DESENVOLVIMENTO DE UM MICRO NÚCLEO DE TEMPO REAL PARA O MICRO CONTROLADOR 89C8252 PARA O ENSINO DE PROGRAMAÇÃO PARA TEMPO-REAL

André Luiz Duarte Cavalcante – andrecavalcante@ufam.edu.br
Vicente Ferreira de Lucena Junior – vicente@ufam.edu.br
Universidade Federal do Amazonas - UFAM
Centro de P&D em Tecnologia Eletrônica e da Informação - CETELI
Avenida General Rodrigo Otávio Jordão Ramos, 3000, Campus Universitário
Setor Norte, CETELI, Sala de Professores 3
69077-000 - Manaus – Amazonas
José Antônio de Oliveira Filho – jaoliveirafilho@gmail.com
Universidade Federal do Amazonas - UFAM
Curso de Engenharia da Computação
Avenida General Rodrigo Otávio Jordão Ramos, 3000, Campus Universitário
Setor Norte, CETELI, Secretaria
69077-000 - Manaus – Amazonas

Resumo: *O desenvolvimento de núcleos e executivos para a família de controladores 8051 é uma tarefa corriqueira, dada a quantidade deles que são encontrados em qualquer simples busca na Internet. Entretanto, o ensino de disciplinas ligadas a sistemas operacionais e, em especial, a programação de tempo real, utilizando-se um microcontrolador, esbarra tanto nas limitações do dispositivo quanto da possibilidade de se desenvolver de forma mais completa a teoria vista. Este artigo aborda o desenvolvimento de um destes núcleos para o 8051, que pretende contornar as limitações normalmente encontradas em núcleos do gênero e poder desenvolver toda a teoria necessária para o bom domínio da disciplina, tornando-se, portanto, uma ferramenta extremamente válida do ponto de vista do aprendizado, tanto do controlador, quanto da construção do núcleo em si. O trabalho aqui apresentado foi resultado de experimentos técnico-pedagógicos desenvolvidos na disciplina Programação para Tempo-Real dos cursos de engenharia elétrica e engenharia da computação da Universidade Federal do Amazonas em Manaus.*

Palavras-chave: *Uso de Microcontroladores em Laboratório, Ensino de Programação de Sistemas Mínimos, Programação em Tempo Real, Ensino de Sistemas Operacionais de Tempo Real.*

1 INTRODUÇÃO

Os cursos de engenharia elétrica e engenharia da computação possuem uma quantidade considerável de disciplinas cobrindo aspectos básicos e avançados de eletrônica digital e

temas associados. Uma seqüência natural do ensino desta área do conhecimento envolve uma disciplina com os fundamentos de eletrônica digital, uma segunda disciplina com temas avançados chegando aos fundamentos de microprocessadores e microcontroladores e em alguns currículos disciplinas optativas sobre o uso particular de microcontroladores.

Na Universidade Federal do Amazonas (UFAM) este modelo é reproduzido nos cursos acima citados. Com nomenclaturas ligeiramente diferentes segue-se o roteiro acima descrito. Complementarmente oferecem-se ainda disciplinas avançadas nesta área como sistemas embarcados, engenharia de software e programação de sistemas de tempo real.

Esta tendência consolida-se na UFAM com a experiência prática de um grupo de professores que vêm desenvolvendo pesquisas aplicadas a problemas industriais no Centro de Centro de Pesquisa e Desenvolvimento de Tecnologia Eletrônica e da Informação (CETELI). Um exemplo desta abordagem é o uso de robôs móveis para ensino de engenharia de software descritos em (Lucena *et al*, 2006) no COBENGE 2006 em (Lucena & Brito 2006).

De fato, dada as características da indústria local, o uso de sistemas microcontrolados para solucionar problemas de automação e de controle deve ser um dos conhecimentos adquiridos pelos graduandos de engenharia aqui em Manaus. Da literatura pode-se afirmar que esta não é uma preocupação única da UFAM. A preocupação com o ensino de sistemas de tempo real e a apresentação de novas abordagens para garantir um aprendizado mais eficiente são os temas centrais de trabalhos como (Vallino & Czernikowski, 2004 e 2005), ou (Guzman *et al* 2005). De fato desde a década de 90 trata-se deste assunto (ver por exemplo Wick, 1996 ou Horowitz *et al*, 1999).

Este trabalho descreverá uma experiência pedagógica realizada na disciplina Programação para Tempo Real, ministrada para alunos dos cursos de engenharia elétrica e engenharia da computação na Faculdade de Tecnologia da UFAM. Apesar de sua importância no cenário tecnológico atual, trate-se de uma disciplina optativa para os dois cursos. Esta situação demanda do professor um esforço maior em conquistar alunos que venham a cursar a matéria efetivamente e desenvolver os trabalhos de forma a garantir o seu aprendizado.

Apesar de farta disponibilidade de livros e outros materiais bibliográficos referentes à programação de tempo real, nota-se que são poucos os recursos para viabilizar experimentos práticos relacionados a este tema. O uso de computadores de mesa com simuladores preenche uma lacuna importante, mas não atende as reais expectativas dos alunos. Experiências bem sucedidas de tais simuladores apontam para a necessidade de uma complementação através do desenvolvimento de dispositivos reais de tempo real (Wick, 1996 ou Martinez *et al*, 1999). Relatos interessantes de experimentos práticos bem sucedidos podem ser encontrados em (Durfee & Waletzko, 2004; Greewald & Kopena, 2003 ou Jazdi *et al* 2006).

