Um Gerente de Energia para Sistemas Profundamente Embarcados

Geovani R. Wiedenhoft, Arliones Hoeller Jr., Antônio A. Fröhlich

[1]Laboratório de Integração Software e Hardware
Universidade Federal de Santa Catarina
PO Box 476 – 88049-900 – Florianópolis, SC, Brasil

{grw,arliones,guto}@lisha.ufsc.br

Abstract. *Deeply embedded systems are designed to perform a certain set of tasks, and present limitations regarding processing and memory capabilities. In many cases, these systems are powered by batteries, requiring efficient power management. In this paper, we present a dynamic power manager with no significant overhead to the application. This manager uses the power management infra-structures present in the* EPOS *operating system, and is able to save power in different application scenarios.*

Resumo. *Sistemas profundamente embarcados são projetados a atender um determinado conjunto de tarefas, e apresentam limitações com relação as capacidades de processamento e memória. Em muitos casos, esses sistemas são alimentados por baterias, requerendo uma gerência de energia eficiente. Neste artigo, nós apresentamos um gerente de energia dinâmico sem sobrecustos significativos para a aplicação. Esse gerente usa a infra-estrutura de gerência de energia presente no sistema operacional* EPOS*, e é capaz de economizar energia em diferentes cenários de aplicações.*

1. Introdução

Sistemas profundamente embarcados são plataformas projetadas a atender um determinado conjunto de tarefas, como monitorar e/ou controlar diferentes ambientes nos quais estão inseridos. Normalmente, esses sistemas apresentam limitações nas capacidades de processamento e memória, e em muitos casos, são alimentados por baterias. Dessa forma, o tempo de vida da bateria é um fator determinante para a disponibilidade desses sistemas.

Dadas as limitações de sistemas profundamente embarcados, a gerência de energia deve ser realizada de uma forma eficiente e sem sobrecustos para a aplicação. As plataformas de *hardware* desses sistemas permitem mudanças de modos de operação para economizar energia, através de mecanismos como DVS (*Dynamic Voltage Scaling*) [Pouwelse et al. 2001, Weiser et al. 1994] e hibernação de recursos [Kravets and Krishnan 1998, Helmbold et al. 1996, Flinn and Satyanarayanan 1999]. As trocas de modos de operação devem ser realizadas com cautela para não interferir na execução da aplicação e para não aumentar o consumo de energia em vez de diminuir, como as trocas de modos de operação consomem uma considerável quantia de energia. Com objetivos de diminuir os sobrecustos em termos de processamento e de memória, muitas vezes a gerência de energia é realizada pela própria aplicação, e não pelo sistema operacional.

Entretanto, se existir uma infra-estrutura de gerência de energia adequada no sistema operacional, deve ser possível construir um gerente de energia, que seja independente da gerência de energia da aplicação para sistemas profundamente embarcados.

Este trabalho explora um gerente de energia que não ocasiona sobrecustos significativos para sistemas profundamente embarcados, ou seja, com uma reduzida interferência na execução de aplicações. Hoeller [Hoeller et al. 2006] implementou uma infra-estrutura de gerência de energia para o sistema operacional EPOS [Marcondes et al. 2006]. Nessa infra-estrutura todos os componentes do sistema respondem a uma API uniforme de gerência de energia, e as relações entre os componentes são formalizadas através de redes de Petri para permitir a geração automática dos métodos de trocas de estados de energia, com baixos sobrecustos. Nós usamos essa infra-estrutura para desenvolver um gerente de energia dinâmico para sistemas profundamente embarcados. Esse gerente de energia usa heurísticas "replugáveis" para a gerência de energia, permitindo configurabilidade e adaptabilidade para aplicações específicas. O algoritmo de gerência de energia usado neste trabalho apresentou resultados promissores de economia de energia em diferentes cenários de aplicações.

Este artigo é estruturado como segue. Seção 2 apresenta a infra-estrutura de gerência de energia do EPOS. Seção 3 descreve a implementação do gerente de energia proposto no EPOS. Seção 4 abrange as avaliações da estratégia de gerência de energia dinâmica, comparando com a gerência de energia dirigida pela aplicação. Seção 5 apresenta os trabalhos relacionados. Seção 6 finaliza resumindo o artigo.

