Análise de Estratégias de Escalonamento em Sistemas de Tempo Real Utilizando o Zephyr RTOS

Luciano dos Santos Nascimento[1], **Wesley Silva Araújo**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista- RR- Brasil

{wesleysilvaaraujo19, luciano.santos.nasciment}@gmail.com

Abstract. *This work presents the implementation and analysis of scheduling strategies in real-time systems using the Zephyr RTOS. The study was conducted with three concurrent tasks representing sensors, keyboard, and display, each with different priorities and execution frequencies. The scheduling policies analyzed include preemptive, cooperative, fixed-priority, and Round Robin, evaluating their behavior under various execution scenarios. The simulation was performed using the QEMU emulator, and the results demonstrate how the choice of policy impacts the responsiveness and predictability of the embedded system.*
Keywords: Zephyr RTOS, scheduling, real-time systems, QEMU, threads.

Resumo. *Este trabalho apresenta a implementação e análise de estratégias de escalonamento em sistemas de tempo real utilizando o Zephyr RTOS. O estudo foi conduzido com três tarefas concorrentes representando sensores, teclado e display, cada uma com diferentes prioridades e frequências. Foram analisadas as políticas de escalonamento preemptiva, cooperativa, prioridade fixa e Round Robin, avaliando seus comportamentos sob diferentes cenários de execução. A simulação foi realizada com o emulador QEMU, e os resultados demonstram como a escolha da política impacta a responsividade e previsibilidade do sistema embarcado.*
Palavras-chave: Zephyr RTOS, escalonamento, sistemas de tempo real, QEMU, threads.

1. Introdução

Sistemas de tempo real exigem que tarefas críticas sejam executadas dentro de restrições temporais rígidas, garantindo previsibilidade e resposta determinística. Esses sistemas estão presentes em aplicações embarcadas de grande relevância, como dispositivos médicos, sistemas automotivos, aeronáutica e automação industrial, onde atrasos ou falhas no agendamento de tarefas podem comprometer a integridade do sistema.

Para garantir esse comportamento, é fundamental a utilização de estratégias eficientes de escalonamento de tarefas. O Zephyr RTOS — um sistema operacional de tempo real moderno, modular e open source — oferece suporte a diferentes políticas de escalonamento, como preemptiva, cooperativa, prioridade fixa e Round Robin. Com isso, desenvolvedores podem adaptar o kernel para atender às necessidades específicas da aplicação embarcada.

Neste trabalho, realizamos a implementação de um sistema embarcado simulado com três tarefas concorrentes, cada uma representando uma funcionalidade típica: leitura de sensores, comunicação com teclado e atualização de display. Essas tarefas possuem prioridades distintas e diferentes períodos de execução, permitindo observar o comportamento do sistema sob diferentes estratégias de escalonamento fornecidas pelo Zephyr.

As simulações foram realizadas por meio do emulador QEMU, o que viabilizou a experimentação em um ambiente controlado. Os resultados obtidos permitem analisar as implicações práticas de cada política de escalonamento quanto à preempção, regularidade de execução e impacto na latência das tarefas. A proposta visa promover uma compreensão mais aprofundada dos mecanismos internos do Zephyr RTOS e de sua aplicabilidade em sistemas críticos de tempo real.

2. Fundamentação Teórica

Sistemas de tempo real são definidos como aqueles em que a correção das operações depende não apenas da lógica computacional, mas também da temporalidade com que os resultados são produzidos. Em outras palavras, um sistema de tempo real deve não apenas fornecer resultados corretos, mas também garantir que esses resultados sejam entregues dentro de prazos específicos. Tais sistemas estão presentes em aplicações como controle de tráfego aéreo, dispositivos médicos e sistemas automotivos.

