Universidade do Vale do Itajaí
Curso: Engenharia de Computação
Disciplina: Sistemas em Tempo Real
Professor: Felipe Viel
Avaliação M2 – Latência, WCRT e Medições

Instruções

- Deverá ser usada as linguagens **C/C++** para o desenvolvimento.
- Não será permitido o uso de bibliotecas ou funções prontas e disponibilizadas em base de dados ou em outros meios, salvo situações expressas no trabalho.
- Os códigos desenvolvidos devem ser postados e estarem funcionais. Caso a aplicação não esteja correta, será atribuída nota proporcional ao trabalho.
- O trabalho poderá ser feito sozinho, em dupla ou trio.
- O(s) aluno(s) deverá(ão) apresentar o trabalho presencialmente. Caso seja requisitado, deverão dar esclarecimentos adicionais. O não comparecimento a explicação, mesmo a adicional, implicará em nota máxima (proporcional ao desenvolvido) 5,0.
- Trabalhos copiados implicará na nota zero para todos os envolvidos.
- Não serão aceitos trabalhos entregues em atraso.
- As aplicações desenvolvidas deverão exibir resultados corretos para qualquer caso de teste possível dentro do especificado pelo enunciado.
- O código fonte deverá estar comentado para auxiliar o entendimento. A postagem deverá ser feita no github. O usuário do professor é: VielF.
- A responsabilidade pela demonstração do código funcionando é do(s) aluno(s).
- Deverá ser entregue (postado) um relatório em formato PDF contendo: a) Identificação do autor e do trabalho; b) enunciado de cada projeto; c) imagens, tabelas e/ou quadros demonstrando os resultados obtidos; d) explicação dos resultados obtidos; e e) descrição das técnicas utilizadas. Poderá ser usado modelo de artigo, usando o template IEEE (versão A4) ou ABNT.
- O trabalho deverá ser postado por um dos alunos no material didático.
- O trabalho deverá ser postado no material didático até as 8h00 do dia 30/10/2025. Trabalhos entregues em atraso implicará em desconto de 2,0 por dia, incluindo feriados e finais de semana.
    - Apresentação, descrição do problema, utilização correta do português, profundidade de análise a partir das questões elencadas, diagramação (uso de imagens e diagrama para explicar);
  - Codificação totalmente funcional e atendimento aos requisitos.
  - Qualidade da apresentação do trabalho (presencialmente). Compilação do software no momento da apresentação é obrigatório.
  - Pontualidade.

Baseado na temática escolhida e utilizando códigos e análises extraídas no trabalho da M1, aplique as modificações e realize as novas análises solicitadas.

Temática 1 - Drone

O contexto dessa temática implementa, em uma ESP32 com FreeRTOS, um autopiloto didático que reproduz o ciclo essencial de um drone: fusão sensorial periódica, controle de atitude, navegação e rotinas de segurança. Para tornar o experimento reprodutível em laboratório, todos os estímulos externos são simulados por sensores touch presente na própria ESP32, mapeados para “nova amostra inercial”, “comando de rota/telemetria” e fail-safe (emergência). Com isso, investigamos como restrições temporais rígidas (hard) e brandas (soft) se comportam sob diferentes políticas de prioridade (Rate Monotonic, Deadline Monotonic e um critério customizado), além das configurações do escalonador.

O foco está na previsibilidade temporal: a tarefa periódica de fusão (T=5 ms) alimenta o controle de atitude com *deadline* curta, enquanto eventos de navegação/telemetria são assíncronos e o fail-safe exige baixa latência (gerada a partir de uma ISR que ativa task). O desempenho é medido por *misses* de *deadline*, *jitter*

e latência ao toque, variando preempção/time slicing, frequência de CPU (80/160/240 MHz) e execução em unicore. O resultado permite discutir trade-offs entre robustez do controle, resposta a emergências e custo de preempções no tempo de execução.

Introdução

O sistema de controle do drone utiliza uma tarefa periódica para manter o estado inercial (fusão de sensores simulada) e tarefas encadeadas para controle de atitude. Eventos de navegação, telemetria e fail-safe são gerados por toques em pads touch. Todo o fluxo é instrumentado para medir latências e verificar deadlines sob diferentes políticas de prioridade e configurações do FreeRTOS.

Mapeamento de entradas (touch)

- Touch B: comando de rota (evento de navegação).
- Touch C: solicitação de telemetria/flush.
- Touch D: fail-safe/emergência.
- Touch A (opcional): injeta perturbação nos sinais para estresse de carga.

