Universidade do Vale do Itajaí - Univali

Engenharia da Computação
Sistemas em Tempo Real

Alunos: Victor Menezes Ferreira e Gabriel Tourinho Laurentino
Professor: Felipe Viel

1 - Introdução

O presente trabalho tem como objetivo o desenvolvimento e análise de um sistema em tempo real implementado em uma plataforma ESP32 utilizando o sistema operacional FreeRTOS. A temática escolhida corresponde à simulação de uma esteira industrial, na qual se integram diferentes tarefas com restrições temporais rígidas (hard real-time) e brandas (soft real-time). Nesse contexto, são exploradas funções como a leitura periódica de um encoder, o controle de velocidade da esteira, a detecção e triagem de objetos, bem como a parada de emergência e a interação com uma interface de supervisão simples.

A motivação central é compreender como os sistemas em tempo real se comportam diante de diferentes políticas de escalonamento e configurações de execução, avaliando a previsibilidade temporal das tarefas críticas e a robustez da aplicação frente a variações de carga e frequência de CPU. Para tanto, o sistema desenvolvido foi instrumentado de modo a permitir a coleta de métricas de desempenho, como tempos de resposta, latência e ocorrência de perdas de prazo (*deadline misses*).

Essa abordagem permite não apenas a validação prática dos conceitos estudados em sala de aula, mas também a análise comparativa entre as demandas funcionais do sistema e sua implementação em um ambiente embarcado de baixo custo. Assim, o trabalho contribui para a consolidação do aprendizado sobre sistemas de tempo real, fornecendo subsídios para futuras aplicações em contextos industriais e acadêmicos.

2 - Enunciado da esteira

O trabalho consiste na modelagem e implementação, em ambiente ESP32 com FreeRTOS, de um sistema em tempo real que emula uma linha de produção industrial. O sistema deve reproduzir as principais funcionalidades de uma esteira, contemplando controle de velocidade, detecção e triagem de peças e mecanismos de segurança. Para padronizar os estímulos externos, utilizam-se sensores *touch* da própria placa como substitutos dos sensores reais, mapeados da seguinte forma:

- **Touch B**: detecção de objeto na esteira (evento de triagem);
- **Touch C**: solicitação de HMI/telemetria (evento de interface);
- **Touch D**: parada de emergência (*E-stop*)

O sistema integra uma tarefa periódica de amostragem/estimativa de velocidade com tarefas encadeadas e dirigidas a eventos, cada uma com deadlines específicas. As tarefas críticas (controle e segurança) devem obedecer a restrições rígidas de tempo (hard real-time), enquanto funções auxiliares, como interface de telemetria, podem ser tratadas como de tempo brando (soft real-time).

A análise do sistema deve considerar diferentes políticas de atribuição de prioridades (Rate Monotonic, Deadline Monotonic e uma política customizada que privilegie a segurança), bem como variações do escalonador (preemptivo/cooperativo, com ou sem *time slicing*) e da frequência de CPU (80, 160 e 240 MHz). A coleta de resultados envolve a medição de jitter, latência, *deadline misses* e utilização de CPU, a fim de avaliar a previsibilidade temporal e a confiabilidade do sistema desenvolvido.

3 - Metodologia

A implementação do sistema em tempo real foi realizada na plataforma ESP32, utilizando o sistema operacional FreeRTOS. O código foi desenvolvido integralmente em linguagem C.

3.1 - Estrutura de tarefas

O sistema foi estruturado em múltiplas *tasks* independentes, cada uma responsável por uma função específica da esteira industrial:

- **ENC_SENSE (periódica)**: realiza a estimativa de velocidade (RPM) e posição a partir de um encoder simulado;
- **SPD_CTRL (encadeada)**: executa o controle PI da velocidade, além de processar solicitações de HMI em trecho não crítico;
- **SORT_ACT (evento Touch B)**: aciona o atuador de desvio sempre que um objeto é detectado;
- **SAFETY_TASK (evento Touch D)**: implementa a parada de emergência, zerando a velocidade da esteira e sinalizando alarme;
- **TOUCH_POLL (polling dos sensores)**: faz a leitura dos sensores *touch*, define limiares (baseline + threshold) e gera eventos para as demais tarefas;
- **UART_CMD (auxiliar)**: converte comandos do terminal serial em eventos equivalentes aos *touchs*, permitindo testes sem interação física;
- **STATS (periódica)**: registra e imprime métricas de tempo real, incluindo pior caso de execução (Cmax), latência (Lmax), resposta total (Rmax) e número de *deadline misses*.

