SISTEMA DISTRIBUÍDO DE ANÁLISE DA REGULARIDADE DE FIOS TÊXTEIS.

João Luis Marques Pereira Monteiro

Tese de doutoramento em
Informática Industrial apresentada à
UNIVERSIDADE DO MINHO.

Braga, Setembro de 1990

Departamento
de
Electrónica Industrial

UNIVERSIDADE DO MINHO

Agradecimentos:

A prossecução de um trabalho de investigação como o que se descreve, atravessa fases que só o apoio, quer científico quer moral, dos colegas, permite ultrapassar. O projecto que agora se apresenta não constitui, obviamente, excepção. São assim naturais algumas palavras de agradecimento para toda a equipa de docentes, investigadores e técnicos ligados ao Departamento de Electrónica Industrial, bem como outros departamentos da Escola de Engenharia.

Dada a sua grande contribuição para o bom êxito do presente trabalho, o autor agradece em particular:

- Dr. Carlos Couto, pela sua orientação, ajuda e encorajamento durante todo o trabalho;

- Eng. Júlio S. Martins pelas inestimáveis sugestões e frutuosas discussões que muito contribuiram na implementação e desenvolvimento do trabalho;

- Doutores A. Cabeço Silva e Elizabete C. Silva pela colaboração e orientação no desbravar dos conhecimentos associados à regularimetria têxtil;

- Engenheiros Paulo Garrido, João Noivo, J. Aparício Fernandes e Alexandre Santos pela sua pronta cooperação na apresentação de sugestões e na sua permanente disponibilidade para arcar com algumas das tarefas associadas à docência e gestão, garantido assim a conclusão do presente trabalho em tempo útil;

- Toda a equipa das Oficinas de Electrónica, chefiada pelo Eng. Manuel Romero, pela prontidão e eficiência com que colaboraram na montagem das diversas placas utilizadas ao longo do projecto.

- Doutores Vasco Freitas e Guimarães Rodrigues pelas inúmeras sugestões apresentadas em diversas fases do trabalho que permitiram minimizar o tempo despendido em algumas etapas.

INTRODUÇÃO.

A análise da regularidade dos fios têxteis - regularimetria - , associada a outras técnicas de análise da qualidade de produção, é indispensável para a caracterização dos produtos resultantes da laboração de uma fiação têxtil.

Esta importância advém da possibilidade de certo tipo de imperfeições, que se encontram por vezes nessa produção, produzirem defeitos graves no tecido final; estes provocam um prejuízo considerável, problema que é obviado pela opção, tomada com inusitada frequência, de diminuir os padrões de qualidade do produto final que acaba por implicar, a longo prazo, um prejuízo acrescido e nem sempre contabilizado.

Como parece ser consenso geral, é neste campo - o da melhoria da qualidade da produção - que o país, e particularmente o ramo têxtil, terão de investir com bastante premência, para poderem garantir a manutenção, e eventual alargamento, do grande mercado existente na actualidade.

Estando a Universidade do Minho particularmente empenhada numa inovadora ligação à indústria nacional, com particular relevo para a têxtil e calçado que têm um elevado impacto na região, não podiam os seus departamentos de engenharia deixar de apreciar esta realidade, tentando apresentar soluções eficazes e económicas para os problemas que vão surgindo.

I - Origem do projecto.

A origem deste trabalho esteve na necessidade de ser apresentada uma solução mais económica e, se possível, mais rápida para a detecção de imperfeições de fio têxtil, causas da sua ocorrência - uma perspectiva nova relativamente aos equipamentos actualmente em uso - permitindo, além disso, uma caracterização (qualitativa e quantitativa) do produto produzido.

Surgindo de uma necessidade da indústria têxtil, e obrigando a uma abordagem não puramente académica do problema mas exigindo, pelo contrário, uma solução eminentemente de engenharia, foi realizado através de uma estreita e prometedora colaboração entre os Departamento de Electrónica Industrial e de Engenharia Têxtil.

As premissas essenciais para a definição dos objectivos deste trabalho têm por base a realidade industrial existente no campo da análise da regularidade de fios, que se pode caracterizar sucintamente nos seguintes pontos chave:

- A análise da regularidade dos fios têxteis passa por duas fases essenciais: determinar imperfeições e caracterizar o fio recorrendo a parâmetros estatísticos;

A primeira destas fases é efectuada junto à produção e destina-se à determinação das imperfeições (pontos grossos ou finos que correspondem, como sugerido pelo nome, a variações consideráveis da massa do fio em pequenos segmentos de fio) que ocorrem na produção; a segunda corresponde à determinação de valores estatísticos (coeficiente de variação e desvio relativo à média, da massa do fio por unidade de comprimento) tradicionalmente utilizados nestas indústrias.

- Só a detecção e contagem de imperfeições é efectuada em tempo real;

A quase totalidade das fiações utiliza já algum equipamento, rudimentar em diversas unidades, que permitem a contagem das imperfeições. Este equipamento, o único a funcionar em tempo real, permite contar essas imperfeições e, eventualmente, substituí-las por nós resultantes do corte do fio e sua posterior ligação.

- Todo o trabalho de caracterização mais complexa do fio é efectuada posteriormente à produção em laboratórios de controlo de qualidade;

A análise mais cuidada das caracteristicas do fio, como sejam algumas medidas estatísticas da sua massa por unidade de comprimento e o espectrograma correspondente, são efectuadas "off-line".

- O equipamento que efectua esta análise é muito dispendioso;

- A interface homem-máquina obriga ao recurso a operadores especializados;

Este aspecto está essencialmente ligado ao facto do equipamento vulgar apresentar os relatórios tradicionais, sem efectuar qualquer tratamento automático dos dados recolhidos. Quando se torna importante conhecer mais e diferentes resultados impõe-se o recurso a técnicos especializados. Acrescente-se ainda o facto de não existir ligação clara entre o equipamento que produz os relatórios laboratoriais e o que efectua a contagem das imperfeições, anteriomente mencionados.

II - Objectivos.

A proposta inicial de trabalho para este projecto, consistia na implementação de um sistema de gestão e controlo da regularidade de fios têxteis, com detecção de irregularidades e, a partir destas, apresentar algumas das causas mais prováveis para a sua ocorrência.

Para a obtenção deste resultados impunham-se algumas condições, derivadas no essencial da "performance" conseguida com os equipamentos comerciais já implantados em diversas unidades fabris.

Assim, exigia-se à partida que, no campo da apresentação de resultados, fosse atingido o nível dos actualmente fornecidos pelos equipamentos em uso nos laboratórios de controlo de qualidade das unidades fabris de fiação.

A partir de uma certa fase do projecto, impuseram-se objectivos mais amplos e ousados. Estes prendiam-se no seu essencial com a possibilidade de permitir, para além da possibilidades de controlar a ocorrência de imperfeições localmente, na rápida centralização dos resultados da análise do fio; com esta metodologia aponta-se no sentido de ser possível efectuar um controlo de qualidade, de todo o produto, poucos instantes após a produção com a evidente melhoria dos tempo de resposta e consequente aumento da qualidade (ou preço) do produto final.

Estes objectivos permitiriam assim melhorar significativamente os resultados, nomeadamente no que se refere aos seguintes aspectos:

- Sistema mais económico.

Este ponto, de importância capital para uma indústria com elevados níveis de produção, adquire um maior realce se tivermos em conta que o equipamento a implementar utilizará parte do material já instalado na maioria das fiações, especialmente os mais dispendiosos: os sensores.

Associado a este aspecto não é de mais salientar o facto de a utilização de equipamento genérico de microinformática permitir apresentar soluções mais económicas que as apresentadas pelos grandes construtores de maquinaria têxtil.

- Diminuição do tempo de resposta do controlo de qualidade, apontando para uma solução de controlo de produção.

A possibilidade de existir um controlo muito próximo da produção permitirá uma considerável redução dos tempos de resposta, acarretando uma natural diminuição dos custos de fabrico.

Efectivamente o equipamento mais especializado em voga na maioria das unidades fabris, com custos na ordem das dezenas de milhões de escudos, destinam-se a uso exclusivo em laboratório. Neste serão analisadas alguma bobinas com fio produzido, permitindo-se determinar a qualidade efectiva do produto; a detecção de irregularidades mais ou menos graves poderão acarretar uma significativa redução do seu valor para venda, podendo mesmo obrigar à destruição do fio produzido.

A existência de equipamento que permite efectuar esta análise mais próxima da produção permitirá uma significativa redução dos custos directos, com a detecção em tempo reduzido de anomalias graves que imponham a destruição do produto, e indirectos, com a possibilidade de aumentar os níveis de qualidade do fio.

A acrescentar a estes aspectos é de realçar o facto de se ter considerado a possibilidade de ser efectuada uma detecção rápida e **automática** das causas que provocaram determinada ocorrência, permitindo a correcção rápida dessa anomalia.

- Melhoria da interface do utilizador com o sistema de análise da regularidade.

Ao impormos a existência de similaridade dos relatórios a apresentar, com os produzidos nos equipamentos tradicionais, garante-se a manutenção dos critérios habituais de caracterização do fio produzido, quando analisados no âmbito do controlo de qualidade.

O carácter inovador deste método advém da implementação de uma ligação homem-máquina que permita ao operador, após a detecção de imperfeições, seleccionar as acções a tomar para corrigir as anomalias observadas; esta detecção de avarias e proposta de soluções pretende-se ainda extensível ao diagnóstico do funcionamento das máquinas e do próprio equipamento de análise.

Esta ligação que veio a ser implementada tendo presente o objectivo de, futuramente, actuar directamente sobre as máquinas, sem necessidade de recorrer a um operador humano para seleccionar as soluções possíveis, permitirá resolver os problemas detectados com uma percentagem de sucesso progressivamente maior.

Este tipo de projecto exige, na prática, um suporte físico que pode ser utilizado nas mais diversas situações. Constitui também a base, quer de sistemas inteligentes de controlo distribuído, na mira de uma utilizações de nível tão elevado quanto as "C.I.M.", quer de projectos de controlo ou gestão de produção de unidades fabris. Este aspecto veio a tornar-se ainda mais evidente aquando da definição da arquitectura a adoptar para a implementação do equipamento de análise da regularidade de fio têxtil.

Estando estes projectos a ser alvo da atenção do Departamento de Electrónica Industrial da Universidade do Minho, optou-se pela definição de **um objectivo mais geral** para o sistema a desenvolver, alargando a especificação das características desse suporte, tendo em conta a extensão do âmbito de aplicação do mesmo.

Este aspecto verificou-se posteriormente ter sido uma aposta correcta, já que foram utilizados com êxito, em projectos complementares a este, diversos módulos implementados no âmbito deste trabalho. Estes módulos de "hardware" e "software"

foram naturalmente especificados tendo em vista uma fácil utilização e seguindo **sempre uma filosofia modular**, permitindo assim uma adaptação à previsível evolução de técnicas e tecnologias. Este aspecto modular correspondeu desde início a um objectivo, ou melhor, uma imposição no desenvolvimento das diversas fases do trabalho, com vista a uma minimização dos custos do projecto.

Foi dispensado um particular cuidado a alguns temas do trabalho, de entre os quais podemos destacar:

- O desenvolvimento de uma arquitectura do sistema, o mais modular possível, com particular ênfase para a comunicação entre os diversos intervenientes;

Neste âmbito foi desenvolvido uma estação remota de aquisição e tratamento de dados, bem como um protocolo de comunicação; este protocolo foi implementado seguindo uma metodologia idêntica à definida pela ISO, com vista a permitir a sua eventual evolução para outros protocolos.

- O desenvolvimento de algoritmos eficientes para o processamento digital do sinal adquirido;

A este tema foi dedicada uma atenção muito especial, quer na melhoria dos tempos de processamento dos algoritmos tradicionalmente utilizados (FFT e outros), quer no desenvolvimento de técnicas inteiramente novas para a determinação de certos parâmetros (Transformada de Walsh, determinação de frequência de erro, etc.).

- A implementação de um módulo de diagnóstico de avarias;

Utilizando a linguagem PROLOG, desenvolveu-se um módulo, fácil de utilizar, que fornece uma ajuda considerável na identificação de avarias ou perturbações que ocorrem no processo de fabrico; este módulo apresenta ainda sugestões sobre as acções passíveis de resolver estas avarias, numa base probabilística do seu sucesso.

- Garantir a apresentação de resultados dentro dos parâmetros têxteis tradicionais;

Sendo os resultados fornecidos por parte dos algoritmos de processamento de sinal de difícil interpretação, implementaram-se métodos complementares que permitem uma fácil e imediata compreensão destes, permitindo assim detectar características de difícil observação utilizando os métodos tradicionais.

III - Organização do documento.

No presente documento são descritas as fases percorridas na implementação do presente sistema, subdividindo o texto em 4 partes fundamentais, em que se apresentam os aspectos relacionados com a filosofia geral do regularímetro, concepção, desenvolvimento, análise de resultados e perspectivas para o trabalho futuro, como se sumaria de seguida:

PARTE 1 - FILOSOFIA GERAL DO REGULARIMETRO DIGITAL: Esta parte é dedicada à análise da situação actual a nível da indústria no que respeita à regularimetria de fios; é ainda efectuado um estudo de diversas alternativas de implementação e são analisadas as restrições que lhes são impostas, tendo como finalidade a definição da arquitectura geral do sistema, quer a nível físico ("hardware"), quer a nível de programação.

Foram estudadas algumas alternativas para a arquitectura física do equipamento, que permitiram algumas tomadas de posição iniciais com vista a balizar o trabalho. Esta parte é composta pelos seguintes capítulos:

Capítulo 1 - É efectuada uma pequena análise das aplicações de electrónica e informática à indústria têxtil, incluindo o ponto da situação da regularimetria, e referências às potencialidades das técnicas e tecnologias existentes hoje ao nosso dispor.

Capítulo 2 - Descrevem-se as premissas base na definição das estratégias adoptadas no que diz respeito à configuração a implementar (uma estação mestre diversas estações escravas) bem como ao tipo de programação necessária.

São nomeadamente analisadas todas as limitações impostas e, após selecção de uma opção relativa à arquitectura geral, é verificada a sua capacidade para cumprir com os requisitos especificados, mesmo consideradas as piores condições de funcionamento do sistema.

Capítulo 3 - Apresentam-se os processos para efectuar o controlo das estações a partir do microcomputador central. Neste ponto descrevem-se as alternativas e soluções adoptadas para a utilização de algoritmos de controlo dedicados e/ou gerais para a estação local comandar o dispositivo que controla.

Nesta fase foi despendido um particular cuidado com a previsível generalização da aplicabilidade deste sistema em projectos mais alargados.

PARTE 2 - CONCEPÇÃO: Nesta parte do trabalho são discutidos diversos algoritmos e estudadas diversas soluções de comunicação, como se explicita com maior pormenor de seguida:

Capítulo 4 - Apresentam-se genericamente os algoritmos de processamento de sinal implementados para a determinação de características dos fios têxteis produzidos. São analisados algoritmos para determinação de espectros, usando diversos tipos de transformadas, a que se acrescenta a referência a tratamento posterior de resultados; apresenta-se por fim (secção 4.3) uma transformada desenvolvida para a determinação rápida de frequência de erros.

Capítulo 5 - Neste capítulo discutem-se os algoritmos para a determinação de parâmetros têxteis; expõem-se não só os algoritmos relativos a resultados tradicionais mas também são apresentados os respeitantes a parâmetros novos.

Capítulo 6 - Discutem-se as técnicas de 'Inteligência Artificial' desenvolvidas na detecção automática de avarias e apresentação de acções para as colmatar. Esta análise é efectuada numa perspectiva de aprendizagem com o operador, criando assim um pequeno "Expert System" (sistema pericial).

Capítulo 7 - Descreve-se o protocolo de comunicação adoptado para a comunicação, nomeadamente no que diz respeito às camadas 2 e 3 do "standard" definido pelo I.S.O. (International Standard Organization) para a interligação de sistemas abertos (OSI = Open System Interconection). Este protocolo embora baseado numa máquina específica é apresentado, para cada camada, como um módulo independente, só sendo necessário precauções especiais a nível de qual a máquina em uso para a especificação da camada 2 (*eventualmente a camada 1, camada física, poderá ser alterada*).

PARTE 3 - DESENVOLVIMENTO: Nesta parte são discutidos os aspectos relacionados com a implementação prática do sistema desenvolvido, segundo as considerações de concepção tomadas anteriormente. Estes aspectos - que dizem respeito quer às soluções de "hardware" quer de "software" - são apresentados nos seguintes capítulos:

Capítulo 8 - Descreve-se o projecto físico das estações remotas, baseando-se esse estudo em alguns opções chave; essas opções são discutidas e apresentadas ao longo de diversas secções deste capítulo.

Capítulo 9 - Neste capítulo são estudados e discutidos diversos aspectos da implementação da programação residente nestas estações.

São assim descritos os módulos de programação relativos à aquisição de dados, Comunicações, Módulo de Operação, etc. Além destes apresenta-se uma breve descrição de um Módulo de Interpretação desenvolvido especificamente para o efeito.

Capítulo 10 - Descrevem-se os algoritmos implementados no processador central além de "hardware" para comunicações (secção **10.1**) que foi necessário incluir. A nível de programação são referidos os diversos módulos de processamento digital de sinal (secção **10.4**), de detecção e correcção de avarias (secção **10.6**), de comunicação com as estações remotas (secção **10.5**) e o módulo operativo (secção **10.3**), para além de ser efectuado um estudo prévio da estratégia de apresentação de resultados dos algoritmos implementados (secção **10.2**).

Tem particular relevo a descrição e análise da implementação prática do módulo responsável pela detecção automática de causas de ocorrência de avarias, e de alguns resultados que apresenta.

É também efectuada uma primeira análise dos algoritmos de comunicação implementados, quer nas estações remotas quer no computador central.

PARTE 4 - RESULTADOS, APLICAÇÕES DIVERSAS, RECOMENDAÇÕES PARA TRABALHO FUTURO E CONCLUSÕES: Nesta parte são apresentados os resultados obtidos com o processamento e apresentam-se recomendações (e perspectivas) para o trabalho futuro; finalmente são apresentadas as conclusões gerais deste trabalho:

Capítulo 11 - Apresentam-se os resultados obtidos nos diversos passos de implementação do sistema de regularimetria para a indústria têxtil, nomeadamente os relatórios apresentados pelo microcomputador central.

É ainda efectuada uma análise dos tempos de processamento necessários para os diversos algoritmos, funcionando em diversas máquinas.

Capítulo 12 - É efectuada uma apresentação dos resultados obtidos em diferentes aplicações.

Apresentam-se resultados genéricos, nomeadamente a nível de "performance" para uma utilização diversificada de todo o sistema desenvolvido. Destaca-se a avaliação da eficácia das comunicações.

Capítulo 13 - Neste capítulo discutem-se as possibilidades de evolução deste sistema, ou de um seu 'derivado' ("up-grading").

Um dos aspectos a tomar em conta prende-se com a evolução provável da placa que implementa as estações remotas, sendo a discussão centrada especialmente na provável evolução desta estação na Universidade do Minho (Electrónica Industrial), tendo contudo sempre em conta as modificações do mercado nas aplicações de Controlo Industrial, mantendo assim as premissas e condicionantes relativas à economia de recursos e investimentos.

É proposta uma nova arquitectura para uma estação remota de aquisição e processamento de dados. Como é de esperar, tendo em conta a grande evolução do mercado desde a altura em que foi optada a arquitectura presente, a nova estação terá recursos muito superiores, recorrendo nomeadamente a duas unidades de processamento.

São também analisados e discutidos os aspectos associados à comunicação através da rede, especialmente no que diz respeito aos protocolos (ou protocolo de várias camadas) a implementar com vista ao encontro com a provável evolução destes no meio industrial.

São ainda apresentados algumas das aplicações já efectuadas para parte deste equipamento, bem como as solicitações apresentadas por diversos industriais e técnicos dessa indústria, com vista a utilização deste equipamento em situações concretas.

Capítulo 14 - Neste capítulo são apresentadas as conclusões. Estas referem-se não apenas aos aspectos associados ao presente também mas também ao futuro da investigação na área de Electrónica e Informática Industrial, particularmente para o caso da indústria portuguesa.

Apêndices - Incluem-se dois apêndices em que se apresentam, com mais detalhes, algumas características do módulo de comunicação (*no microcomputador central e estações remotas*) e do interpretador (*da estação remota*) implementados.

PARTE 1.

FILOSOFIA GERAL DO REGULARÍMETRO.

Ao contrário do que acontecia à alguns anos, nenhum projecto industrial pode gabar-se de, presentemente, recorrer aos conhecimentos de uma área de estudo particular. A disseminação do uso da informática e electrónica na indústria, obriga os projectos em desenvolvimento a recorrer não apenas a uma, mas a um leque diversificado de especialidades, dentro de cada uma das matérias necessárias ao desenvolvimento de protótipos.

Pode-se afirmar com certa segurança que, a tendência actual aponta para o desenvolvimento de sistemas suficientemente flexíveis, facilmente adaptáveis às variações, quer do processo de fabrico, quer das técnicas e tecnologias utilizáveis.

A necessidade de flexibilização, quer da produção, quer dos sistemas de gestão e controlo, obrigam à reformulação das características tradicionalmente atribuídas a estes. Um sistema de controlo com "hardware" e "software" próprios, deve permitir a

fácil alteração de qualquer um dos seus módulos sem obrigar à completa substituição ou reconstrução do sistema.

Torna-se indispensável que o sistema a implementar seja modular, sendo cada módulo definido não só pela necessidade do sistema em projecto mas também com base num conjunto de especificidades comuns que permitam a sua utilização final em sistemas bastante diversificados. Todas estas considerações ruirão, no entanto, se enveredarmos por um caminho que, de tão generalista, não permita uma rentabilização relativamente rápida.

Esta preocupação é por demais notada na indústria têxtil em que a concorrência se faz frequentemente notar, pela diferença no custo de um produto a nível de 'miliescudos', custo esse que pode ser contabilizado pelos minutos (quando não segundos) que é possível melhorar na produção em cada unidade ou posto de trabalho.

Nesta parte do texto, agrupando 3 capítulos, serão discutidas os aspectos básicos referentes à implementação de um regularímetro digital, baseando-nos nas particularidades da indústria têxtil e nas potencialidades das técnicas e tecnologias disponíveis.

São também analisadas as possíveis abordagens para a implementação prática deste sistema, a partir de algumas premissas iniciais e tendo em conta alguns dos aspectos já apresentados e relacionados com a flexibilização e modularidade pretendidas para o produto final.

Capítulo 1. Aplicações da electrónica e informática na indústria têxtil.

O uso de informática e electrónica na indústria têxtil, tal como noutras áreas industriais, está naturalmente bastante implantado.

Algumas destas utilizações são já consideradas indispensáveis, estando em uso há longo tempo. Incluem-se de entre elas o uso de Informática de Gestão, essencialmente para efectuar:

- gestão de produção,
- contabilidade,
- gestão de pessoal,
- gestão de stocks,

Também a electrónica está actualmente bem espalhada, podendo mesmo afirmar-se que, praticamente não existe um equipamento industrial não utilizador de dispositivos electrónicos.

Verifica-se actualmente uma utilização mais diversificada da informática derivada directamente da exigência de baixo custo e elevados padrões de qualidade, aspecto que interessa profundamente a indústria nacional; este ponto integra-se na perspectiva actual de minimização dos custos, mantendo (ou aumentando) os padrões de qualidade, ao invés da utilização intensiva de mão-de-obra não especializada, como se tem verificado até agora.

Efectivamente, a explicação do sucesso da indústria portuguesa no ramo têxtil, não sendo o nosso país produtor de praticamente nenhuma da matéria prima (exceptuando apenas uma componente de certo peso na produção de lã), só pode conseguir-se com o tradicional recurso à mão-de-obra intensiva, de baixo custo.

Estas considerações apontam assim para um aumento dos investimentos na área da electrónica e microinformática industriais que, sendo no entanto recuperáveis a médio prazo, poderão levantar diversas dúvidas e hesitações quer por parte dos industriais quer por parte dos assalariados; estas derivam no essencial da passagem da situação de mão-de-obra intensiva para uma caracterizável pelos seguintes traços tipo:

- Recurso a mão-de-obra muito especializada;

Esta especialização, provocando ou um aumento de desemprego generalizado ou, na melhor das hipóteses, no sector dos trabalhadores menos qualificados, acarreta naturalmente implicações de ordem económico-social.

- Gestão de produção eficaz;

A implementação de esquemas de controlo de produção eficaz, não os tradicionais métodos de controlo de qualidade, permitirá melhorar consideravelmente a relação qualidade/preço do produto de cada unidade fabril. Realça-se ainda que uma gestão de produção, de bom nível, implica, não apenas uma boa metodologia de controlo da fabricação, mas também uma gestão e controlo dos intervenientes directos do processo de fabrico (quer estejam ou não dependentes de factores humanos).

- Automatização e mecanização progressiva sempre que possível;

Considerando os aspectos referidos anteriormente deriva-se, naturalmente para esta solução. Ressalva-se apenas o facto de, actualmente, nem sempre se optar por metodologias eficazes para a mecanização e automatização, preferindo-se frequentemente uma solução imediatista a uma de longo prazo.

Pode-se assim afirmar, sem grandes riscos, que a electrónica e informática, num futuro bastante próximo, sejam utilizadas de modo ainda mais intensivo, especialmente em áreas até agora apenas fruto de experiências piloto, tais como,

- **gestão de produção**, que não se limita a um sistema gerir a informação de "stocks" e imobilizados, mas também a ser efectuada uma gestão e controlo dos intervenientes no processo de fabrico, impondo uma ligação rápida e eficiente entre os dispositivos de gestão geral da unidade e as linhas de produção, caminhando no sentido da tão idealizada fábrica automática,

- **desenho assistido por computador** para o desenho de padrões de tecidos ou dos cortes de peças para confecção, prevendo ou não uma ligação directa aos equipamentos que trabalham a matéria-prima,

- **controlo de qualidade**, no que diz respeito quer à melhoria das técnicas derivadas do aperfeiçoamento dos sistemas de controlo e medida, quer devido à crescente automatização do processo de controlo,

- **armazenamento automático**, passando naturalmente pela utilização de mecanismos de armazenamento controlados por computador.

O uso mais diversificado destas técnicas obriga ao recurso a novos métodos e tecnologias, levantando problemas mais complexos, de entre os quais poderemos destacar:

- a interligação dos diversos sistemas informáticos utilizados implementando um controlo do processo de fabrico bem hierarquizado, rápido e eficaz,

- a possibilidade de reconfigurar a topologia geral, o funcionamento, o tipo de resultados, bem como as acções de controlo a gerar,

- a rentabilidade económica do investimento originado pela instalação de um novo sistema.

Neste trabalho, baseando-nos nos pressupostos apresentados ao longo desta secção, considerámos uma preocupação dominante a minimização destes problemas.

Assim garante-se a facilidade de interligação entre sistemas hierárquicos diferentes e o desenvolvimento modular das diversas partes do projecto, permitindo uma fácil actualização ("up-grading").

Apontou-se, tal como exposto na secção de apresentação de objectivos, para a implementação de um sistema que preveja, ou permita evoluir para, uma solução final contemplando 3 aspectos basilares:

- Economia,
- Diminuição do tempo de resposta na gestão da qualidade de produção, e
- Melhoria da interface homem-máquina.

1.1 Ponto da situação da regularimetria.

Equipamento para a medição da regularidade de fio têxtil é usada na indústria há já alguns anos. Tal como nas aplicações de electrónica e informática nas restantes indústrias, existe uma grande evolução na qualidade e sofisticação destes equipamentos; esta evolução tem sido acompanhada de um aumento de preço acentuado, que se pode considerar exagerado especialmente se comparármos com outras áreas industriais ou mesmo a electrónica e informática de consumo 'caseiro', que apresentam uma variação de sinal contrário.

Apesar deste evidente progresso tecnológico, é ainda bem patente que alguns dos problemas referidos anteriormente não foram capazmente solucionados.

Efectivamente o equipamento actualmente em uso destina-se, na sua quase totalidade, a uso laboratorial isto é, em controlo de qualidade; em geral é nos laboratórios responsáveis pela caracterização do fio **após a produção** que este trabalho é efectuado **[61]**.

Este equipamento fornece no entanto resultados bastante completos, de que se destaca o Coeficiente de Variação (CV%, medida da variação do desvio padrão relativamente à média), o Coeficiente de Irregularidade (U%, medida do desvio médio absoluto) e, por vezes, um espectro da imagem de espessura do fio (espectrograma da massa de fio por unidade de comprimento) **[62, 63, 64, 65]**.

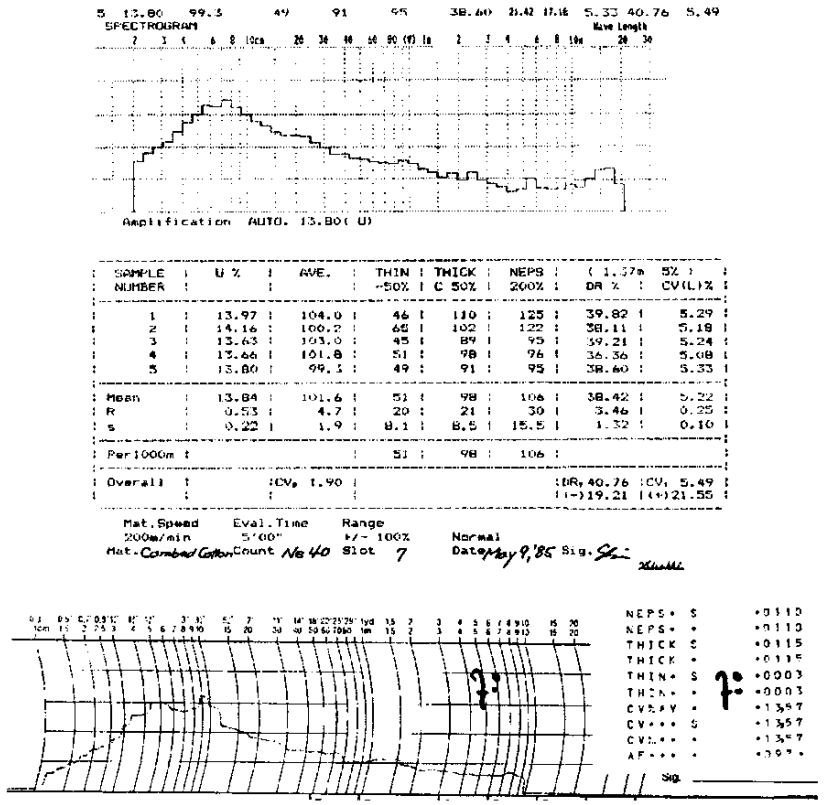


Figura 1. Relatórios apresentados por equipamento USTER e KEISOKKI.

A determinação destes parâmetros, efectuada durante largos anos por processos quase artesanais (a determinação do coeficiente de variação chegou a ser efectuada a peso), é actualmente indispensável dado os elevados padrões de qualidade exigidos pelo mercado; torna-se no entanto muito dispendiosa não só pelo preço do

equipamento mas também pela necessidade de dispor de elementos especializados para efectuar o controlo e pelo tempo de atraso que provoca na análise do fio têxtil.

Apesar da presente evolução destas máquinas ter sido no sentido de baixar os tempos despendidos na análise do fio têxtil, não deixa de possuir as características de um controlo de qualidade efectuado "a posteriori".

Quando alguns destes dispositivos, ou outros similares, são utilizados directamente na linha de produção não é possível recorrer à determinação de todos os parâmetros referidos, pelo menos em tempo útil, sendo vulgar executarem apenas a classificação do número e tipo de imperfeições.

Apresentam-se na figura 1 os resultados de dois desses equipamentos, *Uster* e *Keisokki*, usados como controlo de qualidade.

Podem avançar-se duas alternativas de automatização capazes de melhorar os resultados de uma empresa industrial:

A primeira reside na modificação do processo de fabrico por forma a permitir produzir mais quantidade por unidade de tempo, garantindo-se ou não a manutenção da qualidade de produção nos valores tradicionais; esta solução, ultrapassando o âmbito do presente projecto, resulta em geral da modificação dos intervenientes directos na produção (máquinas e operários).

A segunda, ligada ao aumento da qualidade do produto final (fio no presente caso), recorre usualmente à melhoria do funcionamento da unidade fabril, não por modificações globais do processo industrial, mas pela utilização de novas técnicas e/ou metodologias.

No caso em análise estes factores estão associados a uma mais eficiente caracterização do fio. Tornando-se possível detectar em tempo útil, logo necessariamente mais curto que o habitual, a ocorrência de imperfeições 'catastróficas', garante-se que a qualidade do fio produzido é superior a um certo padrão. Com esta actuação poderemos melhorar significativamente os resultados económicos associados à produção.

1.2 Potencialidades das técnicas e tecnologias disponíveis.

Tem existido um enorme incremento na percentagem de automatização dos dispositivos intervenientes num determinado processo de fabrico. Este autêntico "boom", responsável frequente pela grande inconstância de alguns mercados, tem de

ser bem 'aproveitado', impondo assim uma grande utilização dos conhecimentos existentes.

As ferramentas tecnológicas hoje em dia ao nosso dispor, aliadas ao baixo custo e grande divulgação de dispositivos electrónicos integrados, permitem-nos ultrapassar algumas das limitações existentes e explorar novas formas de controlo:

- O uso de diversos processadores permite a distribuição das tarefas, por exemplo associando cada uma a um processador próprio; esta opção garante a não sobrecarga de todos os processadores a montante de um dos que está responsável pela execução de uma tarefa local. Este aspecto é tanto mais importante quanto se verifica, em geral, que as tarefas de controlo locais só obrigam ao recurso aos computadores a montante ("hierarquicamente superiores") em casos de emergência ou de controlo da actividade do próprio processador; acrescente-se que esta situações ocorrem com pouca frequência **[1, 10, 8]**.

A banalização do uso de microcontroladores integrados vem também aumentar a capacidade de processamento sem um aumento significativo de custos.

- Por outro lado está ao nosso alcance a utilização de facilidades de comunicação que permitam a interligação de diversos sistemas com capacidades de processamento mais elevadas, recorrendo a linhas de comunicação de alta velocidade (de uma centena de Kbit/s a Mbit/s) **[1]**.

- Com vista a explorar novas técnicas de controlo, é possível recorrer a técnicas avançadas de processamento da informação.

Inclui-se neste área o recurso às chamadas técnicas de Inteligência Artificial para ajudar a determinar as causas, efeitos e minimização dos erros, até agora determinados com base na experiência adquirida por pessoal que, para além de ser altamente especializado, possuía um longo tempo de experiência profissional, e logo de morosa e dispendiosa formação **[51, 52, 54]**.

- A utilização de algoritmos complexos de processamento digital de sinal não são uma sobrecarga tão pesada, dada a existência de dispositivos aceleradores (placas e co-processadores) **[31, 32]**.

Estas considerações permitiram-nos definir um sistema base a aplicar para o projecto do regularímetro.

Capítulo 2. Filosofia geral da implementação do regularímetro digital.

O sinal eléctrico correspondente à 'imagem' da espessura do fio obtém-se por variação da capacidade de um condensador (ver figura 2).

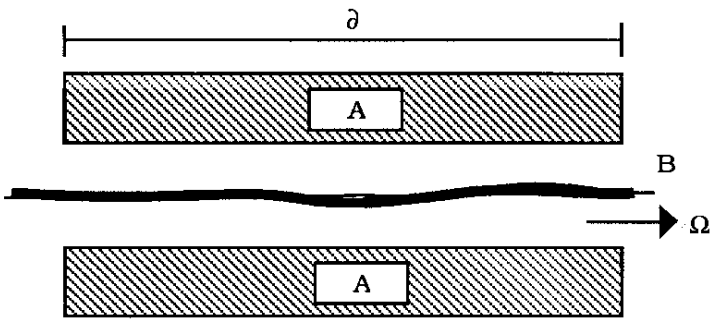


Figura 2. Captação da informação da espessura do fio no sensor capacitivo.

∂ - Comprimento do sensor,

Ω - Velocidade de deslocamento do fio têxtil,

A - Placas do condensador,

B - Fio têxtil.

O fio têxtil atravessa 2 placas metálicas que constituem as armaduras de um condensador cujo dieléctrico é constituído pelo ar e pela massa de fio sob essas placas. A variação da massa de fio têxtil sob essas placas pode ficar a dever-se a eventuais irregularidades existentes no fio, e também às próprias características das fibras que o constituem **[62, 63, 64, 65]**.

Conforme a variação desta massa, assim varia o dieléctrico do condensador e, consequentemente, a sua capacidade. Essa variação, por recurso a circuitos ressonantes, traduzem-se num sinal com frequência directamente proporcional à massa do fio em análise em cada instante; deste sinal é bastante simples obter um sinal eléctrico com amplitude directamente proporcional a esta massa.

No estudo da regularidades do fio têxtil existem, para além de alguns parâmetros estatísticos, três imperfeições típicas, cujo número médio de ocorrência tem grande importância na classificação, em termos de qualidade, do fio produzido.

As imperfeições em causa são as seguintes:

- **Botões** (são vulgarmente conhecidos por **NEP**),
Uma grande massa de fio, superior a 100-200% da média, num curto espaço.

- **Pontos Grossos**
Uma massa de fio exagerada, inferior a 50% do valor médio, num espaço maior.

- **Pontos Finos**
Uma diminuição significativa da massa num espaço não muito curto.

Estas irregularidades são capazes de, ao ocorrerem com um uma frequência fixa ou quase fixa, provocar defeitos graves no tecido fabricado, podendo originar uma risca que origina uma modificação grave do padrão. São mais graves as imperfeições de frequência fixa do que imperfeições aleatórias, mesmo que em valor médio ocorram em igual número.

Por outro lado, o número de ocorrências destas irregularidades é uma medida eficaz da qualidade do fio têxtil produzido.

Dos parâmetros tradicionalmente calculados destacam-se:

Coeficente de Variação (CV%):

Fornece uma informação sobre a variação do desvio padrão relativamente à média, e cuja definição é dada por,

$$CV\% = \frac{\partial}{\overline{x}}\,100\% \quad (\text{em que } \partial \text{ é o desvio padrão e } \overline{x} \text{ a média}).$$

Para a determinação do CV% usam-se as amostras relativas à massa de fio captadas consecutivamente, isto é, cada amostra corresponde à massa do fio produzido no espaço coberto por um sensor; estas amostras são utilizadas para o cálculo do desvio padrão e média, ao longo de um segmento.

Desvio Médio Absoluto ou Coeficiente de Irregularidade (U%):

Este parâmetro fornece indicação sobre o desvio de cada amostra de fio (ou melhor da sua massa), relativamente à média de um conjunto de amostras, sendo definido como,

$$U\% = 100\% \frac{\sum |x_i - \overline{x}|}{\overline{x}\ N}$$ em que N é o nº de amostras recolhidas no segmento;

$\overline{x}$ é o valor médio das N amostras;

x_i é o valor de cada amostra.

Espectro de massa:

Utilizando os valores da massa do fio têxtil sob a acção dos sensores em cada instante (directamente relacionáveis com a espessura), determina-se o respectivo espectro; com este é possível verificar dados importantes sobre a qualidade do fio, bem como caracterizar as fibras constituintes do mesmo.

Existem, além destes parâmetros, outros coeficientes bastante menos divulgados a que nos referiremos num capítulo próprio, e que permitem uma caracterização adicional do fio.

2.1 Condicionantes do sistema.

A análise do fio em produção baseia-se na determinação da espessura que é efectuada 'através' de medidas de massa de fio por unidade de comprimento.

Esta medida deve ser efectuada em bobinadeiras ou contínuos que, tal como o nome sugere, são máquinas que constroiem bobinas com o fio têxtil; funcionando na fase final de produção do fio têxtil, estas máquinas permitem construir bobinas utilizáveis posteriormente nos teares. Nestas máquinas, que também recebem a matéria-prima de bobinas (embora de outro tipo), nem sempre é efectuado o controlo de qualidade do fio produzido; existindo, este resume-se à contabilização de imperfeições. Por vezes é incluído equipamento que permite, caso seja detectada uma imperfeição, cortar o fio e 'dar um nó', que tem dimensões fixas e conhecidas; limita-se assim o *tipo* de problemas posteriormente detectáveis, uma vez que a partir desta situação será apenas necessário conhecer a frequência e número da ocorrência de imperfeições (todas com dimensões iguais às do nó).

Os contínuos, tendendo actualmente a desaparecer devido à sua baixa produtividade, permitem a utilização de mais fusos ('ou bobines') em simultâneo mas a sua velocidade de bobinagem não excede os 50-70 m/minuto (as bobinadeiras permitem até 500-700 m/minuto).

Sendo efectuado algum controlo de qualidade, está generalizado o uso de sensores capacitivos idênticos aos da figura 2. Com base nestes sensores pretende-se, associando equipamento electrónico adequado, efectuar a análise da regularidade dos fios têxteis em produção em tempo real; este aspecto é bastante importante se tivermos em conta que sem um grande investimento é possível melhorar consideravelmente a "performance" da unidade fabril.

O número de canais a analisar em simultâneo (dependente da maquinaria existente) pode atingir o máximo de 1000 (em contínuos), sendo contudo típico um valor inferior a 250 canais (correspondente a 250 fios a analisar em simultâneo); estes canais, se em grande número, obrigam à utilização de máquinas de grandes dimensões que implicam um grande afastamento físico (até uma centena de metros) entre os pontos onde se fará a recolha da informação.

Para simplicidade da montagem, economia de materiais, facilidade de manutenção, pretende-se que as ligações a efectuar para o diverso equipamento a desenvolver sejam de fácil instalação e económicas. Com base nestes objectivos podemos já adoptar algumas premissas base para o sistema:

- Existem limitações do custo do equipamento a utilizar, sendo uma condicionante o baixo custo quer dos sistemas de aquisição, quer da sua manutenção.

- Sendo necessário adquirir informação do fio produzido em simultâneo num elevado número de fusos (ocupando fisicamente uma grande extensão), é necessário recorrer a um elevado número de sistemas de aquisição.

- Os sistemas de aquisição devem resolver automaticamente a maior parte das situações de erro no próprio local (sem recorrer à intervenção de um operador humano ou processador poderoso para a tomada de decisões).

- Em caso de impossibilidade de resolver os casos de erro, e/ou para tratamento estatístico da produção e arquivo, pretende-se uma interligação económica a um micro/mini computador (μC) de baixo custo. Dada a grande divulgação e baixo custo, este deve ser um microcomputadores pessoal, tanto mais que existe uma grande diversidade de equipamentos complementares para estas máquinas. Por outro lado, é uma das máquinas com que, mesmo pessoas inexperientes, possuem maior contacto.

- O funcionamento dos sistemas de aquisição de dados não deve ser afectado pela falha de um ou mais deles. Esta premissa, sendo uma condição geral do funcionamento de sistemas em rede, aplica-se a este caso por maioria de razão.

- Qualquer falha referida anteriormente não pode afectar o microcomputador central, nem qualquer outra máquina a montante; isto é, estando o computador pessoal ligado a outro dispositivo de processamento mais complexo, este não pode ser afectado pela falha total ou parcial das trocas de dados, pelo menos no que diz respeito à sua função principal.

- É necessário garantir que a falha deste microcomputador central (ou pelo menos centralizador de informação) não perturbe o funcionamento das estações de aquisição de dados excepto no que diz respeito à comunicação, limitando assim a extensão da falha.

- Dado que os erros têm padrões típicos, cuja forma e ocorrência, em conjunto com a história do funcionamento do sistema, fornecem informação de grande relevância sobre a causa que os provocou, pretende-se um tratamento estatístico, no microcomputador central; este deve fornecer resultados que ajudem os operadores a determinar as causas que provocaram certa perturbação.

2.2 Estudo prévio de alternativas de "hardware / software".

Nesta secção será discutida a selecção da estrutura do sistema global (porquê uma rede e qual o microcomputador central) bem como da arquitectura de cada estação de aquisição de dados (requisitos de processamento). Inclui-se o estudo das necessidades de comunicação, definindo, por fim, qual o tipo de informação a transitar na linha de comunicação.

A primeira decisão tomada neste projecto foi relativa à arquitectura global do sistema, tomando como premissa os valores relativos ao número de canais de aquisição da espessura do fio indispensáveis. Consideraram-se ainda outros pontos-chave influenciadores desta arquitectura e já anteriormente mencionados, que poderemos resumir:

- Pretende-se a utilização de estações de aquisição de baixo custo;
- O microcomputador central deve ser económico, fiável e de uso generalizado ;
- Impõe-se a observância das regras de a falha de um elemento não comprometer os outros;

Cumprindo algumas destas especificações optou-se, como microcomputador central, por um computador pessoal do tipo IBM-PC, XT, AT ou compatível. Esta consideração manteve-se para todas as hipóteses de arquitectura analisadas.

Para a implementação de um sistema de controlo desta índole podemos, sem grandes preocupações, considerar três tipos de arquitecturas do sistema (figura 3).

A primeira (figura 3.a), utilizando um microcomputador equipado com um número de placas de aquisição de dados suficientes para a análise de todos os fios.

Na segunda (figura 3.b) existiriam várias placas possuindo um processador próprio e diversos dispositivos de aquisição de dados, residindo quer no microcomputador, quer comunicando com este por intermédio de porta série.

Na terceira (figura 3.c) utilizar-se-ia uma arquitectura em rede, em que cada placa possuiria um processador próprio para controlar a captação dos dados do fio e gerir as trocas de informação com o computador central.

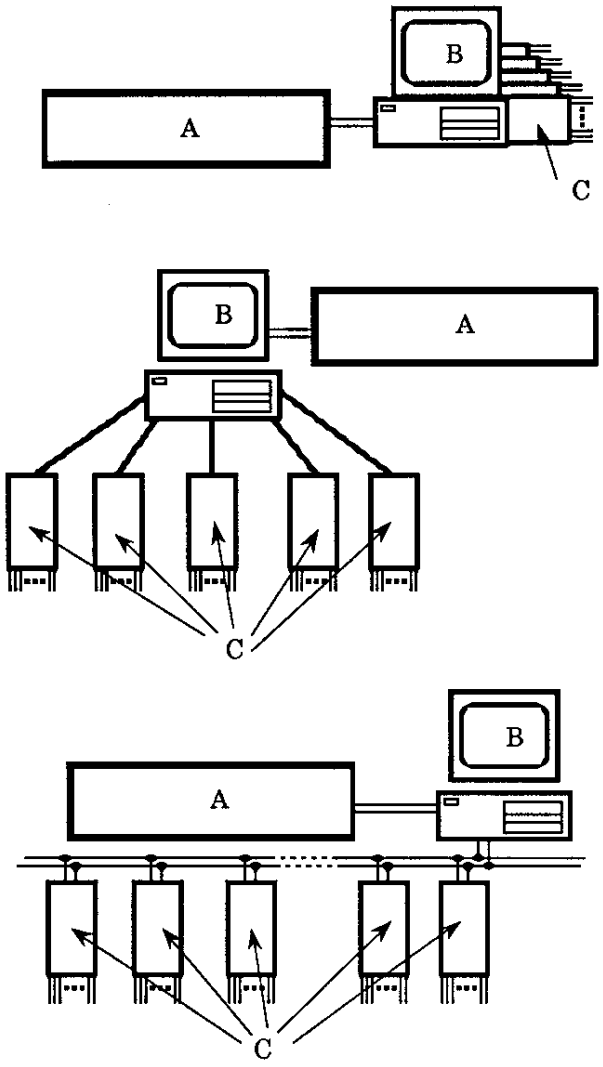
Tanto a primeira como a segunda solução foram abandonadas sem estudos adicionais dado os inúmeros problemas que levantavam:

- A falha do microcomputador central origina a quebra de todo o sistema;
- Seriam necessárias várias placas adicionais no barramento do "PC";
- No caso da primeira solução seria o "PC" responsável pela aquisição de dados.

A vantagem destas soluções residiria apenas na facilidade de adaptar este equipamento por forma a implementar um sistema de análise para um número limitado de canais (1 a 10), como acontece nos laboratórios de controlo de qualidade; apesar disso, só a 3ª opção permite uma flexibilidade suficiente para ser possível um funcionamento que cumpra ambos casos (controlo de qualidade em laboratório com um nº reduzido de canais e controlo de produção na fábrica com múltiplos canais).

Com base nas premissas expostas optou-se assim por uma arquitectura em rede, com diversas estações de aquisição de dados espalhadas fisicamente ao longo da unidade fabril, interligadas entre si e a um microcomputador central IBM-AT/XT ou compatível **[67, 68]**.

Eventualmente este microcomputador central estará por sua vez ligado a uma unidade central de processamento de dados mais complexa e orientada para o trabalho de gestão, garantindo-se assim a interligação de toda a unidade fabril, desde as fases de fabrico e controlo de produção até à gestão financeira.



a) Arquitectura completamente centralizada; as estações de aquisição de dados são instaladas directamente no barramento do μC.

b) Arquitectura em que as estações de aquisição de dados residem fora do μC, mas comunicam directamente com este através de 'portas' série do μC (uma para cada placa)

c) Arquitectura distribuída, em que cada estação de aquisição comunica através de uma rede com o μC.

Em todas as figuras:

A - Equipamento adicional.

B - Microcomputador central.

C - Estações de aquisição.

Figura 3. Possíveis arquitecturas de sistemas de aquisição de dados.

a) Solução centralizada em que o microcomputador é equipado com diversas 'placas' de aquisição de dados;

b) Solução em que cada 'placa' de aquisição de dados comunica directamente com o μC através de linha série dedicada;

c) Solução distribuída em que todas as 'placas' comunicam pela mesma linha.

Para esta ligação (microcomputador central à unidade gestão) seria eventualmente utilizado um "standard" de comunicação de utilização corrente no mercado, recaindo a sua escolha em aspectos mais dependentes do equipamento de nível hierárquico superior bem como da sua arquitectura (se é apenas um processador, se existe uma rede própria, etc.), do que aspectos relacionados com o sistema presentemente em estudo **[1]**.

Está prevista a criação de um nó de rede num destes 'micros' que permita a sua interligação com uma *tradicional* rede "Ethernet" já existente na Universidade. Esta solução, já testada para a interligação de redes de microcomputadores diferentes permitirá, com o recurso à programação adequada, controlar todo o funcionamento da linha de produção de fio remotamente; esta perspectiva abre campo a um sem número de alternativas, que passam pela completa reformulação dos conceitos de gestão, dependendo apenas da implementação de programas de comunicação convenientes.

2.2.1 Arquitecturas possíveis para as estações remotas.

Escolhida a arquitectura geral do sistema de controlo da regularidade do fio têxtil, debrucemos-nos sobre a estrutura de cada uma das estações remotas de aquisição de dados.

A arquitectura das estações remotas, podendo derivar de grande diversidade de estruturas, terá de garantir alguns objectivos básicos; estes - já atrás enunciados - prendem-se essencialmente com a optimização da relação "performance"/custos.

Usando estes critérios, centrou-se o estudo inicial de arquitecturas em 2 fases, associadas uma à escolha do elemento processador e outra às características da aquisição de dados.

Por fim, procurou decidir-se qual a solução mais eficaz para implementar a comunicação entre microcomputador e estações remotas de aquisição de dados.

As hipóteses consideradas na escolha do elemento processador foram as seguintes:

- Usar um processador de uso genérico;
- Usar um processador orientado para o processamento de sinal **[32, 36, 47, 66]**;

Antes de optar por uma destas hipóteses, definiram-se duas soluções , que foram cuidadosamente analisadas (como se apresenta na secção seguinte).

A primeira (figura 4) usa um microcontrolador da família Intel MCS-51, que alia as vantagens de possuir uma unidade processamento poderosa e diversos periféricos integrados.

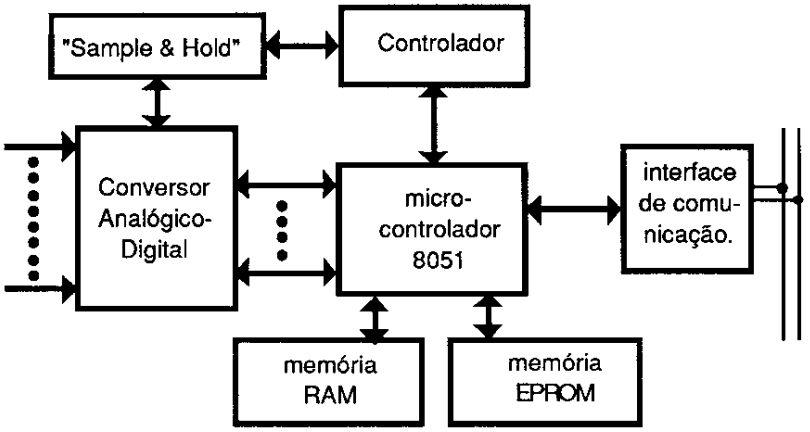


Figura 4. Arquitectura da estação baseada no microcontrolador 8051.

A segunda hipótese de arquitectura (figura 5) baseia-se num processador geralmente utilizado para processamento digital de sinal , pertencendo à família NEC μPD7720.

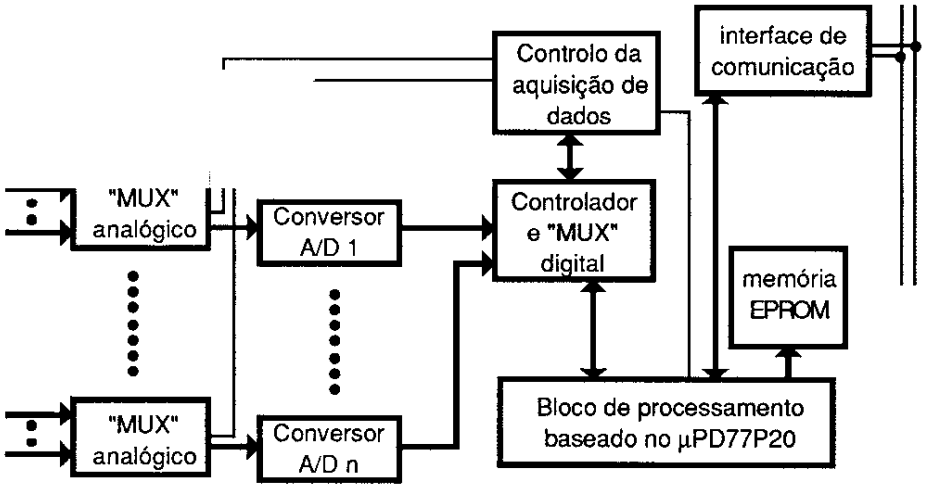


Figura 5. Arquitectura da estação baseada no microprocessador NEC 77P20.

Estes apresentam diversas vantagens em termos de tempo de processamento já que recorrem a um processamento paralelo (tipo Harvard). Nesta arquitectura evitam-se os conflitos do barramento ao serem separados os caminhos das instruções e dos dados; por outro lado, recorrendo ao "pipelining" é possível efectuar a execução de diversas tarefas concorrentemente. Este processador, ocupando um só circuito integrado, pode ser encadeado em cascata e liga-se aos periféricos (incluindo os conversores) através de linha série.

A nível de aquisição de dados as restrições a impor não são de molde a exigir a utilização de conversores analógico-digitais com elevadas "performances" em termos de tempos de aquisição e número de bits de conversão; para o tipo de aquisição pretendida, é suficiente a quantificação em cerca de 100 níveis, pelo que um conversor de 8 bits é aceitável. Em termos de velocidade de conversão, não antecipando a discussão que se apresenta nas secções seguintes, não são de prever necessidades muito restritivas. Salvaguarda-se o facto de, para cada uma das opções ser efectuado a interface entre processador e periférico de modo diferente: em paralelo numa (8051) e série noutra(NEC).

As diversas estações deverão estar interligadas entre si e a um microcomputador central (μC). Esta interligação, no sentido de reduzir custos de instalação e manutenção, deverá ser série, e se possível seguindo um "standard" oficial ou pelo menos industrial **[1, 21]**.

Foi com base nestes dados, ao que poderemos acrescentar o facto de na 1ª arquitectura se utilizar um microcontrolador da família Intel MCS-51, que optámos, ***como referência***, pelo "standard" industrial BitBus **[7, 26]**, especialmente no que se refere às camadas respeitantes ao protocolo físico (RS-485) e de controlo de ligação (protocolo orientado ao bit tipo HDLC/SDLC) **[14, 17, 18, 22, 23, 24, 25]**.

O estudo comparativo que se seguirá será centrado nos seguintes 3 pontos:

- **ritmo da aquisição de dados;**
- **velocidade de processamento dos diversos intervenientes;**
- **velocidade de comunicação (ritmo de transferência de dados);**

Esta análise comparativa considera comuns às duas arquitecturas alguns aspectos relacionados com a minimização dos tempo associados a:

I - É possível diminuir consideravelmente o tempo despendido numa comunicação se tivermos em conta que, em condições de funcionamento normal não é necessário enviar um grande lote de informação entre as estações remotas e o microcomputador central, dada estas terem boa capacidade de processamento e de armazenamento de informação.

II - A frequência de amostragem a utilizar na aquisição de dados de um determinado canal não necessita ser fixo; utilizando uma frequência auto-ajustável (mais elevada quando perante situações de erro) será viável utilizar um tempo de conversão maior para os canais sem erros de uma determinada estação, e um mais curto para o ou os canais que apresentaram situações de erro e necessitam uma análise mais cuidada.

III - O tempo de processamento no microcomputador central é consideravelmente diminuído com a utilização destas estratégias, aliviando-o de tarefas repetitivas e de pouca utilidade.

2.2.2 Análise de 2 arquitecturas de estações remotas apresentadas.

Sendo certo que este tipo de equipamento pode estar na base do controlo de produção de diversos materiais, foi contudo projectado para garantir plenamente o caso do regularímetro digital. É com base em valores conhecidos do funcionamento de bobinadeiras e contínuos que vamos basear o estudo. Sendo,

∂ (mm) -> distância mínima de fio que é analisada.
(este valor ∂ corresponde à distância mínima que o sensor pode analisar imposta na prática pelo comprimento de referido sensor - ver figura 2).

$\Omega\left(\frac{m}{mn}\right)$ -> velocidade de deslocamento do fio.
(esta velocidade é imposta pelo movimento de tracção dos fusos das bobinadeiras ou pela tracção de dois rolos que, provocando a estiragem do fio, o vão puxando).

Para adquirir correctamente informação sobre a espessura de um fio é necessário uma frequência de amostragem f,

$$f = \frac{\Omega * 1000}{60 * \partial} \text{ (Hz). equivalente a } \Omega = \frac{f * 60 * \partial}{1000} \qquad (II.1)$$

permitindo obter informação de segmentos contíguos de fio, com o comprimento ∂.

Implementando uma estação usando a arquitectura da figura 4 (baseada no 8051), em que cada estação de aquisição de dados adquira informação de 16 canais multiplexados, usando um conversor A/D com tempo de conversão de 100 μs obtemos uma frequência de amostragem f de 625 Hz por canal.

Usando a expressão (***II.1***) a velocidade máxima possível de analisar é $\Omega = 37.5 * \partial$; para um valor típico de ∂=8 mm teríamos como velocidade máxima de deslocamento de fio o valor de 300 m/mn, que está em conformidade com os valores dos contínuos e de uma grande generalidade das bobinadeiras.

Contudo, prevendo a necessidade de garantir frequências de amostragem mais elevadas, e partindo do princípio que se utilizaria este tipo de arquitectura, podemos alterar a frequência de amostragem para valores até 16 vezes o referido sem substituir o conversor analógico-digital; basta diminuir o número de canais por estação, o que obriga no entanto ao aumento do nº de estações (para a mesma quantidade de canais no sistema). Assim será possível garantir frequências de amostragem até 10 KHz a que corresponderiam, para ∂=8 mm, velocidades até aproximadamente 4800 m/mn, garantindo o funcionamento em qualquer tipo de bobinadeira (relembre-se que industrialmente são ainda consideradas velocidades elevadíssimas valores inferiores a mil metros por minuto). Em caso de necessidade poderíamos ainda substituir o conversor A/D por um circuito com menor tempo de conversão.

Nesta arquitectura adoptou-se para o conversor analógico-digital um tempo de conversão por canal na ordem dos 100 μs, apesar de existirem disponíveis diversos modelos mais rápidos; esta escolha tem razão de ser dado o microcontrolador despender entre 1 e 2 μs em cada instrução (excepto multiplicação e divisão em que o tempo é de 4 μs). Durante o serviço de um determinado canal o microcontrolador está a analisar e operar os dados relativos ao canal lido anteriormente, o que permite concluir que 100 μs é o tempo que o processador local tem para dedicar a cada canal.

Este tempo não é particularmente dilatado (permite, em média, a execução de 50 a 70 instruções), o que leva a que não baste colocar um conversor mais rápido para **podermos garantir** uma frequência de amostragem mais elevada. Para o caso do microcontrolador em causa, sendo previsível a execução, no mínimo, de 50 instruções entre cada aquisição, compreende-se esta afirmação. Esta conclusão é igualmente válida para a substituição do microcontrolador mantendo o conversor. Só uma modificação mais global permitirá uma melhoria sensível.

A arquitectura da figura 5, baseia-se num microprocessador bastante rápido, o NEC μPD77P20, e em conversores analógico-digitais, com um tempo de conversão extremamente pequeno (para não impor um estrangulamento na "performance" da placa). Com esta configuração poderemos tentar impor as piores condições (em termos de tempo).

Assim, usando a expressão (***II.1***) e para uma velocidade Ω = 500 m/mn (∂ = 8 mm, e 16 canais por estação), necessitaríamos garantir um frequência de amostragem f = 6.25 KHz; o tempo de conversão para os diversos canais teria de ser inferior a 160 μs, condição fácil de cumprir, existindo a título de exemplo um conversor analógico-digital NEC 7003 com um tempo de conversão de 4 μs, perfeitamente interfaceável com este microprocessador.

Estes valores permitem ainda que um determinado conversor possa ser utilizado para vários canais por meio de um multiplexador analógico (MUX, 2÷1 no presente exemplo), diminuindo o número de nós indispensáveis na rede e, por arrastamento, a complexidade desta arquitectura. É inclusivamente possível implementar este nó utilizando apenas um conversor analógico-digital e um multiplexador de 16÷1, minimizando assim os custos de cada estação.

Relativamente ao tempo de processamento disponível para cada valor adquirido, como o μP está preparado para uma ligação em cascata de diversas unidades de processamento, realizando cada uma apenas uma parte do algoritmo, é apenas necessário garantir, em termos de programação, que em cada um dos elementos da cascata não é ultrapassado o tempo crítico de $\frac{160}{16}$ =10 μs.

A possibilidade de encadear, em cascata, diversos destes processadores (um pouco na filosofia dos "transputers") **[32, 66]**, garante a eficácia desta arquitectura para cumprir os 'tempos limitadores'; basta garantir que o tempo de um elemento processador receber um dado, processá-lo e passá-lo ao elemento seguinte esteja de acordo com o tempo de aquisição.

Concluindo, para as 2 arquitecturas consegue-se garantir o funcionamento com valores de velocidade do fio:

estação de 8 canais,	Ω = 600 m/mn	**(1ª arq.)**	Ω = 3750	m/mn	**(2ª arq.)**
estação de 16 canais,	Ω = 300 m/mn	**(1ª arq.)**	Ω = 7500	m/mn	**(2ª arq.)**

Considerando os valores máximos de velocidade existentes, conclui-se a exequibilidade de qualquer uma das arquitecturas é exequível.

2.2.3 Análise das restrições impostas à comunicação.

Para uma frequência de transmissão de informação de 2.4 Mbit/s (velocidade máxima definida num dos modos de funcionamento do "standard" BitBus), é possível transmitir, durante o tempo em que uma estação está ocupada a servir um canal, 240 ou 9.6 bits, conforme a arquitectura escolhida seja a 1ª ou a 2ª.

Se pretendermos uma transmissão de informação para o microcomputador central que permita uma análise por este último, de todos os dados recolhidos e se, além disto, impusermos que esta transmissão seja efectuada em tempo real - situação pouco provável se atendermos à quantidade de informação a 'digerir' - então, durante o tempo de conversão dos X canais existentes num nó terá de ser feita a transmissão dos dados referentes a N canais. Equivale a dizer que durante o tempo de serviço desses X canais, igual a $X * t_{conv}$, terão de ser enviadas as informações referentes a N canais.

Por outro lado, para a transmissão "on-line" de dados ter 'significado', e garantirmos que o mau funcionamento de uma estação não acarrete a falha de todo o sistema, é necessário que a informação a transmitir seja correctamente reconhecida pelo microcomputador, obrigando a informação a recorrer a um protocolo; este definirá os pacotes de informação a transitar e incluirá um campo que identifique qual a estação e/ou o canal a que diz respeito a informação enviada.

Estudámos assim diversas soluções alternativas de protocolo, na perspectiva de verificar a possibilidade (ou vantagem) da selecção de uma arquitectura em detrimento da outra.

Foi também analisado o tráfego (**g**) existente na rede de comunicação usando protocolos tipo HDLC/SDLC, com vista a verificar se não existe uma boa probabilidade de ocorrer uma situação de ruptura (nomeadamente pela existência de alguns erros eventuais, decorrentes da colisão de "frames" enviados em simultâneo).

Para X=16 N=1024, **g** será superior a 100% inviabilizando o funcionamento da rede; para X=16 e N=250 então $g \approx 65$ %. Este valor, embora parecendo aceitável, obrigaria a uma perfeita sincronização de todas as comunicações, com vista a garantir a não ocorrência de colisões nas transmissões a efectuar; não permite ainda o recurso a alguns métodos de acesso ao meio de comunicação, nomeadamente os do tipo CSMA-CD, que estatisticamente colapsam a partir de uma taxa de ocupação desta ordem de grandeza **[2, 9, 11, 13, 15, 16]**.

Convêm ainda realçar que este aspecto é agravado, dado não terem sido considerados alguns atrasos que podem ocorrer, de que se destacam os:

- devidos a repetições de parte de comunicações originadas pela detecção de erros em transacções anteriores;
- devidos ao tempo que decorre entre cada mensagem. Mesmo existindo uma comunicação perfeitamente sincronizada não é possível garantir que uma estação inicie imediatamente a comunicação;
- devidos ao tempo despendido no estabelecimento da ligação lógica para a comunicação se efectivar. Este atraso poderia ser obviado se optarmos por uma solução em que todas as estações estão permanentemente ligadas à estação central;

Além dos factores apresentados, que limitam consideravelmente a eficiência da comunicação se enviarmos todos os dados adquiridos para a estação central, existem alguns aspectos adicionais que nos levaram a não considerar esta solução:

- Em situações de emergência é aconselhável uma actuação a muito curto prazo, tradicionalmente a ser efectuada pela estação onde é feita a aquisição de dados. Nestes casos seria provavelmente ineficaz e redundante efectuar esta transmissão para a unidade central. Será provavelmente mais eficiente enviar a estação remota uma informação do ocorrido e da acção tomada.

- Quando é necessário uma resposta do microcomputador para a estação que efectuou uma transmissão, esta última tem de esperar que este envia a respectiva resposta; nesse caso o microcomputador central terá de receber a informação, processá-la com vista à tomada de uma resolução e devolvê-la com sucesso à estação. Neste tempo, o microcomputador esteve ocupado em grande parte a dar atenção a este problema particular, obrigando assim ao não atendimento de outros pedidos; por outro lado a estação teve de aguardar a resposta sem nenhuma acção particular.

- Quanto maior for o "software" distribuído pelas diversas estações mais tarefas podem ser atribuídas ao microcomputador central (nomeadamente trabalho estatístico e de controlo da produção). Seria absurdo possuir uma arquitectura multiprocessadora distribuída sem utilizar os seus recursos de cálculo para melhorar a eficiência de todo o sistema.

- A transmissão de toda a informação adquirida deve obrigar a uma resposta do microcomputador (nem que seja de reconhecimento de boa recepção), o que faria com

que o tempo gasto em comunicações subisse, no mínimo, até um valor próximo do dobro do calculado anteriormente.

- Se todo o processamento fosse feito "on-line" o microcomputador poderia dispor para cada canal de um tempo de processamento muito curto.

Estas considerações confirmaram a a metodologia adoptada:

- Dotar as estações de aquisição de dados com a maior fatia possível de programação.
- Incluir nessa fatia as respostas a situações de emergência.
- Fazer parte do processamento digital de sinal nas estações de aquisição de dados.
- Transmitir para o microcomputador central apenas informação relevante, o que provavelmente passará pela transmissão de quando e quantas vezes ocorreram erros.
- Utilizar o microcomputador central apenas para trabalho estatístico e de controlo da produção.
- Transmitir o conjunto de dados tal como adquiridos, apenas em situações em que, dada a precisão e rapidez pretendidas, é indispensável recorrer a um processador de maiores recursos.

2.2.4 Informação a transmitir.

Vejamos algumas alternativas para definir a informação a enviar por cada estação ao microcomputador central, tendo em vista **validar** as opções tomadas, garantindo a eficiência do sistema de comunicações:

1 - Enviar, relativo a cada canal, um campo de informação referente às irregularidades detectadas com dados relativos ao seu número, média dos valores detectados nas irregularidades, média do tempo entre cada uma do mesmo tipo, desvio padrão dos valores, desvio padrão do tempo entre as ocorrências. Deve ainda incluir o valor dos limites, superior e inferior, para a detecção destas irregularidades.

2 - Enviar um campo de informação que contenha todos os dados compactados. Assim, além da informação referente aos limites para a detecção de erros (ou imperfeições para o caso do regularímetro), enviaria o nº de bytes detectados sem erro,

seguido do valor do erro, seguido do nº de bytes sem erro, seguido do novo erro, etc.. Esta alternativa tem um campo de informação de tamanho variável.

3 - Esta alternativa é um misto das 2 anteriores. A forma de reunir as 2 alternativas anteriores pode contudo obedecer a várias filosofias, que serão posteriormente discutidas.

Não foi excluída a possibilidade de, em determinadas circunstâncias e dada a gravidade da situação, ser enviado um ou mais pacotes com um conjunto de dados adquiridos (quer a informação adquirida quer resultados de processamento local); esta situação torna-se também viável para o caso de se pretender funcionar com o sistema para a análise de apenas 1 fio, como acontece num sistema de controlo de qualidade, ou seja, em laboratório.

Para além destas 3 alternativas, chegaram a ser consideradas outras que abandonámos de seguida por, ou não permitirem à unidade central a publicação dos relatórios típicos da indústria têxtil devido à insuficiência dos dados, ou obrigarem à utilização de um número demasiado elevado de recursos.

Analisam-se de seguida as 3 primeiras alternativas apresentadas, tendo em conta o tipo de imperfeições detectadas na análise regularimétrica de fio têxtil.

2.2.4.1 Alternativa 1.

No caso da 1ª alternativa a distribuição da ocupação da informação a enviar por cada canal seria do tipo do indicado no quadro 1.

Tipo de informação	NEP	Pontos Grossos	Pontos Finos
nº de irregularidades	1 byte	1 byte	1 byte
média de irregularidades	1 byte	1 byte	1 byte
desvio padrão das irregularidades	1 byte	1 byte	1 byte
média de tempo entre irregularidades	1 byte	1 byte	1 byte
desvio padrão do tempo entre irregularidades	1 byte	1 byte	1 byte
limite da irregularidade	1 byte	1 byte	1 byte
	Total de 18 bytes.		

Quadro 1. Nº de bytes a transmitir por cada canal segundo alternativa 1.

Neste caso, um canal só teria de enviar uma unidade de comunicação (embora fosse indispensável fazê-lo), quando algum destes subcampos estivesse à beira de atingir o "overflow". Como a média e o desvio padrão são sempre inferiores ou iguais a 255, tal como os limites de detecção das irregularidades (dado os ADC considerados, serem de 8 bit), não teremos de nos preocupar com estes items.

No que diz respeito aos outros, quer o nº máximo de irregularidades, quer o nº máximo de amostras entre irregularidades, sendo o "overflow" atingido no valor 255, é necessário limitar o número máximo de ocorrências de irregularidades antes de ser enviada uma comunicação.

Assim o tempo disponível para ser necessário garantir uma comunicação ocorre quando se verifica uma situação de erro **permanente**, ou seja 25.5 ms na 1ª arquitectura e 1.02 ms na 2ª.

Se atendermos a que este tempo só é atingido numa situação de erro grave, é previsível que a situação não seja resolvida imediatamente pela estação de aquisição, sendo natural que os tempos venham a ser ainda mais agravados.

Utilizando o protocolo HDLC/SDLC, para a transmissão deste 'pacote' de informação seria necessário dispôr de um "frame" de 23 bytes, correspondente aos 18 bytes de informação propriamente dita e de 5 bytes suplementares de controlo e supervisão (também neste caso a utilização de outro protocolo não inviabiliza este estudo, pelo menos se considerado como referência).

Mesmo agrupando num "frame" a informação de 2 canais, teremos uma média de $20.5 = \left(\frac{18*2+5}{2}\right)$ bytes por canal, com cada "frame" constituído por 41 bytes, concluimos só ser possível o correcto funcionamento para o máximo de 373 canais na arquitectura baseada no 8051 e 15 canais na baseada no microprocessador NEC.

Dado o baixo valor de N é consideravelmente limitado o uso desta alternativa; para uma aplicação de 1000 canais, seria impossível cumprir as especificações e, para 250 canais, seria obtida uma taxa de ocupação do canal de comunicação elevada.

Uma modificação que permite melhorar estes valores baseia-se no aumento do tamanho dos campos que limitam a 255 o número de erros possíveis de ocorrer antes de efectuar uma comunicação; estendendo esse limite a um valor mais elevado, por exemplo 65535, passamos a ter uma estrutura de "frame" como a indicada no quadro 2.

Baseando-nos nesta definição dos campos do "frame" a enviar pelas estações, teremos as limitações do número de estações agora em ≈66000 e 2600 respectivamente.

Concluiu-se assim ser possível efectuar com toda a segurança a comunicação na 1ª arquitectura conquanto para a 2ª não seja possível garanti-lo (pelo menos com a

mesma certeza), tendo em conta o tempo despendido na resposta do microcomputador central, para além dos problemas associados ao volume de tráfego imposto.

Tipo de informação	NEP	Pontos Grosso	Pontos Finos
nº de irregularidades	2 bytes	2 bytes	2 bytes
média de irregularidades	1 byte	1 byte	1 byte
desvio padrão das irregularidades	1 byte	1 byte	1 byte
média de tempo entre irregularidades	2 bytes	2 bytes	2 bytes
desvio padrão do tempo entre irregularidades	2 bytes	2 bytes	2 bytes
limite da irregularidade	1 byte	1 byte	1 byte
	total de 27 bytes		

Quadro 2. Bytes a transmitir referentes a cada canal, usando a alternativa 1 e com alguns dos campos com 2 bytes de comprimento.

No entanto, optando pela 1ª arquitectura, com um microcomputador 8051 ou 8052 e não o 8044, não é possível utilizar uma velocidade das comunicações de 2.4 Mbit/s (utilizada no BitBus e nos cálculos precedentes); além disso não temos acesso ao protocolo HDLC, sem modificar profundamente a própria arquitectura da estação. Esta última limitação é facilmente colmatável (como veremos noutro capítulo) com o recurso a um protocolo definido por nós e que, dada a estrutura assíncrona da unidade de comunicação, seja orientado ao caracter (ou byte); é aconselhável, no entanto, manter uma estrutura de "frame" similar à do HDLC, garantindo assim o mesma interface com a programação de nível superior .

Relativamente à primeira objecção (velocidade) a solução do problema não é tão simples. É possível utilizar uma velocidade até 1 Mbit/s para a comunicação mas nesse caso seria impossível utilizar todas as potencialidades deste microcontrolador, especialmente no que diz respeito à comunicação não interromper nem mesmo perturbar o normal funcionamento das estações para as quais ela não é dirigida. Esta impossibilidade construtiva prende-se com o facto deste protocolo necessitar de 11 bits para a composição de cada grupo de 8 bits de informação e o 8051 para enviar ou receber informação a 1Mbit/s só permitir utilizar 8 bits. Estaríamos assim limitados a uma velocidade máxima de 375 Kbit/s, utilizando um protocolo de controlo de ligação orientado ao caracter. Para o efeito será lógico considerar a limitação física das "portas" de comunicação série existentes nos microcomputadores do tipo IBM-AT, que impõem um limite de 115 Kbaud (115 Kbit/s).

Considerando esta limitação, calculemos de novo os parâmetros anteriormente analisados para a 1ª arquitectura.

A taxa de ocupação do canal de comunicação **g** (=43%) só toma um valor aceitável se considerarmos uma taxa de erro ≤ 20 %, caso em que **g = 8.6 %**.

O valor máximo de N na 1ª arquitectura será definido pela expressão $\frac{11 * (5 + K + 27 * J) * \frac{N}{J}}{115.2 * 10^3} \leq 65535 * 100 * 10^{-6}$, utilizando uma metodologia de cálculo igual a estudada anteriormente (em que N é o nº de canais analógicos da rede, X o nº de canais por nó da rede, J o nº de canais enviados em cada "frame", K é uma variável auxiliar com valor 0 se X ≥ 4 e K=1 se X < 4); para X=16 (logo K=0) e J=2 apresenta **N ≤ 2326.5** (g=100%).

Estes valores são aceitáveis, embora imponham algumas limitações relativas ao tráfego na linha de comunicação.

A restrição associada à velocidade da comunicação não existe para a 2ª arquitectura, dado o microprocessador μPD77P20 não possuir qualquer módulo interno responsável pela comunicação. A adopção desta última arquitectura obrigaria assim a implementar uma estação que possuisse um dispositivo dedicado à comunicação, com a velocidade que entendermos conveniente; este facto permite-nos considerar como viável os 2.4 Mbit/s. A única limitação a colocar prender-se-ia com a transferência de 1 byte entre este microprocessador e o dispositivo de saída que, sendo feita à velocidade máxima de 2Mbyte/s, obrigaria à eventual construção de uma memória tampão ("buffer") entre os dois dispositivos **[23, 3, 4]**.

2.2.4.2 Alternativa 2.

Nesta alternativa, a informação a enviar será constituída pelo número de amostras sem ter sido detectada qualquer irregularidade, seguido da irregularidade, seguido do número de amostras sem irregularidades, e assim sucessivamente. Constituem ainda parte do campo de informação dados relativos aos limites destas irregularidades.

O campo de informação terá o nº máximo de bytes quando existirem permanentemente erros ou existir alternadamente 1 erro e 1 valor correcto. Devido às considerações relativas à dimensão de memória rápida disponível na 1ªarquitectura para construir "frames", vamos limitar o tamanho deste campo de informação a 32 byte, em que são enviados dados de 2 canais. No campo de informação incluem-se os 3

limites de detecção de irregularidades, pelo que cada "frame" terá o total de 43 bytes divididos pelos 5 bytes de controlo + 3 bytes com os limites das irregularidades para cada canal + 32 bytes representando os 16 bytes com informação de cada canal. De cada um deste grupo de 16 bytes, 8 representam o número de bytes que não ultrapassaram os limites de irregularidades e os outros 8 (que aparecem alternados com os anteriores) representam o valor atingido pela irregularidade. No sentido de minimizar a informação a enviar impomos que esse valor seja representado com 6 bits, sendo reservados os 2 bits adicionais à identificação do tipo de irregularidade detectada.

Verificando, usando um processo análogo ao anterior, qual o valor máximo de canais analógicos na rede teremos, considerando a maior taxa de erros, $N \leq 11.2$ e $N \leq 0.45$ para a 1ª e 2ª arquitecturas respectivamente.

Estes resultados inviabilizam, por completo esta alternativa, considerando ou não a hipótese de ocorrer resposta pelo microcomputador central.

Podemos no entanto impor algumas condições; assim, não necessitamos de considerar que o número de bytes que exigem a transmissão é imposto pelo taxa de erro máxima. Obviamente que, ao aceitarmos definitivamente esta consideração, seremos obrigados a recorrer a uma metodologia alternativa de comunicar, adoptando na prática a solução 3. Necessitamos contudo de verificar se é possível utilizar uma compactação dos dados (alternativa 2), suficientemente rentável se comparada com outras, permitindo a sua inclusão como parte de um algoritmo misto.

Considerámos um aumento da zona de memória e, por outro lado, um limite do funcionamento desta alternativa a uma determinada taxa de erros. O aumento da zona de memória para o campo de informação a enviar na 1ª arquitectura, não apresenta grandes dificuldades. Optando pela utilização de um microcontrolador 8052 podemos inclusivamente usar 128 bytes, pelo que poderemos escrever a inequação:

$$\textbf{tempo de transmissão do frame} \leq \textbf{tempo de conversão de 1 canal} * R * \frac{128}{J}$$

(notar que $R \leq 255$)

$$8\,\frac{(5+3*J+K+128)\frac{N}{J}}{V_{com}} \leq t_{conv} * R * \frac{128}{J} \text{ em que}$$

$N \rightarrow$ Nº de canais analógicos da rede; *$R \rightarrow$ Nº mínimo de amostragens s/ erro;*

$V_{com} \rightarrow$ Velocidade de transmissão; *$t_{conv} \rightarrow$ tempo de conversão de 1 canal;*

$J \rightarrow$ Nº de canais c/ dados num "frame"; *$K \rightarrow$ 1 se nº de canais/estação < 4;*

Nota: Nos quadros seguintes considerou-se J=2 e K=0.

V_{com}	**1ª arquitectura**	**2ª arquitectura**
2.4 Mbit/s	$R \geq \frac{N}{27.63}$ (se N=1000 R ≥36.2)	$R \geq \frac{N}{1.105}$ (se N=1000 R ≥905)
115.2Kbit/s	$R \geq \frac{N}{1.33}$ (se N=1000 R ≥754)	$R \geq \frac{N}{0.053}$ (se N=1024 R ≥ 18853)

Estes resultados não apresentam nenhuma melhoria de vulto relativamente ao caso apresentado inicialmente; se tomarmos em consideração o facto de cada elemento (1 byte), do campo de informação, com o nº de amostras sem erro (**R**) não poder exceder 2^6=64, ficamos limitados ao uso desta alternativa apenas na 1ª arquitectura à velocidade de 2.4 Mbit/s.

Este problema é facilmente ultrapassável se cada elemento do campo de informação passar a ocupar 2 bytes, sendo reformulada a relação anterior para:

$8\frac{(5+3*J+K+128)\frac{N}{J}}{V_{com}} \leq t_{conv}*R*\frac{128}{2*J}$ a que acrescentamos a condição R ≤ 16384,

e em que

N → Nº de canais analógicos da rede; *R → Nº mínimo de amostragens s/ erro;*
V_{com} *→ Velocidade de transmissão;* t_{conv} *→ tempo de conversão de 1 canal.*
J→ Nº de canais c/ dados num "frame"; *K→ 1 se nº de canais/estação < 4;*

V_{com}	**1ª arquitectura**	**2ª arquitectura**
2.4 Mbit/s	$R \geq \frac{2N}{27.63}$ (se N=1000 R ≥72.4)	$R \geq \frac{2N}{1.105}$ (se N=1000 R ≥1809)
115.2Kbit/s	$R \geq \frac{2N}{1.33}$ (se N=1000 R ≥1504)	$R \geq \frac{2N}{0.053}$ (se N=1024 R ≥ 37736)

Estes valores já permitem o funcionamento da estação para as duas arquitecturas, sob certas condições, sendo de realçar o facto de ter sido **muito** agravada a "performance"; esta é caracterizável pelo período mínimo para a ocorrência de imperfeições (este valor é idêntico a **R** ou $R*T_{amostragem}$ se pretendermos unidades temporais).

Podemos portanto concluir que esta alternativa não pode ser considerada, a não ser com a limitação da taxa de erro máxima admissível e/ou aumentando consideravelmente a memória disponível para construir o "frame"de comunicação. De realçar a sua eficácia para casos em que a taxa de detecção de irregularidades é baixa. Uma das vantagens da consideração desta alternativa prende-se com a detecção da frequência com que ocorrem as irregularidades.

2.2.4.3 Alternativa 3.

Esta alternativa obtem-se com uma combinação das 2 primeiras alternativas. estudadas. A "performance" desta alternativa resume-se a um estudo comparado das 2 primeiras e, eventualmente, na sua combinação.