De fato pode-se afirmar que a programação de microcontroladores visando aplicações industriais implica no uso de técnicas de programação de tempo real, uma vez que estes dispositivos microcontrolados serão dispositivos voltados para aplicações onde o tempo é um fator determinante. Neste sentido, o ensino em disciplinas de graduação de tais matérias deve contar com uma parte teórica sólida, mas deve igualmente privilegiar a prática com experiências e trabalhos que se aproximem o mais possível das aplicações industriais reais.

Este artigo aborda o desenvolvimento de um experimento voltado para o melhor aprendizado de um sistema operacional de tempo real com aplicações industriais para microcontroladores da família 8051. Foi desenvolvido um núcleo operacional (kernel) abordando toda a teoria relacionada ao assunto, tornando-se uma ferramenta extremamente válida do ponto de vista do aprendizado, tanto do microcontrolador, quanto da construção do núcleo em si. Para tal iniciaremos o artigo descrevendo parcialmente o controlador 89C252, um exemplar típico da família 8051, mas com algumas grandes vantagens em relação a um microcontrolador 8051 padrão. Depois passaremos a uma visão rápida de como ocorre a relação ensino/aprendizagem das disciplinas relacionadas à programação em tempo real. Por

fim, descreveremos o desenvolvimento do núcleo alvo desde artigo, bem como os trabalhos futuros relacionados a este núcleo e ao ensino de microcontroladores.

2 O MICROCONTROLADOR 89C8252

O microcontrolador 89C8252 é um microcontrolador de 8 bits, desenvolvido pela Atmel, da família 8051, que conta com alguns periféricos integrados e a facilidade de gravação *on-board* de sua memória de programa interna. A memória em questão possui a capacidade de 8 Kbytes, o que é plenamente satisfatório para processadores desta categoria e para o experimento desejado. Também possibilita a execução de instruções a cada 4 ciclos de máquina, em vez dos 12 ciclos do 8051 padrão. A única restrição deste microcontrolador é o fato de possuir apenas os 252 bytes de RAM, quantidade exatamente igual à disponível no 8051 padrão. Tal limitação pode ser minimizada se utilizarmos 89CS8252, que conta com 1 KBytes de RAM e é pino compatível com o outro.

Controladores da família 8051 são bastante utilizados tanto em aplicações reais quanto em aplicações experimentais em cursos de engenharia. Esta popularidade se deve principalmente pelo custo relativamente baixo dos dispositivos, pela grande quantidade de periféricos integrados existentes e por possuir uma comunidade grande e ativa de usuários que o conhecem muito bem. A quantidade de ferramentas que auxiliam os projetistas de hardware e de software para a família 8051 é também respeitável o que reforça a escolha desta plataforma para os mais diversos usos.

Há diversos kits montados a partir deste processador. Para o experimento aqui relatado, procurou-se um kit didático que facilitasse o desenvolvimento do software, sem que se perdesse muito tempo com a programação básica do hardware. Ou seja, não foi escopo deste trabalho a construção e programação dos elementos básicos para que microcontrolador pudesse funcionar, nem do hardware periférico, como teclado, display de LCD etc.. Optou-se por uma solução que já contasse com este hardware mínimo e que o pudéssemos utilizar *out-of-the-box*. Também era importante que fosse de baixo custo, uma vez que se tem um objetivo didático e tal kit deveria ser adquirido tanto pela instituição de ensino como pelos próprios alunos.

Escolheu-se um kit chamado de KIT AT89S. Abaixo elenca-se as suas principais características. Por simplicidade, utilizou-se a terminologia corrente, mantendo-se alguns termos do jargão em inglês.

- Microcontrolador AT89S8252, com 8KBytes de memória FLASH, 256 bytes de RAM, 2kBytes de EEPROM;
- Possui 8 LEDs para indicação visual na porta 0, onde cada um pode ser habilitado/desabilitado através de um *jumper* no próprio kit. Cada LED possui um *driver* com transistor, respeitando a capacidade de corrente do microcontrolador;
- Possui 8 chaves do tipo *push-button*, para aplicações como teclado. Cada chave possui um circuito RC para eliminação do efeito *bounce*. As chaves podem ser individualmente habilitada/desabilitada através de um *jumper* no próprio kit. Quatro chaves podem ser usadas para gerar as interrupções externas 0 e 1 e os pulsos para os contadores 0 e 1;
- Todas as portas do microcontrolador (P0, P1, P2 e P3) estão disponíveis em 4 conectores, permitindo que o kit possa controlar um hardware adicional a ele conectado;
- Possui uma memória EEPROM externa (24C02), com capacidade de 256 bytes, acessada pelo protocolo de comunicação I2C. Os resistores de *pull-up* já estão no kit. A memória pode ser habilitada/desabilitada através de 2 *jumpers* no próprio kit;

- Possui uma porta serial *full duplex* UART, já no padrão RS-232 para ligar o kit em um PC, ou um PDA, por exemplo. A mesma porta pode ser usada com nível TTL através do conector da porta 3;
- Possui regulador de +5V e a entrada da alimentação possui proteção contra inversão de polaridade na fonte;
- O conector da porta 2 também pode ser usado para ligar um display de cristal líquido. Neste caso, já existe um *trimpot* para ajuste de contraste do LCD no kit;
- O microcontrolador pode ser gravado diretamente (*on-board*), sem a necessidade de retirar o chip da placa. Para isso, existe um conector para a ligação do cabo de programação que acompanha o kit;
- Possui uma chave para RESET forçado e um circuito de *Power On Reset*.
- O cristal do kit é de 11.0592MHz, permitindo as velocidades padrões para a comunicação serial (600bps, 1200bps, 4800bps, 9600bps, 14400bps, 19200bps etc.).