Uma característica fundamental desses sistemas é o escalonamento de tarefas, ou seja, a política que determina a ordem e o momento em que cada tarefa será executada. Um bom escalonador deve garantir que tarefas críticas sejam priorizadas e que o sistema se mantenha responsivo, mesmo sob carga. O comportamento do sistema depende diretamente da política de escalonamento adotada, podendo impactar na latência, throughput e previsibilidade. Entre as estratégias clássicas de escalonamento, destacam-se:

- **Preemptiva:** permite a interrupção de uma tarefa em execução caso uma tarefa de maior prioridade esteja pronta para ser executada. Essa abordagem é comum em sistemas de tempo real por possibilitar respostas rápidas a eventos críticos.
- **Cooperativa:** a troca de contexto entre tarefas ocorre apenas quando a tarefa em execução explicitamente cede o controle, por exemplo, por meio de chamadas como k_yield(). Esse modelo reduz a sobrecarga de preempção, mas exige que as tarefas sejam bem comportadas.
- **Prioridade fixa:** cada tarefa é atribuída a uma prioridade estática, usada como critério principal para o escalonamento. Tarefas de menor prioridade podem sofrer de starvation se tarefas de prioridade superior estiverem constantemente prontas.
- **Round Robin:** distribui o tempo de CPU entre tarefas de mesma prioridade de forma equitativa, por meio de fatias de tempo (time slices). Esse modelo promove justiça na execução, evitando que uma única tarefa monopolize a CPU.

O **Zephyr RTOS** é um sistema operacional de tempo real de código aberto que suporta múltiplas políticas de escalonamento, oferecendo ao desenvolvedor controle fino sobre o comportamento do kernel. Sua arquitetura baseada em microkernel e seu suporte a múltiplos dispositivos embarcados o tornam adequado tanto para simulações em QEMU quanto para aplicações reais com microcontroladores como ESP32 e STM32. O Zephyr permite a definição de tarefas concorrentes por meio de **threads**, as quais podem ser configuradas com diferentes prioridades e comportamentos. A API fornece mecanismos

como k_msleep(), k_yield() e K_THREAD_DEFINE() que facilitam a criação e controle dessas tarefas. O modelo de escalonamento padrão é o de prioridade fixa preemptiva, mas pode ser modificado em tempo de compilação para incluir o modo cooperativo ou Round Robin.

Entender o comportamento dessas estratégias e sua influência sobre tarefas com diferentes frequências de execução é essencial para o desenvolvimento de sistemas embarcados confiáveis e eficientes.

3. Metodologia

A metodologia adotada neste trabalho consistiu no desenvolvimento de uma aplicação embarcada simulada, utilizando o sistema operacional Zephyr RTOS e o emulador QEMU, com o objetivo de observar e comparar o comportamento de diferentes políticas de escalonamento em um ambiente de tempo real. A aplicação foi composta por três tarefas concorrentes, modeladas como threads independentes:

- **Thread 1 – Sensor:** executada a cada 500 ms, com prioridade alta. Simula a leitura de um sensor crítico.
- **Thread 2 – Teclado:** executada a cada 1000 ms, com prioridade média. Representa a leitura de entrada do usuário.
- **Thread 3 - Display:** executada a cada 2000 ms, com prioridade baixa. Responsável por exibir dados na saída.

As threads foram definidas utilizando a macro K_THREAD_DEFINE() do Zephyr, com parâmetros que especificam a pilha, prioridade e comportamento de execução. O tempo de espera entre execuções foi configurado com a função k_msleep(). Foram testadas três estratégias de escalonamento:

1. **Preemptiva com prioridade fixa (default)**
2. **Cooperativa** , com uso explícito de k_msleep() para liberação de CPU
3. **Round Robin,** ativado via configuração no arquivo prj.conf com a diretiva CONFIG_TIMESLICING=y

Para cada modo de escalonamento, foram observadas as mensagens de log geradas por printk(), analisando a ordem e o tempo de execução de cada thread. As simulações foram realizadas com o QEMU, sem a necessidade de hardware real, o que permitiu reprodutibilidade e controle das variáveis experimentais.

A coleta de dados se baseou na observação dos tempos de execução e preempção, frequência de alternância entre tarefas e o impacto de bloqueios ou atrasos, permitindo comparações qualitativas entre as políticas adotadas.

4. Resultados e Discussão

Os testes realizados com a aplicação simulada permitiram observar o comportamento do sistema em diferentes modos de escalonamento do Zephyr RTOS. A análise foi conduzida com base nos registros de saída gerados pelas funções printk(), que indicam o momento e a ordem de execução de cada thread.