Tasks

1. FUS_IMU – Hard RT (analisar se é hard ou soft) – Periódica
    - T (período): 5 ms; D: 5 ms; C (WCET): 1,0 ms (deve ser medido)
    - Ação: filtragem/estimador (complementar/Madgwick) produz estado inercial.
    - Observação: produz amostra/estado e sinaliza a próxima task (queue/notify).
2. CTRL_ATT – Hard RT (analisar se é hard ou soft) – Encadeada (por FUS_IMU)
    - Gatilho: notificação/queue da FUS_IMU.
    - D: 5 ms após a amostra; C: ~0,8 ms (deve ser medido)
    - Ação: PID e atualização de atuadores (simulados).
3. NAV_PLAN – Soft RT (analisar se é hard ou soft) – Evento (Touch B)
    - D: 20 ms; C: 3–4 ms (deve ser medido)
    - Ação: atualiza waypoint/rota; se flag de telemetria (Touch C), envia estado.
4. FS_TASK – Hard RT – Evento (Touch D)
    - D: 10 ms desde o toque; C: 0,8–1,0 ms
    - Ação: pouso/hover seguro; reduz throttle imediatamente.
5. MONITOR_TASK
    - D: 500 ms e periódica; C: deve ser medido.
    - Ação: enviar e receber comandos de um PC via conexão WiFi. Deverá ser enviado o Log de eventos para o PC e receber comandos do PC para acionar ativar as task
        1. FS_TASK
        2. NAV_PLAN
6. TIME_TASK
    - D: 1 s e periódica; C: deve ser medido.
    - Ação: capturar a hora de um servidor SNTP e monitorar quantos ciclos se passaram desde a última atualização do servidor.

Temática 2 – Esteira industrial

Nesta temática, a ESP32 em FreeRTOS emula uma linha de produção com controle de velocidade da esteira, detecção e triagem de peças e parada de segurança. Para padronizar os cenários, todos os eventos vêm de sensores touch: “tick” de encoder/velocidade, detecção de objeto, solicitação de HMI/telemetria e E-stop. O sistema combina uma tarefa periódica (amostragem/estimativa de velocidade) com tarefas encadeadas e dirigidas a eventos, impondo *deadlines* rígidas às ações críticas (controle e segurança) e *soft* às funções de interface.

A investigação concentra-se em garantia de prazos sob diferentes distribuições de prioridade (RM, DM e customizada priorizando segurança) e estratégias de escalonamento (preemptivo/cooperativo, com/sem *time slicing*). Medimos *jitter*, *misses* e latência do E-stop enquanto variamos a frequência de CPU e fixamos a execução em um único core, quantificando quando os prazos críticos podem ser formalmente atendidos e como carga adicional e preempções afetam a estabilidade do controle e a confiabilidade da triagem.

Introdução

Uma esteira faz leitura de encoder, detecta objetos por sensor (IR/indutivo), aciona atuador de desvio e registra produção. Há interface HMI simples e telemetria. O touch sensor injeta paradas de segurança (*E-stop*) ou confirma modos de operação, testando respostas sob diferentes políticas de prioridade e escalonadores.

Entradas externas (touch)

- Touch B: detecção de objeto
- Touch C: solicitação de HMI/telemetria
- Touch D: E-stop (parada de emergência)
- Touch A (opcional): injeta “picos” de carga — não é gatilho principal

Tasks

1. ENC_SENSE – Hard RT (analisar se é hard ou soft) – Periódica
    - T: 5 ms; D: 5 ms; C: 0,6–1,0 ms (deve ser medido)
    - Ação: estima RPM/posição a partir de encoder simulado.
2. SPD_CTRL – Hard RT (analisar se é hard ou soft) – Encadeada (por ENC_SENSE)
    - Gatilho: notificação/queue de ENC_SENSE.
    - D: 10 ms após ENC_SENSE; C: ~1,2 ms (deve ser medido)
    - Ação: PI e atualização de PWM (simulado). Pode atender flag de HMI/telemetria (Touch C) em trecho não crítico.
3. SORT_ACT – Hard RT (analisar se é hard ou soft) – Evento (Touch B)
    - D: 10 ms desde o toque; C: ~0,7–1,0 ms (deve ser medido)
    - Ação: agenda/aciona solenoide de desvio no instante correto.
4. SAFETY_TASK – Hard RT (analisar se é hard ou soft) – Evento (Touch D via ISR)
    - D: 5 ms desde o toque; C: 0,8–1,0 ms (deve ser medido)
    - Ação: zera PWM e sinaliza alarme.
5. MONITOR_TASK
    - D: 500 ms e periódica; C: deve ser medido.
    - Ação: enviar e receber comandos de um PC via conexão WiFi. Deverá ser enviado o Log de eventos para o PC e receber comandos do PC para acionar ativar as task:
        1. SORT_ACT

2. SAFETY_TASK

6. TIME_TASK
    - D: 1 s e periódica; C: deve ser medido.
    - Ação: capturar a hora de um servidor SNTP e monitorar quantos ciclos se passaram desde a última atualização do servidor.