Além de todas as funcionalidades mencionadas, o kit acompanha um CD com diversos programas que servem de exemplo de uso de todos os periféricos. Isto minimiza sobremaneira o esforço para se começar a utilizar o kit em práticas das disciplinas de programação.

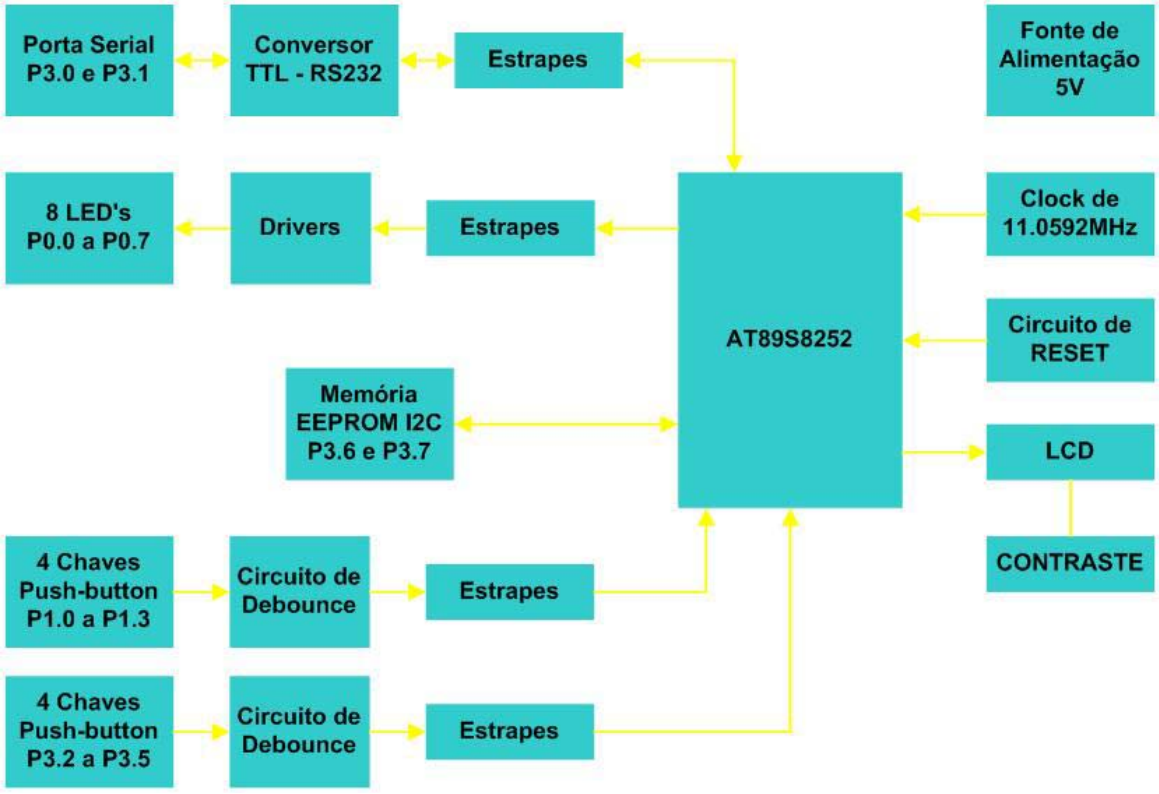

Figura 1 – Diagrama em Bloco do Kit AT89S

Maiores detalhes sobre o microcontrolador pode ser obtido em (ATMEL, 2006) e sobre o kit em (LIMA, H. O, 2006). Na Figura 1, vê-se o diagrama em blocos do kit AT89S e na Figura 2, vê-se uma foto do kit. Observar alguns dos periféricos já se encontram disponíveis na placa principal, tais como *leds* e chaves, permitindo prontamente o seu uso.

