Gestão do consumo em microprocessadores:
Aumento da autonomia sem comprometer a funcionalidade

Gonçalo Rijo
Instituto Superior Técnico – Universidade Técnica de Lisboa
DEEC, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
e-mail: goncalorijo@gmail.com

Carlos Almeida
Instituto Superior Técnico – Universidade Técnica de Lisboa
DEEC, AC-Computadores, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
Telefone: +351-21-8418397 FAX: +351-21-8417499 e-mail: cra@comp.ist.utl.pt

José Rufino
Faculdade de Ciências da Universidade de Lisboa
Campo Grande, Bloco C5, 1700 Lisboa, Portugal
Telefone: +351-21-7500254 FAX: +351-21-7500084 e-mail: ruf@di.fc.ul.pt

A redução do consumo energético em sistemas embebidos é hoje em dia um ponto fulcral no projecto e desenvolvimento dos mesmos. Neste artigo é feita uma comparação entre várias estratégias de poupança de energia tendo em conta algumas variáveis globais de um sistema. É feita também uma análise da relação consumo/desempenho de um sistema. Os resultados demonstram que se conseguem melhorias significativas em sistemas com este tipo de mecanismos implementados.

Palavras-chave: Consumo energético em sistemas embebidos, Aumento da autonomia, Controlo de recursos

1. Introdução

NOS dias que correm praticamente em todos os locais existem sistemas embebidos, desde o mais simples sistema de detecção de incêndio autónomo até ao complexo telemóvel, estamos rodeados por micro-sistemas com capacidades relativamente limitadas comparativamente com sistemas mais gerais. Todos estes sistemas, devido às suas pequenas dimensões, não têm capacidades elevadas de armazenamento de energia eléctrica, o que pode mesmo constituir um ponto crítico no seu funcionamento e limitar o seu desempenho.

Existem várias formas de reduzir o consumo energético em sistemas embebidos, dependendo das características específicas do equipamento e sistema operativo utilizados. Pode-se, por exemplo, baixar a frequência de funcionamento do processador, associada também a uma redução da tensão de alimentação, ou entrar em estados de consumo reduzido em que nada é processado até que um evento exterior coloque o sistema novamente activo.

Outras alternativas podem passar por desligar temporariamente unidades que não estejam a ser utilizadas durante um determinado intervalo de tempo.

A integração no sistema destes vários mecanismos de controlo do consumo energético, também poderá ser feito de uma forma mais ou menos profunda, implicando maiores ou menores alterações ao sistema existente.

No trabalho aqui apresentado optou-se por uma integração mais superficial, o que, embora apresente uma menor precisão, permite uma maior flexibilidade na sua utilização.

This work was partially supported by FCT through Project POSC/EIA/56041/2004 (DARIO).

2. Diminuição da frequência

Actualmente os sistemas embebidos contêm sistemas autónomos de processamento construídos praticamente só com transístores CMOS. A energia dissipada por esta tecnologia tem duas componentes:

$$P_{dissip} = P_{cond} + P_{comut}, \quad (1)$$

em que P_{cond} é a potência de perdas dissipada por condução e P_{comut} é a potência dissipada durante a transição de estado lógico.

1) Perdas por condução: Teoricamente a tensão num regime estacionário de condução (V_{cond}) num transístor ideal seria nula. Na realidade tal não se passa uma vez que temos uma resistência interna associada a cada transístor, logo, existe uma queda de tensão quando o transístor funciona na zona de condução.

Como a potência associada a esta queda de tensão é dissipada pela forma de calor a energia envolvida no processo é considerada perdida uma vez que não é utilizada de forma eficiente.

A expressão aproximada das perdas de condução é dada por:

$$P_{cond} = R_{\text{int}} \times I_{cond}^{\ 2}, \quad (2)$$

em que R_{int} é a resistência interna do transístor e I_{cond} é a corrente de condução do mesmo. Admite-se, uma vez que é uma boa aproximação, que a resistência interna é constante e que as correntes de fuga são desprezáveis.