Se nenhuma das considerações efectuadas com a análise teórica apresentada - para um sistema com determinado número de canais (totais e por estação) - for suficiente para uma decisão, podemos adoptar alguns dos parâmetros calculados (limite para a taxa de ocorrência de irregularidades) como o factor de decisão; neste caso o que decidirá, nesta alternativa, o ponto de transição entre o uso de uma alternativa como a primeira relativamente ao uso da segunda, será o número de irregularidades que vão ocorrendo.

2.2.5 Conclusão.

Neste momento, tendo em consideração todos estes dados podemos optar por uma das 3 alternativas iniciais. Relativamente às duas primeiras podemos concluir:

- A 1ª não contém nenhuma informação que não possa ser obtida da informação enviada pela 2ª alternativa.
- O "software" necessário para obter a informação contida na 2ª alternativa constitui uma fatia importante do "software" total com que se obtêm os dados que compõem o campo de informação da 1ª alternativa.
- A 2ª alternativa pode, por si só, quando a taxa de erros é superior a um certo limite, provocar uma ruptura do meio de comunicação. Os limites fixados para a taxa de erro são, usando o protocolo de controlo de ligação HDLC, os calculados no ponto anterior.

Sendo assim parece aconselhável utilizar uma combinação das 2 alternativas referidas, isto é, optando pela 3ª alternativa, que se resume a compor um campo de informação seguindo o método de uma das 2 primeira; a escolha por qual delas dependeria do tempo disponível para a comunicação e do nº de irregularidades detectadas.

No início do estudo, chegou a considerar-se um conjunto de soluções possíveis para esta alternativa 3 (ver figura 6).

Facilmente se verifica, com base nos estudos realizados para as alternativas 1 e 2, da impossibilidade de seleccionar a opção 3.b)

Por sua vez, a alternativa 3.a) está na 3.c), pelo que podemos desde já adoptar como princípio a utilização da última. No caso da taxa de erros ser demasiado elevada, esta alternativa será utilizada apenas no formato definido pela 3.a).

Esta alternativa seria um misto das 2 anteriores. A forma de reunir as 2 alternativas anteriores pode contudo obedecer a várias filosofias, que serão posteriormente discutidas, nomeadamente:

a) - Utilizar tipicamente uma alternativa e passar para a outra em caso de erros demasiado frequentes.
b) - Utilizar em cada unidade de comunicação as 2 alternativas, o que levaria a que o campo de informação fosse ainda mais alargado.
c) - Utilizar em caso de erros frequentes uma alternativa ou, se o μC central o exigir, a reunião das 2 alternativas. Em caso de funcionamento dentro dos padrões vulgares (nº de erros aceitável), utilizar a alternativa com menor nº de bytes.

Figura 6. Descrição das hipóteses alternativas consideradas para a alternativa de comunicação 3.

Sendo assim, a optar por uma metodologia deste tipo, deveria ser uma semelhante à definida pelo algoritmo da figura 7.

```
SE o nº de erros é pequeno
ENTÃO
    SE tem tempo para transmitir
        ENTÃO: utiliza um campo de informação com a alternativa 2 e 1.
        SENÃO: utiliza a alternativa com menor número de byte.
SENÃO
    SE tem tempo para transmitir
        ENTÃO: utiliza um campo de informação com a alternativa 2 e 1.
        SENÃO: utiliza a 1ª alternativa.
```

Figura 7. Algoritmo para a composição de um eventual "frame" que garante o funcionamento do sistema sem entrar em colapso.

Nada impede no entanto que, existindo uma perturbação frequente apenas num ou em alguns dos canais, seja definido um algoritmo diferente para cada canal de comunicação, permitindo eventualmente a transmissão de toda a informação recolhida durante um certo período de tempo em que foi detectada uma anomalia grave. Com este processo, é possível recorrer a algoritmos de processamento digital de sinal bastante complexos, que poderão ser executados em tempo diferido ("off-line") no microcomputador central, fazendo uso quer das suas possibilidades gráficas, quer das suas possibilidades de armazenamento bem como o seu potencial de processamento.

Com esta análise, efectuada no início do trabalho, verificou-se que é possível garantir um funcionamento correcto, usando qualquer das alternativas. Considerando assim que a 1ª arquitectura satisfaz, no essencial, as necessidades de processamento da informação, assegurando-se que a comunicação é perfeitamente garantida, optou-se por esta arquitecura uma vez que apresenta diversas vantagens associadas ao uso de um microcontrolador Intel da família MCS-51.

2.3 Funcionamento autónomo ("stand-alone").

O funcionamento em rede de diversas estações num ambiente industrial normalmente adverso - devido ao elevado ruído eléctrico existente, grande quantidade de poeiras de pequenissima dimensão, elevadas vibrações mecânicas produzidas pela maquinaria, etc - está sob as piores condições possíveis.

Nestas condições é previsível que a taxa de falhas da rede, motivada por exemplo pelo desligar de um conector do microcomputador, seja bastante mais frequente que num ambiente laboratorial, por muitos testes que se efectuem.

Sendo necessário, ou pelo menos conveniente, a manutenção em funcionamento da maior parte possível do sistema, será preciso definir um modo de operação tal que, no caso de falha de um elemento da rede ou mesmo da rede, seja possível continuar a garantir o pleno funcionamento de um certo número de operações básicas.

Naturalmente, este variará conforme o tipo, gravidade e número de elementos afectados pela ocorrência da avaria. Mais, será de todo em todo aconselhável que o operador do microcomputador central tenha possibilidade de detectar essa eventual avaria e apresente um primeiro diagnóstico relativamente às acções a empreender para solucionar o problema.

Para conseguir garantir este tipo de funcionamento é necessário que esteja implementado um módulo de programação nas estações remotas de aquisição de

dados que permita um funcionamento pleno no que diz respeito ao controlo das funções associadas ao regularímetro digital.

Para se atingir este objectivo definiram-se diversas estratégias para a escrita e actualização dos módulos responsáveis pelo processamento nas estações remotas. Estas estratégias garantem o funcionamento das estações remotas mesmo no caso da falha da rede ou de parte dos orgãos disponíveis na estação, sendo mais desenvolvidas noutro ponto deste texto.

De entre os módulos de programação implementados no microcomputador central destaca-se de o de análise de avarias e apresentação de acções de correcção, desenvolvido para uma utilização na produção do fio têxtil. Este módulo apresenta contudo a caracteristica de permitir um uso mais generalizado, possibilitando a sua utilização para a detecção de eventuais avarias deste sistema bem como para uma primeira apresentação de correcções a efectuar com vista a colmatar as deficiências encontradas.

2.4 Reformulação de objectivos.

A garantia do cumprimento das especificações definidas na introdução não obriga contudo à construção de um sistema de uso restrito à indústria de fiação têxtil.

Considerou-se assim como objectivo prioritário, a implementação de um sistema versátil que, mercê de programação adequada, permita a sua utilização em diferentes aplicações. Esta opção permitiu que parte do trabalho desenvolvido para este projecto, nomeadamente as estações remotas de aquisição de dados, e alguma programação desenvolvida fossem utilizadas em projectos distintos, como se apresenta noutro capitulo.

Capítulo 3. Controlo a partir do microcomputador central.

Com vista a garantir um controlo eficiente de todo o sistema é necessário que as estações locais executem diversas tarefas, sem qualquer intervenção do microcomputador central.

O algoritmo a estabelecer nestas terá de ser adaptado à tarefa que se pretende controlar, mesmo que essa tarefa corresponda apenas a uma monitorização do equipamento sob controlo.

Apesar deste algoritmo decorrer independentemente da actividade do microcomputador deverá existir sempre a possibilidade deste último intervir na evolução desses algoritmos de controlo.

O processo como essa intervenção se efectua pode assumir diversas formas, dependendo das características do algoritmo de controlo implementado nas estações remotas.

Para a implementação da programação a residir nas estações de aquisição foram analisadas as seguintes metodologias:

A) - Implementar um algoritmo fixo, isto é, um programa adequado ao processo a controlar que permaneça inalterável ao longo do tempo. Esse algoritmo acederia no entanto a primitivas de comunicação com a estação mestre, com vista a transmitir resultados e/ou dados.

B) - Construir um pequeno algoritmo fixo, que comunique com o microcomputador central com vista a receber deste um programa executável. Este programa seria carregado na memória e posteriormente executado, sendo ele próprio o algoritmo de controlo a utilizar.

C) - Implementar, nas estações remotas, um interpretador que receba do microcomputador central um programa escrito numa linguagem de mais alto nível. Esta linguagem, que poderia ser apropriada para o controlo, não seria directamente executável, cabendo essa tarefa ao referido programa de interpretação.

D) - Programar um grupo de comandos (um monitor) que serão executados de uma forma rígida, isto é, sempre de modo similar e por acção do microcomputador

central; estes comandos sendo suficientemente gerais seriam executados conforme as necessidades.

A opção por uma destas soluções apresenta, relativamente às restantes, vantagens e desvantagens. Assim, sendo verdade a afirmação que as hipóteses A e D apresentarão menores dificuldades de implementação, maior imunidade às eventuais incorrecções geradas pelo microcomputador e provavelmente velocidade de cálculo maximizada, podemos sempre contrapor as dificuldades associadas à rigidez e pouca flexibilidade quando comparadas com as alternativas B e C.

Efectivamente estas últimas (B e C) permitem uma fácil modelação do sistema para a prossecução de objectivos completamente diversos, conforme a utilização a efectuar. Realça-se contudo que, estas soluções necessitam, sempre, da existência de comunicação através da rede, ficando à partida inviabilizada o funcionamento minimamente correcto das estações quando o microcomputador falhar ou a rede não funcionar correctamente.

No presente projecto foram implementados módulos de "software" que executam cada uma destas soluções, tendo-se adoptado por uma solução de compromisso.

Com base nas considerações expostas, optou-se por implementar uma solução que utiliza cada uma das 4 opções definidas e articuladas da seguinte forma:

- No desenho das estações remotas de aquisição, previu-se a possibilidade de aceder à memória de escrita/leitura (RAM) para gravar código executável; esta preocupação tem razão de ser dado a unidade de processamento escolhida (8051 ou similares) separar a memória de código da memória de dados (que apresentando a vantagem de utilizar 128 Kbytes de memória tem a desvantagem de não facilitar a execução de programas residentes na memória de dados).

Esse código poderá ser acedido pelo processador da estação para execução.

- Desenvolveu-se um interpretador de uma linguagem de controlo, que permite receber, através da linha de comunicação (rede), um programa escrito nessa linguagem e que poderá posteriormente ser executado.

- Implementou-se um monitor que permite a execução de um grande número de comandos que actuam sobre a memória e todos os outros dispositivos da estação.

Entre os comandos desenvolvidos destacam-se os relativos ao processamento digital de sinal, nomeadamente o cálculo da Transformada Rápida de Fourier (FFT), autocorrelação, Coeficiente de Variação, etc.

- Foi construído um pequeno algoritmo fixo que será executado após uma falha de tensão ou a inicialização forçado do sistema (RESET); este programa é encadeado com um outro, implementado seguindo uma das metodologias anteriores. Este último executa um algoritmo de controlo mínimo, previamente estabelecido, cujo início de execução estará eventualmente guardado em memória não volátil. Este endereço poderá ou não ser alterado para os casos em que um novo algoritmo for introduzido e guardado em memória não volátil.

A implementação destas soluções será descrita, com maior pormenor, nos capítulos relativos ao desenvolvimento do sistema. Realça-se, no entanto, a grande versatilidade do sistema desenvolvido para cumprir especificações das mais diversas, especialmente dado não ser obrigatório optar por uma única solução, dependendo dos condicionalismos impostos, optar-se-á pela que será mais conveniente utilizar. Assim, no caso de existirem importantes imposições de tempo disponível, poder-se-á optar por exemplo por utilizar um programa desenvolvido e optimizado no microcomputador central e posteriormente guardado em memória não volátil (EPROM), seguindo a solução do algoritmo fixo.

PARTE 2.

CONCEPÇÃO.

Na parte do texto que agora apresentamos serão discutidos os aspectos teóricos associados à implementação e concepção do regularímetro digital.

Existem diversos parâmetros de medida da qualidade do fio calculados a partir das amostras deste fio; a implementação de um sistema como o que nos propusémos tem necessariamente de apresentar informação similar aos que se conseguem com os industriais. Este cálculo, explicitado em pormenor no ***capítulo 5*** , visa o conhecimento dos seguintes parâmetros:

- Coeficiente de variação (CV%),
- Desvio Absoluto da média ou Coeficiente de Irregularidade (U%) e
- Taxa de Variação (DR%).

Para além destes parâmetros é indispensável determinar o espectro das amostras de massa por unidade de comprimento, do fio têxtil; o resultado deste cálculo permitir-nos-á, para além do conhecimento do comprimento das fibras que entraram na constituição do fio, a detecção de alguns defeitos no processo de fabrico, como se explica com maior pormenor no ***capítulo 4***. Neste capítulo são ainda expostos, além do método de determinação de espectros, outros algoritmos de processamento de sinal construídos com vista a uma melhor parametrização das características do fio produzido e eventual detecção de imperfeições.

Com vista a um funcionamento eficaz de todo o sistema, os algoritmos que implementam parte destas funções foram desenvolvidos quer no microcomputador central (IBM-AT ou compatível), quer nas estações remotas, sendo a análise das suas atribuições efectuada com mais cuidado na *parte 3* (respeitante ao desenvolvimento).

Conforme referido, foi dedicada uma grande importância ao estudo de um sistema de ajuda à tomada de decisões (de correcção e/ou diagnóstico), quando é detectada uma anomalia; este módulo de programação é aqui discutido de uma forma genérica e conceptual; as particularidades relativas à implementação real no sistema em desenvolvimento serão focadas posteriormente.

Este tema é apresentado no ***capítulo 6***, antecendo a análise da comunicação a efectuar entre as estações remotas e o microcomputador central.

A discussão e análise das necessidades a nível de comunicações, tendo já sido efectuadas, servem como base para a definição dos conceitos básicos a aplicar na definição de toda a estrutura de troca de informações; embora no ***capítulo 7*** seja efectuada uma análise conceptual deste tema, não deixam de se apontar as pistas que levaram à definição real das primitivas globais implementadas, descritas noutra fase do presente trabalho.

Capítulo 4. Estratégias de processamento de sinal.

4.1 Determinação de espectros.

Um dos principais módulos implementados foi o da determinação do espectro de um sinal adquirido. O programa que realiza essa operação é importante para esta aplicação têxtil associada à determinação da regularidade de fios, podendo no entanto ser utilizado para o cálculo de sinais de outras origens **[29, 38, 41, 49]**.

No sentido de garantir que a apresentação de resultados do regularímetro digital seja idêntica à dos apresentados vulgarmente pelo equipamento industrial em uso, é de prever a possibilidade de esses resultados serem pelo menos de dois tipos: um dedicado à indústria têxtil, e outro mais geral para utilização para sinais distintos ou para uso por operadores mais especializados. Essa diferenciação far-se-á notar especialmente na forma de definir as riscas do espectro, em que cada harmónico está associado a um comprimento de onda para a utilização têxtil, e a uma frequência em Hertz para o caso geral **[61, 62, 64]**.

O conhecimento do espectro de um sinal que representa a massa de um fio têxtil é importante quer para determinar a composição do fio produzido, quer para detectar eventuais erros. Efectivamente as fibras que constituem um determinado fio provocam no espectro resultante um pico nas componentes sinusoidais de frequência correspondente ao comprimento dessa fibra (tradicionalmente alguns centímetros). Esse comprimento está relacionado com uma frequência (representada em Hz), relação essa que depende do período de tempo que decorre entre cada aquisição de dados do canal em causa (dependente da frequência de amostragem) e o comprimento de fio entre cada amostra adquirida. Sendo a aquisição feita para ***segmentos de fio têxtil contíguos*** e não havendo sobreposição entre nenhuma parte de fio analisado, a frequência máxima que será possível determinar corresponde a um comprimento de onda superior (igual no limite) ao dobro do comprimento do sensor.

Os erros referidos agora e que se podem detectar por análise do espectro, ao invés dos analisados noutro ponto deste trabalho, correspondem a componentes que, tendo eventualmente um período bastante grande, têm um "duty cycle" próximo de 50%.

Uma das causas mais frequentes para este tipo de perturbação é a apresentada no segundo exemplo da figura 8, provocada pela deposição de uma camada irregular de 'lixo' à superfície dos rolos que provocam a estiragem.

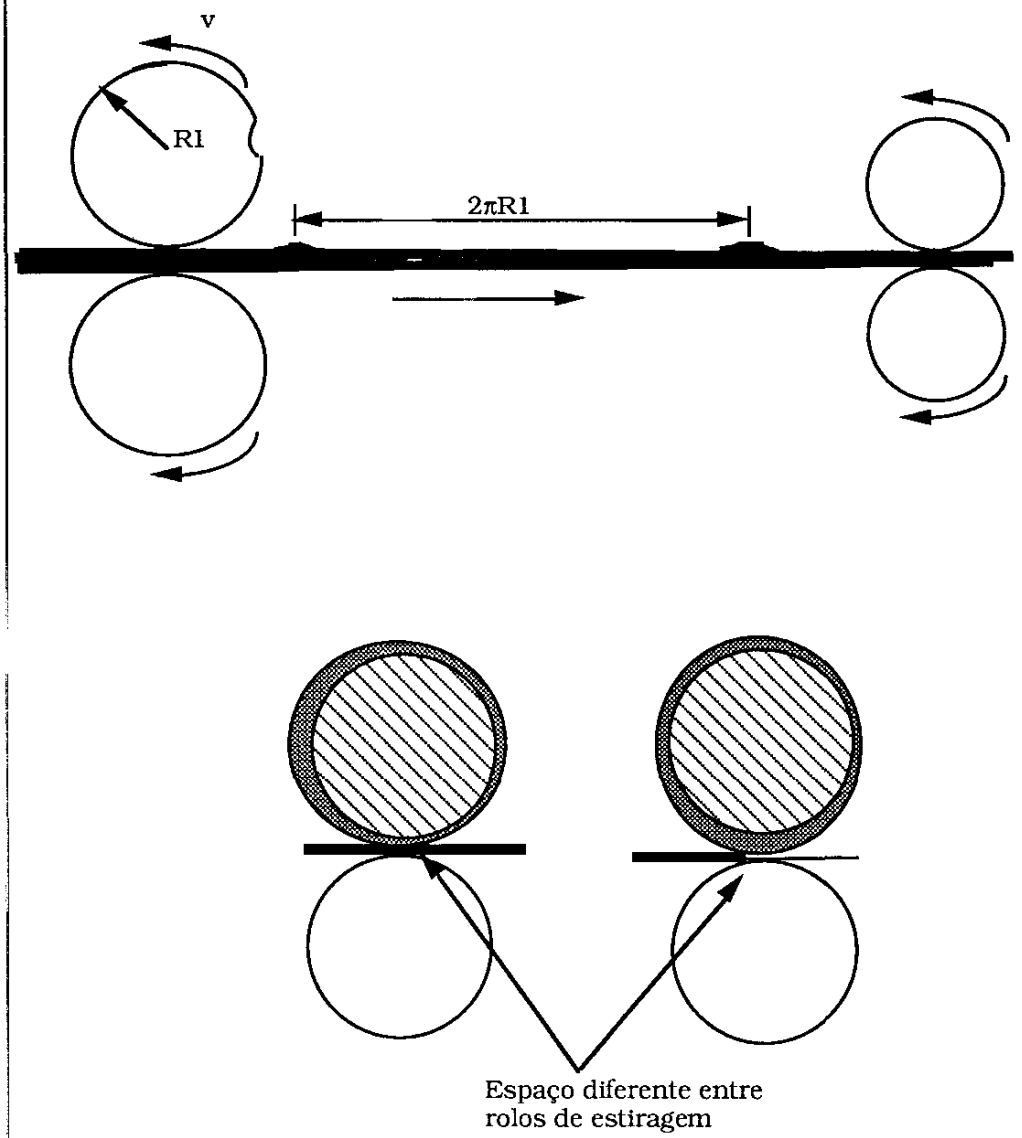


Figura 8. Imperfeições observáveis em determinadas fases do fabrico do fio têxtil. A estiragem do referido fio causa dois tipos de imperfeições devido à existência de deficiências nos rolos; o segundo tipo é facilmente detectável a partir de análise espectral.

Para a determinação do espectro da massa do fio, ou melhor, do espectro da imagem digital da massa por unidade de comprimento do fio têxtil, foram utilizados e testados diversos algoritmos **[34, 37, 39, 46, 50]**, estando o módulo final preparado para a determinação de espectros de componentes sinusoidais usando um algoritmo

baseado na Transformada Discreta de Fourier, e um espectro de componentes rectangulares que é implementado recorrendo a um algoritmo especialmente modificado da Transformada de Walsh.

Foram considerados estes dois métodos por apresentarem os resultados com informação facilmente relacionável com os dados em estudo; este pode ser caracterizável pela pretensão de conhecer as componentes espectrais sinusoidais ou rectangulares *mas de "duty cycle" próximo de 50%.*

Apesar disso foram implementados outros métodos que se revelaram ineficientes, quer devido ao longo tempo de processamento necessário para a obtenção de resultados, quer por não apresentarem resultados claros (caso da transformada de Hartley). Entre estes últimos destaca-se ainda a transformada de Haar, especialmente indicada para a decomposição de um sinal caracterizado por "bursts" (trem de impulsos próximos ocorrendo durante poucos instantes), razão pela qual foi abandonada; também um método desenvolvido no âmbito do presente trabalho, especialmente indicado para a determinação de frequência de impulsos, não é analisado nesta secção dado o carácter do "duty cycle" das componentes que pretendemos conhecer.

Para além destes algoritmos foram estudados e posteriormente implementadas diversos módulos de pré e pós-processamento da informação com vista à detecção de eventuais parâmetros típicos e minimizar os erros inerentes a todo o processo de aquisição de dados e cálculo do espectro.

Destacam-se, para o pré-processamento, a utilização de diversas janelas para a truncagem dos dados adquiridos, a possibilidade de recorrer à média (normal ou deslizante) e a possibilidade de derivar o sinal de entrada utilizando diversas metodologias.

Após o cálculo do espectro de um sinal é possível utilizar algoritmos de inversão do espectro, determinação do "CEPSTRUM", desconvolução, derivação, etc. Estes aspectos serão concretizados noutro capítulo, quando da explicação das soluções implementadas, muito embora seja apresentada uma análise teórica nesta parte.

4.1.1 Transformadas de Fourier e Walsh.

Para o cálculo do espectros de componentes sinusoidais, utilizou-se a Transformada Discreta de Fourier que tem os fundamentos na Transformada e na

Série de Fourier. Foi utilizado um algoritmo idêntico ao desenvolvido por Cooley e Tukey, com vista a efectuar o cálculo computacional, vulgarmente conhecido como FFT ("Fast Fourier Transform").

A transformada discreta de Fourier pode ser vista como uma evolução da transformada de Fourier. Algumas particularidades, associadas ao facto de ser efectuado um cálculo numérico, originam por vezes alguns erros (figura 9).

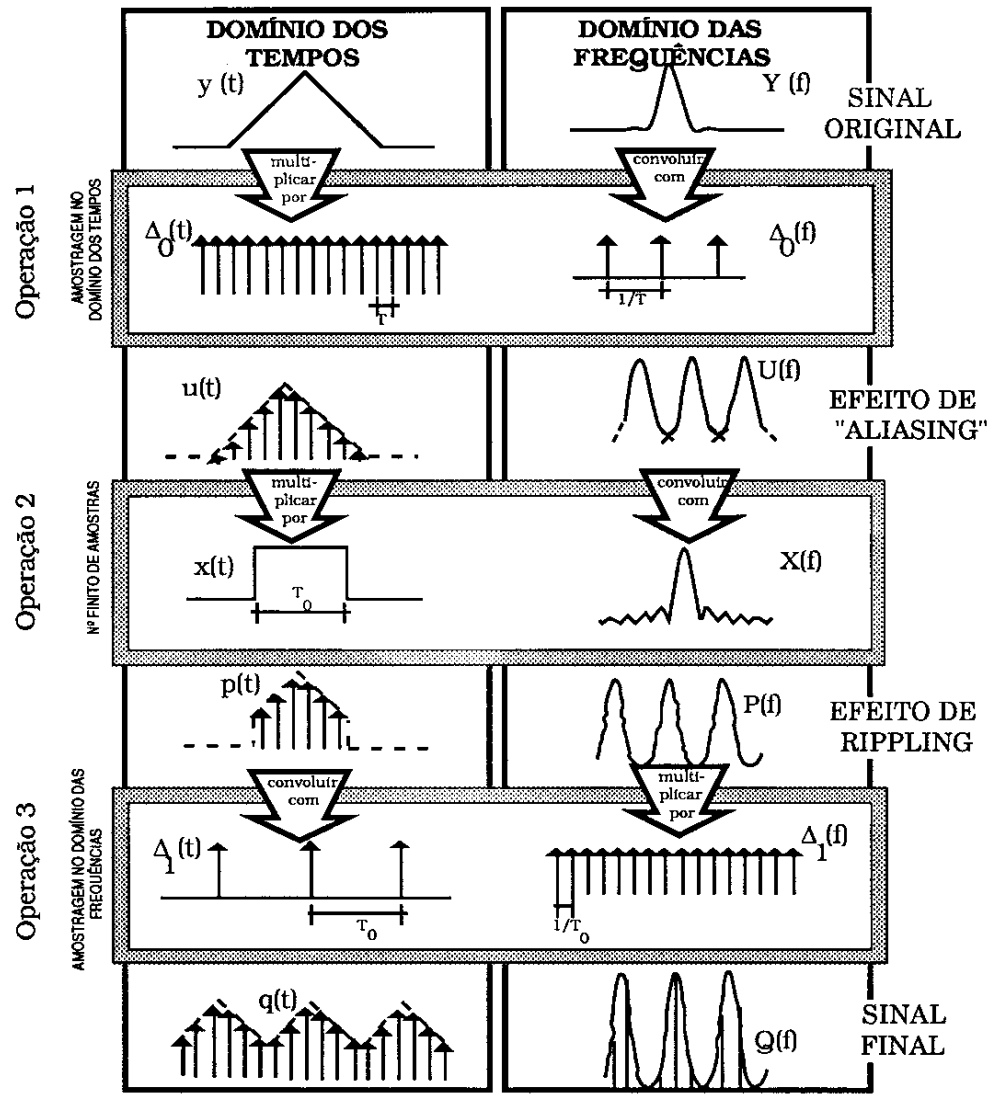


Figura 9. Esquema gráfico do cálculo da transformada discreta de Fourier e sua relação com a transformada de Fourier.

A primeira perturbação diz respeito ao efeito de "aliasing". Este é devido a ser utilizada uma frequência de amostragem que não cumpre completamente as especificações, ou seja, é inferior ou igual ao dobro da componente de maior frequência.

Esta situação, que existe para a maior parte dos espectros de sinais com duração limitada, não pode ser ultrapassada senão com uma filtragem adequada do sinal antes da aquisição. Esta filtragem, para ser bem sucedida e não implicar um aumento elevado da frequência de amostragem, deve ser analógica.

O segundo efeito mencionado é o de "rippling", advindo da necessidade de utilizar um número finito de amostras o que obriga à truncagem do sinal após a amostragem. Esta truncagem corresponde à multiplicação do sinal original, por um impulso rectangular.

A multiplicação referida origina um sinal cujo espectro é obtido pelo convolução do cálculo anterior com o espectro do referido impulso (do tipo $A\frac{sen(Kx)}{Kx}$).

Um processo de reduzir este efeito consegue-se com a utilização de pesos na aquisição de cada dado, ou seja efectuar a truncagem utilizando um sinal auxiliar, vulgarmente designado de janela; este sinal possui um espectro mais apropriado, que ao ser convoluído introduz menos erros. Este espectro apresenta uma relação muito maior entre o lobo principal (em torno do harmónico principal) e os restantes lobos, originando assim uma convolução no domínio dos espectros que acarreta menores perturbações no espectro resultante; estas janelas aproximam-se mais da solução ideal, que corresponderia a efectuar a convolução com um impulso unitário (Dirac). É vulgar utilizar janelas triangulares, de Hanning, Hamming, Blackman,etc.

Outro problema, igualmente associado à necessidade de truncagem do sinal após a amostragem e capaz de provocar discrepâncias acentuadas entre a transformada discreta e contínua de Fourier, é a desta truncagem de um sinal periódico, não ser efectuado a uma frequência múltipla do seu período; como o cálculo da FFT periodiza o sinal amostrado (figura 9), calcula-se o espectro de um sinal diferente do original.

Estas perturbações, chamados erros de "leakage", (figura 10), originam um sinal periódico que apresenta descontinuidades acentuadas.

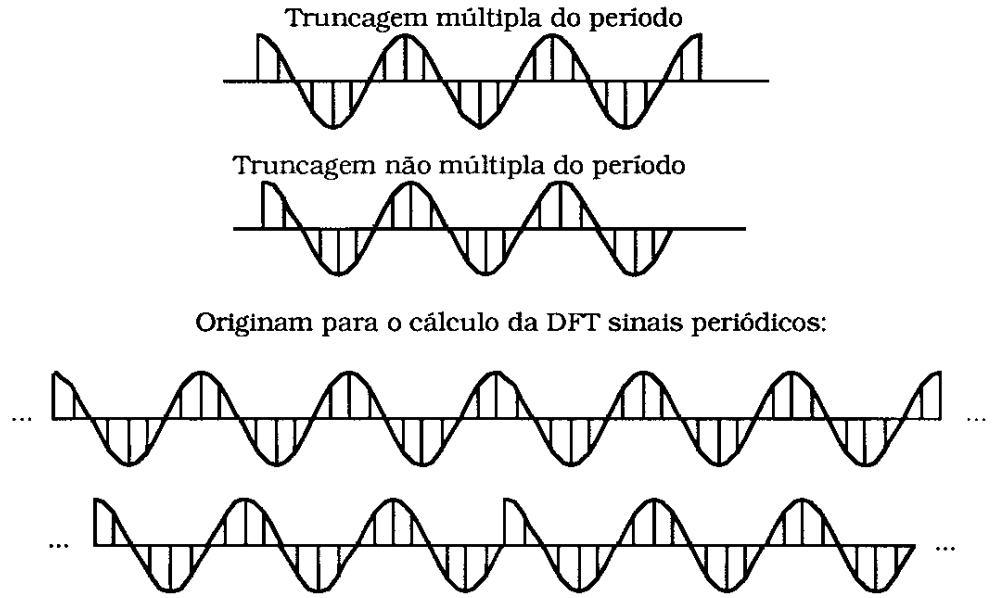


Figura 10. Exemplo dos sinais para os quais é calculada a DFT (ou FFT) quando o intervalo de amostragem é ou não múltiplo do período.

Diversas estratégias podem ser utilizadas para evitar, ou melhor, reduzir, este efeito pernicioso.

- Uma reside na utilização de janelas (intervalos de truncagem) que apresentem um espectro apropriado, ou seja, um lobo principal entre as frequências $-f_0$ e a f_0 em que $f_0 = \frac{1}{\text{Intervalo truncagem}}$; este lobo principal deve ter uma peso muito acentuado relativamente aos outros lobos quando os comparamos com a relação existente entre esses lobos na janela rectangular.

A técnica em causa apresenta-se com mais pormenor na figura 11, sendo exemplificado para o caso mais simples de ondas sinusoidais de frequência diferente e que foram truncadas com janelas iguais para cada sinal, mas que, num dos casos, não eram múltiplas do período.

Apresentam-se os espectros encontrados, quer para os sinais conforme amostrados, quer para esses sinais depois de aplicada uma janela que, no presente caso é, para além da rectangular, uma janela de Hanning ***h(t)***, definida como

$$h(t) = \frac{1}{2}\left(1 - \cos\left(\frac{2\pi n}{T}\right)\right)$$

sendo T o intervalo de truncagem.

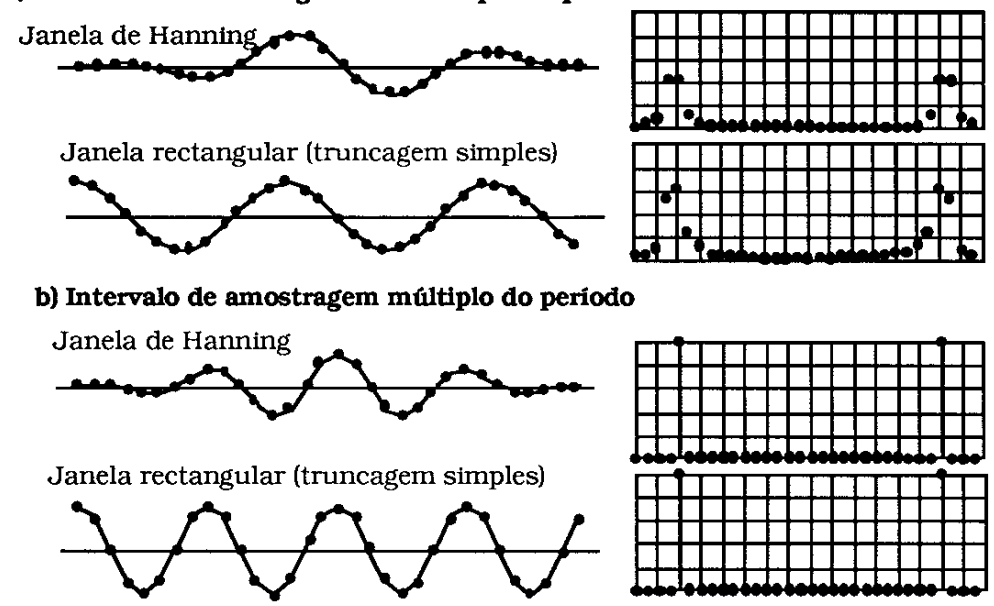


Figura 11. Efeito da utilização de uma janela (Hanning neste exemplo) para obviar os efeitos de "leakage", para um sinal múltiplo do período (b) e outro não múltiplo (a). De notar que em a) a utilização de uma janela de Hanning melhora consideravelmente os resultados.

- Outro dos métodos consiste na filtragem analógica do sinal a adquirir. Esta filtragem, para além de evitar, no que diz respeito a frequências elevadas, os problemas associados ao "aliasing", permite obviar alguns dos erros de "leakage". Esta filtragem só poderia contudo ter efeito se efectuada após a truncagem, tudo apontando então para a utilização de um filtro digital. No exemplo apresentado, sendo sempre um sinal sinusoidal, efectuar uma filtragem antes da aquisição não iria resolver qualquer problema. Pelo contrário, uma descontinuidade poderia ser eliminada por filtragem posterior. Apesar disso, esta solução limita superiormente a banda de frequências detectáveis, ou obriga a aumentar consideravelmente a frequência de amostragem, para além das dificuldades que levanta ao desenvolvimento.

- Por último, com uma apresentação dos resultados diferente consegue-se, embora reduzindo a informação disponível para cada risca constituinte do espectro, uma informação geral mais correcta. Este processo obtém-se apresentando não o valor de cada risca, mas a informação de **todas** as riscas numa banda de frequências, sendo assim mais provável obter uma informação de todas as componentes com frequências entre limites pré-definidos, mascarando-se assim os erros devido a uma truncagem que não seja múltipla do periodo. Este método foi também aplicado com bastante sucesso na representação do espectro da massa do fio têxtil.

Vejamos o algoritmo apresentado por Cooley e Tukey para a determinação do espectro (figura 12).

Este algoritmo é derivado da definição teórica da Transformada Discreta de Fourier (DFT). Consideremos a definição da DFT,

$$X(n) = \sum_{k=0}^{N-1} x_0(k)\, e^{-j\frac{2\pi\, kn}{N}} = \sum_{k=0}^{N-1} x_0(k)\, W^{kn}, \quad \left[\text{em que } W = e^{-j\frac{2\pi}{N}}\right]$$

que pode ser escrito de uma forma mais simplificada, como uma multiplicação de uma matriz por um vector, resultando um vector em que cada linha corresponde ao valor de X para o índice correspondente:

$$[X] = [W]\,[x_0]$$

Os elementos da matriz [W] são definidos como W^{n*k} (cada linha correspondente a um determinado n e cada coluna correspondente a um determinado k).

Vejamos como se apresentam estas matrizes para um valor de N=8:

$$\text{Matriz W} = \begin{vmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^8 & W^{10} & W^{12} & W^{14} \\ W^0 & W^3 & W^6 & W^9 & W^{12} & W^{15} & W^{18} & W^{21} \\ W^0 & W^4 & W^8 & W^{12} & W^{16} & W^{20} & W^{24} & W^{28} \\ W^0 & W^5 & W^{10} & W^{15} & W^{20} & W^{25} & W^{30} & W^{35} \\ W^0 & W^6 & W^{12} & W^{18} & W^{24} & W^{30} & W^{36} & W^{42} \\ W^0 & W^7 & W^{14} & W^{21} & W^{28} & W^{35} & W^{42} & W^{49} \end{vmatrix}$$

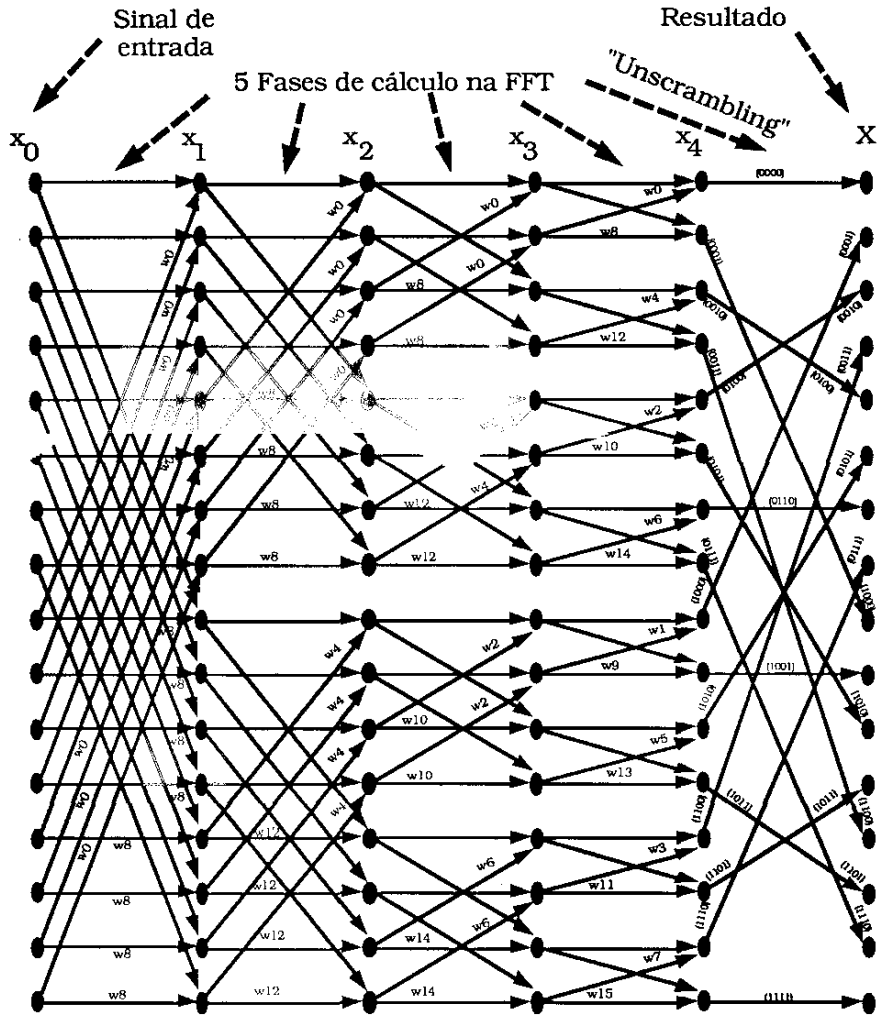


Figura 12. Algoritmo de cálculo da transformada rápida de Fourier, em que $W_k = e^{-j\frac{2\pi k}{N}}$. Neste exemplo consideraram-se 16 pontos (N=16), pelo que existe 4 fases de cálculo para determinar o resultado a que se acresce uma outra, completamente diferente, para o "unscrambling" dos resultados (ordenação pelos índices). Cada grafo indica uma operação de multiplicação seguida de uma soma (ou diferença) para determinar o resultado seguinte. De realçar que apenas necessitamos de uma zona de memória dado na determinação de um par de *nós* só intervir outro par e este não ser mais utilizado.

Dado as funções seno e cosseno (uma vez que $W = e^{-j\frac{2\pi}{N}}$) serem periódicas, reduz-se esta matriz a,

$$\text{Matriz } W = \begin{vmatrix} W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 & W^7 \\ W^0 & W^2 & W^4 & W^6 & W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^1 & W^4 & W^7 & W^2 & W^5 \\ W^0 & W^4 & W^0 & W^4 & W^0 & W^4 & W^0 & W^4 \\ W^0 & W^5 & W^2 & W^7 & W^4 & W^1 & W^6 & W^3 \\ W^0 & W^6 & W^4 & W^2 & W^0 & W^6 & W^4 & W^2 \\ W^0 & W^7 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{vmatrix}$$

com vista a facilitar as operações matemáticas a efectuar, esta matriz é decomposta em tantas quanto o número dos bits L utilizados no cálculo isto é, $L=\log_2 N$. Assim o vector resultado X é determinado da seguinte forma:

$$\begin{vmatrix} x(0) \\ x(4) \\ x(2) \\ x(6) \\ x(1) \\ x(5) \\ x(3) \\ x(7) \end{vmatrix} = \begin{vmatrix} 1 & W^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & W^4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & W^6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & W^5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & W^7 \end{vmatrix} \begin{vmatrix} 1 & 0 & W^0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W^0 & 0 & 0 & 0 & 0 \\ 1 & 0 & W^4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & W^4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & W^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W^2 \\ 0 & 0 & 0 & 0 & 1 & 0 & W^6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & W^6 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 & W^0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W^0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W^0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W^0 \\ 1 & 0 & 0 & 0 & W^4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & W^4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & W^4 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & W^4 \end{vmatrix} \begin{vmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \\ x_0(4) \\ x_0(5) \\ x_0(6) \\ x_0(7) \end{vmatrix}$$

Esta apresentação da matriz decomposta, não contraria o algoritmo inicial de cálculo da Transformada Discreta de Fourier (DFT), tendo sido utlizada esta forma com vista a facilitar as operações. Cada uma destas matrizes traduz uma das fases de cálculo, exemplificadas para N=16 na figura 12, que por ser muitíssimo mais rápido tomou o nome de "Fast Fourier Transform" (FFT). Esta rapidez está no entanto limitada dado ser necessário um nº de amostras $N(= 2^{\mathbf{K}})$ que cumpra a especificação de **K** ser um inteiro

Para o cálculo da transformada de Walsh, correspondendo à decomposição de um sinal em componentes do tipo rectangular, utilizou-se um algoritmo similar ao do cálculo da F.F.T.

A grande vantagem conseguida, reside no facto de, não existindo multiplicações nem divisões, ser consideravelmente acelerado o processamento da informação.

A transformada de Walsh é definida por,

$$X(n) = \frac{1}{N} \sum_{i=0}^{N-1} x(i)\, W(i,n) \quad \text{, em que} \quad \begin{cases} x(i) & \Leftrightarrow \text{vector com amostras} \\ X(n) & \Leftrightarrow \text{vector resultante.} \end{cases}$$

A construção dos valores W(i,n) é efectuada baseando-nos na definição,

$$W(i,n) = \begin{cases} sal\left[int\left(\frac{i+1}{2}\right),n\right] & \text{para i impar} \\ cal\left[int\left(\frac{i+1}{2}\right),n\right] & \text{para i par} \end{cases}$$

em que as funções **sal()** e **cal()** correspondem a funções rectangulares.

Estas funções são obtidas a partir de ondas sinusoidais (seno para as sal() e cosseno para as cal()) como se mostra na figura 13, para frequências definidas pelo factor **int** $\left(\frac{i+1}{2}\right)$, em que **i** corresponde ao número de ordem de cada amostra adquirida. Obtém-se esta informação por inspecção do sinal (positivo corresponde ao valor 1, negativo que corresponde ao valor 0) da função sinusoidal associada, exceptuando-se o caso do zero. Neste último caso considera-se o valor da última amostra determinada.

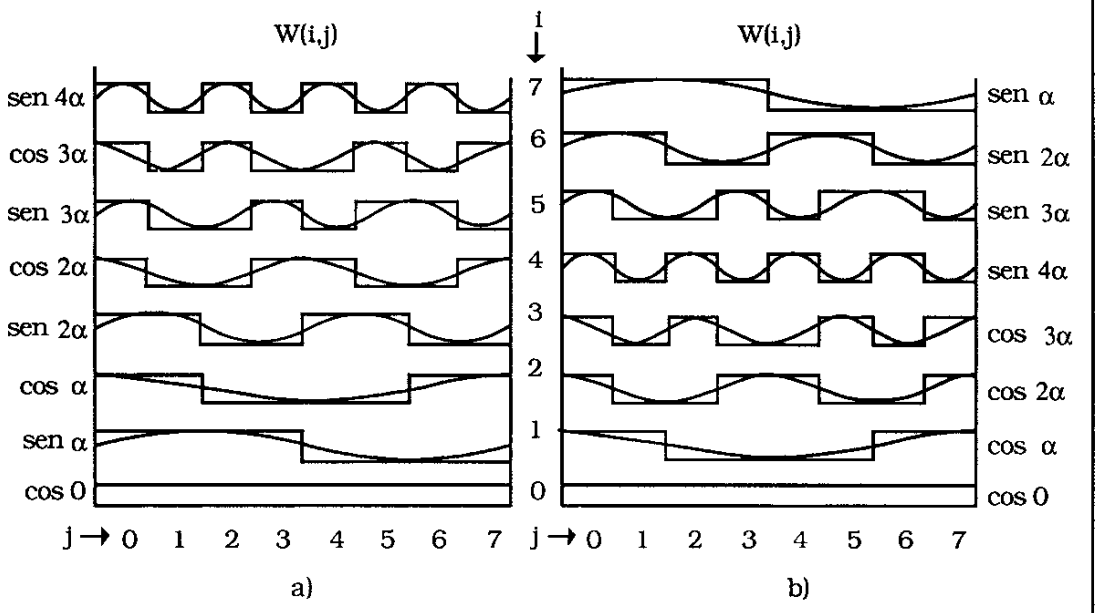


Figura 13. Processo de calcular a matriz em uso no cálculo da transformada de Walsh. À esquerda (a) o processo de cálculo tradicional e à direita (b) o utilizado no algoritmo implementado neste trabalho; note-se que a grande diferença reside no processo de ordenamento das funções.

Destas considerações resulta uma matriz de valores $W_{i,n}$ que para N=8 é:

$$\text{Matriz W} = \begin{vmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \end{vmatrix}$$

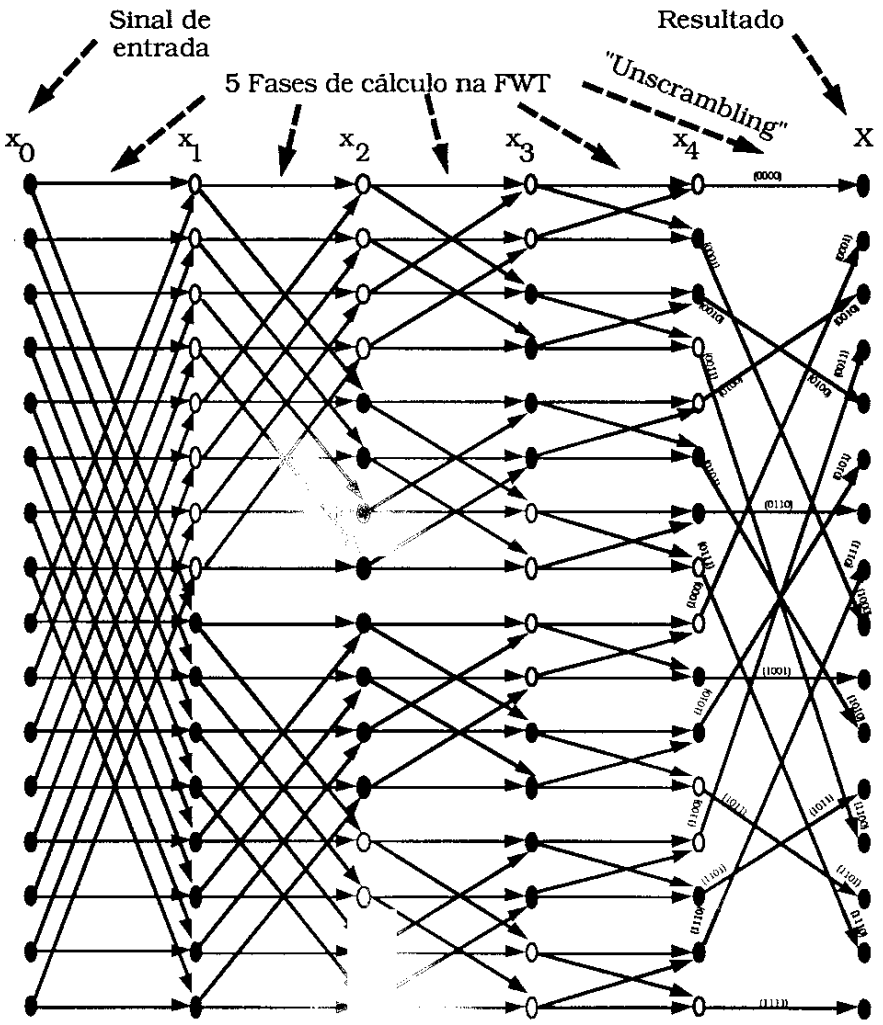
Esta matriz foi conseguida utilizando a organização tradicional da informação como exemplificado na figura 13 a). Com vista a simplificar as operações de detecção da frequência associada a cada valor detectado, reconstruiu-se esta matriz na forma:

$$\text{Matriz W} = \begin{vmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & -1 & -1 & -1 & -1 & +1 & +1 \\ +1 & -1 & -1 & +1 & +1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 & -1 & +1 & -1 & +1 \\ +1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\ +1 & -1 & -1 & +1 & -1 & +1 & +1 & -1 \\ +1 & +1 & -1 & -1 & +1 & +1 & -1 & -1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \end{vmatrix}$$

Esta técnica de ordenação (exemplificada na figura 13 b) permite aplicar um algoritmo muito eficiente. Embora semelhante ao desenvolvido para o cálculo da Transformada Rápida de Fourier (F.F.T), permite a utilização de uma metodologia em que não é necessário determinar o valor da função W(i,j), dado o respectivo valor variar alternadamente entre 1 e -1 para cada ciclo da fase de cálculo em questão (ver exemplo da figura 14, para o caso de N=16). Esta método conjugado com o facto de não se efectuarem multiplicações, acelera consideravelmente todo o processamento.

Tal como no caso da FFT, o algoritmo é decomposto em tantas fases de cálculo quanto o número de bits utilizados na definição do número de pontos adquiridos. Cada uma destas fases de cálculo corresponde à decomposição da matriz **W** noutras matrizes. Para o caso de N=8, existem três fases de cálculo que correspondem à multiplicação do vector de dados adquiridos pelas matrizes decompostas:

$$\begin{vmatrix} X(4) \\ X(4) \\ X(2) \\ X(6) \\ X(1) \\ X(5) \\ X(3) \\ X(7) \end{vmatrix} = \begin{vmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{vmatrix} \begin{vmatrix} x_0(0) \\ x_0(1) \\ x_0(2) \\ x_0(3) \\ x_0(4) \\ x_0(5) \\ x_0(6) \\ x_0(7) \end{vmatrix}$$



o - Nó soma (soma os valores dos nós anteriores)

● - Nó diferença (diferença dos valores do nó anterior de índice mais baixo do de índice mais alto)

Figura 14. Algoritmo de cálculo da transformada rápida de Walsh, em que $W_{i,j} = \pm 1$, correspondendo assim à soma ou diferença dos valores do cálculo anterior. Nos nós representados a **cheio** o valor presente é determinado pela diferença entre o valor de índice mais baixo subtraído do de índice mais elevado e os **ocos** representam somas. Neste exemplo com 16 pontos (N=16), existem 4 fases de cálculo a que se acresce uma outra, completamente diferente, para o "unscrambling" dos resultados (ordenação pelos índices).

Existe ainda uma fase de cálculo suplementar associada ao "unscrambling" dos resultados.

De realçar que não existe nenhuma multiplicação a efectuar em todo o algoritmo. Mesmo a utilização de uma soma ou uma diferença como operação entre dois nós não necessita do recurso a um factor de ±1.

Como todos os sinais que vão ser estudados são sinais reais, e no algoritmo não intervem nenhuma variável complexa, o resultado é também real.

O algoritmo apresenta assim uma estrutura do seguinte tipo:

```
n2 ← N/2
i ←1
enquanto i ≤ N2
  |temp ←x(k+n2)
  |x(k+n2) ← x(k)+temp
  |x(k) ← x(k)-temp
  |k ← k+1
  |i ← i+1
fim enquanto
l ← 2
enquanto l < nº de bits
  |k ← 0
  |n2 ← N/2
  |enquanto k < N-1
  |  |enquanto i ≤ N2
  |  |  |temp ← x(k+n2)
  |  |  |x(k+n2) ← x(k)+temp
  |  |  |x(k) ← x(k)-temp
  |  |  |k ← k+1
  |  |  |i ← i+1
  |  |fim enquanto
  |  |k ← k+n2
  |  |enquanto i ≤ N2
  |  |  |temp ← -x(k+n2)
  |  |  |x(k+n2) ← x(k)+temp
  |  |  |x(k) ← x(k)-temp
  |  |  |k ← k+1
  |  |  |i ← i+1
  |  |fim enquanto
  |  |k ← k+n2
  |fim enquanto
  |l ← l+1
fim enquanto
```

Convém notar que a única multiplicação existente (na realidade uma divisão por 2), e que é efectuada tantas vezes quanto o número de bits, pode ser implementada através de um "shift" para a direita, instrução essa bastante rápida.

Salienta-se ainda que é possível acelerar o processo com o recurso a dois vectores em memória. Esta solução não é de desprezar, uma vez que não existe o cálculo de nenhum valor complexo, pelo que comparativamente ao cálculo da Transformada de Fourier, existe ainda esta vantagem. Acrescente-se que no cálculo da FFT, metade da informação era redundante, funcionando apenas como controlo dado o espectro ter uma simetria em torno do ponto N/2.

Neste caso, é possível minimizar o tempo de processamento, utilizando o seguinte algoritmo:

```
f ← 0
g ← 1
n2 ← N/2
i ← 1
enquanto i ≤ N2
  |x(k,g) ← x(k,f)-x(k+n2,f)
  |x(k+n2,g) ← x(k,f)+x(k+n2,f)
  |k ← k+1
  |i ← i+1
fim enquanto
l ← 2
enquanto l < nº de bits
  |k ← 0
  |n2 ← N/2
  |enquanto k < N-1
  |   |temp ← f
  |   |f ← g
  |   |g ← temp
  |   |enquanto i ≤ N2
  |   |   |x(k,g) ← x(k,f)-x(k+n2,f)
  |   |   |x(k+n2,g) ← x(k,f)+x(k+n2,f)
  |   |   |k ← k+1
  |   |   |i ← i+1
  |   |fim enquanto
  |   |k ← k+n2
  |   |enquanto i ≤ N2
  |   |   |x(k,g) ← x(k)+x(k+n2,f)
  |   |   |x(k+n2,g) ← x(k)-x(k+n2,f)
  |   |   |k ← k+1
  |   |   |i ← i+1
  |   |fim enquanto
  |   |k ← k+n2
  |fim enquanto
  |l ← l+1
fim enquanto
```

Por fim convém salientar a possibilidade de, através de algumas operações, ser possível obter o valor de uma risca do espectro sinusoidal a partir dos valores da transformada de Walsh **[42]**; este processo é puramente matemático mas permite, nos casos em que se pretende conhecer apenas alguns dos valores do espectro, acelerar este cálculo relativamente ao uso da F.F.T. tradicional.

Este algoritmo, com o **reordenamento da tabela** apresentado, constituiu um significativo aumento de eficácia relativamente às transformadas tradicionais, não acarretando contudo nenhuma modificação drástica da transformada de Walsh; refira-se que, apesar da pesquisa bibliográfica efectuada, ***não se encontraram referências a esta metodologia*** em qualquer trabalho anterior.

Como termo de comparação entre a aplicação da transformada de Fourier e Walsh, refira-se que este algoritmo, na primeira versão apresentada, é três vezes mais rápido que o mais rápido dos da FFT.

A aplicação deste algoritmo a amostras representativas da espessura do fio têxtil apresentou por seu lado resultados bastante satisfatórios, similares aos da FFT; estes são por sua vez idênticos aos obtidos em equipamentos industriais. Foi ainda possível caracterizar picos em certas frequências permitindo identificar os comprimentos típicos das fibras que constituem um certo fio têxtil.

Estas comparações dos resultados serão ainda analisadas num capítulo posterior.

4.1.2 Processamento adicional na determinação de espectros.

Para a análise das características de um fio têxtil é necessário converter a informação obtida sob a forma de componentes de certa frequência para comprimentos de onda.

Para o cálculo destes valores consideram-se as seguintes relações:

$$\text{Intervalo de amostragem} = T_a = \frac{\text{Velocidade dos rolos (em m/mn)} * 1000}{\text{Comprimento do sensor (em mm)} * 60} = \frac{\Omega * 1000}{L * 60} = 4.8\,\text{ms} \quad \text{para } \Omega = 100 \text{ m/mn e } L = 8\text{mm}$$

O comprimento de onda, λ, de uma dada componente espectral com frequência f_d Hz é assim $\lambda = L * \frac{f_a}{\text{frequência detectada}} = L * \frac{f_a}{f_d}$.

Para qualquer uma das transformadas apresentadas existem diversos programas complementares que permitem um pré ou pós-processamento conseguindo-se assim obter resultados particularmente eficazes em alguns casos. Alguns destes aplicam-se particularmente aos sinais após a aquisição englobando-se assim na classe de algoritmos de pré-processamento.

Alguns dos métodos de pré-processamento implementados têm uma aplicação garantida na análise de sinais diferentes dos tratados no âmbito do presente trabalho de regularimetria têxtil; a sua generalidade levou à sua inclusão nesta na aplicação (módulo) de determinação de espectros. A decisão de desenvolver todos estes programas baseou-se em dois vectores essenciais:

- testar diversos métodos de preparar os dados adquiridos para o cálculo espectral de forma a fornecer resultados próximos dos pretendidos;
- colocar ao dispor do utilizador ferramentas de processamento de sinal tradicionais, flexibilizando o sistema para diversas aplicações;

Foram assim estudados os seguintes métodos de pré-processamento:

- Retirar o valor médio (ou componente contínua do sinal). Esta operação, que também deve poder ser executada após o cálculo do espectro final, interessa para o caso em que se usam janelas de truncagem não rectangulares; estas introduzem por vezes uma componente contínua, ou modificam a existente, sendo então necessário eliminá-la ou compensá-la.

- Efectuar o "fold" da informação adquirida; neste processo, após a recolha de N/2 amostras, duplicam-se os dados numa forma de "espelho", originando uma função par. Este algoritmo corresponde à utilização de uma janela diferente do habitual, mas que garante que o sinal adquirido é uma função par, logo com um espectro real e par.

- Derivar o sinal adquirido, função particularmente útil para a detecção de ocorrência de irregularidades; apesar desta importância é por vezes vantajoso efectuar a derivação de um sinal adquirido com vista à detecção, 'a priori', de transições bruscas.

A utilização deste algoritmo para a detecção de irregularidades é discutida com mais pormenor na secção respectiva. O sinal resultante da derivação pode ser utilizado para o cálculo de espectro do fio.

- Comparar o sinal adquirido com valores limite. Este algoritmo transforma o sinal adquirido num sinal limitado superior e inferiormente, resultado da comparação do primeiro sinal com dois valores de limiar ("threshold"). Com este pré-processamento é possível normalizar um sinal adquirido para valores esperados numa dada gama (idêntico a fazer passar o sinal por um "Schmitt Trigger").

É ainda possível com este algoritmo efectuar a eliminação de ruído analógico acrescentado a um sinal digital. Uma das aplicações mais importantes deste tipo de pré-processamento foi na do cálculo do espectro de sinais resultantes da modulação por largura de impulso (PWM), em que é sabido que a informação é digital (um bit em cada instante), mas em que o conhecimento dos harmónicos componentes é da maior importância.

Também foram estudados diversos algoritmos de pós-processamento, que têm por função a preparação dos resultados com vista, quer a uma detecção rápida de imperfeições ou características fora do comum, quer com o fito de uma apresentação dos resultados mais eficaz.

De realçar ainda a possibilidade destes algoritmos serem usados de forma encadeada, ou seja, ser possível utilizar um certo pós-processamento e, sob os resultados deste, utilizar de novo o mesmo ou outro destes métodos.

De entre os módulos testados foram considerados como mais eficazes, e fazendo parte do módulo final, os seguintes algoritmos:

- inversão do espectro;
- desconvolução;
- derivar os resultados;
- modificar riscas;
- determinar "cepstro";

- A inversão de espectros permite determinar o inverso de um espectro previamente calculado. Por este processo consegue-se realçar certas 'bossas' no espectro de frequências, mantendo no entanto uma relação entre as diversas riscas.

- A desconvolução é um algoritmo que permite a obtenção do espectro original de um sinal depois de convoluido com o espectro de um impulso rectangular de 8 amostras.

A necessidade da existência deste algoritmo advém da a aquisição de amostras com um intervalo superior à imposta pelo comprimento do sensor implicar uma convolução do espectro real do fio com o espectro de um sinal correspondente a um impulso rectangular (ver ponto de detecção de irregularidades no capítulo 5).

- A derivação do espectro calculado é outro dos algoritmos disponíveis. A sua inclusão foi decidida dentro da perspectiva de generalização do algoritmo, não tendo apresentado resultados interessantes para o estudo do regularímetro digital.

- A modificação de riscas do espectro resultante da aplicação de qualquer uma das transformadas rápidas expostas, permite efectuar uma filtragem condicionada de algumas ou todas as riscas.

Este algoritmo consiste em filtrar (ou amplificar) algumas das riscas que cumprem uma determinada especificação, isto é, permite modificar as riscas que são inferiores e/ou superiores a determinados limites.

Esta possibilidade apresenta enormes vantagens nos casos em que, não existindo um algoritmo claro de selecção, é possível ao operador seleccionar partes do espectro para eliminar ou intensificar relativamente a outras.

É ainda possível por recurso a este algoritmo, efectuar a eliminação de riscas correspondentes a ruído e que vulgarmente surgem por indução, como por exemplo as componentes de 50 Hz e 100 Hz.

- Por último, está prevista a utilização do algoritmo de cálculo do "CEPSTRO" (palavra originária de ESPEC + TRO) geralmente designada de "CEPSTRUM" (em inglês) **[69, 46]**.

Este algoritmo tem por finalidade distinguir 2 espectros mascarados num resultante; este último corresponde ao espectro de um sinal obtido pela convolução no domínio dos tempos do sinal de que se pretende conhecer o espectro, com um outro sinal de que se conhece minimamente o espectro; o espectro resultante deriva da multiplicação dos dois espectros.

Este algoritmo foi desenvolvido, e é principalmente utilizado, no reconhecimento de padrões/formas ("PATTERN RECOGNITION"), especialmente no processamento de sinais sonoros:

- Identificação de veículos por análise dos sons por eles produzidos,
- Reconhecimento de fonemas,
- etc.

Baseia-se na logaritmização da transformada de Fourier do sinal (separando assim os dois espectros multiplicados), o que permitirá distinguir dois espectros com maior facilidade; posteriormente procede-se ao cálculo da transformada de Fourier inversa deste resultado.

Resumindo,

$$\text{CEPESTRO}[w(t)] \leftrightarrow \textbf{FFT}^{-1}\left[\log\left((W(f))^2\right)\right]$$

A principal dificuldade reside no cálculo do logaritmo que é uma função de uma variável complexa apresentando um resultado no mesmo tipo.

Esta função, ***o logaritmo complexo***, define-se, para uma variável x tal que $x = a + jb$, como:

$$\log_e(x) = \text{Real} + j\,\text{Imag} = \log_e(a + jb) = \log_e\left[\left(\sqrt{a^2 + b^2}\right) e^{-j \arctan\left(\frac{b}{a}\right)}\right]$$

$$\log_e(a + jb) = \log_e\left[\left(\sqrt{a^2 + b^2}\right)\right] - j \arctan\left(\frac{b}{a}\right);$$

Existe no entanto uma dificuldade associada a este cálculo, e que reside na determinação da parte 'imaginária' do logaritmo complexo. Este problema deve-se à função ARCTAN() fornecer os resultados entre π e $-\pi$ apresentando assim descontinuidades que não deveriam existir (figura 15).

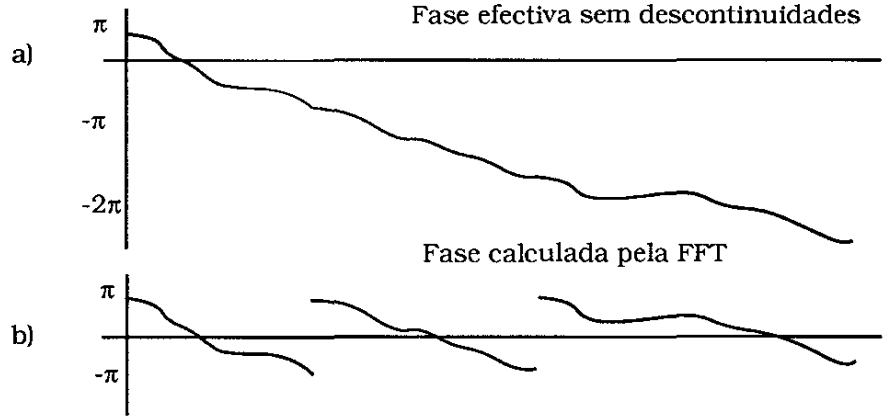


Figura 15. Cálculo da parte imaginária do logaritmo complexo = arc tan$\left(\frac{b}{a}\right)$, sendo **b** e **a** respectivamente a parte imaginária e real resultante da aplicação do algoritmo da FFT. Sendo uma função **garantidamente** contínua, é necessário efectuar uma normalização.

Este problema pode ser corrigido somando (ou subtraindo) múltiplos de 2π.

A decisão de quando somar (subtrair) ou não esse valor foi tomado com base na premissa de que os valores da função arctan() não apresentam descontinuidades e são valores obtidos com intervalos suficientemente próximos; esta opção, inválida para alguns casos dado a 'amostragem' da função arctan() não ser contínua, só é possível efectuar impondo um limite à variação máxima admissível entre cada 'amostra'.

4.2 Cálculo da Autocorrelação.

O resultado da correlação de dois sinais, efectuada no domínio dos tempos, tem uma transformada correspondendo a um espectro igual ao produto dos espectros dos sinais iniciais. Uma das funções mais utilizadas deste algoritmo, a autocorrelação, corresponde à correlação do sinal com ele próprio. Naturalmente a transformada de Fourier deste processamento é um espectro cujo módulo é igual ao quadrado do módulo do espectro do sinal original.

Este facto traduz-se na prática no aumento da relação sinal/ruído do resultado, se entendermos que o sinal, ao contrário do ruído, é constituído pelos harmónicos fundamentais. Efectivamente, ao calcular o quadrado de duas grandezas em que uma é superior à outra, no final a primeira será ainda maior, relativamente à outra; este facto traduz-se, no domínio dos tempos, na visualização mais acentuada das componentes sinusoidais de maior peso.

Este processo foi largamente utilizado para a detecção de qual a frequência fundamental de sinais analógicos (é ainda bastante utilizado no cálculo estatístico), antes de ser eficaz recorrer ao processamento digital para a determinação de espectros; em geral, por simples inspecção do sinal temporal resultante da autocorrelação, é possível 'extrair' (deduzir) qual a componente sinusoidal de maior peso. Este processo de análise, um tanto ou quanto empírico, era em geral re-utilizado para a detecção de outras frequências, efectuando a autocorrelação do conjunto de amostras após lhe subtrair a(s) componente(s) já detectadas.

Um dos principais óbices à utilização deste método em cálculo numérico, prende-se com o número de amostras possíveis de utilizar no cálculo ser limitado **[33, 40]**.

Devido a este problema, a autocorrelação de um sinal amostrado pode ser definida de duas formas *circular* ou *não circular*.

Na *circular*, consideramos que o número de amostras correspondem a um período **T** do sinal pelo que, procedemos à autocorrelação do sinal adquirido com um sinal periódico em que cada período é igual ao sinal originalmente adquirido.

Na *não circular*, consideramos que as amostras recolhidas correspondem a um impulso de pequena duração, sendo todas as restantes amostras nulas.

A utilização de um ou outro método advém das necessidades da aplicação; utilizando o mesmo critério do da determinação do espectro em que considerámos que adquirimos informação de um período, fomos levados a optar pela autocorrelação circular.

4.3 Detecção rápida de irregularidades (erros).

A existência de uma frequência constante (ou quase) das irregularidades do fio têxtil, é uma das causas do aparecimento de imperfeições graves no tecido, traduzíveis nomeadamente pelo aparecimento de riscas visíveis.

A existência de uma frequência fixa aponta por outro lado, dado o processo de fabrico do fio têxtil (ver figura 16), para uma deficiência típica associado por exemplo a uma falha de determinado rolo utilizado na estiragem.

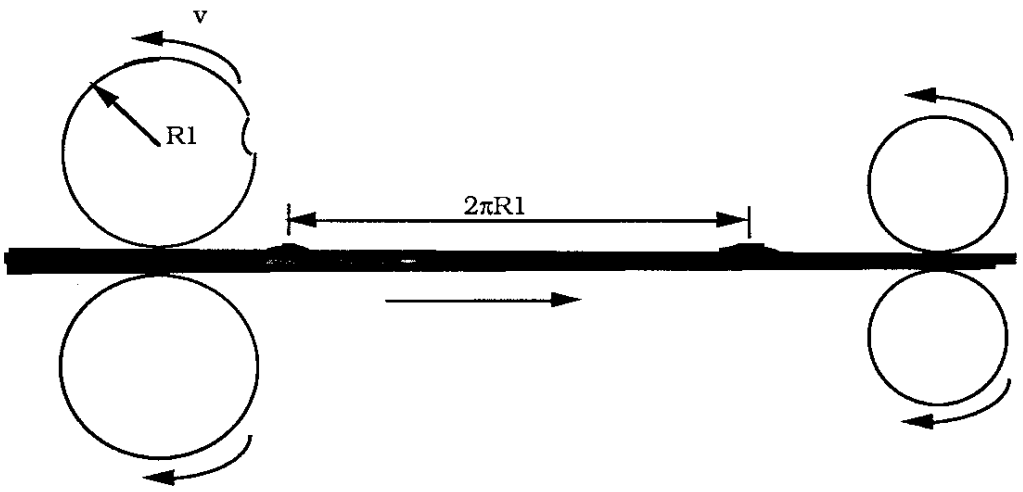


Figura 16. Imperfeições observáveis em determinadas fases do fabrico do fio têxtil. A estiragem do referido fio causa imperfeições que sendo periódicas (como neste exemplo), levantam graves problemas.

Com base nestas considerações torna-se imperioso detectar esta frequência típica com vista quer à correcção de eventuais erros quer para a manutenção (ou melhoria) dos níveis da qualidade da produção. Essa informação terá ainda uma grande importância como processo de efectuar uma manutenção preventiva de todo o equipamento, na medida em que as irregularidades detectadas advêm com bastante frequência de perturbações na produção.

4.3.1 Caracterização dos erros.

Consideremos um sinal x(t), exemplificado na figura 17. Independentemente deste sinal ser captado directamente do sensor, ou ser um sinal a que já foi efectuado um certo processamento, pretendemos detectar a ocorrência de eventuais erros para a partir deles inferir, a causa que os provocou.

Vamos apenas debruçar-nos sobre os erros definidos pela ultrapassagem de um certo limiar, correspondendo por exemplo à ocorrência de *pontos grossos* ou *Neps*; para determinar a ocorrência de *pontos finos*, o processo é análogo, com a diferença de se detectar quando é que o sinal é inferior a um certo limite.

4.3.2 Determinação da frequência dos erros.

Como já vimos, para a determinação da causa que provocou o 'erro' é de máxima importância o estudo da frequência típica com que este ocorre. Com vista a que este dado seja fàcilmente calculado é conveniente proceder a algumas transformações.

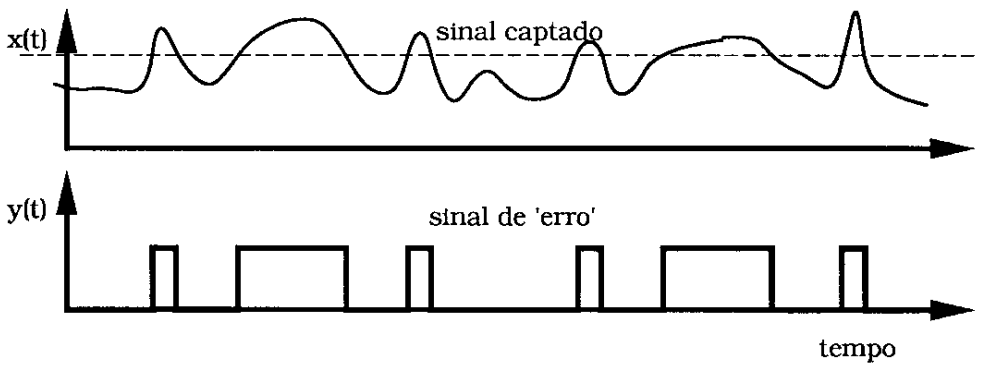


Figura 17. Sinal x(t) e respectivo sinal de erro y(t).

Para o efeito vamos considerar um sinal binário com o valor 1 quando é ultrapassado o limiar de detecção e 0 nos restantes casos. Aplicando esta transformação, obteremos sinais como o exemplificado na figura 17 (y(t)), diferente conforme o limiar de detecção (nível de detecção do erro que no presente trabalho corresponde ao valor de uma irregularidade do tipo NEP, Ponto grosso ou Ponto fino) .

A análise da frequência com que ocorrem estas perturbações é indispensável dado que apenas as periódicas ou quasi-periódicas podem ser atribuídos a uma causa sistemática; no entanto, erros extemporâneos, isto é, sem periodicidade clara, não podem ser atribuídos a uma causa desse tipo, razão pela qual não devem confundir a informação respeitante aos erros periódicos. Convém no entanto notar, que com esta afirmação não se pretende desprezar os erros extemporâneos; a sua contabilização é importante embora venha a provocar imperfeições de cariz completamente distinto do que se pretende analisar neste ponto.

Muito embora a detecção de frequência de erros seja (ou pareça) intuitiva, não existe um método claro, sistemático e rápido de detecção da *eventual* frequência de ocorrência dos erros, *muito menos uma operação ou transformação matemática (como em* ***[44, 45]****)*; estas imperfeições podem, além do mais, estar completamente mascarados por outras que tenham ocorrido no mesmo segmento de fio têxtil.

Na figura 18 estão ilustradas 3 das múltiplas hipóteses de erros periódicos, que poderiam provocar o sinal y(t) da figura 17. De notar que este sinal pode ser obtido pela disjunção lógica dos sinais de cada hipótese apresentadas (realça-se que o sinal y(t) é binário). Assim o sinal y(t) pode obter-se,

a) $y(t) = a1(t) \vee a2(t)$,
b) $y(t) = b1(t) \vee b2(t) \vee b3(t)$,
c) $y(t) = c1(t) \vee c2(t) \vee c3(t) \vee c4(t) \vee c5(t)$, etc.

Convém realçar que pode sempre ser considerada uma hipótese obtida pela disjunção lógica de qualquer um dos sinais que constituem uma hipótese já analisada, desde que o resultado da disjunção seja o sinal y(t), e excluindo repetições. Podemos assim definir o sinal y(t), por exemplo,

$y(t) = a1(t) \vee a2(t) \vee b2(t) \vee c4(t)$, ou
$y(t) = a1(t) \vee a2(t) \vee b1(t) \vee b2(t) \vee c1(t) \vee c2(t)$, etc.

observando, contudo, que alguns dos erros seriam provocados no mesmo instante.

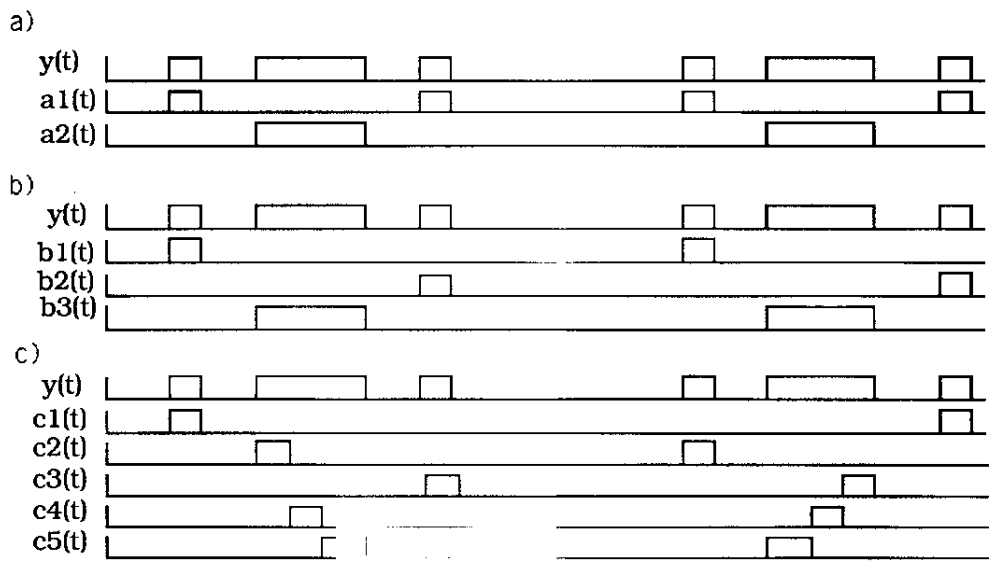


Figura 18. Possíveis combinações dos erros do sinal x(t).

Acrescente-se ainda que, sendo o sinal analógico, não podemos considerar que uma irregularidade, por muito curta que seja a sua duração, corresponda apenas a **um** erro e não a **um conjunto** de erros contiguos.

Numa situação limite, podemos afirmar que as combinações dos erros poderiam ser feitas com trens de impulsos tipo Dirac de tal forma que a disjunção desses trens de impulsos originassem o sinal inicial, existindo, neste caso, uma frequência de erros *sem limitação prática imediata*; dado no presente sistema existir o limite físico do tamanho do sensor, podemos sempre considerar uma frequência máxima para as imperfeições. Este aspecto tem ainda relação com o facto de o processamento a efectuar ser do tipo numérico e não contínuo.

Com base em todas as considerações efectuadas - muito especialmente a que diz respeito à possibilidade de simultaneidade das irregularidades (ou erros) -, somos levados a concluir que não será fácil definir uma metodologia de **medição** das frequências de irregularidades; poderemos, na melhor das hipóteses, apresentar um conjunto de frequências que podem constituir, com uma certa probabilidade, o sinal de erro original.

4.3.3 Desenvolvimento do algoritmo de determinação da frequência.

Nesta secção estudar-se-á o método de determinar a frequência com que surgem irregularidades, procurando utilizar uma abordagem rápida que não sobrecarregue os elementos de aquisição nem de processamento; para o conseguir interessa que este processamento seja efectuado o mais próximo possível da estação de aquisição (*provavelmente nesta*) e seja simples não exigindo elevados recursos no seu processamento (*em especial espera-se não obrigar a grandes e complicadas operações matemáticas*), pelo que o método implementado será rápido e com poucas operações.

As perturbações caracterizam-se por aparecerem durante um curto intervalo de tempo (quando comparado com o intervalo até ao aparecimento de nova irregularidade) pelo que se mostrará da impossibilidade de utilizar as transformadas tradicionais (*Fourier, Walsh, Hartley, Haar*), razão que levou ao desenvolvimento de um algoritmo próprio.

4.3.3.1 Discretização.

Considerámos até este momento que os sinais a analisar são contínuos no tempo, o que não corresponde à verdade, uma vez que o sinal x(t) é adquirido por um conversor analógico-digital em instantes discretos de tempo, e o processamento posterior será feito por um microcomputador. Assim apresenta-se na figura 19, a título de exemplo, o sinal x(t) depois de amostrado com uma frequência de amostragem $f_a = \frac{1}{T}$, transformando-se assim no sinal x(nT), bem como o sinal de erro y(nT).

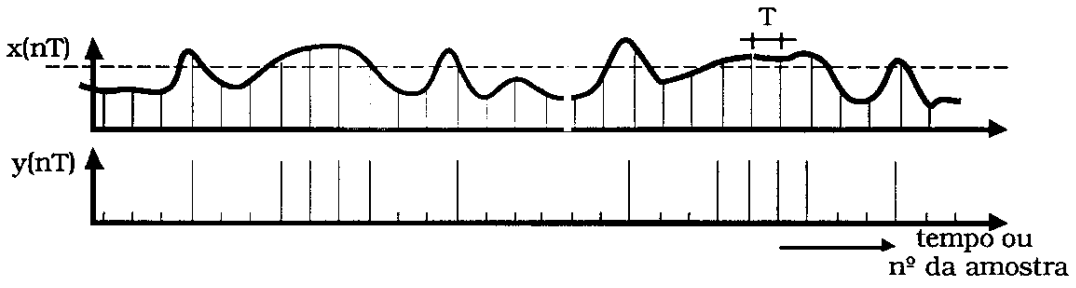


Figura 19. Sinal x(nT) resultante da amostragem do sinal x(t) e respectivo sinal digital de erro y(nT).

Apresenta-se igualmente na figura 20, um conjunto de sinais digitais que poderiam constituir o sinal de erro y(nT).

Também neste caso podemos afirmar que existem inúmeras hipóteses de combinação de possíveis frequências de erros. Não recorrendo a dados complementares considerá-las-emos equiprováveis; a opção por uma, obriga ao recurso a mais informação, obtida quer por experimentação quer tendo por base os resultados anteriores.

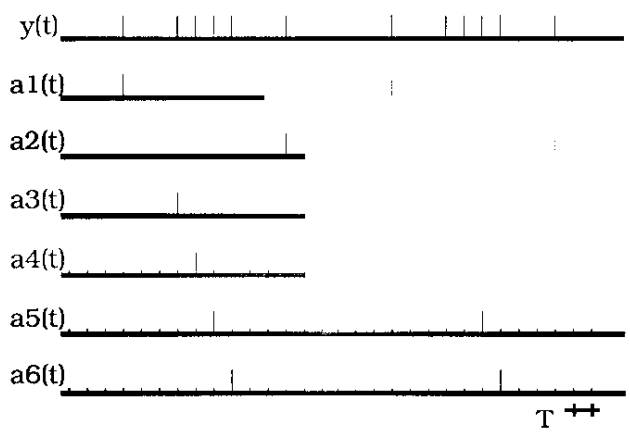


Figura 20. Uma possível combinação de sinais de erro (irregularidades) equiprováveis para o sinal de erro digital y(nT).

4.3.3.2 Estudo da aplicabilidade das transformadas: Fourier, Walsh e Haar.

Com vista a garantir tempos de processamento aceitáveis é conveniente garantir que o algoritmo que pretendemos construir (designado abreviadamente por **F**),seja simples, rápido e eficiente; o resultado deve ser uma indicação da possibilidade ou não de uma função $a_i(nT)$ compor o sinal de 'erro' y(nT), podendo ser esquematizado como:

$$y(nT) \quad \Leftrightarrow \quad \mathbf{F}[\, y(nT)\,]$$

$$y(nT) = \bigvee_{i=0}^{N-1} (\, \Re(i).a_i(nT)) \quad \Leftrightarrow \quad \mathbf{F}[\, y(nT)\,] = \Re(i) \qquad (\textit{IV.1})$$

em que
- $\vee$ é a função disjunção de um conjunto de funções $a_i(nT)$
- $\Re(k)$ é o coeficiente, 0 ou 1, associado a cada uma das funções $a_i(nT)$
- k é o nº de ordem identificador de cada função $a_i(nT)$

As necessidades de rapidez e eficiência apontam no sentido de, sendo possível, a '*transformação*' ser efectuada no sistema de aquisição, obrigando portanto a um estudo cuidado do algoritmo a implementar. Interessar-nos-ia, com vista a minorar o tempo de desenvolvimento, utilizar algoritmos (ou parte) de certas transformadas (Fourier, Walsh, Haar, etc.), pelo que analisaremos a aplicabilidade de algumas.