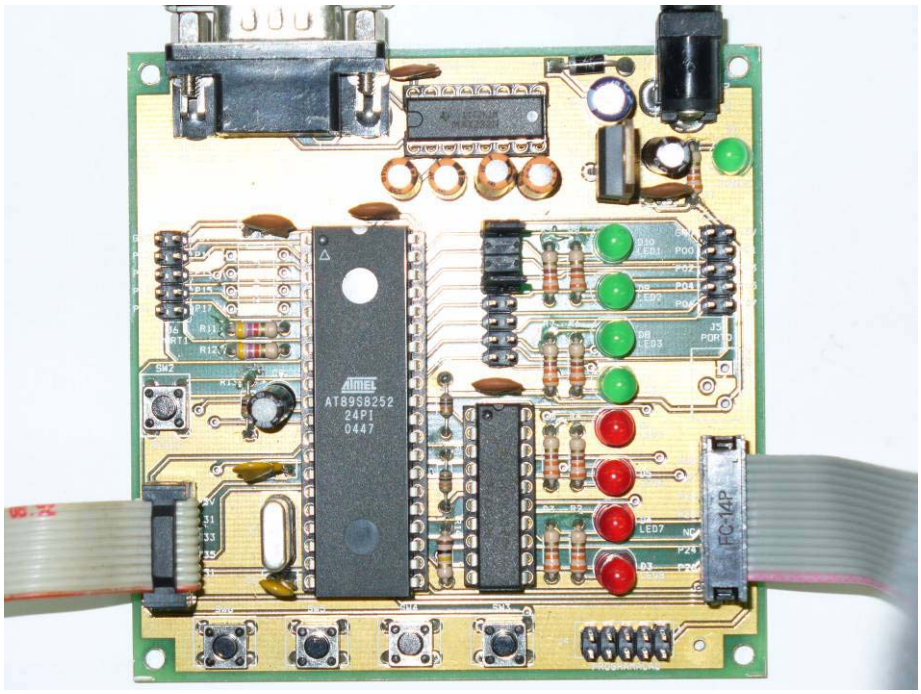

Figura 2 – Placa principal di Kit Didático AT89S e algumas de suas ligações

3 ENSINO E APRENDIZAGEM DE MICROCONTROLADORES

Em cursos de Engenharia Elétrica e/ou Computação, um grande dilema para os alunos e professores é adequar a carga horária de teoria com práticas condizentes, em especial nas disciplinas relacionadas aos microcontroladores e à sua programação. De fato, não basta somente ficar com um belo conjunto de teorias sobre microcontroladores, e sobre a programação deles, como por exemplo, a programação de sistemas de tempo real, programação de micro núcleos e executivos. Também não basta possibilitar aulas de laboratório que não condizem com as teorias vistas, como o que acontece ao se utilizar simuladores em um PC, onde o tempo da simulação não corresponde ao tempo real de execução do programa em um dispositivo real. De fato, é fácil recordar certos ensaios de laboratório em que o professor por fim diz: "Como vocês sabem, na prática as coisas acontecem um pouquinho diferentes...", para justificar alguma falha em um ensaio.

Entretanto, em programação de microcontroladores, não há espaço para tais justificativas. É extremamente frustrante, tanto para professores quanto para alunos, não experimentar os algoritmos e técnicas descritas nos livros em sistemas reais. Por outro lado é extremamente gratificante quando ao final de um curso tem-se a experiência prática de programar e ver funcionando um sistema real.

Aliar teoria e prática de maneira coerente e efetiva sempre foi um grande desafio, mas com um pouco de planejamento e esforço tanto de professores quanto de alunos, é possível vencer este desafio. A proposta aqui apresentada é viabilizar, através de um kit de fácil uso, um instrumento real em que o aluno pode testar a validade dos algoritmos e técnicas vistas em sala de aula. Mais ainda, verificar o que acontece quando não se utiliza de tais técnicas na programação de dispositivos mínimos. Um ganho duplo se verifica: a disciplina se torna extremamente interessante para professores e para os alunos e o aprendizado se torna mais fácil que é o objetivo maior deste processo.

Em disciplinas de programação de tempo real há ainda um agravante: os alunos não têm muita noção, nem experiência com sistema em que o tempo se torna a principal variável para o sucesso do sistema. Muitos não têm nenhuma vivência industrial ou estão acostumados

demais com a programação de PCs, onde a variável tempo e as capacidades de processamento e memória são raramente relevantes. Chega-se mesmo a perceber certo espanto quando o aluno é apresentado a um microcontrolador com "meros" 8 KBytes de memória de programa e 256 bytes de memória de dados, sendo que neste modesto hardware há a necessidade de se colocar um sistema com várias tarefas rodando simultaneamente.

Claro, longe se está da segurança e robustez, no que tange a regras de acesso a recursos, de um sistema microprocessado de 32 ou 64 bits, com modos protegido e de usuário, mas também as aplicações são outras. Para aplicações de controle de processos, por exemplo, o importante é que os eventos sejam tratados no momento em que ocorram e a máquina de estados que responde a estes eventos seja confiável o suficiente para funcionar em situações de repetição extremas. Por exemplo, um jig de testes em uma linha de televisores repete as suas ações a cada 15s a 20s durante praticamente as 24h (3 turnos de trabalho) por dia. Um PC comum numa situação destas estressaria rapidamente suas partes móveis, em especial o *cooler* do processador e o disco rígido, os quais são vitais para o funcionamento do mesmo. Soluções utilizando-se PCs industriais parecem ser a escolha mais adequada nesta situação, entretanto seu custo pode ser proibitivo. Pequenos dispositivos, que não possuem partes móveis, confiáveis e baratos acabam sendo a solução de escolha neste cenário.

Entretanto, a maioria dos profissionais da Engenharia da Computação está pouco a vontade neste mundo em que tudo é limitado: memória, capacidade de processamento, velocidade etc. Não raro profissionais da Engenharia Elétrica serem os responsáveis pelo desenvolvimento de tais sistemas, mesmo não tendo cadeiras específicas para tal em sua formação.

Isto posto, percebe-se claramente a necessidade de se dar certa ênfase no ensino desta categoria de hardware e, em especial, no desenvolvimento de software para aplicações de tempo real utilizando-se este tipo de hardware mínimo, tanto em cursos de Engenharia Elétrica quanto de Engenharia da Computação.

O contato do aluno com este novo pequeno mundo pode ser facilitado pelo seu acesso a kits didáticos, como o citado anteriormente. Entretanto, uma abordagem ao conjunto de técnicas e teorias sobre a programação de tais sistemas é ainda relevante para dar segurança ao aluno. Cabe ao professor a orientação adequada para o uso do dispositivo real e a sua programação com o material teórico visto.