2) Perdas por comutação: As transições entre estados são idealmente instantâneas mas, mais uma vez, na realidade o funcionamento não é ideal. A razão para tal deve-se, maioritariamente, à existência de capacidades parasitas entre os terminais que têm que ser carregadas entre as diversas áreas do transístor quando se dá uma mudança de estado. Esses tempos de carregamento e descarregamento, como não são instantâneos, vão dar origem a perdas sempre que se

efectua uma comutação.

A expressão aproximada das perdas de comutação é dada por:

$$P_{comut}(t) = v(t) \times i(t), \quad (3)$$

em que $v(t)$ e $i(t)$ são a tensão e corrente instantâneas respectivamente.

Em seguida é apresentado um gráfico de forma a clarificar estas últimas perdas:

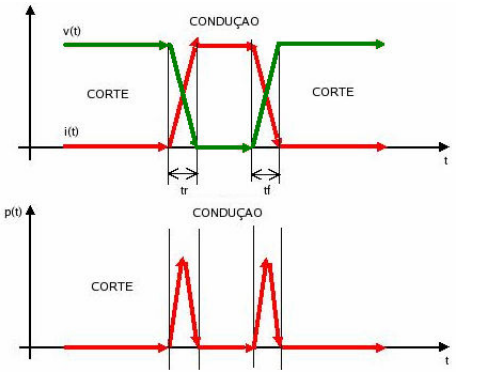

Figura 1 – Perdas por comutação

Actualmente a tecnologia de criação e desenvolvimento de processadores encontra-se focada no aumento da frequência dos processadores de forma a obterem uma maior capacidade de processamento. Existem outras aproximações que apostam mais no sentido de processadores com *pipelines* paralelos, aumentando a capacidade de processamento por ciclo de relógio mas são menos frequentes.

Uma vez que os processadores funcionam em frequências elevadas (na ordem dos MHz) as perdas por condução são praticamente desprezáveis, isto quando comparadas com as perdas originadas por comutação, logo, uma diminuição da frequência de funcionamento do processador, terá um efeito redutor mais significativo no consumo por unidade de tempo.

De salientar que a eficiência energética aumenta, isto porque a uma diminuição da frequência podemos ter associado também uma diminuição da tensão, o que provoca uma diminuição da potência de perdas por comutação. Assim, para a mesma quantidade de energia gasta, aumenta-se o número das instruções processadas.

3. Desempenho

A uma diminuição da frequência de funcionamento está associada uma diminuição do desempenho do processador uma vez que se processa menos instruções por segundo devido ao menor número de ciclos de relógio.

De forma a quantificar as diminuições de desempenho associadas a uma diminuição de frequência, foram feitos ensaios em que se varia a frequência de um processador tendo-se medido o consumo e o número de instruções por segundo, BogoMips[1], que o processador, a uma determinada frequência, consegue executar. Este último valor, apesar de não ser um mecanismo real de avaliação do desempenho do processador, fornece uma estimativa.

1) Consumo e Frequência:

Os resultados apresentados na Figura 2 são para uma taxa de ocupação do processador de 100%, isto é, são dadas tantas instruções ao processador quantas as que ele consegue processar de forma a maximizar o número de comutações de estado nos transístores internos.

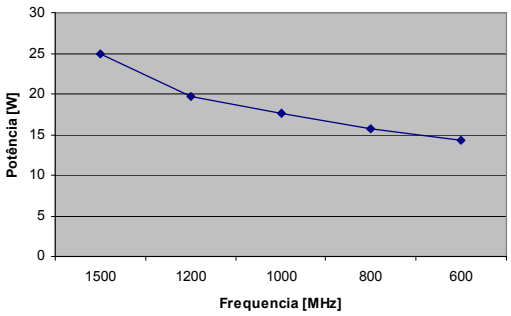

Figura 2 – Consumo e Frequência

Como se pode verificar o consumo diminui com uma redução da frequência de funcionamento do processador.