A transformada de Fourier, que permite obter informação sobre as componentes sinusoidais de um sinal, não nos convém utilizar, pois:

1 - A informação que se pretende obter não está minimamente relacionada com um sinal sinusoidal mas sim com um sinal binário (ou há erro ou não há).
2 - As operações matemáticas necessárias para os cálculos associados à determinação da transformada de Fourier de um sinal são dispendiosos, dado, para além do tempo de processamento próprio, obrigarem a cálculo complementar.

As considerações aqui presentes sobre as desvantagens do uso de uma decomposição em ondas sinusoidais tipo Fourier generalizaram-se a outras transformações similares, de entre as quais se destaca a transformada de **Hartley [27, 34, 37, 48]**.

Será assim aconselhável recorrer a uma transformação que permita decompor y(nT) em sinais do tipo 'rectangular'; isto apontaria para o recurso a uma transformada do tipo Walsh **[42, 43]**.

O grande inconveniente reside no facto de, nesta transformada, o sinal ser decomposto em ondas rectangulares com "duty cycle" próximo dos 50%.

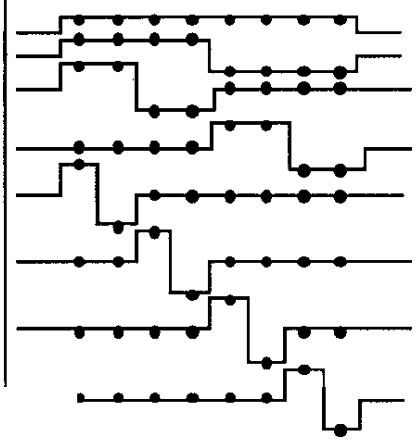
A informação resultante de uma análise deste tipo não permitiria conclusões claras, uma vez que os erros são, presumivelmente, de pequena duração quando comparados com o período de tempo sem erros.

Assim, esta transformada só poderá ser considerada, como alternativa à transformada de Fourier, para estudar outro tipo de erros e/ou características, como referido noutro capítulo.

A transformada de Haar permite resultados já consideravelmente próximos dos pretendidos, sendo especialmente aplicável para a detecção de "bursts"; considera-se um "burst" como um sinal de "elevada amplitude com largura de banda estreita e de curta duração com pequenos tempos de subida e descida" **[39]**.

Esta transformada é bastante útil na análise de erros extemporâneos, permitindo conhecer as componentes de um impulso deste tipo, (figura 21); no entanto não fornece qualquer informação quanto às componentes em frequência de um sinal

periódico (embora permita analisar as componentes de frequência do "burst"), mesmo que cada período seja bastante longo se compararmos o tempo em que existe sinal, com o tempo em que não existe (característica do sinal y(t) ou y(nT)).



Matriz correspondente:

$$\begin{vmatrix} +1 & +1 & +1 & +1 & +1 & +1 & +1 & +1 \\ +1 & +1 & +1 & +1 & -1 & -1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sqrt{2} & \sqrt{2} & -\sqrt{2} & -\sqrt{2} \\ +2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & +2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & +2 & -2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & +2 & -2 \end{vmatrix}$$

Figura 21. Componentes de um sinal usando a transformada de Haar (para N=8).

Em conclusão, podemos afirmar que as transformadas de **Walsh, Fourier, e Haar** não são soluções viáveis, já que, nenhuma delas permite decompor directamente o sinal de erro y(nT), num conjunto de sinais a_i(nT), ou pelo menos num conjunto de sinais a partir dos quais seja possível sem muito e complicado processamento, inferir esses coeficientes.

Como consequência de todas estas conclusões, verificamos ser necessário proceder à implementação da transformação seguindo um algoritmo próprio.

4.3.3.3 Um novo algoritmo para determinar a frequência de impulsos (DFI).

Para conseguir obter a informação pretendida foi desenvolvido no âmbito deste trabalho, um algoritmo inteiramente novo; este método, permite, a partir de um sinal de erro, obter um conjunto de valores (*a que* ***apenas por semelhança*** *com outras transformadas podemos chamar módulo*) que fornece uma informação, do tipo probabilístico, da possibilidade de uma determinada componente fazer parte do sinal de erro original.

Este resultado não é peremptório devido a diversas imprecisões e inconstâncias dos intervenientes no processo de fabrico (por exemplo a velocidade dos rolos).

Para além das flutuações associadas ao processo de fabrico, existe ainda a possibilidade de o sinal y(n) ser definido por um sem número de combinações de erros (a que acresce o facto de as irregularidades poderem ser constituídas por diversos erros contíguos); assim só será garantido fornecer como resultado um valor probabilístico sobre cada uma das possíveis frequências; este, na melhor das hipóteses, poder-nos-á garantir 'ser possível' que uma determinada frequência de irregularidades entre na constituição do sinal y(n).

Para estudar este algoritmo consideremos um exemplo, em que o sinal x(nT) é constituído por 16 pontos e em que T, por questões de operacionalidade, corresponde à unidade de tempo pelo que x(n)=x(nT) e y(n) = y(nT).

Suponhamos então que depois da detecção dos valores acima de um certo limite obtinhamos um sinal y(n) (figura 22),

$$[Y] = y(n) = \underset{\substack{y(0)\uparrow \\ \uparrow y(15)}}{1\,1\,0\,0\,1\,0\,1\,0\,1\,0\,0\,1\,1\,0\,0\,0}$$

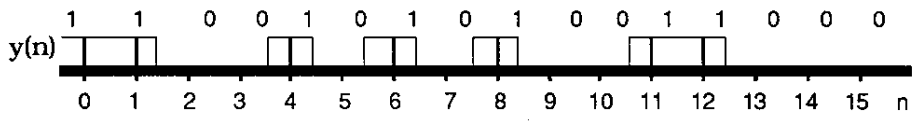


Figura 22. Sinal de erro y(n) considerado no exemplo exposto no texto.

Esta transformada baseia-se no cálculo dos coeficientes $\Re(i)$ de $a_i(n)$ indicados em (***IV.1***); estes coeficientes $\Re(i)$ são valores numéricos que representam a amplitude de cada componente $a_i(n)$, devendo ser 0 ou 1. Quando o valor for 0 concluímos da ***impossibilidade*** de esse $a_i()$ fazer parte do sinal de erro e, ao invés, quando o resultado é 1, concluímos da ***possibilidade*** de fazer parte. Como veremos adiante, este valor pode ser considerado não apenas 0 ou 1 mas qualquer um no intervalo [0,1], fornecendo assim um valor de uma probabilidade menos peremptória que o *sim/não*.

O resultado da transformação indicada de seguida (***IV.2***), é o vector $[A]=\{a_0, a_1, \ldots, a_{15}\}$, calculado, a partir do sinal de erro y() da figura 22, como,

$$[A] = N\ ([R]\ [Y]), \quad \Leftrightarrow \quad a_n = N \sum_{j=1}^{N} (r_{n,j}\ \ y_j). \qquad (IV.2)$$

Este vector [A] dar-nos-á uma informação sobre a possibilidade do sinal y(n) ser composto pelos sinais exemplificados na figura 20, similar á apresentada pelos coeficientes $\Re(i)$; nestes a indicação resultava do valor ser **0** ou **1**, enquanto nos coeficientes a_n podemos extrair essa informação a partir do resultado ser ou não inferior a N; este aspecto será esclarecido com mais detalhe nos parágrafos seguintes.

Para efectuar esta transformação foi necessário considerar uma matriz [R] com a forma indicada na figura 23, em que o índice, **n**, dos coeficientes $a_n \in \left[1, \sum_{k=N_4}^{N_2} k \right]$, sendo $N_2 = \frac{N}{2}$ e N_4 o maior inteiro acima de $\frac{N}{4}$; estes limites correspondem aos maior (N_2) e menor (N_4) períodos dos eventuais sinais constituintes do sinal y(n), surgindo estes valores limite uma vez que:

- limitámos os possíveis valores do nº de amostras a uma potência de 2;
- só nos interessa considerar 'sinais' com pelo menos **dois** impulsos (razão de $\frac{N}{2}$);
- sinais de erro com período inferior a N_4 obtêm-se pela composição de sinais com período superior já considerados noutras componentes;

Os valores obtidos por este processamento, a_n, não apresentam uma relação imediata com os valores pretendidos de $\Re(i)$; para os obter é necessário comparar os resultados com **N**: se for **igual a N** então o sinal $a_n(i)$ ***pode constituir*** o sinal de erro y(); caso contrário não pode. Esta comparação tornar-se-á mais evidente ao analisarmos a composição do sinal de erro da figura 22, como faremos, após o estudo de ***IV.2***.

Em primeiro lugar é necessário definamos as características da matriz $[R] = \{ r_{i,j} \}$, que multiplicada pelo vector [Y] permitirá obter o resultado [A].

Esta matriz [R] é construída da seguinte forma:

- As linhas correspondem às amostras de eventuais sinais de erro, com uma certa frequência e esfasamento. O objectivo do algoritmo será, no final, obter um grau de relação entre esse sinal de erro e o sinal y(n).

- Para cada frequência, são consideradas **todas as hipóteses de esfasamento** com vista a ser efectuada a correlação entre o sinal de erro y(n) e o eventual sinal representado numa linha para uma certa frequência; este aspecto corresponde na

prática a efectuar a correlação do sinal original com um que possua impulsos com uma certa frequência.

- A amplitude considerada, está inversamente relacionada com o número de amostras em cada um desses sinais, para que a possibilidade de um determinado sinal fazer parte de y(n) poder ser verificada de igual modo para cada um deles.

$$
\begin{bmatrix}
\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} \\
\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\
0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\
0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\
\frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\
0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\
\frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} \\
0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 \\
0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} & 0
\end{bmatrix}
$$

Figura 23. Matriz [R] utilizada para o cálculo dos coeficientes a(n).

Para o exemplo em estudo obteríamos um vector coluna [A] com os seguintes valor,

$$[A] = \{a_n\} \quad \Leftrightarrow \quad y(n) = \{1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0\}$$

Período = 8 amostras	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
	16	8	0	8	16	0	8	0

Período = 7 amostras	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}
	5.3	10.7	0	0	16	8	8

Período = 6 amostras	a_{15}	a_{16}	a_{17}	a_{18}	a_{19}	a_{20}
	6	5.3	5.3	0	8	0

Período = 5 amostras	a_{21}	a_{22}	a_{23}	a_{24}	a_{25}
	4	16	5.3	5.3	5.3

Podemos então afirmar que, dado que os elementos $\{a_0, a_4, a_{12}, a_{15}, a_{22}\}$ serem iguais a N, os sinais representados pelas linhas {0, 4, 12, 15 e 22} da matriz [R] **podem** fazer parte do sinal y(n), enquanto os restantes coeficientes, de ordem diferente, **não podem** pertencer; ou seja $\Re(0)$, $\Re(4)$, $\Re(12)$, $\Re(15)$ e $\Re(22)$ têm o valor 1 enquanto todas os outros têm valor 0.

Caso pretendessemos calcular um valor percentual, que apontasse para o grau de correlação existente entre os sinais definidos pelas linhas da matriz [R] e o sinal de erro efectuaríamos a transformação:

$$a_n = \frac{a_n}{N} * 100\,\% = 100\,\% \sum_{j=1}^{N} r_{n,j}\ y_j$$

Neste caso, se considerarmos os valores de $\Re() = a_n\ /100\%$, teremos estes coeficientes com valores entre 0 e 1, conforme referido.

Na figura 24 estão representadas todas as componentes possíveis para a constituição do sinal y(n). De realçar que foram excluídas todas as combinações de sinais em que os erros aparecessem apenas uma vez, sendo apenas considerados aqueles em que aparecem pelo menos 2 erros em cada linha da matriz. Este facto origina que nenhum factor da matriz R seja, alguma vez, superior a $\frac{1}{2}$.

Com a informação existente, os erros que ocorram apenas uma vez no intervalo de tempo em que se adquiriram as amostras, têm de ser considerados como extemporâneos, não apresentando qualquer periodicidade típica.

Realça-se, mais uma vez que os erros podem ser provocados por qualquer conjunto de sinais $a_i(n)$ desde que da sua disjunção resulte o sinal y(n).

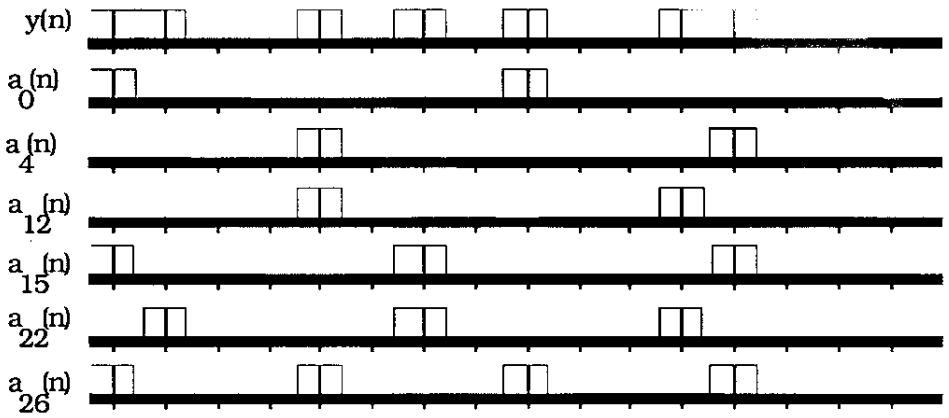


Figura 24. Combinação possível dos erros do exemplo considerado (ver figura 22); o sinal y(n) pode obter-se pela disjunção de todos ou alguns destes $a_i(n)$ que possuem amplitude, definida pelo coeficiente $\Re(i)$, = 1.

Na figura 25 estão representados os sinais r(n) representados na matriz R, isto é os sinais correspondentes a linhas da referida matriz, que originaram os elementos do vector [A] com valores numéricos iguais a N e, consequentemente os coeficientes $\Re(i)$ unitários.

Como se pode verificar, estes sinais correspondem a sinais já representados na figura 24, desde que o seu período seja superior a um valor definido como o maior inteiro acima de $\frac{N}{4}$.

Convém notar que o factor N da expressão (***IV.2***) é facultativo. Efectivamente, o que se obtém no vector [A] é uma informação do número de impulsos acima de 0 que fazem parte quer do sinal y(n), quer do sinal r(n). Esse número é multiplicado por um factor ($\frac{1}{2}$ para r(0), $\frac{1}{3}$ para r(8), $\frac{1}{4}$ para r(21), etc.), com vista a uma normalização, e que está directamente relacionada com o número de impulsos existentes em cada sinal.

O resultado referido está compreendido entre 0 e 1. A multiplicação por N visa, apenas, facilitar as operações matemáticas a executar num processador, pois, provavelmente estas serão operações inteiras.

Se compararmos as linhas da matriz [R], para as quais os a_i's têm o valor N (figura 25), com os valores esperados para a decomposição de y(n) (figura 24), concluimos que, são equivalentes se desprezarmos o sinal $a_{26}(n)$. Da análise dos sinais da figura 24, verificamos não haver informação suplementar por

considerarmos o sinal $a_{26}(n)$, uma vez que este é obtido pela disjunção do sinal $a_0(n)$ e $a_4(n)$; este sinal, ao poder ser decomposto em dois outros, não nos permite esclarecer com precisão suficiente qual a origem da perturbação, uma vez que esta tanto podia ser devida às originadas pelos sinais $a_0(n)$ e $a_4(n)$ como pelo $a_{26}(n)$. Para fazer esta distinção necessitariamos recorrer a dados externos.

É por esta razão que apenas consideraremos sinais, representados na matriz [R], com valores de período até ao limite N_4.

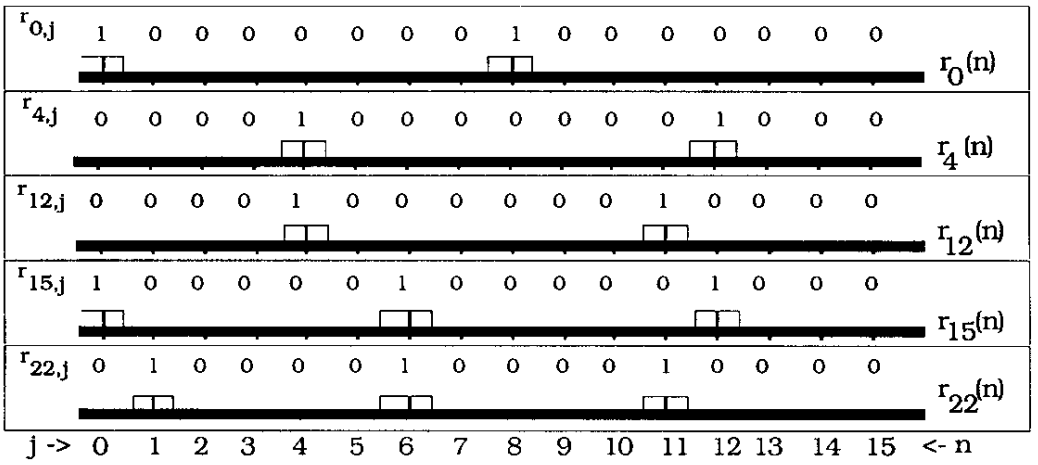


Figura 25. Representação de $r_{i,j}$ que produzem a_n = N, 16 para o exemplo em causa; de notar que estes elementos do matriz [R] têm uma correspondência clara com sinais temporais.

Generalizando, podemos afirmar que basta ter em conta sinais $a_i(n)$ em que, a frequência dos erros correspondente a um período até $\frac{N}{4}$, dado não podermos obter informação fiável relativamente a uma frequência de erros múltipla de uma já considerada. Assim, só seriam considerados, para o exemplo da figura 24, os sinais com períodos entre 8 e 5 amostras (inclusivé), correspondendo na prática a sinais com períodos entre metade do valor N e o valor de período mais baixo que não corresponda a uma frequência múltipla de outra já analisada. Veremos posteriormente que nem sempre isso poderá convir.

O exemplo aqui apresentado apenas considerou uma conjunto de 16 amostras do sinal original x(n). É possível generalizar este algoritmo para um número diferente de

amostras dimensionando convenientemente a matriz [R], o que leva a um aumento dos custos, no que diz respeito a memória e tempo de processamento.

Apresentam-se, no quadro 3, os valores para a dimensão da matriz [R], conforme o número de amostras consideradas.

Número de pontos	Número de linhas	Número de colunas
16	26	16
32	100	32
64	392	64
128	1552	128
256	6176	256
...	...	...
N	$\sum_{i=int\frac{N}{4}+1}^{\frac{N}{2}} i = \frac{\frac{N}{2}+int\frac{N}{4}+1}{2}\left(\frac{N}{2}-int\frac{N}{4}\right)$ se N for múltiplo de 2 e ≥ 4 então $= \frac{N(3N+4)}{32}$	N

Quadro 3. Número de linhas da matriz [R], bem como do número de coeficientes a_n para diferentes valores do número de pontos N.

Resumindo, esta transformação corresponde a fazer a multiplicação de cada amostra do sinal de erro adquirido, com o sinal $r_i(n)$ (correspondendo, na prática a efectuar a conjunção ou o "e" lógico destes sinais), em que n corresponde na matriz [R] à variável j. Se o sinal resultante for igual ao sinal $r_i(n)$, então estaremos perante um possível componente do sinal. A transformação inversa corresponde à disjunção lógica de todos os sinais $r_i(n)$ para os quais $a_i(n)$ tem um valor igual a N.

Veremos na secção seguinte alguns passos que permitirão o incremento da velocidade de processamento, evitando algumas operações aritméticas mais demoradas.

4.3.3.4 Simplificação e generalização do algoritmo DFI.

Com vista a simplificar o processamento a efectuar para conseguir obter o vector $[A]=\{a_i(n)\}$, vamos considerar a matriz equivalente a N * [R] factorizada em duas matrizes [R1] e [R2] (ver figuras 26 e 27).

1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0
1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0

Figura 26. Matriz [R1] utilizada para o cálculo dos coeficientes a_n, quando N=16.

Utilizando este método a operação (***IV.2***), correspondente a uma transformação, passa a ser definida pela expressão:

$$[A] = ([R1] * [Y]) * [R2]$$

$$\left[\frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{2}\ \frac{16}{2}\ \frac{16}{4}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{3}\ \frac{16}{3}\right]$$

Figura 27. Matriz linha [R2].

O resultado da multiplicação da matriz [R1] pelo vector correspondente ao sinal de erro origina o vector [A], apenas depois de multiplicarmos cada elemento do vector resultante por um valor do vector [R2]; cada um destes obtém-se dividindo o valor N por um divisor inteiro, que pode ser 2,3 ou 4, conforme o número de impulsos que existe ao longo da linha correspondente da matriz [R] (ou [R1]).

Com vista a maior rapidez, interessa garantir que os tempos de processamento despendidos quer na multiplicação quer na divisão possam ser minimizados.

Se for previamente estabelecida qual a ordem dos divisores, limitaremos consideravelmente o tempo de processamento, uma vez que a divisão por 2 ou por 4 corresponde ao "shift" (deslocamento) do valor binário do vector inicialmente calculado.

Além disso, ao limitarmos o número de pontos do cálculo, isto é o número de amostras, a um múltiplo de 2, reduziremos o tempo de processamento ainda mais, pois também podemos executar a multiplicação por "shifts".

De destacar que o factor multiplicativo em causa só pode tomar valores compreendidos entre $\frac{N}{2}$ e $\frac{N}{4}$, em que como já foi referido o divisor representa o número de impulsos considerados; esta imposição é facilmente demonstrável com base nas seguintes considerações:

- Dado um conjunto N de amostras, a frequência mais baixa de uma componente será a que tem o período $\frac{N}{2}$;

- Não 'vale a pena' analisar componentes com frequência múltiplas de outras já consideradas; por exemplo, dado N=256, não interessa estudar a componente com período 51 ($\text{int}\left(\frac{256}{5}\right) = 51$), que pode ser conseguido com duas componentes (esfasadas) de período 51*2 = 102 tendo esta componente já sido estudada.

- Com base nestas premissas pode-se concluir que o referido factor terá de pertencer ao intervalo $\left[\frac{N}{2}, \frac{N}{2} * \frac{1}{2}\right]$;

Pela própria definição da matriz [R], facilmente se verifica não poder existir mais de 4 impulsos em cada linha (relembrar a limitação do maior inteiro superior a $\frac{N}{4}$). Sendo assim, apenas teremos de considerar a matriz [R2] como constituídos por valores $\frac{N}{2}$, $\frac{N}{3}$ ou $\frac{N}{4}$. Com isto podemos garantir a existência de um algoritmo de multiplicação e divisão específico que seja bastante rápido, entrando já em linha de conta com os possíveis divisores.

Para uma análise mais cuidada, vejamos o quadro 4, que representa os valores possíveis dos divisores, para diferentes valores de N=16.

Número de pontos	Valor do vector	Linhas em que é usado o valor do vector
16	16/2	1, 2, 3, 4, 5, 6, 7, 8,11, 12, 13, 14, 15, 20, 21
	16/3	9,10,16, 17, 18, 19, 23, 24, 25, 26
	16/4	22

Quadro 4. Valores dos divisores dos elementos da matriz [R2].

É de realçar que, dada a pretensão de executar este algoritmo na placa de aquisição de dados, e com vista a garantir tempos de processamento aceitáveis, todas as divisões serão inteiras. Isto implica que, quando o valor do vector [R2] for $\frac{N}{3}$, e sendo $N = 2^{\alpha}$ (caso mais aconselhável), incorremos num erro que é necessário prever.

Sendo α e β números inteiros verifica-se a relação,

$$2^{\alpha} - (-1)^{\alpha} = 3 * \beta,$$

pelo que podemos estabelecer a condição,

$$N = 2^{\alpha} \Rightarrow \frac{N - (-1)^{\alpha}}{3}, \text{ ou seja } \frac{N \pm 1}{3} \text{ é um valor inteiro.}$$

Assim, o maior erro resultante da divisão inteira de N por 3 acontecerá para o menor valor de N, sendo 6.25% para N=16 e 0.39% para N=256. A expressão que permite calcular o erro è:

$$\Delta R = Erro = \left| \frac{N - (-1)^{\alpha}}{N} * 100\ \% \right|$$

Estes erros, para um número de amostras não exageradamente pequeno, mostra que podemos adoptar o critério de efectuar divisões inteiras, sem risco de afectar, em grande escala, a precisão do cálculo.

Convém esclarecer que, na prática, não será efectuada qualquer divisão, mas somente multiplicação, por um factor $\frac{N}{2}$, $\frac{N \pm 1}{3}$ ou $\frac{N}{4}$.

Estas multiplicações correspondem a "shifts" para $\frac{N}{2}$ e $\frac{N}{4}$ sendo só uma operação mais demorada para o outro caso.

4.3.3.5 Conclusões.

O algoritmo proposto para a implementação da transformação pretendida, resume-se assim em,

1 - Processar o sinal x(n) de forma a encontrar y(n).

2 - Com as amostras deste sinal construir o vector [Y].

3 - Proceder à multiplicação deste sinal pela matrix N*[R].

4 - Escolher as frequências que podem constituir y(n).

Os pontos 1, 2 e 3 foram já analisados em pormenor. Relativamente ao último ponto convém referir que a escolha pode ser efectuada quer na placa de aquisição, quer no microcomputador central, já que as frequências dos erros poderem ser definidas de uma forma não exacta. Efectivamente é provável que existam algumas flutuações nas frequências dos erros, ou que em dado instante, um erro tenha sido mascarado.

Neste caso será conveniente considerar como frequências constitutivas do sinal y(n) não todas aquelas a que corresponde um valor N no vector [A], mas sim todas as que tenham pelo menos uma certa percentagem desse valor (ex. 80% de N). Neste último caso, é de prever uma interacção entre operador e sistema, pelo que será mais conveniente ter a informação residente no microcomputador central.

De facto esta consideração está de acordo com a adopção para o coeficiente $\Re()$ de valores entre 0 e 1 como exposto anteriormente; esta toma ainda maior significado se tomarmos em conta os aspectos, já mencionados, de flutuações nas características dos intervenientes do processo de fabrico.

Uma objecção grave a esta possibilidade reside no facto de apenas estarmos a considerar como possíveis componentes do sinal y(n), sinais com 2, 3 ou 4 impulsos. Nesse caso, a detecção só será possível para valores N, 75%N, 66%N, 50%N, 33%N, 25%N.

Esta limitação é contudo fàcilmente ultrapassável, para erros originados por frequências mais elevadas, se considerarmos como componentes do sinal y(n) sinais com período entre $\frac{N}{2}$ e 1, ao invés de entre $\frac{N}{2}$ e $\frac{N}{4}$. O aumento de tempo de processamento e memória necessária, podem ser diminutos comparados com as vantagens decorrentes da utilização deste grupo de valores para o período.

Este tipo de resultado pode também ser conseguido a partir da análise dos valores obtidos para períodos múltiplos dos considerados inicialmente.

No quadro 5, apresentam-se os dimensões da matriz [R] para diversos valores de N, necessárias a este processamento (b) e com a versão anterior (a).

Número de pontos	Número de linhas (b)	Número de linhas (a)	Número de de colunas
16	36	26	16
32	136	100	32
64	528	392	64
128	2080	1552	128
256	8256	6176	256
...	...	...	...
N	$\sum_{i=1}^{N} i$	$\sum_{i=int(\frac{N}{4})+1}^{\frac{N}{2}} i$	N

Quadro 5. Linhas das matrizes [R] ou [R1], necessárias para calcular o vector [A], prevendo qualquer tipo de componentes (b) ou limitando-as (a).

Até esta altura considerou-se que N corresponde ao número de amostras recolhidas.

Com vista a uma análise de segmentos com número de amostras mais elevado e, *dado ser previsível um erro ocorrer apenas durante o tempo de aquisição duma amostra*, podemos transformar o período de amostragem, conseguindo desta forma diminuir ***drasticamente*** o nº de amostras do cálculo; se considerarmos que esse novo período de amostragem (T1) é igual T^*W em que W é o número de amostras contidas ***no menor segmento de informação com erros*** e T o período de amostragem inicial, transforma-se o sinal y(n) num sinal com menos amostras para o mesmo período de tempo.

Esta metodologia corresponde a limitar o nº de amostras a processar, fazendo com que o menor intervalo de tempo permanentemente em erro corresponda, no novo sinal, a apenas uma amostra (no máximo 2 para garantir que nenhuma outra irregularidade fique por detectar); com esta metodologia libertamos quantidades consideráveis de memória.

Apresenta-se na figura 28 como seria o sinal y(n) e o sinal resultante $y_1(n)$ de um processamento como o indicado.

Falta por último referir que o tempo de processamento necessário, seguindo este algoritmo, depende directamente do número de erros que existem, uma vez que o processamento, pode ser feito em grande parte na rotina de aquisição de dados.

Quando existe a detecção de um erro é actuado um contador associado a cada linha da matriz [R].

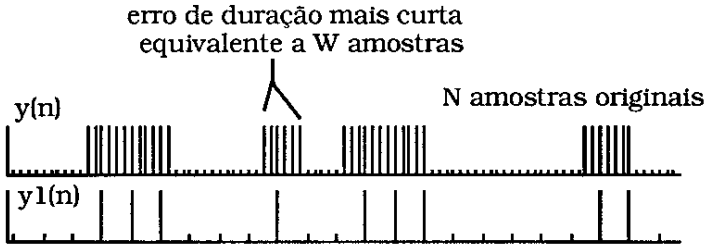


Figura 28. Exemplo da concatenação de informação do sinal de erro, estando representado y(n) (em cima) e $y_1(n)$; o erro com menor duração tem de estar representado no novo sinal, muito embora essa duração passe a ser a nova unidade de tempo (aprox.).

Apesar de todas estas considerações é irrefutável que este algoritmo consome algum tempo de processamento e memória. A solução apresenta contudo diversas vantagens, podendo servir de base para um algoritmo modificado, em que se procura detectar a existência de uma frequência de erros ***pré-determinada***. Nesse caso, utilizar-se-iam apenas os sinais com uma frequência fixa, embora com qualquer esfasamento.

Estas vantagens podem resumir-se em:

- as operações lógicas essenciais à execução da transformação podem ser efectuadas na altura da aquisição de dados;
- não é necessário recorrer a grandes operações matemáticas no cálculo;
- existe a possibilidade de limitar, consideravelmente o tempo de processamento;

De notar, que o método pode ser distribuído entre as estações de aquisição e o microcomputador central, nomeadamente deixando à estação a construção do sinal de erro e o restante processamento ao microcomputador central; a comunicação entre estes só seria efectuada se o nº de irregularidades fosse elevado (*limiar a definir pelo operador*). Acrescente-se que esta comunicação não ocupa muito tempo pois a informação a enviar seria do tipo digital, em que cada amostra ocupa um bit, sendo ainda possível utilizar técnicas de compactação de dados para diminuir o tamanho da comunicação.

Capítulo 5. Determinação de parâmetros têxteis.

5.1 Cálculo do CV%.

Uma das grandezas (funções) estatísticas mais importantes é o desvio padrão. Este fornece resultados sobre o desvio de um conjunto de amostras de um lote seleccionado relativamente à média dos referidos valores. O desvio padrão ∂ é calculado usando a expressão,

$$\partial = \sqrt{\frac{\sum_{i=0}^{N-1}(x_i-\overline{x})^2}{N-1}},$$ em que: x_i são as amostras,

$\overline{x}$ é o valor médio e

N o nº de amostras recolhidas;

esta expressão, não sendo muito favorável para o cálculo em tempo real, pode ser modificada dando origem a

$$\partial = \sqrt{\frac{\sum_{i=0}^{N-1}x_i^2 - N\overline{x}^2}{N-1}},$$

permitindo o cálculo à medida que a informação é adquirida.

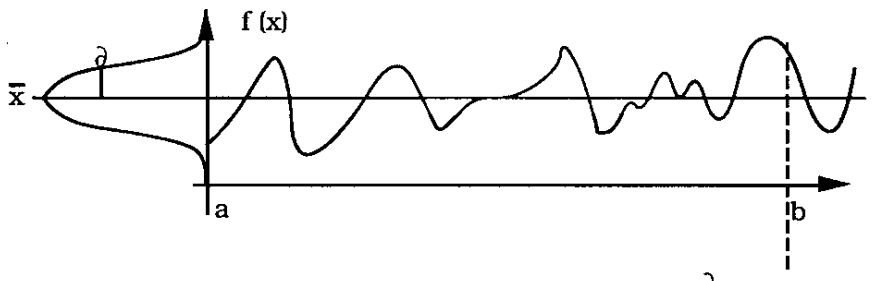
Esta medida do desvio padrão apresenta contudo a grande desvantagem de fornecer o resultado nas unidades das amostras, não permitindo assim uma informação percentual sobre as flutuações relativas à média.

Para obviar este problema, definiu-se o coeficiente de variação que fornece uma informação sobre a variação percentual das amostras relativamente à média, sendo definido pela seguinte expressão:

$$CV(\%) = \frac{\partial}{\overline{x}}100\% = \frac{\sqrt{\frac{\sum_{i=0}^{N-1}\left(x_i^2\right) - N\overline{x}^2}{N-1}}}{\overline{x}} * 100\% = \sqrt{\frac{\sum_{i=0}^{N-1}\left(x_i^2\right)}{(N-1)\overline{x}^2} - \frac{N}{N-1}}$$

em que: x_i são as amostras, $\overline{x}$ é o valor médio e N o nº de amostras recolhidas.

Este valor é vulgarmente utilizado na caracterização da regularidade do fio têxtil, sendo usual a obtenção de um valor na ordem de 10 a 16% **[63, 64]** (figura 29).



f(x) - sinal da massa do fio ∂ - desvio padrão $CV\% = 100\% \frac{\partial}{\overline{x}}$ $\overline{x}$ - média de f(x)

Figura 29. Relação do Coeficiente de Variação calculado com o fio analisado.

Para a determinação do CV%, *tradicional*, usam-se as amostras relativas à massa de fio captadas consecutivamente, isto é, cada amostra corresponde à massa do fio produzido no espaço coberto por um sensor. Para um sensor de 8 mm de comprimento, sendo efectuada a análise de N amostras, é estudado um segmento de fio têxtil com um comprimento total de N*8 mm; para conhecer CV% é calculada a média do conjunto dessas amostras, bem como o desvio padrão de N amostras .

Os valores calculados do coeficiente de variação são uma das principais características de um determinado fio têxtil. Além desse aspecto, existe uma relação típica entre este parâmetro e outros (como o U%) para cada tipo de fio têxtil ou, pelo menos, para a mesma qualidade de fio.

Para a determinação deste Coeficiente de Variação considera-se cada amostra correspondendo à massa de fio têxtil sob a 'acção' do sensor capacitivo. Esta medida de CV% é também por vezes referida com o CV_{total}%, com vista a distinguir de outra medida que referimos de seguida.

Esta outra medida, um coeficiente com definição e características muito semelhantes, é também um coeficiente de variação, denominado CV_1% (figura 30), que considera cada uma das amostras definida de outro modo: assim, cada uma delas corresponde ao valor médio de Ω amostras utilizadas no cálculo tradicional; podemos

assim dizer que cada amostra corresponde a um *pequeno segmento de fio* com o comprimento múltiplo do do sensor.

Para o cálculo do $CV_1\%$, consideram-se um conjunto de amostras obtidas a partir das N amostras atrás referidas. Cada uma das novas amostras é obtida calculando a média das amostras recolhidas directamente. O índice 1 de $CV_1\%$ fornece uma indicação do nº de amostras ***iniciais*** consideradas para o cálculo de uma nova amostra.

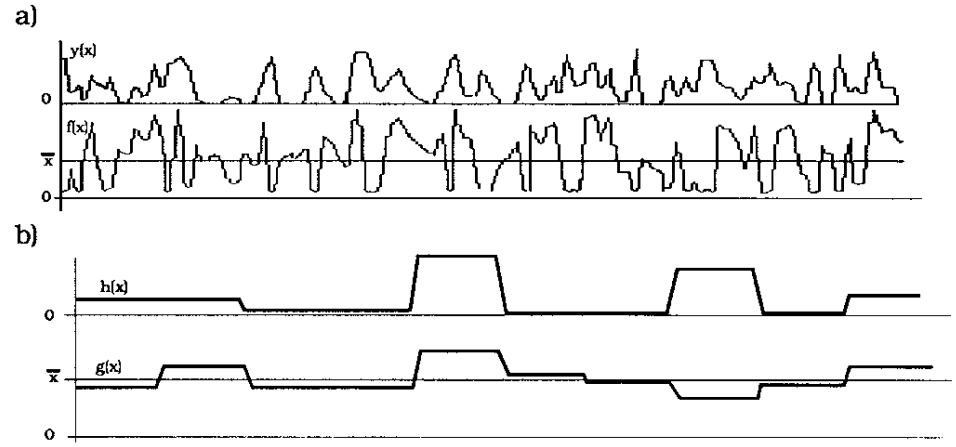


Figura 30. Determinação do $CV_1\%$, com Ω=10. Em a) apresenta-se o sinal a amostrar, f(x) e, com vista a facilitar a interpretação, o sinal $y(x) = (f(x)-\overline{x})^2$; em b) apresenta-se o sinal g(x), em que cada amostra é constituída pela média de Ω amostras do sinal f(x) e, de igual modo, o sinal $h(x) = (g(x)-\overline{x})^2$.

Assim, sendo x_i as amostras, $\overline{x}$ o valor médio (que se mantem igual), N o nº de amostras recolhidas, Ω o nº de amostras agrupadas, $R = \frac{N}{\Omega}$ o número de segmentos, e sendo cada nova amostra $y_j = \frac{\sum_{i=j*\Omega}^{(j+1)*\Omega-1} x_i}{\Omega}$ pelo que $CV_1\%$ pode então passar a ser definido por:

$$\sqrt{\frac{\sum_{j=0}^{R-1}\left(y_j^2\right) - R\overline{x}^2}{(R-1)\,\overline{x}^2}} * 100\,\%$$

Naturalmente, à medida que Ω tende para N o valor de CVj% tende para 0 atingindo esse valor quando Ω=N (nesta altura existe apenas uma única amostra, pelo que o desvio padrão é nulo e consequentemente CV%=0).

5.2 Determinação do Coeficiente de Irregularidade (U%).

A medida do parâmetro Desvio Médio Absoluto (conhecido pelo acrónimo MAD na língua inglesa), é utilizada na indústria de fiação têxtil através de uma medida relacionada, U%, fornecendo uma indicação similar ao do CV%.

Este parâmetro também fornece informação relativa ao 'desvio' de cada amostra para a média e, com vista a não depender das unidades, é também apresentado como um valor percentual da referida média (ver figura 31).

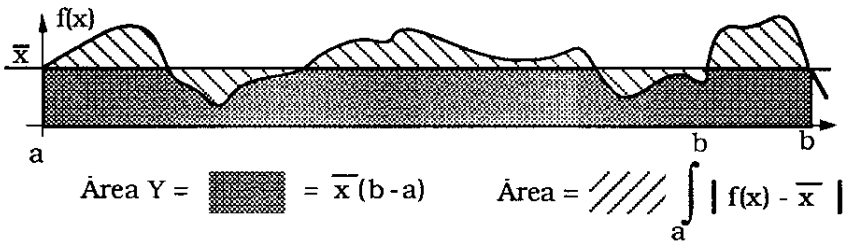


Figura 31. Relação do U% com o fio produzido; f(x) são as amostras adquiridas e U%, o valor do desvio médio absoluto, é igual a $\frac{\text{Área X}}{\text{Área Y}}$.

A partir desta descrição (e visualização gráfica), podemos concluir que o valor de U% é definido, e descrito matematicamente por uma expressão relacionando duas áreas:

$$U\% = \frac{\dfrac{\sum_{i=0}^{N-1} | x_i - \overline{x} |}{N}}{\overline{x}} 100\%,$$

em que: x_i são as amostras, $\overline{x}$ é o valor médio e N o nº de amostras recolhidas.

O valor de U%, que oscila tipicamente entre 8 e 12%, é sempre determinado com base nas amostras adquiridas consecutivamente e correspondentes a segmentos de fio contíguos; isto quer dizer que nenhum ponto do fio sob o sensor interfere ***mais que uma vez*** numa amostra.

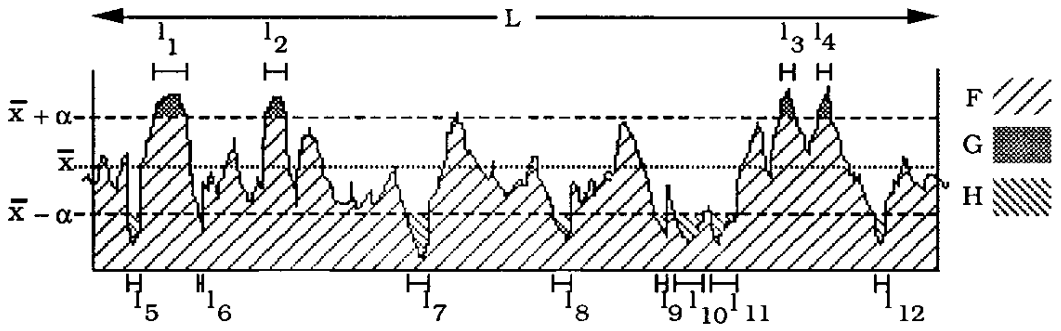
Relativamente ao parâmetro precedente, CV%, este valor não faz pesar tanto as diferenças relativas à média, uma vez que essa diferença não é maximizada (consideram-se as diferenças de cada amostra relativamente à média e não o seu quadrado); tal como no caso anterior os valores são independentes das unidades associadas às amostras, bem como do eventual factor de escala, sendo para o efeito efectuada a divisão do resultado pela média.

Existe, para a grande parte dos fios têxteis produzidos dentro de padrões aceitáveis, uma relação tradicional entre o CV% e o U%, na ordem de $\left(\frac{CV\%}{U\%}\right)$ 1.2.

Tanto a medição do CV% como do U% não dependem da velocidade de tracção do fio, uma vez que cada amostra *base* é definida pela **massa de fio sob a acção** do sensor; pelo contrário, a dimensão deste (em especial o comprimento) influencia esta medida.

5.3 Determinação do DR% e de IDR%.

A medição da Taxa de Desvio (DR derivado de "Deviation Rate"), fornece um resultado sobre o comprimento do fio analisado que ultrapassou (superior ou inferiormente) limites previamente estabelecidos (figura 32); este resultado é apresentado sobre a forma percentual.



$$DR\% = \frac{\sum_{i=1}^{S} l_i}{L} * 100\% \qquad IDR\% = \frac{\text{Área G + Área H}}{\text{Área F + Área G}} * 100 = \frac{(G+H)}{\bar{x}\ N} * 100\%$$

Figura 32. Cálculo do DR% e do IDR%; N é o nº de amostras recolhidas, $\bar{x}$ a média das amostras, L o comprimento de fio analisado, S o nº de segmentos que ultrapassaram o limite imposto, l_i o comprimento de cada um e as áreas F, G e H representadas na figura indicam a massa dos segmento de fio que ultrapassaram cada limite (G e H) ou não (F).

Com esta medida é possível classificar, ou melhor, obter mais um dado para qualificar o fio têxtil produzido. Muito embora este algoritmo seja vulgarmente utilizado (na indústria têxtil) para *um único nível de detecção*, optou-se no presente caso pela apresentação dos valores para diversos níveis de detecção, constituindo assim uma função ***f(α)***.

Sendo verdade a existência de uma relação entre a qualidade do fio produzido e o valor do parâmetro DR%, já utilizada por alguns fabricantes de aparelhos de medida, considerou-se também como um parâmetro a ter em conta o conhecimento de um valor que nos informasse da percentagem de massa de fio que excedeu os limites definidos para o cálculo do DR%.

Para conseguir obter essa característica definiu-se ***neste trabalho*** um novo parâmetro, IDR%, caracterizado pelos seguintes passos:

Em primeiro lugar é necessário 'escrever' uma função, y(n) definida como,

$$y(n) = \begin{cases} f(x) - (\overline{x} + \alpha) & \text{se} \quad f(x) \geq \overline{x} + \alpha \\ 0 & \text{se} \quad \overline{x} - \alpha < f(x) < \overline{x} + \alpha \\ (\overline{x} - \alpha) - f(x) & \text{se} \quad f(x) \leq \overline{x} - \alpha \end{cases}$$

que corresponde a definir uma função cujo valor indica ***em quanto*** foi excedido o limiar de detecção α; esta função só toma valores ***não nulos*** no caso de estes limites serem excedidos (figura 32); o parâmetro em questão, IDR%, para valores discretos , fica assim definido pela expressão

$$IDR\% = \frac{\sum_{i=0}^{N-1} y(n)}{\overline{x}\,N} * 100\%;$$

Como foi indicado, este valor é calculado para diferentes valores de α, pelo que na realidade definimos um parâmetro $IDR_{\alpha}\%$.

No caso de se considerar como limite superior e inferior o mesmo nível (isto é α=0 e portanto os limites $\overline{x}+\alpha$ e $\overline{x}-\alpha$ coincidirem com o valor médio) obtemos um valor de IDR% igual ao de U%, ou seja $\mathbf{IDR_0\% = U\%}$.

Também o parâmetro DR% pode ser descrito numa forma algébrica similar:
Em primeiro lugar foi necessário define-se uma função, p(n),

$$p(n) = \begin{cases} 1 & \text{se} \quad f(x) \geq \overline{x} + \alpha \\ 0 & \text{se} \quad \overline{x} - \alpha < f(x) < \overline{x} + \alpha \\ 1 & \text{se} \quad f(x) \leq \overline{x} - \alpha \end{cases}$$

pelo que $DR_\alpha\% = \frac{\sum_{i=0}^{N-1} p(n)}{N} * 100\%$; isto corresponde a efectuar a contagem das amostras que ultrapassaram o limite estabelecido, fornecendo um resultado idêntico ao da expressão $\frac{\sum_{i=1}^{s} l_i}{L} * 100\%$ se multiplicarmos cada *p(n)* e *N* pelo comprimento do sensor.

De referir que as funções associadas a ambos os parâmetros agora discutidos ($DR_\alpha\%$ e $IDR_\alpha\%$) apresentaram, em análises reais efectuadas, resultados similares e mantêm uma constância do aspecto da respectiva função.

Alguns estudos desenvolvidos por técnicos têxteis permitiram classificar com maior precisão diversos fios, para valores de α=5% da média, utilizando o parâmetro DR%; ao ser permitido utilizar esta função da forma descrita, que mesmo por inspecção visual, permite observar eventuais 'invulgaridades' (*a que podemos associar ainda o facto de dispormos do parâmetro IDR%*), concluímos ser da máxima utilidade utilizar estes parâmetros na caracterização do fio têxtil..

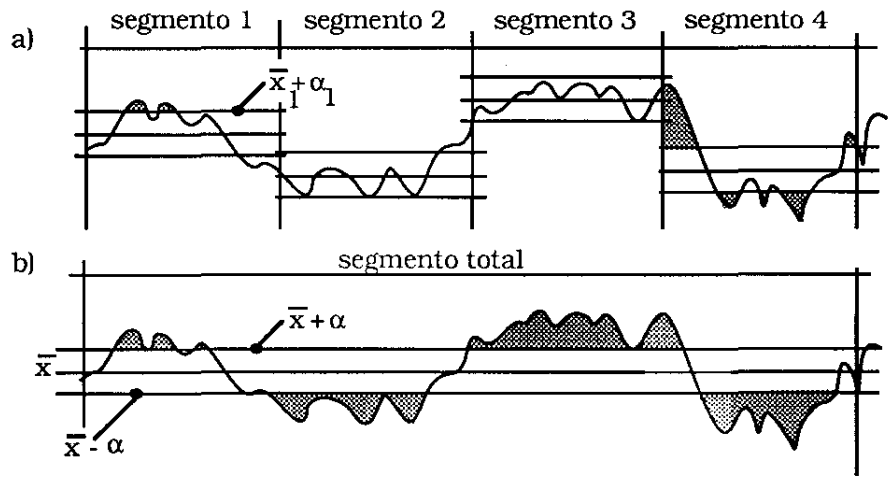


Figura 33. Relação entre o cálculo do DR% e IDR% total com o do $DR_1\%$ e $IDR_1\%$.

Como para o cálculo do CV% é possível efectuar os cálculos, quer do DR%, quer do IDR%, em vários segmentos consecutivos. Apresenta-se na figura 33 o efeito da utilização desta metodologia para efectuar o referido cálculo, neste caso o DR_1.

Estes novos valores calculados tendem para 0 à medida que aumentarmos o número de segmentos em causa, tal como acontecia para o CV_1%.

Realçar, por fim, que estes índices não constituem ainda "standard" na indústria têxtil (ao contrário do CV% e U%), embora forneçam resultados que contêm os tradicionais (notar que U%=IDR_0%, se exceptuarmos a análise espectral.

O parâmetro IDR% não foi até agora encontrado na literatura desta área, sendo uma proposta deste trabalho; relativamente ao parâmetro DR%, definido por técnicos da especialidade, permite 'retirar' informação complementar dos fios produzidos, sendo natural e previsível a generalização da sua aplicação (é já um parâmetro tradicional do equipamento *Keisokki*). Sendo muito facilitada a metodologia de cálculo utilizada, com recursos a processadores económicos e vulgarizados, parece-nos de extrema utilidade a utilização do parâmetro IDR%; de referir que este parâmetro teve grande acolhimento por parte de diversos técnicos têxteis sendo possível a generalização do seu uso como elemento complementar de análise da qualidade de um fio.

5.4 Determinação das irregularidades.

Como exposto, uma das principais necessidades do sistema a implementar centra-se na detecção dos 3 tipos de irregularidades do fio têxtil produzido: Pontos Grossos, Pontos Finos e Botões ("Neps").

São considerados valores típicos de limite para a detecção destas irregularidades os seguintes:

- ***Botões***: Massa superior a 2 * Valor médio durante um período muito curto e inferior a 4 mm.
- ***Pontos Grossos***: Massa superior a 1.5 * Valor médio durante um período mais longo, superior a 4 mm.
- ***Pontos Finos***: Massa inferior a 50% do Valor médio durante um período suficientemente longo, em geral limitado inferiormente a 4 mm.

Existe no entanto um problema delicado na detecção destas irregularidades. Esse problema, esquematizado na figura 34, deriva da utilização de um sensor com comprimento superior ao comprimento mínimo de algumas destas irregularidades (provavelmente 1mm para os NEPS).

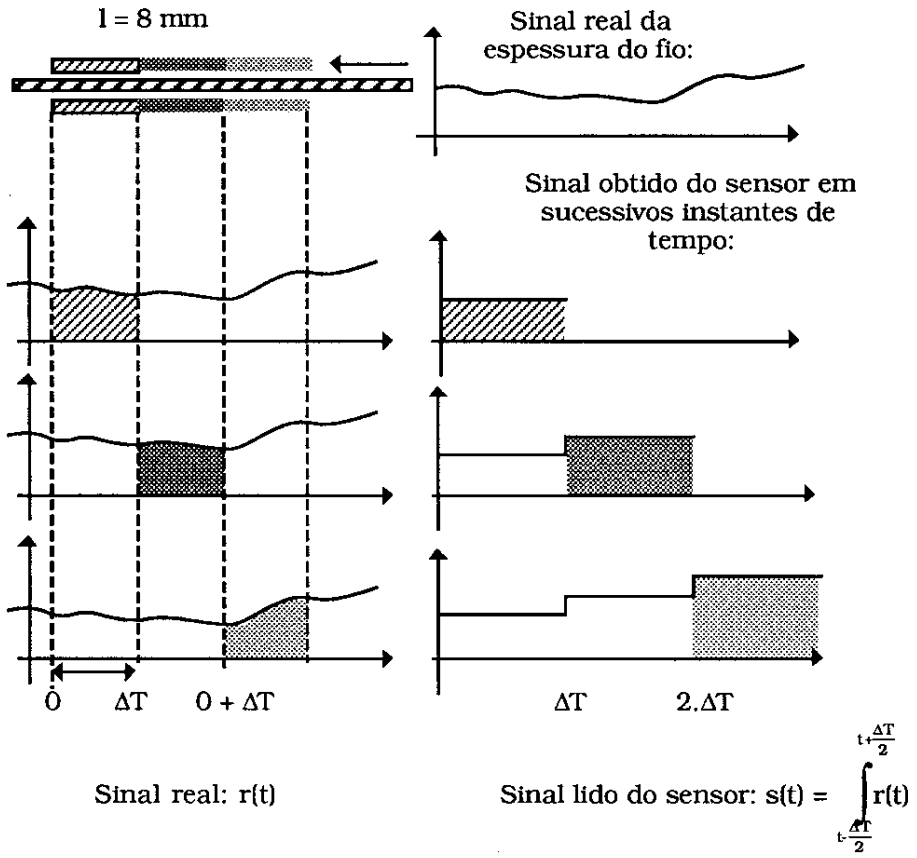


Figura 34. Exemplificação do sinal detectado no sensor e seu efeito sobre as amostras reais conseguidas.

Cada amostra obtida no (do) sensor capacitivo utilizado é igual à massa de **todo** o fio que está 'debaixo' do sensor. Este valor corresponde ao integral, ao longo do comprimento do sensor, da '*massa de cada secção*' de fio.

Sendo assim, o sinal resultante (se considerarmos a aquisição em instantes contíguos) mais não é que a correlação do sinal real com um impulso rectangular; este tem amplitude 1 e duração igual ao tempo que demora um ponto do fio a percorrer uma distância igual ao comprimento do sensor (na realidade é uma convolução, que neste caso apenas difere na consideração de quais os instantes positivos e negativos, dado o resultado de uma ou outra das operações serem simétricas).

5.4.1 Método tradicional de determinação de irregularidades.

Para obter o sinal real, basta-nos efectuar a desconvolução **[30, 64, 28]** (ou 'descorrelação) do sinal encontrado. A correlação no domínio dos tempos traduz-se, no das frequências, numa multiplicação de espectros, em que um deles é o conjugado. Para a convolução, o espectro resultante é igual ao produto dos dois espectros iniciais, isto é, o espectro do sinal da massa do fio e o espectro do impulso rectangular. Representa-se na figura 35 este último bem como o seu módulo.

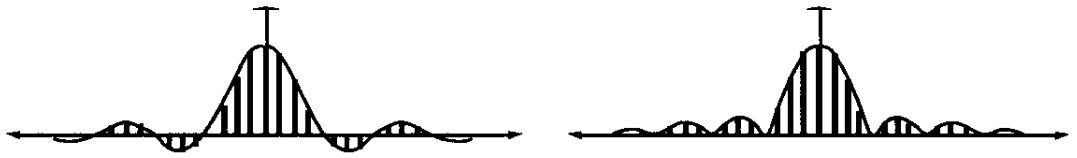


Figura 35. Espectro de um impulso rectangular e respectivo módulo, correspondentes aos da imagem do sensor do fio.

Sendo assim, e tendo em conta a provável rapidez do cálculo do espectro de sinais, seria previsível poder obter o espectro real, dividindo o correspondente ao do sinal encontrado, pelo respeitante ao impulso rectangular; esta suposição baseia-se no facto deste último ser perfeitamente conhecido e quantificado.

Após o cálculo do espectro do sinal real, aplicar-se-ia de novo o algoritmo da FFT, agora com vista a obter a transformada inversa, que corresponderia à imagem do sinal de espessura pretendido. O resultado desta operação seria então o sinal temporal com a imagem do sinal de espessura do fio têxtil realmente existente.

Como se verifica, por simples inspecção da figura 35, é de todo impossível efectuar a reconstrução exacta do espectro pretendido, uma vez que o respeitante ao impulso rectangular representado pelo comprimento do sensor apresenta valores nulos em algumas componentes, a que **não correspondem** riscas de valor nulo no espectro real do fio têxtil (pelo menos não o podemos garantir).

Este método de *desconvolução*, que quando implementado apresentou, para **certos casos**, resultados inaceitáveis, foi abandonado como método **generalizado** de detectar as imperfeições, tendo-se optado por uma abordagem diferente do problema.

Apesar desta posição, este algoritmo apresentou resultados convincentes para *alguns casos*, permitindo tirar conclusões muito próximas das esperadas.

Realça-se, no entanto, a grande dificuldade de aferir os resultados obtidos dado os métodos utilizados vulgarmente na indústria têxtil, sendo também utilizado um sensor de comprimento superior ao da imperfeição mais curta (em geral 8 mm), e não sendo explicitado qual o método utilizado para obter o sinal real, parecem carecer de uma validação clara.

5.4.2 Processos alternativos de cálculo das irregularidades.

A decisão de utilizar também outros métodos, levou a uma análise do problema que apontasse para a reconstrução o mais fiel possível do sinal original.

O primeiro aspecto a ter em conta é a descrição do sistema de medida em termos de um modelo. Assim, a medida efectuada cada 1 mm (por se considerar este valor como o mínimo comprimento a ter em conta para uma imperfeição tipo NEP) contém informação que diz respeito a K amostras reais (*em que* ***k*** *representa o comprimento do sensor em mm*), podendo ser escrita uma relação entre cada uma destas amostras reais e a amostra que se mede na realidade (figura 36):

$$y(n) = \sum_{i=n-(k-1)}^{n} x(i)$$

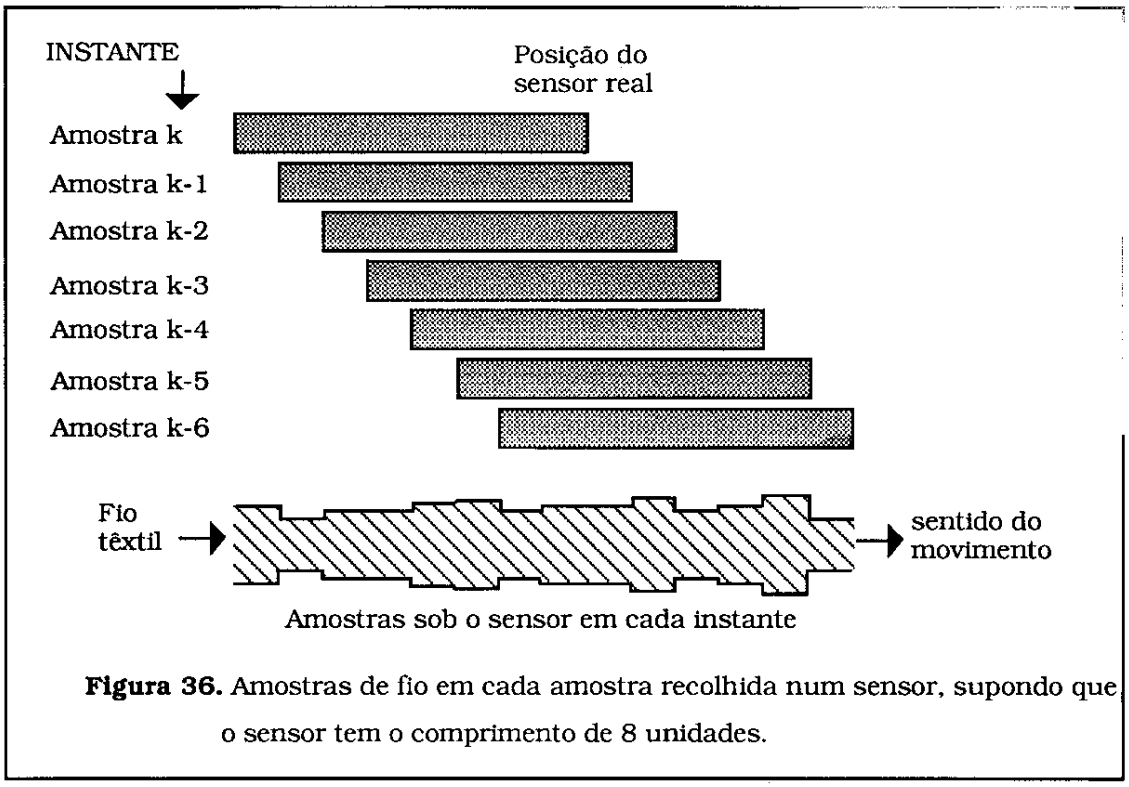


Figura 36. Amostras de fio em cada amostra recolhida num sensor, supondo que o sensor tem o comprimento de 8 unidades.

Nesta expressão, y(n) representa o valor das amostras sucessivamente recolhidas, correspondente à massa de segmentos de k mm de fio recolhidas cada 1 mm. Por seu turno, x(n) são as amostras que representam a massa de segmentos de 1 mm de fio, caso pudessem ser recolhidas (seria indispensável dispor de um sensor com 1 mm de comprimento).

Esta expressão pode ser reescrita da seguinte forma, para um sensor genérico:

$$y(n) = y(n-1) + x(n) - x(n - K) \qquad (\textbf{\textit{V.1}}).$$

Esta expressão pode ser modificada resultando: $x(n) = y(n) - (y(n-1) - x(n-k))$.

Resumindo, a amostra **real *actual*** é igual à diferença entre os valores medidos neste instante e no instante anterior, adicionando a amostra real **k** instantes antes.

Pode dizer-se que para a obtenção da amostra actual real só necessitamos de conhecer as condições iniciais de funcionamento.

A esta expressão corresponde a transformada em Z:

$$X(z) = Y(z) - \left(Y(z)\, z^{-1} - X(z)\, z^{-k} \right) \quad \Leftrightarrow \quad X(z) = Y(z)\, \frac{z^{k} - z^{k-1}}{z^{k} - 1} = z^{k-1}\, \frac{z-1}{z^{k} - 1}\, Y(z)$$

Para k=8: $Y(z) = X(z)+X(z)\, z^{-1}+X(z)\, z^{-2}+X(z)\, z^{-3}+X(z)\, z^{-4}+X(z)\, z^{-5}+X(z)\, z^{-6}+X(z)\, z^{-7}$,

$$\text{ou seja} \qquad Y(z) = X(z)\frac{z^{7}+z^{6}+z^{5}+z^{4}+z^{3}+z^{2}+z^{1}+1}{z^{7}}$$

que corresponde à convolução do sinal adquirido, representado pela transformada Z de y(), Y(z), com um sinal rectangular de 8 impulsos, com a transformada Z de um impulso rectangular de 8 amostras definida por

$$\frac{X(z)}{Y(z)} = \frac{z^{7}}{z^{7}+z^{6}+z^{5}+z^{4}+z^{3}+z^{2}+z+1} = \frac{z^{7}}{(z+1)(z^{2}+1)(z^{4}+1)} = \frac{z^{8} - z^{7}}{z^{8} - 1}$$, tal como já sugerido.

Para a obtenção dos valores representativos de x(n), o sinal de espessura real, teremos então de, ou resolver estas equações de forma a relacionarmos os valores reais com os obtidos das leituras, ou conseguir conhecer as amostras e os valores reais obtidos no instante anterior e 8 instantes antes respectivamente. Se optarmos pela 1ª hipótese, efectua-se a '**desconvolução**' do sinal, com as desvantagens já expostas.

Optando pela 2ª opção, verificamos com facilidade que o sistema só pode ser resolvido se encontrarmos os valores iniciais das primeiras **k** amostras, em que k é o número de amostras 'debaixo' da acção do sensor.

Existem duas aproximações possíveis a este problema.

Uma fornece o resultado **exacto** mas obriga ao conhecimento de um conjunto de amostras iniciais. Esta solução passa por adicionar a cada segmento de fio um outro já conhecido (provavelmente de espessura constante) ou seja, consiste em '**calibrar**' as medidas a efectuar.

Outra solução passa por utilizar um algoritmo de processamento que permita inferir o sinal *x(n)* a partir do sinal *y(n)* (exemplos na figura 37, 38 e 39).

Se é verdade que o inconveniente de acrescentar um segmento de fio (provavelmente de secção constante e portanto não um fio têxtil) de espessura conhecida não ser muito grande se tivermos de proceder a essa operação apenas uma vez por bobina, existe no entanto a desvantagem de todo o sistema depender da garantia da não ocorrerem erros na medição nem o fio partir; se porventura ocorrer uma destas situações, os erros de medição propagar-se-ão sucessivamente às amostras seguintes, dado cada amostra depender de valores anteriores. A única solução para este problema passa pela recalibração intermédia com o referido segmento conhecido.

Realça-se no entanto a particularidade deste método ser preciso (*especialmente para segmentos curtos, em que existe menos probabilidade de ocorrerem erros e serem propagados*), sendo portanto adequado a uma utilização mais restrita e controlada (controlo de qualidade).

Foram concebidos e simulados, por outro lado, diversos métodos de análise, quer exista ou não a calibração. A simulação destes diferentes métodos, para diversos níveis de percentagem de ocorrência das imperfeições permitiu concluir da possibilidade de os utilizar.

Esses métodos baseiam-se todos no cálculo de derivadas dos valores amostrados, com algum processamento suplementar:

- Derivada Simples,
- Derivada Controlada,
- Derivada Composta,
- Derivada Composta Relativa à Média.

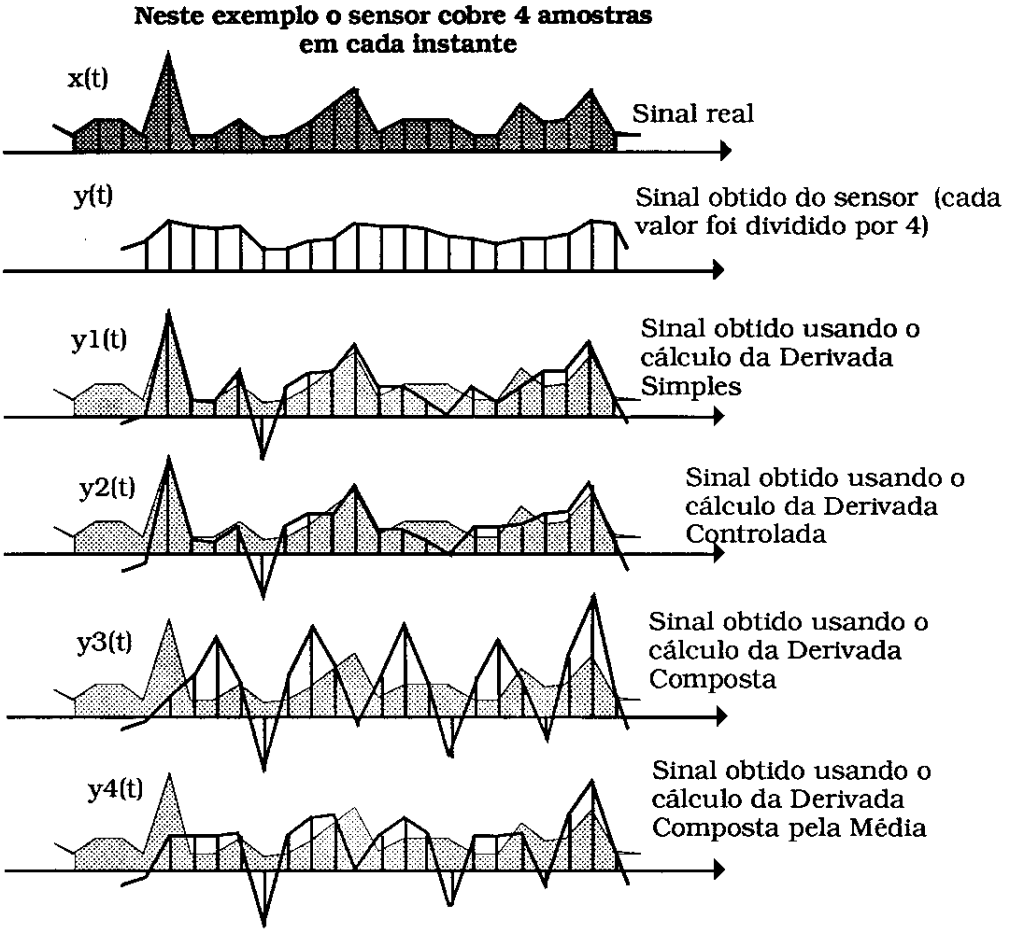


Figura 37. Exemplos da reconstrução do sinal original utilizando um sensor que 'cobre' 4 amostras; foram testados diversos métodos para a obtenção do sinal real utilizando algoritmos baseados no cálculo de derivadas (ver texto). De notar as diferenças/parecenças entre o sinal real, x(n) e o sinal resultante desse processamento $y_{1..4}(n)$.

Intervindo em diversos cálculos que se expõem de seguida, a média a utilizar é encontrada, a partir da informação recolhida directamente dos sensores:

$$\bar{x} = \frac{\sum_{i=1}^{N} y(i)}{N * k}$$

em que N é o número de amostras de todo o segmento, k o número de amostras sob o sensor e y(n) as amostras recolhidas cada mm (N = 21 e k = 4 na figura 37).

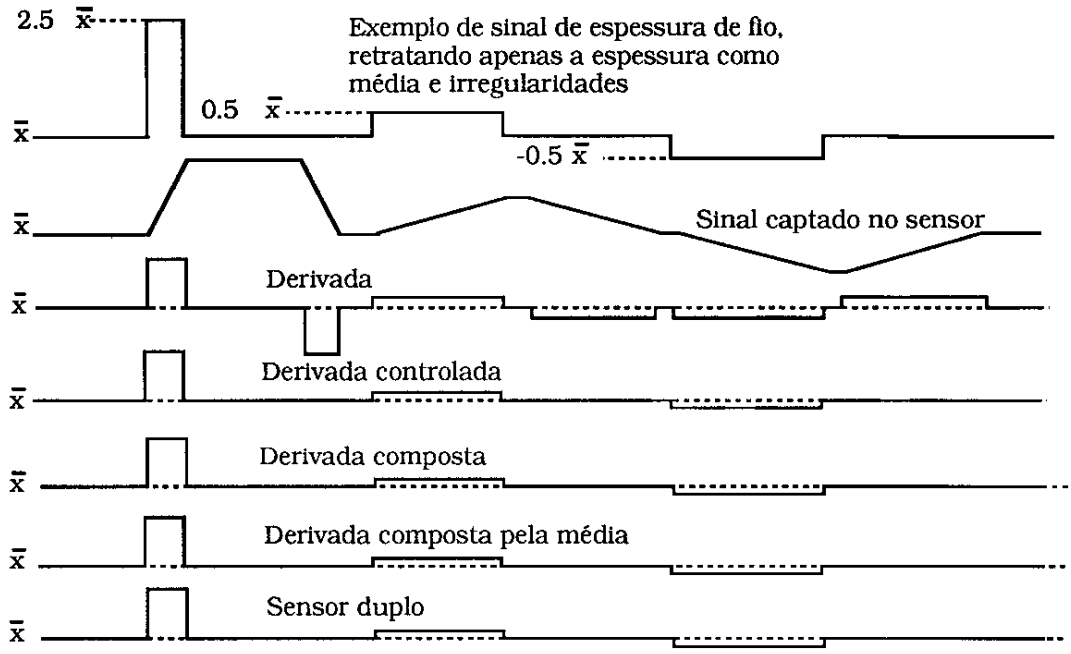


Figura 38. Exemplo dos resultados da reconstrução da imagem de um fio têxtil considerando apenas as irregularidades.

5.4.1.1 Derivada simples.

No primeiro caso, cada amostra x(n) é obtida através da derivação do sinal adquirido sendo adicionado o valor médio (ou componente contínua); a derivação, dado o sinal ser amostrado reduz-se ao cálculo da diferença entre amostras consecutivas, sendo assim obtido um sinal a(n), que pretende representar o real x(n), definido pela expressão:

$$a(n) = y(n) - y(n-1) + \bar{x} \qquad (V.2)$$

em que a(n) é a designação do sinal adoptado como imagem real do fio, y(n) as amostras recolhidas e $\bar{x}$ a média das amostras.

Este método é bastante eficaz para o caso em que existem poucas perturbações por unidade de comprimento, permitindo obter um sinal resultante bastante próximo do original (ver figura 38 e comparar com a 39). Existe no entanto um erro permanente que, 'além do mais', não é igualmente relacionável para **todas as amostras**, como se mostra em seguida:

Com base nas expressões (***V.1***) e (***V.2***) que nos relacionam ***y(n)*** com, respectivamente, o sinal real ***x(n)*** e o calculado (designado por ***a(n)***), podemos escrever uma relação entre estes. Esta relação **deveria ser**, note-se, **uma igualdade**.

$$x(n) = a(n) + (x(n-k) - \bar{x}) \quad \Leftrightarrow \quad a(n) = (x(n) - x(n-k)) + \bar{x})$$

Com estas expressões podemos afirmar que cada amostra detectada só depende da amostra real do presente instante e de k instantes anteriores adicionada à média. Esta última pode ser considerada uma constante com influência limitada ou pelo menos controlável. Conclui-se assim que os valores encontrados em cada instante não são mais do que uma combinação de **2 amostras reais** e não das k necessárias, muito embora o erro não seja acumulado.

Note-se que para a(n) ser igual ao valor x(n) pretendido, é necessário impor a condição de x(n-k) ser igual à média, o que naturalmente não é possível.

No caso de existir calibração, os resultados não são mais próximos da realidade, por esse motivo.

5.4.1.2 Derivada Controlada.

O algoritmo utilizado nesta solução é um pouco mais complexo, procurando evitar o efeito de uma imperfeição ser notada 2 vezes, como acontecia no caso da primeira derivada; como se pode verificar nas figuras 38 e 39, uma imperfeição isolada provoca uma subida do sinal seguido de uma descida que, após a derivação se traduz na existência de **dois** impulsos de sinal contrário. Com vista a minorar este problema foi definido um algoritmo de derivação, que procura evitar que este segundo efeito se faça sentir.

Assim, é utilizado o seguinte algoritmo:

se amostras são superiores à média
- **então**
  - **se** amostras estão a subir
    - **então** valor previsto = derivada do valor amostrado
    - **senão** valor previsto = média
- **senão**
  - **se** amostras estão a descer
    - **então** valor previsto = derivada do valor amostrado
    - **senão** valor previsto = média

A que corresponde, refinando um pouco mais:

se $y(n) > k.\bar{x}$ **então** **se** $y(n) > y(n-1)$ **então** $a(n) = y(n) - y(n-1) + \bar{x}$
senão $a(n) = \bar{x}$
senão **se** $y(n) < y(n-1)$ **então** $a(n) = y(n) - y(n-1) + \bar{x}$
senão $a(n) = \bar{x}$

Com este algoritmo, conseguem-se resultados bastante aceitáveis na detecção de irregularidades, quando a sua ocorrência não é muito frequente e representam uma variação considerável relativamente à média.

De realçar que apenas existe a preocupação de detectar a derivada quando se prevê a possibilidade de ocorrência de uma irregularidade.

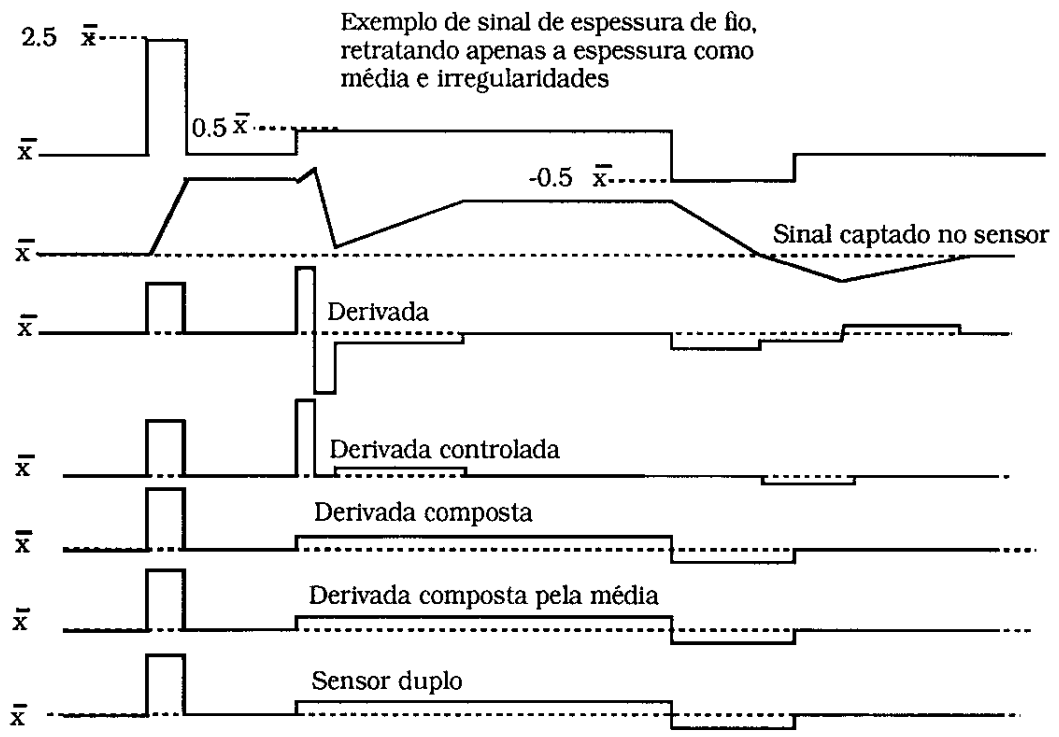


Figura 39. Exemplo da reconstrução da imagem do fio têxtil utilizando os diversos algoritmos propostos, mas com uma taxa de ocorrência de imperfeições mais elevada que o exemplo da figura 38. Verifica-se que diversas irregularidades se interpenetram com especial evidência em alguns dos métodos de análise.

5.4.1.3 Derivadas Compostas.

Existem dois algoritmos utilizados neste método designado de uma forma geral de Derivada Composta.

Estes algoritmos apenas diferem na metodologia de calcular as primeiras **k** amostras. Na primeira alternativa **Derivada Composta** as **k** primeiras amostras 'são' iguais ao valor $\frac{y(i)}{k}$ enquanto que, na **Derivada Composta Relativa à Média**, as **k** primeiras amostras são iguais à média de todo o segmento (actual ou anterior).

Existindo calibração, as **k** primeiras amostras (em geral k = 8) são constantes ou pelo menos conhecidas. Assim, usando este método, será sempre possível reconstruir o sinal original.

A equação que descreve o funcionamento destes algoritmos rege-se pela seguinte expressão:

$$a(n) = y(n) - (y(n-1) - a(n-8))$$

Esta expressão é de todo idêntica à referida em (***V.1***), com a diferença de utilizarmos diferentes variáveis; se pudermos garantir condições iniciais correctas (ou seja a(n - k) = x(n - k)), o que só poderá ser efectuado com o recurso à calibração, obteremos resultados 100% correctos

Com a simulação usando estes algoritmos conseguiram-se excelentes resultados, na detecção imperfeições. Na reconstrução do sinal verificou-se que as incorrecções, quando existem, são cíclicas e com periodicidade de k amostras.

O grande inconveniente, não testado durante a simulação por não existir informação sobre o assunto, reside no acumular de erros em todo o processo de cálculo. Efectivamente, sendo efectuada uma medida incorrecta num certo instante (por indução de um sinal na linha do conversor A/D por exemplo), será afectado não apenas o valor que está a ser calculado nesse instante, mas também todas as posteriores a este cálculo, de k em k amostras. Naturalmente que, existindo novo erro numa posição X períodos posteriores a esta, será eventualmente agravado esse erro.

5.4.1.3 Outro tipo de sensor.

Para ultrapassar estes problemas a melhor solução passa pela modificação do método de obter um sinal proporcional à espessura do fio em cada instante, ou seja modificar o sensor. A solução ideal seria assim a construção de um sensor de comprimento igual ao mínimo considerado na detecção de imperfeições, que seria

provavelmente na ordem de 1 mm. Dado este valor ser muito pequeno originando aos sensores a diminuição da sua precisão, será necessário modificar ou o tipo do sensor (um sensor óptico, provavelmente do tipo CCD usado nas câmaras video) ou utilizar outro processo de abordar esta questão; uma possível solução consiste na utilização de dois sensores do tipo capacitivo.

Como se mostra na figura 40, não será difícil acrescentar um segundo sensor ao já existente.

A finalidade desta utilização reside no facto de ser possível obter informação de segmentos mais pequenos de fio, por subtracção das massa que estão 'debaixo' de 2 sensores, por muito compridos que sejam.

Será no entanto necessário garantir que esta diferença de comprimento está no limites de detecção exigidos e não obriga a que o sensor tenha uma sensibilidade (e resolução) demasiado críticas.

Se pretendermos medir as imperfeições que ocorrem desde um limite inferior a Δx, basta para o efeito utilizar um sensor de comprimento L e outro de comprimento $L + \Delta x$. Por este método obviam-se as dificuldades associadas à construção de sensores demasiado pequenos.

Efectuando a leitura das amostras de fio que estão sob um dos sensores e efectuando, posteriormente, a medida do mesmo segmento de fio acrescido de um segmento de comprimento Δx sob o sensor mais comprido, conseguimos, por simples subtracção, conhecer a medida da porção Δx de fio.

Como se espera, esta metodologia obriga a uma eficiente calibração dos elementos responsáveis pela tracção do fio, provàvelmente rolos de estiragem, com vista a ser mantida constante a respectiva velocidade (figura 40 a) e b)). Esta necessidade advém neste particular da imposição de perfeita sincronização entre as medidas; esta deve-se iniciar (ou terminar) no mesmo ponto físico do do outro sensor, por forma a que a diferença entre os valores em causa não seja completamente disparatado.

Esta metodologia impõe algumas restrições:

- é necessário impor que a velocidade de tracção do fio seja constante;
- é necessário impor que os instantes em que são efectuadas as aquisições de informação coincidam com o início do mesmo segmento de fio;
- estas conclusões levam à imposição de que o atraso na aquisição da informação deva estar sincronizado com a velocidade de movimento do fio têxtil.

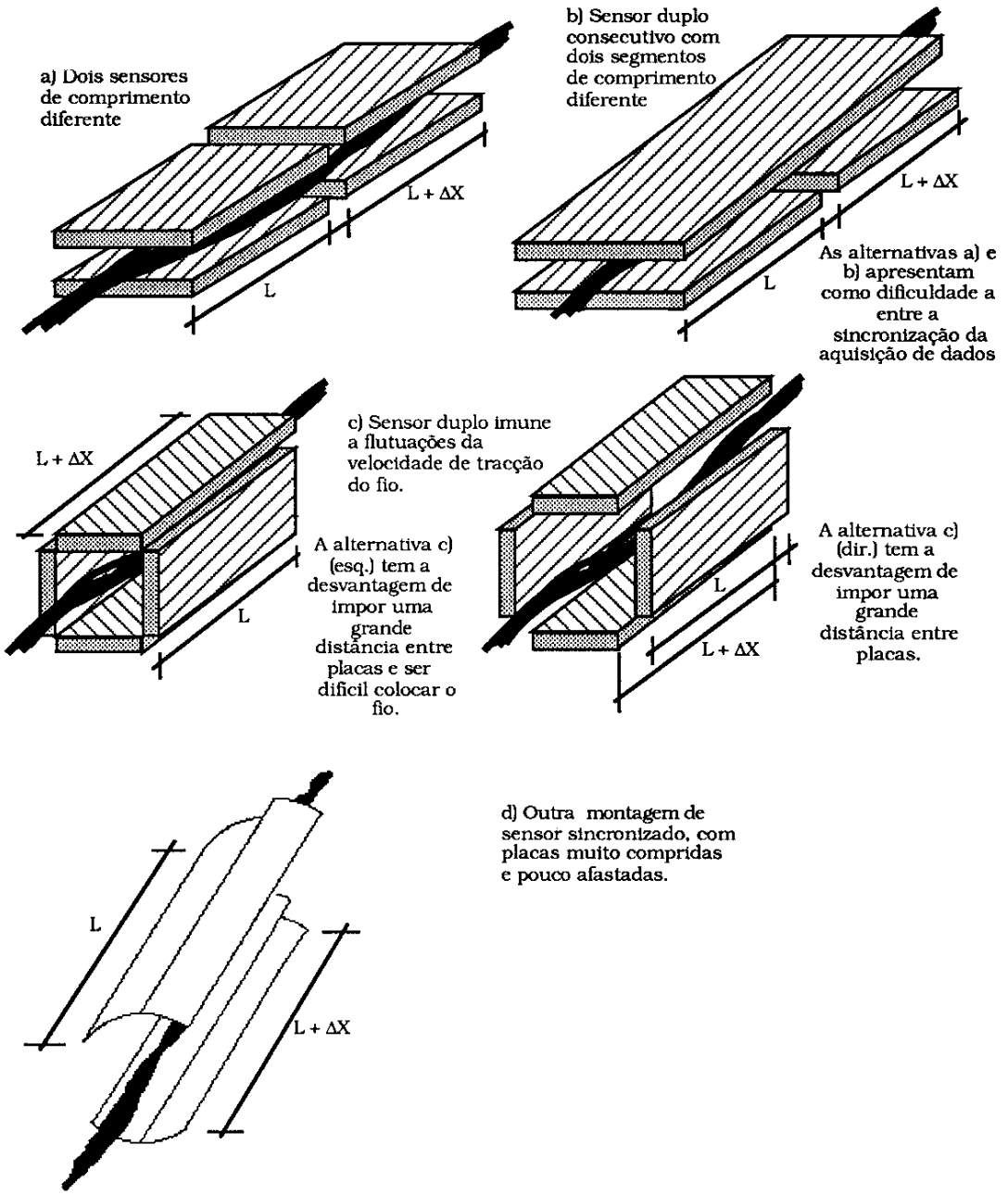


Figura 40. Exemplificação do aspecto físico dos sensores a utilizar para conseguir detectar imperfeições muito reduzidas no fio têxtil em análise.

