Sistema Operativo de Tempo Real para ATmega baseado em microkernel preemptivo

Rui Graça (201004124), Eduardo Almeida (201000641), Tiago Costa (200601289)

Relatório

***Resumo*—O produto do trabalho realizado é um sistema operativo de tempo real (RTOS) baseado num microkernel preemtivo, tendo como tecnologia alvo o microcontrolador ATmega328P.**

O sistema desenvolvido implementa stacks separadas para cada tarefa e escalonamento baseado em prioridades fixas, sendo que a API oferece funções como temporizadores, sinalização para sincronização entre tarefas e semáforos, baseados em SRP, assim como a possibilidade de lançar novas tarefas durante a execução do sistema.

I. Introdução

OS requisitos temporais a que estão sujeitos sistemas de tempo real levam à necessidade da utilização de sistemas operativos adequados, que permitam um comportamento totalmente determinístico, em que pode ser feita, de forma simples, uma análise do cumprimento dos requisitos temporais.

O trabalho aqui apresentado tem como produto final um sistema operativo que se enquadra neste contexto, possibilitando ao programador de sistemas de tempo real garantias de um comportamento determinístico e uma interface (API) com funcionalidades básicas para a implementação de um sistema de tempo real.

O sistema desenvolvido é um sistema com preempção, o que significa que existe uma interrupção periódica (tick) que verifica qual é a tarefa de maior prioridade pronta a executar e atribui-lhe o CPU. Cada tarefa representa uma thread de funcionamento do sistema, à qual é atribuída uma stack independente, que mantém o seu contexto entre as suas diversas ativações. Para isto, é necessário que, na troca de contexto, o stack pointer seja devidamente alterado.

O lançamento de novas tarefas pode ser feito tanto na inicialização do sistema como durante a execução, permitindo a uma tarefa lançar uma nova tarefa. Não existe, no entanto, nenhuma relação hierárquica entre tarefas criadoras e criadas, nem nenhuma relação especial entre elas.

Na API do sistema operativo são fornecidas funções que possibilitam a criação de temporizadores periódicos, que podem ser locais às tarefas que os criam ou globais, sendo que, neste caso, várias tarefas podem esperar por um mesmo temporizador. Após a criação de um temporizador, uma tarefa pode invocar uma função que espera que seja ativada por esse temporizador, colocando a tarefa num estado inativo e adicionando a tarefa à lista de espera do temporizador, que é esvaziada no tick do sistema, ativando todas as tarefas nela existentes.

A API possibilita, também, que uma tarefa fique inativa e seja acordada após um tempo determinado. Esta funcionalidade não é mais que a criação temporária de um temporizador.

Outra funcionalidade de extrema importância em sistemas com preempção são os semáforos, que controlam o acesso a regiões críticas, impedindo que a sequência de execução de tarefas tenha consequências imprevistas e indesejadas, as chamadas race conditions. Em sistemas de tempo real, dado que o escalonamento é baseado em prioridades, a utilização de semáforos como implementados noutros sistemas é problemática, dado que causa inversão de prioridade não limitada, que acontece quando uma tarefa de alta prioridade pode ficar

ilimitadamente bloqueada porque necessita que uma tarefa de mais baixa prioridade liberte um semáforo, sendo que esta tarefa de mais baixa prioridade não poderá executar enquanto houver tarefas de prioridade intermédia a ocupar o CPU. Surge, desta forma, a necessidade de mecanismos de controlo de race conditions que limitem ao mínimo a inversão de prioridade. Estes mecanismos passam, em geral, por uma herança temporária de prioridade, em que a tarefa que tem um semáforo (no qual está bloqueada uma tarefa de prioridade superior) fica temporáriamente com uma prioridade superior à original.

O mecanismo utilizado no sistema apresentado baseia-se em Stack Resource Policy (SRP), implementando semáforos binários (mutexes). De acordo com este protocolo, existe um teto do sistema (System Ceiling), que corresponde à prioridade mais alta entre todas as tarefas que utilizam os semáforos utilizados num dado momento. Por exemplo, se um semáforo é o único trancado num dado momento, e é utilizado por duas tarefas, T_1 com prioridade 1, e T_2 com prioridade 5, sempre que este semáforo está trancado o teto do sistema é 5, mesmo que seja T_1 a trancar o semáforo. Em cada tick, de acordo com SRP, só há uma mudança de contexto no caso de haver uma tarefa pronta a executar com prioridade superior à tarefa que tem o CPU nesse instante e com prioridade superior ao teto do sistema. Este mecanismo tem propriedades bastante boas, dado que, ao mesmo tempo que limita a inversão de prioridade, garante que uma tarefa, após começar a executar, nunca é bloqueada, o que reduz as mudanças de contexto.

O trabalho tem como objetivo a aplicação de conceitos abordados em diversas unidades curriculares do MIEEC, onde foram introduzidos sem uma componente prática que possibilite compreender de que forma é que eles são aplicados num contexto de implementação real.