UDESC – UNIVERSIDADE DO ESTADO DE SANTA CATARINA

CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMAÇÃO DE SISTEMAS DE TEMPO REAL

TAREFA 4

LUCAS CORDEIRO BUTZKE

JOINVILLE/SC

2020/01

**Tarefa de Simulação**

1. Para o conjunto de tarefas dados os seguintes tempos de resposta são mostrados na Tabela 1 para um escalonador EDF. Numa tarefa perdeu seu prazo e o *job* aperiódico também executou sem maiores complicações.

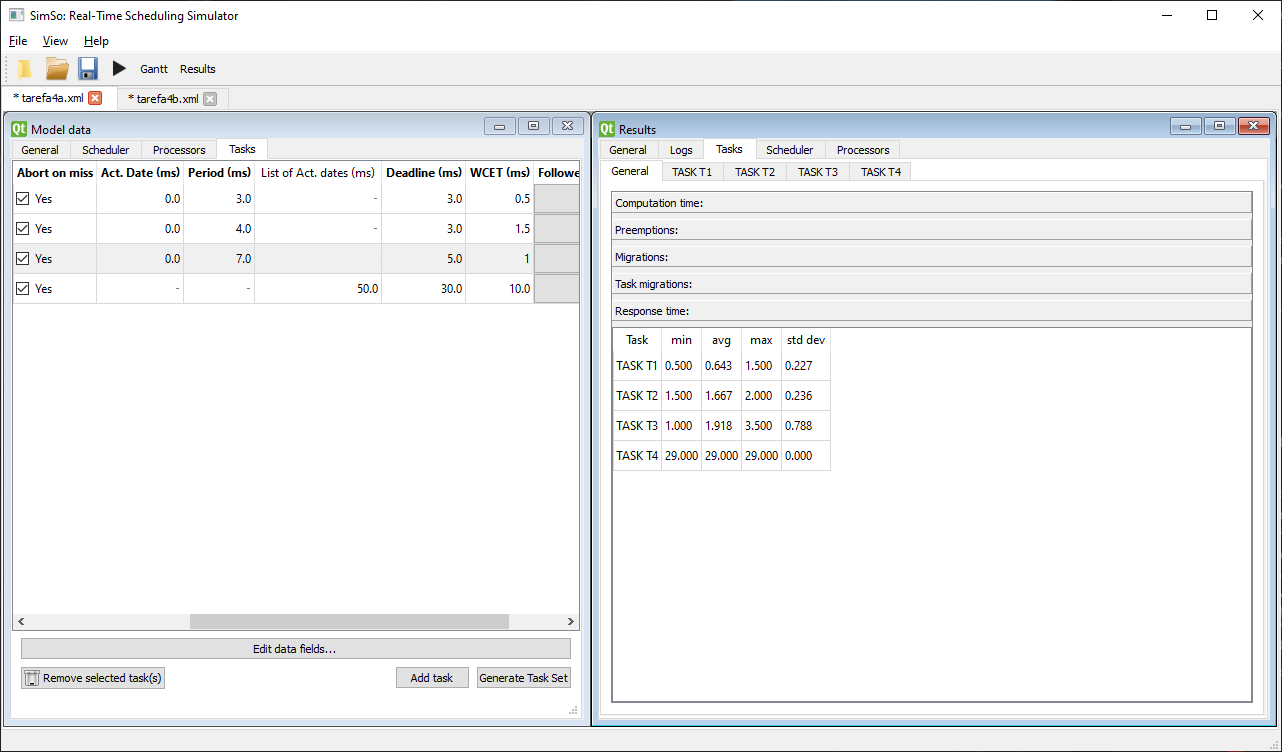


Tabela 1 - Tabela do tempo de resposta das tarefas.

Com o *job* aperiódico introduzido dentro desse sistema, a execução deste tarefa é mostrada abaixo, mesmo com um tempo de resposta quase atingindo o *deadline* de execução, todos critérios são atendidos.