2) Desempenho e Frequência:

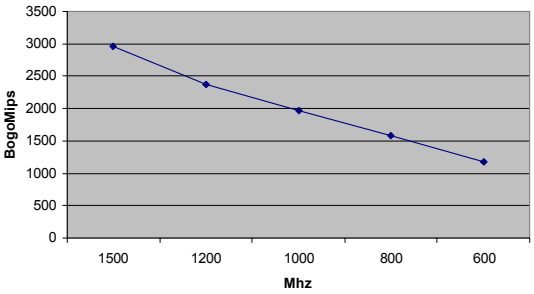

Figura 3 – Desempenho e Frequência

Como se pode verificar pelo gráfico apresentado na Figura 3, e tal como esperado, a capacidade de processamento é menor quando a frequência de funcionamento diminui.

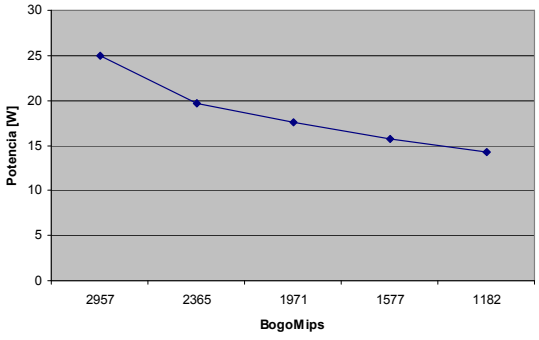

Figura 4 – Desempenho e Consumo

3) Desempenho e Consumo:

A Figura 4 representa o número de instruções por segundo para um dado consumo, isto é, o número de instruções que conseguimos processar para um valor fixo de energia. Este valor é importante uma vez que oferece informação útil para aplicações que necessitam de executar periodicamente um determinado número de instruções por segundo de maneira a

[1] BogoMips – Unidade para o número de instruções por segundo num sistema Linux

que estas tenham conhecimento da sua frequência mínima de funcionamento.

4. Mecanismos de poupança de energia

Em seguida iremos apresentar dois mecanismos de poupança de energia, ambos com diferentes tipos de concretizações.

1) Controlador automático: Uma primeira aproximação a um mecanismo de poupança de energia poderá ser um processo que se execute em paralelo no sistema e através da análise global desse mesmo sistema decida qual o melhor estado de funcionamento.

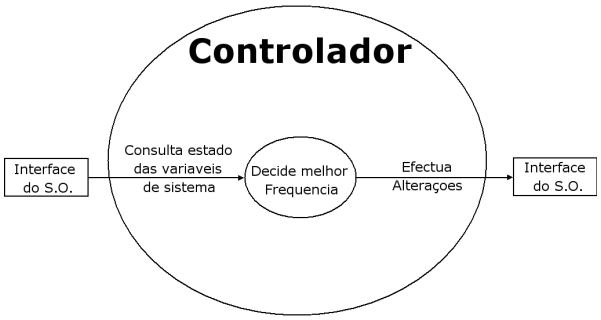

Figura 5 – Diagrama de funcionamento do controlador automático

A análise do sistema consiste simplesmente numa leitura das variáveis mais relevantes, tais como, o estado da bateria e a taxa de ocupação do processador. O processo de decisão e actuação não é tão simples, uma vez que oferece liberdade para vários tipos de implementação. Em seguida serão apresentadas duas formas diferentes de decisão:

a) Linear com a carga: A frequência de funcionamento do sistema é decidida com base na taxa de ocupação do processador, carga. Após o conhecimento das frequências disponíveis é feita uma linearização e definida uma correspondência entre um determinado valor da carga e uma frequência. Assim, quando temos uma carga elevada no sistema teremos uma frequência elevada e quando a carga é diminuta temos uma frequência reduzida.

b) Escalável / Mínima frequência: Neste caso a frequência também é decidida com base na taxa de ocupação do processador, mas em vez de analisar a carga actual e instantânea, analisa-se a carga prevista tendo em conta todas as aplicações a correr no sistema. Se esta nova carga subir acima do valor máximo previamente definido então aumenta-se a frequência para o patamar seguinte criando assim mais recursos para as aplicações. Quando a carga imposta pelas aplicações diminui abaixo do valor máximo então também se diminui a frequência para o patamar inferior.