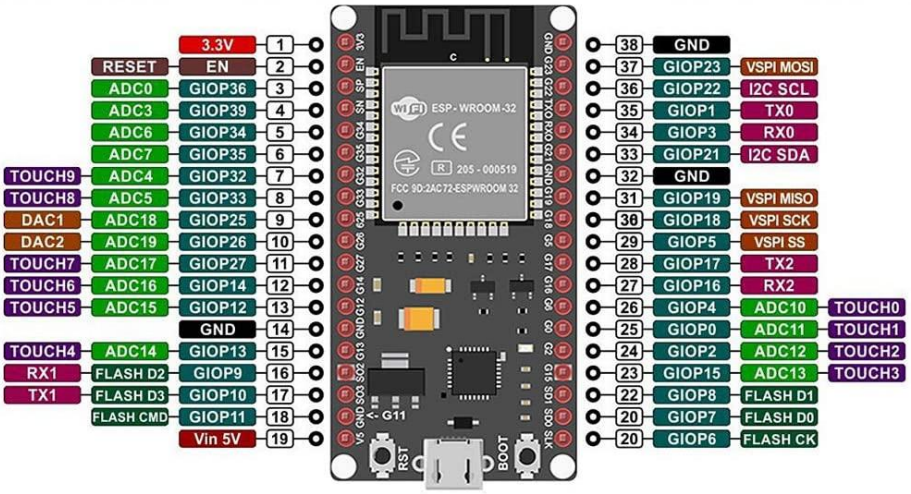
Metodologia de teste

1. Testar com a configuração política de atribuição de prioridade, frequência e configuração de escalonador que teve, segundo resultados da M1:
    - Melhor atendimento de prazos
    - Pior atendimento de prazos
2. A comunicação com o PC deve ser feita com os testes usando UDP e TCP. A análise deve identificar qual o pior e melhor protocolo para essa operação (mostre evidências). A comunicação deve ser feita usando WiFi e os alunos poderão usar os códigos disponíveis no Github: Link
3. Parte do log deve considerar a captura da hora atual capturada via SNTP (TIME_TASK) mais ms no momento do envio pela ESP32 e PC para analisar o delay de uma comunicação WiFi e o quanto ela compromete o atendimento de restrições de tempo. Isso permitira calcular uma espécie de Round Trip Time (RTT).
    - Para fácil análise posterior, você pode gerar um pacote com mensagens contendo o número de sequência, o protocolo utilizado, o tamanho do pacote, e timestamps coletados via SNTP, que permitirão medir o atraso de transmissão (latência) entre os dispositivos.
4. O WCRT (Worst Case Response Time) deve ser obtido para cada tarefa, determinando o maior tempo de resposta observado durante os experimentos. Você pode utilizar a metodologia HWM (High Water Mark) na análise, como por exemplo HWM(100%), HWM(99%), ...
5. O modelo (m,k)-firm será usado para avaliar o cumprimento parcial de deadlines em tarefas do tipo soft real-time. Deve-se definir janelas de k execuções consecutivas e contar quantas delas (m) cumpriram seus prazos.
    - Por exemplo, um sistema (9,10)-firm garante que pelo menos 9 de cada 10 execuções respeitam o deadline.
6. Com base na distribuição de prioridades, indicar:
    - Quais tarefas sofrem interferência.
    - Quais tarefas sofrem bloqueio (com base no uso de mecanismos que provocam inversão de prioridade).
7. Critérios de sucesso:
        - Todos os deadlines foram atendidos durante vários testes
8. Coleta e relatório:
    - Logar timestamp_evento, timestamp_task_start, timestamp_task_end.
    - Calcular misses, latência da IS ativar a task e ocupação de CPU.
    - Tabela comparativa por cenário (política × preempção × freq.)
9. Template de Tabela de Resultados (deve ser alterado conforme resultados e escolha de temática), exemplo:

Tema	Configuração da M1	Protocolo	WCRT (ms)	HWM	(m,k)-firm	Hard misses	Soft misses	Latência Touch (ms)	Latência Protocolo (ms ou s)	Conclusão
Drone	Melhor atendimento de prazos	UDP	10	HWM(100%) = 10 ms	(9,10)-firm	10%	4%	3,1	2 ms	OK hard; soft aceitável

Itens importantes

1) Você(s) deverá(ão) desenvolver o software requisitado pela empresa, seguindo a especificação sobre sensores, controladores e threads.
2) Deverá ser usado os sensores touch da ESP32, conforme indicado na imagem a seguir, para gerar as interrupções. Observe que os sensores então destacadas em cor roxa e iniciando com TOUCH. Há um total de 10 sensores.



Fonte: https://grobotronics.com/esp32-development-board-devkit-v1.html?sl=en

3) Com o software, vocês deverão fazer uma análise temporal criteriosa do sistema desenvolvido. Poderão usar as bibliotecas indicadas nesse link: General Purpose Timer.
4) Devem ser identificadas a tasks com requisitos temporais hard e soft, e descrever o que levou a essa análise. Além disso, as operações podem ser consideradas sem requisito temporal (essa escolha deve ser embasada). Lembre-se de atualizar as prioridades de execução das threads do sistema no qual você faz uso.

Ponto extra (máximo 1,0):

- Como forma de simular a operação de FS_TASK ou SAFE_TASK, o PC ao enviar o comando para a ESP32 deverá ligar o LED presente na placa.