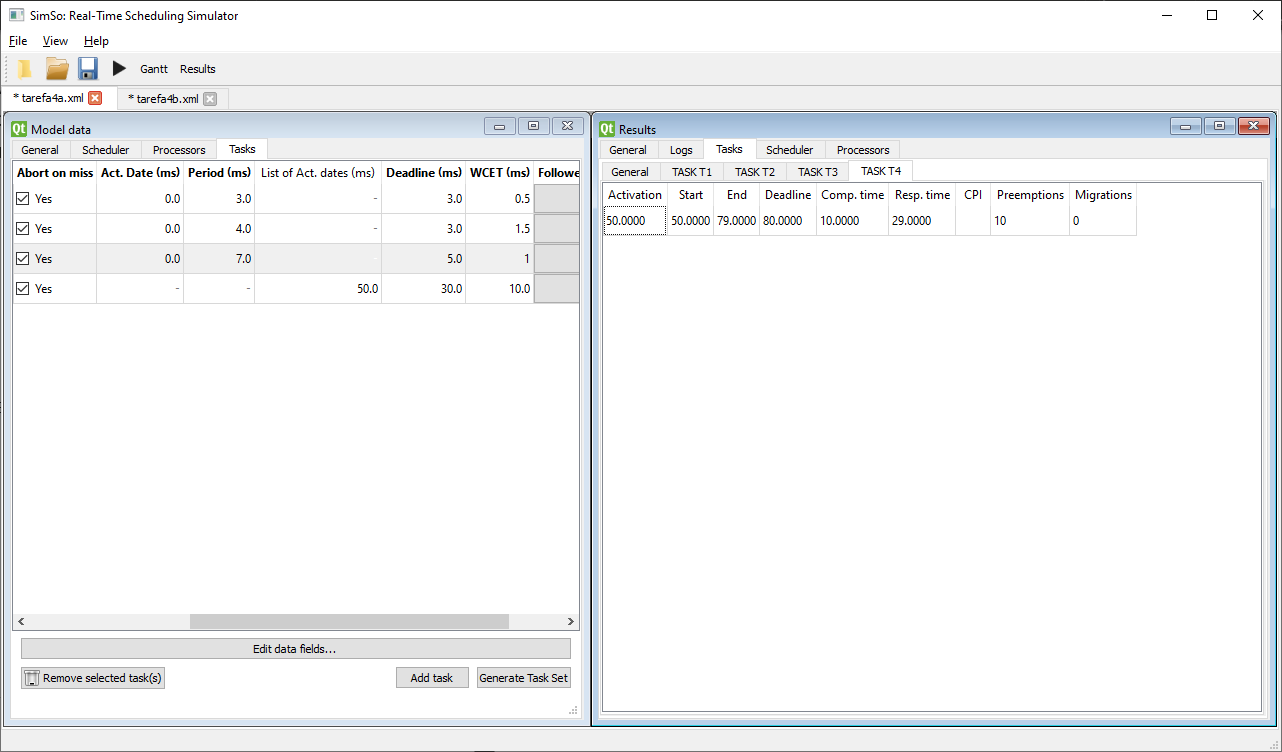


Tabela 2

1. Para o conjunto de tarefas dados os seguintes tempos de resposta são mostrados na Tabela 3 para um escalonador RM. Numa tarefa perdeu seu prazo e o *job* aperiódico perdeu seu prazo e foi abortado.