Na figura 40 a) e b) apresentam-se 2 soluções possíveis em que se usa este método. Na b) apresenta-se uma mais económica e que permite uma construção mais fácil,

sendo de notar que, para detectar um comprimento mínimo de 1 mm num sensor que tem tipicamente 8 mm, poder-se-ia optar pelas placas inferiores estarem pràticamente encostadas, embora com isolamento, e possuir um comprimento de 4.5 mm e 3.5 mm ou 9 e 8 mm.

Na figura 40 c) mostram-se 2 hipóteses de construção do sensor, em que não existe o problema, referido anteriormente, do sincronismo entre a captação de cada amostra. Efectivamente, dado os sensores (ainda constituídos por um par de placas cada) estarem colocados fisicamente a partir do mesmo ponto, é possível efectuar a aquisição em simultâneo.

Existem no entanto três grandes inconvenientes associados:

- É necessário dispor de dois canais de conversão analógico-digitais.
- É necessário efectuar a aquisição do sinal no mesmo instante.
- Estando as placas do condensador, utilizado como sensor, muito próximas (1 mm a 5 mm), pode não ser possível colocar ortogonalmente, como proposto em c), o segundo par de placas.

Com vista a minimizar o último inconveniente da montagem da figura 40 c), propõem-se 2 soluções alternativas (figura 40 d), em que os sensores são bastante longos embora com uma pequena largura.

Assim, é possível manter a área de cada placa constituinte garantindo ainda uma grande proximidade entre cada uma das que formam um par. Além disso, se utilizarmos placas que constituam um segmento longitudinal da superfície exterior de um cilindro, consegue-se manter uma grande facilidade de manuseamento do fio pelo operador.

Capítulo 6. Técnicas de Inteligência Artificial no apoio ao diagnóstico de avarias.

A ocorrência de imperfeições no fabrico de um determinado produto provocam em regra consequências nefastas, que poderão levar à diminuição da qualidade do produto final produzido e mesmo, em certos casos, ocasionar a rejeição de um grande lote da produção.

As falhas que estas imperfeições originam são em geral bem conhecidas dos diversos intervenientes do processo de fabrico e podem mesmo ser evitadas desde que haja indícios claros da sua ocorrência prévia. Não podemos contudo generalizar esse conhecimento a todos os intervenientes uma vez que, por um lado a quantidade de conhecimentos e informações que cada um teria de possuir seria em quantidade exagerada e, por outro lado, muito deste conhecimento deriva directamente da experiência conseguida ao fim de bastante tempo de trabalho com situações similares.

A associação dos efeitos de eventuais falhas, que possuem uma taxa de ocorrência bastante elevada, com outras que têm uma frequência bastante inferior, também não é garantida pelo interveniente no processo de fabrico, em parte devido à quantidade de informação que seria necessário possuir em simultâneo.

A existência de uma interdependência entre a ocorrência de uma determinada falha e um conjunto de causas que a podem provocar, embora tradicionalmente exista, não é detectada senão por um operador bastante treinado. A resolução da causa que provocou essa falha nem sempre é efectuada no ponto exacto do processo de fabrico, sendo por vezes efectuada uma correcção que, muito embora melhore os resultados não garante a não ocorrência da mesma falha posteriormente.

Além disto, quando uma falha surge numa certa fase do processo de fabrico e a causa que a origina está numa outra fase a que o interveniente não tem acesso, é frequente não ser possível detectar rapidamente a sua origem ou ser mesmo diagnosticada uma causa incorrecta, na primeira etapa de teste. Apesar de a sua correcção ser provavelmente indispensável, implica uma perda de tempo elevada.

O melhor processo de resolver estes problemas seria conceber um modelo matemático que permitisse **prever** a ocorrência destas falhas ou, pelo menos, a existência de um modelo que permitisse relacioná-las de imediato com as causas que, com maior probabilidade, as originaram.

Por outro lado, seria ideal a existência de um modelo paralelo que de imediato apontasse para as acções a tomar para resolver um determinado problema, minorando assim custos de produção, gestão e consequentemente permitindo uma

melhoria significativa da qualidade ou do preço do produto acabado (eventualmente e preferencialmente ambos).

Em geral para uma grande gama de actividades esta solução é impossível ou pelo menos bastante limitada.

Estamos assim condenados à existência de especialistas em tarefas muito particularizadas, especialistas esses que têm um grande conhecimento sobre alguns dos problemas típicos de um certo passo do processo de fabrico mas desconhecem por completo o processo a montante e a jusante. Estes especialistas têm portanto o seu conhecimento cimentado numa **base de dados** pessoal construída ao longo do tempo e em que os aspectos associados a ocorrências menos frequentes são progressivamente esquecidos.

O desenvolvimento de um módulo de programação associado a este tipo de ocorrências aponta para a utilização de um microcomputador que vá recolhendo a informação associada às avarias; este microcomputador será o responsável pela gestão das estações remotas **[42, 54, 55]**.

Numa fase posterior esta máquina forneceria **sugestões** sobre as causas que originaram eventuais falhas ou mau funcionamento de máquinas e/ou as acções a empreender para a sua correcção.

Se inicialmente a máquina mais não faria do que "atrapalhar" o trabalho do operador posteriormente, seguindo um "esquema de raciocínio" indêntico ao humano (*quer dizer por pesquisa relacional numa base de dados*), indicaria soluções prováveis para evitar imperfeições no fabrico. Espera-se que no cômputo geral a máquina seja capaz de apresentar informação que o operador dificilmente conseguiria, em tempo útil, recordar **[51]**.

Refira-se, a título de exemplo, que uma das utilizações mais típicas desta análise é a associada ao diagnóstico médico; neste caso, com base em sintomas de doenças e respectivas curas o médico utiliza um algoritmo (idêntico ao da detecção de falhas) sugerindo posteriormente as acções que obviarão o problema. Mesmo que recorra a métodos auxiliares de diagnóstico, o interveniente baseia-se sempre na pesquisa de uma base de dados de sintomas, ou melhor uma base de dados de associação de sintomas, que apontam, com diferentes probabilidades para as acções passíveis de tomar **[52, 58, 60]**.

Para todo o esquema ser efectivamente aplicável é indispensável que, para um determinado grupo de sintomas, sejam conhecidas todas as acções já utilizadas em instantes diferentes. Por outro lado é necessário que as acções apresentadas para a resolução do problema sejam também **inferidas** das acções já utilizadas para resolver situações similares, isto é, correlacionar falhas diferentes mas que tenham aspectos comuns nas imperfeições detectadas **[56]**.

O sistema que se preconiza, procura assim utilizar as relações de **causa efeito** realmente existentes, para **prever e solucionar** ocorrências indesejáveis no processo de fabrico.

O processo de implementar esta metodologia, baseando-se em técnicas do que vulgarmente se designa como '**inteligência artificial**', provou ser de grande utilidade, permitindo estabelecer relações entre sintomas e 'curas' que não é fácil de detectar genericamente.

Naturalmente, como já se referiu, estas técnicas só têm possibilidade de aplicação em casos, como o presente, em que os diversos factores intervenientes no processo de fabrico não permitem estabelecer (***ainda***) um modelo matemático preciso; sendo possível definir um modelo que relacione as falhas com as causas que os provocaram, seria preferível implementar um algoritmo que as resolvesse. Realce-se que a utilização destas metodologias aponta para no futuro se poder **inferir um modelo matemático** do sistema a controlar a partir dos resultados que se vão obtendo.

O sistema a desenvolver tem assim todas as características de um ***sistema pericial*** ("expert system") **[57, 59, 60]**; uma base de conhecimentos é permanentemente actualizada com informação referente a avarias e acções de correcção, permitindo estabelecer ***ligações*** entre ambas, procurando-se definir ***regras*** que permitam reger toda a informação disponível com o objectivo final de ***prever*** avarias e/ou ***aconselhar soluções***.

Sendo verdade que todo este esquema não é garantidamente eficaz, não sendo possível provar que todas as acções são garantidamente correctas, apresenta algumas vantagens sobre os métodos actualmente em uso.

Destacam-se de seguidas algumas das razões que nos levam a essa conclusão:

- A base de dados está sempre presente.

A base de dados só poderá ser esquecida devido a uma falha do microcomputador central ou dos seus periféricos, ou de uma incorrecta actuação do operador; apesar disso nunca será 'apagada' a informação sobre a falha e acção seleccionada.

- A base de dados está permanentemente actualizada.

Nunca se corre o risco de, por exemplo por mudança de um operador, passarmos a dispor de uma base de dados desactualizada, como ocorreria se a troca fosse apenas a nível de especialistas.

- Mesmo as falhas menos frequentes terão uma sugestão para a sua correcção.

Por muito improvável que seja a ocorrência de uma falha, ou poucas vezes tenham sido utilizadas certas acções para corrigir anomalias, será sempre tomada em linha de conta a informação existente sobre elas. Com isto permite-se que uma ocorrência, que por vezes é muito grave, embora seja pouco frequente não seja completamente esquecida.

- Existindo correlação entre falhas, esta nunca será desprezada ao apresentar as possíveis soluções.

Mesmo que não exista uma acção que solucione uma falha, provavelmente por esta última nunca ter ocorrido, será sempre apresentada informação relativa a soluções de falhas similares, isto é, falhas que apresentem pelo menos um sintoma comum.

- Existe a possibilidade de estender a base de dados a todos os operadores de fases de fabrico idênticas.

Este é um dos aspecto mais atraentes, e que se prende com a possibilidade de utilizarmos o conhecimento do mais experiente dos especialistas. Com efeito, é sempre possível colocar ao dispor de operadores de fase de fabrico idênticas, as bases de dados mais desenvolvidas ou de melhor qualidade. Poderá, ao fim de um certo tempo, ser seleccionada uma base correspondente à reunião de toda a informação conseguida em **todas as fases de fabrico idênticas**.

- É possível aceder, por meio da rede, à base especializada.

Estando os microcomputadores de uma fábrica todos interligados, a interligação de todas as bases é possível e provavelmente poderá ser efectuada em tempo real.

A par das vantagens descritas acrescentam-se, infelizmente, algumas desvantagens:

- Mais uma tarefa a realizar pelo operador.

O operador, além de controlar a fase do processo de fabrico de que está encarregado, necessita ainda de, no caso de ocorrer uma falha, estabelecer um diálogo com o sistema para saber qual a acção a tomar, não sendo sequer garantido que este último forneça um resultado ou, fornecendo-o, que este seja convincente. Isto passar-se-á quanto menor for o número de avarias existentes na base, por outras palavras, quanto maior for o número de avarias já detectadas mais eficaz será a resposta. Quando este número for bastante pequeno, pode obviar-se este problema introduzindo a informação em tempo diferido ("off-line").

- Depende sempre da qualidade do operador a actualização da base.

A informação a residir na base de dados será sempre introduzida pela acção do operador, com excepção de falhas detectáveis automaticamente pelas estações remotas; como em princípio estas não serão o grosso delas, depende da veracidade das informações introduzidas a qualidade das acções propostas para corrigir uma anomalia.

- Inicialmente não são apresentados resultados satisfatórios.

Dado o sistema necessitar da ocorrência de avarias para ser actualizado com informação, inicialmente não são apresentados muitos resultados ou não aparecem sequer quaisquer resultados.

Embora um sistema como o descrito (ver figura 41) possa ter uma utilização genérica, consideraram-se alguns aspectos essenciais para a sua concepção; este têm principalmente em conta a arquitectura do sistema de controlo, as tarefas que é necessário executar, o tipo de utilizadores que trabalharão com o sistema.

Estes pontos podem resumir-se brevemente nas seguintes linhas de orientação:

- O sistema vai funcionar em tempo diferido.

A definição deste sistema obriga a um interface entre o utilizador e a máquina, uma vez que a base de dados é construída assente nas respostas que o primeiro vai fornecendo; é portanto natural que este sistema funcione em tempo diferido, por forma a não atrasar o funcionamento global.

Apesar deste aspecto, refira-se que após algumas sessões de trabalho é previsível que a relação qualidade/tempo de resposta seja melhorada; este facto que advém de o sistema ir 'aprendendo', permitirá a sua eventual utilização em tempo real **[51]**.

- Existem diversos tipos de utilizadores do sistema.

Dado estar previsto que a utilização do sistema de detecção de avarias seja efectuada por diversos perfis de utilizadores, é conveniente prever um interface simples e eficiente entre o sistema e o operador. A título de exemplo refira-se que deve ser efectuado o "backup" automático da base de dados.

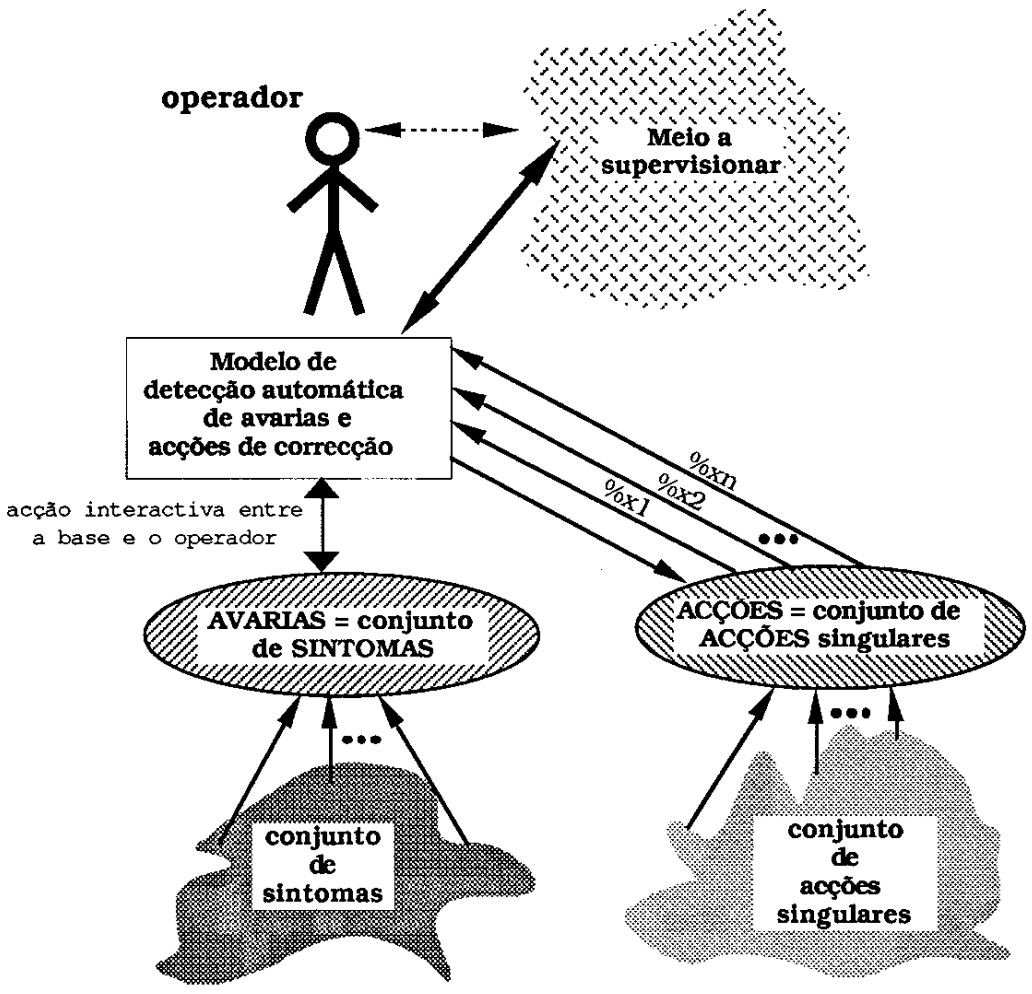


Figura 41. Representação do sistema de detecção de avarias e proposta das acções de correcção respectivas; cada avaria é definida como um conjunto de sintomas e cada uma das possíveis acções de correcção é definida como um associação lógica de diversas acções. A cada uma destas acções está associada uma probabilidade de solução da avaria.

- O sistema funcionará quer para detectar imperfeições no produto, quer para corrigir avarias das máquinas, quer falhas ocorridas no processo de fabrico.

Ao especificar este algoritmo, chegou-se à conclusão que, no essencial, pode ser aplicado à detecção de falhas, avarias ou imperfeições dos diversos intervenientes do processo de fabrico do fio têxtil. Este aspecto leva-nos a considerar como condição essencial a existência de diversas bases de dados, uma para a análise da produção do fio têxtil, outras para a análise e supervisão da maquinaria e, eventualmente, outras para a supervisão do próprio regularímetro.

Como se esquematiza na figura 41, existem dois grandes conjuntos de dados, sobre os quais é efectuado uma pesquisa; esses conjuntos dizem respeito, em traços gerais, às avarias um, e às acções de correcção dessas mesmas falhas outro; enquanto um é utilizado para guardar a informação respeitante às avarias, através dos sintomas detectados, o outro é geralmente pesquisado para obter informação relativamente às acções a empreender para corrigir a falha. Pode assim dizer-se que no sistema a implementar, deve existir uma **correspondência** mais ou menos directa entre uma avaria e as acções que a solucionam; esta correspondência varia, conforme o sucesso encontrado na acção seleccionada, isto é, depende da experiência acumulada.

Capítulo 7. Comunicação entre microcomputador central e estações remotas

Todo o sistema a implementar baseia-se na transacção de informação entre os dispositivos de aquisição de dados e o microcomputador central de gestão e controlo dessa informação. Nesse sentido são requisitos do sistema a existência de uma comunicação rápida, eficiente e segura entre o microcomputador central e as estações remotas.

Dado o tipo de sistema concebido apresentar requisitos de complexidade variável - embora bastante precisa para cada caso -, a concepção das primitivas de comunicação terá de levar em conta esse factor; nesse sentido, conforme os diversos níveis de programação que recorrem às primitivas de comunicação, quer nas estações quer no computador central, é necessário prever diferentes metodologias no uso destas, obrigando-as assim a ser o mais "standard" e transparentes possível.

Estas considerações levam a que se torne indispensável o recurso a um protocolo de comunicações definido por camadas; conforme o nível de programação que utiliza as primitivas de comunicação, assim se 'entrará' neste módulo em diferentes camadas.

Com esta opção garante-se também que a evolução de módulos, quer do "hardware", quer do "software", não obriguem a uma completa reformulação de todo o sistema de comunicação de dados.

Considerou-se como modelo inspirador do protocolo a implementar a definição da ligação entre Sistemas Abertos (OSI- Open Systems Interconnection) proposta pela ISO (International Standard Organization). Foram assim definidas as seguintes camadas:

1 - Camada Física **[19]**.
2 - Camada de Controlo de Ligação **[14, 18]**.
3 - Camada de Controlo de Mensagem.

7.1 Camada física

O protocolo físico seleccionado é o RS-485; este protocolo permite já uma ligação do tipo barramento dos diversos intervenientes na comunicação, sendo uma actualização de outros "standards" físicos.

Comparativamente a protocolos tradicionais, este apresenta diversas vantagens, de que se apresenta nas linhas seguintes o respectivo destaque.

Relativamente ao RS-232 C:

- Possibilidade de interligação de diversos nós de comunicação no mesmo barramento.
- Suporta velocidades de comunicação bastante elevadas (na ordem dos Mbit/s).
- Necessita apenas de um par de fios para a comunicação.
- Usa comunicação diferencial (balanceada).
- Necessita apenas de uma fonte de alimentação para o emissor e/ou receptor.

Relativamente ao RS-422:

- Suporta uma tensão de modo comum mais elevada.
- Necessita de uma tensão diferencial mais baixa para ser detectado um bit.
- Suporta velocidades de transmissão mais elevadas.
- Tem uma protecção de isolamento da linha que é accionada quando é atingida uma temperatura elevada do "chip"de comunicação (emissor/receptor).
- Suporta 32 nós na rede numa extensão de até 1200 metros
- A entrada em funcionamento simultâneo de diversos emissores não provoca a destruição intempestiva de nenhum deles.

Este protocolo tem também algumas desvantagens:

- Por um lado os circuitos integrados que fazem o interface com o exterior são ainda bastante dispendiosos .
- Para a interligação de certos equipamentos é sempre necessário utilizar um par de fios por linha de comunicação.

A escolha deste protocolo não acarreta qualquer imposição nem limitação nas camadas superiores do protocolo a implementar.

7.2 Camada de Controlo de Ligação

As estações remotas de aquisição de dados estão previstas para suportarem dois tipo de microcontroladores, que implementam processos diversos de comunicar, no que diz respeito a esta camada:

Comunicação orientado ao caracter:

- Microcontroladores tipo 8051/2, com comunicação implementado por uma UART interna (Universal Asynchronous Receiver Transmiter) **[5]**.

Comunicação orientada ao bit:

- Microcontroladores tipo 8044/8344, com comunicação implementada por um controlador de SDLC **[7]**.

A UART do primeiro grupo de máquinas implementa automaticamente diversos métodos de comunicação de dados. Entre eles destaca-se o "Protocolo do Nono Bit".

O recurso a este protocolo permite que diversas máquinas interligadas entre si possam ser "acordadas" automaticamente com a recepção de um caracter com o nono bit 'activo'. A recepção de caracteres sem esse bit 'activo' garante que as estações não serão perturbadas, exceptuando aquela com que já se está a efectuar uma comunicação. De realçar que o acordar de uma determinada estação está associado a uma interrupção física.

Nas estações presentemente em uso optou-se pelo primeiro tipo de microcontroladores, o que obrigou à definição de um protocolo de controlo de ligação orientado ao caracter **[20]**.

Contudo, dada a possibilidade de evolução para uma solução englobando o segundo tipo de microcontroladores, é necessário garantir que essa evolução acarrete apenas uma modificação da programação que depende directamente do tipo de microcontrolador; esta camada, dado o microcontrolador enviar automaticamente a informação pretendida (caracter ou pacote), é designada, no modelo OSI, por *camada de controlo de ligação*.

Sendo assim, define-se a metodologia das chamadas das primitivas que implementam o controlo de ligação; estas serão mantidas, qualquer que seja o microcontrolador utilizado, sendo provável que esta chamadas sejam efectuadas pelos módulos de programação de nível superior, isto é, a camada de controlo de mensagem **[12]**.

O protocolo a implementar deverá ser do tipo "master/slave" tendo o microcomputador central como mestre ("master") e as estações remotas de aquisição de dados como escravos ("slaves"); esta consideração advém da estrutura do sistema a implementar, dado ser improvável as estações remotas necessitarem de comunicar entre si. A acrescentar a este aspecto, sublinhe-se a frequência com que serão

efectuados cálculos e demais processamento sobre dados, adquiridos em cada estação, por parte do microcomputador.

Tomando esta decisão podemos dividir as tarefas entre os diversos intervenientes nas comunicações: o"master" poderá ***enviar comandos e receber respostas*** originárias das estações de aquisição; as estações remotas poderão ***receber comandos e enviar respostas***.

Considera-se que nenhuma ligação (correspondendo provavelmente às trocas de todos os '*pacotes de dados*' que constituem uma mensagem) poderá ser terminada sem erro se não tiverem sido reconhecidos (um a um ou em grupo) todos os 'pacotes' que entretanto circularam na linha (*o conjunto de dados designado por 'pacote' não tem ainda uma correspondência directa com as estruturas básicas de transferência de dados*).

Foi considerada ainda uma figura de excepção, que permite a qualquer nó da rede enviar uma comunicação excepcional (normalmente de emergência); não necessitando de reconhecimento para ser considerado válido, ficará a cargo do remetente decidir sobre a sua provável *boa recepção* no destinatário.

Apresentam-se de seguida as características do protocolo orientado ao caracter definido. Essas características são especificadas em 3 partes:

- Definição da estrutura das unidades mínimas de comunicação (vulgarmente designadas de "frames"), que permitem definir o que pode circular na linha, bem como a sua estrutura e funções.

- Descrição dos elementos de procedimento. Estes permitirão sequenciar "frames", detectar incongruências e falta de trocas de informação em tempo útil. Nesta fase serão também descritos os reportórios (reportório = conjunto de) de comandos e respostas utilizáveis.

- Descrição das classes de procedimentos que definem o encadeamento de comandos e respostas que permitirão a transacção eficiente de unidades de comunicação através da linha bem como recuperar de eventuais erros que possam ocorrer, sem recorrer, tanto quanto possível a camadas de programação superiores. Neste ponto serão descritos os algoritmos de algumas das primitivas de comunicação que serão posteriormente refinados.

7.2.1 Estrutura da unidade básica de comunicação ("frame").

A estrutura da unidade básica de comunicação (*UC* ou "frame") é a porção mínima de comunicação com significado, neste protocolo. Sendo este orientado ao caracter, esta unidade será constituída por caracteres. A falha de recepção ou transmissão de um ou vários dos caracteres que constituem a "frame" tornam-a ininteligível e originam a sua não compreensão.

Cada caracter transmitido será, como tradicionalmente, precedido de um "start bit" e terminado por um ou dois "stop bits". Entre os "start" e "stop" bits está contido o caracter definido por um conjunto de *8 bits* que representam a informação e um *nono* bit. Este nono bit terá sempre o valor 0 excepto *para o primeiro caracter* do "frame", numa comunicação **do** *microcomputador central* **para** *uma estação*, (a "flag"); neste caso toma o valor 1.

A estrutura da unidade de básica de comunicação (**UC**, **trama** ou usando o termo inglês "**frame**") é a seguinte:

"flag"	Nºbytes	Cabeçalho	Numeração	Tipo	Endereço	Informação	Erro	
1	1	3	1	1	2	até: 64	1	bytes

Esta estrutura mantem-se quer a direcção da comunicação seja num sentido quer noutro, exceptuando o caso da "flag"

O comprimento desta estrutura embora limitado em regime experimental a 74 bytes, pode apresentar comprimento variável.

I - "flag".

A função da "flag" é garantir um sincronismo entre o receptor e o emissor da UC. A sua recepção inicializa o receptor para a detecção de um "frame". A "flag" é facilmente detectável pelas estações já que, a sua recepção com o nono bit em 1 provoca uma interrupção que permite sincronizá-la com o "frame" a chegar.

Quando o microcomputador central pretende enviar informação para alguma estação remota de aquisição de dados, esta "flag" será composta por um byte igual ao endereço da respectiva estação. Para ser garantida a detecção da UC esta "flag" é implementada recorrendo ao uso do nono bit.

Este nono bit transmitido através da linha de comunicação fará acordar todas as estações de aquisição *penduradas* na rede. Aquela que possuir o endereço lógico igual ao recebido na "flag" preparar-se-á para a recepção de um frame, a partir dos caracteres que se seguirem e que terão obrigatoriamente o nono bit a 0, enquanto as restantes estações voltam à tarefa em curso sem nenhuma perturbação adicional. A troca de informação que se segue com a estação "acordada" não perturba nenhuma das outras pois o nono bit de todos os caracteres seguintes é 0 o que não provoca a interrupções de outras estações "adormecidas".

Quando o microcomputador central pretende enviar informação para todas as estações remotas de aquisição ("broadcast"), comporá uma "flag" específica que acordará todas as estações.

Para o envio de informação de uma estação para o microcomputador convencionou-se um valor fixo (55 hexadecimal), por o computador nesta fase não possuir capacidade em "hardware" para o 'protocolo do nono bit'.

II - Nº de bytes.

Neste campo é recebida (ou transmitida) a informação do número de bytes de todo a UC.

A limitação de 74 bytes presentemente em vigor, (embora já tenham sido efectuada experiências bem sucedidas com 138 bytes), garante o funcionamento da estações remotas recorrendo, apenas, à memória interna do microcontrolador para guardar a informação de cada "frame" transacionado.

Prevê-se brevemente analisar a possibilidade de recorrer à utilização de UCs com comprimento ilimitado (ou pelo menos bastante superior); esta passo vai no sentido de utilizar **poucas** comunicações com comprimento **elevado**, ou seja, considerando a comunicação através de "datagrams".

III - Cabeçalho.

O campo do cabeçalho está previsto com o objectivo de, por um lado garantir a ressincronização entre receptor e emissor e, por outro lado, permitir futuramente a troca de informações de controlo de rede entre os "pares comunicantes".

IV - Numeração.

Ao ser enviada uma mensagem de um nó para outro, dividida em "frames", é necessário não perder o controlo da sequência como estes são enviados e recebidos; o campo de numeração permitirá garantir este controlo através da sequenciação dos "frames".

Ao ser transmitido um comando, o campo de numeração indica o número de ordem do comando enviado; nas respostas, este campo indica qual o "frame" que foi reconhecido ter chegado em boas condições em último lugar. Com este método é possível garantir que nenhuma comunicação será perdida mesmo que os reconhecimentos de boa recepção sejam efectuados para um grupo de comandos/respostas recebidos e não um a um.

Tradicionalmente esta numeração é efectuada em módulo 8, isto é, a numeração transmitida e/ou recebida está compreendida entre 0 e 7. Com isto é possível garantir a transmissão de um máximo de 8 "frames" sem receber o reconhecimento de *boa recepção* por parte do receptor. Eventualmente uma mensagem com não mais de 512 bytes poderá ser transmitida necessitando apenas de um reconhecimento de boa recepção ("acknowledge") por parte do receptor.

O valor do módulo poderá ser modificado pelo "software" que apelar aos recursos desta camada, podendo assim ser adaptado às características impostas à comunicação.

V - Tipo

Este campo identifica o comando ou resposta no "frame" que está a ser enviado ou recebido.

Foram concebidos quatro grupos de "frames", cada um com um reportório de comandos e/ou respostas possíveis:

Grupo	**Nome do grupo de "frames"**	**Comandos**	**Respostas**
A	Controlo de Ligação	ON, RR , RNR e OFF	RR e RNR
B	Informação	DATA	DATA
C	Supervisão		ACK, NACK
D	Excepção	EXCEPT	

Os "frames" do grupo A controlam a ligação (destacando-se as fases de estabelecimento de uma ligação lógica e fim dessa ligação) entre emissor e receptor. Neste tipo de "frames" **nunca existe campo de informação**; é frequente englobarem-se

neste grupo os comandos e respostas de pedido de informação ao outro nó bem como a resposta que informa não ter capacidade para neste instante receber uma comunicação.

Os "frames" do grupo B implementam as trocas de informação efectivas entre os "pares" comunicantes. Nestes **existe sempre um campo de informação**, não sendo efectuado qualquer controlo da ligação estabelecida.

Os "frames" do grupo C efectuam a supervisão da ligação em curso, nomeadamente no que diz respeito à correcta ou incorrecta recepção de "frames". **Nunca existe campo de informação** neste tipo de comandos e respostas.

Os "frames" do grupo D são relativos a comunicações extemporâneas, originárias tipicamente (mas não exclusivamente) das estações remotas de aquisição de dados. Este tipo de comunicação é prevista para garantir que o funcionamento, muito embora seja ainda do tipo "master/slave", não impeça que uma ocorrência excepcional, necessitando de uma actuação de emergência, seja atendida. Estes "frames" **contêm sempre campo de informação**, e não necessitam de reconhecimento, pelo menos ao nível desta camada do módulo de comunicações.

VI - Endereço.

O campo de endereço permite identificar a quem se dirige e qual a procedência de uma determinada comunicação.

Neste campo o primeiro byte representa o destinatário e o segundo byte o remetente. O microcomputador central terá sempre um endereço fixo.

Embora a "flag" forneça já informação sobre o destinatário do "frame", é útil a existência de um campo de endereço com o remetente.

Quando o "master" pretende enviar um comando a todas as estações (comando de "broadcast"), compõe o campo de endereço com um determinado valor, combinado entre todas as estações.

VII - Informação.

O campo de informação é o que contém toda a informação a transmitir de um ponto para outro.

Este campo, embora presentemente limitado a 64 bytes de dados, pode vir a ter qualquer número de bytes; além disso, a informação que compõe este campo não necessita estar formatada em subcampos, nem possuir uma codificação particular (ASCII ou outra).

Apesar destas considerações note-se que cada byte de informação é codificado em 8 bits a que se acrescenta 1 bit de início ("start bit") 1 ou 2 de fim ("stop bits") e 1 respeitante ao protocolo de nono bit já explicado.

De realçar que determinados "frames" poderão não conter um campo de informação.

VIII - Erro.

O campo de erro é constituido pelo "sumcheck" em complemento para 2 de todos os bytes constituintes do "frame" **com excepção da "flag"**.

Este campo permitirá detectar a ocorrência de alguns erros durante a transmissão que obriguem a uma eventual retransmissão do "frame".

7.2.2 Elementos de Procedimento.

Os elementos de procedimentos são elementos base para a definição das regras que regem a ligação. São elementos de procedimento a numeração/sequenciação de "frames" que constituem uma mensagem, e os comandos/respostas possíveis de utilizar. Podemos ainda considerar como um elemento de procedimento o tempo permitido para a ocorrência da resposta a um pedido de informação do microcomputador ou o reconhecimento de um "frame" enviado.

A sequenciação de "frames", dependendo directamente do campo de numeração do "frame", só é tomada em conta relativamente a "frames" com campo de informação; este facto não quer dizer que os restantes não possuem esse campo. Pelo contrário são os "frames" que procedem à supervisão da ligação previamente estabelecida que enviam/recebem a numeração que permite o controlo da numeração destes "frames".

Considerando que é possível, por "software", alterar o número de "frames" transmissíveis sem reconhecimento (8 é o valor mais vulgar), bem como o tempo limite para o aparecimento de uma resposta de reconhecimento de *boa recepção* do receptor do "frame", é possível adaptar estas características às condições de funcionamento da linha em cada instante, garantindo a máxima eficiência nas transacções.

De realçar que a numeração dos "frames" do grupo D (de Excepção) é independente da numeração restante (grupo B e grupo C). Esta última diz apenas respeito à sequenciação dos "frames" do grupo B transaccionados, numeração que será mantida completamente independente da numeração dos "frame" de excepção.

Para garantir o acesso a uma primitiva que contenha informação sobre o tempo que está a decorrer, torna-se obrigatório a existência de um relógio, em funcionamento em paralelo com a unidade de processamento.

Uma das fatias mais importantes dos elementos de procedimento são os comandos/respostas utilizáveis; foram implementados os seguintes comandos e respostas:

Os "frames" definidos no tipo A são:

Comandos:

ON - Estabelece a ligação entre o "master" (microcomputador central) e o "slave" (estação remota de aquisição).

RR - Microcomputador central informa estar à espera de uma resposta;

RNR - Microcomputador central informa não estar pronto a comunicar.

OFF - Termina a ligação anteriormente estabelecida;

Resposta :

RR - Estação informa estar pronta mas não tem mais informação para enviar;

RNR - Estação informa não poder enviar a informação requisitada.

Os "frames" definidos no tipo B são:

Comando :

DATA - "frame" com informação no campo;

Resposta:

DATA - "frame" com informação no campo;

Os "frames" definidos no tipo C são:

Resposta:

ACK - Informa que recebeu "frame" correctamente;

NACK - Informa que recebeu "frame" incorrectamente;

Os "frames" definidos no tipo D são:

Comando:

EXCEPT - "Frame" de excepção que pede atenção. Informação complementar sobre quem pede atenção e porquê não cabem nesta camada do protocolo. No entanto essa informação deverá ser transmitida no campo de informação. Como já foi referido este comando não necessita de reconhecimento. Para além disso é possível enviar uma mensagem a todas as estações de aquisição por parte do microcomputador central.

7.2.3 Classes de Procedimento.

As classes de procedimento definem o tipo de funcionamento e o enquadramento dos comandos e respostas para a efectivação clara de uma transacção.

No sistema actualmente implementado considerou-se o funcionamento normal como sendo "master/slave" embora esteja prevista a possibilidade dos "slaves" tomarem a iniciativa de enviar uma comunicação. Esta comunicação (comando EXCEPT), é a única que não necessita de um reconhecimento de boa recepção por parte do destinatário, cabendo ao remetente prever da boa ou má recepção por parte do destinatário.

As comunicações não excepcionais a partir do microcomputador serão sempre definidas por três etapas principais:

- **Estabelecimento da ligação**, inicia a *conversa* entre dois nós.
- **Efectuar as transacções**, *efectiva* a *conversa* (troca de "frames") entre dois nós.
- **Terminar a ligação**, acaba a *conversa*.

De realçar que, apesar da comunicação ser do tipo "master/slave" não é necessário que esteja terminada a ligação entre o microcomputador central e uma estação para efectuar a transacção de mensagens para outras estações. O microcomputador pode, em qualquer instante, decidir estabelecer uma ligação com **outras** estações remotas de aquisição de dados (processo similar ao de ter diversos ficheiros abertos em simultâneo), efectuando troca de informações com uma de cada vez, sem necessitar de terminar a comunicação com as outras.

As regras para a utilização destes comandos e respostas descrevem-se num capítulo apropriado e versam, quer o encadeamento dos comandos com as respostas admissíveis, quer as situações geradoras de erros; estas últimas podem ocorrer por escoamento de um certo tempo sem ser detectada uma resposta, ou por ter 'retentado' uma transacção de dados mais vezes do que um certo número limite. Para ser possível a detecção de 'ultrapassagem' de um limite temporal sem receber uma resposta esperada, existe um relógio em funcionamento em simultâneo com a unidade de processamento.

Estes valores limites de controlo de erro, são passados da camada de programação superior. Apresentam-se nesse capítulo os algoritmos usados na implementação destas primitivas de comunicação até ao nível da implementação, quer no microcomputador central quer nas estações remotas de aquisição de dados. Em secções posteriores serão explicitados alguns aspectos relacionados com a implementação.

7.3 Camada de Controlo de Mensagem.

A camada de controlo de Mensagem usa as primitivas acessíveis da camada imediatamente inferior, nomeadamente:

- Estabelecimento da ligação, entre uma estação local e a unidade central de processamento.
- Transacção de uma comunicação entre uma estação local e a unidade central.
- Terminação desta ligação.

Esta camada de 'software' é independente quer do tipo de microcontrolador em uso na placa local, quer da programação da camada inferior; isto é, quer o protocolo de controlo de ligação seja orientado ao bit (como no caso do HDLC e SDLC) e implementado pelo microcontrolador 8044, quer seja orientado ao byte como no caso presentemente em uso, as primitivas disponíveis serão as mesmas. Assim, o resultado da comunicação a efectuar será o mesmo em qualquer caso, se excluirmos o tempo despendido na efectivação da transacção da informação.

Os módulos de programação implementados para cumprir outros requisitos do sistema, nomeadamente os módulos de processamento de sinal e os módulos de detecção automática de avarias, recorrem a primitivas definidas nesta camada.

Estas primitivas permitem a efectivação de trocas de mensagens, programas, parâmetros de controlo e gestão de funcionamento das estações locais e **funcionam concorrentemente** com os próprios módulos. Com isto garante-se a detecção de toda a informação relevante sem contudo obrigar, quer as estações locais, quer a unidade central a despender tempo útil de processamento à espera que a linha de comunicação esteja disponível para a continuação da transacção.

PARTE 3.

DESENVOLVIMENTO.

Nesta fase do texto serão expostas em pormenor as opções tomadas a nível de desenvolvimento dos diversos módulos ("hardware/software") do sistema do regularímetro digital.

Esta exposição, dividida em 3 capítulos, expõe os diversos passos empreendidos para:

- a construção da estação remota de aquisição de dados;

- a implementação dos programas residentes nas estações locais, em que se dá um particular ênfase ao módulo de comunicações e ao módulo operativo;

- o desenvolvimento da programação no microcomputador central ou estação mestra, em que se apresentam as opções de implementação de módulos tão diversos quanto o de comunicação e o de processamento digital de sinal.

Capítulo 8. Projecto de "hardware" das estações remotas.

A utilização do sistema desenvolvido, baseado nas necessidades impostas pelo processo de fabrico de fio têxtil, aponta para a necessidade de captar sinais analógicos em locais afastados entre si. Esta imposição apresenta como melhor solução a opção por uma arquitectura em rede das estações de aquisição de dados.

Prevendo a utilização das estações remotas para algumas tarefas de controlo da máquina a que está ligada, e sendo necessário garantir que a falha de uma estação ou do computador central não provoque perturbação em nenhuma das outras optou-se, para estas, por uma arquitectura baseada num microcontrolador. A distribuição de inteligência pelas diversas estações garante o seu funcionamento autónomo em caso de ocorrência de anomalias em qualquer outra ou na rede.

Para a comunicação entre as diversas estações de aquisição e o computador central, utilizou-se um protocolo físico de comunicação económico, necessitando apenas de um par de fios entrançados, o RS-485.

Dado o seu baixo preço escolheu-se para computador central um sistema compatível com o IBM-AT. As capacidades gráficas destes sistemas permitem a apresentação de relatórios bastante completos, nomeadamente no que diz respeito aos resultados dos módulos de processamento de sinal.

Por outro lado a velocidade de processamento dos sistemas microcomputadores tipo IBM-AT é suficiente para a maioria das utilizações, sendo ainda simples e relativamente económico aumentá-la recorrendo ao uso de co-processadores (matemáticos, de "input/output", etc.) ou de placas aceleradoras.

Outra significativa vantagem da escolha deste microcomputador reside na existência de um grande número de placas adicionais que o permitem tornar ainda mais versátil. De entre essas placas destacam-se nomeadamente as de comunicação, que permitem garantir elevadas velocidades de transferência de dados e o funcionamento em rede, usando mesmo algumas destas o protocolo físico RS-485.

As estações remotas de aquisição de dados (ver figura 42) são inteligentes tendo já uma boa capacidade de processamento de informação. Têm a possibilidade de adquirir informação com uma frequência máxima de 10 KHz bem como diversas saídas analógicas e digitais, utilizáveis quer para controlo, quer para acesso ao exterior.

As estações permitem a introdução de "software" por parte do utilizador, provisoriamente (para RAM) ou *semi-definitivamente* (para E^2PROM). Existem

residentes, em EPROM, diversos módulos que garantem o seu funcionamento e controlo, bem como a tomada de acções sobre o dispositivo que está a controlar.

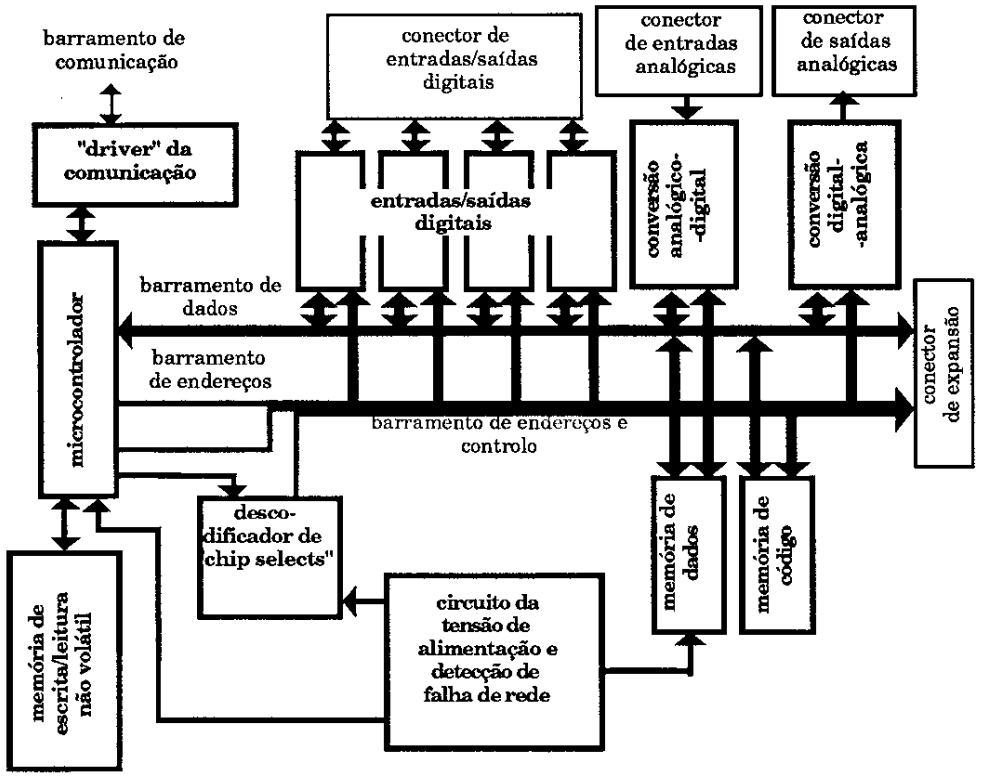


Figura 42. Diagrama de blocos das estações remotas construídas.

As estações têm como unidade de processamento um microcontrolador intel da família 51 (8051, 8052 ou 8044) . Possuem ainda memória tipo RAM ou E^2PROM, 16 entradas analógicas, 2 saídas analógicas, 88 entradas/saídas digitais, um relógio de tempo real autónomo, um interface para um "display" LCD de 2 linhas de 40 caracteres.

Esta estações foram implementadas numa placa de 22 por 10 cm (tamanho EUROCARD estendido), embora possa ser utilizada apenas uma área com tamanho EUROCARD normal caso seja dispensável a utilização de entradas/saídas digitais.

Toda a concepção da placa baseou-se no pressuposto da sua utilização diversificada, razão pela qual, para além de dispor de "performances" superiores às necessidades, estão ao dispor do utilizador diversos sinais de controlo que permitirão implementar em pequenas placas adicionais, quer dispositivos específicos (p. ex.

relés, interfaces, adaptadores de sinal, etc.), quer dispositivos com características mais apropriadas para uma determinada aplicação e que permitem eliminar o uso de alguns dos constantes na placa (p. ex. a implementação de um ou mais conversores A/D com tempo de conversão ou número de bits mais favoráveis).

8.1 Escolha do microcontrolador.

Como exposto na primeira parte deste trabalho, foram estudadas duas arquitecturas típicas para a implementação das estações de aquisição de dados: uma utilizando um processador muito rápido especialmente vocacionado para o processamento de sinal; outra utilizando um microcontrolador. Nesse estudo concluiu-se pela vantagem da utilização do microcontrolador 8051 (ou 'derivados' como o 8052 ou 8044).

A opção pelo microcontrolador referido baseou-se ainda em outros aspectos técnicos sobre as suas possibilidades que se resumem de seguida.

- Comunicação série programada internamente:

Quer o microcontrolador 8051/52, quer o 8044 possuem módulos de comunicação internos.

O microcontrolador 8051/52 tem um módulo de comunicação assíncrono programável (UART), enquanto o 8044 [7] possui um módulo interno de comunicação mais complexo que implementa o protocolo de comunicação síncrono orientado ao bit, o SDLC. Para utilizações em que a velocidade de transferência de dados através da rede de comunicação é mais crítica a utilização deste último melhora significativamente os tempos de comunicação.

- Interrupção automática com a chegada de uma unidade de comunicação:

A chegada de uma comunicação para o microcontrolador (um caracter no 8051/52 ou um "frame" no 8044), pode 'acordá-lo', por intermédio de uma interrupção. Esta facilidade permite facilmente implementar uma estrutura mestre/escravo ("master/slave"), como se depreende da exposição já efectuada.

- Possuem RAM interna (128 bytes a 256 bytes):

Esta memória, com baixo tempo de acesso quando comparada com a externa, (as instruções que acedem à memória interna demoram metade do tempo da externa) contém todos os registos. Conforme o tipo de endereçamento é directo ou indirecto, é

possível em certos 'descendentes' do 51 (como o 8052) aceder a memória interna ou registo especiais ("Special Function Registers) com o mesmo endereço.

- Têm 32 entradas/saídas digitais que também implementam os barramentos:

As entradas/saídas digitais implementam os barramentos de endereço, dados e controlo; as restantes linhas disponíveis podem ser utilizadas como entradas ou saídas, tendo a particularidade de poderem ser acedidas na forma de byte ou bit (cada uma é acedida individualmente).

- Memória de dados e código independentes:

Este microcontrolador separa a zona de código da zona de dados, ao contrário das máquinas tradicionais em que o espaço de endereçamento da memória de dados e código é o mesmo, permitindo assim duplicar o espaço de memória disponível; apesar desta característica, é deixado ao critério do utilizador a possibilidade de sobrepor **toda** ou **parte** da memória por forma a ser acedida quer como memória de código quer como memória de dados.

- Contêm "timers"/contadores internos:

Estes "timers" permitem controlar o intervalo entre acontecimentos, bem como gerar um "baud rate" programável; esta utilização é especialmente eficaz dado não obrigar a nenhuma intervenção da unidade de processamento.

- Possuem um poderoso conjunto de instruções:

Esse grupo de instruções subdividem-se essencialmente em 2 tipos: orientados ao byte e orientadas ao bit; de notar que este último tipo de instruções, na altura pouco frequente nos microprocessadores, é de grande interesse para aplicações de controlo, simplificando a programação e reduzindo os tempos de execução.

As instruções necessitam, tipicamente, de 1 ou 2 μs para a sua execução, incluindo-se neste grupo de instruções as funções lógicas de bit e byte. Inclui ainda instruções de multiplicação/divisão em 4 μs.

8.2 Arquitectura geral das estações.

Dado o barramento de dados ser multiplexado com os 8 bits menos significativos dos endereços, houve necessidade de os desmultiplexar. Por outro lado, prevendo a possibilidade de se pretender expandir os barramentos, foram consideradas as possibilidades de colocar "drivers" nos barramentos (figura 45); como para a

utilização de todos os recursos da estação sem recorrer a expansão tal não é necessário, optou-se por uma solução que os dispensa, embora esteja prevista a sua posição no circuito e estejam previstas todas as ligações necessárias.

8.2.1 Comunicação.

Estando presentemente a ser utilizado um microcontrolador 8051/8052, foi implementado um protocolo de comunicação orientado ao caracter.

Para garantir um funcionamento em rede, com uma topologia tipo barramento, em que o microcontrolador central é o 'mestre' e as estações são escravas, foi especificado um protocolo que utiliza as facilidades concedidas pelo ***protocolo do nono bit*** implementado internamente no microcontrolador; este protocolo será posteriormente descrito em pormenor no capítulo correspondente ao "software".

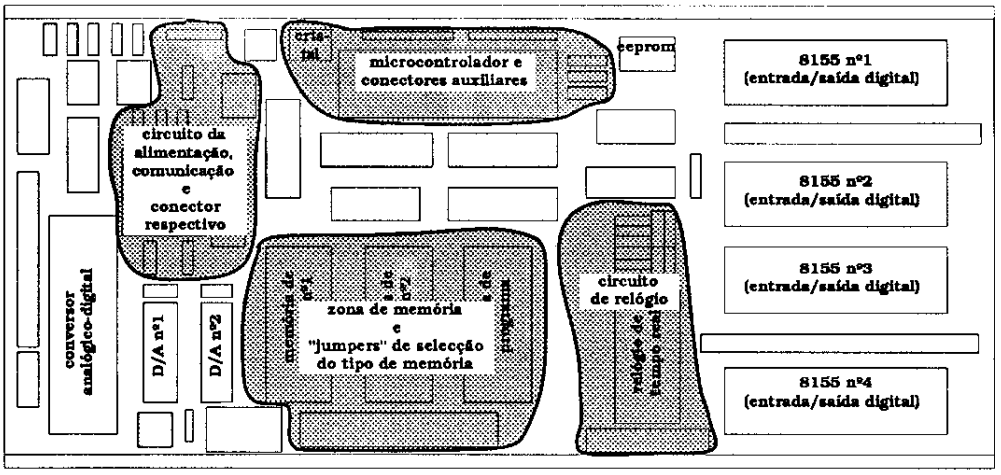


Figura 43. Esquematização da planta da 'placa' das estações remotas de aquisição de dados com individualização de alguns blocos principais: microcontrolador, memórias, relógio e circuitos da 'alimentação' e "drivers" da comunicação.

O protocolo físico, RS-485, é implementado na estação remota por intermédio de um "driver" de 8 pinos (4*2) 75176 (figura 43). Nesta 'pastilha' foram efectuadas as ligações que permitem o funcionamento quer utilizando um 8051/52 quer um 8044; alimentados apenas a 5 volts permitem um interface simples e eficaz do barramento com as linhas de comunicação.

8.2.2 Aquisição de dados.

A aquisição de dados sobre as características do fio produzido é uma das funções que necessariamente tem de ser efectuada ao nível das estações remotas de aquisição de dados (figura 44).

Embora o número de canais necessários não tenha um valor fixo, *podendo mesmo ser suficiente um canal*, optou-se por atribuir a cada estação a possibilidade de utilizar até 16; o nº real de canais a utilizar varia bastante, dependendo das necessidades do sistema a gerir.

Optámos pela utilização do conversor NEC 8016 (ou 8017) para as estações remotas tendo em conta quer as suas características e "performance" quer o seu custo. Este conversor de aproximações sucessivas tem um tempo de conversão de aproximadamente 100 μs e permite a utilização de 16 canais analógicos através de um "*multiplexer*" analógico interno.

O 8 "bits" da conversão destes conversores A/D, são suficientes para a utilização prevista; realce-se que um NEP (botão) é tipicamente considerada quando excede em 200% ou mais o valor médio da espessura do fio. É possível assim obter uma discriminação considerável, tendo além disso em conta a qualidade do método de captação do sinal.

Apesar destas considerações não se inviabilizou a possibilidade de recorrer a um conversor de maior resolução; uma alternativa passaria pela substituição do circuito integrado do conversor por uma pequena placa (*na realidade um híbrido ou "ASIC"*). Outra alternativa consiste na utilização das possibilidades de expansão existentes nestas placas (quer as directamente associadas à conversão quer as dos barramentos do processador e "chip selects" já descodificados), recorrendo aos conectores com o exterior .

Este circuito integrado dispõe, internamente, de um "multiplexer" analógico (MUX) e um conversor A/D; para o funcionamento normal é necessário 'ligar' exteriormente a saída do MUX à entrada do conversor.

Dado estarem acessíveis a saída do "multiplexer" analógico e a entrada do conversor é possível utilizar um circuito de retenção do sinal ("sample & hold"), que está previsto na placa. Não sendo necessário, basta fazer uma ligação extra (curto-circuito entre a entrada e saída do "sample & hold").

Pode-se ainda utilizar a entrada do conversor propriamente dito para, ligando outros canais, aumentar o número de canais a converter; esta solução obrigaria à utilização de circuitos adicionais.

A saída que indica ***fim de conversão*** está directamente ligada a uma entrada que **pode** provocar uma interrupção do microcontrolador.

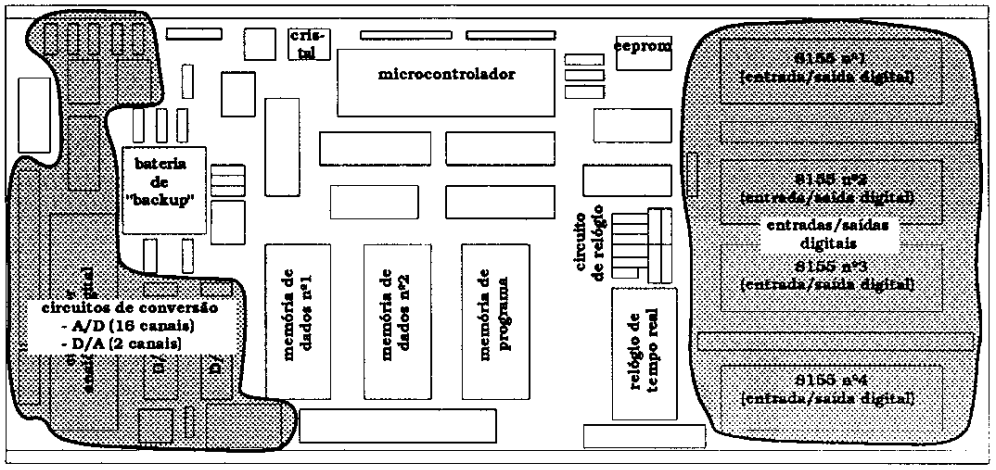


Figura 44. Apresentação da zona responsável pelas entradas/saídas analógicas e digitais da 'placa' das estações remotas.

8.2.3 Saídas analógicas.

Foram implementadas na placa da estação remota 2 canais analógicos de saída (figura 44). Estes canais são implementados por 2 conversores digitais/analógicos NEC 7011, e são utilizados para gerar um sinal correspondente à autocorrelação de um sinal analógico de entrada, visualizável num osciloscópio, como se discute num capítulo posterior (programação nas estações remotas).

Construindo um pequeno circuito adicional é possível usar estes conversores para gerar a tensão de referência do A/D, sendo esta assim alterável por "software".

8.2.4 Entradas/Saídas digitais.

As estações remotas dispõem de 88 linhas de entrada/saída digitais. Estas linhas são conseguidas utilizando 4 pastilhas Intel 8155 de 40 pinos. Esta

possibilidade, permite adquirir informação sobre o funcionamento das máquinas envolvidas no processo de fabrico.

Por outro lado, considerou-se importante a construção de uma pequena consola destacável autorizando um operador a aceder localmente às estações. É por intermédio destas linhas digitais que é feito o "drive" e recolhida a informação do teclado destacável, pertencente à consola portátil. Com este teclado será sempre possível garantir o controlo da estação mesmo que não exista, ou esteja danificada, a ligação em rede.

Foi directamente ligado ao barramento, utilizando algumas linhas adicionais de controlo, um "display LCD", visor que constitui, com o teclado, a consola portátil. No presente exemplo foi utilizada um "display" GALAXY de 80 caracteres por linha e 1.5 ou 2 linhas, das quais são visíveis permanentemente 40 caracteres de cada linha (figura 45).

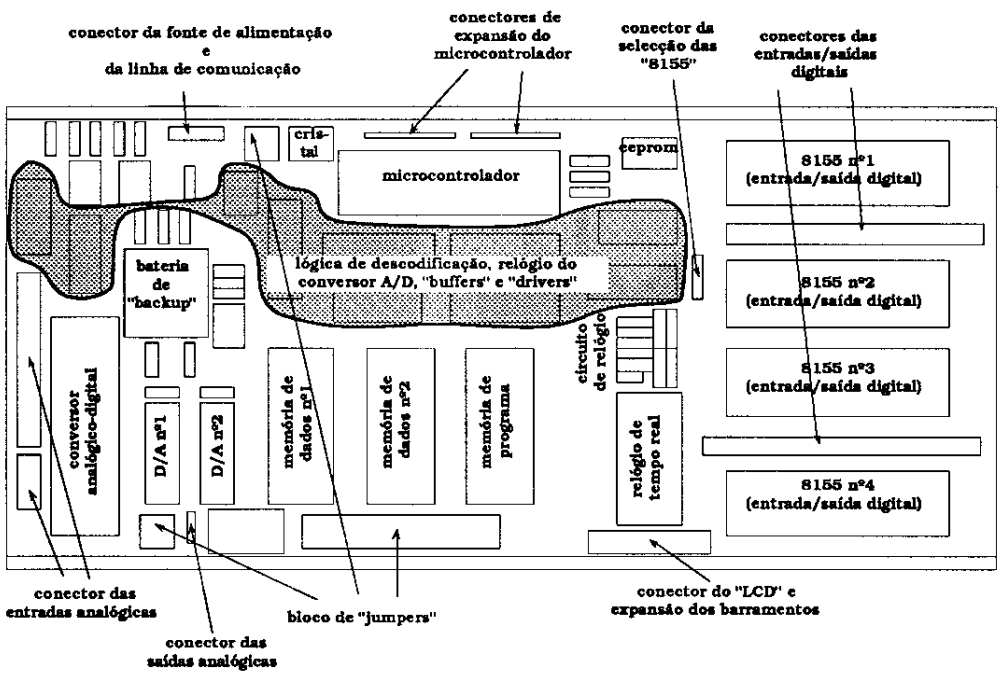


Figura 45. Apresentação da zona com os conectores disponíveis; identifica-se ainda a zona da 'placa' responsável pela descodificação de endereços e lógica diversa (incluindo "buffers" e "drivers").

8.2.5 Alimentação das componentes da estação remota.

A estação remota recebe a alimentação através de um conector de 5 pinos, também utilizado para a comunicação com o exterior.

Destes 5 pinos, uns fazem a ligação à rede de comunicação (2 pinos) e os outros a uma fonte de alimentação (3 pinos). Esta fonte de alimentação poderá ser comum a todas ou a um grupo de estações remotas.

Estas 3 linhas incluem, para além da alimentação propriamente dita que poderá ser de 8 a 15 volts DC, uma linha que indica a eminente falha de tensão. Esta falha de tensão terá de ser detectada, como é habitual, na fonte de alimentação, provavelmente pela falha de um semi-ciclo da rede de alimentação de 220 volts AC. Um sinal nessa linha desencadeará, por interrupção, uma série de acções, inclusivamente a desinibição do acesso à memória RAM, que embora tradicionalmente volátil, estará alimentada por uma bateria. Esta bateria garante o funcionamento de dispositivos indispensáveis, nomeadamente o relógio de tempo real, toda a memória RAM e a memória interna do microcontrolador.

Esta montagem permite efectuar um "power-on reset": é gerado um sinal de "RESET" quando é aplicada a tensão da fonte de alimentação.

8.2.6 Memórias disponíveis, e outros recursos.

As estações possuem RAM (eventualmente alimentada por bateria), visível para o microcontrolador como memória externa (até 64 Kbytes), que pode ser substituída no todo ou parcialmente por memória E^2PROM. De realçar que, muito embora os microcontroladores utilizados, separem a memória de código da memória de leitura/escrita é permitido, por construção, que a memória de escrita/leitura (RAM) seja carregada com código e este seja posteriormente executado (ver figura 46)..

A memória de código disponível (tipicamente uma EPROM), pode ter diversos 'tamanhos' (de 8 a 64Kbytes), utilizando apenas **um** componente.

Quer a selecção do tamanho das memórias de dados e de código, quer a sua organização, são decididas por intermédio de um banco de selecção constituído por "jumpers".

Está ainda disponível nas estações remotas uma E^2PROM de 256 bits organizada sobre a forma de 16 palavras de 16 bits. Permite guardar dados importantes para o microcontrolador na altura do arranque, sendo actualmente utlizada para guardar a

informação de qual o microcontrolador existente, qual a velocidade de comunicação vigente ("baud rate"), qual o valor da frequência do cristal, etc.

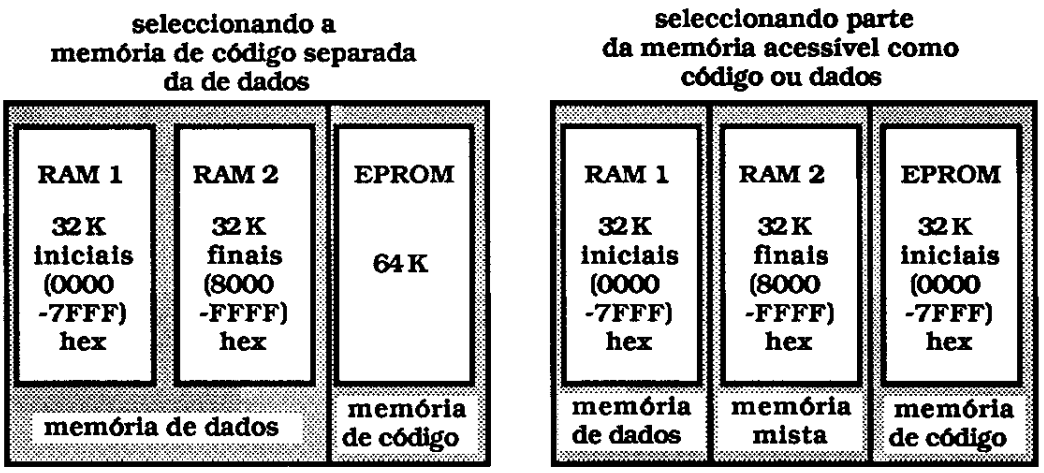


Figura 46. Organização da memória de código e de dados conforme se pretende ou não um acesso partilhado a parte dela; esta selecção é efectuada por intermédio de "jumpers".

As estações remotas possuem para além das já referidas, as seguintes características:

- Com vista a permitir alguma expansão, estão acessíveis ao exterior as 8 linhas de "chip select" das 8155, do LCD e 3 suplementares não utilizados, para além dos barramentos de dados e endereços multiplexados, diversos sinais de controlo como o ALE, $\overline{WR}$ e $\overline{RD}$.

- Possui um relógio de tempo real que permite datar todos os acontecimentos. Este relógio é autónomo, não necessitando da intervenção do microcontrolador para o seu funcionamento. Este relógio está alimentado pela bateria em caso de falha da alimentação.

8.2.7 Mapeamento da memória e periféricos.

Todos os periféricos atrás referidos são acedidos pelo microcontrolador através de instruções de escrita e leitura de memória, uma vez que o microcontrolador escolhido funciona apenas em modo "memory mapped". Para seleccionar o

dispositivo ou memória que se pretende aceder, garantido a possibilidade de aceder a 64 Kbytes de RAM, foi necessário utilizar alguns bits de uma porta do 8051 para seleccionar qual a memória ou periférico a aceder; podemos assim dizer que o barramento de endereços é constituído por 18 bits e não os tradicionais 16 gerados pela unidade de processamento.

É necessário garantir que duas linhas da porta estejam com os valores correctos (bit 2 e 5 da porta 1 do microcontrolador); este facto fica a dever-se à necessidade de prever a desinibição do funcionamento da memória e do descodificador de "chip selects" quando, por exemplo, se verifica uma quebra da alimentação.

Capítulo 9. Projecto de "software" das estações remotas.

Considerações gerais.

O desenvolvimento da programação a instalar nas estações remotas passou pela definição dos módulos necessários ao funcionamento global do sistema, isto é, quais os programas a desenvolver (ex: determinação de espectros, diagnóstico de avarias, etc.) e o local onde devem residir (estação mestre ou estações remotas). Apresenta-se um diagrama destes módulos, e da sua interligação, na figura 47.

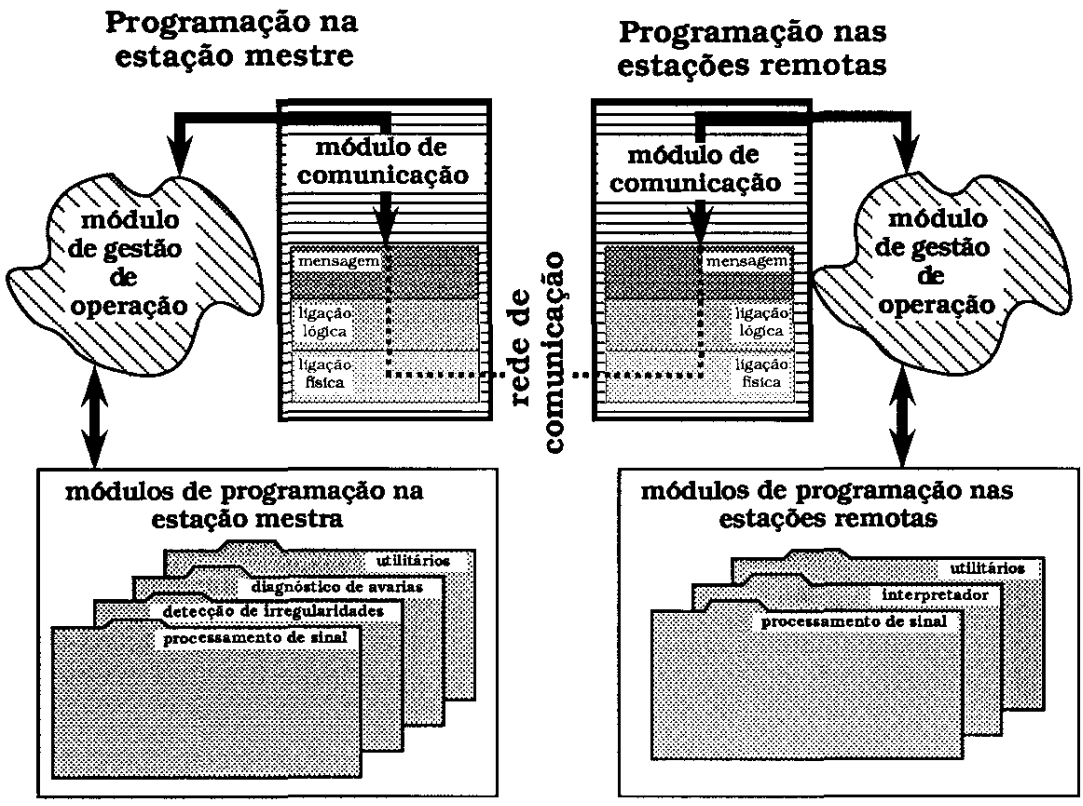


Figura 47. Módulos de programação desenvolvidos e sua distribuição entre microcomputador central e estações remotas.

Apesar das considerações efectuadas não foi inviabilizada a existência de programas similares (nomeadamente o da determinação de espectros) nos dois tipos de processadores (estações remotas e microcomputador central); a opção por esta dualidade ficou a dever-se a três factores primordiais, directamente relacionados com

o tempo de processamento, o tempo despendido na transmissão de um grande volume de dados e na *possibilidade de se garantir o funcionamento de certos nós apesar da eventual inoperacionalidade de outros nós e/ou da rede.* A título de exemplo refira-se o cálculo do coeficiente de variação:

- falhando a estação central é possível calcular o CV% nas estações remotas;

- sendo necessário dispor de uma transmissão de dados para a estação central, é vantajoso em termos de tempo efectuar este cálculo no microcomputador central;

- não sendo necessário efectuar qualquer transmissão da informação recolhida para a estação central, é mais eficaz proceder ao cálculo nas estações remotas *e eventualmente enviar os resultados.*

É da responsabilidade da estação remota (figura 47) a transmissão de dados necessários à estação central; como mencionado anteriormente (capítulo referente à concepção da comunicação), e se especificará em pormenor posteriormente, esta comunicação de dados só se deve efectuar a pedido do microcomputador.

Para efectuar a gestão de todos os módulos implementados é necessário dispor de um gestor a que chamaremos módulo operativo.

O "software" desenvolvido, para além de gerir toda placa, inclui ainda módulos de: processamento rápido da Transformada de Fourier (FFT), determinação da autocorrelação, gestão da comunicação e um interpretador. Este interpretador permite ao computador central ou a um utilizador local carregar um programa de controlo através da linhas de comunicação, que pode ser modificado em qualquer altura.

Em resumo, foram desenvolvidos os seguintes módulos (usando o "assembler" do microprocessador 8051 e o PLM51, uma linguagem de mais alto nível):

- **Módulo operativo.** Garante a interligação entre todas as tarefas em curso na estação; garantem ainda o interface com a consola portátil.

- **Módulo de processamento de sinal.** Módulo que, com base em informação adquirida, recebida da rede ou gerada internamente, permite determinar o espectro de Fourier, o coeficiente de variação e o desvio da média; calcula ainda a autocorrelação de um canal de entrada do A/D.

- **Módulo de comunicação.** Módulo que garante a ligação da estação ao microcomputador central.

- **Módulo do interpretador.** Módulo que permite executar, listar e inserir um programa de controlo escrito numa linguagem de mais alto nível.

Foram definidas duas soluções diferentes na interligação destes módulos, cujas distinções se devem no essencial a duas filosofias diferentes para a comunicação

através da rede partilhada; enquanto na 1ª solução se optou por considerar a rede de comunicação como sendo um terminal virtual não inteligente, na segunda solução optou-se por considerar que todas as comunicações se regem por um protocolo de controlo de ligação, orientado ao caracter.

Estas duas soluções corresponderam também a duas fases distintas deste projecto; a primeira foi necessária para efectuar o desenvolvimento e experimentação dos diversos programas, em que é indispensável efectuar testes sucessivos de apenas um dos módulos, interactivamente, portanto sem grandes preocupações quanto ao funcionamento da comunicação. Acrescente-se ainda o facto de na fase inicial do trabalho não existir ainda a rede de comunicação mas sim uma estação remota ligada ao microcomputador central. Posteriormente, com a rede já em funcionamento, um utilizador a trabalhar no microcomputador central, mesmo que ligado por um barramento a várias estações, comunica com ***uma delas*** de cada vez tal como ao utilizar um terminal assíncrono directamente ligado à estação.

A segunda solução permite um significativo aumento do rendimento ao serem minimizados muitos tempos mortos da comunicação; por outro lado, ao não permitir um funcionamento caracter a caracter tão simples, dispensando a obrigatoriedade de um operador permanente, obriga a um funcionamento muito 'automatizado'.

Um dos aspectos que também surgiu facilitado na 1ª solução foi o uso da consola destacável, que se torna mais difícil ao optarmos pela 2ª.

As soluções consideradas, a que chamaremos **estrutura 1** e **2**, são assim caracterizadas por:

Estrutura 1.

A comunicação do microcomputador central com cada estação remota de aquisição, embora efectuada através da rede partilhada, situa-se ao nível da simulação de um terminal pelo primeiro.

Apesar do nível do protocolo estabelecido ser bastante baixo, garantem-se já as tarefas fundamentais, tais como:

- Estabelecer ligação do microcomputador central com a estação;
- Comunicar com a estação;
- Enviar ficheiro para a rede;
- Capturar ficheiro da rede;
- Terminar a ligação do microcomputador central com a estação

O estabelecimento de uma ligação com uma estação é efectuado utilizando o protocolo do nono bit, o que garante que todas as estações para quem a comunicação

não é dirigida permaneçam desligadas e não sejam interrompidas por toda a comunicação que se efectuará entre a estação a ligar e o microcomputador central.

As possibilidades de capturar ficheiros e enviar ficheiros através da linha de comunicação corresponde, na realidade, à implementação de uma camada do módulo de comunicação de nível superior.

Esta estrutura encontra-se em funcionamento à mais de 2 anos, *ininterruptamente*, tendo se mostrado eficiente especialmente como método de efectuar experimentações e testes sobre o suporte físico do sistema distribuído.

Estrutura 2.

Nesta estrutura, mais elaborada e mais complexa, a comunicação entre cada estação remota e a estação central é efectuada segundo um protocolo bem definido, traduzida pela troca de "*pacotes*" de informação (chamados "frames" ou Unidades de Comunicação) e não de caracteres um a um.

Agora o acesso a qualquer tarefa por parte da estação é efectuada através da camada de controlo responsável pela troca de **mensagens**, razão pela qual é denominada de *camada de controlo de mensagem*; para enviar ou receber uma mensagem, o módulo implementado tem de executar alguns passos, similares aos da estrutura 1, para garantir o estabelecimento de ligação entre os nós mestre e escravos, a efectivação da transacção e o terminar da ligação. Em geral, qualquer programa que necessite de recorrer à comunicação recorrerá às primitivas de troca de mensagem existentes (*muito embora esteja autorizado o recurso às primitivas de nível mais baixo para certos casos*).

Apesar destas duas estruturas serem diferentes, mantêm-se as possibilidades de aceder aos recursos definidos na primeira por parte da segunda estrutura (o inverso não é possível); podemos dizer que esta última é um *super-conjunto* da primeira estrutura, ou melhor, a primeira é um subconjunto da segunda. As principais diferenças residem na interligação entre os módulos operativos dos dois tipos de nós que, como veremos posteriormente não se traduz numa alteração de grande vulto; todos os restantes módulos de programação serão acessíveis quer optando por uma quer optando por outra.

9.1 Módulo operativo.

Este módulo permite ao utilizador aceder a todos os recursos físicos da estação remota de aquisição de dados e eventualmente executar o módulo interpretador.

Numa primeira fase foram implementados diversos comandos acessíveis, quer através da rede de comunicação, quer através da consola portátil que liga às entradas digitais das estações remotas.

Cabe ao módulo operativo aceder a todas as primitivas disponíveis (tal como acontece com um sistema operativo ao aceder ao "BIOS" e " BDOS") para implementar, quer estes comandos, quer a comunicação com o nó central. Este módulo garante ainda o interface com o utilizador a nível de consola portátil (seja qual for a estrutura utilizada) bem como através de um eventual terminal, que na prática é a rede (correspondendo ao microcomputador central na estrutura 1).

Os comandos podem ser associados em diversos grupos conforme as suas funções; estes são essencialmente derivados da comunicação, aquisição de dados e processamento de sinal sobre os dados recolhidos. Existe ainda um conjunto de comandos genéricos (utilitários) que foram desenvolvidos quer para o teste de todas as capacidades das estações, quer com vista a facilitarem o teste ("debug") do "hardware" e "software" destas.

A passagem para a estrutura 2 faz com que a execução destes comandos a partir da rede (leia-se estação central) seja modificada, mas não impõe praticamente qualquer restrição suplementar; assim, os comandos deixarão de funcionar de uma forma interactiva, necessitando de um interface mais complexo entre a camada superior do módulo de comunicação e este módulo; a nível imediato, a execução dos comandos deixará de ser efectuada por selecção de um caracter seguida da resposta a perguntas, dependentes do comando seleccionado, passando para o envio de uma linha de comandos sem qualquer resposta como se exemplifica na figura 48.

Estrutura 1 (Em itálico as respostas da estação e em normal as do utilizador)
end. inicial ? 80 <enter>
end. final ? 93 <enter>
constante ? 11 <enter>

Estrutura 2
D 80,93,11

Figura 48. Exemplificação da alteração de um comando ("fill" da memória interna) ao ser alterada a estrutura do módulo operativo.

De realçar que esta modificação **não é indispensável** mas corresponde a um passo na direcção da minimização dos custos de programação e do tempo gasto na

comunicação; na realidade, a existência de uma rede de comunicação no caminho entre um terminal assíncrono e um computador não obriga nem a modificar o programa nem ao utilizador 'decorar' um conjunto de comando complexos. Acresce ainda a facilidade de poder executar, a partir do microcontrolador central, comandos em "batch". Conforme a tarefa para que estão destinados assim se podem associar estes comandos em grupos distintos (figura 49).

```
Comandos de Visualização/Edição de memória:
            C - Visualização de memória de código
            D - Visualização/Edição de memória interna
            E - Visualização/Edição de memória EEPROM
            X - Visualização/Edição de memória externa
            I - Inserção de informação na memória externa
            O - Compactação dados na memória
            P - Visualização/Edição das portas do microcontrolador
            Q - Mover zonas de memória externa

Comandos de Processamento de sinal:
            J - Cálculo da autocorrelação
            L - Determinação de CV% e U%
            S - Geração um sinal dente de serra
            U - Determinação de espectros

Comando do interpretador:

            * - Iniciar o interpretador

Comandos de controlo da estação:
            G - Executar um programa
            N - Modificar o "baud rate"
            Y - Executar "batch" de comandos
            Z - Mudar consola de saída
            .  - Termina ligação à rede

Comandos associados à Aquisição de dados:
            A - Adquirir informação de N canais e calcula média
            K - Adquirir um bloco de dados de um canal
            M - Adquirir informação de N canais
            V - Adquirir informação de 1 canal

Utilitários:
            ? - Mostrar ecrã de ajuda
            B - Executar operações (tipo calculadora)
            F - Mudar formato da saída
            H - Mostrar horas e data
            R - Acertar relógio
            T - Testar da memória
            W - Escrever na consola de saída
```

Figura 49. Grupos de comandos e caracteres que os desencadeiam usando a estrutura 1.

Os comandos iniciam-se pela recepção na estação de um caracter proveniente quer da rede quer da consola portátil no caso da estrutura 1 e, no outro caso, pela recepção de um comando completo (ocupando um ou mais "frames"). Na estrutura 1

será posteriormente necessário fornecer dados complementares, respondendo a questões levantadas pelo módulo operativo. Apesar destas diferenças, os comandos têm funções idênticas, pelo que continua a manter relevância uma descrição das tarefas associadas a cada comando (figura 50).

Comandos de Visualização/Edição de memória.

C - Visualiza memória de código seguindo o formato dos "dump" tradicionais. Este comando não permite alterar qualquer código, mesmo que parte dele resida em RAM; para esse efeito o utilizador terá de utilizar o comando X.

D - Visualiza e/ou modifica valores da RAM interna. Quer a visualização, quer a modificação é feita usando endereçamento indirecto pelo que, por este processo não é possível aceder a nenhum registo interno de controlo (como TCON, SCON, etc.).

E - Visualiza e/ou modifica os valores da E^2PROM de 256 bits. Nesta memória são guardadas informações relativas ao "baud rate", microcontrolador utilizado e o cristal que gera os sinais de relógio do microcontrolador. Os restantes valores disponíveis podem ser utilizados como o utilizador pretender.

I - Insere em memória informação recebida no formato intel através da linha de comunicação série (ou a partir do teclado auxiliar). Por indicação do operador é possível introduzir esta informação num periférico ou memória externa auxiliar mesmo que resida fora da placa mas seja actuada por um "chip select" de um periférico já descodificado.

O - Compacta dados guardados em memória a partir do endereço 0000. Cada byte resultante passa a ser obtido pela média de X bytes contíguos. Quer o endereço final da compactação, quer o valor de X são introduzidos pelo utilizador. O espaço de memória ocupado pelo resultado terá o tamanho igual ao espaço inicial dividido por X.

P - Modifica o valor de qualquer das "portas" do microcontrolador. Dado as portas 0 e 2 estarem ocupadas (implementam os barramentos de dados e endereço) só é possível alterar as portas 1 e 3.

Q - Move um bloco de memória de uma zona de memória para outra zona de memória inferior ou superior, mesmo sendo reentrantes.

X - Este comando permite visualizar e alterar qualquer posição de memória externa. Para além de podermos ver o conteúdo de uma memória externa, é possível preencher uma zona de memória com um valor constante. Por opção permite-se que seja colocado esse valor, apenas se o valor presente estiver entre 2 limites introduzidos na altura (incluindo os limites). De notar que dado a arquitectura da estação ser "Memory mapped" é possível com este comando aceder a diferentes periféricos.

Comandos associados à Aquisição de dados.

A - Adquire e apresenta os valores da média de 16 valores de todos ou alguns dos canais do conversor analógico-digital, conforme a selecção efectuada pelo utilizador. Se a consola de saída de resultados for o "display" LCD, e para a estrutura 1, apresenta continuamente a média de novos valores obtidos, funcionando como um voltímetros de diversos canais.

K - Adquire um conjunto de bytes de um canal do conversor analógico-digital, num máximo de 32 Kbytes. A frequência com que é adquirido cada byte é introduzida pelo utilizador.

M - Apresenta os valores de todos ou alguns dos canais do conversor analógico-digital.

V - Apresenta o resultado hexadecimal da conversão de uma amostra recolhida num canal do conversor analógico-digital.

Figura 50. Lista dos comandos disponíveis nas estações remotas.

Comando do interpretador.

* - Executa o módulo do interpretador.

Comandos de Processamento de sinal.

J - Calcula a autocorrelação de um canal do conversor analógico-digital, enviando o resultado por um canal do conversor digital-analógico para visualização em osciloscópio, sendo utilizado o 2º canal do D/A para gerar o sinal de sincronismo.

L - Calcula o valor de CV% (coeficiente de variação) e o U% (desvio médio do erro absoluto) de informação existente em memória, valores que são tipicamente utilizados na indústria de fiação têxtil.

S - Envia para um canal dos conversores digital-analógico uma onda tipo dente de serra em que o incremento do valor digital pode ser variado. Este comando tem essencialmente funções de teste.