2) Controlador interactivo: Outra aproximação a este problema consiste num controlador que também corre em paralelo com as aplicações mas tem mecanismos de interacção com as mesmas. Assim tem informação directa das aplicações e dos recursos por elas consumidos. Comparativamente aos sistemas anteriores, pode optimizar o sistema de decisão da frequência uma vez que poderá ter acesso a eventuais metas das aplicações, entre outros aspectos.

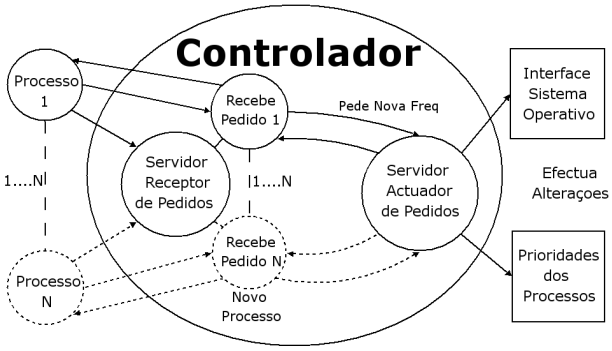

Figura 6 – Diagrama de funcionamento do controlador com interacção com as aplicações

Em seguida são apresentadas duas formas possíveis de interacção com este controlador e feita uma comparação entre ambas.

a) Aplicações com conhecimento do sistema: As aplicações fazem inicialmente um estudo das capacidades do sistema e definem os recursos que ocuparam para o bom funcionamento das mesmas. Em seguida enviam a informação para o controlador através de um mecanismo de comunicação tal como *sockets*. Com esta informação o controlador faz um tratamento e escolhe a melhor frequência, garantindo recursos disponíveis para todas as aplicações que enviaram dados ao controlador.

b) Satisfação por parte do utilizador / aplicações: Nesta implementação as aplicações não necessitam de um conhecimento prévio da máquina onde irão ser executadas, mas têm definidas variáveis que indicam se a aplicação se encontra a funcionar correctamente ou se os recursos disponíveis não são satisfatórios. Um exemplo prático será o número de imagens por segundo num jogo, em que se este valor está abaixo de um número previamente definido pelo programador então a aplicação informa o controlador que esta não se encontra satisfeita com os recursos disponíveis. Se este valor estiver dentro dos parâmetros normais de funcionamento então a informação enviada é o código correspondente ao nível de satisfação aceitável. Caso os recursos disponíveis para a aplicação superem as necessidades da mesma, então é enviada a informação de que uma diminuição dos recursos disponíveis não afectará o desempenho aceitável da aplicação.

Em certas aplicações previamente construídas é difícil a alteração do código e a nova compilação para respeitar este novo sistema de controlo energético. Para facilitar a inserção deste mecanismo nos sistemas actuais sugere-se uma aplicação que funciona em paralelo com as aplicações do sistema. Esta nova aplicação implementa a comunicação com o sistema controlador e interage com o utilizador perguntando o seu nível de satisfação para a aplicação em execução.

5. Resultados Práticos

De forma a quantificar os benefícios que a implementação dos mecanismos de poupança de energia propostos possam trazer procedeu-se à concretização dos mesmos tendo por base de desenvolvimento um sistema que oferecesse os

mecanismos de interacção necessários.

Após um estudo das diversas plataformas de desenvolvimento optou-se por um portátil com um processador que permitisse a variação da frequência. Quanto ao software usou-se um sistema operativo Linux devido as facilidades de interacção com o hardware e de inserção de novos mecanismos no sistema.

Para além dos mecanismos de poupança de energia foram ainda desenvolvidos outros sistemas de simulação. Começou-se por identificar classes típicas de aplicações e para cada classe criou-se uma aplicação tipo com comportamento semelhante. Em seguida serão enumeradas as diversas classes com o comportamento típico de cada uma:

Classe 1) *Processamento de texto / Folhas de cálculo / Programação de aplicações simples*

- São normalmente aplicações de interacção com o utilizador em que na maioria do tempo não se encontram em processamento.
- Quando existe algo a processar esse processamento é simples e não necessita de muitos recursos disponíveis.