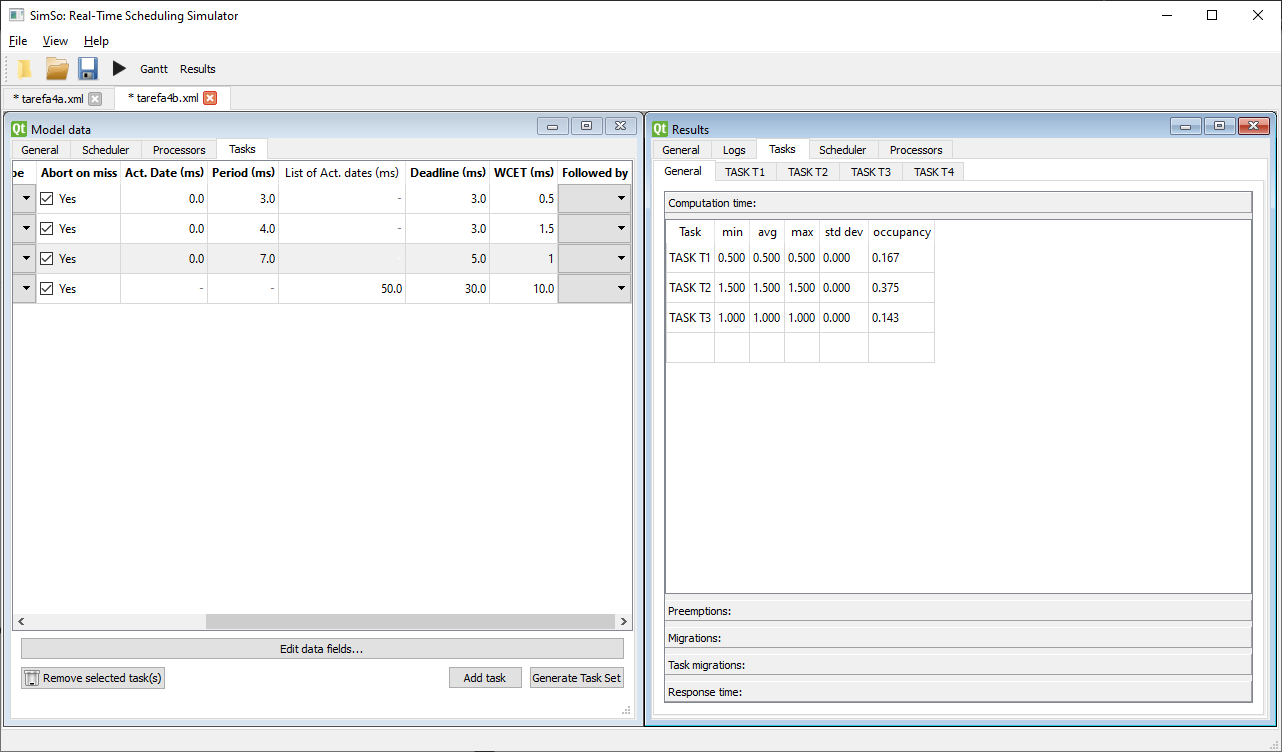


Tabela 3

A execução da tarefa aperiódica é mostrada abaixo, o escalonador RM não consegue preemptar a tarefa de modo que consiga executar dentro do tempo. Logo, um escalonador com prioridades dinâmicas fornece um melhor desempenho para o conjunto de tarefas.

