Sistemas Embarcados
Módulo 3 –Sistemas Operacionais de Tempo Real

Professores: Alexandre Sales, Moacy Silva e Rafael Lima

FreeRTOS

Questão 1 – Criação e Inicialização de Tarefas

Objetivo:

Compreender o processo de criação e inicialização de tarefas em um sistema operacional de tempo real usando a API CMSIS-RTOS v2.

Implementação:

- Criar duas tarefas (Task1 e Task2) utilizando a função osThreadNew.
- Atribuir diferentes prioridades a cada tarefa (ex.: Task1 com prioridade normal e Task2 com prioridade baixa).
- Em cada tarefa, alternar o estado de um LED em intervalos distintos (por exemplo, Task1 a cada 500ms e Task2 a cada 1000ms).
- Inicializar o kernel com osKernelInitialize() e iniciá-lo com osKernelStart().
- Observar o comportamento das tarefas e verificar como a prioridade influencia a execução.
- Opcional: Usar UART no modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit para enviar mensagens indicando a execução de cada tarefa, avaliando o impacto no escalonamento conforme os diferentes níveis de prioridade.

Recursos envolvidos:

GPIO, LED, UART (modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit), FreeRTOS (CMSIS-RTOS v2)

Questão 2 – Tarefa Periódica com osDelay

Objetivo:

Criar uma tarefa que execute uma ação com intervalos fixos de tempo, utilizando a função de atraso cooperativo.

Implementação:

- Desenvolver uma tarefa (Task1) que pisque um LED a cada 500ms.

- Usar osDelay(500) dentro do laço infinito da tarefa para garantir o espaçamento temporal.
- Observar o comportamento da tarefa em funcionamento e verificar se o intervalo é estável.
- Utilizar um cronômetro ou monitorar a saída para verificar a precisão do periódico.
- Opcional: Utilizar UART para enviar uma mensagem a cada execução e avaliar o impacto da transmissão no tempo de execução da tarefa.

Recursos envolvidos:

osDelay, GPIO, LED, UART (modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit)

Questão 3 – Exploração dos Estados de uma Tarefa

Objetivo:

Permitir que o aluno compreenda os diferentes estados de uma tarefa no FreeRTOS e como transitar entre eles por meio de funções específicas. Essa atividade também deve permitir identificar e registrar cada vez que uma tarefa é executada.

Implementação:

- Criar uma tarefa que alterne entre os estados Running, Blocked e Suspended.
- Usar vTaskDelay() para colocar a tarefa em estado Blocked.
- Criar uma tarefa auxiliar que, após um tempo definido, suspenda a tarefa principal com vTaskSuspend() e posteriormente a retome com vTaskResume().
- Cada tarefa deve enviar uma mensagem identificadora pela UART no modo assíncrono (9600 bps, 8 bits, paridade None, 1 stop bit) indicando que foi executada. Exemplo de mensagem: Tarefa 1 - Executed, Tarefa 2 - Running, Tarefa 1 - Suspended, etc.
- Observar o comportamento através de LED e/ou mensagens via UART.

Recursos envolvidos:

vTaskDelay, vTaskSuspend, vTaskResume, UART (modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit), GPIO

Questão 4 – Medição de Tempo entre Execuções

Objetivo:

Medir o tempo decorrido entre execuções consecutivas de uma tarefa e apresentar esse valor como forma de diagnóstico.

Implementação:

- Criar uma tarefa (Task1) que, a cada execução, leia o tick atual com osKernelGetTickCount().
- Calcular a diferença entre o tick atual e o anterior, armazenado em uma variável estática.

- Exibir essa diferença em UART ou usar LED para representar atraso (por exemplo, mudar frequência do LED proporcional ao tempo).
- Analisar se a diferença se mantém constante com osDelay ou se varia conforme a carga do sistema.
- Opcional: Avaliar o impacto da transmissão UART sobre a regularidade da execução da tarefa em diferentes prioridades.

Recursos envolvidos:

osKernelGetTickCount, UART (modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit), GPIO

Questão Opcional – Comunicação entre Tarefas com Fila

Objetivo:

Implementar uma comunicação segura e organizada entre tarefas por meio de uma fila de mensagens (message queue).

Implementação:

- Criar uma fila usando osMessageQueueNew para armazenar mensagens do tipo char ou uint8_t.
- Task1 deve enviar mensagens para a fila em intervalos regulares (ex.: enviar os caracteres 'A', 'B', 'C', ...).
- Task2 deve ler a fila com osMessageQueueGet e executar uma ação com base na mensagem recebida (ex.: acionar LED ou enviar via UART).
- Testar o funcionamento com diferentes taxas de produção e consumo de mensagens.
- Opcional: Utilizar UART para registrar o fluxo de comunicação entre as tarefas e analisar o impacto da transmissão nos tempos de resposta.

Recursos envolvidos:

filas (osMessageQueue), LEDs, UART (modo assíncrono, 9600 bps, 8 bits, paridade None, 1 stop bit)

Formato de Entrega das Respostas

Os alunos devem entregar uma pasta compactada (.zip) contendo:

- Uma subpasta para **cada questão**, com o projeto correspondente criado no **STM32CubeIDE**;
- O código-fonte completo e devidamente comentado, indicando claramente onde e como as modificações foram feitas;
- Um **print da saída do Termite** (ou terminal UART equivalente) evidenciando os resultados obtidos;
- Opcionalmente, um **vídeo curto (em formato .mp4 ou .mov)** demonstrando a execução do projeto na placa;

- Um arquivo README (em PDF ou .txt) com:
    - Nome do aluno;
    - Lista das atividades respondidas;
    - Breve descrição de cada solução apresentada.

A pasta deve ser nomeada no seguinte padrão: RTOS_Atividade_2_NomeAluno.zip às questões e por fim anexadas ao Moodle conforme data configurada na plataforma.