3.2 - Comunicação e instrumentação

Para coordenar as interações entre as tarefas, foram empregados mecanismos de comunicação e sincronização do FreeRTOS, tais como filas (*queues*), semáforos binários e notificações diretas entre tarefas. Além disso, funções de instrumentação baseadas no esp_timer_get_time() foram utilizadas para mensuração dos tempos de liberação, início e término de cada tarefa, viabilizando a análise temporal do sistema.

Os experimentos consistiram na execução de cenários controlados, nos quais se variaram as políticas de prioridade (Rate Monotonic, Deadline Monotonic e customizada), o tipo de escalonador (preemptivo ou cooperativo) e a frequência de operação do processador (80 MHz, 160 MHz e 240 MHz). Em cada cenário foram coletados dados referentes a *deadline misses*, latência dos eventos de *touch* e jitter. Esses resultados foram posteriormente organizados em tabelas comparativas para embasar a análise crítica do desempenho do sistema.

3.3 - Políticas de prioridade

Foram avaliadas três abordagens: Rate Monotonic (RM), Deadline Monotonic (DM) e uma política Customizada, que prioriza explicitamente a segurança.

- **RM:** prioridade definida pelo período da tarefa (menor período = maior prioridade). ENC_SENSE (T=5 ms) foi a mais prioritária, seguida por SPD_CTRL; tarefas baseadas em eventos (SORT_ACT e SAFETY_TASK) ficaram em nível intermediário; STATS recebeu a menor prioridade.

- **DM:** prioridade definida pelo menor *deadline* absoluto. SAFETY_TASK (D=5 ms) foi a mais prioritária, seguida por ENC_SENSE (D=5 ms), SPD_CTRL (D=10 ms) e SORT_ACT (D=10 ms); STATS permaneceu com menor prioridade.
- **Custom:** priorizou diretamente a segurança, colocando SAFETY_TASK no topo, seguida por ENC_SENSE e SPD_CTRL; SORT_ACT em nível intermediário; STATS com menor prioridade.

3.4 - Configuração dos experimentos

Os experimentos consistiram na execução de cenários controlados, variando-se as políticas de prioridade (RM, DM e Custom), o tipo de escalonador (preemptivo ou cooperativo) e a frequência de operação do processador (80 MHz, 160 MHz e 240 MHz). Cada cenário foi executado por 60 segundos, sendo coletados dados referentes a hard misses, soft misses, latência da tarefa SAFE e jitter. Os resultados foram organizados em tabelas comparativas para embasar a análise crítica do desempenho do sistema.

Tabela comparativa de Prioridades

Tarefa	**RM**	**DM**	**Custom**
ENC_SENSE	Máxima	Alta	Alta
SPD_CTRL	Alta	Média	Média
SORT_ACT	Média	Média	Média
SAFETY_TASK	Média	Máxima	Máxima
STATS	Mínima	Mínima	Mínima

Legenda:
Máxima = prioridade mais alta (preempta todas as outras).
Alta = segunda prioridade, tarefas críticas mas não absolutas.
Média = importante, mas pode esperar as críticas.
Mínima = menor esforço, executa apenas quando sobra CPU.

4 - Resultados

Os experimentos foram realizados com a plataforma ESP32 configurada em modo unicore, sob três políticas de atribuição de prioridade (Rate Monotonic – RM, Deadline Monotonic – DM e Custom) e em diferentes frequências de operação do processador (80, 160 e 240 MHz). Cada cenário foi executado por aproximadamente 60 segundos, com coleta das métricas de hard misses, soft misses e latência do E-stop (SAFE). Foram avaliados dois modos de escalonamento: preemptivo e não-preemptivo (cooperativo).