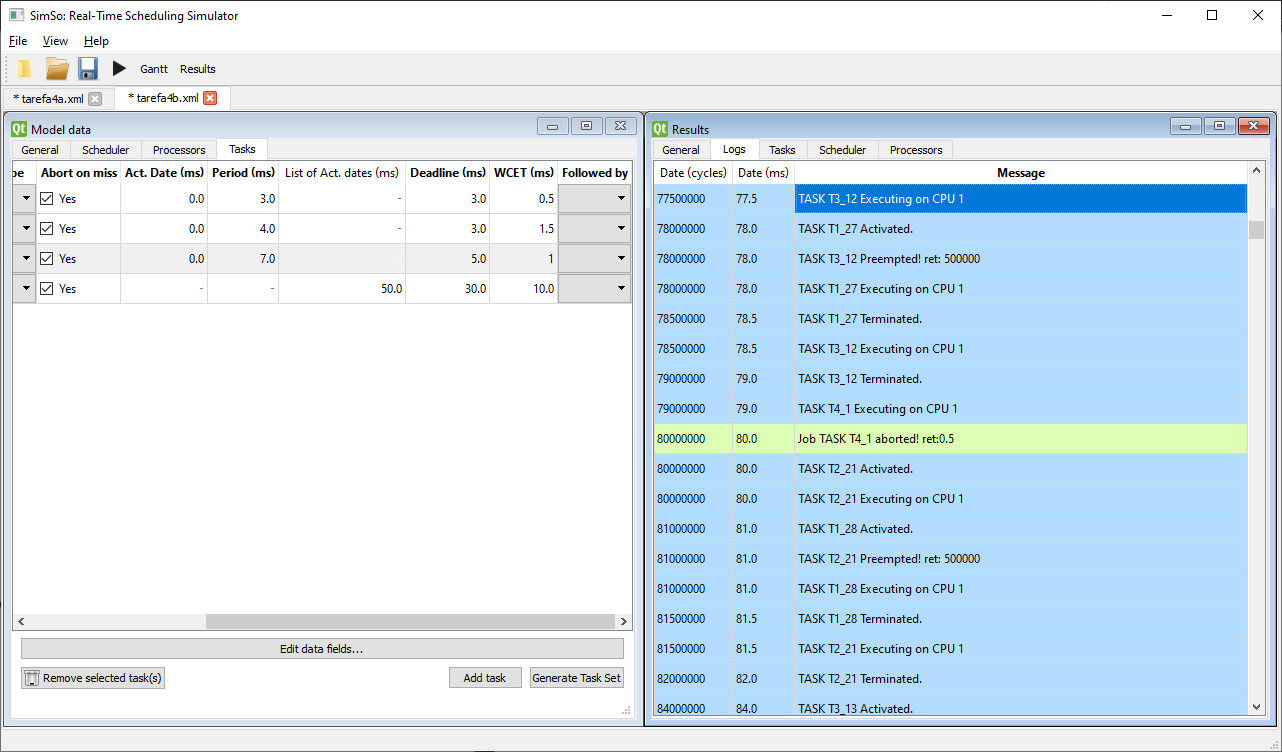


Tabela 4

**Tarefa de Programação**

Para as prioridades fornecidas pela tarefa o sistema não é rápido o suficiente para executar as tarefas antes do término do tempo do *timer*, como mostrado na Figura 1. Durante a execução da tarefa “*matrix\_task*” o sistema é completamente utilizado e não permite que a tarefa aperiódica termine antes do tempo do *timer*, criando inúmeras cópias na fila de execução do processador.