Classe 2) *Processamento de sinais / Programação de aplicações que requerem muitos recursos quando em teste*

- Aplicações com recursos variáveis e de grandes oscilações.
- Normalmente estão em repouso mas quando existe a necessidade de processar algo há o requisito de fazê-lo de forma rápida.
- Nos períodos de processamento requerem recursos elevados.

Classe 3) *Jogos / Processamento contínuo*

- Aplicações que consomem todos os recursos disponíveis independentemente das capacidades de processamento totais.
- Para funcionarem "bem" podem não ter necessidade de consumir todos os recursos mas devido aos mecanismos de programação usados no desenvolvimento não o fazem.

Para as últimas duas classes foram criadas três aplicações tipo: as aplicações que têm requisitos baixos; as que requerem recursos médios; e as que, para ficarem satisfeitas, requerem mais recursos do que aqueles que o sistema tem capacidade de oferecer. Denominaram-se estas aplicações por *min*, *med* e *max* respectivamente. A primeira classe não será usada nas simulações uma vez que as aplicações desta classe não têm requisitos especiais.

De referir que as aplicações de simulação para a ***classe 2*** subdivide-se em outras duas, as que têm uma característica de consumo periódica e as que são perfeitamente aleatórias.

Define-se ainda que uma aplicação encontra-se satisfeita quando consegue cumprir os requisitos temporais. Para simulação, a diferença entre uma aplicação com requisitos reduzidos e outra com requisitos médios são as metas temporais, na primeira o tempo disponível para processamento é superior à segunda.

Para medir os resultados criou-se um programa específico para esse efeito. O funcionamento consiste no cálculo do tempo que demora a consumir uma unidade de energia. Neste caso a unidade de energia ficou definida como sendo 22mAh uma vez que a interface do sistema operativo para a bateria só era actualizada após um consumo desta ordem.

Por fim, definiu-se uma nomenclatura para as figuras que indica o número e tipo de aplicações em execução:

```
a min b med c max
```

em que a, b e c indica o número de aplicações em execução no teste com requisitos mínimos, médios e máximos respectivamente.

Sem Controlador

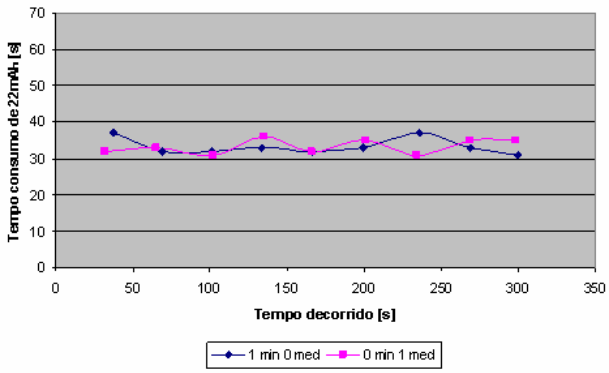

Figura 7 – Tempo do consumo de 22mA num cenário **sem** controlador

	1 min 0 med	**0 min 1 med**
Tempo médio de consumo [s]	*33,3*	*33,1*

Tabela 1 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário **sem** controlador

A Figura 7 apresenta duas situações típicas de funcionamento de uma aplicação. Esta não necessita de tantos recursos disponíveis mas usa-os, logo faz com que o consumo seja bem mais elevado do que o mínimo indispensável.

1) Resultados com Sistema Automático

Este sistema é muito útil quando não existe maneira de obter informação sobre as aplicações, quer por dificuldades de implementação da aplicação, quer quando não é possível executar o sistema paralelo que interage com o utilizador questionando-o sobre o seu nível de contentamento. Neste teste tentou-se simular uma situação o mais próxima possível da realidade. Os resultados foram retirados de um cenário com várias aplicações de classe 2 a funcionarem em simultâneo.

