Implantação de conceitos de sistemas de tempo real em uma envasadora utilizando arduíno

João Paulo Cunha Ávila

Departamento de Ciências da Computação – Centro Universitário de Brasília
jpcunha4@gmail.com

Abstract:

Resumo:

1. **Introdução**
2. **Referencial Teórico**

Segundo Buttazzo (2011), sistemas ou softwares de tempo real são aqueles que devem executar suas funções de maneira que atendam a requisitos temporais de maneira precisa. Benveniste e Berry (1991) explicitam também que sistemas de tempo real são sistemas reativos, ou seja, respondem à eventos do mundo externo, que tem restrições de tempo previamente definidas. Shaw (2003) afirma que quando um sistema desse tipo é parte de um outro maior que está embutido, chamamos tal computador componente de sistema embarcado.

Estas restrições de tempo são representadas por deadlines, que nada mais são do que limites de tempo para as tarefas serem executadas. Tendo isto em vista, é possível classificar estes deadlines como rigorosos e tolerantes. Deadlines rigorosos são aqueles que tem de cumprir todos os requisitos de tempo que lhe foram impostos e, caso isto não ocorra, o sistema falha. Deadlines tolerantes possuem também seus requisitos de tempo mas alguns podem falhar sem muitos problemas para a execução da próxima tarefa (SHAW, 2003).

Apesar de tarefas terem deadlines a serem cumpridos, Lehoczky e Sprunt (1989) expõem três diferentes classificações em relação à periodicidade de execução das mesmas. São elas tarefas periódicas, aperiódicas e esporádicas. Tarefas periódicas são executadas entre intervalos de tempo definidos, sendo que seu período é igual ao seu deadline. Tarefas aperiódicas são aquelas que não tem um período definido, ou seja, são executadas em intervalos irregulares. Tarefas esporádicas são tarefas aperiódicas que possuem deadlines rigorosos.

Shaw (2003) explica que nenhum sistema é perfeitamente confiável logo, existirão falhas que podem ter um custo alto. Beveniste e Berry (1991), explicam que estas falhas tem relação à corretudes lógicas, que tem relação a bugs de software, e temporais, restrições de tempo não atendidas. Portanto, é importante evitar estas falhas o máximo possível e criar uma forma de tratar falhas que venha a ocorrer efetivamente e com o menor custo (SHAW, 2003).

Levando em consideração que o tempo é tratado pelo processador um recurso, faz-se necessário utilizar o escalonamento de processos, que refere-se a capacidade de um processador em executar todas as computações, ou seja, se o mesmo tem clocks suficientes para executar tais processos à tempo. Para alcançar este propósito são utilizados diversos tipos de algoritmos (SHAW, 2003). Para este experimento faz-se

necessário a análise de apenas três algoritmos: por taxa monotônica, EDF(Earliest Deadline First) e Deadline Monotônico.

O escalonamento por taxa monotônica trabalha com o princípio que tarefas com o menor período tem maior prioridade (BRANICKY; PHILLIPS; ZHANG, 2002). O algoritmo EDF trabalha com o princípio de que a tarefa com o deadline mais próximo tem maior prioridade de execução (SHAW, 2003). Por fim, o algoritmo de Deadline MonoTônico trabalha com o princípio de que a prioridade é definida pelo menor deadline, ou seja, quanto menor o deadline maior a prioridade (AUDSLEY, 1991).

Um dos maiores problemas do escalonamento é o de concorrência entre threads de um mesmo processo que compartilham a mesma região crítica, causando desta maneira a condição de corrida. Existem diversas maneiras de solucionar este impasse, uma delas é pelo uso de semáforos (TANENBAUM, 2009).

Semáforos atuam como variáveis de controle de regiões críticas. Seu funcionamento é baseado em valores "down" e "up" que vão, respetivamente, colocar um processo no seu estado de bloqueado e liberar o acesso à região crítica por esse processo. Isso já é suficiente para evitar a condição de corrida e resolver possíveis problemas de sincronização (TANENBAUM, 2009).

3. Desenvolvimento

O projeto foi desenvolvido na linguagem C, tendo o Arduíno Uno como responsável pela implementação do sistema embarcado. Foi utilizada a biblioteca NilRTOS para trabalhar com conceitos de tempo real no projeto.