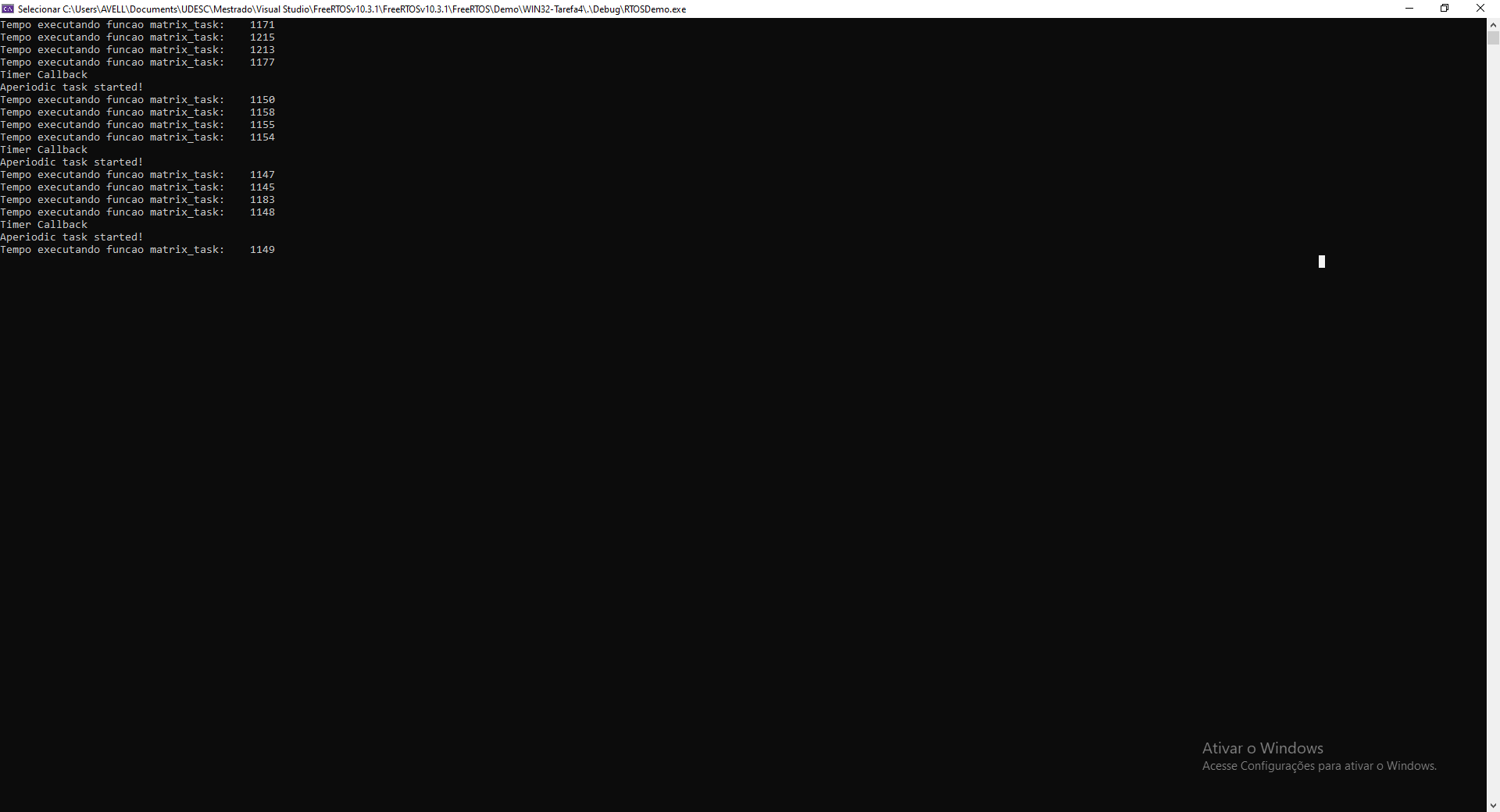


Figura 1 – Saída sem redefinição de prioridades

As soluções mais apropriadas seriam a inclusão de um gerenciador de prioridades, um sistema de suspenção de tarefas para o sistema ou aumentar o período do *timer* para um tempo maior que o tempo de execução da função aperiódica. A Figura 2 mostra a execução do sistema chamando apenas uma única vez a tarefa aperiódica, ela leva um tempo quase 5 vezes o período do *timer* para executar.