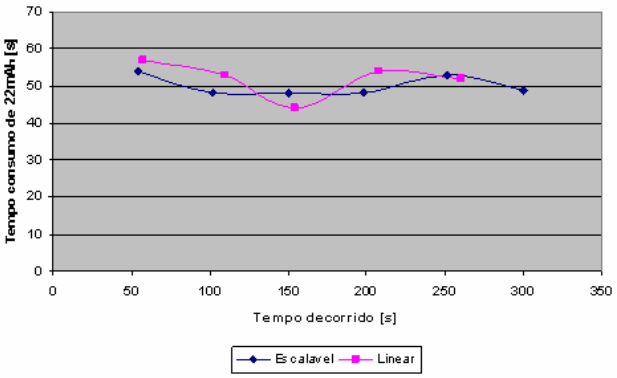

Figura 8 – Controladores automáticos e aplicações de **classe 2**

	Escalável	Linear
Tempo médio de consumo [s]	*51.2*	*50.3*

Tabela 1 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário com controlador automático escalável e linear

Como se pode verificar pela Figura 8, ambos os controladores, escalável e linear, apresentam melhorias na eficiência energética face ao resultado apresentado na Figura 7. O critério de selecção do tipo de controlador automático a usar poderá estar relacionado com a satisfação das aplicações. Para os resultados apresentados as aplicações tiveram uma taxa de satisfação superior no controlador linear. Isto acontece devido a este ser mais rápido a agir mas isso pode vir a influenciar o consumo negativamente, dependendo do estado geral de funcionamento.

2) Resultados com Servidor

Em seguida serão apresentados os ganhos efectivos com o uso de um sistema que receba informação das aplicações em execução. Os servidores criados subdividem-se em dois tipos, o primeiro que simplesmente actua ao nível da frequência e um segundo, que para além de variar a frequência de funcionamento ainda altera as prioridades dos processos dando maiores quotas de processamento aos processos com mais requisitos.

Uma vez que as combinações entre tipos de aplicações são imensas optou-se por restringir os resultados aqui apresentados a três cenários. O primeiro, em que temos três aplicações com recursos reduzidos, o segundo com uma aplicação com recursos medianos e por fim uma aplicação com requisitos reduzidos em simultâneo com uma com recursos médios. Os resultados são apresentados para aplicações de classes 3 e 2.

*Controlador **sem** prioridades e aplicações de **classe 3***

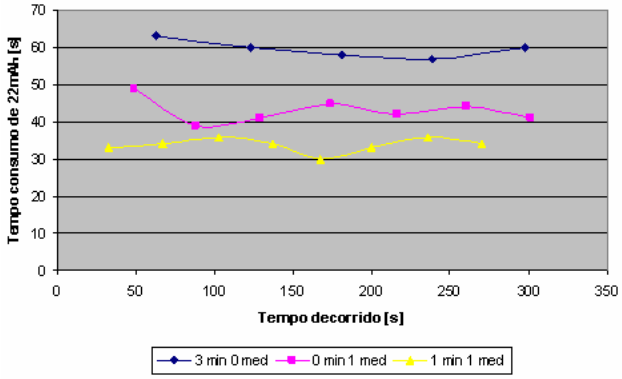

Figura 9 – Controlador **sem** prioridades e aplicações de **classe 3**

	3 min 0 med	0 min 1 med	1 min 1 med
Tempo médio de consumo [s]	*59,6*	*43,0*	*33,7*

Tabela 2 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário com servidor **sem** prioridades

No primeiro cenário, em que existem três aplicações a funcionarem em simultâneo, todas com requisitos semelhantes e reduzidos, verifica-se uma diminuição muito significativa no consumo total. Comparativamente com o cenário em que não existe nenhum controlador (apresentado na Figura 7) na situação em que temos somente uma aplicação em execução os ganhos são evidentes. Obtém-se uma poupança de energia na ordem dos 50%, isto mesmo apesar de estarmos a fazer uma comparação de certa forma desigual, entre um cenário só com uma aplicação e outro com três em simultâneo. Após estabilizadas, todas as três aplicações ficam satisfeitas com o seu desempenho.

No segundo cenário, em que temos somente uma aplicação com requisitos médios, também se verifica uma poupança de energia significativa, na ordem dos 30%. A aplicação também ficou satisfeita com o seu desempenho.

