**IFPE - Instituto Federal de Pernambuco**
**Sistemas Embarcados(Exe5)**
**Prof. Anderson Moreira**
**Aluno: César Augusto B. Gonçalves**

# Java e sistemas de Tempo Real: vale a pena usar?

## INTRODUÇÃO

Java inicialmente foi criada para a utilização em sistemas embarcados. Porém com o tempo e a vasta popularização se tornou uma ferramenta importante para desenvolvimento desktop e web.

Com a linguagem vastamente difundida e bastante utilizada e o atual crescimento no uso de dispositivos móveis e portáteis, surge o interesse em se utilizar Java para sistemas embarcados e tempo-real. Assim surgem os questionamentos: Seria ideal utilizar uma linguagem interpretada e baseada em máquina virtual para sistemas de tempo-real? Teríamos vantagens ou desvantagens nisto?

## DESAFIOS EM DESENVOLVIMENTO DE SISTEMAS TEMPO REAL

Existem vários problemas bem específicos a tratar em sistemas de tempo-real.

– **Resposta em tempo real**: O sistema reagirá a estímulos e informações externas. Ou seja, interagirá com o mundo real para realizar suas ações. E deve ser pensado de forma a estar preparado para tais estímulos;

– **Tolerância a falhas**: O sistema dever ser o mais tolerante a falhas possível, e livre de interrupções. Não se tem a possibilidade de disparar uma caixa de diálogo quando algum erro acontece ou ser reiniciado quando acontecer algo crítico. O sistema deve estar preparado para receber e contornar problemas sem interromper o processamento e suas ações;

– **Trabalhar com arquiteturas distribuídas**: Muitos desses sistemas podem trabalhar com multiprocessamento e para isto é necessário suporte para manutenção da consistência dos dados, tratamento de interfaces, compartilhamento de recursos, dentre outros pontos;

– **Timming**: A questão principal dos sistemas de tempo-real. O timming é tudo. Limites de tempo, processamento, reações etc. tudo deve ter seu tempo controlado e estar sincronizado.

## JAVA

As facilidades e produtividade que desenvolver um sistema em tempo-real em Java trariam seria um dos motivos para se escolher esta linguagem. A perspectiva de reaproveitamento de projetos, da experiência de profissionais da área em programação para diversas plataformas de processadores e sistemas operacionais na area de desenvolvimento para tempo-real é bastante bem vinda, seja devido à dificuldade de se treinar novos profissionais nesta área, seja pelos custos de adaptar um sistema já existente para uma nova plataforma.

Outro forte motivo para esta escolha seria ainda a ampla aceitação, acadêmica e comercial, que atualmente goza a plataforma Java, usufruindo de ampla literatura, e número de pessoas familiarizadas na linguagem.

Porém estes fatores impactam com certas características da linguagem que trariam problemas para se tentar desenvolver sistemas de tempo-real. Entre eles tem-se:

– **Garbage collector**: Esta funcionalidade do Java a princípio pode parecer ser uma boa ferramenta para um sistema em tempo-real. Afinal, liberaria recursos de memória – muitas vezes escassos – para o sistema. Porém pode acontecer, por exemplo, de sua custosa execução entrar em ação no momento em que uma tarefa crítica do sistema tivesse que ser realizada. Esses tipo de comportamento é inaceitável.
– **Threads** : Java tem um modelo inserido para concorrência, que simplifica a programação concorrente porém a definição destas funcionalidades são muito vagas para sistemas de tempo-real. O comportamento das threads é definido de forma muito fraca, permitindo por exemplo, que processos de menor prioridade preemptem outros de maior prioridade ou mesmo uma implementação sem preempção é permitida. Isto protege as threads de problemas como starvation mas não é aceitável em sistemas de tempo real.
– **Controle e acesso de memória**: Java traz restrições no acesso a memória. E certo desperdício quando por exemplo, faz alocação dinâmica de classes, a resolução e verificação de classes é uma função complexa, consome memória e seu tempo de execução é praticamente impossível de prever.
– **Outros problemas**: sincronização de recursos, tamanho das bibliotecas, etc. Portanto a especificação Java original, apesar dos benefícios na hora de codificar, não seria nada aprovável para desenvolver aplicações de tempo real.