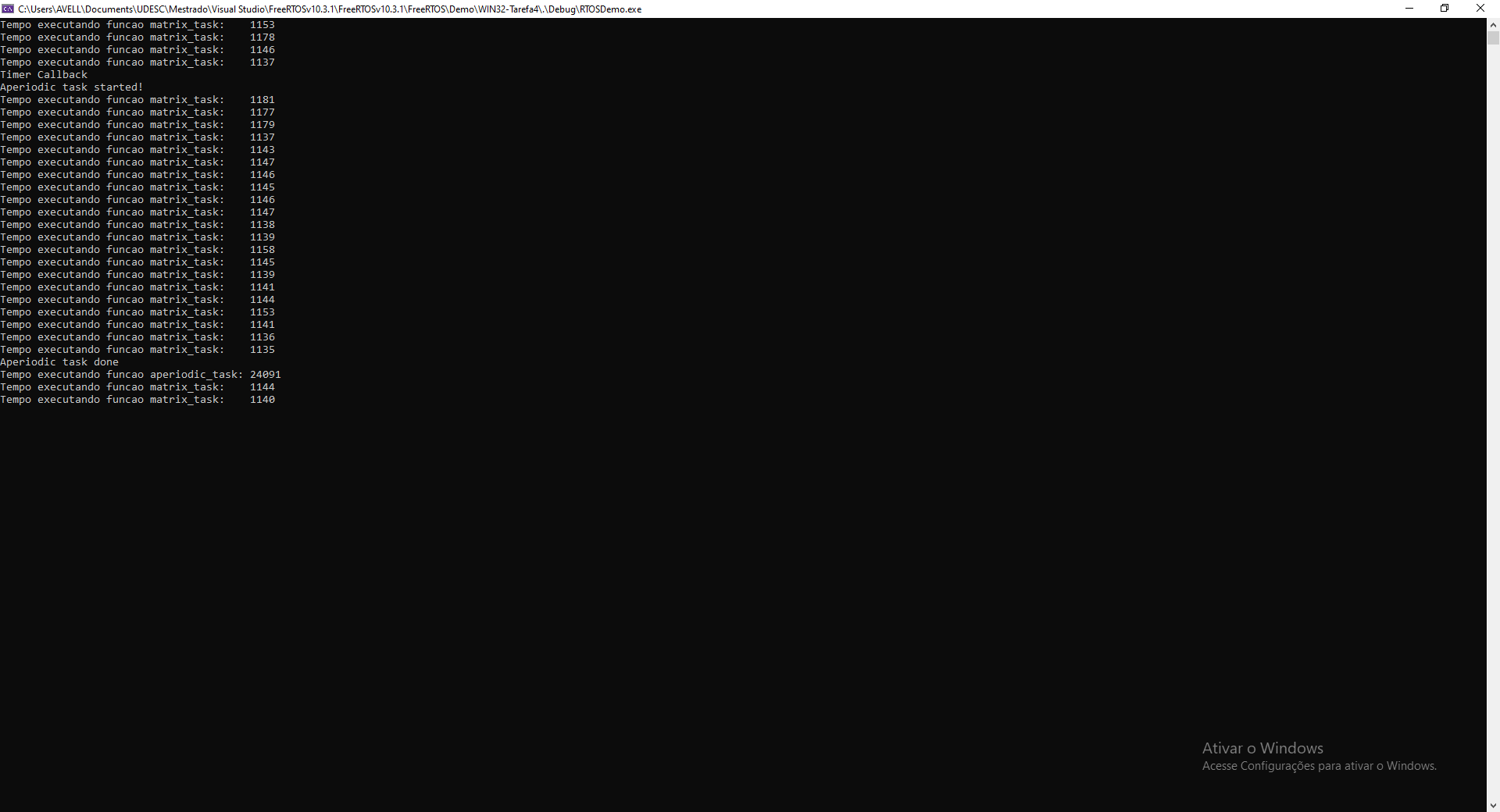


Figura 2 - Execução tarefa aperiódica.

Utilizando a função “*priority\_set*” para alterar as prioridades das tarefas com a interrupção gerada pelo *timer* torna o sistema adequado para executar as tarefas dentro de cada ciclo, como mostrado na Figura 3. Assim, o sistema realiza uma execução que alterna entre as tarefas sem perder o prazo de execução ou criando inúmeras tarefas na fila FIFO sem elas poderem entrar em execução. O *Deferrable Server* é uma ótima opção para escalonar tarefas aperiódicas ou esporádicas pois fornece uma “janela” de execução entre as tarefas e que são executadas apenas quando o “orçamento” é suficiente.