Pode-se citar o caso da programação do agendador de um micro núcleo. É sabido que existem diversas técnicas para se programar o agendador, como pode ser visto no capítulo 2 de (TANENBAUM & WOODHULL, 2000), entretanto, para sistemas de tempo real, uma abordagem teórica e simples é se utilizar uma única prioridade para cada tarefa. Assim a própria identificação da tarefa no sistema se dá pela sua prioridade, conforme se pode encontrar em outros núcleos de tempo real (LABROSSE, 1999). O agendador, neste caso, mesmo utilizado dentro de um sistema com preempção, somente escolhe para rodar a tarefa de maior prioridade que está pronta no sistema. Intuitivamente, percebe-se que ao se garantir, de alguma forma, que a tarefa de menor prioridade consiga ser executada dentro de sua restrição temporal, todas as demais também conseguirão ser executadas dentro de suas restrições temporais. Essa "de alguma forma" normalmente recai para algum tipo de compartilhamento do tempo por cooperação, isto é, as tarefas de maior prioridade é que definem a quantidade de tempo que utilizarão e permitirão que as outras tarefas executem no tempo restante, novamente isto mesmo em sistemas com preempção.

Veja-se, somente a partir deste exemplo, que a relação entre o que se estuda em termos de teoria com o que se pode programar em um dispositivo real está intimamente ligada a ter um completo domínio tanto de um quanto do outro elemento da equação. Uma orientação segura do professor é fundamental para que o aluno adquira a intuição da programação em tempo real em dispositivos mínimos.

Para tal o processo ensino/aprendizagem deve iniciar com a escolha dos conteúdos e do kit didático. Para cada conteúdo, deve ser inserido um algoritmo ou algum desenvolvimento no laboratório. No início do curso, aproveita-se ainda uma lacuna existente entre o que está sendo visto em termos de teoria para se iniciar o aluno na linguagem e no ambiente de desenvolvimento do kit. Como sugestão indica-se uma ordem da teoria a ser apresentada e os ensaios em laboratório, o que deve ser feito simultaneamente, e está exposto na Tabela 1. Cabe ao professor orientar os alunos no uso do kit, bem como na linguagem utilizada (C ou Assembly).

Tabela 1 – Sugestão de Ordem de Apresentação dos Conteúdos

Conteúdo	***Laboratório***
Noção de preempção, concorrências e paralelismo, e processos e tarefas.	Estudo do ambiente de desenvolvimento do kit e suas linguagens (normalmente C e Assembly)
Programação concorrente em tempo real (brando e rígido)	Utilização de temporizadores (*timers*) do microcontrolador
Implementação de concorrência (troca de contexto)	Bloco de Controle de Tarefa (BCT), fila de tarefas, utilização da pilha, criação de tarefas
Comunicação interprocesso (CIP): Semáforos	Chamadas *sleep* e *wakeup*.
CIP: Semáforos temporizados	Chamadas *sleep* e *wakeup* com temporização
CIP: Mensagens	Chamadas *send* e *receive*.

Neste artigo, mostraremos alguns exemplos da aplicação destes conteúdos e dos algoritmos utilizados através da descrição do micro núcleo Sauim, bem como o resultado destes esforços no ensino/aprendizado do microcontrolador e da programação concorrente, em especial a programação de tempo real.

4 DESENVOLVIMENTO DO MICRO NÚCLEO SAUIM

O micro núcleo Saium foi desenvolvido durante o período letivo 2006/2 da disciplina programação de tempo real do curso de engenharia da computação da Universidade Federal do Amazonas. O nome é uma homenagem a um tipo de macaco habitante das matas do Campus Universitário da UFAM, cujos exemplares somente são encontrados lá e nos arredores da cidade de Itacoatiara, cidade vizinha de Manaus, onde fica a sede do da Campus da UFAM, e que encontra-se ameaçado de extinção, o Sauim de Coleira.

Esta seção descreve o desenvolvimento do micro núcleo a partir dos problemas encontrados no seu desenvolvimento, tendo sempre em mente o seu objetivo de ser didático, isto é, auxiliar no amadurecimento dos alunos quanto da programação de microcontroladores e a programação de tempo real. A Figura 3 mostra um diagrama esquemático deste núcleo operando entre um hardware e gerenciando tarefas de aplicação.

Tendo isto em mente, este micro núcleo foi construído para comportar a maior quantidade de elementos teóricos possíveis vistos em sala. Por si só isto é um grande desafio dadas as limitações do hardware utilizado. Um exemplo claro de que a maior quantidade de elementos teóricos, vistos em sala de aula, e as limitações do hardware escolhido são conflitantes, pode ser visto em exemplos a seguir, da implementação de concorrência, baseada em preempção, que foi o primeiro problema encontrado no seu desenvolvimento.