## UMA ESPECIFICAÇÃO JAVA PARA TEMPO REAL

Tendo em vista os problemas acima descritos, o sucesso e aceitação da linguagem e a proposta inicial do Java que era para sistemas embarcados, foi criado um grupo de peritos para estudar o melhoramento de Java para sistemas de tempo real. O RTJEG (Real-Time for Java Expert Group) publicou em novembro de 2001 a especificação de Java para Tempo Real.

– **Real Time Specification for Java (RTSJ)**. A RTSJ é um conjunto de especificações de comportamento para a linguagem Java para permitir que a mesma possua comportamentos determinísticos, essenciais no desenvolvimento de sistemas de tempo real. A sua aprovação ocorreu em 2002, e a primeira implementação comercial em 2003. Algumas de suas maiores inovações são: acesso direto à memória física, *threads* de tempo real, e um novo modelo de gerência de memória.
O acesso direto à memória se dá através da classe *RawMemory*. Ela permite o acesso de dados inteiros e de ponto flutuante em endereços específicos do programa. Áreas da memória física podem ser definidas com determinadas características (por exemplo, RAM estática) e usadas para alocação de objetos.
A solução da IBM é a *WebSphere Real Time*, uma Máquina Virtual de tempo real (*real-time* VMJ9) estendida com a tecnologia *Metronome* de coleta de lixo em tempo real, compilação *Ahead of Time*, e total suporte à RTSJ, tudo isso sendo executado em um sistema operacional Linux de tempo real em desenvolvimento. Essa solução visa atacar uma das maiores causas de indeterminismo em Java, a coleta de lixo.

A RTSJ traz algumas modificações para tempo real:
– **Threads e escalonamento**: Um escalonador preemptivo básico, é definido, com 28 prioridades. Podendo outros escalonadores (como EDF) serem carregados dinamicamente.
Novas classes de threads são definidas: RealtimeThread acessa todas as regiões de memória e é afetada pelo coletor de lixo; NoHeapRealtimeThread só acessa a memória do escopo e pode sempre preemptar o coletor de lixo(nunca é atrasada por ele); AsyncEventHandlers para tratar eventos, pode ter prioridade maior que o coletor. Todas elas acabam sendo eventos escalonáveis.
– **Memória**: Como o Garbage collector é problemático em aplicações de tempo real, a RSTJ define novas regiões de memória. Immortal memory é a memória compartilhada por todas as threads, sem coletor de lixo e permanece até o término do programa. Scooped memory memória onde os objetos são alocados até que a última thread a "deixe". Phisycal memory usada para controlar a alocação em memórias com tempo de acesso diferentes. Raw memory para acesso de memória a nível de byte e mapeamentos de I/O.
– **Sincronização**: Herança de prioridade para controlar inversão de prioridades.
– **Eventos assíncronos**: para conectar com os Asynchronous Event Handlers. Os eventos podem ser disparados arbitrariamente por um método fire(), por temporizadores, por sinais ou por ocorrências.
– **Transferência de controle assíncrona**: exceções de interrupção podem ser lançadas por cada thread em uma thread específica. É segura, pode ser desabilitada (por default) ou ativada em determinada thread.

## CONCLUSÃO

A princípio podemos perceber que a padronização de Java para tempo real acaba com alguns dos problemas que Java trazia para este tipo de aplicação.
A partir do lançamento da RTSJ, desenvolver sistemas de tempo real utilizando a linguagem Java tornou-se uma realidade. Desenvolvedores agora possuem uma nova alternativa atraente a ser considerada para os projetos de sistemas de tempo real. Portanto podemos concluir através desta pesquisa que Java para tempo real pode ser uma boa solução, os estudos mostram que a RTSJ é suficiente para desenvolver aplicações de tempo real. Porém a confiabilidade fica ainda um pouco em dúvida.
E a transição e adaptação do Java comum para o Java tempo real pode ser tão complexa e demorada para os desenvolvedores que fica em cheque toda a questão da portabilidade, produtividade e facilidade que Java traria para um projeto de sistema tempo real.
Fazendo com que a escolha entre métodos e linguagens tradicionais para muitos casos ainda seja a mais sensata.

## REFERÊNCIAS


### Java e sistemas de Tempo Real: vale a pena usar?

https://pc2008evermat.wordpress.com/2008/07/02/java-e-sistemas-de-tempo-real-vale-a-pena-usar/

### Java e Sistemas de Tempo Real: Vale a pena utilizar?

https://determinismotemporal.wordpress.com/2008/06/10/java-e-sistemas-de-tempo-real-vale-a-pena-utilizar/