No último cenário apresentado, com uma aplicação *min* e outra *med* em simultâneo, a aplicação com recursos médios não foi satisfeita uma vez que não existe implementado neste servidor um mecanismo de atribuição de quotas de processamento às aplicações (servidor sem prioridades), logo, se temos uma aplicação a pedir mais recursos a solução é aumentarmos o nível de processamento. Desta forma, todas as aplicações em funcionamento ficam com mais tempo de processamento associado mas esse incremento de recursos é distribuído de forma igual por todas, isto porque o sistema de escalonamento é round-robin[2] para aplicações com a mesma prioridade.

*Servidor **com** prioridades e aplicações de **classe 3***

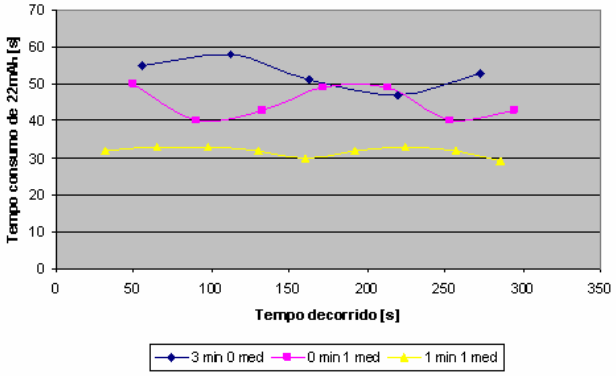

Figura 10 – Servidor **com** prioridades e aplicações de **classe 3**

	3 min 0 med	0 min 1 med	1 min 1 med
Tempo médio de consumo [s]	*52.5*	*44.8*	*31.7*

Tabela 3 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário com servidor **com** prioridades

No caso do servidor com prioridades, os resultados apresentados, relativamente ao consumo, são semelhantes ao cenário anterior, uma vez que o sistema de decisão da frequência é idêntico. As pequenas oscilações de consumo deste teste para o anterior estão associadas à necessidade de um processamento extra para o cálculo das novas prioridades. A principal diferença deste sistema está ao nível de satisfação das aplicações.

No controlador anterior sem prioridades, no cenário em que existiam duas aplicações em execução simultânea, não tínhamos satisfação das duas aplicações, somente uma, a com menores requisitos é que era satisfeita. Neste caso, com um servidor que controla as prioridades das aplicações, podemos definir quotas de processamento relativas entre aplicações com um aumento da prioridade de uma aplicação face a

[2] Round-Robin – Escalonamento que atribui a cada processo um tempo de processamento igual

outra. Assim podemos retirar tempo de processamento a uma aplicação que tenha mais recursos ao seu dispor do que aqueles que necessita na realidade e atribui-los a outras aplicações que os têm em falta. Logo, com este mecanismo, ambas as aplicações neste terceiro cenário estão satisfeitas.

*Servidor **sem** prioridades e aplicações de **classe 2***

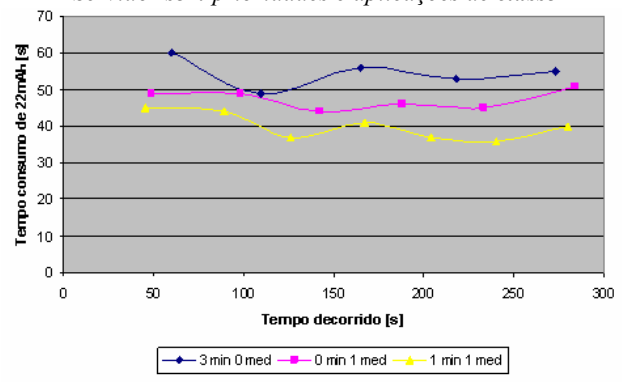

Figura 11 – Servidor **sem** prioridades e aplicações de **classe 2**

	3 min 0 med	0 min 1 med	1 min 1 med
Tempo médio de consumo [s]	*54.6*	*47.3*	*40,0*

Tabela 4 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário com servidor **sem** prioridades

Neste caso, em que temos um servidor com controlo de frequência sem o sistema de decisão de prioridades, ganhos são da mesma ordem de grandeza quando comparados com o cenário onde existem aplicações de classe 3 em funcionamento. De referir que todas as aplicações ficaram satisfeitas.