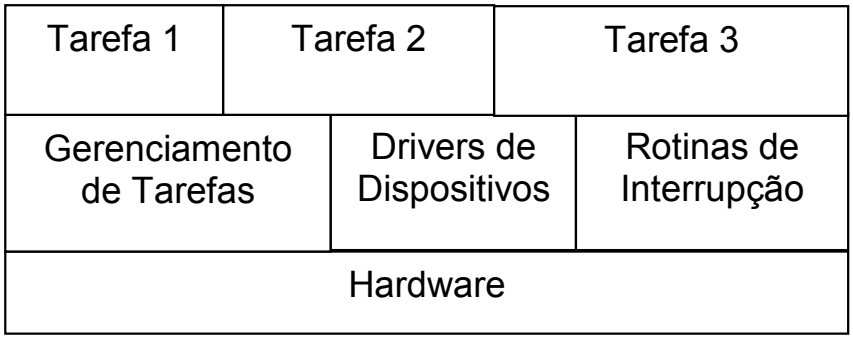

Figura 3 – Diagrama esquemático do micro núcleo Sauim

Núcleos comuns para a família 8051 se baseiam na existência de bancos de memória no controlador. Estes bancos facilitam a tarefa de realizar a troca de contexto entre tarefas, pois, para tal, basta guardar qual é o banco ativo e restaurá-lo de acordo com qual tarefa está sendo colocada como ativa no momento da troca. No 8051 isso pode ser feito por meio de instruções PUSH e POP sobre o registrador de Flags do sistema, algo que deve ser feito de qualquer maneira na troca de contexto. Entretanto, isso significa que o número de tarefas fica limitado ao número de bancos, sendo que a cada tarefa é alocado um banco de registradores. Se o banco zero for destinado às interrupções e os demais bancos (um, dois e três) às tarefas da aplicação propriamente falando, o número destas cai para 3. A troca de contexto, neste caso, é extremamente rápida. É isto que é feito em núcleos de uso comercial da família 8051, muitos deles disponíveis pelas próprias ferramentas de desenvolvimento. De forma geral, este tipo de troca de contexto é o utilizado em microcontroladores baseados em bancos de registradores.

A teoria para a troca de contexto, no entanto, está baseada no salvamento de todo o contexto de uma tarefa (de SOUZA LEÃO, 1996), em geral os registradores em uso na CPU, em uma estrutura chamada de BCT (Bloco de Controle de Tarefas). Também parte do pressuposto que cada tarefa possui a sua pilha própria, cujo ponteiro também é guardado no BCT. No caso do 8051, o contexto seria o conjunto de todos os 4 bancos de memória (bancos 0 a 3), assumindo-se que não há um banco definido para uma tarefa em dado momento. Somente estes bancos daria 32 bytes de armazenamento dos bancos (são 8 registradores R0 a R7 para cada banco) mais o registro de Flags e o acumulador, além, é claro da pilha. Chega-se a marca de 35 bytes por tarefa . Inviável para um sistema que contém somente 252 bytes para a memória de dados para tudo, inclusive pilha. Também a troca de contextos neste cenário é extremamente custosa em termos de tempo.

O problema é criar uma estrutura tipo BCT e realizar a troca de contexto num microcontrolador com bancos de registradores. A solução encontrada foi a mais simples que se poderia utilizar: salva-se o número do banco atual, que está no registrador de Flags (o que é obrigatório em todo caso), e move-se todos os dados dos registros R0 a R7 do banco atual para o banco 0. Depois salva-se na pilha apenas o banco zero. Claro, os demais registros de uso geral também são salvos (Acc e pilha). Após a troca da tarefa ativa, restaura-se o acumulador e a pilha da tarefa destino, os registradores de R0 a R7 no banco 0, e por fim o banco de destino (registro de Flags), e então move-se o conteúdo do banco 0 para o banco de destino. A primeira troca é gerenciada pelo próprio núcleo, de tal forma que não há perda alguma em nenhum momento, em nenhuma tarefa e nem no núcleo dos dados contidos nos registros de troca (R0 a R7 do banco 0). Como somente há um banco ativo por vez, este tipo de troca garante que todo o contexto da tarefa está salvo e que, no retorno do processamento para ela, a situação anterior está completamente estabelecida, salvo, é claro, se outro processo mexer nas áreas de memória reservadas para aquela tarefa. Veja-se a Figura 4 um diagrama de

como ocorre a troca de contexto no Sauim, sem a restrição do uso de um banco de registradores para cada tarefa.

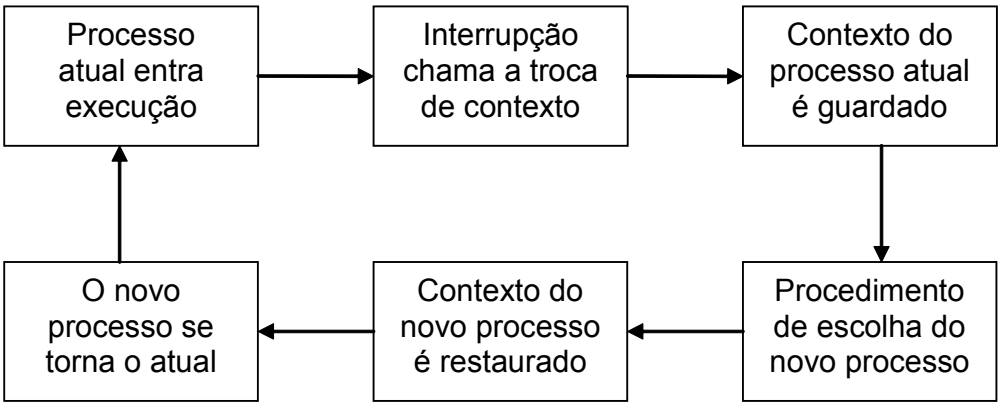

Figura 4 – Troca de contexto no núcleo Sauim

Esta seqüência de comandos é necessária porque o controlador não possui instruções para se guardar diretamente os registradores do banco ativo. E isto era de se esperar, pois ele não foi desenvolvido para ser somente uma ferramenta acadêmica, mas para ser utilizado em aplicações práticas em que a performance é uma variável extremamente relevante para as aplicações.

Com esta técnica, pode-se ter a construção da quantidade que se queira de tarefas, não limitadas, em número, ao número de bancos existentes. Claro que o número de tarefas será sempre limitado a quantidade de memória disponível, mas aqui se está referindo apenas a uma limitação específica do microcontrolador. Também não se tem um custo tão relevante do tempo quando da troca de contexto.