2. Infra-estrutura de Gerência de Energia no EPOS

EPOS (*Embedded Parallel Operating System*) [Fröhlich 2001] é um *framework* baseado em componentes hierarquicamente organizados para a geração de sistemas específicos a uma determinada aplicação embarcada. Através de ferramentas de análise da aplicação é possível gerar sistemas que agreguem apenas os componentes requeridos pela determinada aplicação, sem adicionar componentes que não serão utilizados. Além disso, através das abstrações e mediadores de *hardware* [Polpeta and Fröhlich 2004], o EPOS permite o desenvolvimento de aplicações totalmente independentes de plataformas.

No EPOS, todo componente do sistema implementa uma interface uniforme de gerência de energia [Hoeller et al. 2006]. Essa infra-estrutura permite realizar a gerência de energia em diferentes níveis através da hierarquia de componentes do sistema, desde uma abstração de alto nível até o acesso direto aos modos de operação do *hardware*. Além disso, a infra-estrutura permite realizar a gerência de diferentes subsistemas (processamento, comunicação) ou no sistema como um todo, através de um componente global, que "conhece" todos os outros componentes instânciados no sistema. As relações entre os componentes foram formalizadas através de redes de Petri para permitir a geração automática dos métodos de troca de modo de operação em cada componente, com um reduzido sobrecusto e de uma forma consistente.

A interface de gerência de energia no EPOS é composta por dois métodos: um para acessar o modo de operação corrente do componente, e outro para mudá-lo. Essa interface permite aplicações realizarem a gerência de energia sem ocasionar sobrecustos significativos à aplicação. Com essa interface, o programador insere diretamente as instruções de gerência de energia no código da aplicação. Por exemplo, se em um certo

ponto da aplicação o programador sabe que o componente UART não será utilizado por um longo período de tempo, ele manualmente insere uma instrução para mudar o modo de operação do componente para `OFF` ou `STANDBY`. No momento em que o componente é acessado novamente, a infra-estrutura de gerência de energia do EPOS assegura que o componente retorne ao seu modo de operação anterior (`FULL` ou `LIGHT`).

Como a gerência de energia é uma propriedade não funcional no contexto de sistemas computacionais [Lohmann et al. 2005], a infra-estrutura de gerência de energia do EPOS foi implementada como um aspecto [Kiczales et al. 1997], associado à programação de meta-nível no contexto de AOP. Com isso, essa infra-estrutura é isolada do restante do sistema, e deve ser ativada ou desativada conforme as necessidades de cada aplicação.

3. O Gerente de Energia

A gerência de energia dinâmica é usualmente realizada por um agente externo à aplicação, como por exemplo, pelo sistema operacional ou a BIOS, o qual monitora a utilização de certos recursos. Esse monitor analisa o comportamento do sistema, verificando estatísticas de uso de diferentes componentes. Essas informações são usadas para guiar as estratégias de gerência de energia, e usualmente causa alterações dos modos de operação de certos recursos. Isso libera o programador da necessidade de implementar as estratégias de gerência de energia na própria aplicação. Os resultados das estratégias de gerência de energia em termos de economia de energia são diretamente relacionados a qualidade dos algoritmos utilizados pelo monitor. Entretanto, as limitações de sistemas profundamente embarcados impede a utilização de algoritmos de gerência de energia complexos.

Uma estratégia simples para a gerência de energia dinâmica faz uso de um escalonador no sistema operacional. No momento em que o escalonador não tenha tarefas a serem escalonadas, o processador principal do sistema é colocado no modo *standby*. Entretanto, essa técnica é muito limitada, pois ela não considera componentes periféricos, os quais em sistemas profundamente embarcados usualmente consomem mais energia do que o micro-controlador principal. Para contemplar esses dispositivos, um gerente de energia é requerido. Esse gerente verifica a utilização de cada componente, e se um certo componente está inativo por um determinado período de tempo, ele altera o modo de operação desse componente para um estado mais baixo. Uma técnica para detectar a ociosidade de componentes utiliza “contadores” de uso em *hardware* para cada componente. Entretanto, as plataformas de sistemas profundamente embarcados tipicamente não possuem essa técnica em *hardware*, e uma infra-estrutura baseada em *software* deve ser usada.