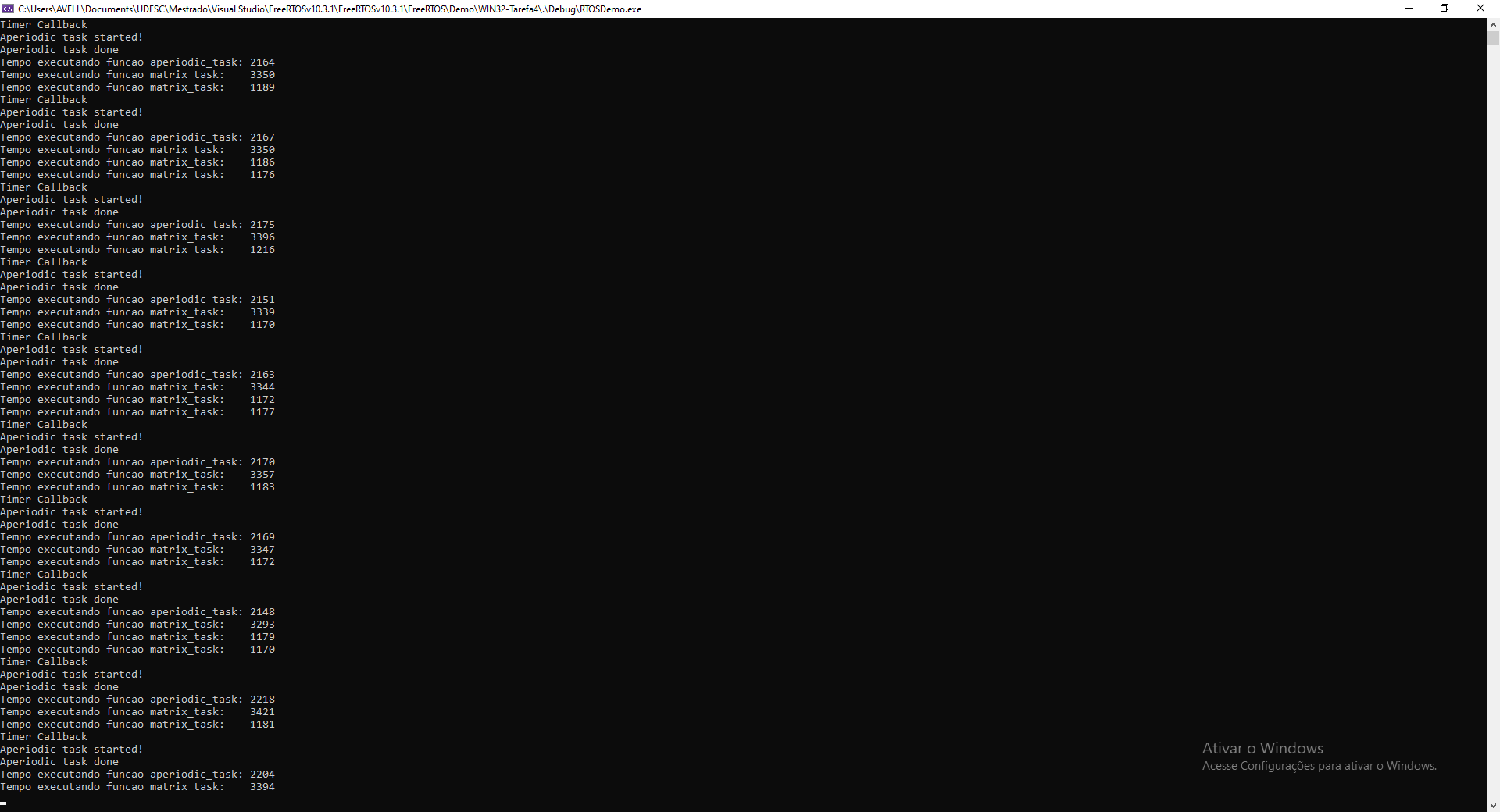


Figura 3 - Execução com gerenciamento de prioridades.

**Conclusão**

As tarefas aperiódicas geralmente possuem prioridades menores em relação as tarefas periódicas dentro de um sistema de tempo real mas, dependendo do período das tarefas aperiódicas ou até os seus *deadlines*, o processador poderá não ter capacidade de executar todas instruções dentro do tempo requerido. Um gerenciador de prioridades é desejável para tornar viável um conjunto de tarefas caso não tiver folgas dentro da fila de processamento necessárias para executar todos os *Jobs*, desse modo podemos criar espaços dentro dos *frames* para assegurarmos que o *Job* terminará antes do próximo começar a sua execução.

Escalonador RM necessita que se desenvolva um projeto de folga dentro dos *frames* para preemptar as tarefas esporádicas, caso não houver até a próxima, outra tarefa esporádica entrará na fila e o processador perderá capacidade de processamento em cada ciclo ou a tareafa será rejeitada.

Para conseguir escalonar as tarefas com menor quantidade de folga entre os *frames* utilizamos o escalonador EDF. Alterando a prioridade dentro da fila de processamento é possível remanejar as necessidades dentro do prazo das tarefas e assim criando menor sobrecarga na fila do processador.

*Deferrable Server* serve muito bem para o propósito deste trabalho, criando um orçamento de execução para as tarefas não agendadas fornece um meio para execução de processos que antes seriam rejeitados pelo processador. Futuramente o orçamento de execução poderá ser feito com *Message Buffers* ou *Queue Send* dentro das próprias funções, assim se obterá controle total dos processos com semáforos e bloqueios.