U - Calcula a transformada rápida de Fourier (F.F.T.). O algoritmo permite a utilização de janelas (rectangular, Hanning e Hamming), diversos números de pontos (256 a 8192 valores) e permite o cálculo da transformada inversa de Fourier. Este algoritmo permite o calculo da transformada de Fourier de sinais complexos de 8 Kbytes no máximo de 30 segundos. Este algoritmo é bastante mais rápido que um algoritmo idêntico implementado no computador central, embora apresente a grande desvantagem de ter uma precisão limitada.

Comandos de controlo da estação.

G - Executa um programa a partir de um endereço.

N - Altera o "baud rate" utilizado na comunicação série, conforme o microcontrolador (8051 ou 8052) e o cristal utilizados. Estes valores serão guardados na memória não volátil para serem utilizados na programação da comunicação que é desencadeada como resposta ao RESET.

Y - Executa, sem interface com o utilizador, um conjunto de comandos ("batch"). Isto permite libertar a linha de comunicação para o computador central servir outra estação, enquanto a estação vai efectuando tarefas em "batch". Estão previstos como comandos executáveis em batch os seguintes:
D, E, F. G, K, L, N, O, P, Q, R, U, W, X, Z e .
Estes comandos, exceptuando o comando de escrita (W), não enviam qualquer caracter para a rede nem consola portátil.

Z - Permite ao computador central modificar a consola para o "display" LCD. Com esta possibilidade é possível, depois de estabelecida a ligação e pese embora manter-se a possibilidade de receber comandos a partir da rede, enviar mensagens para a consola portátil, isto é, o "display LCD".

. - Desfaz a ligação lógica entre o microcomputador central e a estação remota.

Utilitários.

? - Apresenta um ecrã de ajuda ao utilizador.

B - Executa expressões algébricas de 32 bits, como a raiz quadrada e divisão, e converte um valor hexadecimal para decimal, apresentando o resultado.

F - Modifica o formato (decimal ou hexadecimal) dos valores numéricos a introduzir como resposta a pedidos decorrentes dos comandos seleccionados, ou os apresentados ao utilizador como resposta ao comandos pedidos.

H - Apresenta a data e hora do relógio de tempo real.

R - Modifica a data e hora do relógio.

T - Testa a RAM total e apresenta o número de bytes disponíveis em memória externa, quer na memória principal (até 64 Kbytes) quer na memória auxiliar (4*256 bytes das 8155 + 50 bytes do relógio de tempo real).

W - Permite escrever uma mensagem no "display" LCD, mesmo que esteja estabelecida uma ligação através da rede.

Figura 50 (*Continuação*). Lista dos comandos disponíveis nas estações remotas.

O módulo operativo inicia a sua execução, como resposta ao RESET, portanto mal se liga a estação remota. Verificou-se que esta solução permite o funcionamento em rede bastante simples em que a **intervenção do operador é indispensável** através de comandos a enviar pela rede; todo o interface entre o controlo a efectuar pelo μC central e as estações locais é assegurado por acção directa do operador. Eventualmente esse operador executará numa determinada estação remota um comando. Pode ainda recorrer ao comando "BATCH" das estações locais (comando Y) para executar, sequencialmente, um conjunto de comandos .

Numa versão posterior (associada à Estrutura 2) optou-se por um encadeamento diferente das acções deste módulo; enquanto na primeira, em resposta ao RESET, se iniciava o módulo operativo funcionando como monitor interfaceando com um terminal não inteligente, nesta segunda opção estará residente em E^2PROM uma indicação de qual a acção a tomar como resposta ao RESET. Caso não tenha sido dada nenhuma indicação iniciar-se-á o módulo operativo; em caso contrário, a partir da informação residente na E^2PROM, iniciar-se-á um programa do utilizador ou um módulo diferente.

Assim, após a primeira instalação do sistema, esta memória passará a guardar o endereço de execução de resposta ao RESET, indicando também se este endereço corresponde a um programa imediatamente executável ou um programa a ser interpretado.

Saliente-se mais uma vez que a opção pela Estrutura 1 ou 2 só depende da complexidade do sistema e do nível de desenvolvimento do projecto em curso, não obrigando em qualquer caso à modificação global do módulo operativo. No presente, utilizou-se o funcionamento definido pela Estrutura 1, pela facilidade de desenvolvimento e teste dos algoritmos, pela simplicidade do algoritmo de comunicação necessário implementar e pela facilidade da sua utilização no diagnósticos de eventuais anomalias do "hardware".

Como veremos na secção relativa ao módulo de comunicação, foram implementadas já as suficientes primitivas de comunicação com vista à utilização da segunda estrutura.

9.2 Módulo de processamento de sinal.

Tal como foi referido anteriormente foram implementados, nas estações remotas de aquisição de dados, os seguintes algoritmos de processamento digital de sinal:

- Determinação do Coeficiente de Variação e Desvio da Média;
- Determinação de espectros com base na Transformada de Fourier **[31, 32]**;
- Cálculo da Autocorrelação.

Nos pontos seguintes serão analisados com mais pormenor quer as particularidades da implementação, quer o tipo de resultados e funcionamento dos comandos referidos.

9.2.1 Determinação do Coeficiente de Variação (CV%) e Desvio da Média (U%).

O valor do Coeficiente de Variação associado ao valor de Desvio da Média estão bastante divulgados na indústria têxtil, sendo ambas medidas do desvio das amostras relativamente à sua média; estas medidas, desenvolvidas como uma das funções estatísticas tradicionais, tem a grande vantagem de fornecer estes resultados independentemente das unidades das amostras, concretamente sob a forma de uma percentagem da média; a expressão que permite calcular estes parâmetros é:

$$CV\% = \frac{\text{desvio padrão}}{\text{média}} 100\% \quad U\% = \sum \frac{|\text{cada amostra - média}|}{\text{média}} 100\%$$

$$\Leftrightarrow$$

$$CV\% = \frac{\sqrt{\frac{\left(\sum_{i=1}^{N} x_i^{\,2}\right) - \overline{x}}{N-1}}}{\overline{x}} *100\% \qquad U\% = \sum_{i=1}^{N} \frac{|\, x_i - \overline{x} \,|}{\overline{x}} *100\%$$

em que x_i representa o valor de cada amostra e $\overline{x}$ é a média dessas amostras.

A implementação realizada na estações remotas utiliza os dados residentes na zona de memória de ***ram externa*** a partir do endereço 0000_{hex} e com comprimento até $7FFF_{hex}$ (32 Kbytes); é possível também efectuar o cálculo do coeficiente de variação função de um comprimento *l*. Para esse efeito é necessário que cada amostra x_i corresponda a uma média de amostras reais, de tal forma que *l* corresponda a esse número de amostras vezes o comprimento do sensor; para conseguir obter estes valores de CV_l seria necessário recorrer ao comando de compactação implementado no módulo operativo.

Apresentam-se no quadro 6 alguns dos tempos despendidos no cálculo destes parâmetros.

$2048 \geq N \geq 256$	t< 0.6 segundos (*dificilmente mensurável)*
N = 4096	t = 0.6 segundos
N = 8192	t = 0.9 segundos
N = 16384	t = 1.4 segundos
N = 32768	t = 2.5 segundos

em que N representa o número de amostras utilizadas no cálculo.

Quadro 6. Tempos despendidos nas estações remotas para efectuar o cálculo do CV% e U%.

O cálculo é efectuado com precisão de 32 bits (4 bytes para cada valor desde o início ao fim), por forma de garantir uma boa precisão já que, nunca ocorrendo "overflow", não é excedido nenhum limite que obrigue a uma truncagem ou aproximação.

Além dos resultados de CV e U, apresentados sob a form de percentagens (% de desvio relativamente à média), é ainda fornecido ao utilizador o valor inteiro da média do conjunto das amostras.

Não existem dificuldades especiais no cálculo quer do U% quer no do (*desvio padrão*)^2, podendo mesmo ser calculados sem agravar consideravelmente o tempo de processamento, uma vez que à medida que seja efectuada a aquisição será possível acumular dois valores para obter o $\sum x_i$ e o $\sum x_i^2$ que permitirão calcular respectivamente a média e o desvio padrão.

Existe no entanto um cálculo, **a raiz quadrada**, que se torna mais complexo e obriga à utilização de um algoritmo mais 'rebuscado'; assim, são percorridos quatro passos fundamentais no cálculo da raiz quadrada **[31]**:

- normalização do valor inicial **x**,
- pesquisa numa tabela da raiz quadrada do byte mais significativo,
- interpolação do valor em falta,
- normalização do valor final *(inversa à normalização efectuada no 1º passo).*

No primeiro passo é modificado o valor **x** por forma a que o byte mais significativo **não seja 0**; o segundo passo corresponde a uma pesquisa numa tabela de 256 pares de valor-raiz quadrada qual a raiz quadrada do byte mais significativo.

Por fim, basta efectuar, com base nos bytes restantes e nos valores de tabela seleccionados no ponto anterior, uma interpolação linear e posteriormente refazer a normalização já efectuada, agora em sentido contrário.

Este algoritmo foi primeiramente simulado, utilizando uma linguagem de alto nível, com vista a quantificar os erros esperados, devidos essencialmente à interpolação a efectuar; os resultados obtidos, com erros muito inferiores a 1%, tornam-o perfeitamente aceitável. A tabela de 256 bytes não é de dimensão exagerada, sendo mesmo possível considerar a hipótese de, pretendendo um cálculo mais preciso, alargar esta tabela.

Refira-se por fim que o valor de raiz calculado é sempre um inteiro de 2 bytes, em que se garantiu a maior precisão possível nos cálculos intermédios. Ao ser efectuado o cálculo dos parâmetros (CV% e U%) não existe contudo essa limitação dado garantir-se que não existe "overflow" em nenhum cálculo; além da raiz quadrada só a divisão pode acrescentar alguma imprecisão.

Dada a necessidade de minimizar o tempo de processamento, este programa foi implementado em "assembler", partindo da utilização quer de um conjunto de rotinas e "macros" implementadas no início do trabalho, quer de uma especificação inicial cuidada.

9.2.2 Determinação do espectro pela Transformada de Fourier (FFT)

O cálculo do espectro do fio, como explicado, permite, para além da determinação de características construtivas, detectar eventuais imperfeições periódicas provocadas por um processamento incorrecto na unidade fabril (ver parte de **concepção** neste trabalho).

O algoritmo desenvolvido é aplicado directamente sobre dados guardados na memória de dados externa utilizando como base o algoritmo discutido e apresentado na parte de concepção.

Os dados referentes ao sinal de entrada estão armazenados em duas zonas de memória distintas, uma que contem as amostras correspondentes à parte real do sinal e outra com a parte 'imaginária'; dado o sinal de entrada ser temporal e derivado de um processo meramente físico, estaremos a trabalhar sem parte 'imaginária' o que nos permite utilizar a zona de memória da parte 'imaginária' para acelerar o algoritmo, guardando nessa zona N amostras adicionais. Calculando o espectro deste sinal com ambas as componentes, é possível determinar posteriormente o resultado final de um conjunto de 2N amostras, combinando parte real e imaginária dos resultados obtidos.

Se pretendermos efectuar o cálculo da transformada inversa (determinar o sinal temporal a partir do espectro), podemos usar este algoritmo dado que a única

diferença, *aliás prevista no programa*, é a utilização ou não de uma divisão por N (= número de amostras).

É possível utilizar janelas de Hanning e Hamming (além da rectangular) para operar directamente sobre o canal evitando erros tradicionais inerentes à amostragem e minimizando os efeitos associados a uma truncagem não múltipla do período do sinal em análise.

Apresentam-se no quadro 7 os tempos despendidos no algoritmo implementado para a determinação do espectro afectado do factor N^{-1} (N = nº de amostras utilizadas no cálculo) e sem o referido factor; estas duas alternativas existem para o caso de se pretender efectuar o cálculo da tansformada inversa.

	Janelas utilizadas					
Número de pontos	**Rectangular**		**Hanning**		**Hamming**	
	*** N^{-1}**	***1**	*** N^{-1}**	***1**	*** N^{-1}**	***1**
8192	30.6	29.0	33.0	31.4	32.8	31.8
4096	5.0	14.3	15.6	14.9	15.7	15.0
2048	7.0	6.8	7.4	7.2	7.5	7.1
1024	3.4	3.4	3.5	3.5	3.6	3.4
512	1.7	1.7	1.8	1.7	1.8	1.7
256	0.8	0.8	0.9	1.0	0.9	1.0

sendo N o número de amostras a utilizar no cálculo.

Quadro 7. Tempo (em segundos) gasto no processamento do espectro (FFT) nas estações remotas.

Este aspecto do factor N (nº de amostras) ser ou não considerado, embora desprezado por alguns autores tem alguma razão de ser; assim, considerando a definição da tranformada discreta de Fourier (DFT)

$$X(n) = \sum_{k=0}^{N-1} x(k)\, e^{-j\frac{2\pi kn}{N}}$$

verifica-se ser necessário considerar um divisor por N, como se constata quase intuitivamente na determinação de X(0), a componente contínua, em que não deve ser obtido por $X(0) = \sum_{k=0}^{N-1} x(k)$, mas sim por $X(0) = \frac{1}{N}\sum_{k=0}^{N-1} x(k)$, correspondendo assim ao valor médio de todas as amostras.

A informação adquirida é guardada em dois vectores, cada um com o limite máximo de 8192 (8 K) valores; cada amostra é representada por um conjunto de 2 bytes com um bit (o mais significativo) representando o sinal. Se utilizarmos o maior número de pontos possíveis despenderemos 32 Kbytes de memória para efectuar o cálculo (8192 valores * 2 bytes por valor * 2 vectores: real e complexo).

Embora na aquisição de dados se digitalize cada amostra em 8 bits, optámos por utilizar 2 bytes para ser possível garantir uma melhor precisão do cálculo e, por outro lado ser permitido utilizar o mesmo algoritmo e os mesmos vectores para o cálculo da transformada inversa de Fourier.

Os 2 bytes que representam cada amostra não são guardados contíguos com vista a evitar a necessidade de processar a informação adquirida, sempre no sentido de poupar tempo; assim, os 1ºs 8 k bytes contêm os bytes menos significativos das diversas amostras (tal como são recolhidas), seguindo-se os 8 K mais significativos e, por processo análogo seguem-se as amostras correspondentes à parte 'imaginária'.

Como as operações sobre os valores, em cada uma das fases de cálculo (o nº de fases = $\log_2(N) + 1$), é executado no local, isto é, nas posições de memória dos vectores iniciais, não é necessário 'gastar' muito mais memória; naturalmente será indispensável recorrer a vários registos internos para efectuar os cálculos.

Muito embora o algoritmo a utilizar não seja novidade, investiu-se bastante tempo na sua optimização, com particular relevância para a minimização do seu tempo de processamento; para atingir este objectivo, não se recorreu a nenhuma linguagem de alto nível do 8051 (PL/M ou C) disponíveis na altura, utilizando-se sempre o "assembler" como aliás já tinha acontecido para o cálculo do CV%.

Esta preocupação dominante obriga, em compensação, a um uso intensivo de memória, quer de código executável quer de tabelas. Neste sentido optou-se pelo uso preferencial de "macros" *relativamente* a sub-rotinas, dado estas últimas consumirem mais recursos de tempo. É também neste sentido que se optou pela existência de uma tabela com o valor do seno e cosseno ao invés de efectuar o cálculo destas funções. Assim, implementaram-se diversas "macros" e construiu-se uma tabela de funções sinusoidais, que ocupa 10 Kbytes de memória, correspondendo aos valores de um seno entre 0 e $(2 + \frac{1}{4})\ \pi$, que naturalmente permitem aceder quer aos valores do seno quer do cosseno; embora fosse possível efectuar a determinação do seno e cosseno com uma tabela de dimensões mais reduzidas (bastaria uma tabela do seno de 0 a $\frac{\pi}{2}$), optou-se

pela utilização de uma tabela de maiores dimensões no sentido de minimizar os tempos de processamento.

9.2.3 Cálculo do sinal de Autocorrelação.

A autocorrelação é um sinal temporal, cujo espectro corresponde, em módulo, ao quadrado do espectro (espectro de potência) de um determinado sinal.

Ao ser efectuada a autocorrelação de um sinal, pode geralmente observar-se por simples inspecção qual é a componente sinusoidal de maior peso no sinal, método usado durante muito tempo por, apesar de ser um cálculo um pouco lento, ser mais simples e directo de efectuar que o cálculo da transformada de Fourier.

Contudo, dado os algoritmos actualmente existentes, este tipo de cálculo tornou-se pouco prático e com uma relação 'custo'/resultados bastante fraca. De realçar ainda que, possuindo um resultado nas frequências, seria fácil obter o resultado temporal da autocorrelação a partir do espectro, procedendo ao cálculo do quadrado do módulo e aplicando a respectiva transformada inversa (pese muito embora os tempos de processamento agravados).

Apesar destes contras, o cálculo da autocorrelação é bastante mais rápido que o método precedente (elevar ao quadrado e calcular a transformada inversa), especialmente se apenas pretendermos obter uma informação sobre a frequência de maior peso (neste caso qual a fibra constituinte do fio que apresenta maior massa); a implementação deste cálculo permitiu, além disso, confirmar a ineficiência do método quando comparado com a determinação espectral já discutida.

Tendo em consideração estas premissas e dado ter-se optado pela implementação do presente algoritmo, a principal questão a colocar prendia-se com a forma de apresentar os resultados.

Uma possibilidade, rapidamente abandonada, seria a de enviar pela consola ou pela rede os bytes correspondentes à correlação. Como é facilmente previsível, dado o caracter temporal e portanto *sem fim* do resultado, a quantidade de informação seria muito elevada e de difícil 'digestão', sendo ainda difícil extrair conclusões válidas sem um cálculo adicional ou sem o recurso a um operador. Se pretendermos proceder ao cálculo relativo a apenas um segmento, dadas as características pouco auspiciosas dos resultados, seria preferível enviar os dados para o microcomputador central e aí efectuar o cálculo.

No sentido de rentabilizar o algoritmo, foi decidido considerar apenas a hipótese que apresenta o resultado da autocorrelação digital de um certo nº de amostras em

contínuo, ou seja, são permanentemente apresentados os resultados temporais de N amostras; esta solução acarretando o problema de *como dispor* da informação resultante obrigou a considerar um tipo diferente de "saída de dados". Utilizam-se os dois canais do conversor D/A para a apresentação dos resultados de uma "forma" analógica, visualizável por meio de um osciloscópio.

Assim enquanto um canal envia os sucessivos valores da autocorrelação, para um segmento de N amostras, o outro canal envia um sinal de sincronismo que é actuado sempre que se inicia o envio desse conjunto de resultados. Com esta metodologia é possível visualizar num osciloscópio o resultado, **temporal**, da autocorrelação sincronizando-o pelo canal que gera o sinal de sincronismo.

Apontando claramente para a utilização deste algoritmo em aplicações muito específicas, é conveniente ter presente que os resultados deste método só permitem uma análise qualitativa dos resultados por parte de um operador.

Este programa foi implementado usando a linguagem PL/M-51.

9.3 Módulo de comunicação.

O módulo de comunicação implementado nas estações remotas permite a interligação destas com o microcomputador central através da rede de comunicação.

Como já foi referido, foram implementadas duas soluções diferentes para a comunicação, relacionadas directamente com as duas estruturas definidas anteriormente na apresentação do módulo operativo.

Estrutura 1:

Uma em que, estando estabelecida a ligação entre o microcomputador central e uma determinada estação, a comunicação circula através da rede como se estivessemos perante uma ligação com um terminal 'estúpido'; nesta metodologia há a possibilidade de ser efectuada a captura da informação circulante para um ficheiro do microcomputador e, eventualmente, ser enviado o conteúdo de um ficheiro para a rede de comunicação. De realçar que neste caso toda a iniciativa a tomar estará, directa ou indirectamente, a cargo do operador.

O estabelecimento de uma ligação é efectuada enviando um conjunto de caracteres de controlo, em que o primeiro deles tem o nono bit activo (utilizando a interrupção do microcontrolador associada ao nono bit). Todas as estações serão interrompidas pela recepção deste caracter que, no caso de corresponder ao seu código interno levarão a estação a considerar que a comunicação lhe é dirigida; além desta

condição é necessário que os caracteres de controlo seguintes (em número de 3 mas com o nono bit inactivo) sejam os esperados, caso que fará com que a estação passe ao estado de ligado.

A partir desta altura, todos os comandos digitados pelo operador serão executados pela estação que estiver ligada, sendo as respostas encaminhadas para a rede de comunicações.

Existe ainda um comando, **da responsabilidade do operador**, que terminará esta ligação (comando '.' referido no módulo operativo).

Dado o carácter "master/slave" da rede de comunicação, o estabelecimento de uma ligação pelo microcomputador central obrigará a estação remota a enviar os resultados, não para a consola portátil mas para a rede; além deste aspecto, a estação passará a responder a todos os pedidos de execução de comandos por parte do mestre. Apesar disto, considerou-se que, em determinadas circunstâncias pode não interessar que a estação seja interrompida de **imediato** após a recepção do pedido de estabelecimento da ligação; em contrapartida há situações em que essa interrupção é indispensável.

Consideraram-se assim duas possíveis situações: uma em que o microcomputador central interrompe a estação qualquer que seja a tarefa em curso, ficando na situação de *espera de comandos*; outra em que a estação é interrompida para modificar a consola de saída (que passa a ser a rede), passando a enviar toda a informação para a rede, mas não interrompendo o algoritmo em curso. Cabe ao operador na estação central decidir qual das opções pretende tomar; para além desta tarefa é sua responsabilidade gerir a ligação estabelecida.

Estrutura 2:

A segunda solução implementada, utiliza o protocolo apresentado na parte de concepção, funcionando para sistemas (entenda-se por sistema o conjunto microcomputador, rede e estações) em que existe uma necessidade de garantir melhores tempos de processamento ou se pretende libertar o operador das tarefas de gestão básicas que estão a seu cargo na primeira solução.

Esta tarefa, processada em "background" (concorrentemente com outras), embora não prejudicando gravemente o programa em curso exige o máximo cuidado, com vista a garantir que não há tempos críticos a ser excedidos.

O algoritmo do módulo de comunicação foi baseado na especificação de diagrama de estados da figura 51.

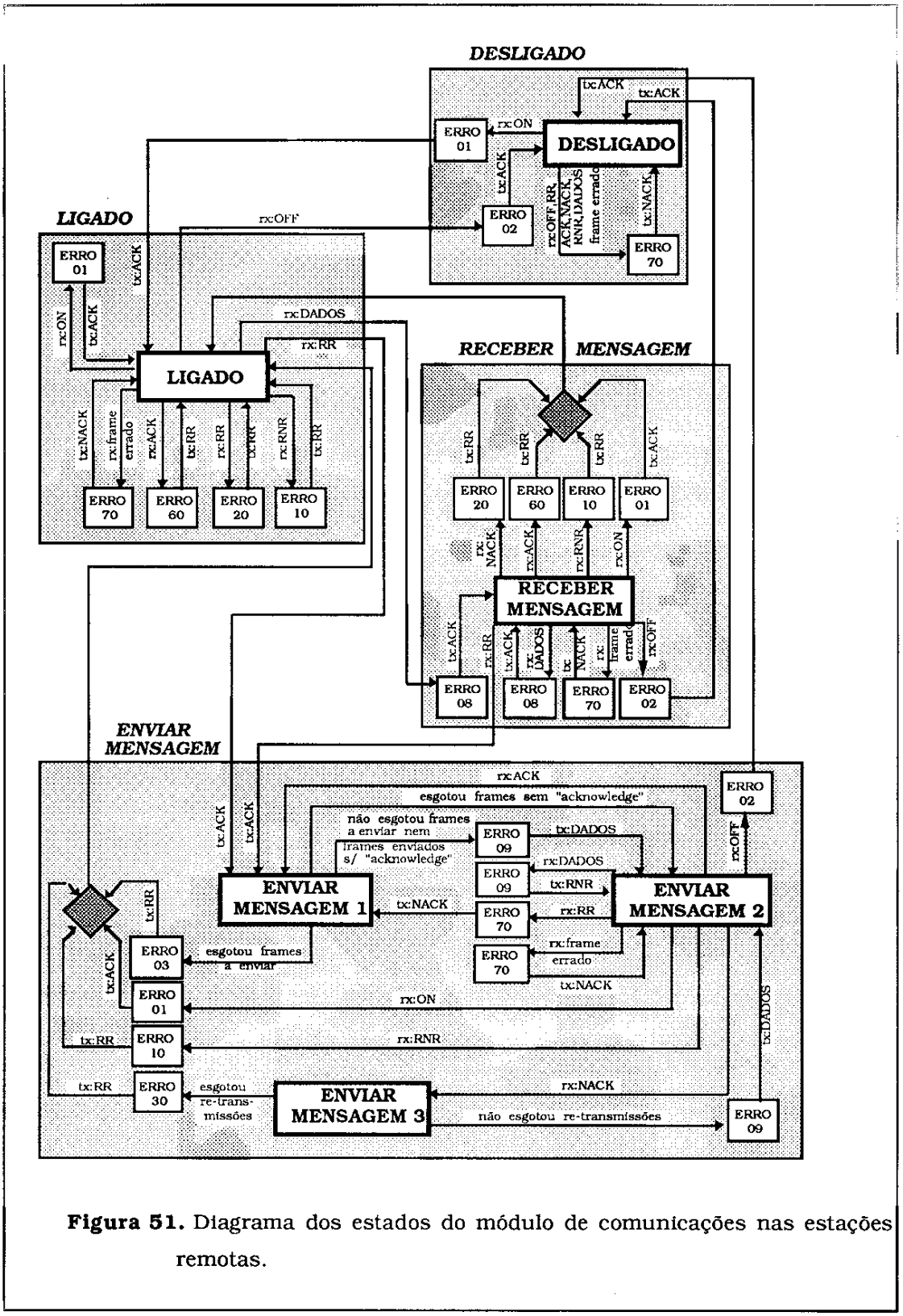


Figura 51. Diagrama dos estados do módulo de comunicações nas estações remotas.

De notar que alguns dos 'sub-estados' (erro 1, erro 2, etc.) não correspondem obrigatoriamente a situações de erro, dependendo do contexto em que se inserem para poderem ser considerados como sucesso ou erro; note-se ainda que o sistema não possui uma 'entrada' propriamente dita, necessitando da chegada de um comando de estabelecimento da ligação (ON) para se iniciar o processo associado à comunicação.

Segue-se a descrição dos algoritmos que implementam algumas das primitivas deste protocolo.

9.3.1 Algoritmos das primitivas de comunicação, utilizando o protocolo orientado ao caracter.

Tal como para o microcomputador central também existe um algoritmo específico para a comunicação das estações remotas. Como estas são escravas, isto é, **não têm direito a estabelecer nem terminar qualquer ligação**, apenas podem responder às iniciativas de comunicação tomadas pela estação central. Exceptuam-se as comunicações extemporâneas que, como já foi referido, não possuem qualquer protocolo de verificação do sucesso da comunicação.

Para as comunicações vulgares (não extemporâneas), apesar das estações serem escravas, mantêm-se as 3 etapas principais para a efectivação de uma comunicação: *estabelecimento de ligação*, *efectuar transacção* e *terminar ligação*. No entanto as estações nunca podem tomar a iniciativa de efectuar ou terminar a ligação e apenas podem enviar informação sob pedido de quem estabeleceu a ligação.

Como as estações apenas respondem a pedidos, o algoritmo de comunicação é bastante diferente do implementado na estação central. Assim, estas interpretarão a chegada de um comando de cada vez que ele aparece, respondendo sobre (e conforme) a sua correcção. Responderão com mais informação apenas no caso de o microcomputador central o requisitar. A chamada a estes algoritmos é feita quer pela chegada de um comando da rede de comunicação, quer pela conclusão da preparação de um conjunto de dados que foram pedidos pelo microcomputador central.

Eventualmente não poderá ser efectuada a transmissão desses "frames" se não tiverem sido reconhecidos alguns dos já enviados; este último aspecto prende-se com a sequenciação dos "frames" que transitaram na linha como exposto na concepção, relativamente aos elementos de procedimentos.

Algoritmo:

Inicia-se pela chegada de um comando e tendo originado uma interrupção.

```
Interrupção devido à recepção de um comando
  no caso de
    [recebeu comando ON] e [recepção correcta] então
      se [estava desligada] então
        [passa a estado LIGADO]
        [envia resposta ACK]
      senão
        [envia ACK]   "já estava ligada! Não era necessário refazer ligação"
    [recebeu comando OFF] e [recepção correcta] e [estava ligada] então
      se [estava ligada] então
        [passa a estado DESLIGADO]
        [envia resposta ACK]
      senão
        [envia ACK]   "já estava desligada!" (este passo pode ser eliminado)
    [recebeu comando RR] e [recepção correcta] então
      se  [estado LIGADO]  ou [estado RECEBER] então
        [passa a estado ENVIAR MENSAGEM]
        [envia resposta ACK]
        [preparar comunicação a efectuar]
        [envia dados até acabar ou ocorrer número elevado de erros]
    [recebeu comando RNR] e [recepção correcta] e [estava ligada] então
      [passa a estado LIGADO]
      [envia resposta ACK]
    [recebeu comando ACK] e [recepção correcta] e [estava ligada] então
      se [estado de ENVIAR MENSAGEM] então
        se [ainda tem dados para enviar] então
          [envia resposta DATA]
        senão
          [passa a estado LIGADO]
          [envia resposta RR]
    [recebeu comando NACK] e [recepção correcta] e [estava ligada] então
      se  não[estado ENVIAR MENSAGEM] então
        [passa a estado LIGADO]
      senão
        se [não esgotou nº de retransmissões] então
          [reenvia resposta]
        senão
          [passa a estado LIGADO]
    [recebeu comando DATA] e [recepção correcta] e [estava ligada] então
      se  [estado LIGADO]  ou [estado RECEBER] então
        [passa a estado RECEBER MENSAGEM]
        [envia resposta ACK]
        [prepara campo de informação recebido]
  senão
    [envia resposta NACK]
```

Figura 52. Algoritmo base do módulo de comunicações, relativo à camada de controlo de ligação, a residir nas estações remotas, equivalente ao diagrama de estados da figura 51.

A estrutura do algoritmo de comunicação foi descrita por intermédio de um diagrama de estados, com as transições determinadas pela detecção de ocorrências

específicas, nomeadamente a detecção de chegada de um comando, o envio de uma resposta ou a detecção de um erro (figura 51), que permitiu estabelecer o módulo final, apresentado na figura 52.

O módulo de comunicação, pode estar num dos seguintes estados:

- ***LIGADO***, estado resultante do estabelecimento de uma ligação, ou da ocorrência de um erro grave sem ter sido efectuado o terminar da ligação,
- ***DESLIGADO***, estado de funcionamento normal quando não está estabelecida qualquer ligação entre o microcomputador central e a estação,
- ***ENVIAR MENSAGEM***, estado em que a estação está a enviar "frames" pedidos para a rede,
- ***RECEBER MENSAGEM***, estado em que a estação está a coleccionar "frames" recebidos da rede.

Existem uma "flag" (flag de RECEPÇÃO) que indica ter havido uma comunicação, permitindo assim às camadas superiores detectá-la.

O módulo de comunicações implementado funciona concorrentemente com os programas em execução na estação remota de aquisição; esta concorrência obriga o utilizador ou o programa a verificar se ocorreu alguma comunicação a que seja necessário dar atenção. Em última instância esta tarefa cabe à camada de controlo de mensagem, qualquer que seja a sua complexidade ou forma de implementação.

O funcionamento em simultâneo com a comunicação do programa principal só é autorizada durante os tempos de espera, que correspondem:

- **na transmissão:** aos instantes que decorrem entre o momento em que é carregado o registo de transmissão da UART, e o momento em que é gerada a interrupção devida a este dispositivo estar pronto a enviar novo caracter. Esta situação de concorrência termina com a transmissão do último caracter do "frame".
- **na recepção:** na situação de espera de um comando é permitida a execução do programa principal; este só é interrompido quando é gerada uma interrupção pela chegada de um caracter ou por ter sido esgotado o tempo limite sem receber o comando. Esta situação de concorrência termina com esta última interrupção ou com a recepção do último caracter de um "frame".

O desenvolvimento destes programas (primitivas de comunicação) foi efectuado sem recorrer a estas possibilidades de concorrência, com vista a, por um lado isolar

eventuais problemas que pudessem surgir e, por outro lado, validar todo o algoritmo que seria posteriormente adaptado para permitir a concorrência; esta metodologia, esquematizada na figura 53, permitiu minimizar o tempo total de desenvolvimento.

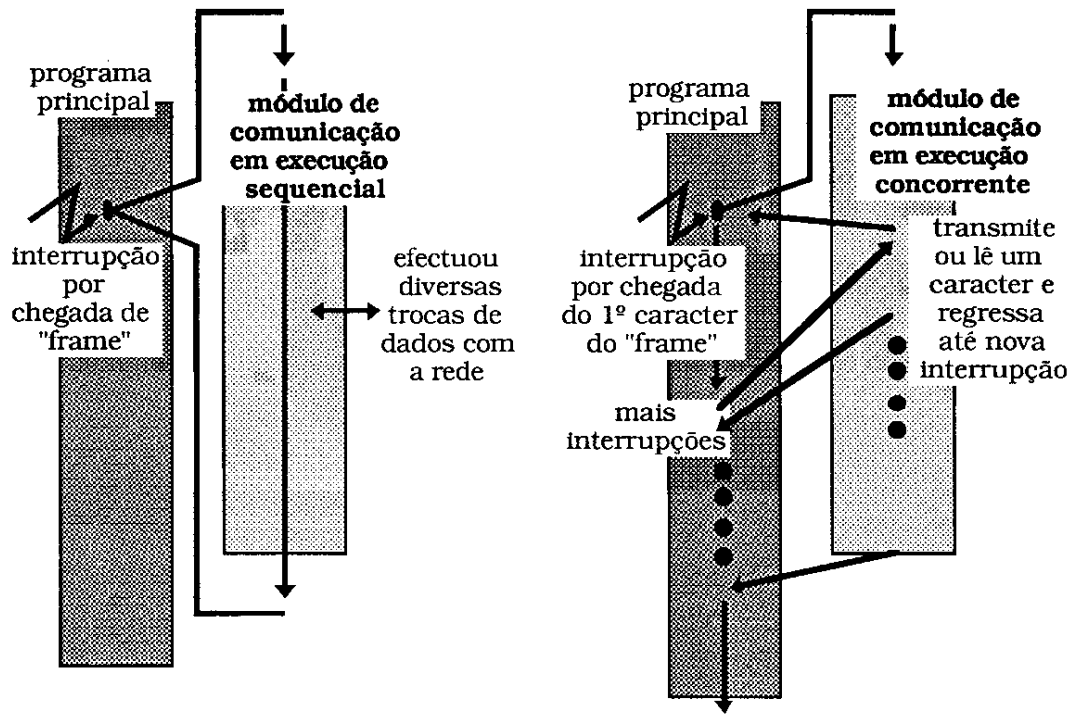


Figura 53. Esquema da implementação da concorrência no módulo de comunicação (*à dir.*) *inicialmente* sem essa característica (*à esq.*).

Como já foi referido todo este módulo foi implementado em "assembler" do 8051, com vista à minimização do tempo de processamento; esta opção permitiu conter o referido tempo em limites muito aceitáveis, não sendo imposta nenhuma restrição muito grave pela utilização do protocolo.

A informação a enviar é composta por "frames" de comprimento limitado, com um campo de informação máximo de 64 caracteres; posteriormente este limite, embora a título experimental, foi alargado para 128 bytes, com vista a permitir um maior **rendimento** ("throughput") na comunicação.

Na parte deste trabalho relativa à análise dos resultados, são apresentados os tempos despendidos na comunicação usando diversos tamanhos de "frames".

Apresentam-se em apêndice alguns pormenores adicionais relativos à implementação destas primitivas, bem como as do microcomputador central.

9.4 Módulo do interpretador.

Este módulo permite, quer ao utilizador, quer ao microcomputador central ***introduzir***, ***listar*** e ***executar*** programas escritos numa linguagem própria e de mais alto nível que o "assembler".

Uma vantagem da utilização de um interpretador reside no facto de uma eventual evolução do "hardware" disponível nas placas de aquisição de dados, incluindo o próprio microcontrolador, não obrigar a modificações drásticas dos algoritmos de controlo residentes no microcomputador central e que são carregados e executados quando necessário.

É possível aceder a todos os recursos disponíveis na estação remota, bem como enviar e receber programas ou qualquer conjunto de dados, quer usando o formato "intel hex." quer por acesso directo à memória, carregando-os em qualquer memória (excepto os primeiros 32K de código). Por este processo é possível carregar rapidamente um programa e posteriormente executá-lo mesmo que este esteja na memória de um periférico, como seria o caso das 8155.

O programa de controlo tanto pode estar escrito usando exclusivamente as instruções definidas no interpretador, quer usando uma combinação de instruções do interpretador e código do microcontrolador, uma vez que permite chamar sub-rotinas escritas em código máquina e residentes na memória de código, para além dos módulos escritos na linguagem do interpretador residentes em ***qualquer zona*** de ***qualquer memória***.

Este interpretador tem ainda a vantagem de ser bastante simples (é constituído apenas por 11 instruções) podendo o utilizador recorrer a ajudas. A escrita de um programa, de forma interactiva está bastante simplificada, sendo o utilizador 'ajudado' na sua introdução.

Dispensamo-nos neste ponto de referir as diferenças associadas a esta visualização conforme o nível do protocolo de comunicação que estiver na altura em uso, considerando-se que a nível de utilizador final, o interface entre o utilizador e a estação remota é transparente, podendo nesta fase corresponder à de uma utilização tradicional com uma consola privada.

Ao utilizador estão acessíveis os seguintes comandos:

1 - ***Listar*** um programa previamente carregado numa zona de memória.

2 - ***Transmitir*** pela linha de comunicação uma zona de memória, a receber e eventualmente guardar pelo μC central.

3 - ***Receber*** informação da rede e carregá-la numa zona de memória.

4 - ***Inserir*** numa zona de memória, de forma interactiva, um programa.

5 - ***Executar*** um programa já introduzido numa zona de memória.

6 - ***Visualizar*** (ver figuras) quatro ecrãs de ajuda ("help").

7 - ***Terminar*** as operações com o interpretador.

O recurso ao comando de ajuda ("help"), permite visualizar uma série de ecrãs bastante elucidativos sob a forma de utilizar as diversas potencialidades do interpretador. Apresentam-se, na figura 54, dois ecrãs de ajuda genéricos, enquanto no apêndice B, figura B1, podem ver-se dois outros ecrãs de ajuda, com descrição mais pormenorizada das instruções previstas.

O programa a implementar pode residir em qualquer dispositivo que possua memória; embora existindo a limitação de não poder ser introduzido pelo operador na EPROM, nada impede que esta memória não tenha já programado um *subprograma* usando esta linguagem, muito embora fosse de esperar que este fosse escrito em linguagem máquina; este aspecto não invalida a sua utilização por parte de qualquer outro módulo (seja ou não escrito em linguagem interpretada).

Outra vantagem (ou melhor possibilidade) deste (ou outro) interpretador, reside no facto de ser possível implementar no microcomputador central um compilador desta linguagem; por este processo seria possível optimizar o programa a escrever (em tempo de processamento e/ou memória ocupada), depois de se ter verificado o seu funcionamento utilizando o interpretador.

Considerou-se um espaço de endereçamento completamente diferente do habitual, definindo cada posição de memória por 4 bytes; assim, ao invés dos tradicionais 64 Kbytes (65536 bytes) existe o limite máximo de 64*64 Kbytes, isto é 4 Mbytes. Este limite, embora na realidade seja inatingível, deve-se à consideração de que cada posição de memória é identificada por um "*record* " (nomenclatura derivada da linguagem PASCAL); este é constituído por 3 *campos* : tipo de memória, endereço propriamente dito e o banco da memória.

a)

```
Este interpretador de comandos pode ser utilizado sob a forma imediata
  ou por um programa existente em qualquer memoria.

    Usado a partir da consola dispoe dos seguintes comandos:

                  i ou I - inserir comandos numa  zona de memoria que nao seja
                   codigo.
                  x ou X - executar um programa ja introduzido ou residente em
                   em qualquer memoria.
                  e ou E - enviar uma zona de qualquer  memoria para a consola
                   no formato intel hex.
                  l ou L - listar um  programa ja introduzido ou  residente em
                   qualquer memoria.
                  r ou R - receber  para qualquer  memoria (excepto codigo), e
                   num certo segmento  informacao a partir  da consola
                   no protocolo intel hex.
                  f ou F - fim do trabalho com o interpretador.
                  h ou H ou ? ou ! - este ecra.
PRIMA UMA TECLA
```

b)

```
As instrucoes (comandos) disponiveis para compor o programa sao as seguintes:
    (em maiusculas as abreviaturas usadas no comando inserir do modo imediato)

COMANDO     |           DESCRICAO
------------|-------------------------------------------------------------------
MUdar       |Transfere conteudo de posicoes de memoria ou valores imediatos para
            |uma posicao de memoria, demorando certo tempo.   Sao transferidos N
            |sendo os ponteiros das posicoes de memoria actualizados de cada vez
            |conforme indicacao do incremento definida na instrucao.
Compactar   |Compacta informacao de conteudo de n posicoes de memoria ou valores
            |imediatos para memoria a partir de um endereco.
ENviar      |Envia para a consola em formato intel hex.
Receber     |Recebe para memoria num segmento informacao no formato intel hex.
Terminar    |Termina a execucao de um modulo (fim ou return).
SAltar      |Salta para uma posicao de memoria.
SOmar       |Soma uma "word" a uma posicao de memoria.
SE          |Se uma relacao for verdadeira salta para uma posicao de memoria.
EXecutar    |Executa a partir de uma posicao de memoria.
MArcar      |Marca o fim de um modulo visivel na listagem.
Lcall       |Chamar uma subrotina.
Filtrar     |Filtrar uma zona de memoria.
Operar      |Executa uma funcao (log./aritm.) entre zonas de memoria

PRIMA UMA TECLA
```

Figura 54. Ecrãs de ajuda do interpretador: a) relativo ao funcionamento e b) com informação relativa às instruções e suas funções.

Esta metodologia foi adoptada como processo de 'jogar' com todos os tipos de memória disponíveis (interna ou externa, de código ou de dados) bem como com os periféricos, dado serem obrigatoriamente do tipo "mapped memory".

Foi aliás esta última particularidade que veio impor uma condição associada a algumas das instruções disponíveis; efectivamente, a utilização de alguns periféricos, obriga a restringir o tempo disponível entre duas operações, ou melhor, entre a execução de duas instruções consecutivas. Esta limitação obrigou a que se previsse, associada a algumas instruções, um tempo mínimo para a sua execução, garantindo-se assim que o periférico pudesse estabilizar os valores que vai fornecer ou receber.

A implementação deste módulo foi efectuada, a partir de uma especificação cuidada, usando a linguagem PL/M-51; esta linguagem permite diminuir consideravelmente os tempos de desenvolvimento já que é de nível mais elevado que o tradicional "assembler", codificando uma série de instruções de uma forma bastante eficaz. Só foi possível utilizar esta linguagem dado os tempos de processamento não serem críticos; sendo fácil a interligação com o "assembler", foi possível recorrer a uma série de sub-rotinas escritas para outros módulos, para serem acedidas por este.

Capítulo 10. Implementação da estação mestra.

Considerações gerais.

A estação mestra é um microcomputador pessoal tipo IBM-XT/AT ou compatível. Para ser possível a sua utilização para controlo da rede de comunicação de dados foi necessário reformular a metodologia tradicional de comunicação; em particular, a opção por um protocolo físico de comunicação de tipo menos vulgar, o RS-485, obrigou à utilização de uma placa de comunicação própria que implementa uma porta assíncrona.

No microcomputador central foram desenvolvidos diversos módulos de "software", que controlam quer a comunicação com as estações quer o funcionamento geral do sistema (ver figuras 55 e 56).

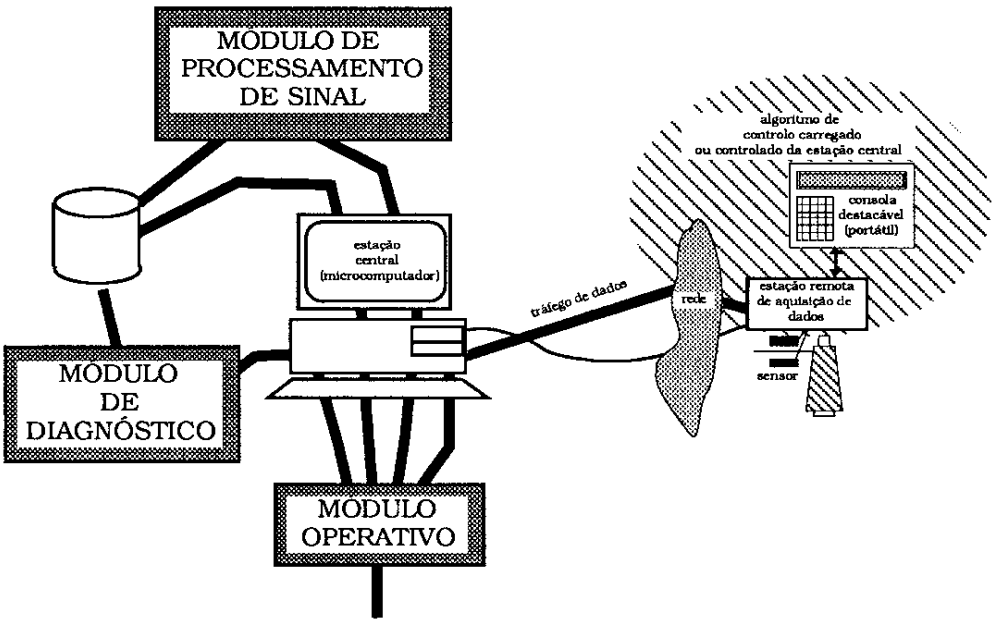


Figura 55. Esquema geral dos módulos implementados na estação mestre e sua interligação com o funcionamento da estação remota.

Dadas as altas velocidades pretendidas para a comunicação, foi necessário desenvolver este módulo em assembler do IBM (ASM86) conseguindo-se garantir uma

velocidade de comunicação de 115 Kbaud máximo, na presente arquitectura do sistema.

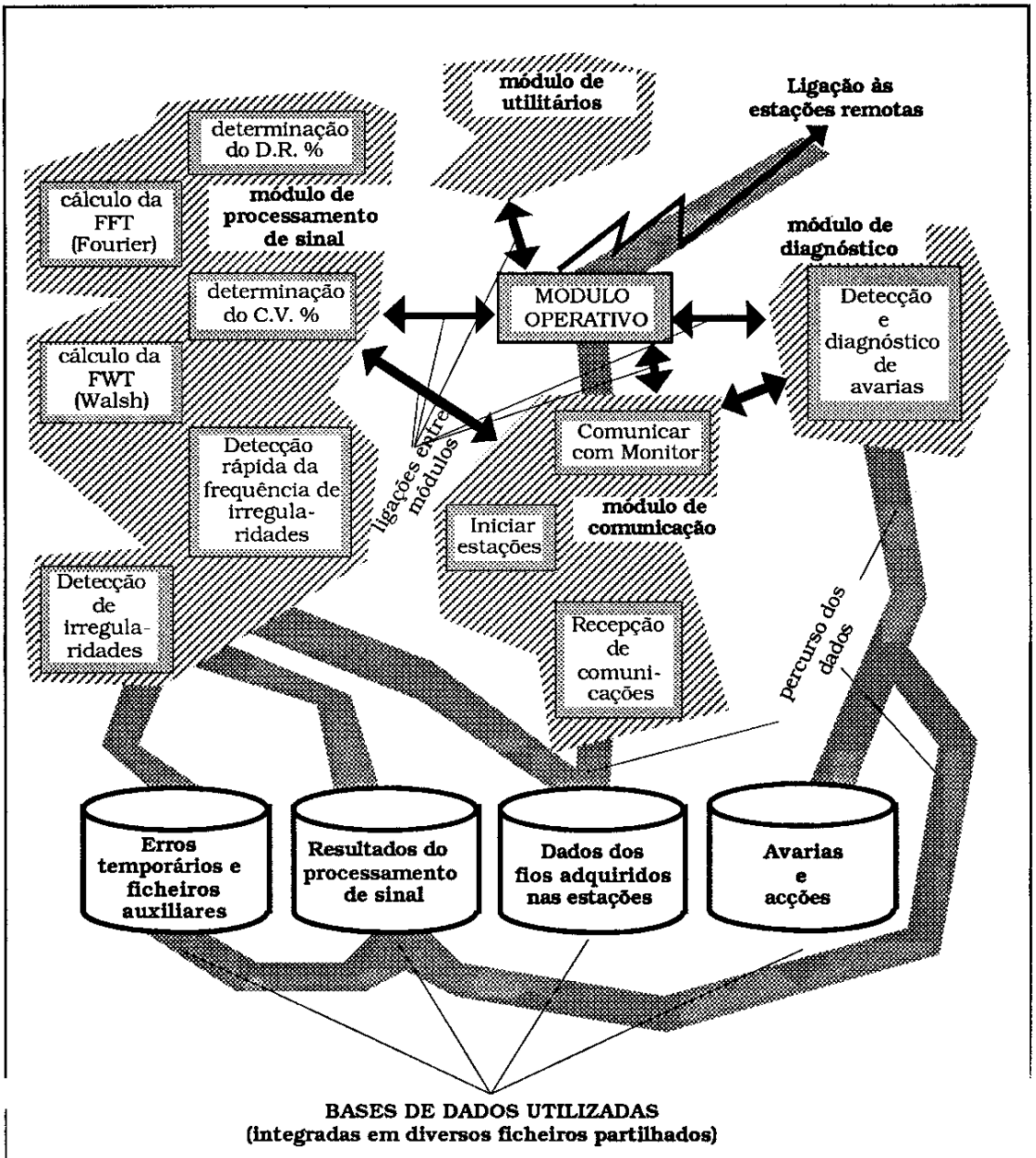


Figura 56. Interligação dos diferentes módulos; representam-se as ligações entre os diversos módulos (linhas com setas) e o 'percurso' dos dados para cada módulo (linhas a ponteado).

O módulo de comunicações pode ser acedido pelos restantes módulos que o pretendam; com esta facilidade é possível aceder às estações remotas em quase qualquer instante.

Além deste aspecto, refira-se que se procurou limitar consideravelmente o peso, em termos de tempo, do processamento associado à comunicação; para o efeito, e tal como para as estações remotas optou-se pelo funcionamento dos programas de comunicação em concorrência com o programa que o chama. Todos estes pormenores serão esclarecidos com maior profundidade na secção respectiva.

Construiu-se um módulo de processamento digital de sinais discretos, que inclui a determinação de espectros utilizando diversos tipos de transformadas rápidas (Transformadas rápidas de Fourier, Walsh e outras), bem como programas para o processamento dos dados que fornecem como resultado parâmetros relativos a qualidade de fio têxtil (Coeficiente de Variação, Taxa de Desvio, etc.). Este módulo foi implementado usando um misto de linguagens de alto e baixo nível: QuickBasic (versão 3), e "Assembler" (versão 5) do 8086.

Foi desenvolvido um módulo inteligente para a detecção de avarias quer das estações remotas quer dos sistemas que estas controlam; este módulo permite também a tomada de decisões automáticas com base nas avarias detectadas, funcionando com base na experiência acumulada ao longo do tempo de utilização. Este sistema implementa, na realidade, um *sistema pericial*, funcionando sobre a acção de um operador e que permite o diagnóstico e correcção de avarias numa base de causa efeito.

Este módulo foi desenvolvido com o recurso a técnicas de Inteligência Artificial, utilizando a linguagem PROLOG (Arity Prolog compilado).

Foram por fim desenvolvidos dois módulos de gestão global do sistema:

- um módulo de interligação que permite por meio de menus aceder a qualquer dos outros, incluindo a possibilidade de a partir de qualquer um deles comunicar com as estações remotas;

- outro que, permitindo a utilização de um conjunto de utilitários gerais, autoriza o utilizador a efectuar a gestão do funcionamento da estação central, em termos de microcomputador pessoal; é assim possível criar, modificar, editar ficheiros ou mesmo executar outros programas, sem obrigar ao abandono do módulo em uso em cada instante;

Estes módulos foram escrito em QuickBasic e "Assembler".

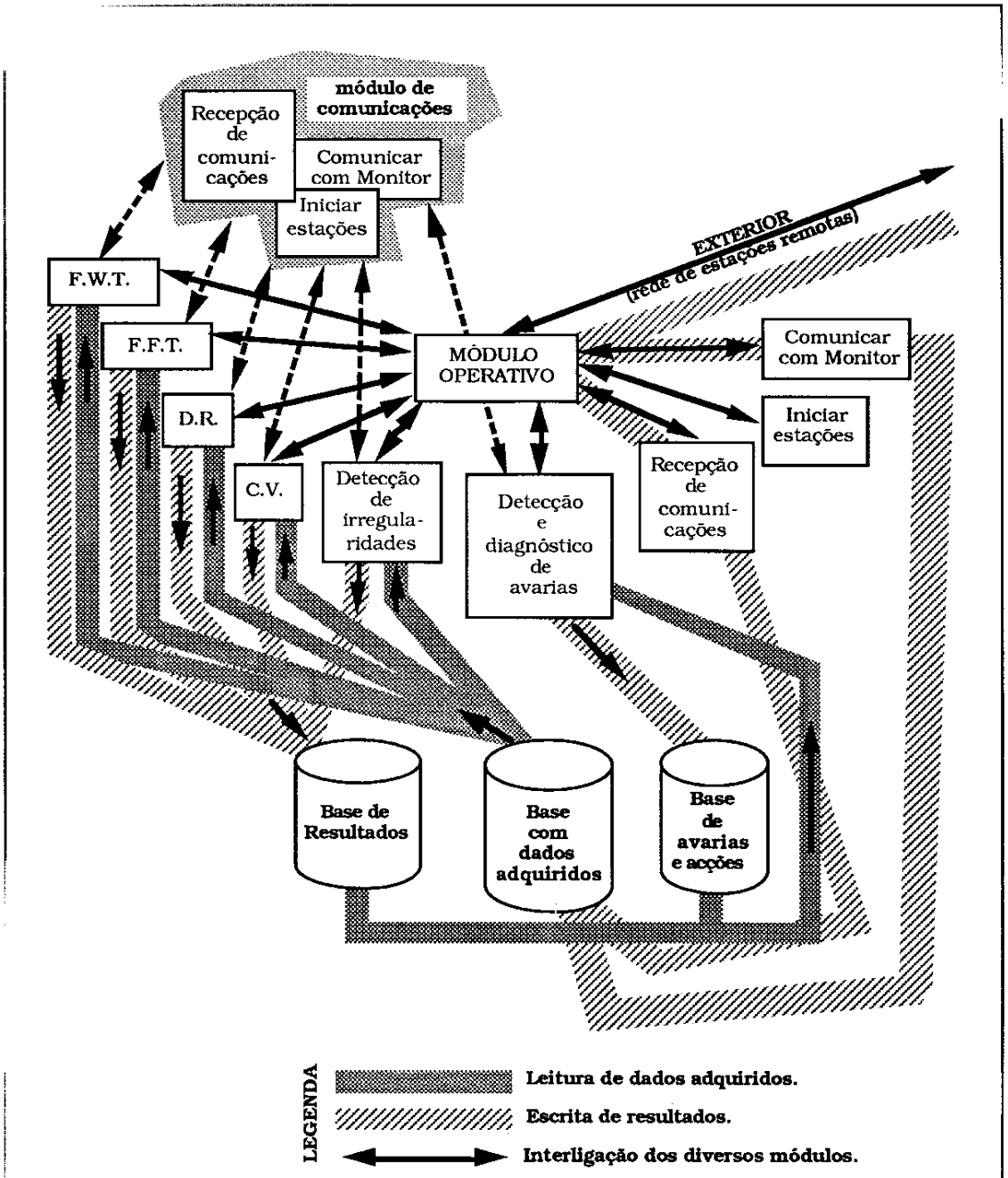


Figura 57. Esquema geral da interligação dos módulos; representam-se na figura os percursos principais dos dados adquiridos nas estações remotas e a interligação entre módulos. Esta é sempre efectuada por intermédio do módulo operativo com a excepção do módulo de comunicações que, para além de ser acedido deste, pode sê-lo dos outros.

Como se apresenta na figura 57, os módulos interligam entre si através de um outro, de gestão e supervisão do funcionamento do sistema. Esta metodologia de interligação centralizada não é obrigatória, mas não foi considerado importante, de momento, permitir em cada instante 'saltar' para qualquer um dos módulos disponíveis.

A única excepção considerada para a transição entre os diversos módulos foi a ligação do módulo de comunicação aos outros; neste caso, sendo previsível o vulgar recurso a dados existentes nas estações de aquisição de dados, optou-se por permitir esta ligação ao programa de comunicação em qualquer instante.

Para a ligação entre os diversos módulos escritos em linguagens tão diferentes como é o caso do "Assembler", BASIC ou PROLOG, foram implementados pequenos módulos privados; uma dificuldade grande surgiu na ligação de qualquer módulo escrito em "QBASIC" e "ARITY PROLOG", que não estando prevista por nenhum dos compiladores das 2 linguagens impediu a transferência do controlo entre os diferentes módulos com a passagem tradicional de parâmetros e variáveis (não é possivel usar um "LINKER" vulgar); este problema foi ultrapassado através de um pequeno módulo em "assembler" que permite a transição entre os dois módulos, ficando a eventual troca de variáveis garantida pela utilização de ficheiros auxiliares.

Os diversos módulos usam ficheiros de vários tipos, caracterizados pelas funções a que estão associados; assim existem ficheiros com a informação recolhida nas estações remotas, outras com resultados do processamento de sinal e outros com os ficheiros da base de conhecimentos das avarias e acções de correcção; este último é utilizado exclusivamente pelo módulo de diagnóstico de avarias, enquanto os restantes poderão, na sua maior parte ser acedidos pelos diversos programas que constituem os módulos.

Existem ainda, e para além destes, outros ficheiros, especialmente dedicados às tarefas de gestão do microcomputador central independentemente do sistema implementado bem como informação sobre o sistema, nomeadamente o tipo de "display" da estação central as estações existentes anteriormente na rede, etc.

As opções sobre quais as linguagens a utilizar em cada um dos módulos parcelares referidos não foi uma decisão fácil nem se ficou a dever a opções puramente técnicas.

A utilização de uma linguagem de baixo nível, como o "assembler" ficou a dever-se, no essencial, à necessidade de implementar algumas primitivas de comunicação, em que o tempo disponível ou não era muito ou era, pelo menos, indispensável

minimizar. Esta linguagem acabou por vir também a ser utilizada na construção de outro tipo de primitivas genéricas (janelas, "driver" para 'rato', etc.) devido, por um lado ao facto de ter sido adquirida muita experiência no uso desta quando da implementação do módulo de comunicações e, por outro, dada a proximidade que algumas destas funções têm das camadas mais básicas do sistema operativo, nomeadamente o "BIOS".

A utilização generalizada de uma linguagem da família "BASIC", ficou inicialmente a dever-se a aspectos não directamente relacionados com o presente trabalho: por um lado existia desde à bastante tempo um profundo e continuado contacto com alguns interpretadores e compiladores desta linguagem, a que se pode ainda acrescentar a característica das novas versões destas linguagens aceitarem os programas antigos; por outro lado, o facto do núcleo de alguns dos programas, especialmente os de processamento de sinal, terem sido previamente desenvolvidos em BASIC, embora num âmbito muito mais restrito do que o pretendido.

O desenvolvimento posterior do trabalho veio no entanto a mostrar que a opção por esta linguagem não se traduziu num rendimento tão elevado quanto o que seria previsível obter com o recurso a outras, nomeadamente o ***C***; esta última, permitindo mais facilmente o desenvolvimento de programas estruturados, e com um interface mais claro e simplificado com o "assembler" e o sistema operativo, apresentava ainda a vantagem de ser praticamente um "standard", permitindo a utilização de diversos programas auxiliares desenvolvidos noutros trabalhos. Com estas considerações não se pretende afirmar que o QBASIC (versão 2.0 a 4.0), a linguagem utilizada para uma grande parte dos módulos, não tenha cumprido as especificações. Apesar das desvantagens apresentadas, verificou-se ser também relativamente fácil a interligação desta com outras linguagens e, por outro lado pode afirmar-se que em termos de rapidez não deslustra de outras; isto deve-se no essencial à implementação interna do compilador (que possui opções para aumentar a rapidez *versus* espaço de memória ocupado) e à definição de procedimentos e funções que permitem facilitar consideravelmente a especificação estruturada de um algoritmo. A acrescentar a estas considerações, é de **enorme** vantagem a possibilidade de, em certas situações, executar de forma interpretada uma pequena parte de um algoritmo, apesar de se manter uma estrutura que obrigue à declaração antecipada das variáveis (idêntico ao PASCAL ou C). Um último inconveniente está associado à versão mais recente (versão 4.0) que apesar de se aproximar ainda mais de outro tipo de linguagens estruturadas (já permite inclusivamente a recursividade de procedimentos e funções) gera um código

muitas vezes mais lento que a de versões anteriores; este aspecto veio impedir a sua utilização generalizada no sistema.

Para o módulo de diagnóstico automático de avarias e apresentação de acções correctivas, não seria viável recorrer a nenhuma das linguagens anteriormente referidas; pelo contrário, existem diversas ferramentas (incluindo linguagens) que permitem a implementação mais eficaz do módulo pretendido.

A opção pela utilização da linguagem PROLOG, dada a inexperiência que possuíamos no uso deste tipo de linguagens ficou a dever-se a dois factores:

- Disponibilidade imediata quer de interpretador quer de compilador desta linguagem (ARITY PROLOG e IF PROLOG);

- Existência de um núcleo de investigadores nesta Universidade com muitos trabalhos desenvolvidos na área da 'Inteligência Artificial' e cujo conselho apontava para a utilização desta linguagem.

Foram desenvolvidos, em geral utilizando "assembler" (MASM V5.0) diversas **ferramentas** que podem ser utilizadas por todos ou alguns destes módulos para facilitar o interface com o utilizador bem como a passagens de uns módulos para outros.

Estas ferramentas constituem uma biblioteca de programas que facilitam grandemente a programação de algumas das tarefas tradicionais, como seja a utilização de janelas e o 'rato' para a selecção de opções e execução de comandos; estas ferramentas traduziram-se numa grande vantagem especialmente no caso dos programas escritos em "QBASIC", dado permitirem o recurso a primitivas pouco usuais nesta linguagem.

Um dos programas desenvolvidos permite a definição de uma janela de comunicação no ecrã genérico; com esta possibilidade é possível efectuar transacções de dados entre o microcomputador central e as estações remotas sem modificar o ecrã principal, que frequentemente possui informação relevante.

Apresenta-se na figura 58 um exemplo da utilização de janelas que funcionam na forma "pop-up menu", tão vulgar nas aplicações tipo "Windows, X-WINDOWS" e no microcomputador "Apple Macintosh".

Neste exemplo, quando o rato estiver entre as colunas 75 e 66 (definido pelo parâmetro &h4B42) e na linha 1 (parâmetro &h0101) com o botão esquerdo pressionado, será apresentada a janela 5 que possui 14 linhas (parâmetro &h050E);

esta janela está definida através da posição do canto superior esquerdo e do canto inferior direito (parâmetros &h013A e &h104F).

```
DefineJanela5:
    gosub EstabeleceJanela5          <<<<< Definição do texto da janela
    for i=1 to 45
        jan5%(i)=int(sadd(jan5$(i))) <<<<< Endereços das variáveis c/ o texto
    next i
    call CalculaEndereco(Offset5%,Segmento5%,varptr(jan5%(1)))
    call DefineJanela( &h050E,&h0101,&h4B42,  <<<<< Definição da janela
                       &h013A,&h104F,Offset5%,Segmento5%,Tipo1%)
return

EstabeleceJanela5:     <<<<< Definição do texto da janela
    Linha%(5%)=3%      <<<<< Linha de início da janela
    Coluna%(5%)=60%    <<<<< Coluna de início da janela
    LinhasJanela5%=14% <<<<< Nº de linhas da janela
    compjanela=20      <<<<< Nº de caracteres máximo de cada linha
    y$=space$(compjanela)
    jan5$(1)=left$("Ajuda (Help)"+y$,compjanela)+"$$"          <<<<< Texto de cada
    jan5$(2)=left$("Acede ao DOS (Shell)"+y$,compjanela)+"$$" <<<<< linha
    jan5$(3)=left$("Fim "+y$,compjanela)+"$$"
    jan5$(4)=left$(" ──────────────── "+y$,compjanela)+"$$"
    jan5$(5)=left$("Preparar controlador"+y$,compjanela)+"$$"
    jan5$(6)=left$("Ligar estação remota"+y$,compjanela)+"$$"
    jan5$(7)=left$("Receber comunicação"+y$,compjanela)+"$$"
    jan5$(8)=left$("Comunicação manual"+y$,compjanela)+"$$"
      ......................
    jan5$(13)=left$("|ACTUALIZA  AVARIAS|"+y$,compjanela)+"$$"
    jan5$(14)=left$("+──────────────────+"+y$,compjanela)+"$$"
return

      EndJanela%=varptr(Janela%)     <<<<< Endereço das variáveis c/ janela
      EndSeleccao%=varptr(Seleccao%) <<<<< e opção seleccionadas
      call resetrato                 <<<<< Proced. que faz o "reset" do rato
      call inibeint(EndSemaforo%,EndJanela%,EndSeleccao%)  <<<<< Inibição da
      ......................                               <<<<< interrupção
      while Fim=0
        Nada$=inkey$           <<<<< Conforme a janela e opção seleccionadas
        if (Janela%=1%) then   <<<<< executa uma sub-rotina ou procedimento
          on Seleccao% gosub Lpontos,Lamostragem,Lcompressao,LDirectoria,LFile,Ldump,
                             Lpasso,Lrecolha,Lmedia,Ljanela,LRetiraMedia,LFold,
                             LComparacao,Lderivada,Ltransforma
          end if
      ......................
```

Figura 58. Exemplo da preparação de janelas de selecção (tipo "pop-up menu") que serão activadas automaticamente quando o 'rato' estiver numa determinada posição com o botão esquerdo pressionado.

Após a apresentação da janela, deverá ser efectuada a selecção de uma das linhas, se se mantiver o botão esquerdo pressionado, sendo libertado quando estivermos na selecção pretendida.

A principal vantagem deste método reside em que este processamento é efectuado concorrentemente com todo o restante, desenvolvido em alto nível (QBASIC), sendo a passagem de parâmetros assegurado por interrupções que modificam o valor de uma variável; por outras palavras, após uma fase de inicialização (figura 58) em que:

- se definiu a variável que receberá a informação sobre a janela seleccionada;
- se definiu o endereço da variável que conterá a informação da opção seleccionada na janela escolhida,
- se indicou o texto que aparece na janela,
- se definiram as características da janela,

basta testar, de tempos a tempos, as referidas variáveis (actualizadas por meio de uma interrupção) com vista a verificar se foi seleccionada algumas das opções; o nome da variável que conterá esta selecção é definida pelo programador.

Esta solução não inviabiliza no entanto a utilização de teclas para a selecção das diversas opções dentro de cada janela, como é tradicionalmente preferido pelos utilizadores mais experientes.

10.1 Placa de comunicações.

Dada a necessidade de implementar o protocolo físico RS-485, foi instalada uma placa adicional de comunicação, usando este protocolo físico. Esta placa adicional é idêntica à existente em alguns microcomputadores tipo IBM-AT para implementar a 'porta' COM2; tem também por base a 'pastilha' 8250 — *uma UART* — que possui vários módulos de funcionamento permitindo variar o "baud rate" (por programação), e gerar uma interrupção por chegada de um caracter, por estar pronta a enviar um caracter ou por recepção de um caracter com erro; inclui ainda os "drivers" para comunicar usando o protocolo RS-485. Esta placa, que pode ser colocada em diversos endereços seleccionáveis por "jumpers", gera uma interrupção ao microcomputador central também seleccionável por "jumpers".

Com este dispositivo é possível comunicar em modo assíncrono através de uma linha de comunicação de um par de fios entrançados a que se podem interligar diversos elementos que comunicam com este microcomputador, sendo permissível que esta linha atinja 1200 metros sem recurso a repetidores (experiências efectuadas nas nossas instalações permitiram garantir o funcionamento correcto pelo menos com algumas centenas de metros).

Esta placa permite, por programação, inibir ou desinibir o funcionamento dos referidos "drivers" além de ser possível, também por "software" decidir sobre a orientação destes, isto é se, em cada instante, funcionam para recepção ou

transmissão. De salientar que esta placa usa para o efeito os circuitos integrados utilizados nas estações remotas, os "drivers" 75176.

10.2 Estratégia de apresentação de resultados.

Considerou-se como essencial para a apresentação de resultados a garantia desta ser efectuada, relativamente ao fio têxtil produzido, sob a forma de relatórios similares ao gerados pelos equipamentos tradicionalmente em uso nesta indústria; esta consideração aplica-se pelo menos ao que se refere à representação dos parâmetros tradicionais.

Assim, os resultados do processamento digital do sinal devem ser apresentados de modo a permitir uma fácil leitura de resultados têxteis, nomeadamente comprimentos de onda das fibras constituintes do fio, ao invés do tradicional valor em hertzs.

Os resultados apresentados por este módulo no cálculo do CV%, U% e DR% seguem a mesma estratégia, sendo no caso do CV% e DR% apresentados valores para diferentes comprimentos de análise, expressos em mm ou cm.

O operador poderá, para cada um dos programas deste módulo, seleccionar diversas opções de apresentação de resultados, conforme as necessidades do sistema em análise no instante. Com efeito, no caso da análise espectral é sempre possível apresentar os resultados sobre a forma de um espectro tradicional, com as abcissas em unidades de hertzs, bem como sob a forma de espectro de potência.

Garante-se assim a possibilidade de utilizar o mesmo algoritmo para diferentes tipos de trabalhos.

Relativamente ao módulo de diagnóstico de avarias, optou-se pela apresentação de resultados em diversas formas, distintas no essencial pela complexidade dos relatórios; a opção por cada uma delas, cabendo ao utilizador do sistema, estará sempre directamente relacionada com os seus conhecimentos sobre o funcionamento do sistema. Isto quer dizer que é possível obter resultados com níveis de complexidade bastante distintos, incluindo por exemplo relatórios sobre as avarias e acções disponíveis (informação qualitativa de fácil e imediata compreensão) a par de outros em que se faz uma análise da correlação entre avarias e acções, requerendo obviamente conhecimentos mais profundos sobre o sistema e sobre as características matemáticas dos parâmetros estatísticos a que se referem.

10.3 Módulo operativo.

Este módulo é responsável pela interligação dos outros módulos residentes no microcomputador central. Apresenta por selecção através de rato ou tecla diversos menus para a selecção de qual o módulo que se pretende correr.

Esta selecção é efectuada por escolha num menu existente na 1ª linha do ecrã (de 25, 43 ou 50 linhas) ou por toque numa das teclas indicadas na 2ª linha desse monitor (teclas de função f1 a f5); ao seleccionarmos uma das opções aparece uma janela com diversas opções, correspondentes aos programas do módulo que poderão ser executados. Na figura 59 mostram-se os ecrãs que 'aparecem' em cada opção.

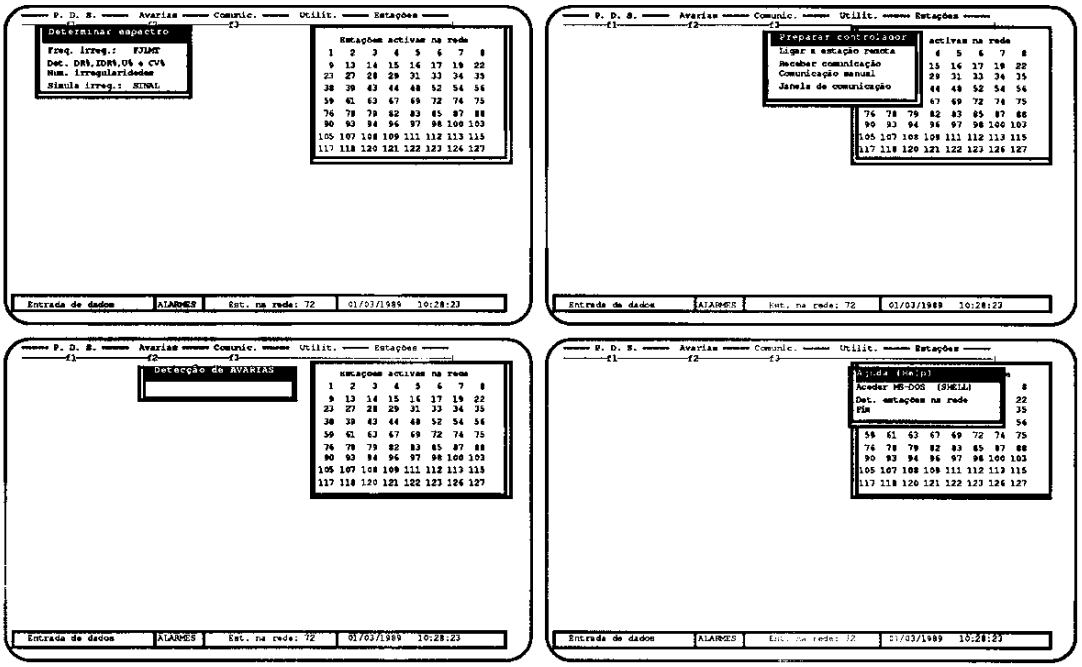


Figura 59. As opções de qual o programa a executar aparecem em cada linha de uma janela, agrupados com base no grupo a que pertencem. No ecrã é ainda fornecida informação complementar (hora, data, etc.).

Processamento Digital de sinal:

Seleccionando uma das opções da janela é possível escolher programas de determinação do espectro do fio têxtil ou de outro tipo de sinal, determinação de CV%,

U% e DR% além de irregularidades de fios, determinação da frequência de eventuais irregularidades e a simulação da aquisição de sinal (figura 59).

Como referido, a selecção pode ser efectuada por teclas para abrir a janela correspondente a cada módulo; neste caso a opção de qual dos programas a executar deve ser efectuada por recursos pelas teclas de direcção ou pela primeira letra do texto de cada linha da janela. Quando o cursor estiver sobre a opção desejada basta pressionar na tecla de "enter" ou "return"; pode ser abortada a selecção pressionado quer a tecla de ESC quer a tecla do espaço.

Diagnóstico de avarias (módulo de I.A.):

Seleccionando esta opção (figura 59) é possível comutar para o módulo de detecção automática de avarias e apresentação de soluções.

Módulo de comunicação com as estações remotas de aquisição de dados:

Com a selecção deste módulo (figura 59) é possível efectuar diversos tipos de comunicações entre o microcomputador central e as estações de aquisição de dados, quer sob o controlo directo do operador quer de uma forma automática.

As alternativas de comunicação ao dispor do utilizador permitem o estabelecer uma comunicação manual, efectuar transacções sob controlo 'manual' do operador, etc. Algumas destas possibilidades são utilizadas por outros módulos, nomeadamente quando se pretende contar automaticamente o número das estações ligadas à rede (estabelecendo e terminando ligação com cada uma sem intervenção do utilizador).

Outra opção, *a janela de comunicações*, permite criar uma janela adicional no ecrã onde se mostram todas as comunicações na rede, em simultâneo com os resultados da aplicação em execução. Esta possibilidade, extensível aos restantes módulos implementados, permite acompanhar tudo o que se passa na rede de comunicações, simultaneamente com o funcionamento do programa principal.

É sempre possível ter no ecrã informação complementar de interesse, nomeadamente o conjunto das estações existentes na rede (desde a última contagem), a data e hora.

Existe também uma zona de entrada de dados e outra de apresentação de mensagens urgentes, relacionadas com mensagens de emergência recebidas de estações.

Inclui também diversos ecrãs de ajuda sobre cada um dos módulos disponíveis.

Módulo de utilitários:

Existe ainda um módulo que permite efectuar algumas funções complementares (figura 59).

Com as opções deste módulo previu-se a possibilidade de aceder ao Sistema Operativo, a nível de "CLI", permitindo por exemplo uma eventual procura de um ficheiro a utilizar posteriormente, efectuar um acesso a outra aplicação, actualizar o tempo ou data, etc.

A identificação das estações que presentemente estão ligadas à rede será guardada num ficheiro auxiliar, com vista à sua utilização por outros programas. A opção de guardar esta informação em ficheiro deve-se a dois factores essenciais: por um lado não interessa efectuar esta contagem com grande frequência dado ser ocupado bastante tempo (basta lembrar que a contagem só pode ser efectuada através do estabelecimento de uma ligação com cada estação); por outro lado esta informação é importante ser partilhada pelos restantes programas.

Por fim pode ser pedida, neste módulo, informação de ajuda relativa a cada opção do menu, seleccionando primeiramente a opção de ajuda e posteriormente seleccionando o programa de que pretendemos obter esclarecimentos.

A passagem para os outros módulos é efectuada directamente, em certos casos, e por intermédio de pequenos programas escritos em "assembler" noutros casos.

10.4 Módulo de processamento digital de sinal.

Este módulo consiste num conjunto de programas de processamento digital de sinal especialmente utilizados para a detecção de irregularidades de fios têxteis, e num módulo de cálculo de espectros de sinais amostrados, quer representando a massa de fios têxteis quer de outras origens.

Tal como no caso anterior para cada programa deste módulo é possível seleccionar diversas opções, sendo uma delas o retorno ao módulo operativo e outra a selecção da opção de ajuda.

Para além destas e comuns a todos os programas do módulo existem opções de estabelecimento de ligação entre a estação central e uma estação remota, a transacção de dados através da rede de comunicação, terminar essa ligação e contar apresentando posteriormente o resultado, o número de estações da rede que estão em operação.

Pode também ser pedida a apresentação de informação sobre o programa actual por selecção da opção de ajuda.

Quando a estação remota adquiriu o conjunto de dados, obtendo a imagem digital do fio, pode transmiti-los ao microcomputador central para serem interpretados. Neste último o programa tem por objectivo a detecção dos erros, e a determinação de características usuais tais como o CV% (coeficiente de variação), U% (média do erro absoluto) e DR% ("deviation rate) e o espectro da imagem de espessura do fio.

As irregularidades periódicas ou quasi-periódicas de fio estão na origem de defeitos graves no tecido. É portanto indispensável conhecer a frequência de ocorrência dos erros, bem como caracterizar o fio produzido.

De notar que para as componentes espectrais obtidas no cálculo do espectro do fio existe uma correspondência entre a frequência dessas componentes e os comprimentos de onda. Conforme o número de pontos escolhidos, poderemos obter espectros que permitem detectar comprimentos de onda no fio até 260 metros (com um sensor típico de 8 mm de comprimento).

Para obter um espectro (não necessariamente com base em componentes sinusoidais), foram estudados, testados e implementados diversos algoritmos (figura 60), nomeadamente:

Algoritmo para a detecção rápida de frequência de impulsos (D.F.I.).
Transformada rápida de Walsh (F.W.T.).
Transformada rápida de Fourier (F.F.T.).

A diferença entre estes últimos reside no facto da FWT, embora com um algoritmo semelhante mas mais rápido que a FFT, apresentar um espectro de componentes rectangulares ao invés do da transformada de Fourier que são componentes sinusoidais. Estas considerações foram discutidos com detalhe na parte referente à concepção.

A apresentação de resultados procura seguir uma metodologia que permita uma fácil interpretação dos valores; apesar disto é possível obter uma representação diferente do espectro:

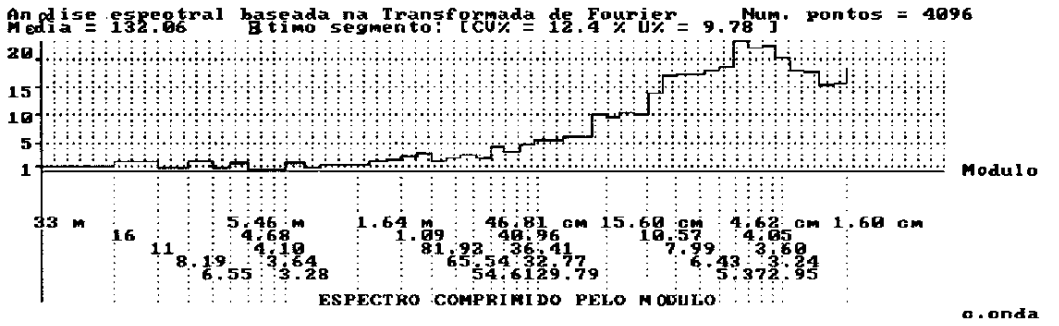
— representação de cada risca (ou banda) por nº de ordem de harmónico,

— apresentação dos resultados como uma lista de valores numéricos,

— etc.

Quer o tipo de apresentação de resultados, quer a aplicação de algoritmos de pré e pós-processamento são seleccionados a partir de janelas, funcionando, tal como no módulo operativo, tipo "pop-up menus".

I - Usando transformada de Fourier:



II - Usando transformada de Walsh:

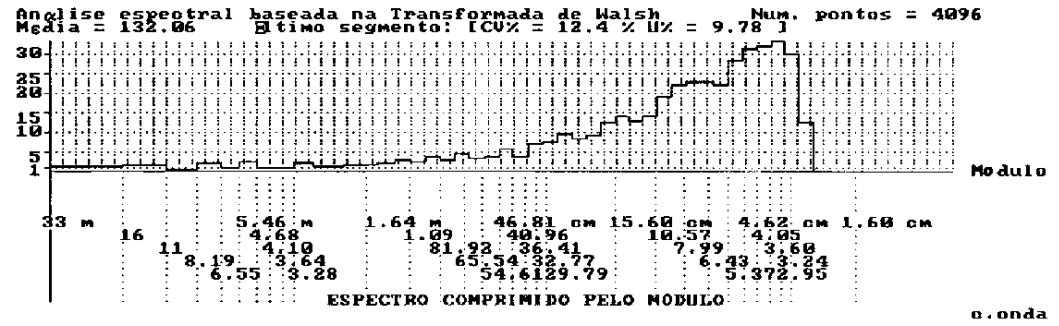


Figura 60. Resultados obtidos no cálculo de espectro usando a FFT e FWT.

Na figura 61 apresentam-se as opções das janelas deste módulo, designadamente as associadas ao pré-processamento; estas permitem, para além da definição das características gerais do processamento como sejam o nº de pontos, a frequência de amostragem, o ficheiro que contém a informação (nome, directoria), a imposição de certos algoritmos de pré-processamento.

Estes algoritmos permitem modificar os sinais adquiridos quer por meio de um ganho e um "offset" quer por intermédio de uma janela (Hamming, Hanning, Blackman ou rectangular); a utilização desta tem por objectivo minimizar alguns erros associados à imposição de efectuar o processamento com um nº finito de amostras e também aos erros derivados de não ser efectuada uma amostragem a uma frequência múltipla da dos harmónicos realmente constituintes do sinal em estudo.