*Servidor **com** prioridades e aplicações de **classe 2***

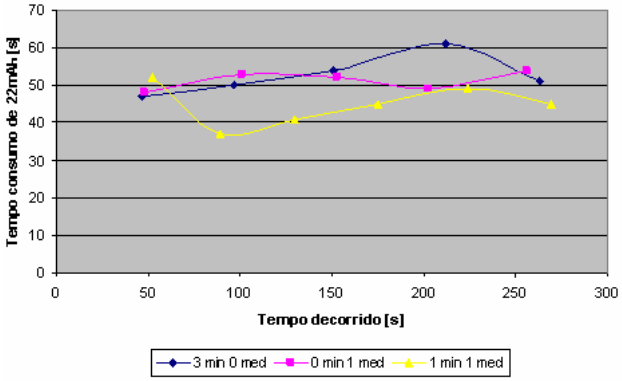

Figura 12 – Servidor **com** prioridades e aplicações de **classe 2**

	3 min 0 med	0 min 1 med	1 min 1 med
Tempo médio de consumo [s]	*52.6*	*51.2*	*44,8*

Tabela 5 – Tempo médio, em segundos, associado ao consumo de 22mA num cenário com servidor **com** prioridades

Neste último cenário também são significativos os ganhos mas quando comparados com o sistema sem controlo de prioridades são relativamente inferiores. Tal acontece devido a um carácter aleatório das aplicações que pode ter sido mais penalizante neste caso e, novamente, à existência de um consumo extra no cálculo de novas prioridades para as aplicações.

De referir, no entanto, que ao contrario do anterior, este sistema tem uma capacidade de satisfação das aplicações muito superior uma vez que lhes atribui diferentes quotas de processamento.

6. Conclusões

Os vários mecanismos de poupança de energia aqui apresentados permitem obter ganhos significativos na redução do consumo quando comparados com sistemas que não implementam qualquer mecanismo de poupança, podendo mesmo, em certas situações, esses ganhos serem superiores a 50%.

De referir, no entanto, que os resultados podem sofrer algumas oscilações, uma vez que alguns dos testes efectuados envolvem aplicações de carácter aleatório. A existência de outras aplicações em execução no sistema também poderá contribuir para essas mesmas oscilações.

De entre as várias soluções propostas para o controlo do consumo energético, a mais adequada dependerá do cenário concreto onde vai ser utilizada. Não existe uma solução óptima, existem sim, várias boas soluções podendo ser melhores ou piores, dependendo do contexto.

7. Futuros desenvolvimentos

Como futuro desenvolvimento propõem-se vários pontos:

- Implementação não só a nível da mudança da frequência e prioridades mas também através do desligamento temporário de certas partes do sistema
- Optimização dos algoritmos de decisão da frequência adequada para o momento
- Maior integração ao nível do escalonador do sistema operativo podendo associar a frequência de funcionamento do processador a requisitos particulares de cada processo.
- Melhoramento dos mecanismos de comunicação entre as aplicações e o controlador enviando informação que possa ser útil noutros mecanismos de poupança de energia
- Optimização do código de forma a poupar mais energia com o cálculo das frequências adequadas e das prioridades

8. Bibliografia

Hewlett-Packard/Intel/Microsoft/Phoenix/Toshiba, "Advanced Configuration and Power Interface Specification", Revision 3.0 September 2, 2004.

Intel, "Intel® Pentium® M Processor on 90 nm Process with 2-MB L2 Cache", January 2005.

Intel, "Intel® Centrino™ Mobile Technology Performance Brief", May 2004.

Manuel de Medeiros Silva, "Circuitos com transístor bipolares e mos", Fundação Calouste de Gulbenkian, 1999.

José Fernando Alves da Silva, "Electrónica industrial", Fundação Calouste de Gulbenkian, 1998.

Trevor Pering, Tom Burd, Robert Brodesen, "The simulation and evaluation of dynamic voltage scaling algorithms", presented at Low Power Electronics and Design, 1998.

C.M. Krishna, Yann-Hang Lee, "Voltage-Clock-Scaling Adaptative Scheduling Techniques for Low Power in Hard Real-Time Systems", presented at IEE Transactions on computers, vol 52, nº 12, dezembro de 2003.