Este trabalho estendeu a infra-estrutura de gerência de energia disponibilizada no EPOS para o desenvolvimento de um gerente de energia para sistemas profundamente embarcados. Esse gerente permite que, no momento da geração do sistema, o programador configure qual será o algoritmo de gerência de energia a ser utilizado, através de heurísticas “replugáveis”. Essa configurabilidade é importante, pois o algoritmo de gerência é um dos principais fatores de influência no equilíbrio entre economia de energia e sobrecustos para a aplicação. Dessa forma, a configuração do algoritmo de gerência permite que o programador da aplicação decida qual o algoritmo de gerência do consumo de energia seu sistema suporta, sem que ocasione interferências significativas para

a aplicação.

Esta seção descreve em detalhes o algoritmo de gerência de energia implementado no gerente de energia e apresenta seus modos de operação.

3.1. Algoritmo do Gerente de Energia

Como descrito anteriormente, sistemas profundamente embarcados não podem arcar com os custos de algoritmos complexos de gerência de energia. Dessa forma, nós implementamos um algoritmo de gerência de energia que objetiva interferir o mínimo possível na execução da aplicação. Esse algoritmo é implementado para a gerência de todos os componentes do sistema.

A figura 1 é um diagrama de tempo da UML que representa nosso algoritmo de gerência de energia. Os estágios do algoritmo são apresentados, desde os testes em busca de um tempo ocioso do componente, até a verificação de ociosidade e então troca do modo de operação do dispositivo, que nesse caso, é a migração do modo ligado para o modo desligado do componente UART. Os estados e valores da *thread* da aplicação, da *thread* do gerente de energia e do componente UART são representados nessa figura. A UART foi utilizada como um exemplo de componente a ser monitorado pelo gerente.

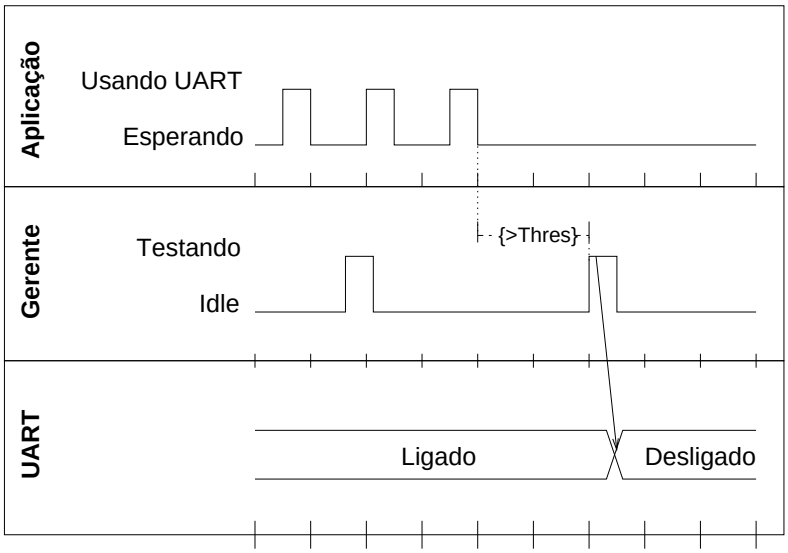

Figura 1. Algoritmo de gerência do consumo de energia.

Em termos gerais, o funcionamento do algoritmo ocorre da seguinte forma:

- A data (*timestamp*) de utilização de cada componente monitorado é atualizada a cada acesso a esse componente (“contador do acesso”);
- O gerente de energia é executado em tempos intercalados (dependente do escalonador);
- No momento da execução do gerente é obtida a data corrente (*timestamp*). Um cálculo é realizado dessa data menos a data armazenada do último acesso ao componente;
- O modo de operação do componente é alterado caso o resultado obtido seja maior do que um valor pré-definido (*threshold*), pois esse valor indica ao gerente que o componente está ocioso.

3.2. Modos de Operação do Gerente

Neste trabalho, o gerente de energia foi implementado como um aspecto para permitir que ele seja isolado do sistema. Isso possibilitou o desenvolvimento de diferentes características de modos de operação que são configurados pelo programador da aplicação na geração do sistema:

- O programador da aplicação pode escolher se o gerente será habilitado para realizar a gerência de energia ou não. Com a ativação do gerente, a aplicação não necessita utilizar interfaces de gerência de energia;
- Configurar apenas os componentes que a aplicação deseja que o gerente de energia monitore, com objetivo de atender aos requisitos de cada aplicação específica. Com isso, a gerência de energia de componentes indesejados pela aplicação não é adicionada no sistema final, apesar de ter sido implementada para todos os componentes do sistema;
- Escolher se o gerente será ativo ou passivo no sistema. Na gerência ativa, o gerente executa ativamente em intervalos de tempo. No modo passivo, outro componente deve iniciar a execução do gerente.