Dessa forma, aproximou-se então a teoria da prática, sem as dependências específicas do microcontrolador e, ao mesmo tempo, estudou-se um tipo muito comum e usado de um microcontrolador e com uma generalização tal que o conhecimento pode ser aplicado a outros tipos de microcontroladores, tanto os que possuem divisão de bancos de memória quanto os que não possuem.

Durante o desenvolvimento, contudo, foram criados algoritmos de troca de contexto baseados apenas no salvamento do registro de Flags, ou seja estudou-se igualmente o método de troca de contexto utilizado comercialmente.

Outro problema encontrado no desenvolvimento do núcleo Sauim foi a questão do agendador. Experimentou-se vários tipos de agendadores e optou-se pelo mais simples, baseado em prioridade única, já descrito na seção 3. Isso possibilitou aos alunos perceberem quais são os tipos de agendador mais apropriados para sistemas de tempo real.

Outro foco de atenção está voltado para o estudo dos vários algoritmos de comunicação interprocesso. A partir do desenvolvimento da troca de contexto e de um agendador de prioridade única, os algoritmos de comunicação entre as tarefas podem ser testados. É neste ponto que a teoria de programação concorrente pode ser mais bem exemplificada a partir do kit didático. Como este possui acesso facilitado a um LCD e a leds diretamente na placa, pode-se construir tarefas que executam concorrentemente e acessam recursos comuns, como áreas de memória ou periféricos (como um led ou LCD) e perceber claramente as condições de corrida ou de bloqueio.

5 *STATUS* ATUAL E TRABALHOS FUTUROS

Por enquanto o núcleo Sauim conta com um gerenciador de tarefas, com troca de contexto por preempção, aproveitando-se de um temporizador do microcontrolador, um agendador por prioridade única e diretivas de sincronização temporal *sleep* e *wakeup*.

O gerenciador de tarefas conta com as chamadas de sistema mostradas na Tabela 2. O agendador funciona com prioridade única. De fato, a prioridade é a identificação da tarefa.

Tabela 2 – Chamadas de Sistema para Gerenciamento de Tarefas

Chamada	***Descrição***
ZCriaTarefa(id, tamPilha, tarefa)	Cria uma tarefa de prioridade e id indicados. A tarefa em si é apenas uma função do sistema
ZMataTarefa()	Apaga a entrada da tarefa em andamento da lista de tarefas do sistema
ZMata(id)	Apaga a entrada da tarefa indicada da lista de tarefas do sistema
ZDorme(id)	Suspende a tarefa indicada (*sleep*)
ZAcorda(id)	Acorda a tarefa indicada (*wakup*)
ZEstado(id)	Retorna o estado da tarefa indicada por id

Falta ainda ser implementado as diretivas de sincronização temporal, isto é, a realização de *wakeups* agendados, ou por outra, *sleeps* com temporização.

Uma versão utilizando-se o 89CS8252 com 1 KByte de RAM está sendo planejada. Na utilização de um microcontrolador com essa capacidade de memória, se projeta a necessidade de um gerenciador de memória para que as tarefas não a utilizem indevidamente. Também a inclusão de um gerenciador de mensagens (diretivas *send* e *receive*) está nos planos para versão 2.0 do micro núcleo Sauim, mas tal somente será levado a efeito após a evolução do hardware com a inclusão de um controlador de rede (e.g. controlador Ethernet).

Não se tem nenhum plano para que ele se torne uma ferramenta comercial. Aliás, algumas características, como a performance, não estão sendo colocadas como prioridade. Desde que não fique algo muito afastado da utilização prática em um dispositivo real, pequenas perdas de performance são toleradas.

Como se trata de um projeto acadêmico a sua evolução está atrelada ao conjunto de fatores do processo ensino/aprendizagem e não há planejamento de se encerrar a evolução deste núcleo. Pretende-se que ele seja a base para novos desenvolvimentos e possibilitar aos alunos uma formação mais completa, permitindo, desde a sua saída da Universidade que já possua em seu currículo a condição para programar dispositivos mínimos em sistemas de tempo real.

Sob o ponto de vista pedagógico, o uso do kit e o desenvolvimento do núcleo apresentam uma oportunidade única tanto para alunos quanto para professores, uma vez que possibilita a ambos o contato com conteúdos avançados da engenharia. É gratificante, do ponto de vista do professor, vê uma turma empenhada na resolução dos problemas apresentados, mesmo que a solução deles requeira algumas vezes a intervenção do professor. Do ponto de vista do aluno também é gratificante a sua participação em projetos deste gênero, uma vez que ele "vê" algo prático surgindo de seu trabalho.

6 CONSIDERAÇÕES FINAIS

O desenvolvimento de núcleos de tempo real necessita tanto de uma base teórica quanto prática para a sua execução. Para tanto, o ensino na Engenharia de disciplinas que visem a programação de tempo real e a programação de dispositivos mínimos podem contar com kits didáticos de microcontroladores, em especial da família 8051 e uma metodologia adequada no desenvolvimento da disciplina, permitindo que alunos e professores alcancem o objetivo maior de uma aprendizagem integral e os primeiros possam usufruir, em seus currículos, de uma bagagem de conhecimento extremamente útil no mundo atual, abrindo-lhes a porta para a pesquisa, o desenvolvimento e a inovação tecnológicas.

Além do projeto Sauim, foram feitos outros projetos com características predominantemente práticas. Durante o curso os alunos envolvidos com os projetos obtiveram um aproveitamento muito bom em termos dos conteúdos apresentados, o que foi medido pelas avaliações efetuadas. Em especial o núcleo Saium foi apresentado para os profissionais do CETELI, alguns deles com vários anos de experiência atuando no desenvolvimento de soluções industriais e outros com vários níveis de especialização. O trabalho foi bastante elogiado o que mostra que o empenho do aluno aliado a uma orientação adequada pode ser a chave para o problema de interseção da teoria e a prática no ensino da engenharia.