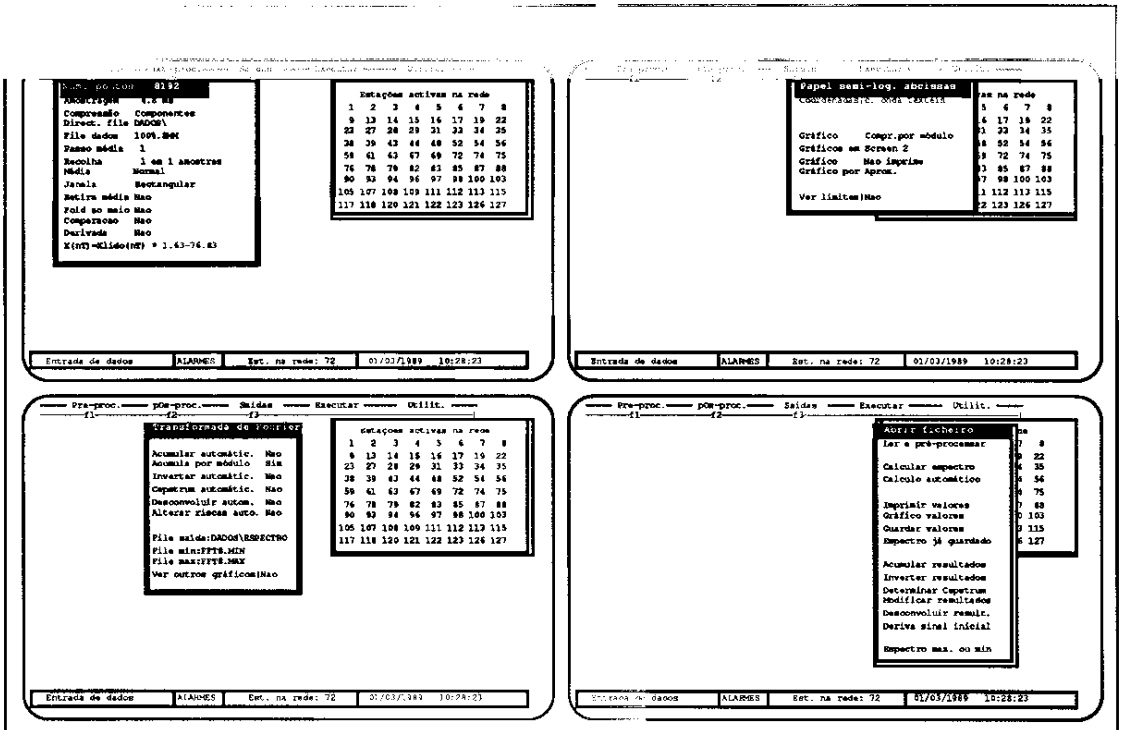


Figura 61. Selecção de condições de pré e pós-processamento, de saída e execução para o programa de cálculo de espectro.

Outros algoritmos de pré-processamento permitem transformar o sinal a analisar noutros, nomeadamente:

- num sinal em que cada amostra constitui o valor médio de X amostras reais; esta média pode ser obtida de forma normal (cada amostra real entra na determinação de **apenas uma** das amostras a considerar) ou deslizante (cada amostra a utilizar no cálculo é obtida pelas amostras de X amostras reais, sendo a seguinte obtida pela média das X amostras reais consideradas anteriormente eliminando a mais antigo e juntando uma nova amostra);

- num sinal obtido a partir de **metade** do número de pontos definidos; as restantes amostras são repetidas simetricamente em torno do valor médio, construindo-se assim uma função par;

- num sinal obtido por derivação do original;

- num sinal obtido por comparação com valores limites, correspondendo na realidade a limitar inferior e superiormente o sinal; com este processo é possível

garantir que num sinal digital, é eliminado qualquer ruído introduzido em todo o processo de amostragem;

- num sinal a que se retira previamente a componente contínua; este aspecto tem especial significado para os casos em que foi utilizada uma janela que introduza componente contínua.

Alguns dos aspectos teóricos associados a este processamento foram expostos na parte 2 deste trabalho.

Para apresentar um relatório de fácil interpretação é contudo necessário fazer um processamento adicional, dado que:

- No processo de fabrico não é possível garantir que o fio produzido mantenha todas as suas características ao longo do tempo; pode correr a existência de uma perturbação periódica durante alguns metros que não se venha a repetir posteriormente, pelo que seria pouco aconselhável utilizar os resultados deste segmento para classificar todo a produção.

- As perturbações no fio, mesmo mantendo as suas características (*não apresentando os problemas referidos no ponto anterior*), dificilmente terão uma frequência constante. De facto, sendo o fio analisado cada 8 mm, é pouco provável que não haja uma diferença nas frequências que leve a repartição de uma componente espectral pelas vizinhas (*comprimentos de onda de 990mm a 1010mm terão provavelmente a mesma causa*). Este aspecto assume uma grande importância se tivermos em conta a natureza mecânica da tracção e estiragem do fio têxtil, em que estas flutuações são perfeitamente admissíveis.

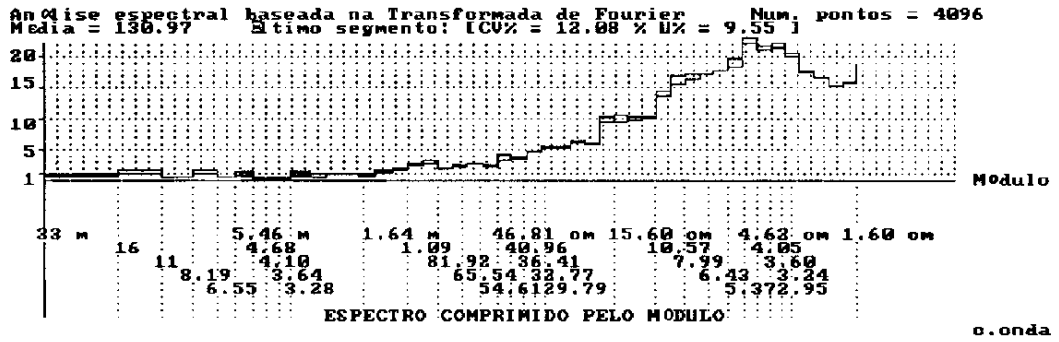
- As componentes espectrais não são seguramente múltiplas do período de amostragem, originando os já explicados erros de "leakage".

Para obviar a estes problemas foram previstos vários métodos complementares de tratamento dos resultados, correspondendo a um algoritmo modificado da transformada de Fourier e de Walsh; estes métodos são seleccionáveis por intermédio de diversas opções opções apresentadas na figura 61 nas janelas respeitantes ao *Pós-processamento*, *Saídas* e *Executar*. De entre as diversas possibilidades, destacamos:

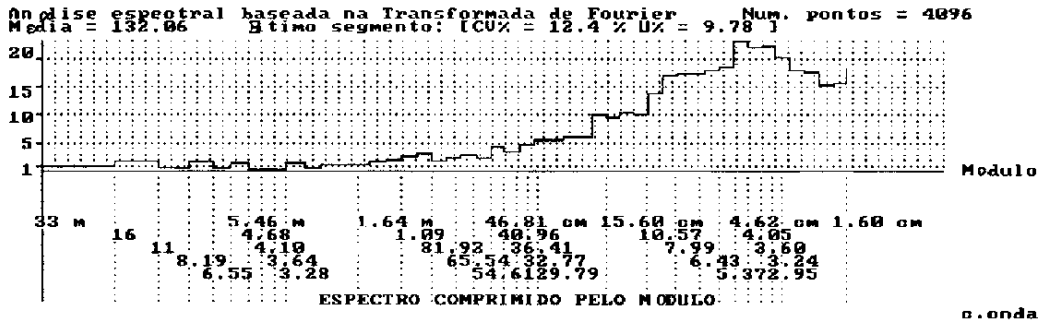
- O espectro é analisado para diversos segmentos de fio. Esses espectros são então 'agrupados'. Quer os erros pontuais, quer as próprias características do fio são detectadas em cada segmento de fio, dado que ao ser efectuado este agrupamento é

diminuida a importância da informação não estável. Se as perturbações persistirem, serão detectados no resultado final, um espectro obtido pelo conjunto de diversos espectros (figura 62.a); de notar a possibilidade de detectar os pontos mais marcantes por intermédio de uma inspecção visual.

a) Acumulação de segmentos (notar seg. acumulado sobre um não acumulado):



b) Espectro comprimido ('papel' semi-logarítmico):



c) Espectro não comprimido ('papel' semi-logarítmico):

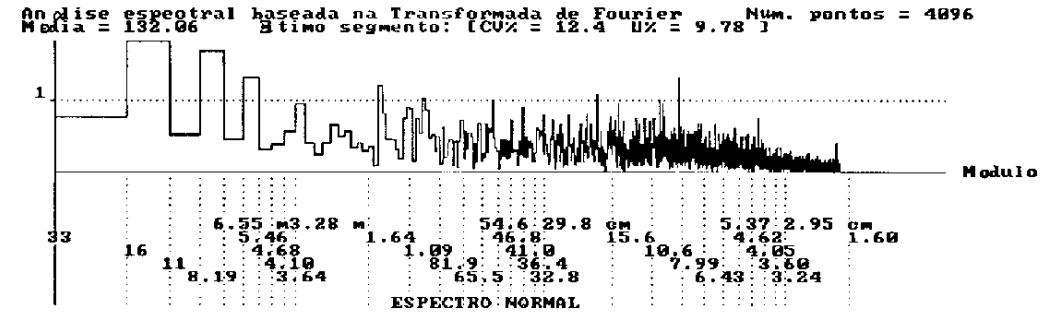


Figura 62. Características dos métodos de apresentação diferenciada de resultados (*b* e *c*) e acumulação de espectros de segmentos contíguos de fio (*a*).

- Para evitar os erros de "leakage", foram implementadas diversas janelas (Hanning, Hamming e Blackman).

- Associam-se diversas 'riscas' do espectro num certo intervalo (*uma banda de energia*), definido pelo utilizador, garantindo assim que a informação obtida não se dispersará por componentes próximas, mascarando um eventual erro.

A apresentação de resultados tem também diversas variantes conforme a apresentação pretendida (gráfica ou numérica); é ainda possível apresentar o resultado com resolução variável e usando diversos tipos de representações (tipo amostrado, Dirac, ou até uma aproximação), a partir da selecção de opções (figura 60).

Esta apresentação passa também pela possibilidade de visualizar o espectro resultante sob uma forma logarítmica ou semi-logarítmica (abcissas ou ordenadas) ou normal; o eixo das abcissas tanto pode estar numa escala de frequências, para o caso de sinais tradicionais, como numa escala de comprimentos de onda, com vista a permitir uma fácil identificação das características têxteis ou ainda numa escala numérica que representa o nº de ordem dos harmónicos.

Para além dos aspectos gráficos já explanados existem ainda opções adicionais que permitem efectuar o "dump" do gráfico para impressora, visualizar em simultâneo com o espectro calculado outros resultados já determinados permitindo a comparação visual ou mesmo para uma impressão desses gráficos em papel, etc.

A execução do programa pode condicionar-se de diferentes formas, conforme as opções de uma janela do menu; baseados nos resultados, quer devidos ao cálculo automático, quer devidos à execução de um comando, é possível acumular valores, criando um espectro médio (figura 62), determinar desvio do espectro em relação a traçados típicos, bem como actualizar um ficheiro com anomalias detectadas.

Tal como para o módulo operativo é possível utilizar uma janela de utilitários para comunicar com as estações, utilizar de menus de ajuda bem como aceder ao sistema operativo para as funções que se considerarem necessárias. É possível garantir o estabelecimento de comunicação com uma determinada estação e, em simultâneo (*concorrentemente*) manter a execução do programa de determinação de espectros; refira-se que em todas estes módulos é possível prever a existência de comunicações urgentes, capazes de activar 'os alarmes' visíveis na última linha.

Um dos defeitos frequentemente detectados é o devido à existência de pontos grossos e pontos finos. A análise da frequência com que ocorrem é bastante importante, e não é possível de detectar através da análise espectral. Foi assim implementado um programa (método *DFI*) próprio para o efeito. Neste programa o sinal original é transformado num sinal digital de erro, definido como ***1*** quando há uma ocorrência e ***0*** nos restantes casos (tal como apresentado no capítulo de detecção rápida da frequência de irregularidades). Aplicando o algoritmo desenvolvido obtem-se informação sobre as possíveis combinações de erros que originariam este sinal.

Este algoritmo é extremamente rápido dado que recorre apenas a operações lógicas. De notar que este algoritmo permite captar informação relativa a erros de igual frequência mas desfasados, como é o caso de erros provocados em diferentes fases do fabrico embora com a mesma frequência.

Para além destes programas de determinação de espectros foram implementados outros programas que calculam e apresentam gráficos de outros parâmetros utilizados na indústria têxtil (figura 63).

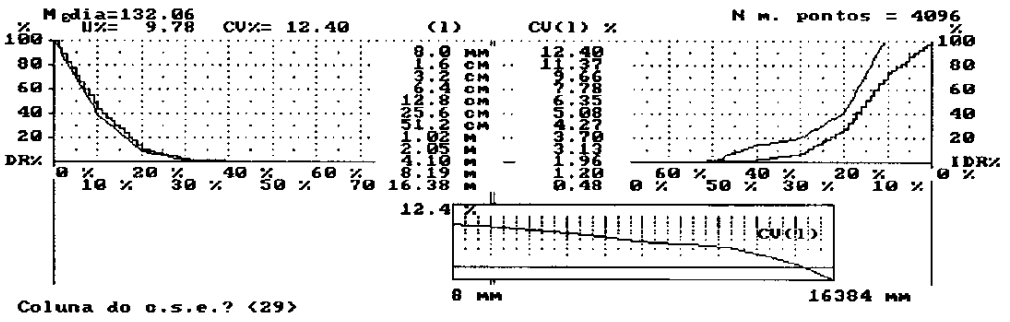


Figura 63. Resultados relativos ao cálculo do CV%, U%, DR% e IDR%.

Os parâmetros determinados são:

- CV% - Coeficente de variação que é definido como a razão entre o desvio padrão e a média do conjunto de amostras recolhidas.

- U% - Representa o desvio médio absoluto.

- DR% - Representa a percentagem de amostras que ultrapassam diversos limiares, relativamente ao conjunto das amostras recolhidas.

- IDR% - Peso percentual das amostras que ultrapassam determinados limites, relativamente ao peso global das amostras.

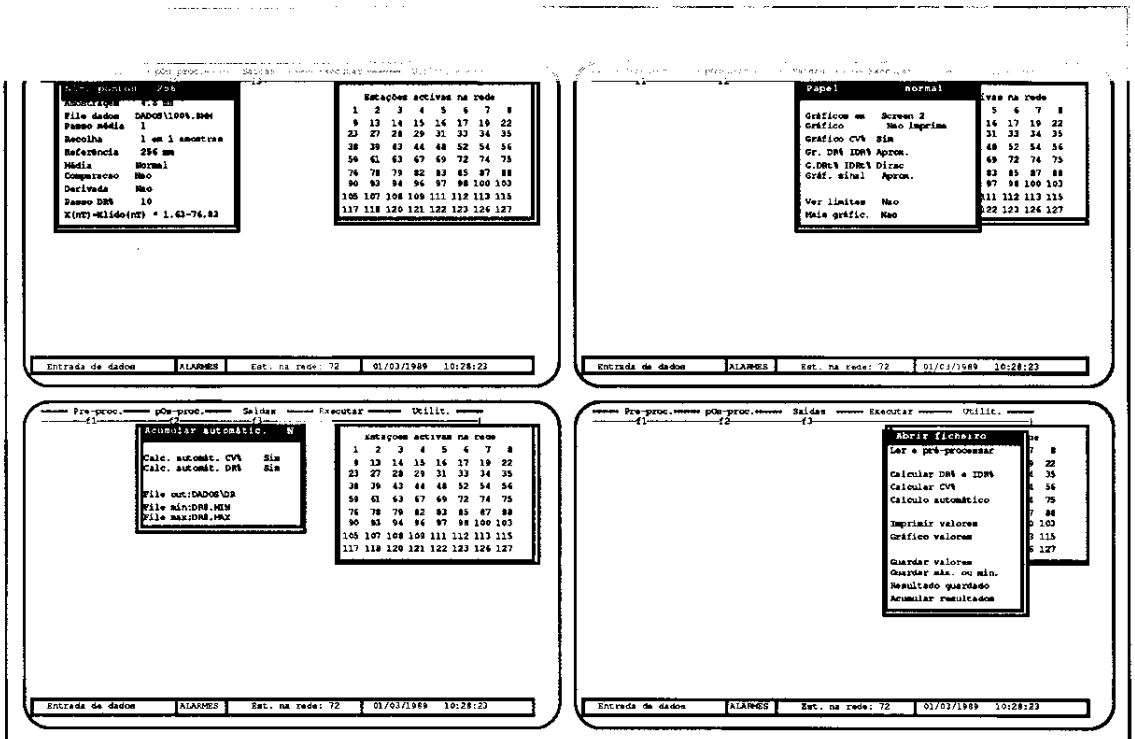


Figura 64. Ecrãs do módulo de determinação de CV%, DR% e U% relativa a 2 janelas com as respectivas opções.

Embora tenham sido desenvolvidos em separado, existe actualmente apenas um programa que efectua o cálculo de todos estes valores. Também para esse existem diversas opções (de apresentação, cálculo, etc.) seleccionáveis como no caso do módulo da FFT (figura 64). Resta referir que alguns dos resultados impressos, quer gráficos quer alfanuméricos, destes algoritmos são apresentados noutro capítulo, em comparação com alguns dos fornecidos pelos equipamentos industriais mais em uso.

10.5 Módulo de comunicação.

Dada a definição de dois possíveis esquemas de comunicação para as estações remotas, foram implementados também módulos distintos.

Pretende-se que para ambos o "software" de comunicação não acarrete elevados atrasos ao processamento global da programação residente no microcomputador central. Para se garantir esta característica é conveniente, por um lado transmitir e receber informação a uma velocidade o mais elevada possível e, por outro lado, não

existir paragem do processamento sempre que estamos perante uma situação de espera de informação qualquer que seja o seu tipo e tamanho (bit, byte, mensagem, ficheiro, etc.).

Para garantir a limitação do tempo destes atrasos é necessário que o módulo de comunicação funcione em concorrência com o processo em curso. A forma mais eficaz, embora mais dispendiosa, de o conseguir seria recorrendo a um processador dedicado que, funcionando em paralelo com o processador central do μC, acedesse por **acesso directo à memória** (D.M.A.) à informação pretendida e a enviasse pela linha de comunicação, que foi apenas considerada para uma evolução do sistema actual.

10.5.1 Módulo de comunicação assíncrono.
(Usando as características da *Estrutura 1*).

Este módulo de comunicação utiliza também a estrutura de efectivação de uma ligação dividindo-a em três etapas principais de: estabelecimento de ligação, efectivação da transacção e terminação da ligação.

O estabelecimento de uma ligação passa pelo envio de um caracter com o nono bit activo (em um) seguido de um cabeçalho constituído por três caracteres sem o nono bit activo; a definição de qual a estação com quem se pretende comunicar é efectuada pelo caracter enviado em primeiro lugar, tendo os restantes caracteres do cabeçalho a função de minimizar as possibilidades de ocorrência de erros ou más interpretações.

Na figura 65 apresenta-se a implementação utilizada para a chamada da primitiva de estabelecimento de ligação, a partir do programa pai, escrito em QBasic.

```
.....
Semaforo%=Semaforo%+1%
a%=val(FNInputFFT$("Estação a acordar ? ",FileCtrl,3))
call Inicia(EndInibicao%,EndMsgUrgente%,a%)
gosub TeclasOn
Semaforo%=Semaforo%-1%
call resetrato
call inibeint(EndSemaforo%,EndJanela%,EndSeleccao%)
Executou=1
.....
```

Figura 65. Processo de chamar a primitiva de estabelecimento de ligação entre estação remota e microcomputador; é apresentada a chamada do programa pai, escrito em QBasic (programas do módulo operativo ou de processamento de sinal).

Mostra-se ainda na figura 66, o módulo com que comunica o programa pai no exemplo da figura 65; este chamará por sua vez as rotinas e procedimentos capazes de implementar as trocas de caracteres que estabeleçam a ligação; este programa foi escrito em "assembler".

```
;=============================================================================
;                          INICIA
;Entrada para o programa de Qbasic para inicializar uma determinada estação.
;O parâmetro passado do Qbasic, o endereço, vem como valor de inteiro.
;=============================================================================
INICIA  PROC    FAR
      PUSH    BP
      MOV     BP,SP
      .....   .....
      MOV     BX,[BP]+10
      MOV     DX,[BX]             ;Variavel vinda do programa principal.
      MOV     BX,[BP]+8
      MOV     CX,[BX]             ;Variavel vinda do programa principal.
      MOV     AX,SS               ;Stack segment do programa principal.
      MOV     CS:END_INIBICAO,DX
      MOV     CS:SEGMENTO,AX      ;Guardou o endereço da variavel semaforo.
      MOV     CS:END_MSG_URGE,CX  ;É ainda passado o endereço da variavel que avisa
      MOV     BX,[BP]+6           ;o programa principal da existencia de uma mensagem
      PUSH    BX                  ;urgente que nao pode ser transmitida por
      PUSH    SS                  ;a comunicação com este ultimo estar desinibida
      MOV     AX,SS:[BX]
      MOV     BX,AX               ;Endereço da estação a inicializar.
      MOV     AX,2                ;Entrada para inicializar uma estação.
      MOV     CS:EXECUTANDO,1
      MOV     CS:NUM_BYTES_BUF_COM,0
      CALL    FAR PTR MODULO_COMUNICACAO
      POP     SS
      POP     BX
      MOV     SS:[BX],AX
      ......  ......
      RET     6
INICIA  ENDP
```

Figura 66. Processo de chamar a primitiva de estabelecimento de ligação entre estação remota e microcomputador; depois da chamada (figura 65) apresenta-se o procedimento em "assembler".

Assim como é possível, a partir do módulo operativo ou de processamento de sinal estabelecer uma ligação com uma estação, é possível terminar essa ligação, sendo para o efeito chamado um procedimento, escrito também em "assembler".

Quando da exposição do módulo operativo e de processamento de dados foi referida a possibilidade de conhecer o número e quais as estações que estão acessíveis na rede de comunicação; essa informação poderá inclusivamente estar presente no

ecrã à medida que se utilizam os programas implementados em cada módulo (figura 64).

Para efectuar esta contagem o módulo respectivo deve estabelecer uma ligação com cada estação; se esta for bem sucedida então a estação está activa e em caso contrário não está. Não é efectuada mais nenhuma troca de dados entre as estações, prosseguindo o algoritmo com a chamada da primitiva de terminar ligação, tal como se apresenta na figura 67, desde que tenha sido possível estabelecer previamente a ligação; em caso contrário, não é necessário terminar dado não ter significado lógico *acabar com uma comunicação não iniciada.*

```
.........
il%=ij%
call Inicia(EndInibicao%,EndMsgUrgente%,il%)  <<<  Estabelece  ligação
if il%=0 then
        il%=ij%
        call termina(il%)                     <<<  Termina a ligação
end if                                             efectuada
.........
```

Figura 67. Contagem das estações existentes na rede.

Este algoritmo é implementado, no programa pai, pela chamada das outras primitivas (estabelecer ligação e terminar ligação), já referidas anteriormente. A sua implementação previu, quer uma pesquisa ao longo de todas as possíveis estações de quais as presentes, quer uma pesquisa singular; como também já foi exposto, o resultado de quais as estações presentes será acessível por diversos módulos de processamento, através de um ficheiro partilhado.

Falta referir a última hipótese de comunicação entre a estação central e as remotas, comunicação cuja efectivação, supervisão e controlo fica integralmente a cargo do utilizador; neste programa do módulo de comunicação, cabe ao operador do sistema efectuar o estabelecimento da ligação, o terminar dessa ligação, bem como o controlo de todo o tráfego que percorre a linha de comunicação.

Esta possibilidade de comunicação, acessível quer dos módulos escritos em QBasic (módulo operativo e de processamento de sinal), quer do módulo de diagnóstico de avarias permite, para além das acções relacionadas directamente com a ligação entre as estações, controlar diversas características do controlador de comunicações, a UART 8250; de entre essas tarefas salienta-se a possibilidade de definir a porta de

comunicação (em qualquer endereço) bem como o "baud rate" da linha de comunicação série.

Para além destas tarefas de gestão é ainda possível recorrer no microcomputador central a um menu de ajuda (figura 68), que informa de todas as potencialidades desta primitiva.

```
                                HELP
PARA OPERAR A PLACA REMOTA DE AQUISIÇÃO DE DADOS, INTRODUZA OS COMANDOS
COMO PROCEDE ATRAVÉS DE UM TERMINAL. (? ou ! para HELP)
PODE AINDA UTILIZAR ALGUNS COMANDOS INTERNOS:
              Alt-C  - Limpa o ecrã.
              Alt-N  - Altera a configuração da comunicação.
              Alt-R  - Estabelece comunicação com a placa.
              Alt-X  ou F10  - Termina o programa.
              PgDn   - Inicia captura de ficheiro
              PgUp   - Inicia envio de ficheiro
              Esc    - Termina a recepção ou envio de ficheiro
MENU PARA ALT-N:
+-----------+------+------+------+------+-------+-------+--------+
| baud rate | 1200 | 2400 | 4800 | 9600 | 19200 | 38400 | 57600  | 115200 |
| (prima só)| 12xx | 2xxx | 4xxx | 9xxx | 19xxx | 3xxxx | 5xxxx  | 11xxxx |
+-----------+------+------+------+------+-------+-------+--------+--------+
| endereço  |      |      |      | Prototype |  Espaço de I/O livre  |
| base      | COM1:|  COM2:|     |  card:    |  (multiplos de 8):    |
| (em hex.) |  3F8 |  2F8 |      |  300      |  330 a 370, e 380     |
+-----------+------+------+------+-----------+-----------------------+
| Núm. int. | COM1: | COM2: |  Ou outro espaço no intervalo  reservado   |
| (em hex.) |   C   |   B   |  (Só pode premir 2 caracteres)             |
+-----------+-------+-------+--------------------------------------------+
                                                      Prima uma tecla
```

Figura 68. Menu de ajuda do programa de comunicação manual.

Uma das principais características é contudo, a possibilidade de captar para um ficheiro todo o tráfego de caracteres recebidos, podendo por exemplo ser efectuado um comando de visualização da memória de dados a ser posteriormente utilizado pelos restantes módulos implementados, em especial os de processamento de sinais; além desta possibilidade existe, naturalmente a inversa, que corresponde a enviar pela linha de comunicação um ficheiro de texto. Não existindo qualquer restrição para as características do ficheiro a receber ou enviar, nada impede a utilização de um protocolo como o "intel" hexadecimal, que permite detectar alguns erros.

O início da captura de um ficheiro é efectuado a partir de uma opção seleccionada na microcomputador, sendo terminado pela selecção da tecla ESC; são capturados todos os caracteres que circulam na linha de comunicação e que são vistos no ecrã.

Esteja ou não seleccionada a opção de captura para um ficheiro da informação circulante, os caracteres que vão chegando podem fazê-lo a uma velocidade superior

àquela com que são apresentados (no ecrã ou no ecrã e no ficheiro de captura), dado que existindo velocidades da ordem dos 115 Kbaud não é possível garantir uma escrita tão rápida nem no ficheiro nem no ecrã.

```
;                          LE_STRING (Pública).
;Rotina que le para um buffer os caracteres vindos da porta série COM2:
LE_STRING:
      STI
      PUSH    DX
      ....    .....
L_S_3:MOV     CX,NUM_BYTES_BUF_COM
      CMP     CH,20H                  ;Se o "buffer" ultrapassou 2000 hex.
      JMP     NAO_TERMINA_STR
      JMP     FIM_LER_STRING          ;termina.
NAO_TERMINA_STR:
      MOV     DX,CS:PONT_BUF_COM
      MOV     BX,DX
      ADD     BX,CX
      MOV     [BX],AL                 ;Guarda caracter no buffer.
      INC     CS:NUM_BYTES_BUF_COM

      MOV     BX,CS:EXECUTANDO
      CMP     BX,0
      JE      EXEC_BACKGROUND         ;Está em "background"
      JMP     FIM_LER_STRING          ;Não está em "background"
EXEC_BACKGROUND:
      MOV     BX,CS:SEGMENTO
      MOV     DS,BX
      MOV     BX,CS:END_INIBICAO
      MOV     DX,DS:[BX]              ;Verifica se é permitido enviar
      CMP     DX,0                    ;caracteres para o ecrã (Inibição=0)
      .....   .....
ENVIAR_MSG_URGENCIA:
      PUSH    AX
      MOV     BX,CS:SEGMENTO
      MOV     ES,BX
      MOV     BX,CS:END_MSG_URGE
      MOV     AL,1
      MOV     ES:[BX],AL      ;Assinala a variavel do QBasic que ha uma uma mensagem
      MOV     DH,24           ;urgente no buffer.
      MOV     DL,27           ;Linha 25, Coluna 28 (Zona para alarme)
      MOV     BX,CS
      MOV     DS,BX
      MOV     BX,OFFSET MSG_URGENCIA
      MOV     CX,0CFH
      CALL    FAR PTR IMPRIME_STRING  ;Imprime mensagem de urgência
      POP     AX
FIM_LER_STRING:
      ....    ....
      IRET
```

Figura 69. Apresentação do procedimento do módulo de comunicação assíncrona que respondem à chegada de um caracter; como se pode verificar o caracter é escrito no "buffer" de recepção, sendo permitido em algumas circunstâncias a existência de concorrência

Para obviar a este problema, o módulo tem um funcionamento baseado em dois "buffers", correspondentes a duas zonas de memória distintas; quando o programa está em execução permite escrever no ecrã todos os caracteres que estão numa destas memórias, o "buffer" de apresentação, e ainda não foram apresentados; em simultâneo, sempre que chega um caracter é activada uma interrupção que faz com que seja actualizado a zona de memória relacionada com a recepção de caracteres, o "buffer" de recepção (ver figuras 69 e 70).

```
;=============================================================================
;                        SAI_STRING (Pública).
;Rotina que escreve no monitor os caracteres vindos da porta série COM2: e
;guardados num buffer.
;-----------------------------------------------------------------------------
SAI_STRING:
      CLI
      MOV     CX,NUM_BYTES_BUF_COM       ;Núm. bytes a mover, e o nº de bytes a
      MOV     NUM_BYTES_BUF_SCREEN,CX    ;imprimir no ecrã.
      MOV     NUM_BYTES_BUF_COM,0        ;No buffer de comunicação nao há caracteres.
      MOV     DX,PONT_BUF_SCREEN
      MOV     CX,PONT_BUF_COM
      MOV     PONT_BUF_SCREEN,CX         ;Vai trocar os buffers.
      MOV     PONT_BUF_COM,DX            ;O de comunicação passa a ser o do ecrã
      STI                                ;e vice-versa.
      MOV     CX,NUM_BYTES_BUF_SCREEN
      MOV     BX,PONT_BUF_SCREEN
      ADD     BX,CX
      MOV     [BX],2424H                 ;Põe o caracter $ para
      MOV     AH,09H                     ;delimitar a string.
      MOV     DX,PONT_BUF_SCREEN         ;Buffer a enviar para o ecrã.
      INT     21H                        ;No buffer do ecrã já nao há caracteres.
      CALL    FAR PTR ESCREVE_FILE       ;Se for caso disso escreve no ficheiro d
      MOV     NUM_BYTES_BUF_SCREEN,0     ;captura.
      RET
;=============================================================================
;                        ESCREVE_FILE
;Rotina que escreve caracteres recebidos no ficheiro.
;-----------------------------------------------------------------------------
ESCREVE_FILE:
      MOV     BX,OFFSET FILE_ABERTA
      MOV     AL,BYTE PTR [BX]
      CMP     AL,1
      JNE     FIM_ESCRITA_BUFFER
      MOV     BX,HANDLE
      MOV     CX,NUM_BYTES_BUF_SCREEN
      MOV     DX,PONT_BUF_SCREEN
      MOV     AH,40H
      INT     21H
FIM_ESCRITA_BUFFER:
      RET
```

Figura 70. Apresentação dos procedimentos do módulo de comunicação assíncrona (sem protocolo de ligação) que respondem à chegada de um caracter e que o escrevem no ecrã ou ficheiro.

Sempre que foi esgotada a informação existente no "buffer" de apresentação, é efectuada a troca entre eles; de notar que esta troca não corresponde a mudar os conteúdos destas memórias, mas sim a uma mudança do nome dos "buffers", isto é o de apresentação (que está vazio) passa a ser o de recepção e o de recepção anterior passa a ser o de apresentação.

Como se pode aperceber pelo programa escrito (figura 69) como resposta às interrupções por chegada de um caracter, existe a possibilidade de existir um certo nível de concorrência.

Esta pode ser descrita como implementada em duas etapas distintas: uma relativa ao avisar do programa pai de que foi recebida uma comunicação, provavelmente urgente, quando este não está a utilizar as rotinas de comunicação; a segunda etapa relaciona-se com a apresentação das comunicações a circular na linha no ecrã.

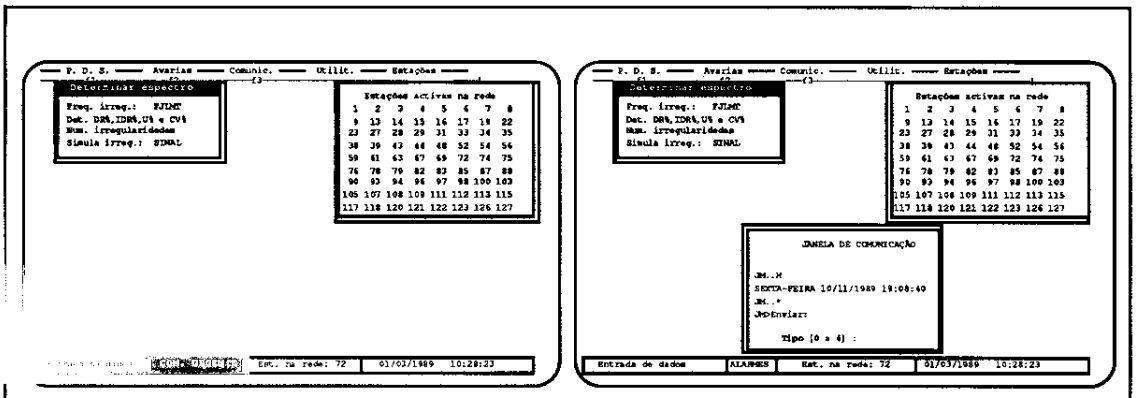


Figura 71. Modificação da linha de controlo, avisando da chegada de uma comunicação da estação remota (*à esquerda*); exemplo do funcionamento concorrente da janela de comunicação; no ecrã apresentado veêm-se 3 janelas em simultâneo: uma para seleccionar um programa do módulo de processamento de sinal (esq.), outra com as estações presentes na rede (canto sup. dir.) e a última (canto inf. dir.) apresentando a comunicação na linha.

Na primeira a nível de programa pai é modificado o conteúdo de uma variável que vai assinar na linha de controlo do ecrã (em caracteres a 'piscar') com o sentido de alertar o operador para a necessidade de prestar atenção à linha de comunicação (figura 71 esq.); o operador deverá então activar a janela de comunicações ou chamar o

módulo de comunicação para visualizar qual a comunicação recebida e eventualmente apresentar a resposta conveniente.

Relativamente à segunda etapa a concorrência é implementada a nível da janela de comunicação; esta apresenta numa parte do ecrã todos os caracteres recebidos pela linha, em simultâneo com a execução dos comandos entretanto seleccionados (figura 71 dir.).

Sendo esta estrutura de comunicação essencialmente virada para um funcionamento dependente do operador e, além disso, uma ligação assíncrona em que o operador trabalha com a estação remota como utilizando um terminal, não parece muito importante e muito menos imprescindível a existência de concorrência a nível de comunicação.

Esta foi implementada, também depois de ter sido construído todo o módulo de comunicação, essencialmente com vista a permitir ao utilizador incumbir a estação remota de tarefas demoradas, voltar ao módulo principal na estação central e aí executar as tarefas que pretender sendo alertado do fim da tarefa na estação remota. Com esta possibilidade o utilizador poderá retomar a comunicação com a estação remota.

10.5.2 Módulo de comunicação usando protocolo orientado ao caracter. (Usando as características da *Estrutura 2*).

Embora não definitivamente abandonada a solução apresentada anteriormente (uso de DMA) como aplicação a implementar futuramente, optou-se, no presente, por definir e construir um protocolo de comunicação orientado ao caracter; esta solução provou ser também bastante eficiente no cumprimento das necessidades impostas a este módulo para cumprir as características mínimas definidas para o funcionamento utilizando a *Estrutura 2* já mencionada em capítulos anteriores; com este protocolo, poderemos atingir um funcionamento mais versátil, na medida em que, tanto é possível controlar as estações remotas a partir da rede (*tal como um terminal virtual*), como é possível automatizar este controlo.

Dividiu-se o módulo de comunicações em submódulos:

A - Movimentação de dados entre memórias.

Esta movimentação corresponde à passagem da informação recebida num "buffer" (memória tampão) para a zona de memória do destinatário e, em sentido inverso, para a transmissão de informação.

B - Comunicação série bi-direccional com o exterior.

Este submódulo está relacionado com o envio e recepção de informação da linha de comunicação para a "memória tampão", seguindo o protocolo já definido noutro capítulo.

C - Teste de integridade dos dados.

Este nível está associado à verificação da correcção e integridade dos dados recebidos.

Com vista a cumprir os objectivos de minimizar o tempo despendido na comunicação, optou-se por garantir a existência de "concorrência" entre o processo em curso no microcomputador central e o submódulo B de comunicação, uma vez que este é bastante dependente do processo físico, mas não necessita da intervenção directa do processador central durante largos períodos de tempo para ser correctamente cumprido.

Enquadram-se tipicamente nesta definição primitivas tais como *transmitir "frame"*, *receber "frame"* ou mesmo *esgotar tempo limite* para a detecção de uma resposta. Estas primitivas podem associar-se a interrupções físicas que originarão uma acção específica apenas quando ocorrerem (figura 72).

Apesar destas considerações realça-se que a especificação do algoritmo, desenvolvimento e teste se processou sem necessitarmos de implementar, *desde a fase de concepção e desenvolvimento inicial*, este tipo de concorrência que apenas foi efectivada depois da verificação do correcto funcionamento do módulo. A utilização de uma determinada primitiva de comunicação não implica, por si só, a necessidade de concorrência; esta só será utilizada no caso da primitiva recorrer a outras que, tal como descrito, ocupem tempo inútil para a efectivação da comunicação.

A chamada de uma primitiva global de comunicação (*da camada de controlo de mensagem*), tal como estabelecer ligação, enviar mensagem, etc., inicia-se passando valores do módulo que chama para essa primitiva, correspondendo à entrada numa das camadas *'inferiores'* do módulo de comunicação; esta ***é a única forma*** autorizada de o fazer.

Conforme o nível da camada (primitiva) chamada assim esta chamará, por processo idêntico, a subcamada (subprimitiva) que necessitar e esta seguirá uma metodologia idêntica.

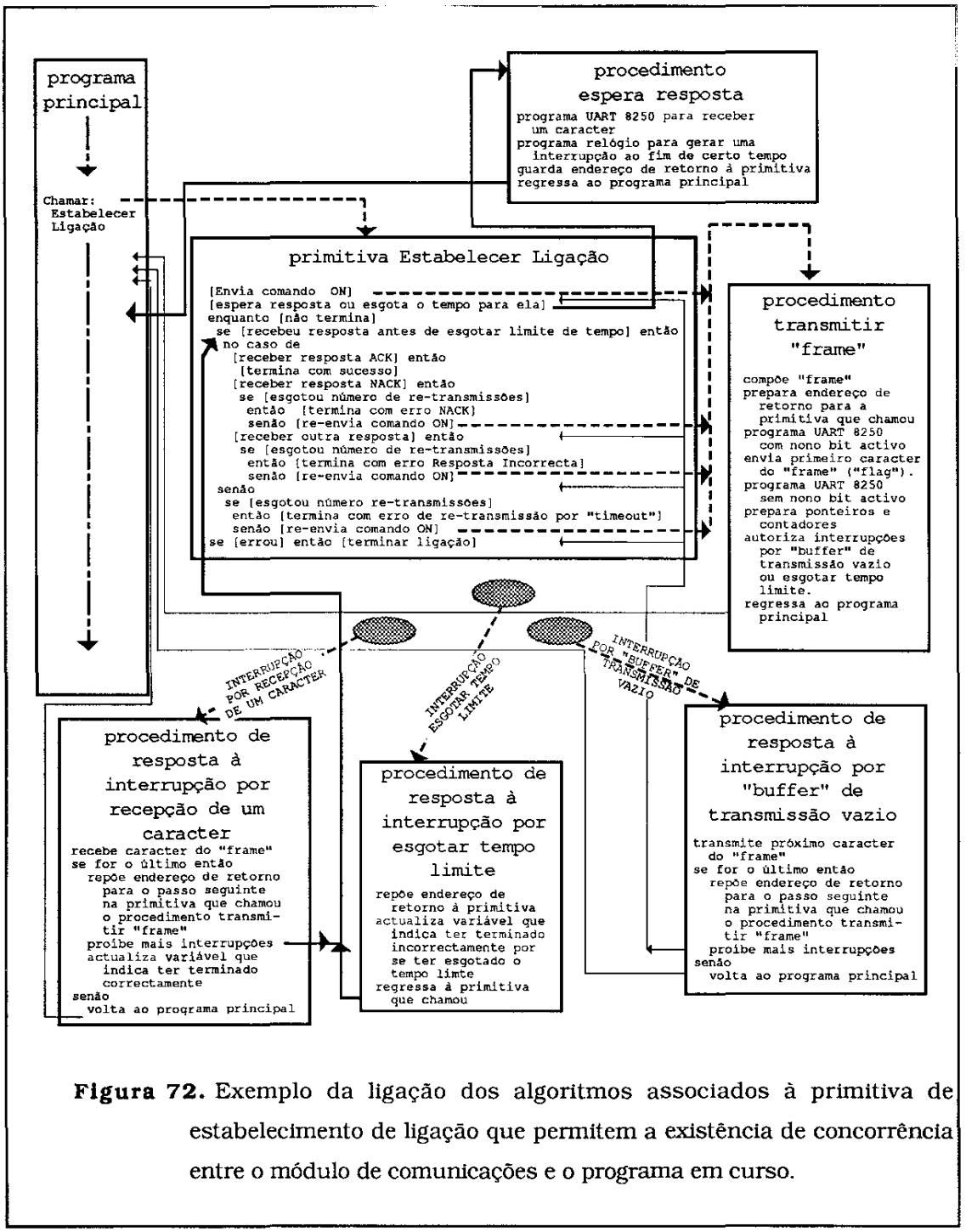


Figura 72. Exemplo da ligação dos algoritmos associados à primitiva de estabelecimento de ligação que permitem a existência de concorrência entre o módulo de comunicações e o programa em curso.

Dada a consideração de possibilidade de concorrência a chamada de uma primitiva não fornece o resultado da comunicação na saída. Ao invés, é na chamada que é dada toda a informação sobre o local onde o processo de comunicações colocará o

resultado. Quem apelou à primitiva terá de testar, de quando em quando, para saber se já possui o resultado pretendido.

No módulo de comunicação tudo se passa de forma diferente; a chamada de uma primitiva inicia a sua execução até se atingir um procedimento que é considerado para funcionar concorrentemente. Este procedimento inicializa os instrumentos de controlo de concorrência (tempo limite, transmissão ou recepção de "frame"), sendo então abandonado o módulo de comunicações, permitindo assim ao processo pai (o que chamou) continuar o seu processamento. Quando um dos instrumentos de controlo o desejar retoma-se de novo ao módulo de comunicações, no ponto onde estava. O procedimento em causa continuará até terminar ou, após re-inicializar os instrumentos de controlo da concorrência; de seguida abandona o módulo estando de novo previsto o regresso por imposição de algum dos instrumentos de controlo.

Quando com este processo de "vai-vem" terminar o processamento que requer concorrência, serão dispensados os instrumentos de controlo desta e o algoritmo prosseguirá até ao seu termo ou até aparecer de novo uma situação que implique o uso de concorrência. Exemplificando, para o caso de *Estabelecer Ligação* na camada de *Controlo de Ligação*, apenas há necessidade de concorrência nos passos de envio ou reenvio de comandos e no passo de esperar resposta, pelo que só se verifica essa situação quando atingimos esses pontos no algoritmo.

Os instrumentos de controlo de concorrência são assim relacionados com a transmissão/recepção de caracteres e com um relógio de tempo real. Concretizando, são instrumentos de concorrência as interrupções geradas por três causas:

A - Quando entrou no procedimento de transmitir um "frame" (enviar ou reenviar comando) é programado o controlador de comunicações para gerar uma interrupção quando este estiver pronto a enviar um caracter. Esta interrupção é desactivada quando todo o "frame" for enviado.

Após a autorização da interrupção o procedimento será abandonado guardando-se no entanto a informação sobre o ponto onde regressar, de tal forma que, quando o caracter for transmitido e o controlador estiver pronto a enviar outro, se regresse a este ponto do programa.

B - As interrupções geradas por um relógio (já existente em todos os AT) sempre que não seja recebida uma resposta dentro do tempo limite considerado para o efeito. Esta situação, que não pode ser automaticamente associada a um erro de comunicação deriva de, após

a transmissão de um comando através da linha série, não ter sido ainda recebida uma resposta. Esta interrupção é autorizada quando se chegar a um passo do procedimento (primitiva) em que se espera a recepção de uma resposta ou o esgotar do tempo limite. Se for esgotado o tempo limite antes da detecção de um "frame" resposta, o relógio originará uma interrupção que ao ser atendida originará uma acção de recuperação, prevista na especificação do algoritmo da primitiva.

C - A terceira interrupção está intimamente associada à anterior. Está ligada à chegada de um caracter pela linha de comunicação. Sempre que chegar um caracter, que ainda não corresponda ao último caracter do "frame", este é guardado e são de novo autorizadas estas interrupções (B e C). Se for esgotado o tempo para a recepção da resposta procede-se como explicitado anteriormente (B). Se ocorrer a interrupção C, e esta corresponder à chegada do último caracter do "frame", desautoriza-se esta interrupção bem como a devida ao relógio e prossegue-se o procedimento sequencialmente até ao fim ou até atingirmos uma nova situação de concorrência, caso em que se re-inicializarão as interrupções necessárias.

Como já foi referido, cabe ao programa que chamou o módulo de comunicação testar se já tem informação pretendida. O teste de fim da comunicação pretendida será efectuado por informação passada por memória pelo módulo anteriormente descrito.

Dada a alta velocidade que é necessário garantir para a comunicação bem como o modo "half-duplex" inerente à comunicação por um par de fios foi desenvolvido um módulo de comunicação próprio, não recorrendo às facilidades do sistema operativo, que aliás se verificaram não ser apropriadas.

Este programa desenvolvido em "assembler" do microprocessador 8086 permite programar a placa de comunicação com diversos "baud rate" (entre 1.2 e 115.2 Kbaud), sendo a recepção de caracteres efectuada por intermédio de interrupção.

10.5.2.1 Descrição dos algoritmos das primitivas de comunicação.

Descrevem-se de seguida os algoritmos genéricos das primitivas de comunicação implementadas no microcomputador central.

Estes algoritmos correspondem a uma aplicação que funcionará concorrentemente com o "software" em uso, recorrendo a interrupções geradas por um

relógio, bem como as originada por terem sido recebidos caracteres ou o periférico de saída estar pronto a enviar caracteres.

A) - Definição da primitiva [Estabelecer ligação]:

```
[Envia comando  ON]                                <<<<<  procedimento  tipo  I
[espera resposta ou esgota o tempo para ela]       <<<<<  procedimento  tipo  I
enquanto [não termina]
   se [recebeu resposta antes de esgotar limite de tempo] então
      no caso de
         [receber resposta ACK] então
            [termina com sucesso]
         [receber resposta NACK] então
            se [esgotou número de retransmissões]
               então  [termina com erro NACK]
               senão [reenvia comando ON]          <<<<<  procedimento  tipo  I
         [receber outra resposta] então
            se [esgotou número de retransmissões]
               então [termina com erro Resposta Incorrecta]
               senão [reenvia comando ON]          <<<<<  procedimento  tipo  I
   senão
      se [esgotou número retransmissões]
         então [termina com erro de retransmissão por "time-out"]
         senão [reenvia comando ON]                <<<<<  procedimento  tipo  I
se [errou] então [terminar ligação]
```

Figura 73. Algoritmo da primitiva estabelecer ligação. Os procedimentos do tipo I são os que aceitam concorrência; quando é chamada esta primitiva, o seu algoritmo desenrola-se sequencialmente até se atingirem esses procedimentos, altura em que é abandonado este módulo, voltando ao programa principal, até surgir uma interrupção (ver figura 72).

Esta primitiva, com o algoritmo apresentado na figura 73, permite estabelecer uma ligação lógica entre o mestre ("master") o microcomputador central, e as estações escravas ("slaves") correspondendo às estações remotas de aquisição e controlo.

Após a efectivação desta ligação, proceder-se-á, unicamente por decisão do mestre, às transacções de dados. Estas transacções que podem incluir envio de dados, pedidos de dados com as correspondentes respostas, acabarão sempre com o terminar da ligação a efectuar, também, por decisão do mestre.

Com vista a melhor ilustrar os aspectos relacionados com a concorrência, apresentou-se na figura 72 uma esquematização desta característica. Embora exemplificado para esta primitiva, esta estrutura mantém-se, no essencial, para as restantes primitivas (transacção de dados e terminar de ligação).

B) - Definição da primitiva [Efectuar transacção]:

```
[Preparar comunicação a efectuar]
enquanto [não termina transmissão]
   se [tem DATA para enviar e não excedeu o nº de DATA sem "acknowledge"]
   então
      [Transacção de comando DATA]          <<<< procedimento do tipo II
      se [transacção foi correcta]
         então [actualiza ponteiros e contadores]
         senão [termina com erro detectado na transacção]
   senão
      se [tem DATA para enviar e já esgotou tempo limite sem resposta] então
         [termina transmissão e recepção com erro de "time-out"]
      senão
         [termina transmissão com sucesso]
se [tem DATA para receber] então
   [Transacção de comando RR]               <<<< procedimento do tipo II
   se [transmitiu RR sem erro] então
      enquanto [não termina recepção]
         [espera resposta ou esgota o tempo para ela]    <<<< procedimento
                                                         <<<< do tipo I
         se [recebeu resposta antes de esgotar o tempo limite] então
            no caso de
               [receber resposta RR] então [termina recepção com sucesso]
               [receber resposta RNR] então [termina recepção com erro de RNR]
               [receber resposta DATA] então
                  se [frame recebido correcto] então
                     [envia comando ACK]       <<<< procedimento do tipo I
                     [prepara campo de informação recebido]
                  senão
                     [envia comando NACK]
               [receber outra resposta] então
                  [termina recepção c/ erro de Resposta Incorrecta]
         senão
            se [esgotou número retransmissões] então
               [termina recepção com erro de "time-out"]
               senão
               [Transacção de comando RR]  <<<< procedimento do tipo II
```

Figura 74. Algoritmo da primitiva efectuar transacção, residente no microcomputador central; os procedimentos do tipo I são os que aceitam concorrência. Os procedimentos do tipo II correspondem a 'subprimitivas' que efectuam um conjunto complexo de trocas de "frames" e a 'refinar' de seguida; contêm procedimentos que são do tipo I a par com outros de tipos diferentes.

Nas figuras 74, 75 e 76 apresentam-se os algoritmos das primitivas responsáveis pela efectivação da transferência de informação. Especificação mais pormenorizada de alguns dos procedimentos é apresentada e discutida no apêndice de comunicações.

```
[envia comando DATA]
[espera resposta ou esgota o tempo para ela]
enquanto [não termina]
   se [recebeu resposta antes de esgotar tempo limite] então
      no caso de
         [receber resposta ACK]  então  [actualiza nº DATA sem confirmação]
         [receber resposta NACK] então
            se [esgotou número de retransmissões]
               então  [termina com erro NACK]
               senão [reenvia comando DATA]
         [receber resposta RNR] então
            se [esgotou nº RNR ] então
               [termina com erro RNR em nº elevado]
         [receber resposta RR] então  [actualiza nº DATA sem confirmação]
         [receber outra resposta] então
            se [esgotou número de retransmissões]
               então  [termina com erro de Resposta Incorrecta]
               senão [reenvia comando DATA]
```

Figura 75. Especificação do procedimento transacção do comando DATA, usado na primitiva de transacção de dados (figura 74).

```
[envia comando RR]
[espera resposta ou esgota o tempo para ela]
enquanto [não termina]
   se [recebeu resposta antes de esgotar o tempo limite] então
      no caso de
         [receber resposta ACK] então [termina com sucesso]
         [receber resposta RNR] então [termina com erro RNR]
         [receber resposta NACK] então
            se [esgotou número de retransmissões]
               então [termina com erro NACK]
               senão [reenvia comando RR]
         [outra resposta] então
            se [esgotou número de retransmissões]
               então [termina com erro Resposta Incorrecta]
               senão [reenvia comando RR]
   senão
      se [esgotou número retransmissões]
         então [termina com erro de retransmissão]
            senão [reenvia comando RR]
```

Figura 76. Especificação do procedimento transacção do comando RR ("Receiver Ready"), utilizado na primitiva transacção de dados (figura 74).

Como já foi referido, qualquer transacção é sempre efectuada sob pedido do mestre. Assim, quando este tem necessidade de receber dados terá de enviar um pedido de informação para a estação com que está presentemente ligado. Se a estação de que pretende obter essa informação não estiver ligada logicamente (de notar que nada impede a existência de mais de uma ligação lógica simultânea), terá de efectuar nova

ligação com a estação pretendida. De realçar o facto de não ser necessário aguardar pela existência de dados a enviar pelo mestre para efectuar este pedido e, por outro lado, a estação remota não possuir a informação pretendida, não ser impeditivo desta consulta.

C) - Definição da primitiva [Terminar ligação]:

```
[Envia comando OFF]
[espera resposta ou esgota o tempo para ela]
enquanto [não termina]
   se [recebeu resposta antes de esgotar limite de tempo] então
      no caso de
         [receber resposta ACK] então [termina com sucesso]
         [receber resposta NACK] então
            se [esgotou número de retransmissões]
               então [termina com erro NACK]
               senão [reenvia comando OFF]
         [receber outra resposta] então
            se [esgotou número retransmissões]
               então [termina com erro Resposta Incorrecta]
               senão [reenvia comando OFF]
   senão
      se [esgotou número retransmissões]
         então [termina com erro Resposta Incorrecta]
         senão [reenvia comando OFF]
se [terminou com erro] então
   se [não ultrapassou o limite de tentativas de desligar] então
      [Terminar ligação]
```

Figura 77. Especificação do algoritmo da primitiva terminar ligação

Esta primitiva apresentada na figura 77, permite terminar a existência de uma ligação lógica entre 2 nós comunicantes. Mesmo que, por lapso ou não, circulem informações na linha após o fim da ligação, serão sempre ignoradas pelo receptor, excepto se for o "frame" correspondente ao estabelecimento de ligação.

A estação considera que essa informação, embora sendo-lhe destinada, não deve ser considerada; as estações remotas permanecem no estado desligado até receberem o comando de estabelecimento de ligação.

10.6 Módulo de detecção automática de avarias.

Este módulo que recorre a técnicas de inteligência artificial foi escrito usando a linguagem PROLOG e implementa um sistema pericial ("expert system"), seguindo alguns princípios gerais já expostos noutro capítulo.

Este módulo assenta numa base de conhecimentos (*base de dados*) que é permanentemente actualizada com informação relativa a avarias e acções que as resolveram. Por pesquisa na base de conhecimentos é possível obter informação sobre a anterior ocorrência de avarias similares (o grau de similitude é definido pelos sintomas comuns) e quais as acções que mais frequentemente as resolveram.

Cada avaria é definida como um conjunto de sintomas. Esses sintomas estão guardados na base segundo uma estrutura em árvore.

Quando ocorre uma falha, o utilizador percorre o conjunto de sintomas já detectados anteriormente noutras avarias ou falhas idênticas. A pesquisa/actualização, no sentido de simplificar a organização da informação, é efectuada percorrendo uma árvore de sintomas; o utilizador selecciona então os sintomas da falha sendo a base actualizada, mesmo que estes nunca tenham sido detectados.

Feita esta selecção são apresentadas soluções já adoptadas anteriormente para a mesma falha, bem como soluções já adoptadas para a solução de falhas similares, isto é, falhas com pelo menos um sintoma comum. São ainda apresentadas as probabilidades de sucesso conseguidos com acções utilizadas para solucionar problemas que sejam constituídos por alguns ou todos os sintomas que constituem a presente avaria (de notar que o número de sintomas não é necessariamente o mesmo).

Para um utilizador com mais conhecimentos e capacidade interpretativa de resultados matemáticos estão também disponíveis os valores de correlação entre a avaria escolhida e outras avarias, relacionando-as conforme o número de sintomas comuns entre elas. O utilizador seleccionará, percorrendo o conjunto de acções disponíveis na base de conhecimentos, qual a acção ou o conjunto de acções (e nesse caso também a relação lógica entre cada uma), que resolveram a avaria (*ou falha*). Conforme as decisões tomadas será actualizada a base de conhecimentos.

Convém realçar o caracter 'autoconstrutivo' da base de conhecimentos, que é actualizada à medida que ocorrem os problemas e são tomadas as decisões relativas à sua solução. O programa construído não necessita de qualquer *base de conhecimentos* inicial para apresentar sugestões; naturalmente que, na 1ª tentativa — com a base em branco — não existirá resposta, quanto mais uma resposta válida ! (*na verdade estas respostas só terão alguma fiabilidade quando a base tiver uma certa dimensão*).

Efectivamente o que foi implementado é um *sistema pericial* ("expert system"), e não um programa de armazenamento e tratamento de ***uma determinada base*** de dados.

Não necessitando de qualquer informação inicial, o sistema pode ser utilizado numa grande diversidade de casos, ou melhor, tipo de casos; é por esta razão que em diversos exemplos, ***e apenas por razões de clareza***, se optou por apresentar exemplos de uma *base* de doenças, provavelmente mais facilmente perceptível para quem não está acostumado à utilização dos termos e técnicas têxteis.

Para ser possível estabelecer estas relações entre avarias e acções foi necessário definir uma estrutura clara para a base de conhecimentos a utilizar; esta, apresentada esquematicamente na figura 78, passa pela existência de 3 grupos lógicos que podemos considerar individualizados sob a forma de bases de dados correspondentes aos sintomas, avarias e acções.

A primeira contém toda a informação relativa a ocorrências distintas (*individuais*) detectadas, enquanto a segunda permite definir cada avaria como um conjunto desses sintomas (é assim definida como um conjunto de elementos da primeira, isto é, uma avaria= {lista} de sintoma).

As acções que solucionam (ou solucionaram) uma avaria constituem a terceira base de dados; as ligações de cada avaria a uma determinada acção têm um valor associado (dependente do nº de vezes que foram utilizadas com sucesso) que nos permite estabelecer uma probabilidade para o seu sucesso futuro (*previsão*). Em certa medida podemos considerar estas probabilidades como constituindo, na prática, uma base de dados suplementar.

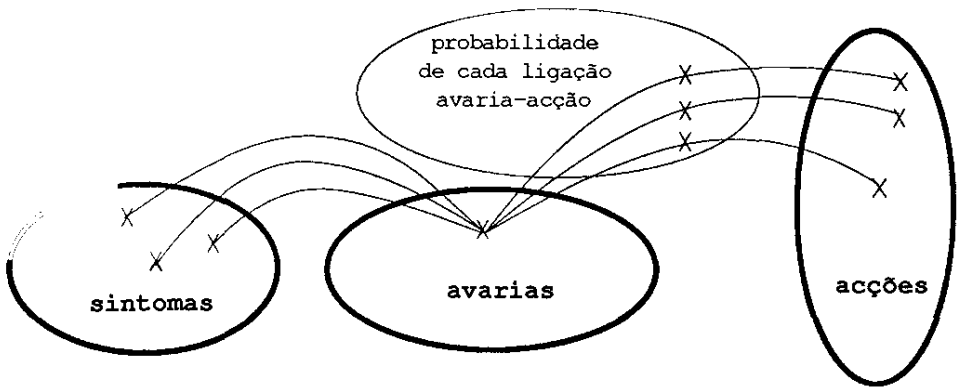


Figura 78. Estrutura das bases de dados e suas ligações com vista ao estabelecimento de uma relação ***sintomas - avaria - acções*** capaz de solucionar a avaria.

De notar que este tipo de estrutura tem esta configuração na medida em que não é possível conhecer à partida qual a probabilidade de uma acção ter sucesso; este facto está directamente relacionado com a condição de não ser possível estabelecer com precisão uma relação causa-efeito. Podemos considerar que a não existência desta relação sob a tradicional forma matemática, obriga à implementação de um modelo deste tipo, permanentemente actualizado, numa relação baseada na experiência do funcionamento do sistema e no conhecimento que os operadores dele possuem.

A base de conhecimentos reside em memória, sendo periódica e automaticamente efectuado um "backup" dessa base para disco. O intervalo entre cada "backup" pode ser modificado pelo utilizador, tornando o sistema facilmente adaptável às caracteristicas de utilização e/ou da experiência do operador em lidar com ferramentas informáticas.

Neste módulo é possível imprimir relatórios estatisticos sobre a informação residente na base de conhecimentos. Este relatório permite visualizar os sintomas avarias e acções já existentes, bem como as relações lógicas entre as diversas acções.

Apresenta também informação que correlaciona as avarias e as acções entre si, associando valores probabilisticos relativos à utilização de determinadas acções para solucionar determinadas avarias.

Como referido anteriormente, é possível aceder directamente ao módulo de comunicação, permitindo entabular uma comunicação com as estações remotas.

Apresenta-se de seguida a informação contida numa base de dados, utilizada a título experimental, e de tamanho reduzido bem como alguns pormenores das saídas apresentadas por este programa para essa base.

Para o efeito considerou-se uma base em que cada "avaria" representa uma doença constituída por diversos sintomas fisicos; a utilização desta base permitiu validar o algoritmo, tanto mais que existe uma similaridade entre o diagnóstico médico e o diagnóstico das avarias de uma máquina.

Para cada uma destas doenças poderá existir informação relativa ao cuidados a ter para se conseguir a "cura".

Todos estes dados correspondem a cláusulas descritas em linguagem PROLOG o que permite a sua fácil associação em estruturas de dados mais complexas como árvores e listas.

Para cada avaria é guardada a informação sobre a **lista de sintomas** que a constituem, sintomas esses também descrito nessa base (figura 79). De realçar que,

com vista a mais facilmente estruturar a informação residente estes podem estar associados sobre a forma de árvore de sintomas. Além da base de dados é apresentado nessa figura o ecrã utilizável para a selecção dos sintomas que constituem uma avaria.

a) Descrição da base de dados com sintomas:

```
sintoma(0,[28,29,30,31,34,35,36,37,38,40],ramo) :-
    write($CABEÇA DA LISTA DE SINTOMAS$).
sintoma(28,[1,5,15,16,19,20],ramo) :-
    write($GRUPO DE SINTOMAS: FEBRES$).
sintoma(29,[3,9,10,12],ramo) :-
    write($GRUPO DE SINTOMAS: DORES$).
    ............
sintoma(35,[21,22],ramo) :-
    write($GRUPO DE SINTOMAS: PROBLEMAS DE VISÃO$).
    ............
sintoma(38,[14],ramo) :-
    write($GRUPO DE SINTOMAS: PROBLEMAS DE FALTA DE EQUILIBRIO$).
sintoma(40,[],ramo) :-
    write($PROBLEMAS DE LOCOMOÇÃO$).
sintoma(1,[],folha) :-
    write($Febre superior a 38.5$).
sintoma(2,[],folha) :-
    write($Corpo "partido"$).
    ............
sintoma(9,[],folha) :-
    write($Dor de cabeça$).
sintoma(10,[],folha) :-
    write($Dor de garganta$).
    ............
sintoma(26,[],folha) :-
    write($Rouquidão$).
sintoma(27,[],folha) :-
    write($Dificuldade em engolir$).
    ............
```

```
avaria(1,[1,2]).
avaria(2,[1,2,3,4]).
avaria(3,[6]).
avaria(4,[7,8]).
avaria(5,[1,6,8]).
avaria(6,[5,8]).
avaria(7,[2,9,10,11,15,16]).
avaria(8,[3]).
avaria(9,[24]).
avaria(10,[9]).
avaria(11,[25,26,27]).
avaria(12,[7,13]).
avaria(13,[11,16,19]).
```

```
numero_de_vezes(avaria(2),6).
numero_de_vezes(avaria(4),2).
numero_de_vezes(avaria(5),1).
numero_de_vezes(avaria(3),4).
numero_de_vezes(avaria(6),1).
numero_de_vezes(avaria(7),3).
numero_de_vezes(avaria(1),5).
numero_de_vezes(avaria(9),1).
numero_de_vezes(avaria(10),4).
numero_de_vezes(avaria(11),1).
numero_de_vezes(avaria(12),1).
numero_de_vezes(avaria(8),20).
numero_de_vezes(avaria(13),1).
```

b) Menus de selecção de sintoma:

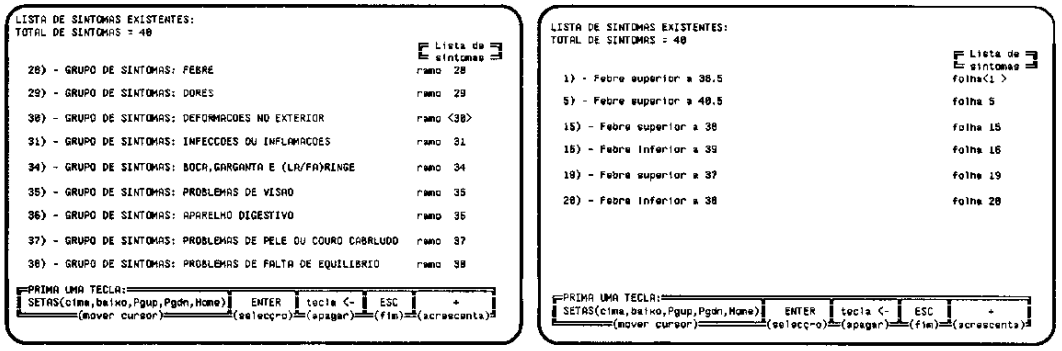


Figura 79. Lista de sintomas e avarias existentes na base de dados; a) cada avaria é definida por uma lista de sintomas, que seja uma ***folha*** da árvore. A alínea b) apresenta os ecrãs visualizados para a introdução de sintomas.

A selecção dos dados relativos a cada avaria é assim efectuada através da pesquisa 'numa árvore' em que os sintomas pertencem a um ramo; esta solução foi utilizada com vista a facilitar o trabalho de selecção dos sintomas que constituirão cada avaria. Ainda com este objectivo é possível modificar a estrutura desta árvore (base de dados de sintomas) sem introduzir obrigatoriamente novas avarias, tendo em vista garantir a gestão desta estrutura.

É também possível efectuar a introdução de um sintoma sem que este faça ***ainda*** parte de qualquer avaria; esta possibilidade fica a dever-se a ter-se verificado com frequência que surgindo um determinado sintoma é possível associar-lhe (por experiência acumulada do operador) um outro, complementar, que se prevê vir a ocorrer; com este método pode-se efectuar um trabalho inicial intensivo construindo uma base que possa apresentar resultados minimamente fiável, logo à partida.

Para além da introdução destes sintomas é possível modificar quer os sintomas quer as suas ligações, isto é, pode-se modificar o grupo a que pertence um dado sintoma, conseguindo-se assim evoluir duma estrutura simples para uma mais complexa, sem afectar as relações que associam esse sintomas a avarias já estudadas.

É guardada informação sobre o **número de ocorrências** das diversas avarias com vista a ser possível apresentar alguns relatórios estatísticos sobre os problemas já detectados.

Cada acção (figura 80) é guardada na base de dados sob uma das duas formas possíveis: uma com a descrição precisa da 'actividade' que deve provocar com vista à resolução do problema detectado; outra em que a acção é guardado sob a forma de **uma associação de outras acções**: neste caso, a associação recorre a diversas funções lógicas (*e*, *ou* e *ou exclusivo*)entre acções já existentes na base.

Está também ao dispor na base de dados, informação relativa às probabilidades de uma acção resolver uma avaria, o que está directamente relacionado com o **número de vezes que uma acção** foi seleccionada.

Quando o utilizador acabou de seleccionar os sintomas que caracterizaram a avaria (*doença*), o programa apresenta (após as necessárias validações e confirmações, e se o operador pretender), informação sobre as acções já utilizadas para resolver a avaria detectada. Como informação complementar é mostrada a percentagem de sucesso que se prevê estar associada a cada uma das acções já utilizadas para resolver o problema.

Ainda numa perspectiva de utilização por pessoas menos qualificadas, é apresentada informação de avarias que possuem ***parte ou todos*** os sintomas que constituiam a definição da avaria presente (figura 80).

Após esta fase é efectuada a selecção da acção capaz de resolver este problema; por vezes não é possível identificar uma acção individual que resolva a avaria. Para prever esta situação é possível associar, por intermédio de relações lógicas, diversas acções. Para facilitar a selecção de cada acção é apresentada, a par com a descrição de

cada uma, o valor probabilístico desta acção resolver a avaria (*com base na experiência anterior*).

a) Acções e probabilidades de cada acção resolver uma avaria:

```
accao(1) :-
    pe(A,B),
    accao(2),
    e(A,B),
    accao(3),
    pd.
.........
accao(6) :-
    write($Administrar um antibiótico$).
accao(7) :-
    write($Administrar um analgésico$).
.........
accao(35) :-
    pe(A,B),
    accao(1),
    ou(A,B),
    accao(5),
    pd.
```

```
probabilidade(avaria(1,A),accao(1),1).
probabilidade(avaria(2,A),accao(5),4).
.............................
probabilidade(avaria(8,A),accao(5),2).
probabilidade(avaria(8,A),accao(29),1).
probabilidade(avaria(8,A),accao(2),3).
probabilidade(avaria(8,A),accao(7),2).
probabilidade(avaria(8,A),accao(27),4).
probabilidade(avaria(8,A),accao(28),5).
.............................
probabilidade(avaria(13,A),accao(35),1).
```

b) Ecrãs apresentados ao utilizador após a introdução da avaria:

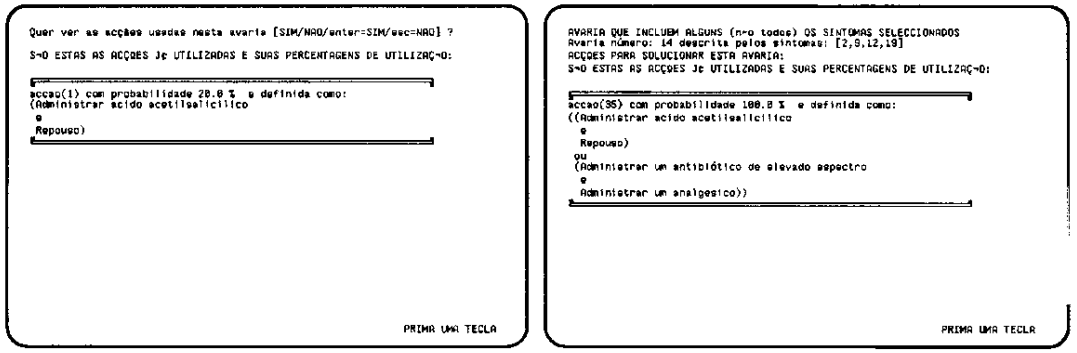


Figura 80. Aspectos base com informação sobre as acções existentes. Na a) apresenta-se a base de conhecimentos das acções; Na b) mostram-se os ecrãs apresentados ao operador após ter seleccionado a avaria (*doença*); aparecem alguns valores probabilísticos relacionados com a experiência passada.

Por fim existe ainda informação relativa à gestão geral da base, nomeadamente o número total de avarias, acções e sintomas bem como dados relativos à sua criação, designadamente o nome da directoria, o instante de criação e o último instante em que foi actualizada.

Na figura 81 apresenta-se um exemplo dessa informação; de notar que, para além desses dados fazem também parte da base a definição das relações lógicas possíveis para a associação de acções; estas foram incluídas na *base de conhecimentos* (ao ser inicializada é criada toda esta informação) por motivo de separar o mais possível a informação correspondente a ***dados*** da relativa ao "programa" propriamente dito.

```
intervalo(1800.0).
ultimo_backup([1989,4,26,18,10,1,76]).
inicio_relogio([1987,5,29,15,26,6,19]).
base($C:\JM\TEMP\DADOS\DOENCASX.ARI$).
e(A,B) :-
    nl,
    tget(C,D),
    tmove(C,B),
    write($e $),
    nl,
    tget(E,F),
    tmove(E,B).
ou(A,B) :-
    nl,
    tget(C,D),
    tmove(C,B),
    write($ou $),
    nl,
    tget(E,F),
    tmove(E,B).
xou(A,B) :-
    nl,
    tget(C,D),
    tmove(C,B),
    write($ou exclusivo $),
    nl,
    tget(E,F),
    tmove(E,B).
pe(A,B) :-
    write($($),
    tget(A,B).
pd :-
    write($)$).
numero_sintomas(40).
numero_avarias(13).
numero_accoes(35).
```

Figura 81. Dados complementares existentes na base de dados.

Com base nos dados gerais sobre a base já referidos, este módulo tem uma opção que permite ao utilizador conhecer o estado actual de toda a base de dados, informando sobre os sintomas existentes, a sua descrição e as avarias em que aparecem, sob a forma apresentada na figura 82 e seguintes. Este resultado pode ser apresentado quer no ecrã quer numa impressora quer num ficheiro.

```
Nome da base: C:\JM\JM_ARITY\DADOS\DOENCASX.ARI
Intervalo entre "backups" automaticos (seg): 1800.0
Base foi criada em: 29/5/1987 15:26:06
Ultimo "backup": 26/4/1989 15:23:52
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
Total de sintomas: 40
Total de avarias: 13
Total de accoes: 35
--------------------------------------------------------------------------------
```

Figura 82. Informação geral sobre a base em utilização.

Apresentada a estrutura da base de dados (ou de conhecimentos), vamos analisar os tipos de resultados apresentados e o interface entre este módulo e o utilizador.

Para o efeito, vamos basear-nos em exemplos que utilizam a informação existentes na base experimental utilizada ao longo deste texto.

Consideremos então exemplos de avarias bem como as respectivas acções de correcção a que se associam as probabilidades de sucesso destas (como mencionado estas decorrem da experiência da utilização das acções anteriormente); assim consideremos as seguintes avarias, de entre um lote de outras já introduzidas na base,

2, 7, 11 e 13

e que são constituídas por alguns sintomas de uma lista também visualizável utilizando o programa na opção de saída de resultados estatísticos (figura 83).

```
--------------------------------------------------------------------------------
|Sintoma|     Descricao do sintoma                    |Avarias c/ sintoma  |
+-------+---------------------------------------------+--------------------+
|   1   |Febre superior a 38.5                        | 1,2,5,             |
|   2   |Corpo "partido"                              | 1,2,7,             |
|   3   |Dor de dentes, nao focalizada                | 2,8,               |
|   4   |Inchaço na boca visivel exteriormente        | 2,                 |
|   5   |Febre superior a 40.5                        | 6,                 |
|   6   |Inchaço no baixo ventre                      | 3,5,               |
|   7   |Coceira interdigital                         | 4,12,              |
|   8   |Azia                                         | 4,5,6,             |
|   9   |Dor de cabeça                                | 7,10,              |
|  10   |Dor de garganta                              | 7,                 |
|  11   |Inflamação da garganta                       | 7,13,              |
|  12   |Dor nas articulacoes                         |                    |
|  13   |Queda de cabelo                              | 12,                |
|  14   |Tonturas                                     |                    |
|  15   |Febre superior a 38                          | 7,                 |
|  16   |Febre inferior a 39                          | 7,13,              |
|  17   |Inchaço na regiao do figado                  |                    |
|  18   |Inchaço na regiao do pancreas                |                    |
|  19   |Febre superior a 37                          | 13,                |
|  20   |Febre inferior a 38                          |                    |
|  21   |Dificuldade de visao                         |                    |
|  22   |Conjuntivite                                 |                    |
|  23   |Panaricio                                    |                    |
|  24   |Tem piolhos na cabeça                        | 9,                 |
|  25   |Pontos brancos na faringe                    | 11,                |
|  26   |Rouquidao                                    | 11,                |
|  27   |Dificuldade em engolir                       | 11,                |
--------------------------------------------------------------------------------
```

Figura 83. Ecrã com lista de sintomas existentes na base em certo instante; são também apresentadas as avarias que utilizam esse sintoma.

É ainda possível obter uma descrição das acções já utilizadas para corrigir avarias detectadas anteriormente (figura 84); nesta apresentação, as acções que são obtidas por relações lógicas de outras são apresentadas de tal forma que mostrem o processo que aconselham na resolução da avaria, sem ser necessário conhecer a descrição de cada uma das avarias.

Acção	Descricao da acção
1	(Administrar acido acetilsalicilico e Repouso)
2	Administrar acido acetilsalicilico
3	Repouso
4	((Repouso e (Administrar um antibiótico de elevado espectro e Administrar um analgesico))
5	((Administrar um antibiótico de elevado espectro e Administrar um analgesico)
6	Administrar um antibiótico de elevado espectro
7	Administrar um analgesico
8	Internamento na maternidade
9	Internamento nos cuidados intensivos
10	((Fazer dieta e Deixar de beber)
11	Fazer dieta
12	Deixar de beber
13	((Lavar as maos com um desinfectante e Usar um desinfectante na higiene diaria)
14	Lavar as maos com um desinfectante
15	Usar um desinfectante na higiene diaria
16	(((Lavar as maos com um desinfectante e Usar um desinfectante na higiene diaria) e Administrar um fungicida)
17	Administrar um fungicida
18	((Repouso e (((Fazer dieta e Deixar de beber) e ((Administrar um antiacido e Ser acompanhado por especialista em medicina interna)))
19	(((Fazer dieta e Deixar de beber) e ((Administrar um antiacido e Ser acompanhado por um especialista em medicina interna))
20	((Administrar um antiacido e Ser acompanhado por um especialista em medicina interna)
21	Administrar um antiacido
22	Ser acompanhado por um especialista em medicina interna
23	(((Repouso e ((Administrar um antibiótico de elevado espectro e Administrar um analgesico)) ou Fazer dieta)
24	(((Repouso xou Administrar um antibiótico de elevado espectro) e ((Administrar um analgesico ou ((Fazer dieta ou Administrar um antipiretico)))
25	((Fazer dieta e Administrar um antipiretico)
26	Administrar um antipiretico
27	Consultar um dentista
28	((Administrar um analgesico e Consultar um dentista)
29	((Consultar um dentista e ((Administrar um antibiótico de elevado espectro e Administrar um analgesico))
30	Aplicar um desinfectante
31	((Administrar acido acetilsalicilico ou exclusivo Administrar um analgesico)
32	((Repouso e (((Administrar um analgesico e Consultar um dentista) ou exclusivo ((Consultar um dentista e ((Administrar um antibiótico de elevado espectro e Administrar um analgesico))))
33	(((Administrar um analgesico e Consultar um dentista) ou exclusivo ((Consultar um dentista e ((Administrar um antibiótico de elevado espectro e Administrar um analgesico)))
34	((Repouso e Administrar um antibiótico de elevado espectro)
35	(((Administrar acido acetilsalicilico e Repouso) ou ((Administrar um antibiótico de elevado espectro e Administrar um analgesico))

Figura 84. Ecrã com as acções disponíveis para corrigir algumas avarias, e que já foram utilizadas para o efeito.

Utilizando os resultados impressos como apresentado na figura 83, pode-se verificar quais os sintomas que constituem cada "avaria".

Para uma visualização mais rápida é possível obter essa informação, utilizando um quadro próprio apresentado pelo programa, em que se referem as avarias

existentes e os sintomas que as constituem; para além destes dados é apresentado o número de vezes que ocorreram as avarias bem como as acções já utilizadas já utilizadas para a resolução do problema.

Esta informação é complementada por informação referente ao número de vezes que foi utilizada uma determinada acção (figura 85)

Avaria			Accoes		
Num.	Vezes	Sintomas	Num.	Vezes	Associacao de accoes
1	5	[1,2]	1	1	(2e3)
			23	1	((3e(6e7))ou11)
			4	1	(3e(6e7))
			7	1	7
			3	1	3
2	6	[1,2,3,4]	4	1	(3e(6e7))
			7	1	7
			5	4	(6e7)
3	4	[6]	8	1	8
			9	1	9
			10	1	(11e12)
			21	1	21
4	2	[7,8]	13	1	(14e15)
			16	1	((14e15)e17)
5	1	[1,6,8]	18	1	(3e((11e12)e(21e22)))
6	1	[5,8]	19	1	((11e12)e(21e22))
7	3	[2,9,10,11,15,16]	4	1	(3e(6e7))
			23	1	((3e(6e7))ou11)
			24	1	((3xou6)e(7ou(11ou26)))
8	20	[3]	5	2	(6e7)
			29	1	(27e(6e7))
			2	3	2
			7	2	7
			27	4	27
			28	5	(7e27)
			32	1	(3e((7e27)xou(27e(6e7))))
			1	2	(2e3)
9	1	[24]	30	1	30
10	4	[9]	7	1	7
			2	2	2
			31	1	(2xou7)
11	1	[25,26,27]	34	1	(3e6)
12	1	[7,13]	17	1	17
13	1	[11,16,19]	35	1	((2e3)ou(6e7))

Figura 85. Ecrã com lista de avarias, respectivos sintomas e acções já seleccionadas para as resolver; de realçar que as acções, ao invés de se apresentam sob a forma descritiva (como na figura 84), são mostradas resumidamente.

As avarias consideradas no exemplo são definidas pelos seguintes sintomas:

2 —> [1 2 3 4]

7 —> [2 9 10 11 15 16]

11 —> [25 26 27]

13 —> [11 16 19]

Em algumas destas avarias existem sintomas comuns a outras avarias, dando a entender a existência de uma certa relação entre essas avarias. Pode-se mesmo afirmar que existe uma certa correlação entre as diferentes avarias, correlação essa que nos propomos quantificar, tomando por base os sintomas comuns.

Comparando as descrições reais dos sintomas (figura 83) somos levados a concluir que, mesmo utilizando uma comparação meramente qualitativa, se verifica a existência de uma certa 'semelhança' entre as avarias.

		Correlacao entre as avarias		
Avaria	Avaria	com todos os sintomas.	com alguns dos sintomas	
			Interseccao.	Reuniao.
2	1	0.0 %	22.22 %	24.32 %
2	2	100.0 %	44.44 %	24.32 %
2	5	0.0 %	11.11 %	16.22 %
2	7	0.0 %	11.11 %	10.81 %
2	8	0.0 %	11.11 %	24.32 %
7	1	0.0 %	9.09 %	19.57 %
7	2	0.0 %	9.09 %	15.22 %
7	7	100.0 %	54.55 %	22.83 %
7	10	0.0 %	9.09 %	22.83 %
7	13	0.0 %	18.18 %	19.57 %
11	11	100.0 %	100.0 %	100.0 %
13	7	0.0 %	40.0 %	30.0 %
13	13	100.0 %	60.0 %	70.0 %

Figura 86. Resultados do cálculo das probabilidades de ocorrência de avarias.

O programa de diagnóstico apresenta relatórios (figura 86 e seguintes), baseados na experiência tida na resolução de cada uma das avarias, em que se sugerem soluções para um determinado problema; para além de apresentar as soluções que já foram utilizadas com êxito, é ainda apresentado um conjunto de sugestões com base nas acções ***escolhidas para resolver outras avarias***.