4. Avaliações do Gerente de Energia

Nesta seção nós avaliamos o gerente de energia implementado neste trabalho, comparando o mesmo com a gerência dirigida pela aplicação que é disponibilizada pela infra-estrutura de gerência de energia do EPOS. As economias de energia, o código agregado e os cenários que cada técnica é capaz de obter melhores resultados são apresentados.

4.1. Ambiente de Execução

Neste trabalho, os testes foram realizados com um micro-controlador ATMega16 [Corporation 2004], da Atmel, no kit de desenvolvimento STK500. Para esses testes foram utilizadas duas aplicações: uma determinística e outra não determinística. A figura 2 apresenta os algoritmos das aplicações testadas neste trabalho. Essas aplicações, de uma forma geral, esperam por um tempo determinado (determinístico) ou aleatório (não-determinístico), e escrevem um valor constante na UART.

Esses testes foram realizados apenas com a utilização do componente UART. A UART consome uma energia considerável para realizar as trocas de modos de operação. Entretanto, a UART é um dos recursos que menos consome energia total do sistema, fazendo com que, os resultados ficassem relativamente baixos.

Neste trabalho foram realizadas diversas abordagens de testes, como as apresentadas a seguir:

- Com o equipamento desligado;
- Aplicação sem a gerência de energia;
- Com a gerência de energia dirigida pela aplicação, que é disponibilizada pela infra-estrutura de gerência do EPOS;
- Com o gerente de energia desenvolvido neste trabalho.

Nas medições com o gerente de energia, nós intercalamos valores para o *threshold* de 10000, 5000 e 0 micro-segundos. Em todos os testes de medições foram realizadas 20 iterações de 10 segundos cada. Os resultados considerados foram obtidos através da média aritmética dos valores restantes com a exclusão dos 20% menores valores e dos 20% maiores valores.

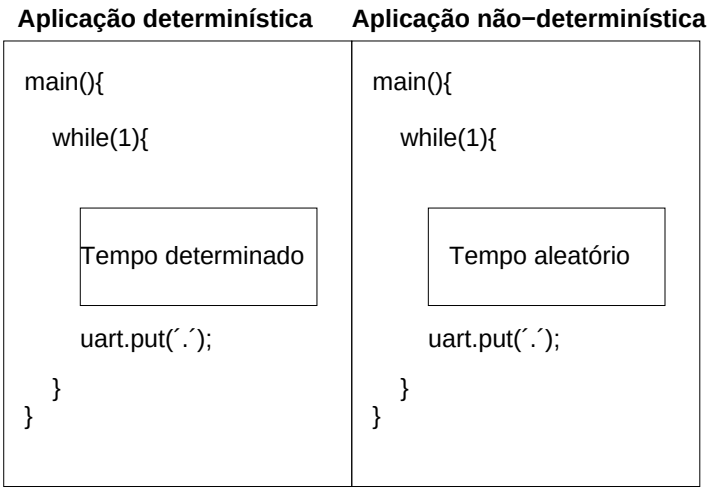

Figura 2. Algoritmo das aplicações.

4.2. Análise dos Resultados

As tabelas a seguir apresentam os resultados obtidos com os testes descritos anteriormente. Nessas tabelas nós avaliamos o código agregado na aplicação e a economia de energia nos diferentes testes realizados.

Na tabela 1 nós temos os resultados obtidos nos testes realizados com a aplicação determinística. Com esses resultados nós podemos inferir que na aplicação determinística a gerência de energia dirigida pela aplicação apresenta um resultado melhor, economizou 4,8% de energia. Além disso, nós podemos observar nessa tabela que os resultados da economia de energia do gerente tenderam a um *threshold* igual a zero, ou seja, essa técnica para uma aplicação determinística tendeu à gerência dirigida pela aplicação. Esses resultados podem ser explicados através do conhecimento do programador, que sabe cada detalhe do programa determinístico. Nesse teste, o programador sabe que a aplicação escreve um valor na UART e fica sem fazer nada por um determinado tempo. Durante o intervalo de tempo ocioso da aplicação é realizada a gerência de energia pelo programador, que aproveita todo o tempo ocioso da aplicação. Já o gerente necessita descobrir através de algoritmos qual é o tempo em que a aplicação fica ociosa.