A partir destes resultados pretendemos fazer com que estas atividades de laboratório passem a ser parte integrante da disciplina Programação de tempo real.

Agradecimentos

Ao Centro de Pesquisa e Desenvolvimento em Tecnologia Eletrônica e da Informação, CETELI, da Universidade Federal do Amazonas (UFAM) pelo espaço, equipamentos e o kit de desenvolvimento utilizados.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ANDREWS, G.R. and SHNEIDER, F.B. **Concepts and notations for Concurrent Programming**, Computing Surveys, vol. 15, no 1, march 1993.

ATMEL. **8-bit Microcontroller with 8 K Bytes Flash**. AT89S8252 Component Datasheet, 2006. Disponível em http://www.atmel.com/atmel/acrobat/doc0401.pdf. Acessado em 20 maio de 2007

CEGLIA, G.; GUZMAN, V.M.; ORELLANA, C.A.; FERNANDEZ, J.M.; GIMENEZ, M.I.; WALTER, J. Training platform for teaching power electronics Using PIC microcontrollers. In: European Conference on Power Electronics and Applications, Dresden: Alemanha, **Anais**. 2005.

de SOUZA LEÃO, J.L. **Programação e Verificação de Sistemas Multitarefas**. Rio de Janeiro: COPPE/UFRJ, 1996.

DURFEE, W.; LI, P.; WALETZKO, D. Take-home lab kits for system dynamics and controls courses. In: American Control Conference, Boston: USA, **Anais**. 2004.

GREENWALD, L.; KOPENA, J. Mobile robot labs. IEEE Robotics & Automation Magazine, Vol. 10, Issue 2, p 25 – 32, June 2003.

HOROWITZ, B.R.; RECLTENWALD, R.J.; KUTE, P.A.; CLEAVER, T.G. Updating an instructional laboratory for a computer interfacing course: a configuration based on the

M68HC11 microcontroller. In: 29th Frontiers in Education Conference, San Juan: Puerto Rico, **Anais**. 1999.

JAZDI, N.; LUCENA JR., V.F.; GÖHNER, P. UNIBRAL, an educational and research cooperation between Brazil and Germany. In: IEEE Frontiers in Education Conference – San Diego: USA, **Anais**. 2006.

LABROSE, J.J. **MicroC/OS-II The Real-Time Kernel**. CMP Books: Lawrense, Kansas, 1999.

LIMA, H. O. **Manual do Kit AT89S**. Manaus, novembro de 2006. (Documentação que acompanha o kit).

LUCENA JR., V.F.; BRITO, A.; JAZDI, N.; GÖHNER, P. A Germany-Brazil Experience Report on Teaching Software Engineering for Electrical Engineering Undergraduate Students. In: IEEE – CSEET – Conference on Software Engineering Education and Training – Hawaii USA, **Anais**. 2006.

LUCENA JR., V.F.; BRITO, A. Uma Metodologia Germano-Brasileira Para o Ensino de Engenharia de Software para Alunos de Engenharia Elétrica. In: COBENGE 2006 – Passo Fundo – RS, **Anais**. 2006.

MARTINEZ, D.; BOLTON, R.; ZOGHI, B. Embedded real-time controls through application. In: 29th Frontiers in Education Conference, San Juan: Puerto Rico, **Anais**. 1999.

TANENBAUM, A. and WOODHULL, Albert. **Sistemas Operacionais – Projeto e Implementação**. Tradução Edison Furmankieewicz. 2. ed. Porto Alegre: Bookman, 2000.

VALLINO, J.R.; CZERNIKOWSKI, R.S. Work in progress – multi-disciplinary real-time and embedded systems laboratory and course sequence. In: 34 th Frontiers in Education Conference, Savannah: USA, **Anais**. 2004.

VALLINO, J.R.; CZERNIKOWSKI, R.S. Thinking inside the box: a multi-disciplinary real-time and embedded systems course sequence. In: 34 th Frontiers in Education Conference, Savannah: USA, **Anais**. 2005.

WICK, C.E. Teaching embedded computer systems with a Windows-based simulator. In: 26 th Frontiers in Education Conference, Salt Lake City: USA, **Anais**. 1996.

DEVELOPMENT OF A REAL-TIME MICRO KERNEL FOR THE 89C8252 MICROCONTROLLER FOR THE TEACHING OF REAL-TIME PROGRAMMING TECHNIQUES

Abstract: *The development of micro-kernels and operating systems for platforms based on 8051 microcontroller family is a very popular issue, mainly because of the great quantities of these devices available in the market. This fact can be easily proved by a simple search in the internet. Nevertheless, the teaching of disciplines related to operating systems, and particularly related to real-time systems, using microcontrollers face limitations that leads to difficulties in understanding the whole related theory. This paper is about one educational experiment using of those micro-kernel systems constructed upon the 8051 microcontroller that solved some of common problems. The main idea was to construct a mini system covering the main theoretical topics about real-time programming using a cheap and poplar platform. The experiment descript here was done in the course of Real Time Programming at the Electrical Engineering and Computer Engineering major of the Universidade Federal do Amazonas in Manaus..*

Key-words: *Microcontroller Laboratories Experiences, Teaching of Programming for Mini Systems, Real-Time Programming, Teaching of Real-Time Operating Systems.*