.
- [48] Alain Knaaf e David C. Niemi Michael Haardt. *The Floppy User Guide*, 2007.
- [49] Scott Mueller. *Upgrading And Repairing PCs (19th Edition)*. Que, 2009.
- [50] 8-inch, 5,25-inch, and 3,5-inch floppy disks [Em Linha], George Chernilevsky. Disponível em WWW: <URL: http://en.wikipedia.org/wiki/File:Floppy_disk_2009_G1.jpg>.
- [51] 3.5 Diskettes Support - Frequently Asked Questions [Em Linha]. [Consult. Setembro 2011]. Disponível em WWW: <URL: <http://www.imation.com/en-us/Imation-Support/Diskettes/35-Diskettes/Diskette-Storage-Support/>>.

- [52] NEC Corporation. *FD1035 3.5" Floppy Disk Drive Product Description*, 1984.
- [53] Michael Tischer e Bruno Jennrich. Diskettes and Hard Drives. In *PC Intern: The Encyclopedia of System Programming (Developers Series)*, chapter 14. Abacus Software Inc, 1996.
- [54] Introduction to the 82077, Floppy Disk Controller [Em Linha]. Disponível em WWW: <URL: <http://www.intel.com/design/archives/periphrl/docs/7281.htm>>.
- [55] Interface Between 82077AA/SL and the Floppy Drive [Em Linha]. Disponível em WWW: <URL: <http://www.intel.com/design/archives/periphrl/docs/7282.htm>>.
- [56] A J Sammes e Brian Jenkinson. *Forensic Computing: A Practitioner's Guide (Practitioner Series)*. Springer, second edition, 2007.
- [57] Trevor G. Marshall e John A. Attikiouzel. Floppy Disk Data Transfer Techniques. *Micro, IEEE*, 3(6):17 – 23, Dezembro 1983.
- [58] Intel Corporation. *8272A Single/Double Density Floppy Disk Controller*, 1982.
- [59] Citizen Systems Europe, United Kingdom. *Model: X1DE-00R 3.5"Micro Floppy Disk Drive Specification*.
- [60] SMSC. *FDC37C665GT / FDC37C666GT High Performance MultiMode™ Parallell Port Super I/O Floppy Disk Controllers*, 2007.
- [61] Bot Lutz et al. *Application Note 505 - Floppy Disk Data Separator Design Guide for the DP8473*. National Semiconductor Corporation, Fevereiro 1989.
- [62] National Semiconductor Corporation. *DP8473 Floppy Disk Controller PLUS-2™*, Jul 1990.
- [63] James Cecil et al. *Application Note 413 - Disk Interface Design Guide*. National Semiconductor Corporation, Janeiro 1986.
- [64] Inc Intel. *SBC 202 Double Density Diskette Controller Hardware Reference Manual*, 1977.
- [65] Siemens Aktiengesellschaft. *SAB2793B/2797B Advanced Floppy Disk Formatter/Controller Family*.
- [66] Michael Tischer e Bruno Jennrich. DOS File System. In *PC Intern: The Encyclopedia of System Programming (Developers Series)*, chapter 17, pages 379–381. Abacus Software Inc, 1996.
- [67] Institute of Electrical and Inc. Electronics Engineers. IEEE 802.15.1-2002 IEEE Standard for information technology - Telecommunication and information exchange between systems - LAN/MAN - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Wireless Personal Area Networks(WPANs), 2002.
- [68] Texas Instruments. *Wireless Connectivity Guide*, 2011.
- [69] Bluetooth SIG, Inc. *Bluetooth Specification Version 2.1 + EDR*, 2007.