Tabela 1. Resultados obtidos nos testes com a aplicação determinística.

Arquitetura	Tamanho agregado (Bytes)	Energia consumida (J)	Economia%
Desligado	-	**0,196**	-
Sem gerência	-	0,3156	-
Dirigida pela aplicação	548	0,3004	4,80%
Threshold=10000	1488	0,3079	2,50%
Threshold=5000	1488	0,3043	3,60%
Threshold=0	1488	0,3008	4,70%

A tabela 2 apresenta os valores obtidos nos testes realizados com a aplicação não determinística. Com os resultados dessa tabela nós podemos inferir que em aplicações não determinísticas o gerente obteve resultados mais satisfatórios, economizou 1% de energia.

Dessa forma, nós podemos concluir que para aplicações não determinísticas é preferível esperar um determinado tempo para verificar se o componente não será utilizado novamente antes de trocar o seu modo de operação. Isso ocorre pelo fato do componente consumir mais energia para trocar seu modo de operação, do que, permanecer no modo ligado e ocioso por um determinado tempo. Com isso, o gerente obtém um melhor aproveitamento em aplicações não determinísticas. Nesses casos, o programador desconhece os detalhes específicos do que a aplicação executará e por quanto tempo essa execução ocorrerá, pois a aplicação pode ser influenciada por parâmetros do ambiente.

Tabela 2. Resultados obtidos nos testes com a aplicação não determinística.

Arquitetura	Tamanho agregado (Bytes)	Energia consumida (J)	Economia%
Desligado	-	**0,196**	-
Sem gerência	-	0,3053	-
Dirigida pela aplicação	552	0,3042	0,40%
Threshold=10000	1040	0,3034	1,00%
Threshold=5000	1040	0,3014	1,30%
Threshold=0	1040	0,3037	0,50%

5. Trabalhos Relacionados

TinyOS [Hill et al. 2000] é um sistema operacional baseado em eventos, desenvolvido para sistemas embarcados, mais especificamente, para redes de sensores sem fio. A gerência de energia no TinyOS é realizada pelo escalonador de tarefas, que utiliza uma interface que provê métodos de início e parada de componentes. No momento em que a fila de tarefas está vazia, o escalonador pára o processador, mas os periféricos ficam ligados. O processador é reativado quando ocorre um evento de interrupção para uma nova tarefa. Nesse sistema, mecanismos mais sofisticados são deixados a cargo da aplicação.

Mantis OS (*Multimodal Networks of In-situ Sensors*) [Abrach et al. 2003] é um sistema operacional *multithread*, que possui uma API baseada em POSIX adaptada às necessidades das redes de sensores sem fio. A gerência do consumo de energia do Mantis OS é similar à função UNIX `sleep`. Para utilizá-la, uma aplicação deve habilitar a gerência de energia através de uma função específica, e a partir disso, é possível utilizar uma outra função que especifica o tempo em que a aplicação pode "dormir". O escalonador é ativado através de uma chamada para essa última função, e caso não hajam *threads* a serem escalonadas, a gerência de energia coloca o processador em um estado de *sleep*. Essa gerência controla apenas o processador e não há um controle específico para os periféricos, além dos fornecidos pelos *drivers*.

SOS [Han et al. 2005] é um sistema operacional para redes de sensores sem fio que possui o objetivo de tratar a reconfiguração dinâmica dos serviços de propagação de mensagens, alocação dinâmica de memória e carga dinâmica de módulos. A gerência de energia do SOS é semelhante ao encontrado no TinyOS e Mantis OS. No momento em que a fila do escalonador não possua mensagens a serem escalonadas, o modo de operação do processador é alterado para um estado de baixo consumo, mas os periféricos continuam ligados. No SOS, os módulos podem implementar mecanismos individuais de gerência de energia de seus recursos utilizados.

A principal gerência de energia realizada por esses sistemas operacionais é a troca do modo de operação do processador no momento em que não existam tarefas, *threads* ou mensagens a serem escalonadas. Dessa forma, eles não abordam técnicas específicas para a gerência do consumo de energia dos periféricos.