O mesmo consiste no controle de uma envasadora de garrafas. Duas esteiras contendo as garrafas moverão em direção a uma terceira esteira onde ocorrerá o processo de preenchimento com um líquido. Assim que a garrafa estiver cheia a máquina encerra o processo e move a esteira, liberando espaço para que uma próxima garrafa possa acessa-la.

As duas esteiras que contém as garrafas vazias serão tratadas como duas threads diferentes que tentarão acessar a região crítica, que é representada pela terceira esteira onde as garrafas serão preenchidas. O controle de acesso a terceira esteira é feito a partir de um semáforo que decidirá qual das esteiras, número um ou dois, ira mover.

O autômato finito abaixo representa a máquina de estados do sistema sugerido:

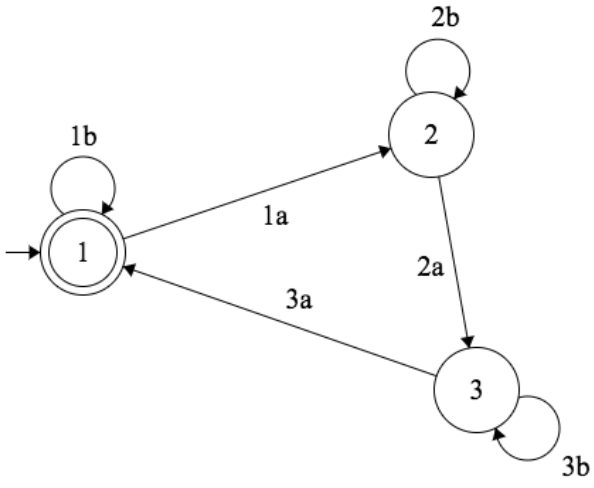

Figura 1. Maquina de Estados de uma Envasadora

Como é possível observar, a máquina possui três estados de funcionamento. O primeiro estado é responsável pelo movimento da esteira que tiver acesso à região crítica e pela verificação da chegada da garrafa até a máquina por um sensor. Caso o sensor detecte a presença de uma garrafa (1a) o próximo estado será ativado, caso contrário (1b) a atividade do estado atual continua normalmente. Ao chegar no segundo estado a garrafa será preenchida com o líquido e será monitorada por um sensor de nível que, quando ativado (2a), cessa sua atividade e ativa o próximo estado, caso contrário (2b) continua o preenchimento. O terceiro estado é responsável pela retirada da garrafa que está acessando a máquina, pelo movimento da terceira esteira, e liberação do semáforo. Quando o semáforo é bloqueado novamente (3a) inicia-se novamente o primeiro estado, caso isso não ocorra (3b) a esteira continua em movimentação.

A execução das tarefas propostas são feitas de maneira periódica e são escalonáveis utilizando o algoritmo de taxa monotônica, como apresentado matematicamente abaixo:

(Aqui conterá todo o cáculo que prova que o algoritmo é escalonável).

4. **Resultados**
5. **Conclusão**

Referências

G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. SHAW, Alan C. **Sistemas e software de tempo real**. Bookman, 2003.

BUTTAZZO, Giorgio. **Hard real-time computing systems: predictable scheduling algorithms and applications**. Springer Science & Business Media, 2011.

SPRUNT, Brinkley; SHA, Lui; LEHOCZKY, John. Aperiodic task scheduling for hard-real-time systems. **Real-Time Systems**, v. 1, n. 1, p. 27-60, 1989.

TANENBAUM, Andrew. **Modern operating systems**. 2009.

BRANICKY, Michael S.; PHILLIPS, Stephen M.; ZHANG, Wei. Scheduling and feedback co-design for networked control systems. In: **Decision and Control, 2002, Proceedings of the 41st IEEE Conference on**. IEEE, 2002. p. 1211-1217.

AUDSLEY, Neil C. et al. Real-time scheduling: the deadline-monotonic approach. In: **in Proc. IEEE Workshop on Real-Time Operating Systems and Software**. 1991.

BENVENISTE, Albert; BERRY, Gérard. The synchronous approach to reactive and real-time systems. **Proceedings of the IEEE**, v. 79, n. 9, p. 1270-1282, 1991.