4.1. Escalonamento Preemptivo com Prioridade Fixa

No modo padrão do Zephyr (preemptivo com prioridade fixa), a execução da thread de maior prioridade (sensor) prevaleceu sobre as demais. Sempre que a tarefa do sensor estava pronta para execução, ela interrompia as tarefas de menor prioridade. Como resultado, a thread de menor prioridade (display) executou com menos frequência, e em alguns momentos teve sua execução adiada indefinidamente. Esse comportamento evidencia a possibilidade de inanição de tarefas, especialmente aquelas com baixa prioridade.

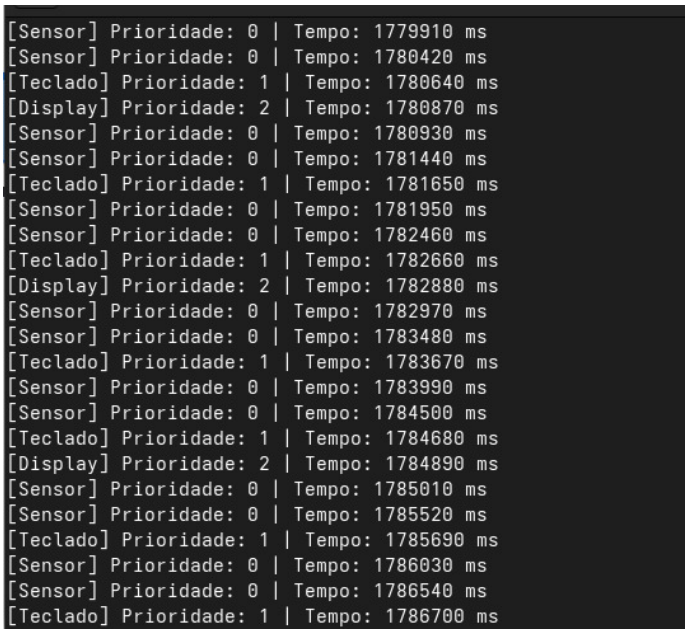

Figure 1. Teste de Tempo Preemptivo.

4.2. Escalonamento Cooperativo

No modo cooperativo, nenhuma tarefa foi preemptada. Cada thread permaneceu em execução até que chamasse explicitamente k_yield() ou entrasse em estado de espera (k_msleep()). Esse modelo exigiu que as tarefas fossem bem comportadas. Quando isso foi respeitado, a execução foi ordenada e previsível, porém a latência da resposta a eventos mais urgentes (como os do sensor) aumentou, comprometendo a responsividade do sistema.

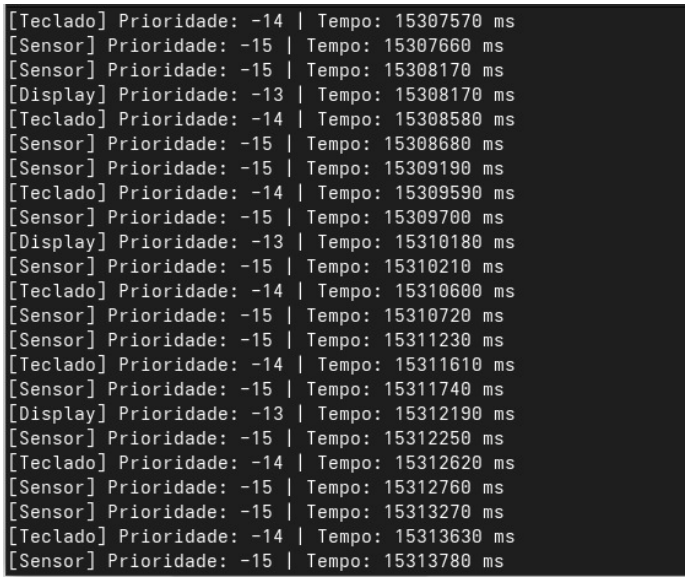

Figure 2. Teste de Tempo Cooperativo.

4.3. Round Robin

Com o escalonamento do tipo Round Robin ativado e todas as threads configuradas com a mesma prioridade, o tempo de CPU foi dividido de forma equitativa entre as tarefas. Observou-se alternância periódica e regular entre as execuções, o que promoveu justiça entre as threads. Contudo, a ausência de diferenciação por prioridade comprometeu a execução de tarefas críticas, como a do sensor, que não conseguiu garantir respostas imediatas em todos os ciclos.