6. Conclusão

Este artigo apresentou o desenvolvimento de um gerente de energia para sistemas profundamente embarcados com um baixo sobrecusto para a aplicação. Esse gerente estende a infra-estrutura de gerência de energia disponibilizada no sistema operacional EPOS, e é capaz de economizar energia em diferentes cenários. A implementação dessa extensão foi realizada como um aspecto, o que permitiu uma alta configurabilidade do gerente, incluindo a possibilidade de escolha se o gerente será habilitado ou não, a possibilidade de configurar somente os componentes desejados pela aplicação para a gerência, e se o gerente será ativo ou passivo na gerência de energia. Além disso, nosso gerente de energia possui a característica de heurísticas "replugáveis" que permite o programador da aplicação selecionar, na geração do sistema, qual estratégia de gerência de energia é mais adequada para seus requisitos. Os resultados obtidos em nossos testes mostraram que o gerente de energia apresenta vantagens com relação a gerência de energia dirigida pela aplicação em diferentes cenários de aplicações.

Referências

Abrach, H., Bhatti, S., Carlson, J., Dai, H., Rose, J., Sheth, A., Shucker, B., Deng, J., and Han, R. (2003). Mantis: System support for multimodal networks of in-situ sensors. In *2nd ACM International Workshop on Wireless Sensor Networks and Applications*, pages 50 – 59, San Diego, CA.

Corporation, A. (2004). *ATMega16L Datasheet*. San Jose, CA, 2466j edition.

Flinn, J. and Satyanarayanan, M. (1999). Energy-aware adaptation for mobile applications. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pages 48–63, New York, NY, USA. ACM Press.

Fröhlich, A. A. (2001). *Application-Oriented Operating Systems*. GMD - Forschungszentrum Informationstechnik, Sankt Augustin.

Han, C.-C., Kumar, R., Shea, R., Kohler, E., and Srivastava, M. (2005). A dynamic operating system for sensor nodes. In *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 163–176, New York, NY, USA. ACM Press.

Helmbold, D. P., Long, D. D. E., and Sherrod, B. (1996). A dynamic disk spin-down technique for mobile computing. In *MobiCom '96: Proceedings of the 2nd annual international conference on Mobile computing and networking*, pages 130–142, New York, NY, USA. ACM Press.

Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, K. (2000). System architecture directions for networked sensors. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, Cambridge, Massachusetts, United States.

Hoeller, A. S. J., Wanner, L. F., and Fröhlich, A. A. (2006). A Hierarchical Approach For Power Management on Mobile Embedded Systems. In *5th IFIP Working Conference on Distributed and Parallel Embedded Systems*, pages 265–274, Braga, Portugal.

Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J.-M., and Irwin, J. (1997). Aspect-Oriented Programming. In *Proceedings of the European Conference on Object-oriented Programming'97*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242, Jyväskylä, Finland. Springer.

Kravets, R. and Krishnan, P. (1998). Power management techniques for mobile communication. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 157–168, New York, NY, USA. ACM Press.

Lohmann, D., Schröder-Preikschat, W., and Spinczyk, O. (2005). Functional and non-functional properties in a family of embedded operating systems. In *Proceedings of the Tenth IEEE International Workshop on Object-oriented Real-time Dependable Systems*, Sedona, USA. IEEEPress.

Marcondes, H., Junior, A. S. H., Wanner, L. F., and Fröhlich, A. A. (2006). Operating Systems Portability: 8 bits and beyond. In *11th IEEE International Conference on Emerging Technologies and Factory Automation*, pages 124–130, Prague, Czech Republic.

Polpeta, F. V. and Fröhlich, A. A. (2004). Hardware Mediators: a Portability Artifact for Component-Based Systems. In *International Conference on Embedded and Ubiquitous Computing*, volume 3207 of *Lecture Notes in Computer Science*, pages 271–280, Aizu, Japan. Springer.

Pouwelse, J., Langendoen, K., and Sips, H. (2001). Dynamic voltage scaling on a low-power microprocessor. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 251–259, New York, NY, USA. ACM Press.

Weiser, M., Welch, B., Demers, A. J., and Shenker, S. (1994). Scheduling for reduced CPU energy. In *OSDI*, pages 13–23.