4.1 - Resultados no modo preemptivo

No modo preemptivo, observou-se que todas as tarefas hard real-time (ENC_SENSE, SPD_CTRL, SORT_ACT e SAFETY_TASK) cumpriram seus deadlines em todos os cenários, apresentando hard misses iguais a zero. A tarefa de HMI, classificada como soft real-time, também não registrou violações de deadline, resultando em soft misses iguais a zero.

A métrica de latência máxima do E-stop (SAFE), obtida a partir de Lmax, manteve-se sempre abaixo de 1 ms, valor muito inferior ao limite de 5 ms estabelecido. Esse comportamento comprova a previsibilidade e a confiabilidade do sistema em condições preemptivas.

Tabela de resultados preemptivo

Política	**Frequência**	**Hard misses**	**Soft misses**	**Latência E-stop**
RM	80Mhz	0	0	0.28ms
RM	160MHz	0	0	0.15ms
RM	240MHz	0	0	0.11ms
DM	80Mhz	0	0	0.07ms
DM	160MHz	0	0	0.04ms
DM	240MHz	0	0	0.03ms
Custom	80Mhz	0	0	0.05ms
Custom	160MHz	0	0	0.05ms
Custom	240MHz	0	0	0.02ms

4.2 - Resultados no modo não-preemptivo

No modo cooperativo, os resultados foram diferentes. Apesar de, na maioria dos cenários, as tarefas terem respeitado seus deadlines, foram registrados casos de hard miss: a tarefa SORT_ACT em RM/80 MHz e a tarefa SAFE no mesmo cenário não concluíram dentro do tempo esperado. Além disso, as latências do E-stop chegaram a valores acima de 8 ms, ultrapassando o limite de 5 ms e caracterizando falhas críticas.

Mesmo nos cenários em que não houve perda de deadlines, a latência da tarefa SAFE foi sistematicamente maior do que no modo preemptivo, evidenciando a vulnerabilidade do escalonamento cooperativo. A HMI continuou sem apresentar soft misses, mas com tempos de resposta variáveis.

Tabela de resultados não preemptivo

Política	**Frequência**	**Hard misses**	**Soft misses**	**Latência E-stop**
RM	80Mhz	2	0	8.31ms
RM	160MHz	0	0	0.84ms
RM	240MHz	0	0	0.99ms
DM	80Mhz	0	0	0.23ms

DM	160MHz	0	0	1.71ms
DM	240MHz	0	0	0.09ms
Custom	80Mhz	0	0	1.28ms
Custom	160MHz	0	0	0.58ms
Custom	240MHz	0	0	1.28ms

4.3 - Análise dos resultados

De forma geral, os testes comprovaram que o sistema é determinístico e seguro quando operando em modo preemptivo, pois todas as tarefas críticas respeitaram os deadlines e a tarefa SAFE apresentou latência de resposta confortável em todos os cenários. Já no modo não-preemptivo, ainda que parte dos experimentos tenha sido bem-sucedida, ocorreram perdas de deadline em situações específicas, além de latências elevadas, o que compromete a previsibilidade do sistema em funções críticas de segurança.

5 - Conclusão

Os experimentos realizados demonstraram que o sistema atende aos requisitos de tempo real no modo preemptivo, sem ocorrência de hard ou soft misses e com latência do E-stop sempre inferior a 1 ms, garantindo resposta confiável às funções críticas. Já no modo não-preemptivo, verificaram-se perdas de deadline e latências elevadas em cenários específicos, evidenciando menor previsibilidade e maior risco para a tarefa de segurança. Assim, conclui-se que o uso do escalonamento preemptivo, aliado a políticas que privilegiam a segurança, é essencial para assegurar a confiabilidade do sistema.

6 - Link do projeto

GitHub_Esteira