Para o cálculo das probabilidades apresentadas nas 3 colunas da figura 86, que apresenta o valor de uma correlação entre avarias, foi usada uma metodologia que tem por base o número de sintomas comuns às diversas avarias; a título de exemplo e para a avaria 13, teremos:

coluna 1 - Nenhuma das avarias tem no seu conjunto de sintomas **todos** os sintomas da avaria 13 (com excepção de ela própria); quando existir uma avaria que possua todos os sintomas (e eventualmente mais alguns) da avaria seleccionada, será efectuado um cálculo idêntico ao que será descrito para o método da coluna 3;

coluna 2 - A *intersecção* entre o conjunto de sintomas da avaria 13 com as outras, e que *não seja conjunto vazio* são:

13 com 13 - [**11** **16** **19**]

13 com 7 - [**11** **16**]

ou seja, entre a avaria 13 e **todas** as restantes avarias existe um nº total de 5 intersecções; há assim uma correlação entre as avarias, baseando-nos nos sintomas comuns, seguindo um método de intersecção, de $\frac{3}{5}$ <=> 60% para a primeira relação e $\frac{2}{5}$ <=> 40% para a segunda. Estes cálculos podem ser resumidos na seguinte expressão,

$$p_i = \frac{\#\{C_i\}}{\sum_{k=1}^{N} \#\{C_k\}} * 100\%$$

em que $\#\{C_n\}$ representa o valor correspondente ao cardinal do conjunto intersecção de sintomas e p_i representa a correlação sob a forma de percentagem. Como seria de esperar, esta percentagem é máxima quando se calcula o valor de correlação entre a avaria e ela própria; no quadro, apresentado na figura 86, os valores apresentados somam 100% para cada avaria, *podendo no entanto ser normalizado esse valor por forma a obtermos o valor 100% para o caso da correlação entre a avaria e ela própria*;

coluna 3 - O método desta coluna baseia-se na reunião dos sintomas das avarias que *têm pelo menos um sintoma comum* , que para o exemplo considerado será:

13 com 13 - [**11** **16** **19**]

13 com 7 - [**2** **9** **10** **11** **15** **16** **19**]

correspondendo a valores de #{conjuntos},iguais respectivamente a 3 e 7. Os valores calculados nesta coluna são $\frac{3}{3} * \frac{1}{\frac{3}{3}+\frac{3}{7}}$ <=> 70% e $\frac{3}{7} * \frac{1}{\frac{3}{3}+\frac{3}{7}}$ <=> 30%, que se pode resumir na seguinte expressão,

$$p_i = \frac{\#\{C_x\}}{\#\{Cr_i\}} * \frac{1}{\sum_{k=1}^{N} \left(\frac{\#\{C_x\}}{\#\{Cr_k\}} \right)} * 100\%,$$

em que $\#\{C_n\}$ representa o valor correspondente ao cardinal de cada conjunto; o conjunto C_x corresponde aos sintomas da avaria seleccionada (avaria 13 neste exemplo), e o conjunto Cr_k o conjunto reunião de sintomas (avaria 13 com 13 e 7 no presente exemplo); p_i representa a correlação sob a forma de percentagem. Tal como para a coluna precedente o cálculo efectuado fornece valores máximos para o caso da

correlação da avaria com ela própria, podendo também optar-se pela normalização deste valor para 1 ou 100%, o que neste se traduzia na utilização da expressão,

$$p_i = \frac{\#\{C_x\}}{\#\{Cr_i\}} * 100\%$$

Este tipo de análise é particularmente importante para ajudar ao cálculo da acções que podem corrigir uma avaria, objectivo final deste módulo de diagnóstico; esta ajuda é materializada por intermédio dos resultados apresentados no ecrã (figura 87) em que, para cada avaria escolhida, são sucessivamente apresentados os valores probabilísticos do sucesso da utilização de uma determinada acção correctiva, utilizando diversas metodologias de cálculo.

Consideremos, a título de exemplo, que estas avarias ocorreram no seguinte número:

2 —> 6 vezes,
7 —> 3 vezes,
11 —> 1 vez,
13 —> 1 vez.

Vamos supor que para estas avarias foram adoptadas as seguintes acções (notar que para cada avaria são indicadas as acções já usadas e em que número):

avaria	acção	Nº de utilizações da acção
2	4	1
	7	1
	5	5
7	4	1
	23	1
	24	1
11	34	1
13	35	1

Por análise do quadro apresentado na figura 85, pode-se verificar que as respectivas acções são definidas como:

acção	associação de acções
4	3 e 6 e 7
7	7
5	6 e 7
23	(3 e 6 e 7) ou 11
24	(3 xou 6) e (7 ou 11 ou 26)
34	3 e 6
35	(2 e 3) ou (6 e 7)

Podemos assim, baseados em toda esta informação, apresentar um quadro em que se relacionam as probabilidades das acções com as correlações já calculadas como se apresenta na figura 87.

Avaria numero: 2
definida pelos sintomas: [1,2,3,4]

ACCOES JA USADAS EM AVARIAS	PROBABILIDADES			
	sem correlacao entre avarias.	com correlacao entre avarias com		
		todos os sintomas.	pelo menos um sintoma	
			Interseccao de sintomas.	Reuniao dos sintomas.
1	0 %	0.0 %	5.56 %	7.3 %
4	16.67 %	16.67 %	15.56 %	12.52 %
7	16.67 %	16.67 %	12.96 %	11.35 %
5	66.67 %	66.67 %	30.74 %	18.65 %
18	0 %	0.0 %	11.11 %	16.22 %
23	0 %	0.0 %	8.15 %	8.47 %
24	0 %	0.0 %	3.7 %	3.6 %
29	0 %	0.0 %	0.56 %	1.22 %
2	0 %	0.0 %	1.67 %	3.65 %
27	0 %	0.0 %	2.22 %	4.86 %
28	0 %	0.0 %	2.78 %	6.08 %
3	0 %	0.0 %	4.44 %	4.8 %
32	0 %	0.0 %	0.56 %	1.22 %

Avaria numero: 11
definida pelos sintomas: [25,26,27]

ACCOES JA USADAS EM AVARIAS	PROBABILIDADES			
	sem correlacao entre avarias.	com correlacao entre avarias com		
		todos os sintomas.	pelo menos um sintoma	
			Interseccao de sintomas.	Reuniao dos sintomas.
34	100.0 %	100.0 %	100.0 %	100.0 %

Avaria numero: 7
definida pelos sintomas: [2,9,10,11,15,16]

ACCOES JA USADAS EM AVARIAS	PROBABILIDADES			
	sem correlacao entre avarias.	com correlacao entre avarias com		
		todos os sintomas.	pelo menos um sintoma	
			Interseccao de sintomas.	Reuniao dos sintomas.
1	0 %	0.0 %	1.82 %	3.91 %
4	33.33 %	33.33 %	21.52 %	14.06 %
7	0 %	0.0 %	5.61 %	12.16 %
5	0 %	0.0 %	6.06 %	10.14 %
23	33.33 %	33.33 %	20.0 %	11.52 %
24	33.33 %	33.33 %	18.18 %	7.61 %
2	0 %	0.0 %	4.55 %	11.41 %
3	0 %	0.0 %	1.82 %	3.91 %
31	0 %	0.0 %	2.27 %	5.71 %
35	0 %	0.0 %	18.18 %	19.57 %

Avaria numero: 13
definida pelos sintomas: [11,16,19]

ACCOES JA USADAS EM AVARIAS	PROBABILIDADES			
	sem correlacao entre avarias.	com correlacao entre avarias com		
		todos os sintomas.	pelo menos um sintoma	
			Interseccao de sintomas.	Reuniao dos sintomas.
4	0 %	0.0 %	13.33 %	10.0 %
23	0 %	0.0 %	13.33 %	10.0 %
24	0 %	0.0 %	13.33 %	10.0 %
35	100.0 %	100.0 %	60.0 %	70.0 %

Figura 87. Apresentação das correlações entre as acções que provavelmente corrigirão uma avaria, conforme apresentado pelo programa, no ecrã, em impressora ou num ficheiro.

A determinação dos valores existentes em cada coluna baseia-se quer na experiência conseguida com a utilização de diversas acções para resolver a presente avaria, quer nas acções utilizadas para resolver outras avarias, com alguma correlação com a seleccionada.

Na **coluna 1** do referido quadro ("sem correlação entre avarias") são apresentadas as probabilidades associadas à frequência com que foram utilizadas as acções para a solução **desta** avaria (sem utilizar nenhuma informação das restantes).

Nas **colunas 2** a **4** são utilizados os valores já calculados para a correlação entre avarias (indicados na figura 86); no entanto, os valores a apresentar terão em conta valores probabilísticos associados às acções utilizadas na solução de cada uma destas avarias; este factor depende directamente do número de vezes em que foram utilizadas cada uma das acções.

Consideremos mais uma vez a título de exemplo a avaria 13, em que foi utilizada uma vez a acção 35; sendo a avaria 7 a única que apresenta valores de correlação diferentes de zero, só necessitamos de considerar as acções utilizadas para resolver **directamente** estas duas avarias. Tendo sido utilizadas as acções 4, 23 e 24 em igual número para a solução da avaria 7, podemos dizer que cada uma destas acções tem uma probabilidade de sucesso de $\frac{1}{3}$, para essa avaria. Dado que a correlação entre a avaria 13 e 7, definida pela probabilidade p_i, é de 0%, 40% ou 30% (figura 86) conforme o método de cálculo utilizado em cada coluna, podemos dizer que a probabilidade de sucesso de cada uma das acções que resolvem a avaria 7 é de, $p_i * \frac{1}{3}$ (0% para a coluna 2, 13.33% para a coluna 3 e 10% para a coluna 4). Este processo aplica-se de forma similar para as outras avarias.

Falta por último referir alguns dos aspectos relacionados com a implementação deste módulo de diagnóstico de avarias.

A linguagem utilizada foi o PROLOG, pelas razões já apontadas noutra parte deste texto, inicialmente numa versão interpretada e posteriormente compilada. Dada a necessidade de recorrer a outros módulos, especialmente o de comunicação, foi implementada a ligação ao "assembler"; esta ligação permite ao utilizador estabelecer comunicação com qualquer estação espalhada na rede com vista a obter a informação que considerar necessária.

Além desta ligação foram implementadas, em "assembler", algumas primitivas, não previstas no PROLOG, que podem ser utilizadas como as já definidas no interpretador (ou compilador); destacamos de entre estas a que permite testar se alguma tecla foi pressionada e, em caso afirmativo conhecer a tecla accionada. Esta primitiva apresenta a vantagem de não obrigar a uma paragem na execução do programa.

É possível construir primitivas similares a estas para o acesso ao exterior, seja por 'portas' série ou paralelo, seja o acesso a 'placas' adicionais.

Além desta primitiva foram ainda implementadas outras que permitem o acesso directo à memória bem como efectuar algumas operações sobre essa memória.

```
?-findall(X, sintoma(X,_,ramo),L).          <<<< Cria lista dos grupos
     .....                                  <<<< de sintomas.
     X=_005D                                <<<< Variável não instanciada
     L=[28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 0, 40]
     .....                                  <<<< Lista de sintomas.
?-sintoma(X,_,ramo).                        <<<< Apresenta sintoma a sintoma
     GRUPO DE SINTOMAS: FEBRE               <<<< desde que seja do tipo ramo.
     X=28 ->;
     GRUPO DE SINTOMAS: DORES
     X= 29 ->;
     .....
?-findall(X,avaria(X,Y),L).                 <<<< Cria lista das avarias.
     X=_005D                                <<<< Variável não instanciada
     L=[1,2,3,4,5,6,7,8,9,10,11,12] ->;     <<<< Lista com avarias existentes
no                                                na base de dados.
?-
```

Figura 88. Apresentação de alguns exemplos da utilização dos recursos do PROLOG na pesquisa de informação contida na base.

Tal como apresentado no início da descrição deste módulo, existe uma base de dados (ou de conhecimentos) que é trabalhada pelo 'programa' , quer para apresentar sugestões sobre como corrigir as avarias, quer para a sua actualização.

Uma linguagem como o PROLOG não utiliza uma distinção tão precisa entre o que é uma base de dados e o programa propriamente dito; ao invés, a base de dados é constituída por cláusulas também 'executáveis'. Assim, o conjunto das cláusulas que contém a informação das avarias e das acções correspondentes (figuras 79 a 81) podem ser 'executadas'.

A implementação deste programa, seguindo também uma metodologia "top-down", aliás bastante propícia dado a estrutura da linguagem, recorreu à utilização das facilidades concedidas pelo PROLOG, nomeadamente no que diz respeito ao "back-tracking" , com vista à pesquisa de quais as cláusulas (eventualmente as constantes da base de dados) que unificam. Apresentam-se alguns exemplos destas possibilidades, executados recorrendo ao uso do interpretador PROLOG, (figura 88); neste exemplo foram utilizadas cláusulas da base de dados.

Naturalmente que, para a apresentação dos resultados como exemplificados na figura 88, não é necessário utilizar nem o interpretador nem conhecer as cláusulas PROLOG; a apresentação de resultados é efectuada através dos quadros representados nas figuras anteriores. Na figura 89 são apresentadas algumas das cláusulas implementadas neste módulo, que permitem mostrar a utilização das possibilidades da linguagem.

```
     ..............................
igual([],[]).                          <<<< Compara 2 listas de caracteres.
igual([H|T1],[H|T2]) :- igual(T1,T2).
igual([H1|T1],[H2|T2]) :- ((H1 is H2 + 32);(H1 is H2-32)),igual(T1,T2).
     ..............................
faz_maiuscula([],[]).     <<<< Passa a maiúscula os caracteres da lista.
faz_maiuscula([X|X1],[Y|Y1]):-
               ifthenelse(X > 96,Y is X /\ 95,Y is X), faz_maiuscula(X1,Y1).
     ..............................
membro(X,[X|_]).          <<<< Verifica se 1 elemento pertence a uma lista;
membro(X,[_|Y]) :- membro(X,Y). Ou é 1º elemento ou está da lista restante.

lista_contem_lista(X,[]).   <<<< Verifica se uma lista contém outra.
lista_contem_lista(X,[H|T]) :- membro(H,X),lista_contem_lista(X,T).
     ..............................
retirar([],A,[]).           <<<< Retira um elemento de uma lista.
retirar([A|B],A,B).
retirar([H|T],A,B) :- retirar(T,A,C),B = [H|C].
     ..............................
acrescenta([],A,[A]).       <<<< Acrescenta elemento a uma lista ordenada.
acrescenta([H|T],A,B) :- ifthenelse(A = H, B = [H|T], (ifthenelse(A < H,
                     B = [A|[H|T]], (acrescenta(T,A,C),B=[H|C])))).
     ..............................
```

Figura 89. Algumas das cláusulas utilizadas neste módulo; neste exemplo mostra-se o uso das possibilidades de "backtracking" e da recursividade.

Na figura 90, apresentam-se outros passos utilizados na implementação, mas com uma importância diferente da dos apresentados na anterior figura.

É de destacar em especial a implementação do módulo utilizando as possibilidades do PROLOG no que diz respeito a uma utilização de uma estrutura modular para o desenvolvimento do programa e por outro lado a utilização de diversos módulos; estes poderão ser implementados em diversas linguagens, incluindo o PROLOG e o "assembler".

Na figura 90 estes aspectos são particularmente explícitos nas primeiras linhas, onde podemos verificar alguns aspectos:

- o módulo principal ***main*** ocupa um pequeno número de linhas, sendo chamados os módulos a partir de opções do menu;

- as chamadas a outros módulo PROLOG são indicadas no início (***extrn***);
- de igual modo para módulos escritos em "assembler" (***extra .....:asm***);

```
:- segment(joao).       <<<< Indicação de utilização de um programa extenso.
:- public main/0.       <<<< Acesso do exterior para este predicado.
:- extrn modcom/0:asm(modcomprolog).      <<<< Ligação ao módulo de comunica.
:- extrn cursor_off/0:asm(cursor_off_p).  <<<< Módulo em "assembler"
:- extrn ler/1:asm(ler).                  <<<< Módulo em "assembler"
:- extrn corre/0:far.
:- extrn muda_base/0:far.                 <<<< Outros predicados PROLOG.
:- extrn acertar/0:far.
     ......................
:- public cria_base_ou_nao/1.
:- public poe_termo/2.                    <<<< Outros predicados PROLOG,
:- public mai_min/2.                      <<<< acessíveis de outros módulos.
:- public limpa_base_antiga/0.
     ......................
main :- corre,                            <<<< Entrada no módulo.
  repeat,
    [!call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)), menu_00!],
  fail.
     ......................
menu_00 :- cursor_off,call(intervalo(Tempo)),call(base(Base)), <<< Menu principal
  call(roda_pe_esta(Onouoff)),ifthenelse(Onouoff = on,true,faz_ecra),
  tmove(24,45),tmove(24,60),imprime_data_hora,verifica_intervalo,!,
  ler(Char),              <<<< Conforme caracter seleccionado assim será
  case([                  <<<< executada uma parte do módulo.
    Char = 315 -> (!,help,poe_termo(roda_pe_esta,off)),
    Char = 323 -> (!,modcom,poe_termo(roda_pe_esta,off)),
    Char = 321 -> (!,backup),
    Char = 317 -> (!,ifthen(not(altera_intervalo),cls)),
    Char = 318 -> (!,ifthen(not(relatorio),cls)),
    Char = 316 -> (!,ifthen(not(muda_base),cls)),       <<<< Teclas de função.
      ......................
    Char = 319 -> (!,ifthen(not(detectar),cls)),
    Char = 322 -> (!,gestao),
    Char = 27 -> (!,cls,write('Confirma SIM/NÃO/enter=SIM/esc=NÃO ? '),
                input($SIM$,$NAO$,[$SIM$,$NAO$],A,3),
                ifthenelse(A=$SIM$, (cls,halt), (poe_termo(roda_pe_esta,off),cls))),
    Char = 324 -> (!,cls,halt),
    igual([Char],"?") -> (!,help,poe_termo(roda_pe_esta,off)),
    igual([Char],"c") -> (!,modcom,poe_termo(roda_pe_esta,off)),
    igual([Char],"b") -> (!,backup),
      ......................
    igual([Char],"g") -> (!,gestao),
    igual([Char],"f") -> (!,poe_termo(garbage,full)),   <<<< Outras teclas.
    igual([Char],"h") -> (!,poe_termo(garbage,half)),
    igual([Char],"z") -> (!,poe_termo(garbage,zero)),
    igual([Char],"t") -> (!,cls,halt)|true ]).
```

Figura 90. Exemplo da implementação do módulo em PROLOG. Apresentam-se as especificações das ligações a programas noutras linguagens bem como a estrutura do menu principal.

Apresenta-se por fim um exemplo das cláusulas utilizadas no cálculo dos diversos tipos de correlação entre avarias e acções; estas são apresentadas em duas

figuras, 91 e 92, em que se apresentam na primeira destas a cláusula que implementa a resposta à selecção da opção de apresentação de relatórios.

```
relatorio :- cls,poe_termo(roda_pe_esta,off),
  Separador = $----------------------------------------------------------------$,
  write('Na impressora/ecrã/file [IMPR/ECRA/FILE enter=IMPR/esc=ECRA] ? '),
  input($IMPR$,$ECRA$,[$IMPR$,$ECRA$,$FILE$],A1,4),
  ifthen(A1=$ECRA$,A=$ECRA$),
  ifthen(A1=$FILE$,(nl,write('Nome do Ficheiro:'),read_string(12,X),mai_min(X,File),
                tell(File), A=File)),
  nl,
  ifthen(A1=$IMPR$,(A=A1,tell(lpt1))),
  write(Separador),nl,write(Separador),nl,
  cabecalho_relatorio(A),nl,
  write(Separador),nl,write(Separador),nl,
  totais_na_base(A),nl,
  write(Separador),nl,write(Separador),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  not(totais_sintomas(A)),nl,
  write(Separador),nl,write(Separador),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  not(totais_accoes(A)),nl,
  write(Separador),nl,write(Separador),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  not(avarias_accoes(A)),nl,
  write(Separador),nl,write(Separador),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  not(avarias_avarias(A)),nl,
  write(Separador),nl,write(Separador),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  not(accoes_avarias(A)),nl,
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),
  write(Separador),nl,write(Separador),nl,
  ifthen(not(A=$ECRA$),(put(12),told)),
  poe_termo(roda_pe_esta,off),
  tmove(24,60),write('PRIMA UMA TECLA'),get0_noecho(_),cls.
      ......................
```

Figura 91. Apresentação de algumas cláusulas usadas na selecção da opção de apresentação de relatórios estatísticos.

Na figura 92 apresentam-se algumas das cláusulas utilizadas no cálculo das correlações entre acções que resolvem avarias com alguma semelhança.

Embora apresentado de uma forma simplificada, o algoritmo da figura 92 necessita de recorrer a cálculos de uma certa complexidade; para o conseguir foi necessário utilizar cláusulas intermédias a residir na base de dados. De notar que, para efeito de simplificar o desenvolvimento, parte das cláusulas foram implementadas inicialmente numa forma conceptual, isto é, previu-se a sua chamada e sucesso antes de proceder à sua implementação real; esta razão levou à existência de um número elevado, permitindo no entanto uma mais eficiente sistematização.

```
      ......................
accoes_avarias(Imp) :-
  ifthen(not(Imp=$ECRA$),tell(con)),nl,
  write('Quer ver as acções de avarias [SIM/NAO/enter=SIM/esc=NAO] ? '),
      ......................
  qual_a_lista($avaria$,L1),!,
  ifthen(not(Imp=$ECRA$),ifthenelse(Imp=$IMPR$,tell(lpt1),tell(Imp))),
  ifthen(not(L1=[]),membro(Avaria,L1)),
  call(avaria(Avaria,L)),ax(Avaria,L,Imp).

ax(Avaria,L,Imp) :-
  [!ifthenelse(call(numero_de_vezes(avaria(Avaria),Nn)),N is Nn,N is 0),
  nl,
  write('+-----------------------------------------------------------------+'),nl,
      ifthen(((Imp=$ECRA$);(Imp=$IMPR$)),write('|   Avaria   numero:
            |')),
  ifthen((not(Imp=$ECRA$),not(Imp=$IMPR$)),write('| Avaria numero:')),
  posiciona_coluna(17,Imp,16),write(Avaria),nl,
  write('| definida pelos sintomas:                                      |'),
  posiciona_coluna(27,Imp,17),write(L),nl,
  write('+-----------+-----------------------------------------------------+'),nl,
  write('|           |           P R O B A B I L I D A D E S               |'),nl,
  write('| ACCOES JA +-----------+-----------------------------------------+'),nl,
  write('|           |    sem    |  com  correlacao  entre  avarias  com   |'),nl,
  write('| USADAS EM | correlacao +-----------+-----------------------------+'),nl,
  write('|           |   entre   |           |  pelo  menos  um  sintoma   |'),nl,
  write('| AVARIAS   |  avarias. | todos os  +---------------+-------------+'),nl,
  write('|           |           | sintomas. | Interseccao   | Reuniao dos |'),nl,
  write('|           |           |           | de sintomas.  | sintomas.   |'),nl,
  write('+-----------+-----------+-----------+---------------+-------------+'),nl,
  limpa_correlacoes,determina_correlacoes(Avaria,L),call(garbage(Gctipo)),
  ifthenelse(Gctipo=zero,true,gc(Gctipo)),not(mostrar(Avaria,L,N,Imp)),
  call(garbage(Gctipo)),ifthenelse(Gctipo=zero,true,gc(Gctipo)),retract(total(_)),
  retract(total_cor(_)),retract(parte_hip1(_)),retract(parte_hip1_cor(_)),
  retract(parte_hip2(_)),retract(parte_hip2_cor(_)),retract(ja_viu(_))!],fail.
      ......................
determina_correlacoes(Avaria,Lista) :- ifthen(det_cor_parte_hip1(Lista),true),
  ifthen(det_cor_parte_hip2(Lista),true),ifthen(det_cor_total(Lista),true).

det_cor_parte_hip1(Lista) :-
  [!poe_termo(parte_hip1_cor,0)!],call(avaria(A,L)),
  [!membro(M,L),membro(M,Lista),merge_lista(Lista,L,L1),length(L1,N1)!],
  length(L,N2),length(Lista,N3),N11 is N2+N3-N1,V is N11/N3,call(parte_hip1_cor(X)),
  X1 is V+X,[!poe_termo(parte_hip1_cor,X1)!],call(numero_avarias(A)).

det_cor_parte_hip2(Lista) :-
  [!poe_termo(parte_hip2_cor,0)!],call(avaria(A,L)),[!membro(M,L),membro(M,Lista),
  merge_lista(Lista,L,L1),elem_comuns(Lista,L1,L4),length(L4,N1)!],
  length(L1,N2),V is N1/N2,call(parte_hip2_cor(X)),X1 is V+X,
  [!poe_termo(parte_hip2_cor,X1)!],call(numero_avarias(A)).

det_cor_total(Lista) :-
  poe_termo(total_cor,0),
  call(avaria(A,L)),lista_contem_lista(L,Lista),length(Lista,N1),
  length(L,N2),V is N1/N2,call(total_cor(X)),X1 is V+X,
  [!poe_termo(total_cor,X1)!],
  call(numero_avarias(A)).
```

Figura 92. Apresentação de algumas cláusulas usadas na apresentação do relatório sobre correlação entre acções.

PARTE 4.

RESULTADOS, APLICAÇÕES DIVERSAS, RECOMENDAÇÕES PARA TRABALHO FUTURO E CONCLUSÕES.

Nos capítulos 11 a 12, constituindo a parte final do presente texto, serão referidos resultados obtidos na utilização do regularímetro digital bem como as aplicações de partes do presente projecto em trabalhos diferentes.

Por fim discutem-se as recomendações e evolução previsível deste sistema; segue-se um ponto conclusões.

Capítulo 11. Resultados do Regularímetro e outras aplicações (outros projectos).

Mais do que apresentar os resultados típicos da aplicação dos algoritmos utilizados em todo o sistema pretende-se, no presente capítulo, apresentar uma análise de algumas características relevantes, nomeadamente o tempo de processamento, medido em diversas fases desses programas.

11.1 Resultados obtidos no projecto de controlo de regularidade de fios.

Segue-se uma análise dos tempos e recursos despendidos para alguns cálculos nas estações remotas de aquisição de dados e na estação central. Para alguns processos, nomeadamente a comunicação entre cada estação remota de aquisição e microcomputador, são apresentados os resultados globais.

A1 - Determinação do CV% e U% nas estações remotas.

No cálculo do CV% e do U%, respectivamente o coeficiente de variação e desvio médio absoluto, *também chamado coeficiente de irregularidade*, nas estações remotas, despendem-se os tempo apresentados no quadro 8.

$2048 \geq N \geq 256$	t< 0.6 segundos (*dificilmente mensurável)*
N = 4096	t = 0.6 segundos
N = 8192	t = 0.9 segundos
N = 16384	t = 1.4 segundos
N = 32768	t = 2.5 segundos

em que N representa o número de amostras utilizadas no cálculo.

Quadro 8. Tempo despendidos para o cálculo do CV% e U% nas estações remotas.

Como já anteriormente foi referido, o cálculo é efectuado internamente com 32 bits, sendo os resultados apresentados sob a forma de uma percentagem.

É possível efectuar a compactação dos dados adquiridos através de alguns dos comandos implementados, permitindo efectuar o cálculo do coeficiente de variação correspondente a segmentos de maior comprimento.

Os valores apresentados no quadro 8 permitem-nos afirmar que, não é este algoritmo de determinação do CV% que impõe restrições impeditivas da utilização do sistema como controlador em tempo real. Para o efeito vejamos que:

- 0.6 segundos para N=256, é o pior resultado dado a relação $\frac{N}{t}$ ser a mais baixa para todos os valores de N analisados;

- é provável que uma grande parte do tempo de processamento seja despendido no cálculo da raiz quadrada e na divisão, os algoritmos mais complexos;

- o tempo despendido por amostra é praticamente constante para todos os valores de N;

- a implementação deste algoritmo na mesma fase em que são adquiridas as amostras, não sobrecarrega o processador dados ser apenas necessário efectuar, para cada amostra adquirida, pouco processamento; efectivamente, estes valores são conseguidos apenas com o recurso a uma multiplicação e 4 somas (acumular os somatórios de x^2 e os somatórios de x); posteriormente seriam efectuados os cálculos complementares para a determinação do CV% usando a fórmula:

$$CV\% = \frac{\sqrt{\frac{\sum_{i=0}^{N-1} x_i^2 - N\overline{x}}{N-1}}}{\overline{x}} * 100\%$$

em que x_i corresponde a cada amostra recolhida e $\overline{x}$ é o valor da média das N amostras recolhidas; de notar que os valores a acumular durante a aquisição das amostras permitem reduzir o tempo total de cálculo, na medida em que esse tempo é igualmente distribuído; já o cálculo de U% não pode ser tão simplificado, razão pela qual não é utilizada esta metodologia para esse parâmetro têxtil.

Poderemos então considerar que o tempo necessário para este cálculo será provavelmente inferior, em cada byte adquirido, a $\frac{0.6}{256}$ = 2.4 ms e estando limitado superiormente num valor próximo de $\frac{2.5}{32768}$ = 75 µs. Este valor é compatível com o tempo de processamento despendido na aquisição. No entanto, pode limitar-nos consideravelmente se existirem outros processos em execução; neste caso é provavelmente mais eficaz utilizar uma execução na estação central, garantindo um peso do processamento em cada estação remota suficientemente baixo.

A1 - Determinação do CV%, U% e DR% no microcomputador central.

Apresentam-se de seguida alguns tempos medidos nos algoritmos construídos no microcomputador central. Os valores apresentados nos quadros 9 e 10 foram obtidos num microcomputador UNYSIS 800 (equipado com microprocessador 80386 a 16

MHz), máquina que podemos considerar como tendo velocidades aceitáveis quando comparado com os XT e AT iniciais.

A título de exemplo refira-se que, os mesmos programas executados num microcomputador transportável TOPIS, é da ordem de 3.6 vezes, sendo este último, naturalmente, o mais lento.

Determinação do DR% e IDR%:

N	Passo	tempo (seg.)
256	10.0	3.0
	5.0	9.0
	2.5	5.1
2048	10.0	16.4
	5.0	30.0
	2.5	56.8
8192	10.0	62.7
16384	10.0	124.0
	2.5	439.2

em que N representa o número de amostras consideradas e o passo é o incremento mínimo entre cada valor percentual a considerar .

Quadro 9. Tempos para a determinação do DRtotal%, DRl%, IDRtotal% e IDRl%.

Determinação do coeficiente de variação CVl%:

N	tempo(seg)
256	1.3
2048	5.9
8192	23.5
16384	47.7

Quadro 10. Tempos para a determinação do CVl%; de notar que estes tempos não se referem ao cálculo apenas do coeficiente de variação tradicional (CV%) mas sim do CVl%, ou seja o valor do coeficiente de variação ao longo de diversos segmentos.

Os valores encontrados não são intoleráveis para a sua utilização por um operador. A aceleração dos microcomputadores seria sempre possível com o recurso a co-processadores aritméticos, ou placas específicas, em caso de necessidade. Realce-se no entanto que uma grande parte destes tempos assumem um valor menos crítico se,

tomando em linha de conta a estrutura em rede do sistema, considerarmos que enquanto é feito um determinado processamento é efectuada a aquisição de informação; esta aquisição, sendo a cargo de outro processador, acarreta um menor consumo de tempo, permitindo ao microcomputador despendê-lo noutras tarefas.

Resultado obtido com equipamento industrial:

SAMPLE NUMBER	U %	AVE.	THIN -50%	THICK C 50%	NEPS 200%	(1.50m DR %	5%) CV(L)%
1	11.08	101.1	0	6	36	19.98	3.51
2	11.08	102.2	0	10	36	18.01	3.45
3	10.98	101.5	0	9	36	20.27	3.50
4	11.22	102.0	1	9	55	21.19	3.71
5	10.99	100.7	0	8	53	17.99	3.39
6	11.01	101.0	0	6	46	17.82	3.37
7	10.72	102.2	0	10	50	15.66	3.23
8	10.77	102.3	1	5	49	21.88	3.67
9	10.62	102.8	0	8	40	15.49	3.17
10	10.65	103.4	0	8	35	18.64	3.50
Mean	10.91	101.9	0	7	43	18.69	3.45
R	0.60	2.7	1	5	20	6.39	0.54
s	0.21	0.9	0.4	1.7	7.8	2.15	0.17
Per1000m			0	15	87		
Overall		CV% 0.84				DR% 19.77	CV% 3.55
						(-) 8.48	(+)11.29

Mat.Speed 200m/min Eval.Time 2'30" Range +/- 100% Normal
Mat. Soz Pe Sob eo Count Slot Date 22/5/92 Sig.

Resultado obtido com o equipamento desenvolvido:

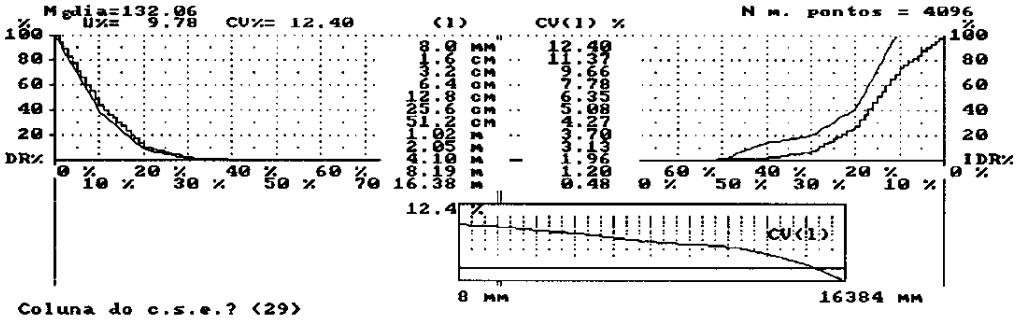


Figura 93. Resultados obtidos com equipamento industrial para determinação de CV% e U% comparado com o obtido com o presente equipamento.

Apresentam-se na figura 93 os resultados obtidos para CV% com dispositivos vulgarmente utilizados nos processos industriais e os resultados apresentados por este sistema. Como é perceptível, na maior parte do equipamento em uso não são apresentados resultados com tanta informação; apesar de os relatórios industriais

apresentados serem relativos a equipamento destinado a uso laboratorial, verifica-se que nem sempre são apresentados todos os coeficientes (DR%, IDR%, etc.) e quando são raramente atingem a mesma precisão.

Convém realçar que o presente sistema permite uma apresentação de resultado sob a forma numérica além do aspecto gráfico mostrado na figura.

B1 - Determinação da Transformada de Fourier (FFT) nas estações remotas.

Apresentam-se de seguida os valores despendidos com o cálculo do espectro simples de Fourier, utilizando diversas janelas e um número de pontos isto é, um número de amostras adquiridas, variável. Os valores estão expressos em segundos.

Número de pontos	Rectangular		Hanning		Hamming	
	*N-1	*1	*N-1	*1	*N-1	*1
8192	30.6	29.0	33	31.4	32.8	31.8
4096	15.0	14.3	15.6	14.9	15.7	15.0
2048	7.0	6.8	7.4	7.2	7.5	7.1
1024	3.4	3.4	3.5	3.5	3.6	3.4
512	1.7	1.7	1.8	1.7	1.8	1.7
256	0.8	0.8	0.9	1.0	0.9	1.0

sendo N o número de amostras a utilizar no cálculo.

Quadro 11. Tempos despendidos no cálculo da transformada rápida de Fourier nas estações remotas.

Os tempos apresentados no quadro 11 dizem respeito ao cálculo da transformada rápida de Fourier, dividindo ou não os resultados por N. Isto deve-se ao facto de o mesmo algoritmo poder ser utilizado, não só para o cálculo do espectro de um sinal constituído por N amostras recolhidas, mas também para o do do cálculo da transformada inversa, ou seja, o sinal temporal possuindo o espectro anteriormente calculado. A diferença, a nível matemático, reduz-se à utilização ou não do divisor N para efectuar o referido cálculo. Esse divisor é utilizado no primeiro e não no segundo dos casos.

De acordo com os resultados apresentados no referido quadro, verifica-se que o tempo de processamento varia numa razão idêntica à do número de amostras utilizadas no cálculo, pelo que podemos considerar um valor típico e aproximadamente constante de tempo de processamento por amostra; este valor pode ser importante se tomarmos em conta o facto de o algoritmo em causa poder ser efectuado em simultâneo com a aquisição de dados (recorrendo eventualmente à

utilização de interrupções). Nesse caso, poderíamos aplicá-lo para efectuar a determinação de espectros em tempo real de sinais utilizando uma frequência de amostragem inferior a $\frac{1}{\frac{32.8}{8192}} \approx 250$ Hz para o caso da janela de Hanning. Esta frequência de amostragem permitir-nos-ia utilizar o algoritmo para a análise de sinais com componentes espectrais até ao limite máximo de ≈120 Hz. Com este valor podemos utilizar o algoritmo para o estudo de um fio têxtil a ser bobinado a uma velocidade até 57 m/minuto; embora não sendo um valor muito elevado, é aceitável para diversos equipamentos utilizados laboratorialmente.

B2 - Determinação da Transformada de Fourier e Walsh na estação central.

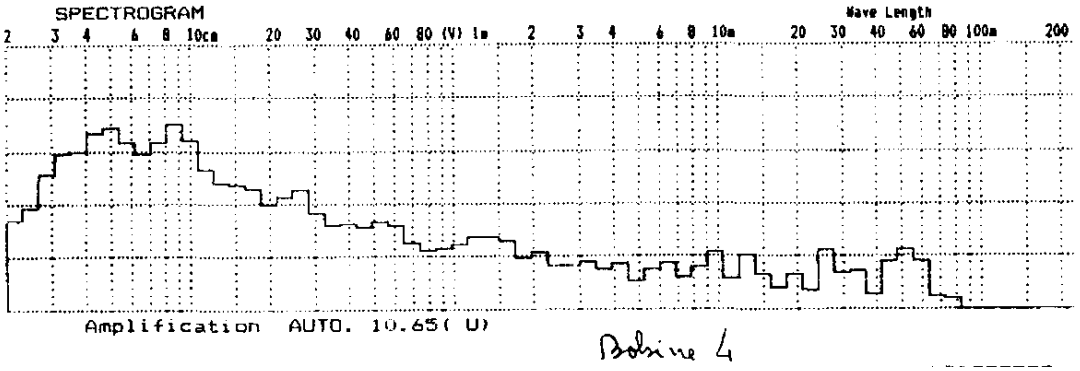
Apresentam-se de seguida (quadro 12) os valores de tempo medidos para os algoritmos de análise espectral, quer usando um algoritmo baseado na FFT quer usando um relativo a uma transformada rápida de Walsh.

N	tempo despendido (segundos) (t. Fourier)	(t. Walsh)	comprimento máximo de onda de fibra possível de analisar com sensor de 8 mm
256	2.1	1.3	4.1 m
512	3.7	2.1	8.2 m
1024	7.0	3.1	16.4 m
2048	14.5	6.3	32.8 m
4096	30.0	11.6	65.5 m
8192	63.5	23.7	131.0 m
16384	134.6	49.0	262.0 m

Quadro 12. Tempo despendido no cálculo da transformada rápida de Fourier no microcomputador central.

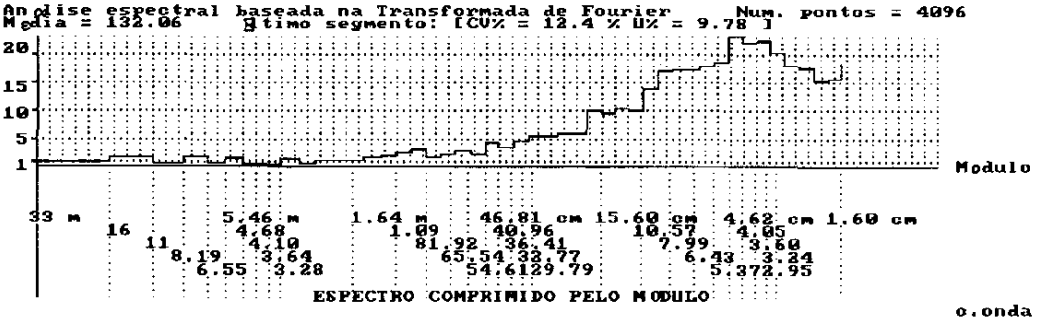
Os valores apresentados nestes quadros foram obtidos num microcomputador UNYSIS 800. Os valores medidos permitem concluir que o processamento embora bastante rápido é ainda superior ao encontrado no algoritmo implementado nas estações remotas (para a FFT); estes valores não podem no entanto ser comparados desta forma, dado que na estação remota a precisão é consideravelmente inferior à existente no microcomputador central. Para além deste factor não podemos deixar de tomar em conta as potencialidades da estação central no que diz respeito ao interface com o utilizador e capacidade de armazenamento. Apresenta-se na figura 94 resultados deste sistema e de equipamento industrial tradicional.

a) Resultado de equipamento industrial:



b) Resultado do equipamento desenvolvido:

I - Usando transformada de Fourier:



II - Usando transformada de Walsh:

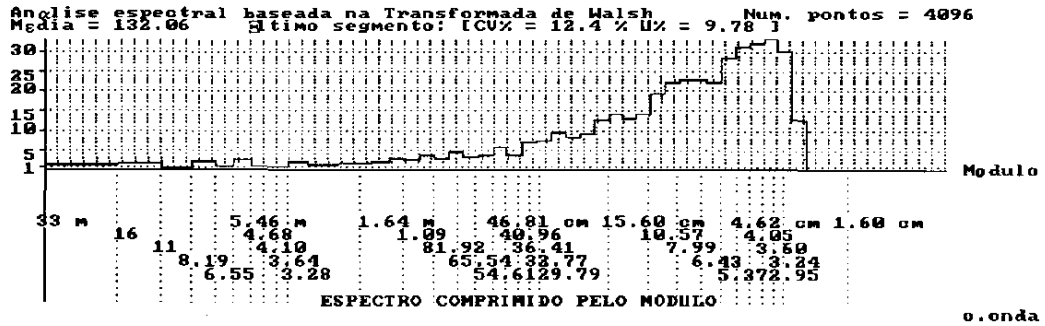


Figura 94. Resultados obtidos com equipamento industrial (*na figura 1 resultados de outro fabricante*) comparados com os obtidos com o presente equipamento, na a determinação do espectro do fio.

É de realçar, comparando os dois algoritmos implementados (baseado nas transformadas de Fourier e Walsh) a relação de aproximadamente 2.6 existente entre os tempos despendidos; esta relação leva a preferir utilizar a FWT para implementar o cálculo do espectro, com os inconvenientes já discutidos noutro capítulo.

Não arriscando dizer que a informação obtida com a transformada de Walsh é suficiente relativamente a uma análise espectral tradicional, dado não termos ainda resultados suficientes para garantir esta validação, convém salientar que os resultados práticos obtidos permitem retirar conclusões muito similares utilizando um método ou outro na análise da regularidade de fios têxteis; só com a realização de um número de testes consideravelmente maior será possível determinar se este tipo de processamento permite retirar conclusões aceitáveis. Embora seja nossa convicção que esta possibilidade é real, parece-nos que só com algum processamento posterior (nomeadamente a eliminação de algumas componentes espectrais correspondentes a pequenos comprimentos de onda) será fácil proceder de imediato a esta verificação.

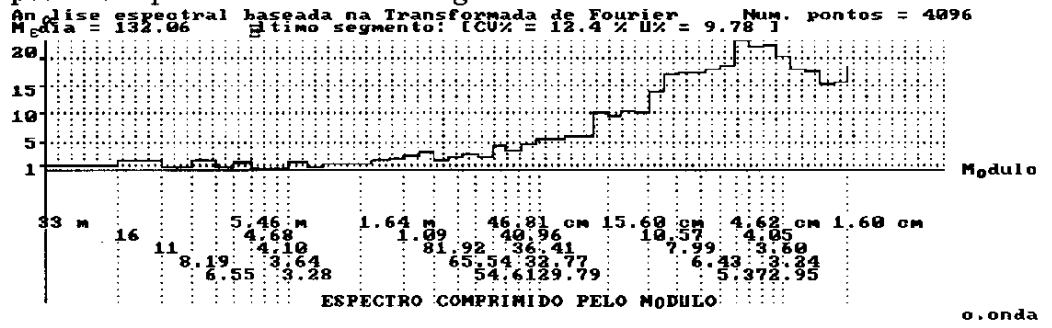
Comparando os resultados apresentados por este sistema com os obtidos pelos sistemas tradicionais verificamos existir uma concordância bastante grande dos parâmetros característicos do fio têxtil em questão; estes aspectos principais dizem respeito aos coeficientes CV%, U% e características do espectro.

Quer o U% quer o CV% têm valores muito semelhantes em cada um dos relatórios apresentados; relativamente ao espectro, esquecendo algumas diferenças na apresentação do gráfico, é possível verificar uma concordância elevada nas componentes de maior peso (zona dos 4 aos 10 cm), correspondentes às fibras constituintes do fio em estudo.

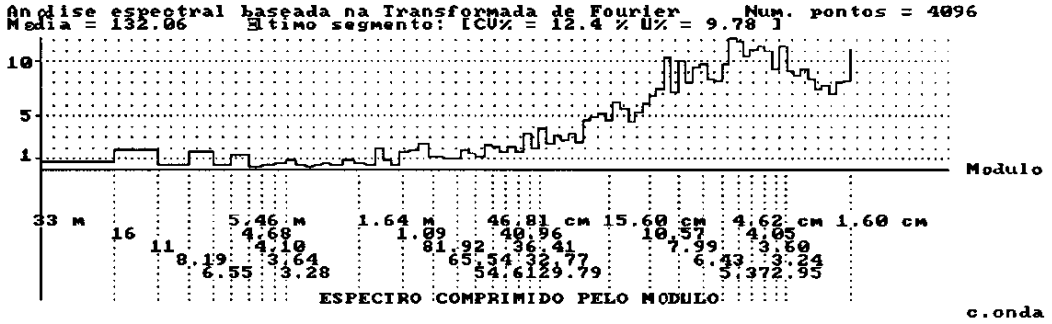
Em qualquer dos relatórios não é evidente a existência de qualquer perturbação fora do vulgar, embora se note que no relatório que propomos existem algumas variações imperceptíveis no outro. A apresentação de resultados sob esta forma não é condição indispensável, sendo possível a apresentação quer com aspectos diferentes, quer utilizando diversos segmentos de cálculo, como se exemplifica na figura 95.

Nestes diferentes tipos de apresentação dos resultados é patente, para além dos aspectos menos relevantes associados ao aspecto gráfico, a possibilidade de utilizar as facilidades de acumulação de resultados bem como a modificação da largura mínima de detecção de frequências; ao acumular resultados é possível obter um espectro médio, permitindo comparar um espectro num determinado instante com esse valor médio, detectando perturbações de uma forma eficaz.

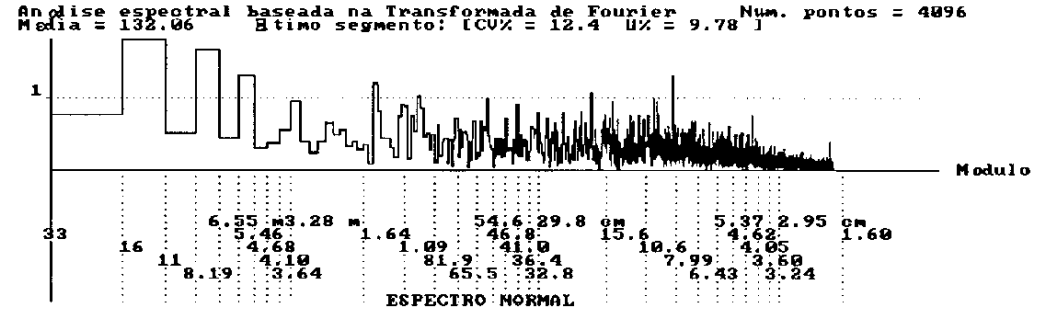
Espectro comprimido com bandas largas:



Espectro comprimido com bandas mais estreitas:



Espectro não comprimido ('papel' semi-logarítmico):



Acumulação de segmentos (vê-se o seg. acumulado sobreposto a um não acumulado) :

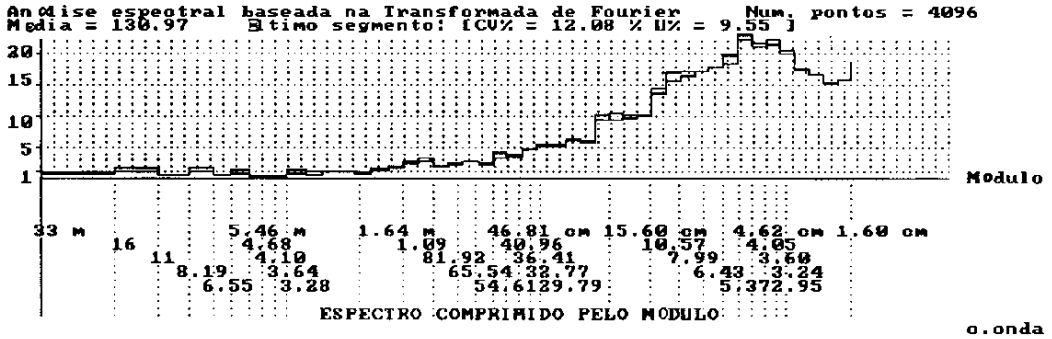


Figura 95. Diferentes tipos de apresentação de resultados da análise espectral.

Em contrapartida, ao modificar-se a largura de cada banda de frequências, é possível detectar alguma componente de peso elevado que tenha surgido e possa estar mascarada por estar numa banda de frequência demasiado larga.

C - Resultados obtidos na comunicação.

Na última implementação do módulo de comunicação, utilizando a estrutura que implementa um protocolo de ligação orientado ao caracter, foi efectuada uma análise dos tempos despendidos para transferir conjuntos de bytes através da linha de comunicação; essa análise pretendia quantificar a taxa de transferência real de informação através de uma linha série assíncrona (Quadro 13).

	Receber 12 Kbytes			**Receber 12 Kbytes Enviar 52 Kbytes**			**Enviar 52 Kbytes**		
Baud rate	**Taxa**	**Nº erros**	**Nº ensaios**	**Taxa**	**Nº erros**	**Nº ensaios**	**Taxa**	**Nº erros**	**Nº ensaios**
115200	5744	0	200	6238	0	200	6471	0	200
57600	3379	0	200	3529	0	200	3599	0	200
38400	2447	0	200	2444	0	200	2470	0	200
9600				634	0	200			
2400	162	0	100	144	0	100			
total		0	700		0	900		0	600

Total de bytes transaccionados: 700 * (12 K) + 900 * (64 K) + 600 * (52 K) = 95 Mbytes
(A coluna referente à **taxa** (*de comunicação*)é expressa em bytes/segundo).

Quadro 13. Valores obtidos na transacção de comunicação através da linha série utilizando "frames" com campo de informação de 64 bytes.

Note-se que em todas esta análise não foi detectado qualquer erro irrecuperável, a nível das transacções efectivadas. Isto não quer contudo dizer que não tenham ocorrido retransmissões de pacotes ("frames") para garantir o cumprimento do protocolo de transferência dos dados, que no entanto não foram (*nem seriam facilmente*) quantificados.

Por outro lado os valores obtidos para a a taxa de transferência de informação, sendo os valores médios para um grande número de comunicações, permitem concluir que a utilização de um protocolo de comunicação eficiente como o presentemente em teste não prejudica grandemente a eficácia da comunicação. Efectivamente com um

certo "baud rate" seria possível transmitir e/ou receber informação, não recorrendo a um protocolo, a uma taxa v definida por,

$$v = \frac{baud}{(Num\_bits + Num\_start + Num\_stop + Num\_paridade)}.$$

Este valor é tradicionalmente igual a, (para um baud rate de 115200), $v = \frac{115200}{11}$ =10472 bytes/seg (Num_bits corresponde ao nº de bits que constituem cada caracter, Num_start = 1 é o bit de paridade, Num_stop é o nº de bits de fim e Num_paridade é o bit de paridade); se atendermos a que existe sempre um certo intervalo entre cada caracter, esta velocidade será sempre menor.

O protocolo implementado apresenta uma taxa de transferência real na ordem dos 60% do valor máximo possível numa ligação assincrona **não utilizando qualquer protocolo** de controlo da ligação, valor perfeitamente adequado se atentarmos além deste aspecto ao facto de estarmos a comparar um valor medido com um valor máximo calculado; efectivamente não está incluído neste cálculo (*ao contrário do teste efectuado*) o tempo despendido na interpretação da informação recebida com vista a ser enviada resposta e, por outro lado, considerou-se que os caracteres são enviados sem nenhuma separação, o que é difícil e profundamente desaconselhável se não definirmos um protocolo claro; acresce ainda o facto de sempre ser despendido tempo nas trocas de "frames" de reconhecimento de boa recepção por parte do destinatário.

Foram igualmente efectuados testes com um número de bytes de informação efectiva que são incluídos em cada "frame" de informação diferente dos especificados inicialmente no protocolo; estes testes (utilizando um campo de informação de 128 bytes ao invés dos tradicionais 64) são apresentados no quadro 14.

Como se pode verificar os resultados não mostram uma eficiência muito superior, apesar de em cada pacote transmitido pela linha de comunicação existir, percentualmente, uma menor quantidade de informação de controlo necessária; este facto advém de os bytes de controlo que constituem cada "frame" não aumentarem em número por maior que seja o campo de informação do mesmo. Apesar disso, é de notar que sempre que exista uma repetição de um "frame" por detecção de qualquer erro, o custo associado à transmissão será maior, dado ter de se repetir um "frame" de maior comprimento.

	Receber 12 Kbytes			Receber 12 Kbytes Enviar 52 Kbytes			Enviar 52 Kbytes		
Baud rate	Taxa	Nº erros	Nº ensaios	Taxa	Nº erros	Nº ensaios	Taxa	Nº erros	Nº ensaios
115200	6615	0	100	6375	0	100	6456	0	200
57600	4024	0	100	3627	0	200			
total		0	200		0	300		0	200

Total de bytes transaccionados: 200 * (12 K) + 300 * (64 K) + 200 * (52 K) = 32 Mbytes
(A coluna referente à **taxa** (*de comunicação*)é expressa em bytes/segundo).

Quadro 14. Valores obtidos na transacção de comunicação através da linha série utilizando unidades de comunicação com campo de informação de 128 bytes.

Os factos anteriormente apontados mostram que a taxa de transferência pode no entanto aumentar se se verificar uma modificação acentuada do nível de ocorrência de erros por unidade de tempo. Esta taxa de erros, pode obrigar a uma retransmissão de determinado número de pacotes de dados, com uma determinada frequência. Existe assim uma relação entre o número de pacotes a retransmitir e o número de bytes de informação efectiva que circulam em cada pacote de dados.

11.2 Aplicações diversas.

Como já mencionado, este projecto tem por base um conjunto de conceitos mais vastos que os directamente associados à regularimetria têxtil, pelo que parte deste trabalho pode ser utilizado em diferentes projectos.

Efectivamente surgiram, e prevê-se que continuem a surgir, outras utilizações para este sistema, quer utilizando apenas o "hardware" desenvolvido para cada estação local (funcionamento isolado ou 'stand alone'), quer utilizando 'hardware' e módulos do 'software' já implementados. De entre as solicitações já apresentadas destacam-se:

- Sistema de controlo e gestão da produção na indústria do calçado.

Neste trabalho o sistema será constituído por uma ou mais estações remotas ligadas a um microcomputador central de baixo custo; cada estação local deve supervisionar o estado de uma linha de produção de calçado com algumas dezenas de

operários. Essa supervisão diz respeito a tempos de paragem, falhas de produção e necessidades de matéria prima. Inicialmente o sistema apreende, ligado a variáveis discretas (contacto aberto/fechado de relés de 220 v), o funcionamento da linha de produção, que é gerida por um chefe de linha. Simultaneamente aconselha-o, com base na informação já armazenada e na que está a adquirir, sobre as acções que deve tomar.

A unidade central apresentará também relatórios estatísticos do funcionamento de todos os intervenientes do processo de fabrico.

Numa fase posterior, após todos os ensaios passar-se-á à fase de automatizar completamente o controlo da linha de produção, correspondente às tarefas do encarregado da linha.

Realça-se que um único microcomputador central pode efectuar a gestão de diversas linhas de produção, minimizando assim os custos derivados desta automatização.

O interface entre o operador, o referido encarregado, e o microcomputador pode ser desenvolvido de uma forma suficientemente eficaz que permita a eventualidade deste encarregado gerir mais de um microcomputador.

Destaque-se ainda o grande número de tarefas complementares possíveis de efectuar por esta unidade, das quais destacamos:

- Contabilidade dos tempos mortos.
- Contabilidade dos níveis de matérias-primas requisitadas.
- Gestão do tempo real de produção.

Este esquema obriga a uma pequena placa de interface entre relés e indicadores luminosos de 220 V~ e as entradas digitais existentes.

- Um controlador lógico programável, utilizando linguagem GRAFCET.

Este trabalho consistiu na utilização do "hardware" de uma estação remota, para a implementação de um controlador lógico programável (mais conhecido por **PLC**), trabalho desenvolvido por elementos do Departamento de Electrónica Industrial. A escolha da placa da estação para esse efeito residiu essencialmente nas potencialidades a nível de entradas saídas digitais, e a possibilidade de inserir programas executáveis directamente na memória, após compilação. Esse módulo de compilação foi desenvolvido directamente na estação.

- Implementação de um sistema de medição do tamanho de partículas em suspensão (granulómetro).

Este projecto consiste na medição do tamanho de partículas em suspensão numa solução.

Essa solução, já com as partículas em suspensão, passa num reservatório que é atravessado por um feixe de **raio laser**.

A difracção do raio laser, provocada por estas partículas, é detectada numa 'parede' revestida de sensores ópticos; conforme a distância ao centro deste novo raio assim se poderá estabelecer uma relação com o tamanho das referidas partículas

Para este trabalho optou-se pela utilização da estação dado o grande número de canais analógicos a analisar em simultâneo (um canal para cada detectar óptico), apesar de ser perfeitamente aceitável uma resolução de poucos bits, dado ser praticamente suficiente detectar se um determinada sensor foi ou não atingido pelo raio laser.

Esta estação remota (melhor seria chamá-la de local dada a proximidade de todos os componentes do sistema) comunica através de uma linha série com um microcomputador que possua linha série, usando o 'standard' RS-232 C.

- Controlo de produção de uma confecção têxtil.

Este projecto tem muitos aspectos concordantes com os de gestão e controlo de uma linha de produção de calçados. Também é necessário monitorizar o tipo e o tempo de execução de diversas tarefas a desempenhar por operários e máquinas da linha de produção de uma unidade fabril da indústria têxtil de confecção.

Aqui, por cada posto de trabalho, existe uma estação local que, para além das entradas saídas digitais e/ou analógicas, possui um pequeno teclado e um mostrador LCD como interface com o operário. Está também incluído nessa pequena consola um leitor de cartões ou código de barras para identificação das peças em produção.

Com base nestes dados são fornecidos resultados estatísticos e sugestões para a melhoria da "performance" dos intervenientes no processo de fabrico.

- Automatização e aquisição de informação de dados de autocarros de transporte público.

Consiste na aquisição de informação sobre todo o estado de um autocarro, desde que inicia a sua marcha até recolher.

Essa informação diz respeito a diversos itens do funcionamento interno do autocarro, do número de passageiros e das acções que o condutor tomou em cada caso concreto.

Ao efectuar a recolha, ou durante o abastecimento de combustível é efectuado o 'desabastecimento' da informação recolhida. Para o efeito liga-se uma saída de dois condutores a uma 'tomada' que, usando o protocolo RS-485, enviar a informação que a estação a bordo recolheu para um micro central, onde se efectuarão os cálculos necessários.

Convém realçar que para todos estes projectos existem bases comuns que, sendo bem exploradas permitem a redução em grande escala do trabalho e tempo de projecto. Nessa perspectiva, definiram-se tarefas bem delimitadas para os diversos intervenientes, partindo-se de uma cuidadosa especificação.

Os responsáveis por cada projecto, terão as tarefas de gestão e coordenação dessas tarefas, e discutirão entre si as temporizações e distribuição das mesmas. Com esta solução evitou-se o desgaste e perda de tempo associado ao desenvolvimento de trabalhos iguais. Esta consideração tornou-se particularmente evidente no que respeita à adaptação dos módulos de comunicação.

Capítulo 12. Recomendações para trabalho futuro e Conclusões.

12.1 Recomendações para trabalho futuro.

A arquitectura base do sistema desenvolvido, uma unidade central com boa capacidade de processamento e múltiplas estações locais e/ou remotas de controlo, será de manter como premissa base na construção de um sistema similar embora mais evoluído. Esta consideração advém, da especificidade do projecto do regularímetro analógico-digital ou, em termos gerais da necessidade de efectuar controlo e gestão de dispositivos e/ou máquinas dispostas a uma distância e em número consideráveis em cada unidade fabril, independentemente do produto a fabricar nesse local.

Tendo em conta a tendência actual do mercado de electrónica e informática, que se tem feito sentir com um considerável impacto especialmente no respeitante à capacidade de processamento as comunicações, é previsível que a evolução deste e outros sistemas similares aponte, no essencial nestes dois sentidos. É assim natural que o incremento da 'inteligência' disponível em cada estação local, a par da decréscimo dos custos associados à comunicação (tempo de processamento e tempo despendido na comunicação) sejam os principais campos de investimento do trabalho futuro nas estações remotas.

O outro aspecto a melhorar é o relativo à programação base a residir quer nas estações remotas quer na central. Esta modificação advirá não só da adaptação de "software" já existente aos novos processadores mas também da implementação de um maior número de algoritmos de controlo, associado a um aumento da eficiência dos mesmos. Este último aspecto tem em conta a utilização de todo o sistema como estrutura básica dos elementos necessários à implementação de módulos de "CIM" (Produção Assistida por Computador).

Este tipo de modificação não implica que parte do trabalho a desenvolver, nomeadamente as estações remotas de aquisição de dados, não possam ser utilizadas em trabalhos de 'nível' inferior, designadamente em funcionamento independente ("stand-alone").

Ainda no campo da programação residente em cada estação, está em projecto a implementação de diversas ferramentas, que permitirão uma utilização genérica. De entre essas destacam-se operações matemáticas em vírgula flutuante, algoritmos de controlo tipo PID e PI, algoritmos de processamento digital de sinal do nível de complexidade, precisão e volume de dados dos já construídos no microcomputador central.

Serão igualmente implementados algoritmos mais complexos para efectivar as comunicações entre estações e a central, nas diversas camadas definidas pela ISO.

Concluindo, podemos afirmar ser previsível a curto e médio prazo a modificação do sistema construído (e algumas das fases desse trabalho estão já a decorrer), quer com uma profunda modificação do "hardware" quer por adaptações e aperfeiçoamentos da programação existente e a instalar. Descrevem-se em seguida alguns dos campos em que estamos a desenvolver trabalho, bem como os objectivos e ideias associados a esses projectos.

12.1.1 Evolução do "hardware" das estações remotas.

Duas restrições, ou melhor, dois limites impostos pela tecnologia disponível quando do desenvolvimento das estações remotas de aquisição de dados e controlo, referiam-se às limitações de memória e tempo de processamento "versus" representação numérica.

A primeira impunha uma limitação de 64 Kbytes de memória, extensível a 128 Kbytes se usássemos a memória de dados separada da de código.

Naturalmente esta limitação poderia ser ultrapassada (o que aconteceu para a memória de dados), com recurso a bancos de memória. No entanto, esta solução é trabalhosa e pouco eficiente para a memória de código.

Por outro lado, toda a informação a transmitir e/ou receber da linha de comunicação, teria de residir em memória de dados, acessível **apenas** pelo microcontrolador da estação, residindo numa zona demarcada **apenas** por "software". De realçar ainda que o único responsável pela transmissão e recepção de informação para e da rede, seria o mesmo microcontrolador, sendo garantido que **para um byte recebido ou enviado existiriam sempre dois acessos à memória** por parte do mesmo processador, situação capaz de agravar a eficiência, pelo menos no que diz respeito ao cálculo do tempo gasto em processamento.

A segunda restrição é associada ao tempo de processamento; uma multiplicação ou uma divisão de números representados sob a forma de 8 bits levaria o mínimo de 4 μs (com cristal de 12 MHz). Esta última imposição poderia levar a tempos de processamento suficientemente elevados dada a necessidade de efectuar comunicações em simultâneo (utilizou-se como exemplo a multiplicação/divisão por, sendo das instruções mais demoradas serem, ainda assim, importantes para o Processamento

Digital de Sinal). Embora estas duas funções possam funcionar concorrentemente e a última exista apenas quando há informação relevante, é possível a existência de um certo tráfego de manutenção na rede quer entre as estações quer entre estas e a unidade central.

Para obviar estes problemas optou-se pela reformulação das características da placa das estações remotas, bem como da estrutura de apoio existente para efectivar essa implementação, sempre com o objectivo de apresentar soluções **à medida** para as necessidades específicas dos projectos que nos aparecem (ver figura 96).

Utilização de novos processadores:

Recorrer-se-á à utilização de diversos processadores (mais de um) em cada estação local, conseguindo um incremento significativo da capacidade de cálculo. Por outro lado, dada a escolha efectuada para a comunicação, nada nos impede que o processador central da estação seja diferente de uma para outra.

O primeiro passo nesse sentido será a utilização de microcontroladores de 16/32 bits - família intel MCS-96 - que já estão a ser utilizados por investigadores do Departamento de Electrónica Industrial no âmbito de outros projectos.

O processador que pensamos utilizar é o 80C196 ou similar, uma evolução do 8096, possuindo uma excelente eficiência, que se pode destacar por:

- Implementarem a multiplicação de 16 * 16 bits em 2.3 µs para um relógio de 12 MHz (Divisão de 32/16 bits em 4.0 µs, adição de 16 bits em 0.66 µs); de realçar estar prevista uma versão de 16 MHz.
- Possuem um conversor analógico-digital que permite a utilização de 8 canais de conversão com 10 bits, funcionando pelo método de aproximações sucessivas com "Sample & Hold", e um tempo de conversão mínimo 26 µs com cristal de 12 MHz.
- Possuem 5 portas de interface de 8 bit cada. 2 Destas são utilizadas para gerar o barramento de dados e de endereços.
- Permitem implementar o acesso directo à memória (DMA), com vista a permutar informação com grande rapidez.
- Tem um poderoso conjunto de instruções que permitem operações orientadas ao bit, byte, duplo-byte e quatro-bytes, etc.
- Além destas excelentes características possuem ainda outras características muito positivas como seja:

  Saídas de PWM (Modulação por Largura de Impulso), Porta série, Saídas e entradas séries de alta velocidade, etc.

Utilização de um co-processador para gerir as comunicações:

O primeiro passo que prevemos dar no sentido de melhorar a comunicação prende-se com os pontos anteriores na medida em que passa pela utilização de um co-processador de comunicações associado ao microcontrolador da estação local. Este co-processador garante toda a interligação com a rede (isto é a unidade de controlo) e controla a ligação com o processador principal da estação por intermédio de D.M.A. (acesso directo à memória).

Controla ainda a conexão de cada estação a uma linha assíncrona, permitindo a existência de uma consola assíncrona portátil para ligar a cada estação. Esta linha assíncrona pode ser nomeadamente uma **rede usando um protocolo orientado ao caracter** tal como especificada no sistema actual, possibilitando-se assim a construção de uma árvore de redes. Esta solução só depende do "software", uma vez que o referido co-processador tem implementado internamente uma UART para a comunicação assíncrona e um controlador de SDLC.

O processador em causa será o 83C152 **[2, 3, 4]**, um microcontrolador especialmente dedicado à gestão de comunicações, da família MCS-51 da intel, com as seguintes características principais:

- É um microcontrolador de 8 bits **[2, 3, 4]**.
- Possui uma UART para efectuar e controlar a comunicação série assíncrona, incluindo o recurso ao protocolo do nono bit.
- Tem capacidade de efectuar acesso directo à memória (DMA).
- Possui 5 portas de oito bits para comunicação, em paralelo, com o exterior.
- Contém, internamente, um controlador de comunicação síncrona que permite a implementação de protocolo SDLC, e o protocolo de acesso ao meio de comunicação partilhado CSMA-CD. Este controlador garante o reconhecimento do endereço, geração de "flag", retransmissão automática, etc., conforme as especificações ISO para a interligação de sistemas abertos (OSI).
- Permite a implementação de diversos tipos de contadores e temporizadores ("timers").
- Possui uma linguagem e modo de funcionamento totalmente compatível com os microcontroladores da família MCS-51, utilizados na implementação das presentes estações remotas.

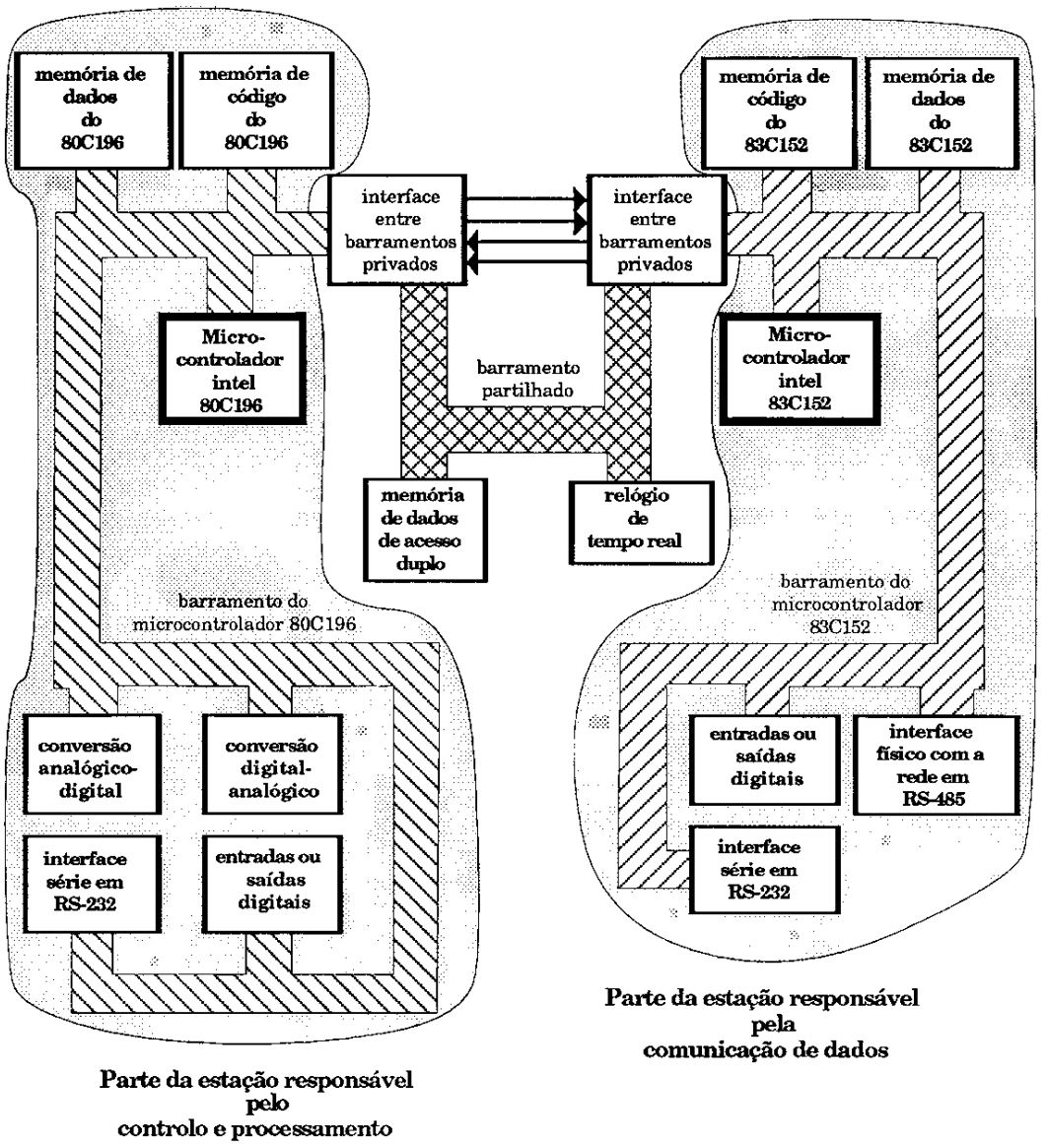


Figura 96. Arquitectura base das novas estações remotas de aquisição de dados e controlo, utilizando dois processadores, um para controlo outro para gestão da comunicação; de notar a existência de zonas de memória partilhada e outras de memória privada. A partilhada permite uma fácil utilização da informação a receber e/ou transmitir pelo módulo de comunicação, enquanto a privada será utilizada para o algoritmo de cada processador **[6]**.

Utilizar o conceito de bibliotecas de circuitos impressos:

Iniciou-se o desenvolvimento de uma biblioteca de circuitos impressos que permitirão desenvolver estações remotas de aquisição e processamento de informação à medida das necessidades de cada aplicação.

Esta orientação prende-se quer com a vulgarização de ferramentas de desenho, teste e simulação de circuitos impressos quer com a facilidade, rapidez e economia com que se obtêm protótipos e placas finais.

Esta opção permite minimizar muito os custos de projecto destes circuitos, além dos associados a cada estação, na medida em que se evita um grande número de conectores e cabos, que são além do mais a fonte de um sem número de problemas, e consequentemente geradores de elevados custos de reparação.

12.1.2 Evolução do "software" das estações remotas.

Sendo efectuada a modificação física das estações remotas de aquisição de dados, terá de ser alterada, em alguns casos com bastante profundidade, a sua programação. Como já foi exposto, prevê-se a curto prazo a implementação de novos módulos de "software", com vista a cumprir outros objectivos.

Modificação do módulo de rede:

Para um funcionamento com as condições definidas anteriormente torna-se premente utilizar um protocolo de controlo de ligação orientado ao bit tipo **HDLC** ou **SDLC**, o que se torna bastante simples com o recurso ao co-processador dedicado às comunicações.

Se com o novo co-processador se implementar um nó de interligação de redes, terá de ser implementado um novo protocolo para as diversas camadas já especificadas neste trabalho (quer a nível de ligação lógica quer a nível de transferência de mensagem). A modificação do protocolo de controlo de ligação (passando a ser usado o SDLC ou HDLC) não vai obrigar à modificação de todas as restantes camadas de programação; naturalmente que sendo o módulo de comunicações o que requer maior complexidade, será sempre o responsável pelo grosso das modificações.

Algumas alterações serão ainda indispensáveis com vista a garantir o funcionamento das estações com dois processadores. A comunicação entre estes dispositivos será efectuada através de uma memória de dados (RAM) com duas portas de acesso com vista a garantir o funcionamento de forma bastante rápida. Está

também prevista a possibilidade de recorrer a acesso directo à memória para efectivar essa transferência de informação, reservando-se o operador, com base no "software" implementado, o direito de opção entre as duas soluções.

Além destas, e como exposto prevê-se o desenvolvimento de outros módulos de programação, que serão descritos num dos pontos seguintes. Estes módulos implementarão algumas das tarefas para as quais têm existido solicitações bastante frequentes de utilizadores industriais.

12.2 Conclusões.

A título de conclusões e considerando os aspectos relacionados com os objectivos do trabalho proposto existem alguns conceitos que poderão, futuramente, ajudar na implementação de um sistema similar ao implementado, mas de uma forma mais eficiente, reduzindo consideravelmente o tempo de desenvolvimento. Estes aspectos, que procuramos actualmente implementar com os trabalhos em curso no ***departamento de Electrónica Industrial*** nem sempre foram correctamente utilizados, quer por deficiente organização quer pela **inexistência de uma equipa de trabalho** para todo o projecto. Dentro dos aspectos que teremos de corrigir, destacamos:

- É necessário utilizar uma especificação muito precisa de todos os objectivos do trabalho. Mesmo que seja necessário efectuar simulações que permitam ao utilizador final verificar o funcionamento geral do sistema, é lucrativo o tempo despendido nesta fase do trabalho.

- Nunca depender da realização de parte do trabalho para efectuar a especificação completa do sistema. Isto prende-se nomeadamente com a possibilidade de dispor de módulos de "hardware" adquiridos para efectuar todo o teste de programação necessário, procedendo posteriormente à construção do pretendido.

- Não colocar sobre a responsabilidade de apenas uma pessoa a execução de todo o trabalho. Este aspecto, claramente assumido pelos responsáveis e intervenientes nos projectos industriais, depara com grandes reticências e obstáculos no meio académico, nomeadamente na execução dos trabalhos de doutoramento. É nossa opinião que este aspecto deve ser revisto, pois é sempre possível conciliar os elevados níveis de investigação exigidos para o doutoramento com a finalidade prática de cada

projecto. Naturalmente estas considerações são defendidas para trabalhos no âmbito da engenharia, existindo outras áreas em que estes aspectos terão de ser repensados.

Apesar destas considerações, pensamos terem ficado demonstrados dois conceitos gerais. Assim consideramos como essencial que a investigação em centros de engenharia, naqueles em que os seus intervenientes são engenheiros ou formadores de engenheiros, tenha uma componente **prática** de elevado nível e sejam, dentro do possível, nos limites da realidade e necessidades da sociedade envolvente.

Outro dos aspectos que nos parecem de maior importância, reside na desmistificação da impossibilidade de serem efectuados projectos na área da electrónica e "hardware". Embora seja comummente aceite que a nossa competitividade é muito baixa nesta área, não podemos extrapolar esse conceito, válido relativamente aos artigos de grande divulgação e consumo, para as áreas em que, usando os conhecimentos gerais, se utilizam dispositivos menos divulgados. Mais especificamente podemos constatar que em indústrias de grande implantação no nosso país, podemos ser competitivos na automatização ou controlo do todo ou partes das linhas de produção. Esta competitividade advém de dois aspectos principais:

- Por um lado a electrónica e informática não estão tão divulgadas nessas indústrias, utilizando na construção dos sistemas de automatização ou controlo dispositivos de disseminação reduzida.

- Existe um elevado nível de conhecimento dos aspectos relevantes e determinantes do processo de fabrico, fruto de anos e anos de experiência por parte dos diversos operadores e industriais desses sectores (casos das indústrias têxteis, de calçado e corticeira).

- Em regra é necessário construir sistemas de **chave na mão** que têm por vezes variantes dentro da mesma unidade fabril, para tarefas que parecem similares. Sendo verdade que este aspecto pode não obrigar a uma grande alteração do sistema de controlo, implica geralmente um contacto directo entre projectista, construtor e utilizador, o que facilita o nosso trabalho.

- Confirmando este aspecto, refira-se a frequência com que alguns industriais portugueses, nomeadamente do sector têxtil, são consultados por fabricantes internacionais de maquinaria, sobre o funcionamento esperado.

Estas considerações, levam assim a pensar na existência de excelente caminho a desenvolver pelos investigadores e docentes portugueses. Embora um pouco fora do âmbito deste relatório, não é de mais reafirmar que este caminho só será percorrido com sucesso se:

- Passarmos a considerar, nas instituições universitárias, que a **Engenharia** é um tema de doutoramento de primordial importância, sendo mesmo perfeitamente defensável que, dado o desenvolvimento actual do país, esta seja uma área prioritária.
- Estes trabalhos tenham apoio, pelo menos nesta fase inicial, da parte das instituições com mais recursos.
- O último aspecto a considerar, porventura o de maior importância, é a aposta que os industriais têm de fazer nestas pessoas e instituições, sob pena de ficarem de tal forma dependentes do "know-how" estrangeiro, que empenhem o futuro das suas indústrias às decisões de outros.

Estas considerações baseiam-se na constatação de que um projecto de engenharia consiste, no seu essencial, na utilização dos conhecimentos e experiência adquirida ("know-how") para a solução de um certo problema; convém realçar que estes aspectos estão intimamente ligados aos da originalidade e inovação do projecto, mas na perspectiva **da implementação** e dos **resultados** obtidos, menos que nas técnicas utilizadas.

Existem ainda diversos aspectos adicionais a tomar em conta em projectos, como o presente, especialmente virados para a automatização de alguns aspectos de um processo de fabrico, a que devemos também reconhecer uma certa importância, como sejam:

- Lucro imediato versus médio/longo prazo, capaz de condenar ao fracasso um projecto.
- Tempo de desenvolvimento, questão que pode levar a que um projecto, perfeitamente realista no início se venha a tornar obsoleto na altura em que termine.
- etc,

Apesar de todas estas dificuldades parece-nos importante, para não dizer premente, a necessidade de associar todo o trabalho efectuado nos nossos dias nas mais diversas instituições universitárias às necessidades reais que surgem cada dia na indústria, aproveitando sempre que possível, as imposições de realizações práticas associadas à progressão na carreira académica para a definição de projectos conjuntos.

PARTE 5.

BIBLIOGRAFIA.

A bibliografia apresentada está dividida em quatro temas específicos para facilidade de consulta e com o objectivo de sistematizar esta apresentação.

Os temas principais dizem respeito às Comunicações, Processamento de Sinal, aplicações de 'Inteligência Artificial' e Regularimetria Têxtil, de acordo com o ênfase dado neste trabalho aos pontos associados a estas matérias.

Acrescentou-se um último grupo que engloba toda a bibliografia genérica utilizada, desde a Estatística até à Electrónica, passando por artigos e livros de Investigação Operacional.

Dentro de cada tema são apresentadas as referências efectuadas no texto, com uma numeração própria; numa subsecção de cada tema, apontam-se artigos e livros sem referência precisa no texto mas que pela sua importância em assuntos colaterais ou por serem aconselháveis como orientações para trabalho futuro merecem ser mencionados.

A - Bibliografia relacionada com Comunicações.

Livros e artigos referenciados:

[1] - ERIC CATIER
Les Reseaux Locaux: Les Grandes Batailles se Mènent sur le Terrain;
Électronique Industrielle, Nº160 1989/4.

[2] - BOB JOHNSON
Applic. Techniques for the 83C152 Global Serial Channel in CSMA/CD Mode;
intel - Application Note, AP-429, 1989/3.

[3] - INTEL
80C152JP Communications Controller, 8-bit Microcomputer with Extended I/O
intel data sheet 1988/2

[4] - INTEL
80C152 Universal Communications Controller
intel data sheet 1987/5

[5] - JON DMUSE; GEORGE R. HAYEK
Standard Protocols are Needed for Distributed Microcontrolers;
Data Communications 1986/1 p. 171.

[6] - BORWORN PAPASRATORN ; PRASIT PRAPINMONGKOLKARN
A Small-Scale Distributed Microp. System Using Shared Memory Technique;
IEEE transactions on Industrial Electronics, VOL. 32, N. 2 1985/5 , p. 97.

[7] - INTEL
8044 BITBUS Enhanced Microcontroller;
intel data sheet 1985/4

[8] - PRANAY CHAUDHURI ; KALARAB RAY ; SUKUMAR GHOSH
A Real-Time Process Scheduler for a Ring-Type Microcomputer Network;
IEEE transactions on Industrial Electronics, VOL. IE-32, N.1 1985/2 p.56.

[9] - MICHAEL FINE
Demand Assign. Multiple Access Schemes in Broadcast Bus Local Area Net.;
IEEE transactions on Computers VOL. C-33, N. 12, 1984/12 , p. 1130.

[10] - J. F. TIRADO; J. J. RUZ; M. MELLADO
The WIRMI System: A Multimicrocomputer-Based Sys. for Hotel Automation;
IEEE transactions on Industrial Electronics VOL IE-31 N.4 1984/4

[11] - NG. X. DANG ; M. DIAZ NAVA ; R. N. HUSOVIC
Small LAN with Collision-Free Techniques for Connecting Heterogeneous Microcomputer systems;
Microprocessing and Microprogramming 1983/12 p. 167.

[12] - A. MOELANDS
I2C in Consumer Applications;
Electronic Components and Applications, VOL.5 N.4 1983/9 , p. 142.

[13] - JAMES S. MEDITCH ; CHIN-TAU A. LEA
Stability and Optimization of the CSMA and CSMA/CD Channels;
IEEE transactions on Communications VOL. COM-31, N.6 1983/6.

[14] - ALAN J. WEISSBERGER
Bit Oriented Data Link Controls;
Computer Design, 1983/3.

[15] - LIANG LI ; HERMAN D. HUGHES ; LEWIS H. GREENBERG
Performance Analysis of a Shortest-Delay Protocol;
Computer Networks 6 1982/6 p. 189.

[16] - V. CARL HAMACHER ; GERALD S. SHEDLER
Access Response on a Collision-Free Local Bus Network;
Computer Networks, 6, 1982/6 , p. 93.

[17] - W. BUX ; K. KÜMMERLE ; H. L. TRUONG
Data Link-Control Perform.: Results Comparing HDLC Operational Modes;
Computer Networks, 6 1982/6 , p. 37.

[18] - DAVID E. CARLSON
Bit Oriented Data Link Control Procedures;
IEEE transactions on Communications vol. COM-28 N.4 1980/4.

[19] - H. V. BERTINE
Physical Level Protocols;
IEEE transactions on Communications vol. COM-28 n.4 1980/4.

[20] - J. W. CONARD
Character-Oriented Data Link Control Protocols;
IEEE transactions on Communications vol. COM-28 n.4 1980/4.

[21] - PAUL E. GREEN Jr.
An Introduction to Network Architectures and Protocols;
IEEE transactions on Communications, VOL. COM-28, N.4, 1980/4 p.413.

[22] - R. A. DONAN ; J. R. FERSEY
Synchronous Data Link Control: a Perspective;
Distributed Computing Systems 1st Int. Conf. 1979/10.