- [70] Building with the Technology: Overview [Em Linha]. [Consult. Setembro 2011]. Disponível em WWW: <URL: <https://www.bluetooth.org/Building/overview.htm>>.
- [71] ZigBee Alliance. ZigBee Specification. Technical report, ZigBee Standards Organization, 2006.
- [72] Texas Instruments, Inc. *HAL Drivers Application Programming Interface*, 2005.
- [73] Texas Instruments, Inc. *Z-Stack OS Abstraction Layer Application Programming Interface*, 2007.
- [74] Texas Instruments, Inc. *Z-Stack Developer's Guide*, 2007.
- [75] Shahin Farahani. *ZigBee Wireless Networks and Transceivers*. Newnes, 2008.
- [76] Radiocrrafts AS. *RC2300/01/02/04 ZigBee™ Ready RF Transceiver Modules*, 2007.
- [77] Radiocrrafts AS. *RC2300/01/02/04 ZigBee™ / IEEE 802.15.4 RF Modules*, 2006.
- [78] Texas Instruments. *CC2431DK Development Kit User Manual*, 2007.
- [79] USBee SX PC and USB Logic Analyzer, Signal Generator, I2C, SPI, Async Protocol Decoder and Analyzer [Em Linha], Inc CWAV, 2010. [Consult. Fev 2011]. Disponível em WWW: <URL: <http://www.usbee.com/sx.html>>.
- [80] Support for the USBee PC and USB Oscilloscope, Logic Analyzer, Signal Generator and more [Em Linha], Inc CWAV, 2010. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://www.usbee.com/faq.htm#%2056>>.
- [81] Bus Pirate [Em Linha], Dangerous Prototypes, 2011. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: http://dangerousprototypes.com/docs/Bus_Pirate>.
- [82] fdrawcmd.sys [Em Linha], Simon Owen, 2010. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://simonowen.com/fdrawcmd/>>.
- [83] The Fdio Project [Em Linha], Ansgar Kückes, 2010. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://www.hp9845.net/9845/projects/fdio/>>.
- [84] SAMdisk [Em Linha], Simon Owen, 2010. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://simonowen.com/samdisk/>>.
- [85] Décio da Costa Macedo. Sistema de Extensão de Sinais IR Usando Tecnologia ZigBee. Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2010.
- [86] Texas Instruments, Inc. *Z-Stack/Z-Tool Serial Port Interface*, 2006.
- [87] Austin Lesea and Rodney Zaks. Floppy-disc controller design must begin with the basics. *Electrical Design News*, pages 129–137, Mai 1978.
- [88] Texas Instruments. *CC2430 A True System-on-Chip solution for 2.4 GHz IEEE 802.15.4 / ZigBee®*, 2008.

- [89] Floppy disk image formats supported by Virtual PC and Virtual Server [Em Linha], Ben Armstrong, 2007. [Consult. Fev 2011]. Disponível em WWW: <URL: http://blogs.msdn.com/b/virtual_pc_guy/archive/2007/01/04/floppy-disk-image-formats-supported-by-virtual-pc-and-virtual-server.aspx>.
- [90] Creating virtual floppy disks in Virtual Server [Em Linha], Microsoft. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://technet.microsoft.com/en-us/library/cc708294%28WS.10%29.aspx>>.
- [91] HP-IL Files: Where, What, Why, How, etc. [Em Linha], Dan McDonald, 2000. [Consult. Março 2011]. Disponível em WWW: <URL: <http://www.hpmuseum.org/cgi-sys/cgiwrap/hpmuseum/articles.cgi?read=24>>.
- [92] The HP LIFUTIL (E2080A) [Em Linha], 1995. [Consult. Setembro 2011]. Disponível em WWW: <URL: ftp://ftp.hpmuseum.org/lif/lifutil/lif_over.txt>.
- [93] [Em Linha]Virtual Floppy Drive. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://vfd.sourceforge.net/>>.
- [94] What is WinImage? [Em Linha], Gilles Vollant. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://www.winimage.com/winimage.htm>>.
- [95] [Em Linha]Chrysocome. [Consult. Fevereiro 2011]. Disponível em WWW: <URL: <http://www.chrysocome.net/rawwrite>>.
- [96] SD Group. *SD Specifications Part 1 Physical Layer Simplified Specification Version 2.00*, Setembro 2006.
- [97] AT25DF161 [Em Linha]. Disponível em WWW: <URL: http://www.atmel.com/dyn/products/product_card.asp?part_id=4334&category_id=162&family_id=668&subfamily_id=2117>.
- [98] Atmel Corporation. *16-Megabit 2.7-volt Minimum SPI Serial Flash Memory AT25DF161*, 2009.
- [99] AT25DF321A [Em Linha]. Disponível em WWW: <URL: http://www.atmel.com/dyn/products/product_card.asp?part_id=4334&category_id=162&family_id=668&subfamily_id=2117>.
- [100] Winbond. *Winbond Serial Flash W25Q32BV 3V 32M-Bit Serial Flash Memory With Dual and Quad SPI*, 2011.
- [101] Microchip Technology Inc. *Microchip SST25VF032B Data Sheet - 32 Mbit SPI Serial Flash*, Fevereiro 2011.