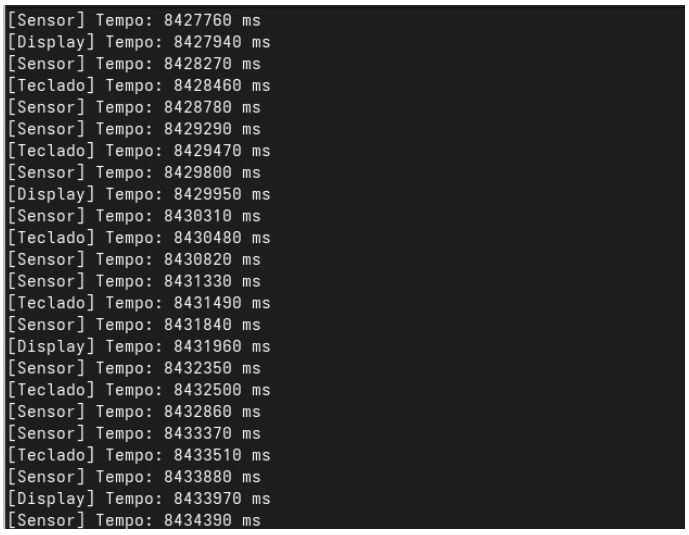

Figure 3. Teste de Tempo Round Robin.

5. Considerações

A escolha da política de escalonamento influencia diretamente na responsividade e previsibilidade do sistema. Enquanto o modelo preemptivo com prioridade fixa garante maior

atenção às tarefas críticas, pode causar bloqueios às demais. Já o Round Robin favorece equidade, mas reduz a prioridade das tarefas urgentes. O modo cooperativo, por sua vez, oferece simplicidade e baixo overhead, mas depende fortemente do bom comportamento das tarefas.

6. Conclusão

Este trabalho apresentou uma análise prática das principais estratégias de escalonamento de tarefas em sistemas de tempo real, utilizando o Zephyr RTOS como plataforma de desenvolvimento e o QEMU como ambiente de simulação. Através da implementação de três tarefas concorrentes com diferentes prioridades e frequências de execução, foi possível observar de forma clara os impactos de cada política sobre o comportamento do sistema.

Os resultados demonstraram que o escalonamento preemptivo com prioridade fixa favorece a execução de tarefas críticas, mas pode causar inanição das tarefas menos prioritárias. O modo cooperativo, embora previsível e com baixo overhead, depende do bom comportamento das tarefas, podendo comprometer a responsividade do sistema. O Round Robin ofereceu maior justiça na distribuição do tempo de CPU, mas sacrificou a execução imediata de tarefas urgentes. A combinação de Round Robin com prioridades distintas apresentou um equilíbrio promissor entre responsividade e equidade.

Conclui-se que a escolha da estratégia de escalonamento deve considerar as características específicas da aplicação, os requisitos temporais das tarefas e a criticidade de cada operação. O Zephyr RTOS se mostrou uma ferramenta eficiente para simulação e estudo de sistemas embarcados em tempo real, oferecendo flexibilidade na configuração do kernel e suporte a múltiplos cenários de execução.

Como trabalhos futuros, propõe-se a extensão da aplicação para cenários com número maior de tarefas, uso de recursos de comunicação entre threads, bem como a análise de desempenho utilizando ferramentas como o SystemView para visualização temporal mais precisa das execuções.

7. Referências

[Tanenbaum and Bos 2015, Silberschatz et al. 2018, Zephyr Project ,
Nordic Semiconductor , Mullins 2020, QEMU]

References

Mullins, C. (2020). Understanding real-time operating systems. `https://www.embedded.com/understanding-real-time-operating-systems/`.

Nordic Semiconductor. Zephyr kernel overview. `https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/zephyr/kernel/index.html`.

QEMU. Qemu emulator documentation. `https://www.qemu.org/documentation/`.

Silberschatz, A., Galvin, P. B., and Gagne, G. (2018). *Operating System Concepts*. Wiley, 10 edition.

Tanenbaum, A. S. and Bos, H. (2015). *Modern Operating Systems*. Pearson, 4 edition.

Zephyr Project. Zephyr rtos documentation. https://docs.zephyrproject.org/latest/.