[23] - JOHN BEASTON
Using the 8273 SDLC/HDLC Protocol Controller;
intel - Application Note AP-36, 1978/3.

[24] - IBM SYSTEM COMMUNICATIONS DIVISION
IBM Synchronous Data Link Control - General Information;
Manual de utilização e apresentação.

[25] - IBM SYSTEMS DEVELOPMENT DIVISION
Binary Synchronous Communications - General Information;
Manual de utilização e apresentação.

[26] - INTEL
The BITBUS Interconnect Serial Control Bus Specification;
Intel.

Livros e artigos não explicitamente referenciados:

JOHN A. STANKOVIC
A Perspective on Distributed Computer Systems;
IEEE transactions on Computers VOL. C-33, M.12 1984/12.

ADRIAN SEGALL ; BARUCH AWERBUCH
A Reliable Broadcast Protocol;
IEEE transactions on Communications VOL. COM-31, N.7, 1983/7 , p. 896.

YARON I. GOLD ; W. R. FRANTA
An Efficient Collision-Free Protocol for Prioritized Acess-Control of Cable or Radio Channels;
Computer Networks 7 1983/7 p. 83.

ALEX GOLDBERGER
Data Communications;
Electronic Components and Applications, VOL.5 N.3 1983/6 , p. 142.

JEFF SELTZER ; NASEER SIDDIQUE
Interfacing Different Local Area Networks;
PHILIPS Technical Publication - 155 Electronic Components and Applications, VOL.6 N.3 1983/3 , p. 141.

KAPALI P. ESWARAN ; V. CARL HAMACHER ; GERALD S. SHEDLER
Collision-Free Access Control for Computer Communication Bus Networks;
IEEE transactions on Software Engineering, VOL. SE-7 N.6 1981/11 .

R. M. METCALFE ; D. R. BOGGS
Ethernet: Distributed Packet Switching for Local Computers Networks;
Tutorial Local Computer Network - IEEE communications of the ACM 1976/7.

IBM
Systems Network Architecture: Concepts and Products;
Manual de utilização e apresentação.

INTEL
Distributed Control Modules Databook;
Intel.

B - Bibliografia relacionada com Processamento de Sinal.

Livros e artigos referenciados:

[27] - ANDREW D. CULHANE ; MARTIN C. PECKERAR ; C. R. K. MARRIAN
A Neural Net Approach to Discrete Hartley and Fourier Transform;
IEEE transactions on Circuits and Systems, VOL. 36 N. 5 p. 695, 1989/5.

[28] - DONALD FRASER
Interpolation by the FFT Revisited - An Experimental Investigation;
IEEE transactions on Acoustics, Speech and Signal Processing, VOL.37,N.5, 1989/5.

[29] - GEORGE DOVEL
FFT Analyzers Make Spectrum Analysis a Snap;
EDN 1989/1.

[30] - ANDERS AHLÉN ; MIKAEL STERNAD
Optimal Deconvolution Based on Polynomial Methods;
IEEE transactions on Acoustics, Speech and Signal Processing, VOL.37 N.2, 1989/2.

[31] - IRA HORDEN
An FFT Algorithm for MCS-96 Products Incl. Supporting Routines & Examples;
intel - Application Note AP-275, 1986/9.

[32] - H. YASSAIE
Discrete Fourier Transform with the IMS A100;
INMOS - INS Application Note 2 1986/6.

[33] - ALLAN O. STEINHARDT; JOHN MAKHOUL
On the Autocorrelation of Finite-Length Sequences;
IEEE transactions on Acoustics, Speech and Signal Processing VOL ASSP-33 N.6, 1985/12.

[34] - HENRIK V. SORENSEN ; D. L. JONES ; C. S. BURRUS ; M. T. HEIDEMEN
On Computing the Discrete Hartley Transform;
IEEE transactions on Acoustics, Speech and Signal Processing VOL ASSP-33, N. 4, 1985/10.

[35] - KENJI NAKAYAMA
A New Discrete Fourier Transform Algorithm Using Butterfly Structure Fast Convolution;
IEEE transaction on Acoustics Speech and Signal Processing VOL ASSP-33 N.4 1985/10 p. 77.

[36] - JONATHAN ALLEN
Computer Architecture for Digital Signal;
Proceedings of the IEEE, VOL. 73 N.5, 1985/5 p. 852.

[37] - RONMALD N. BRACEWELL
The Fast Hartley Transform;
Proceedings of the IEEE, VOL. 72 N. 8, 1984/8.

[38] - IRA S. KOVALINKA
Iterative nonparametric spectrum estimation;
IEEE transaction on Acoustics Speech and Signal Processing N.1 1984/2 p. 59.

[39] - P. R. ROOSER ; M. E. JERNIGAN
Fast Haar Transform Algorithm;
IEEE transaction on Computers VOL C-31 N. 2, 1982/2 p. 175.

[40] - S. GANESAN ; GIRIJA GOPALRATHNAM ; M. RENUKADEVI
A Real-Time Digital Signal Analyzer Correlator Averager Power Spectral Density Analyzer;
IEEE transactions on Industrial Electronics, VOL. IE-29 N. 1, 1982/2.

[41] - STEVEN M. KAY ; STANLEY MARPLE
Spectrum Analysis - A Modern Perspective;
Proceedings of the IEEE, VOL. 69 N. 11, 1981/11.

[42] - YOSHIAKI TADOKARO ; TATSUO HIGUCHI
Discrete Fourier Transform Computation via the Walsh Transform;
IEEE transaction on Acoustics Speech and Signal Processing VOL ASSP-26 N. 3, 1978/6 p. 236.

[43] - H. BUTIN
On an Ordering of Walsh Functions;
IEEE transaction on Computers VOL C-27 N. 1, 1978/1 p. 87.

[44] - BERNARD SALTZBERG
A Model for Relating Ripples in the EEG Power Spectral Density to Transient Patterns of Brain Electrical Activity Induced by Subcortical Spiking;
IEEE transactions on Biomedical Engineering, VOL. BME-23, N. 4, 1976/7 p. 355.

[45] - JACK R. SMITH
Automatic Analysis and Detection of EEG Spikes;
IEEE transactions on Biomedical Engineering, VOL. BME-21 N.1, 1974/1 , p. 1.

[46] - A. V. OPPENHEIM ; R. W. SHAFFER
Digital Signal Processing;
PRENTICE-HALL, Englewod Cliffs, N.J. 1975.

[47] - Autor não identificado
On Chip Speed and Power: NEC Digital Signal Processors;
Nec Application Note.

[48] - E. O. BRIGHAM
The Fast Fourier Transform;
PRENTICE-HALL, Englewod Cliffs, N.J. 1974.

[49] - M. ROBERT MIQUEL
Analyseurs Electroniques de Spectre;
Editora desconhecida.

[50] - RABINER ; SCHAFFER
Digital Processing of Speech Signals;
Editora desconhecida.

Livros e artigos não explicitamente referenciados:

ALFRED FETTWEIS
Wave Digital Filters: Theory and Practice;
Proceedings of the IEEE, VOL. 74 N. 2, 1986/2.

MICHAEL R. SMITH
Interpolation, Differentiation, Data Smoothing, and Least Square Fit to Data With Decreased Computational Overhead;
IEEE transactions on Industrial Electronics, VOL. 32 N. 2, 1985/5 , p. 135.

LAWRENCE R. RABINER ; JONT B. ALLEN
Short-Time Fourier Analysis Techniques for FIR System Identification and Power Spectrum Estimation;
IEEE transactions on Acoustics, Speech and Signal Processing, VOL. ASSP-27 N.2, 1979/4.

LAWRENCE R. RABINER ; JONT B. ALLEN
On the Implementation of a Short-Time Spectral Analysis Method for System Identification;
IEEE transactions on Acoustics, Speech and Signal Processing, VOL. ASSP-28 N.1, 1979/2.

J. A. CADZOW
Discrete Time Systems;
PRENTICE-HALL, Englewod Cliffs, N.J. 1973.

TREVOR J. TERRELL
Introduction to Digital Filters;
MACMILLAN PRESS.

C - Bibliografia relacionada com Técnicas de Inteligência Artificial.

Livros e artigos referenciados:

[51] - CATHERINE GROSS
Les Systemes a Bases de Conaissances et le Temps Reél;
Electronique Industrielle 1989/5.

[52] - CONTROLE ET PREVENTION
FIABEX - An Intelligent Tool for Technical Risk Control of Industrial Systems;
Artificial Intelligence - Conference Proceedings, Frankfurt RFA, 3-5 Junho 1987/6.

[53] - ERNST W. LEBSANFT
Experiences with SAVOIR in the Development of an Expert System Prototype for Preventive Maintenance of Electro-Mechanical Devices;
Artificial Intelligence - Conference Proceedings, Frankfurt RFA, 3-5 Junho 1987/6.

[54] - ALAN GIKES ; ALAN NEKHOM
Applying Expert Systems to On-line Fault Isolation;
Texas Instruments Engineering Journal - AI issue - . 1986/1 vol.3 M1 p. 19.

[55] - BRUCE CHAPMAN ; RICK YEAGER
Artificial Intelligence Applications in PWB manufacturing;
Texas Instruments Engineering Journal - AI issue - 1986/1 vol. 3 M.1 p. 34.

[56] - MICHAEL ROWE ; ANDREW VEITCH ; RENEE KEENER ; BERNIE LANTZ
An Adaptation/Learning Diagnostic System for Complex Domains E.T.;
Texas Instruments Engineering Journal - AI issue - 1986/1 V.3 M.1 p.66.

[57] - PAUL KLINE ; STEVEN DOLINS
Moving from Problems to Expert System Solutions;
Texas Instruments Engineering Journal - AI issue - 1986/1 vol. 3 M.1 p. 50.

[58] - JAMES M. DZIERZANOWSKI ; JOHN R. BOURNE ; R. SHIAVI ; H. S. H. SANDELL
GAITSPERT: An Expert System for the Evaluation of Abnormal Human Locomotion Arising from Stroke;
IEEE transactions on Biomedical Engineering, VOL. BME-32, N. 11, 1985/11 p. 935.

[59] - H. NIEMANN ; H. BUNKE ; I. HOFMANN ; G. SAGERER ; F. WOLF ; H. FEISTEL
A Knowledge Based System for Analysis of Gated Blood Pool Studies;
IEEE transactions on Pattern Analysis and Machine Intelligence, VOL. PAMI-7, N.4 1985/5 p. 246.

[60] - KENNETH P. BIRMAN
Rule-Based Learning for More Accurate ECG Analysis;
IEEE transactions on Pattern Analysis and Machine Intelligence, VOL. PAMI-4, N.4 1982/7 p. 369.

Livros e artigos não explicitamente referenciados:

FLÁVIA MARIA C. RIBEIRO ; PAULO ALEXANDRE C. MARQUES
"LAYOUT" - Gerador de Configurações de Circuitos VLSI;
Laboratório Nacional de Engenharia Civil, ICT Informação Técnica, Informática (ICT ITI 70) 1987/3.

DAVID LAWRENCE
Prototype Resource Selection Program for Sentry Test Systems;
Texas Instruments Engineering Journal março,abril 1986/3 p. 51.

CHARLES W. CHAPOTON Jr.
AI Applications to the ETMP program;
Texas Instruments Engineering Journal - AI issue - 1986/1 vol. 3 M.1 p. 24.

HARRY TENNANT
The Redesign of Thought;
Texas Instruments Engineering Journal - AI issue - 1986/1 vol. 3 M.1 p. 9.

KAREN L. McGRAW
Artificial Intelligence: the Competitive Edge in Integrated Systems Development;
Texas Instruments Engineering Journal - AI issue - 1986/1 vol. 3 M.1 p. 12.

D - Bibliografia relacionada com Regularimetria Têxtil.

Livros e artigos referenciados:

[61] - J. L. MONTEIRO; A. C. SILVA; C. COUTO; E. C. SILVA
Utilisation de la Transformée Rapide de Fourier;
L'Industrie Textile Nº 1024 1989/11

[62] - I. SHARIEFF ; S. G. VINZANEKAR ; T. NARASIMHAM
Spectral Analysis of Yarn Irregularity and its Relations to other Yarn Characteristics;
Textile Research Journal 1983/10 p. 606.

[63] - J. A. TOTTMAN ; K. SLATER
The Use of Microcomputers in Deriving Variance-Length Curves;
Journal of Textile Institute N. 1983/3 p. 103.

[64] - R. FURTER
Evenness Testing in Yarn Productions PART I & II;
The TEXTILE INSTITUTE

Diversos Manuais de equipamento USTER, Keisokki,...

E - Bibliografia diversa.

Livros e artigos referenciados:

[65] - CHRISTOPHER CHATFIELD
Statistics for Technology- A Course in Applied Statistics;
CHAPMAN and HALL London, 1978.

[66] - INMOS
Transputer Reference Manual;
INMOS.

[67] - J. F. TIRADO ; J. J. RUZ ; M. MELLADO
The WITMI System: A Multimicrocomputer-Based Syst. for Hotel Automation;
IEEE transactions on Industrial Electronics, VOL. IE-31, N. 4, 1984/11 , p. 317.

[68] - HUBERT D. KIRRMANN ; FELIX KAUFMAN
Poolpo - A Pool of Processors for Process Control Applications;
IEEE transactions on Computers VOL: C-33 N. 10, 1984/10 p. 869.

[69] - BRUCE G. BATCHELOR et al.
Pattern Recognition;
PLENUM PRESS New York 1978.

Livros e artigos não explicitamente referenciados:

DAVID F. STOUT ; MILTON KAUFMAN
Handbook of Operational Amplifier Circuit Design;
McGRAW-HILL BOOK COMPANY.

STEVEN WHEELWRIGHT ; SPYROS MAKRIDAKIS
Forecasting Methods for Management;
JOHN WILEY, New York 1978.

APÊNDICES.

Apêndice A. Especificação das primitivas de comunicação.

A.1 Especificação de algumas primitivas a residir nas Estações remotas.

Especificam-se algumas das primitivas usadas no algoritmo de comunicações.

A.1.1 Variáveis globais

Dummy - variável definida mas não utilizada, excepto para garantir concordância na passagem de parâmetros.
t - variável que indica o instante de tempo actual, e está dependente de um relógio ("timer") funcionando em paralelo com a unidade de processamento.
buffer_rx - variável tipo "record" com o "frame" recebido da rede.
buffer_tx - idem para a transmissão.
estado - variável que indica o estado presente da estação.

A.1.2 Algoritmos e especificação dos procedimentos, funções e teste.

As primitivas que se descrevem implementam os passos do algoritmo base representado na figura 52 e resultante da especificação definida na figura 51 (***Parte 3***).

Esta forma de desenvolvimento de programas, com uma especificação muito precisa desde o início, embora desvantajoso em alguns aspectos foi compensador.

A principal desvantagem esteve relacionada com o tempo despendido na especificação de todo o módulo de comunicações e posterior validação.

Estes aspectos devem-se, por um lado, à não existência como ferramenta suficientemente "standard", de programas de especificação formal, ou para o presente caso de especificação e validação de protocolos.

Apesar destas considerações pode-se afirmar com toda a certeza que a metodologia adoptada permitiu 'atalhar' muitas curvas do processo, apresentado como resultado um programa bastante modular e de muito fácil modificação.

A título de exemplo mostra-se o refinamento de alguns dos passos do algoritmo.

I. [recebeu comando x] (função teste de condição)

```
Declaração:
      recebeu(x)
Variáveis de entrada:
      x - Variável que contém o código do comando que se quer testar ter sido
      recebido.
Algoritmo:
      buffer_rx.tipo =x
```

II. [recepção correcta] (função teste de condição)

```
Declaração:
      frame_correcto
Algoritmo:
      buffer_rx.sumcheck = buffer_rx.sum_calculado
```

III. [estava desligado] (função teste de condição)

```
Declaração:
      estava_ligado
Algoritmo:
      estado = ligado
```

IV. [estava ligado] (função teste de condição)

```
Declaração:
      estava_ligado
Algoritmo:
      estado <> desligado
```

V. [passa a estado X] (procedimento)

```
Declaração:
      passa_estado(x)
Variáveis de entrada:
      x - variável com o código do estado que passa a ter a estação.
Algoritmo:
      estado <-- x
```

VI. [envia resposta X] (procedimento)

```
Declaração:
      envia_resposta(x, Micro, Pont_campo_inf, flag, Sequencia)
Variáveis de entrada:
      x - resposta a enviar para o microcomputador central.
      Micro - Constante com o valor do endereço do Microcomputador central.
      Pont_campo_inf - Ponteiro para a zona de memória onde reside a informação
      a enviar no campo de informação do "frame".
Variáveis de saída:
      flag - Inicialização da variável que controla o fim do envio.
      Sequencia - Variável que indica a numeração da resposta enviada ao
      microcomputador central.
Procedimentos a chamar:
      compoe_frame - Prepara o "frame" para enviar, construindo cada campo
      conforme a informação passada
      transmite_frame - Inicia a transmissão do "frame" preparado.
Algoritmo:
      compoe_frame(x, Micro, Pont_campo_inf, Sequencia)
      Retransmissões <-- 0
      flag <-- 0
      transmite_frame
```

VII. [reenvia resposta x] (procedimento)

```
Declaração:
      re_envia_resposta(x, Micro, Pont_campo_inf, Retransmissoes, flag,
      Sequencia)
Variáveis de entrada:
      x - Resposta a enviar para o microcomputador central.
      Micro - Constante com o valor do endereço do Microcomputador central.
      Pont_campo_inf - Ponteiro para a zona de memória onde reside a informação
      a enviar no campo de informação do "frame".
Procedimentos a chamar:
      compoe_frame - Prepara o "frame" para enviar, construindo cada campo
                           conforme a informação passada
      transmite_frame - Inicia a transmissão do "frame" preparado.
Algoritmo:
      compoe_frame(x, Micro, Pont_campo_inf, Sequencia)
      transmite_frame
```

VIII. [prepara campo de informação recebido] (procedimento)

```
Declaração:
      prepara_campo_inf(msg_rx)
Variáveis de entrada/saída:
      msg_rx - Ponteiro para a zona de memória com a mensagem a receber
Algoritmo:
      msg_rx <-- buffer_rx.inf
      msg_rx <-- msg_rx + buffer_rx.num_bytes
```

IX. [preparar comunicação a efectuar] (função)

```
Declaração:
      preparar_com(N_segmentos,sgm_sem_ack, segmentos, sgm
                   ultima_conf, pont,  bytes_frame, N_bytes_msg)
Variáveis de entrada e saída:
      N_segmentos - Número de segmentos a enviar.
      sgm_se_ack - Contador do número de "frames" já recebidos sem
      reconhecimento.
      segmentos - Contador do número de segmentos já enviados e reconhecidos.
      sgm - Conta o número de "frames" enviados com ou sem reconhecimento
      ultima_conf - Indica qual  o último "frame" reconhecido.
      pont - ponteiro para a zona de memória onde está a mensagem a segmentar e
      enviar.
      bytes_frame - Número de bytes que podem compor cada "frame"
      N_bytes_msg - Número de bytes que ocupa a mensagem a enviar.
Algoritmo:
      N_sgm <-- int((N_bytes_msg-1)/bytes_frame)+1
      flag_tx <-- 0
      sgm_sem_ack <-- 0
      segmentos <-- 0
      ultima_conf <-- -1
      sgm <-- 1
      preparar_com <-- pont
```

A.1.3 Algoritmos finais das primitivas de comunicação nas estações remotas.

Como foi referido, dado as estações remotas serem escravas não podem por si só estabelecer uma ligação e transferir dados sem prévio consentimento (ou melhor, senão sob pedido) do nó mestre, o microcomputador central.

Sendo assim não são implementadas 3 primitivas associadas a cada fase da concretização de uma ligação, como deverá acontecer no módulo a residir no microcomputador central, mas apenas uma primitiva que é executada com a recepção de um comando; na prática esta recepção desencadeia uma interrupção.

No algoritmo definido, existem 4 estados principais associados a fases distintas da comunicação, LIGADO, DESLIGADO, ENVIAR MENSAGEM e RECEBER MENSAGEM. Estes estados são representados por variáveis globais.

Existem ainda algumas variáveis globais (figura A1) que permitem saber, quando chega um comando, em que fase do estado está o módulo de comunicação; note-se que toda a ligação entre os diversos módulos é efectuada através de variáveis (ou ponteiros para variáveis que não deixam de ser por sua vez variáveis).

Na figura A2 apresentam-se os algoritmos que implementam as regras estabelecidas por este protocolo; nestas figuras apresenta-se a versão algorítmica mais refinada. Na implementação real, não se modificando esta estrutura algorítmica, recorreu-se ao "assembler" para implementar as primitivas, o que originou algumas adaptações indispensáveis.

```
Variáveis globais:
        Estado - variável que indica o estado presente do módulo de
          comunicação.
        Flag_transmissão - Variável que indica ter entrado no estado de enviar
          mensagem.
        Flag_recepção - Variável que indica ter entrado no estado de recepção
          de mensagem.
        msg_rx - Ponteiro para a zona de memória que irá conter a mensagem a
          receber.
        msg_tx - idem para a mensagem a transmitir.
        N_bytes_msg_tx - Nº de bytes da mensagem a transmitir.
        bytes_frame - nº de bytes do campo de informação de cada "frame" (em
          geral 64).
        modulo - Nº de "frames" que podem ser enviados sem confirmação.
        N_re_tx - Número máximo de retransmissões autorizadas.
        t_limite - Tempo limite para a ocorrência de uma resposta.
        erro - Variável que indica o tipo de erro ocorrido (nesta camada) ou
          eventualmente o sucesso.
Variáveis locais:
        Re_tx - Variável que indica o nº de retransmissões efectuadas.
Constantes:
        As indicações de erro ou sucesso ocorridas:
        As indicações do estado:
        LIGADO, DESLIGADO, ENVIAR_MENSAGEM e RECEBER_MENSAGEM.
```

Figura A1. Variáveis e constantes associadas à primitiva final de comunicação residente na estação remota

Tal como referido no início, o módulo de comunicação deve funcionar concorrentemente com os programas em curso; para a concretização destes objectivos é necessário garantir que parte deste módulo seja 'acordado' apenas quando necessário como acontece quando estamos na situação de ser transmitido ou recebido um "frame". Após a execução das tarefas impostas pelo protocolo é abandonado o módulo de comunicação, permitindo a continuação do programa principal, até que ocorra uma situação que obrigue à tomada de uma acção; este retorno ao módulo de comunicação pode ocorrer quando chegar um novo comando, tiver sido atingida uma situação de erro (p.ex. "time-out") ou a estação passe a ter pronta a enviar a informação pedida pelo microcomputador.

Relativamente ao algoritmo pormenorizado descrito na figura A2, este ponto é exemplificado ao constatar-se que o módulo de comunicação só pode ser abandonado nos passos em que se envia ou espera recepção dos caracteres de um "frame".

Para a implementação da primeira versão deste módulo não respeitámos esta condição de existência de concorrência; o algoritmo foi, pelo contrário, desenvolvido como se fosse o módulo principal, tendo sido introduzida esta capacidade de concorrência "a posteriori".

```
no caso de
  recebeu(ON) e frame_correcto então
    se estava_desligado então
      erro <-- sucesso_on
      passa_estado(LIGADO)
      envia_resposta(ACK, Micro, Dummy, Dummy, Dummy)
    senão
      envia_resposta(ACK, Micro, Dummy, Dummy, Dummy)
                    "já estava ligada! Não era necessário refazer ligação"
  recebeu(OFF) e frame_correcto então
    erro <-- sucesso_off
    passa_estado(DESLIGADO)
    envia_resposta(ACK, Micro, Dummy, Dummy, Dummy)
  recebeu(RNR) e frame_correcto e estava_ligado então
    passa_estado(LIGADO)
    envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
  recebeu(NACK) e frame_correcto e estava_ligado então
    se estado <> ENVIAR_MENSAGEM então
      passa_estado(LIGADO)
      erro <-- erro_nack_out
      envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
    senão
      se  Re_tx < N_re_tx então
        Re_tx <-- Re_tx + 1
        se (ultima_conf < 0) ou (buffer_rx.seq> ultima_conf) então
          se buffer_rx.seq>modulo então
            x <-- -1- ultima_conf
          senão
            x <-- buffer_rx.seq-ultima_conf
          se segmentos + x <= sgm então
            segmentos <-- segmentos + x
        senão
          x <-- buffer_rx.seq + módulo - ultima_conf
          se segmentos + x <= sgm então
            segmentos <-- segmentos + x
        se buffer_rx.seq > módulo então
          ultima_conf <-- -1
        senão
          ultima_conf <-- buffer_rx.seq
        Seq <-- mod(sgm,módulo)
        re_envia_resposta(DATA, Micro, pont_campo_inf, Dummy, Seq)
      senão
        passa_estado(LIGADO)
        erro <-- erro_nack_in
        envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
```

Figura A2. Algoritmo final da primitiva de comunicação (residente nas estações remotas).

```
recebeu(RR) e frame_correcto então
    se  estado=LIGADO  ou estado=RECEBER MENSAGEM  então
        se N_bytes_msg_tx >0 então
            passa_estado(ENVIAR_MENSAGEM)
            envia_resposta(ACK, Micro, Dummy, Dummy, Dummy)
            pont_campo_inf <-- preparar_com(N_segmentos, sgm_sem_ack,
                                            segmentos, sgm, ultima_conf,
                                            msg_tx, flag_tx, bytes_frame,
                                            N_bytes_msg_tx)
            Seq <-- Mod(sgm, modulo)
            erro <-- enviando
            envia_resposta(DATA, Micro, pont_campo_inf, Dummy, Seq)
        senão
            erro <-- sucesso_envio
            envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
recebeu(ACK) e frame_correcto e estava_ligado então
    se estado = ENVIAR_MENSAGEM então
        se (ultima_con<0) ou (buffer_rx.seq > ultima_conf) então
            x <-- buffer_rx.seq - ultima_conf
        senão
            x <-- buffer_rx.seq + módulo - ultima_conf
        se segmentos + x <= sgm então
            segmentos <-- segmentos + x
            ultima_conf <-- buffer_rx.seq
        se (segmentos < N_segmentos)ou(segmentos + x > sgm) então
            se sgm - segmentos < módulo então
                se sgm< N_segmentos então
                    sgm <-- sgm+1
                    Seq <--  mod(sgm, módulo)
                    erro <-- enviando
                    envia_resposta(DATA,Micro, pont_campo_inf, Dummy, Seq)
                senão
                    erro <-- erro_parte_msg_por_reconhecer
                    passa_estado(LIGADO)
                    envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
            senão
                erro <-- erro_parte_mensagem_por_enviar
                passa_estado(LIGADO)
                envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
        senão
            passa_estado(ligado)
            erro <-- sucesso_envio
            envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
    senão
        erro <-- erro_msg_por_enviar
        passa_estado(LIGADO)
        envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
recebeu(DATA) e frame_correcto e estava_ligado então
    se  (estado=LIGADO)ou(estado=RECEBER_MENSAGEM) então
        passa_estado(RECEBER_MENSAGEM]
        prepara_campo_inf(msg_rx)
        erro <-- recebendo
        envia_resposta(ACK, Micro, Dummy, Dummy, buffer_rx.numeração)
    senão
        erro <-- enviando
        envia_resposta(RR, Micro, Dummy, Dummy, Dummy)
senão
    envia_resposta(NACK, Micro, Dummy, Re_tx, Dummy, Dummy)
```

Figura A2. (*Continuação*). Algoritmo final da primitiva de comunicação.

Esta implementação da concorrência foi concretizada como se esquematiza com exemplos na figura A3.

Descrição do comando ON

```
;+-----------------------------------+
;|      RESPOSTA AO COMANDO ON       |
;+-----------------------------------+
COMANDO_ON:
   MOV    ESTADO,#LIGADO
                  ;PASSA A ESTADO LIGADO.
   POP    PSW
   POP    DPL
   POP    DPH

   POP    ACC
   POP    B
   POP    0
   MOV    ERRO_TEMP,#SUCESSO_ON
   %ENVIA_RESPOSTA
      (ACK,MICRO,DUMMY,DUMMY,
      (ACK,MICRO,DUMMY,DUMMY,
      DUMMY,DUMMY,DUMMY,DUMMY)
                  ;ENVIA 'ACKNOWLEDGE
   JMP    FIM_INT_SERIE
                  ;E ACABA DE SEGUIDA.
;+-------------------+
;|  FIM DA RESPOSTA  |
;|   AO COMANDO ON   |
;+-------------------+
```

Descrição da "macro" ENVIA_RESPOSTA

```
%*DEFINE(ENVIA_RESPOSTA(
      COM,MICRO,PTR_TIP,PTR_END_H,
      PTR_END_L,PTR_BAN,FLAG,SEQ)) (
MOV   RE_TX,#0       ;RETRANSMISSOES <- 0
MOV   %FLAG,#0
MOV   R1,#%COM       ;COMANDO A ENVIAR.
MOV   R2,#%MICRO     ;ESTAÇÃO A ENDEREÇAR.
MOV   R3,%PTR_TIP    ;TIPO DE ENDEREÇO
                     ;DA MENSAGEM A ENVIAR.
MOV   DPH,%PTR_END_H;ENDERECO PROP. DITO.
MOV   DPL,%PTR_END_L
MOV   R4,%PTR_BAN    ;BANCO DA MENSAGEM
MOV   R5,%SEQ        ;SEQUENCIAÇÃO DA MSG.
CALL  COMPOE_FRAME   ;COPIA MENSAGEM PARA
                     ;BUFFER DE ENVIO E
                     ;PREPARA OS OUTROS
                     ;CAMPOS A ENVIAR.
CALL  TRANSMITE_FRAME
      ↑     ↓        ;INICIA A TRANSMISSAO
sub-rotina que       ;DO FRAME QUE PASSARA
aceita interrupções
e portanto admite;A SER EFECTUADA POR
  concorrência.      ;INTERRUPT.
)
```

Figura A3. Esquema da resposta à recepção do comando ON (estabelecimento de ligação); apresenta-se o programa correspondente à resposta bem como o desenvolvimento da principal "macro". A concorrência é permitida durante a execução da rotina ***TRANSMITE_FRAME*** ; aqui, após ter transmitido um caracter do "frame" o módulo é abandonado até existir uma interrupção a informar que está pronto a enviar novo caracter; esta situação termina quando for transmitido o último caracter do "frame".

A.2 Algoritmos e estrutura dos dados de algumas subprimitivas do microcomputador central.

Neste ponto especificam-se mais pormenorizadamente algumas das primitivas usadas na descrição das primitivas Estabelecer, Efectuar e Terminar ligações, a residir no microcomputador central.

A especificação atinge o nível de chamada de procedimentos (referindo o nº, tipo e estrutura das variáveis necessárias), e também um 'refinamento' do algoritmo, não se pormenorizando mais, dado que, a partir deste ponto, passou-se à implementação recorrendo a uma linguagem.

A.2.1 Parâmetros gerais.

I. Variáveis globais.

Dummy - variável definida (declarada e com contexto) mas não utilizada. São em regra variáveis utilizadas para a passagem de parâmetros que não são utilizados no procedimento.
t - variável que indica o instante de tempo actual. Esta variável está associado ao relógio de tempo real que funciona em paralelo com a unidade de processamento.
buffer_rx - variável tipo "record" correspondente ao "frame" recebido pela linha de comunicação. O acesso a cada campo será do tipo : buffer_rx.flag, buffer_rx.tipo, etc.
buffer_rx - variável idêntica a anterior mas referente à transmissão.
flag_recepção - indica, quando em 1, a recepção de um "frame". Esta variável é actualizada automaticamente com a recepção de um "frame".

II. Variáveis de cada primitiva.

Em cada primitiva consideram-se ainda variáveis globais as que constam dos parâmetros incluídos na invocação da primitiva. Essas variáveis terão âmbito em qualquer procedimento, função de teste ou funções chamados pela primitiva. A título de exemplo refere-se que na primitiva estabelecer ligação as variáveis *Estação, Nº_retransmissões, tempo_limite e erro* são globais, não necessitando portanto de ser redefinidas e tendo valor em toda a primitiva.

III. Funções pré-definidas.

Existem algumas funções que são utilizadas e consideradas pré-definidas, de entre as quais:

Mod(X,Y) - Determina o resto da divisão de X por Y.
conteúdo(X) - Fornece como resultado o conteúdo da variável X.

B. Algoritmos e especificação dos procedimentos, funções e testes.

I. [envia comando x] (procedimento)

Declaração:
envia_comando(x,Estação, Pont_campo_inf, Retransmissões, flag, Sequencia)
Variáveis entrada:
x - comando a enviar.
Estação - Código da estação a receber comunicação.
Pont_campo_inf - Ponteiro para a zona de memória onde reside a informação a enviar no campo de informação do "frame".
Variáveis de saída:
Retransmissões - Conta o nº de retransmissões do "frame" já efectuadas.

```
      flag - Indicação de fim.
      Sequencia - Variável que indica a numeração do comando ou resposta a
      envia, caso pertençam ao grupo B, C ou D.
Procedimentos a chamar:
      compoe_frame - Prepara o "frame" na memória ("buffer") de transmissão.
      transmite_frame - Inicia a transmissão do "frame" no "buffer" de envio.
Algoritmo:
      compoe_frame( x , Estação, Pont_campo_inf, Sequencia)
      Retransmissões  <--  0
      transmite_frame
      flag <--  0
```

II. [reenvia comando x] (procedimento)

```
Declaração:
      re_envia_comando(Retransmissões)
Variáveis de entrada e saída:
      Retransmissões - Conta o nº de retransmissões do "frame" já efectuadas.
Procedimentos a chamar:
      transmite_frame - Inicia a transmissão do "frame" no "buffer" de envio.
Algoritmo:
      Retransmissões  <--  Retransmissões + 1
      transmite_frame
```

III. [não termina flag]. (função teste de condição)

```
Declaração:
      nao_termina(flag)
Variável de entrada:
      flag - Variável que é testada com 0.
Algoritmo:
      flag= 0
```

IV. [espera resposta ou esgota o tempo para ela] (procedimento)

```
Declaração:
      espera(tempo_limite,tipo_fim)
Variável de entrada:
      tempo_limite - Tempo máximo aceitável para a recepção de uma resposta.
Variável de saída:
      tipo_fim - Indica como terminou este ciclo. Se valor for 1 indica ter
      terminado por recepção de um resposta.
Algoritmo:
      enquanto (flag_recepção<>1) e (t < tempo_limite)
        nop
      se flag_recepção =1 então tipo_fim <-- 1
      senão tipo_fim <-- 0
```

V. [recebeu resposta antes de esgotar tempo limite] (função teste de condição)

```
Declaração:
      recebeu_sem_esgotar(tipo_fim)
Variável de entrada:
      tipo_fim - Variável accionada por outro procedimento que indica a
      recepção ou não de uma resposta.
Algoritmo:
      tipo_fim = 1
```

VI. [receber resposta x] (teste de condição)

```
Declaração:
      recebeu(x)
```

Variável de entrada:
x - Variável com o código da resposta que se pretende testar ter sido recebido.
Algoritmo:
buffer_rx.tipo = x (x é ao código identificador de comando ou resposta).

VII. [termina flag com y] (função)

Declaração:
termina(flag,y)
Variáveis de entrada e saída:
flag - Variável que será usada como marca de detecção de fim.
erro - Variável que indica como terminar.
Algoritmo:
flag <-- 1
termina <-- y (y corresponde ao código de erro ou de sucesso)

VIII. [esgotou número de x] (função teste de condição)

Declaração:
esgotou(x ,N_x)
Variável de entrada:
x - Variável a testar se esgotou o nº limite.
N_x - Limite que x não pode ultrapassar.
Algoritmo:
x >= N_x

IX. [preparar comunicação a efectuar] (função)

Declaração:
preparar_com(N_sgm,sgm_sem_ack, segmentos, sgm, ultima_conf, pont,
flag_tx, bytes_frame, N_bytes_msg)
Variáveis de entrada e saída:
N_sgm - Número de segmentos a enviar.
sgm_se_ack - Contador do número de "frames" recebidos sem reconhecimento.
segmentos - Contador do número de segmentos já enviados e reconhecidos.
sgm - Conta o número de "frames" enviados com ou sem reconhecimento
ultima_conf - Indica qual o último "frame" reconhecido.
pont - ponteiro para a memória onde está a mensagem a segmentar e enviar.
flag_tx - flag que é inicializada. Controla o fim de transmissão.
bytes_frame - Número de bytes que podem compor cada "frame"
N_bytes_msg - Número de bytes que ocupa a mensagem a enviar.
Algoritmo:
N_sgm <-- int((N_bytes_msg-1)/bytes_frame)+1
flag_tx <-- 0
sgm_sem_ack <-- 0
segmentos <-- 0
ultima_conf <-- -1
sgm <-- 0
preparar_com <-- pont

X. [actualiza ponteiro e contadores] (procedimento)

Declaração:
actualiza_com(pont, bytes_frame, modulo
confirmação, ultima_confirmação,
sgm, segmentos, sgm_sem_ack,
erro_TX)
Variáveis:
pont - ponteiro para zona de memória com a informação a enviar no próximo "frame".
bytes_frame - Número de bytes que podem compor cada "frame"
modulo - Número de "frames" que podem ser enviados sem confirmação.

```
      confirmação - Indica qual o "frame" reconhecido (se é que foi algum) na
      última comunicação.
      ultima_confirmação - Indica qual o último "frame" reconhecido.
      sgm - Conta o número de "frames" enviados com ou sem reconhecimento
      segmentos - Contador do número de segmentos já enviados e reconhecidos
      pela estação central.
      sgm_se_ack - Contador do número de "frames" já recebidos sem
      reconhecimento.
      erro_TX - Erro detectado (ou sucesso caso em que o erro não ocorreu) na
                última transmissão.
Algoritmo:
      se erro_TX = sucesso_e_confirmação então
        pont <-- pont + bytes_frame
        se  buffer_rx.seq> ultima_confirmação
           então  x <-- confirmação - ultima_comfirmação
           senão x <-- modulo + confirmação - ultima_confirmação
        sgm <-- sgm + 1
        segmentos <-- segmentos + 1
        sgm_sem_ack <-- sgm - segmentos
      senão
        pont <-- pont + bytes_frame
        sgm_sem_ack <-- sgm_sem_ack + 1
        sgm <-- sgm + 1
```

XI. ["frame" recebido correcto] (função teste de condição)

```
Declaração:
      frame_correcto
Algoritmo:
      buffer_rx.sumcheck = buffer_rx.sum_calculado
```

XII. [prepara campo de informação recebido] (procedimento)

```
Declaração:
      prepara_campo_inf(msg_rx)
Variáveis de entrada e saída:
      msg_rx - Ponteiro para a mensagem a receber.
Algoritmo:
      msg_rx <-- buffer_rx.inf
      msg_rx <-- msg_rx + buffer_rx.num_bytes-9
```

A.2.2 Algoritmos finais das primitivas de comunicação do microcomputador central.

Após esta especificação pormenorizada do algoritmo, foi escrito e 'compilado' o programa que implementa este módulo; a codificação foi efectuada recorrendo à linguagem MASM86 versão 5 ("Assembler" para o microprocessador 8086 e similares); a opção por esta linguagem prende-se com a necessidade de garantir a optimização dos tempos de execução.

Um dos aspectos mais relevantes da metodologia usada reside no facto de, após a escrita destes algoritmos finais, se ter passado de imediato à codificação. Este ponto confirma a ideia de que a estruturação e modularidade de um programa dependem menos da linguagem a utilizar, que da correcta e precisa especificação algorítmica.

I - Primitiva Estabelecer ligação.

Um dos aspectos particulares desta primitiva reside na possibilidade desta chamar outra primitiva de comunicação, *terminação de ligação*,.

Esta opção ficou a dever-se à possibilidade de, por qualquer situação anómala, a estação central não tenha conseguido obter (ou 'compreender') a resposta da estação escrava; supondo a possibilidade de a estação escrava ter passado ao estado de **ligada** devido à recepção do comando ON respectivo, é necessário prever a acção de desligá-la, por forma a garantir-se que esta fica no estado correcto.

Assim há a garantia de que o estabelecimento de ligação só é considerado válido se existir resposta correcta. Se esta não ocorrer, manter-se-á o estado de desligado, pelo que as camadas de "software" superiores serão informadas da ocorrência de um erro fatal. Apresenta-se na figura A4 o algoritmo final desta primitiva.

```
envia_comando(ON, Estação, Dummy, Re_tx, flag_fim, Dummy)
espera(t_limite, tipo_fim)
enquanto nao_termina(flag_fim)
   se recebeu_sem_esgotar(tipo_fim) então
      no caso de
         recebeu(ACK) então erro <-- termina(flag_fim,sucesso)
         recebeu(NACK) então
            se esgotou(Re_tx,N_re_tx)
            então erro<-termina(flag_fim,erro_NACK)
            senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
      senão  " na situação de nenhuma das hipóteses do caso ser seleccionada"
            se esgotou(Re_tx,N_re_tx) então
               erro <-- termina(flag_fim,erro_resposta_incorrecta)
            senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
   senão
      se esgotou(Re_tx, N_re_tx) então erro<-- termina(flag_fim,erro_timeout)
      senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
se erro<> 0
   então terminar(Estação,N_re_tx,t_limite,N_desligar,erro_desligar)
```

Figura A4. Algoritmo final da primitiva de estabelecimento de ligação.

Realce-se ainda o facto de a execução da primitiva de terminar ligação poder obrigar à transmissão de diversos "frames" de desligar pela linha de comunicação, como veremos na sua descrição.

Tal como para a utilização tradicional desta primitiva existe um número limite para efectuar tentativas de terminar ligação. Dado este valor ser definido pelo utilizador (ou camada de programação superior) é necessário que, ao chamar a primitiva de estabelecimento de ligação seja indicado (ou melhor 'passado') o valor máximo de tentativas de desligar autorizadas.

II - Primitiva efectuar transacção.

Na figura A5, apresenta-se o algoritmo final da primitiva de transacção de dados; neste, desenvolvido seguindo a metodologia "top-down", indicam-se as chamadas a outros procedimentos, alguns dos quais bastante complexos e que implementam por si só 'subprimitivas' (transacção_data e transacção_RR). A utilização desta metodologia teve a única facilidade de simplificar a especificação do algoritmo.

Estas 'subprimitivas' serão expostas em pormenor noutras figuras a apresentar posteriormente.

Alguns aspectos são ainda de realçar neste algoritmo; é possível efectuar uma transacção de dados em que estes circulam apenas num sentido, ou seja, a estação central pode decidir enviar um conjunto de dados para uma estação remota sem que esta tenha de responder com dados. Para existir resposta da estação remota com dados será necessário que a estação central peça essa informação através do envio do *comando RR*.

Outro aspecto a ter em conta prende-se com a possibilidade de a estação central enviar um grupo de "frames" sem receber reconhecimento de *boa recepção* "frame" a "frame"; este reconhecimento poderá ser enviado pela estação remota para um grupo deles, definido pela variável módulo. Este processo permite à estação remota, com uma capacidade de processamento inferior à da estação central, efectuar a composição e envio do "frame" de reconhecimento apenas quando tiver recebido um lote maior de informação.

II.1 - Subprimitiva transacção de comando DATA.

Na figura A6 apresentam-se as características da 'subprimitiva' transacção_data.

No algoritmo desta 'subprimitiva' são previstas diversas transacções de "frames" entre as estações, quer devido ao tráfego normal quer devido a reenvio de comandos ou

respostas não compreendidos (p. ex. por estarem fora do contexto) ou por terem sido incorrectamente recebidos.

```
pont_campo_inf <-- preparar_com(N_sgmentos,  sgm_sem_ack,  segmentos, sgm
                                ultima_conf,   msg_tx,   flag_tx,
                                bytes_frame,   N_bytes_msg_tx)
enquanto nao_termina(flag_tx)
   se  não(esgotou(sgm_sem_ack, modulo) ) e  não(esgotou(sgm, N_segmentos)
   então
      seq <-- Mod(sgm, módulo)
      Transacção_data( Estação, N_re_tx,N_rnr, modulo, t_limite,
                       pont_campo_inf, sgm_sem_ack,  Seq, erro_DATA)
      se (erro_DATA=sucesso_e_confirmação)ou(erro_DATA=sucesso_sem_confirmação)
      então
         actualiza_com(pont, bytes_frame,modulo, confirmação,ultima_confirmação,
                       sgm, segmentos, sgm_sem_ack, erro_DATA)
      senão
         erro_tx <-- termina(flag_TX, conteúdo(erro_DATA))
   senão
      se esgotou(segmentos,N_segmentos) então
         erro_tx <-- termina(flag_tx,sucesso_tx)
      senão
         erro_tx <--termina(flag_tx,erro_timeout)
         erro_rx <--termina(flag_rx,erro_BASE)
se  nao_termina(flag_rx)
   Transacção_RR( Estação, N_re_tx, N_rnr, t_limite, erro_RR)
   se erro_RR =sucesso então
      ultimo_ack <-- -1
      enquanto nao_termina(flag_rx)
         espera(t_limite,tipo_fim)
         se recebeu_sem_esgotar(tipo_fim) então
            no caso de
               recebeu(RR) então erro_rx <-- termina(flag_rx, sucesso)
               recebeu(RNR) então erro_rx <-- termina(flag_rx,erro_RNR)
               recebeu(DATA) então
                  se frame_correcto então
                     envia_comando(   ACK, Estação,
                                      Dummy, Dummy, Dummy,
                                      buffer_rx.numeração)
                     ultimo_ack <-- buffer_rx.numeração
                     prepara_campo_inf(msg_rx)
                  senão
                     envia_comando(   NACK, Estação,
                                      Dummy, Dummy, Dummy,
                                      ultimo_ack)
            senão
               erro_rx <-- termina(flag_rx,erro_resposta_incorrecta)
         senão
            se esgotou(Re_tx,N_re_tx) então
               erro_rx <--termina(flag_rx,erro_timeout)
            senão
               Transacção_RR(Estação, N_re_tx, N_rnr, t_limite, erro_RR)
      senão
         erro_rx <-- conteudo(erro_RR)
```

Figura A5. Algoritmo final da primitiva de transacção de dados.

Tal como para as restantes primitivas, existem diversos contadores associados ao limite máximo de retransmissões autorizadas ou de respostas de RNR (indicadoras de a estação remota não estar pronta a transmitir).

```
envia_comando(DATA, Estação, pont_campo_inf,re_tx_DATA,flag_DATA, Seq)
espera(t_limite, tipo_fim_DATA)
RNR <-- 0
enquanto nao_termina(flag_DATA)
   se recebeu_sem_esgotar(tipo_fim_DATA)
      no caso de
         recebeu(ACK) então
            erro_DATA <-- termina(flag_DATA, sucesso_e_confirmação)
         recebeu(NACK) então
            se esgotou(re_tx_DATA, N_re_tx) então
               erro_DATA <-- termina(flag_DATA, erro_NACK)
            senão
                  re_envia_comando(Re_tx)
                  espera(t_limite, tipo_fim)
         recebeu(RNR) então
            RNR <-- RNR + 1
            se esgotou(RNR, N_RNR) então
               erro_DATA <-- termina(flag_DATA, erro_RNR)
            senão
                  re_envia_comando(Re_tx)
                  espera(t_limite, tipo_fim)
         recebeu(RR)
            erro_DATA <-- termina(flag_DATA, sucesso_sem_confirmação)
      senão
         se esgotou(re_tx_DATA, N_re_tx) então
            erro_DATA <-- termina(flag_DATA, erro_NACK)
         senão
            re_envia_comando(Re_tx)
            espera(t_limite, tipo_fim)
```

Figura A6. Algoritmo final da primitiva de transacção de dados.

II.2 - Subprimitiva transacção de comando RR.

Na figura A7 apresentam-se as características da 'subprimitiva' transacção_RR, sob a forma do algoritmo final.

Esta 'subprimitiva' tem a tarefa de entabular uma 'conversa' da parte da estação central com a remota, com vista a esta última vir a responder com a transmissão de um bloco de dados.

Também nesta 'subprimitiva' existem variáveis tipo contador para detectar quando é atingido o limite máximo da recepção de "frames" incorrectos.

III - Primitiva terminar ligação.

Também para a primitiva de terminar ligação se apresentam, na figura A8, as respectivas características, sob a forma do algoritmo final da primitiva.

```
envia_comando(RR, Estação, Dummy, Re_tx, flag_fim_RR, Dummy)
espera(t_limite,tipo_fim_RR)
enquanto nao_termina(flag_fim_RR)
   se recebeu_sem_esgotar(tipo_fim_RR) então
      no caso de
         recebeu(ACK) então erro _RR<-- termina(flag_fim_RR, sucesso)
         recebeu (RNR) então erro_RR <-- termina(flag_fim_RR, erro_RNR)
         recebeu (NACK) então
            se esgotou(Re_tx, N_re_tx) então
               erro_RR <-- termina(flag_fim_RR, erro_NACK)
            senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
      senão
         se esgotou(Re_tx, N_re_tx) então
            erro_RR <-- termina(flag_fim_RR, erro_resposta_incorrecta)
         senão
            re_envia_comando(Re_tx)
            espera(t_limite, tipo_fim)
   senão
      se esgotou(Re_tx, N_re_tx) então
          erro <-- termina(flag_fim_RR, erro_timeout)
      senão
         re_envia_comando(Re_tx)
         espera(t_limite, tipo_fim)
```

Figura A7. Algoritmo final da primitiva de transacção de dados, para o passo de transacção de RR

Para esta primitiva foi dedicado um cuidado muito especial, dado que ao não ser possível garantir o fim de uma ligação entre as estações a que for escrava permanecerá no estado ligado o que pode não convir, pois pode provocar a ocorrência de tráfego de dados extemporâneo quando for efectuada uma ligação com qualquer outra estação remota.

Para minimizar essa possibilidade, foi prevista a hipótese de efectuar diversas tentativas de terminar a ligação, definindo no entanto um valor limite para o número autorizado (variável Desligar).

Para além deste aspecto diferente das primitivas anteriores, o algoritmo mantem as características tradicionais, possuindo também contadores para limitar o número máximo de retransmissões autorizadas para cada "frame".

Tal como para o caso de estabelecimento de ligação, esta primitiva se bem sucedida, será conseguida pelo envio de apenas um "frame" a que se seguirá a recepção de uma resposta do tipo ACK.

```
envia_comando(OFF, Estação, Dummy, Re_tx, flag_fim, Dummy)
espera(t_limite, tipo_fim)
enquanto nao_termina(flag_fim)
   se recebeu_sem_esgotar(tipo_fim) então
      no caso de
         recebeu(ACK) então erro <-- termina(flag_fim,sucesso)
         recebeu(NACK) então
            se esgotou(Re_tx, N_re_tx)
            então erro <- termina(flag_fim,erro_NACK)
            senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
      senão  " na situação de nenhuma das hipóteses do caso ser seleccionada"
            se esgotou(Re_tx,N_re_tx) então
                erro <-- termina(flag_fim,erro_resposta_incorrecta)
            senão
               re_envia_comando(Re_tx)
               espera(t_limite, tipo_fim)
   senão
      se esgotou(Re_tx, N_re_tx)
      então erro <-- termina(flag_fim, erro_timeout)
      senão
         re_envia_comando(Re_tx)
         espera(t_limite, tipo_fim)
se erro<> 0 então
   se  não(esgotou(Desligar, N_desligar) ) então
      Desligar <-- Desligar + 1
      terminar(Estação, N_re_tx,t_limite,N_desligar, Desligar,  erro)
```

Figura A8. Algoritmo final da primitiva de terminar ligação.

Falta apenas referir alguns dos aspectos mais marcantes na implementação destas primitivas.

Estas primitivas foram desenvolvidas em "assembler 86", tendo sido utilizado o desenvolvimento do algoritmo tal como indicado nas secções precedentes, isto é, partiu-se de uma definição pouco refinada até uma versão mais pormenorizada que foi utilizada para a escrita do programa final respectivo.

Dado estas primitivas serem no geral acedidas a partir de programas escritos em QBASIC (figura A9), foi necessário implementar pequenos procedimento que garantiam a passagem entre os diversos módulos; estes módulos são chamados do programa *pai* (módulo operativo, processamento de sinal, etc.) de forma igual à utilizada para a chamada a qualquer subprograma da linguagem QBASIC, em que as variáveis são passadas sob a forma de ponteiros quando a sua dimensão é grande ou por conteúdo (quando estas são individuais).

a) Chamada a partir do programa pai (em QBASIC)

```
     EndErro% = varptr(Erro%)              <<<< Endereço da variável Erro%
     EndDesligar% = varptr(Desligar%)      <<<< Endereço da variável Desligar%
     Erro% = -1%
     Desligar% = 0
     Estacao% = &h50           <<<< Endereço da estação
     NumReTx% = 3%             <<<< Nº de retransmissões antes de desistir
     NumDesligar% = 2%         <<<< Nº máximo de tentativas de desligar
     Tlimite% = 18% * 5%       <<<< tempo limite para chegar um "frame" = 5s
     Modulo%=8%                <<<< Nº de "frames" máximos sem receber "ACK"
     NumRNR%=4%          <<<< Nº de respostas RNR máximo antes de desistir
     EndMsgTX%=INT(SADD(MsgTX$)) <<<< Endereços dos "buffers" de
     EndMsgRX%=INT(SADD(MsgRX$)) <<<< transmissão e recepção
     EndInicialRX%=EndMsgRX%
     EndLastRX%=varptr(LastRX%)
     EndErroTX%=varptr(ErroTX%)      <<<< Endereços das variáveis dos erros
     EndErroRX%=varptr(ErroRX%)      <<<< de transmissão e recepção
     call Terminar(Estacao%,NumReTx%,NumDesligar%,Tlimite%,EndDesligar%,EndErro%)
             ^<<<< Chamada da primitiva terminar ligação
     .......
     call TRANSACCAO (Estacao%,EndMsgTX%,NumBytesTX%,BytesFrame%,
                      EndMsgRX%,FlagRX%,Modulo%,NumReTx%,NumRNR%,
                      Tlimite%,EndErroTX%,EndErroRX%,EndLastRX%)
             ^<<<< Chamada da primitiva transacção
```

b) Primitiva de desligar.

```
;DESLIGAR     Parametros  1-Codigo da estacao                 CS:ESTACAO
;             a receber   2-N. de retransmissoes              CS:N_RE_TX
;                         3-N. maximo de tentativas desligar  CS:N_DESLIGAR
;                         4-Tempo limite antes de falhar      CS:T_LIMITE
;                         5-N. tentativas de desligar         CS:DESLIGAR
;                         6-Erro ocorrido                     CS:ERRO
DESLIGAR     PROC  FAR
     MOV     AX,CS:ENTRADA       ;Verifica se entra nesta primitiva
     CPM     AX,LIGADO           ;a partir da primitiva ligar.
     JNE     NAO_ENTROU_LIGAR
     .....   ......
     ;Macro responsavel pelo envio do comando ON
     ENVIA_COMANDO    CS:DUMMY, CS:DUMMY, CS:FLAG_FIM, CS:DUMMY, OFF
             ^<<< Admite concorrência
     ;Macro responsavel por preparar o relogio no caso de nao chegar resposta
     ESPERA     CS:T_LIMITE, CS:TIPO_FIM
             ^<<< Admite concorrência
ENQUANTO_OFF_1:
     CMP     CS:FLAG_FIM,0
     JE      FAZER_OFF_1
     JMP     FIM_ENQUANTO_OFF_1
FAZER_OFF_1:
     .....   ......
```

Figura A9. Apresentação de pequenas partes dos programas que implementam as primitivas e das respectivas chamadas a partir do programa pai.

Como é perceptível pela definição do algoritmo, a característica de concorrência não é fundamental para a verificação da validade do algoritmo do módulo de comunicação; baseado nesta consideração, foram efectuados todos os testes sem

implementar à partida a concorrência. Após a validação do protocolo de comunicação com o recurso a diversas experiências, passou-se à implementação da concorrência, seguindo a metodologia já apresentada.

Para facilitar a implementação seguindo a estrutura modular utilizada na escrita do algoritmo, optou-se pela utilização de diversas "macros"; esta solução apresentou duas vantagens, uma que se prende com a minimização do tempo de processamento, outra com a garantia de se manter a "stack" em níveis constantes e baixos, factor que é importante para a implementação da concorrência.

Na figura A9 apresentam-se pequenas porções dos programas implementados, quer as chamadas às primitivas implementadas, quer a implementação de algumas dessas primitivas.

Apêndice B. Funcionamento e implementação do módulo interpretador.

Este programa, residente nas estações remotas, permite introduzir programas escritos numa linguagem (de mais alto nível que o "assembler"), possibilitando um controlo mais cuidado de todo o funcionamento, garantindo contudo o acesso a todos os recursos das estações. Tal como exposto no texto principal, este módulo permite a manutenção de programas de controlo mesmo em caso de serem alteradas as características do elemento processador.

O módulo interpretador possui diversos tipos de funcionamento:

- interactivamente com o utilizador;
- execução de módulos (diversos) previamente carregados.

Neste último caso, cabe ao operador, quer através da rede, quer através da consola portátil, iniciar a execução do módulo pretendido.

Quando da discussão das duas estruturas de comunicação, mostraram-se já diversas vantagens decorrentes da utilização deste método de funcionamento; de notar que a opção por esta metodologia não invalida a eventual utilização do funcionamento interactivo em certas circunstâncias.

Quando se opta por um funcionamento interactivo com um operador toda a acção decorre directamente das operações executadas por este; de entre as diversas opções, a principal diz respeito ao controlo do(s) programa(s) escritos nesta linguagem (desde a introdução até à verificação).

Dada o caracter da operação ser eventualmente moroso e repetitivo, logo muito propício à ocorrência de erros de operação, é conveniente utilizar processos auxiliares que ajudem o programador nestas tarefas, muito especialmente na introdução interactiva de programas.

Neste sentido, para além de diversos ecrãs de ajuda disponíveis (ver figura 54 do texto principal, Parte 3 e figura B1 neste apêndice), é fornecida informação sobre as instruções possíveis de utilizar, quando na situação de inserir instruções interactivamente.

Tendo ainda em vista minimizar a probabilidade de errar o utilizador não necessita introduzir todos os caracteres que definem uma determinada instrução, bastando para o efeito carregar nas teclas correspondentes aos 1ºs caracteres que

permitem discriminar completamente uma instrução (letras em maiúsculas na figura B1)

a)

```
|---------|-------------------------------------------------------|
|COMANDO  |                        SINTAXE
|---------|-------------------------------------------------------|
|MUdar    |01, ponteiro1,incl, n, ponteiro2,inc2, t
|Compactar|02, ponteiro1, n, ponteiro2
|ENviar   |03, ponteiro, n
|Receber  |04, tipo, banco
|Terminar |05
|SAltar   |06, ponteiro
|SOmar    |07, ponteiro, n
|SE       |08, ponteiro1, relacao, ponteiro2, ponteiro3
|EXecutar |09, ponteiro
|MArcar   |00, 00
|Lcall    |0A, n
|Filtrar  |0B, ponteiro1, n, lim.sup, lim.inf, ponteiro2
|Operar   |0C, ponteiro1,incl,funcao,ponteiro2,inc2,ponteiro3,inc3,n,t
|---------|-------------------------------------------------------|

     PRIMA UMA TECLA
```

b)

```
As estruturas ponteiro,ponteiro1,ponteiro2,ponteiro3 tem 3 campos:
    tipo    - 1 byte |0-codigo 1-Ram int. directa 2-Ram int. indirecta|
                     |3-Ram externa 4-Bit 5-Valor imediato             |
    endereco- 2 bytes
    banco   - 1 byte
A estrutura t tem 2 campos:
    tempo   - 2 bytes |qtd. de unidades de tempo        |
    unidades- 1 byte  |0-microseg. 1-miliseg. 2-segundos|
|RELACAO INSTRUCOES|COD.|
|  SE   |   OPERAR  |    ||ESPACO OCUPADO|     NOME DAS VARIAVEIS        |
|-------|-----------|----||--------------|-------------------------------|
| AND   |    AND    | 01 ||    4 bytes   | As estruturas ponteiro        |
| OR    |    OR     | 02 ||--------------|-------------------------------|
| XOR   |    XOR    | 03 ||    3 bytes   | A estrutura t                 |
| NAND  |    NAND   | 04 ||--------------|-------------------------------|
| NOR   |    NOR    | 05 ||    2 bytes   |{ponteiro,ponteiro1,ponteiro2,|
|  =    |    NXOR   | 06 ||              | ponteiro3}.endereco,t.tempo   |
| <>    |     *     | 07 ||--------------|-------------------------------|
|  >    |     /     | 08 ||    1 byte    | incl,inc2, lim.sup., lim.inf.|
|  <    |     +     | 09 ||              | relacao, tipo, banco,         |
| =<    |     -     | 0A ||              | t.unidades                    |
| >=    |    MOD    | 0B ||--------------|-------------------------------|
|-------|-----------|----|
```

Figura B1. Ecrãs de ajuda: a) com a especificação sintáctica de cada instrução e, b) explicando as estruturas de dados utilizadas e os códigos das operações lógicas e aritméticas das instruções SE e OPERAR.

Com vista a facilitar a utilização interactiva deste interpretador foram implementados diversos comandos que permitem gerir toda a informação introduzida ou a introduzir na estação remota; para além do comando de ajuda ("help") disponível, existem os restantes comandos directamente vocacionados para a gestão da programação.

Estes comandos são utilizados quer a partir da consola portátil, quer através da rede de comunicação por selecção de um caracter.

O comando **Listar** permite visualizar as instruções anteriormente introduzidas numa qualquer zona de memória, apresentando um resultado como o da figura B2.

```
JM>                    «««« selecção de comando inexistente !
Para consultar o ecra de ajuda (HELP) prima: ? ou ! ou H ou h
JM> Listar:            «««« selecção de comando listar (caracter l ou L)
      Tipo [0 a 4] :  00        «««« memória de código
      Endereco     :  0000      «««« endereço inicial
      Banco [P1]   :  00        «««« banco (indiferente para esta memória)
00:0000:00> [C] Compactar: ptr[00:C402:53] num[82FF] ptr[FF:FFFF:FF]
00:000B:00> [C] Compactar: ptr[53:EA25:82] num[F582] ptr[50:0205:83]
00:001B:00> [C] Compactar: ptr[52:85FF:FF] num[FFFF] ptr[FF:0200:31]
00:0058:00> [SE] Se: ptr[75:2080:C2] rel[98] ptr[02:0075:D0] ptr[E0:30E7:14]
00:0071:00> [M] Mudar: ptr[C0:E032:D0] inc[E0] num[1252] ptr[B4:D0F0:D0]
                       inc[82] t[D083:D0]
00:008B:00> [L] Lcall: num[1A12]
00:009B:00> [SA] Saltar: ptr[0A:1200:A9]
00:00A1:00> [T] Terminar/Retornar
00:00A4:00> [C] Compactar: ptr[C3:22C3:22] num[9000] ptr[01:C298:E5]
00:00B4:00> [EN] Enviar em intel hex.: ptr[02:00C2:A3] num[3098]
00:00EB:00> [F] Filtrar: ptr[75:3233:74] num[007B] entre [00] e [12]
                         para ptr[47:DEE5:83]
00:00F9:00> [O] Operar: ptr[03:0201:07] inc[B4]  mod ptr[03:0201:07] inc[02]
                        para ptr[01:8785:83] inc[30] num[8582] t[3274:01]
                                  ....
                                  etc.
                                  ....
JM> Listar:            «««« selecção de comando listar (caracter l ou L)
      Tipo [0 a 4] :  3         «««« memória de dados externa
      Endereco     :  0000      «««« endereço inicial
      Banco [P1]   :  1F        «««« banco de memória (RAM neste caso)
03:0000:1F> [C] Compactar: ptr[00:C402:53] num[82FF] ptr[FF:FFFF:FF]
03:000B:1F> [C] Compactar: ptr[53:EA25:82] num[0000] ptr[00:0000:00]
```

Figura B2. Exemplo da execução do comando **Listar** do interpretados (em letra deste tipo o que aparece no ecrã **e neste tipo os comentários**).

Como se verifica, neste comando são apresentadas as diversas instruções com a informação sobre os parâmetros que usam. Nos exemplos mostrados na figura B2, vê-se a listagem de duas zonas de memória: uma de código e outra de dados.

Os comandos de enviar e receber blocos de dados (ver figura B3) permitem-nos receber ou enviar a partir da estação remota um conjunto de dados de memória, usando um protocolo tradicional para transferência, o *intel hexadecimal*.

Como é de esperar, este método só tem interesse no caso de se utilizar a estrutura de comunicação mais básica, isto é, aquela em a rede é tida como uma consola estúpida; no caso de se ter implementado um protocolo mais complexo (estrutura 2 já anteriormente explicada), será aconselhável modificar este processo de comunicar; esta modificação não é contudo obrigatória, desde que exista uma ferramenta que faça o interface entre o funcionamento por pacotes da rede e o funcionamento completamente assíncrono actualmente implementado para a ligação do interpretador com o exterior.

```
JM> Enviar:          «««« selecção de comando enviar (caracter e ou E)

      Tipo [0 a 4] :  3        «««« memória de dados externa
      Endereco     :  0000     «««« endereço inicial
      Banco [P1]   :  1F       «««« banco de memória (RAM neste caso)
      Num. bytes   :  40       «««« nº de bytes a enviar
:100000000200C4025382FFFFFFFFF0253EA258272
:1000100000000000000000000000000000000000E0
:10002000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE0
:10003000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD0
:00000001FF

JM> Receber:         «««« selecção de comando receber (caracter r ou R)

      Tipo [0 a 4] :  3        «««« memória de dados externa
      Endereco     :  0000     «««« endereço inicial
      Banco [P1]   :  1F       «««« banco de memória (RAM neste caso)

:100000000200C4025382FFFFFFFFF0253EA258272:    «««« "frame" correcto
:0000000177#     «««« "frame" incorrecto, responde com #
:00000001FF:     «««« "frame" correcto, responde com :
```

Figura B3. Exemplos dos comandos de envio e recepção de blocos de memória

Para além destes processos de aceder à memória está ao dispor do utilizador um processo interactivo de introduzir o programa; este processo passa pela selecção de uma instrução de cada vez (figura B4).

Com vista a facilitar esta operação, no início da introdução aparece um pequeno menu com as instruções disponíveis bem como os caracteres que definem cada uma delas. À medida que as instruções são introduzidas aparece uma indicação com o endereço em que foi inserida cada uma delas.

Na introdução é permitido corrigir alguns dos caracteres que tenham sido mal digitados.

```
JM> Inserir:                «« selecção de comando inserir (caracter i ou I)

      Tipo [0 a 4] :  3       «« memória de dados externa
      Endereco     :  0000    «« endereço inicial
      Banco [P1]   :  1F      «« banco de memória (RAM neste caso)
Use um dos seguintes comandos e CTRL-Z, ESC ou ENTER para abandonar: «« menus com
 MU    C    EN   R     T    SA    SO  SE  EX    MA    L     F          «« instruções
Muda Comp. Env. Rec. Term. Salta Soma Se Exec. Marc. Lcall Filt.       «« permitidas
03:0000:1F> Enviar em intel hex.:ptr[03:0000:1F] num[15]
03:0007:1F> Lcall:num[1234]
03:000A:1F> Mudar:ptr[5:41:00] inc[00] num[20] ptr[3:8000:1F] inc[1] t[0000:00]
03:001A:1F> Enviar em intel hex.:ptr[00:0000:00] num[0000]
03:0021:1F> Terminar:                                          «« fim do programa
03:0022:1F> Marca fim (00,00)                                  «« fim do bloco a listar
03:0024:1F>
JM>
```

Figura B4. Exemplo da introdução de um programa com o recurso ao comando de inserir; neste exemplo o programa é escrito em memória externa.

Existem ainda dois comandos não exemplificados: execução de um programa e fim de operação com o interpretador (ver figura B5).

O primeiro deste permite executar um programa já residente em memória, enquanto o segundo faz transitar do módulo do interpretador para o operativo.

```
JM> Executar:               «« selecção de comando executar (caracter x ou X)

      Tipo [0 a 4] :  3       «« memória de dados externa
      Endereco     :  0000    «« endereço inicial
      Banco [P1]   :  1F      «« banco de memória (RAM neste caso)
:10000000030300001F00150A12340105004100001F
:05001000002003800048                          «« executa o programa introduzido
:00000001FF                                       na figura anterior

JM> Fim                     «« selecção do fim das operações (caracter f ou F),

JM..                        «« que obriga a voltar ao módulo operativo
```

Figura B5. Exemplo da utilização dos comandos de execução e de fim de operações.

Como é perceptível nestes exemplos e foi já explicado na Parte 3 do texto, considerou-se um espaço de endereçamento diferente do habitual; ao invés dos tradicionais 64 Kbytes existe o limite de 4 Mbytes. Este limite, embora na realidade seja inatingível, deve-se à consideração de que cada posição de memória é identificada por um "*record* " constituído por 3 *campos* : tipo de memória, endereço propriamente dito e o banco da memória.

Descrevem-se de seguida as instruções actualmente disponíveis ao utilizador, referindo quer as suas funções quer os parâmetros necessários à sua execução.

MARCAR: Tem por finalidade marcar o fim de uma zona de memória, com vista a limitar a duração da listagem, equivalendo a um "NOP" na execução de programas.

MUDAR: Esta instrução permite transferir um certo número de bytes (ou bits) de uma zona de memória ou periférico para outra zona de memória ou periférico com um certo intervalo de tempo definido em microsegundos, milisegundos ou segundos. Este tempo é medido entre a transferência de cada um dos bytes (ou bits).

Pode ainda definir-se o valor do incremento entre o endereço de cada unidade lida, bem como o endereço do destino. Por este processo é possível transferir blocos de memória para outro com incrementos entre cada posição do destino e da fonte independentes uns dos outros, recorrendo no entanto apenas a uma instrução.

De realçar que o valor fonte também pode ser um valor imediato e, neste caso, os incrementos a efectuar incidem directamente sobre esse valor imediato.

De cada vez que é lido um valor, é incrementado o ponteiro da fonte em X unidades, bem como é incrementado em Y unidades o ponteiro de destino. Estes ponteiros podem variar entre 0 e 256, e N (o número de valores máximo a mudar em cada instrução) pode tomar um valor entre 1 e 65536. O valor a guardar pode, em vez de ser lido de uma memória ou periférico, ser um valor imediato, ocupando o campo do endereço.

Exemplo: *Mudar:ptr[5:41:00] inc[00] num[20] ptr[3:8000:1F] inc[1] t[0000:00]*

(Escreve o valor 41 sem incrementar entre cada escrita na memória externa a partir da posição 8000 no banco 1F incrementando o endereço de 1 sem qualquer limitação de tempo; equivale a dizer: escreve o valor 41 na memória a partir da posição 8000 à 801F)

COMPACTAR: Transforma N bytes de uma zona de memória ou periférico guardando-os noutra zona de memória de forma compactada (valor seguido do número de bytes iguais).

Esta instrução permitindo minimizar a ocupação de memória, garante em última análise a diminuição dos tempos gastos na transmissão de grandes quantidades de informação com valores repetitivos.

Exemplo: *Compactar:ptr[3:0000:1F] num[20] ptr[3:8000:1F]*

(Compacta a informação da memória externa banco 1F a partir do endereço 0 até ao endereço 1F, colocando o resultado na mesma memória a partir de 8000)

ENVIAR: Envia um conjunto de dados através da linha de comunicação série no formato intel hex. Nem o número de valores, nem o endereço inicial, nem o local onde está a informação sofrem qualquer restrição particular.

Exemplo: *Enviar em intel hex.: :ptr[3:8000:1F] num[30]*

(Envia em formato intel hexadecimal 48 bytes a partir do endereço 8000 da memória de dados externa)

RECEBER: Recebe para memória ou periférico um conjunto de dados transmitido através da linha de comunicação no formato intel hexadecimal, tal como na instrução anterior.

Exemplo: *Receber em intel hex.: :ptr[3:8000:1F]*

(Recebe informação em formato intel hexadecimal para a memória de dados externa a partir da posição 8000)

SALTAR: Passa a execução do programa para outra zona de memória.

Exemplo: *Saltar: :ptr[3:8000:1F]*

(O programa passa a execução para o endereço indicado)

EXECUTAR: Executa um programa escrito nesta linguagem (interpretada) e localizado na zona de memória indicada, após o que regressa ao ponto de chamada.

O programa a chamar deve finalizar com a instrução TERMINAR, sob pena de não regressar ao passo pretendido.

Exemplo:*Executar :ptr[3:8000:1F]*

(Executa a sub-rotina partir do endereço indicado)

TERMINAR: Termina a execução de um conjunto de instruções, retornando ao ponto de onde tinha sido chamado. Esta instrução funciona como "RETURN" de uma sub-rotina; permite também terminar a execução de um programa se este tiver sido chamado directamente pelo operador, usando o comando executar.

LCALL: Esta instrução é similar à EXECUTAR, distinguindo-se por chamar uma sub-rotina escrita em linguagem máquina.

A sub-rotina, pelas mesmas razões das explicitadas para a instrução anterior, deve terminar com a instrução que permita o retorno, isto é ou a instrução RET, ou a instrução RETI.

Exemplo:*Lcall:num[8000]*

(Executa a sub-rotina partir do endereço **de código** indicado)

SOMAR: Soma uma palavra de 16 bits a 2 posições de memória consecutivas.

Exemplo:*Somar: ptr[3:8000:1F] num[1010]*

(Soma o valor 1010 ao conteúdo de 2 posição de memória a partir de 8000, com o MSB na posição inicial)

SE: Executa uma operação lógica entre 2 valores (bit ou byte), guardados em memória (ou periférico) ou valores imediatos e, caso o resultado seja verdadeiro, a execução passa para outra zona de memória.

Exemplo: *Se: ptr[3:8000:1F] rel[01] ptr[3:1010:1F] ptr[3:2000:1F]* equivalente a *Se: ptr[3:8000:1F] OR ptr[3:1010:1F] ptr[3:2000:1F]*

(Se o ou lógico das posições 8000 e 1010 der resultado verdadeiro então a execução continua na posição 2000.)

FILTRAR: Filtra um número N de valores guardados em determinada zona de memória. Todos os valores que estiverem ou entre 2 limites ou fora desses limites são modificados.

Exemplo: *Filtrar: ptr[3:8000:1F] num[20] entre[10] e [29] para ptr[3:1010:1F]*

(Coloca o conteúdo da posição 1010 da memória de dados externa nas posições entre 8000 e 801F, se os conteúdos destas estiverem dentro do limite [10,20])

OPERAR: Executa uma função lógica ou aritmética entre 2 valores. Como em casos de instruções já explicitadas estes valores tanto podem ser constantes (valores imediatos ou valores guardados na memória de código) como variáveis residentes em qualquer zona de memória ou periférico.

Exemplo: *Operar: ptr[3:8000:1F] inc[1] + ptr[3:1000:1F] inc[2]*
para ptr[3:2000:1F] num[10] t[01:02]

(Efectua a soma dos valores contidos em memória externa de dados a partir da posição 8000 e 1000 até à posição 800F e 101E, demorando 1 segundo entre cada operação; dado os valores diferentes dos incrementos, os ponteiros para a zona de memória que contém os operandos varia diferentemente [8000]+[1000], [8001]+[1002],..., [800F]+[801E])

Para a construção do interpretador optou-se pela utilização da linguagem PL/M-51.

Sendo por natureza uma linguagem estruturada, torna-se bastante facilitada a tarefa de garantir um programa coerente e estruturado, em que o desenvolvimento seja efectuado seguindo a metodologia "top-down"; para o efeito é de muita utilidade a possibilidade de utilizar as funções e procedimentos.

```
                                        .....
                                        .....
inicio:
                    do;
                        erro=0;
                        tr0=0;
                        en=0;
;
                        fim=0;
                        do while fim=0;
                          erro=0;
                          call envia(.('J M> ',0));
                          registo0=ci_vdu_tecla do;
                          if (registo0>='a')a nd(registo0<='z') then registo0=registo0 and 5fh;
                          if registo0='L' then d o;
                                call co(registo0);
                                call listar_cli;
                                call crlf;
                                fim=0;
                          end;
                          else d o;
                                if registo0='X' then do;
                                      call co('E');
                                      call executa r_cli;
                                      call crlf;
                                      fim=0;
                                end;
                                else d o;
                                      if registo0='I' then d o;
                                            call co(registo0);
                                            call inserir_cli;
                                            call crlf;
                                            fim=0;
                                      end;
                                      else d o;
                                            if registo0='R' then do;
                                                  call co(registo0);
                                                  call receber_cli;
                                                  call crlf;
                                                  fim=0;
                                            end;
                                            else d o;
                                                  if registo0='E' then d o;
                                                        call co('E');
                                                        call transmitir_cli;
                                                        call crlf;
                                                        fim=0;
                                                  end;
                                      .....
                                      .....
```

Figura B6. Extracto do módulo do programa principal do interpretador.

As figuras B6, B7 e B8 pretendem mostrar a metodologia utilizada na implementação deste programa; na primeira destas figuras apresenta-se o módulo principal, em que cada um dos comandos executáveis é iniciado a partir da "CLI" (consola portátil ou rede funcionando como terminal virtual). A modificação desta situação, com vista a atender às especificações da **Estrutura 2**, obrigaria apenas a modificar este módulo principal. De notar, ainda, que alguns dos procedimentos invocados podem ser efectuados por chamada de um programa.

```
executa r:proced ure(base_p onteiro);
    decla re ba se_ponteiro  word;

    fim_0=0;
    flag_0=0;
    do while fim_0 < 0ffh;
      valor= fetch(base_p onteiro); /* Usa: temp_b_1,2  */
      if va lor=0  then d o;
            call inc_end(ba se_ponteiro);
            if fla g_0= 0 then  flag_0=1;
                              else fim_0= 0ffh;
      end;
      else d o;
            flag_0=0;
            if(va lor>0)a nd(va lor<13)  then d o;
                  do case  valor;
                        ;       /* 0 */
                        do;       /* 1 = muda r */
                              call inc_end(ba se_ponteiro);
                              call exec_mud ar(ba se_ponteiro);
                        end;
                        do;       /* 2 = compa ctar */
                              call inc_end(ba se_ponteiro);
                              call exec_comp acta r(ba se_p onteiro);
                        end;
                        do;       /* 3 = enviar */
                              call inc_end(ba se_ponteiro);
                              call exec_tra nsmitir(base_p onteiro);
                        end;
            .....
            .....
                        do;       /* 9 = executa r */
                              call inc_end(ba se_ponteiro);
                              sp= sp+1;
                              stack _gera l=fim_0;
                              sp= sp+1;
                              stack _gera l=fla g_0;
                              sp= sp+1;
                              stack _gera l=reg isto0;
                              sp= sp+1;
                              stack _gera l=reg isto1;
                              sp= sp+1;
                              stack _gera l=reg isto2;
                              sp= sp+1;
                              stack _gera l=reg isto3;
                              call exec_ex ec(base_p onteiro,.executa r);
                              registo3= stack _gera l;
                              sp= sp-1;
                              registo2= stack _gera l;
                              sp= sp-1;
                              registo1= stack _gera l;
                              sp= sp-1;
                              registo0= stack _gera l;
                              sp= sp-1;
                              flag_0=sta ck_g eral;
                              sp= sp-1;
                              fim_0=sta ck_g eral;
                              sp= sp-1;
                              registo12= registo12+4;
                              base_p onteiro=.reg isto0;
                        end;
                        do;         /*10 Lca ll */
                              call inc_end(ba se_ponteiro);
                              call exec_ca ll(base_p onteiro);
                        end;
            .....
            .....
      end;
    end;
end executa r;
```

Figura B7. Implementação da instrução **Executar**.

A utilização de uma metodologia modular permite facilmente criar novas instruções, actualizando o interpretador (figura B7).

Nessa figura, pode-se verificar o método utilizado para a execução de sub-rotinas colocadas em memória a partir de outra posição; é necessário guardar certos valores em "stack" para poder posteriormente retomar a execução no mesmo ponto e com as mesmas garantias de funcionamento.

```
exec_mudar: procedure(base_ponteiro);
   declare base_ponteiro word;

   call compoe_ponteiro(.ponteiro1,base_ponteiro);
   call compoe_incremento(.inc1,base_ponteiro);
   call compoe_n(.n,base_ponteiro);
   call compoe_ponteiro(.ponteiro2,base_ponteiro);
   call compoe_incremento(.inc2,base_ponteiro);
   call compoe_t(.t,base_ponteiro);
   voltas_relogio,relogio= 0;
   if (t.tempo>100)or((t.tempo<100)and(t.unidades<> 0)) then do;
     if t.unidades=0 then do;
          relogio= t.tempo/100;      /* micro segundos */
          voltas_relogio=relogio/256;
          relogio= relogio/voltas_relogio;
     end;
     else do;
          if t.unidades=1 then do;
               if t.tempo< 256 then do;
                    relogio= t.tempo;          /* milisegundos */
                    voltas_relogio=10;
               end;
             else do;
                    if t.tempo< 4096 then do;
                         relogio= t.tempo/16;      /* milisegundos */
                         voltas_relogio=160;
                    end;
                    else do;
                         relogio= t.tempo/256;     /* mil segundos */
                         voltas_relogio=2560;
                    end;
               end;
          end;
          else do;
               relogio= 250;               /* segundos */
               voltas_relogio=t.tempo*40;
          end;
     end;
     temp_w_1= relogio*236/256;
     relogio= temp_w_1 and 255;
   end;
   do while n> 0;
     call guarda_byte(.ponteiro2,fetch(.ponteiro1));
     call soma_byte_ponteiro(.ponteiro1,inc1);
     call soma_byte_ponteiro(.ponteiro2,inc2);
     n=n-1;
     if relogio<> 0 then call espera;
   end;
   tr0=0;              /*para timer 0*/
end exec_mudar;
```

Figura B8. Exemplo de um procedimento chamado para a execução da instrução MUDAR.

Na figura B8 apresenta-se um procedimento que efectua a execução de uma instrução MUDAR; depois da parte inicial respeitante à composição dos parâmetros necessários, segue-se o corpo principal em que são guardados os valores nas posições correctas.

Para a implementação da estrutura de ponteiros (entenda-se estrutura como equivalente ao "record" tradicional do PASCAL) usaram-se as facilidades da própria linguagem, que permite a definição desse tipo de dado:

```
declare ponteiro3 structure (tipo byte,endereco word,banco byte) at (0c3h) idata;
declare t structure(tempo word,unidades byte) at (0c7h) idata;
```

Surgiu no entanto a dificuldade de aceder à memória interna quando o objectivo é conhecer ou actualizar o valor de um registo especial ("Special Function Register")

que sendo acedidos de modo directo não é possível de controlar em PL/M a não ser implementando um procedimento do tipo do indicado de seguida:

```
fetch_s: procedure(apont_e) byte;
   declare apont_e word;
   if apont_e=88h then temp_b_4=tcon;
   if apont_e=89h then temp_b_4=tmod;
   if apont_e=8ah then temp_b_4=tl0;
   if apont_e=8bh then temp_b_4=tl1;
   if apont_e=8ch then temp_b_4=th0;
   if apont_e=8dh then temp_b_4=th1;
   if apont_e=90h then temp_b_4=p1;
   if apont_e=98h then temp_b_4=scon;
   if apont_e=99h then temp_b_4=sbuf;
   if apont_e=0a8h then temp_b_4=ie;
   if apont_e=0b0h then temp_b_4=p3;
   if apont_e=0b8h then temp_b_4=ip;
   return temp_b_4;
end fetch_s;
```

Esta metodologia, que também foi adoptada quando se pretende actualizar algum destes valores (neste exemplo estamos a efectuar a leitura de um valor), não foi necessária para aceder aos restantes tipos de ponteiros.

Uma das grandes vantagens na utilização desta linguagem prende-se também com a facilidade de aceder de uma forma bastante simplificada a zonas de memória e variáveis, usando apontadores para as variáveis, como se verifica no exemplo seguinte:

```
exec_call: procedure(base_ponteiro);        «««« Implementação do comando Lcall
   declare base_ponteiro word;

   call compoe_n(.n,base_ponteiro);
   call n;                                  «««« execução da chamada
end exec_call;
```

Este tipo de acesso, conjugado coma a utilização de procedimentos e funções, simplifica consideravelmente o desenvolvimento e modificação posterior da programação.