SOTR - Task Management framework for FreeRTOS Rodrigo Lopes da Silva Santos, nº mec 89180 Rui Pedro Pereira Santos, nº mec 89293

Funcionalidades implementadas:

As funcionalidades implementadas neste projeto consistem no suporte básico de ativação de uma tarefa periódica, com a possibilidade de especificar o período de uma tarefa em ticks do sistema de gestão de tarefas TMAN implementado, a possibilidade de definir uma fase às tarefas também especificada em ticks do sistema, a capacidade de atribuir um deadline relativo e também a possibilidade de definir restrições de precedência entre tarefas.

Para além destas funcionalidades obrigatórias foram também implementadas como extra a permissão de atribuição de precedência múltipla, ou seja, uma tarefa depender de uma ou mais tarefas, bem como a deteção de deadline misses e a possibilidade de modificar o período e a fase das tarefas.

Estruturas de dados utilizados:

Para fazer a sua gestão, foi criada uma estrutura de dados para a definição dos vários parâmetros das tarefas.

A estrutura apresenta o seguinte formato:

char name
int priority
int period
int activations
int phase
int deadline
int deadline_misses
int ready
int precedence[5]
TaskHandle t handler

A estrutura possui atributos para armazenar o nome, prioridade, período, número de ativações, fase, deadline, deadline misses, assim como um atributo que serve de flag para assinalar quando a tarefa se encontra pronta a executar. Contém um array de identificadores para armazenar as precedências da tarefa bem como um identificador *TaskHandle t* usado quando se pretende fazer o seu *resume*.

Descrição da API implementada:

A API implementada conta com um conjunto de 6 funções, *TMAN_Init*, *TMAN_Close*, *TMAN_TaskAdd*, *TMAN_TaskRegisterAttributes*, *TMAN_TaskWaitPeriod*, *TMAN_TaskStats*, que se encontram detalhadas a seguir.

TMAN_Init (int TMAN_TICK_PERIOD_VALUE, int N_TASKS):

Função de inicialização da framework, onde é alocada a memória necessária para armazenar o array de estruturas de tarefas com tamanho *N_TASKS* (número de tarefas a gerir). É nesta função que é lançada a tarefa responsável pelo TICK do sistema, onde

TMAN_TICK_PERIOD é o número de ticks do FreeRTOS que equivalem a um TICK do nosso Task Manager.

TMAN TaskAdd (char name):

Método responsável por inicializar as flags e nome de uma tarefa, bem como chamar a função xTaskCreate() para criação e lançamento da designada tarefa. Após a criação da tarefa, esta é colocada em suspensão imediatamente.

TMAN_TaskRegisterAttributes (char name, int period, int phase, int deadline, int precedence constraints[]):

Função de registo dos atributos de uma tarefa já adicionada. A tarefa em questão é obtida através do seu nome, e os atributos período, fase, deadline, e precedence_constraints[] são inicializados.

TMAN_TaskWaitPeriod (void):

Função chamada pelas tarefas quando terminam a sua execução. Nesta função é executado o método *vTaskSuspend()* com o argumento *NULL* para que a tarefa se auto-suspenda.

TMAN TaskStats (void):

Função que imprime informação estatística relativa a cada tarefa, nomeadamente o seu número de ativações e o número de *deadline misses*.

TMAN Close(void):

Função que termina a execução da framework. Para cada tarefa criada, executa a função vTaskDelete() e por fim a função vTaskEndScheduler(), que pára o TICK do freeRTOS.

Para além destas funções de interação com a plataforma também foram criados mecanismos para o funcionamento da mesma. Nisto inclui-se uma tarefa mais prioritária responsável pelo *TICK* do sistema. Esta tarefa chama a função *TMAN_TaskStats* a cada ciclo, incrementa o *TICK* do sistema (*TMAN_TICK*), chama a função responsável pelo tratamento das tarefas (*task_manager*()) e faz a chamada ao método *vTaskDelayUntil*() do freeRTOS, colocando-se em bloqueio *TMAN_TICK_PERIOD* número de ciclos do freeRTOS.

A função *task_manager()* é responsável pelo *scheduling*, ou seja, faz a gestão das tarefas e quando devem ser executadas.

Para isso, foi criado um *for loop* onde todas as tarefas são percorridas e é verificado se o resto da divisão inteira entre o *TMAN TICK* e o período de cada tarefa é igual à sua fase.

Nesse caso, os atributos *ready* e *activations* da tarefa são incrementados, o que significa que se pretende executar um *resume* desta tarefa.

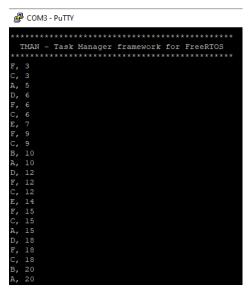
Após ter percorrido todas as tarefas, o *scheduler* já determinou quais é que ficaram *ready* e pode agora, através de outro ciclo, executar o *resume* das mesmas tendo em conta as condições de precedência.

De seguida, são novamente percorridas todas as tarefas e é verificado se estas têm o seu atributo *ready* superior a 0 e é inicializada uma variável *dont_execute* a 0. Nesse caso, é percorrido o seu atributo precedence (array com os índices das tarefas com relação de precedência) e é verificado se estas têm o seu atributo *ready* superior a 0. Nesse caso, a flag *dont_execute* é incrementada, o que significa que a tarefa atual não deve correr neste ciclo visto que tem relações de precedência com outras tarefas que também devem ser

executadas. Após percorrer o atributo *precedence* é verificado se a flag *dont_execute* é igual a 0 e, nesse caso, é executado o *vTaskResume(TASKS[task].handler)* da tarefa. Ao executar, a tarefa deve imprimir o seu nome e tick em que está a executar, deve decrementar o seu atributo *ready* e por fim deve executar a função *TMAN_TaskWaitPeriod*. Por fim, para verificar se ocorreu um *deadline miss*, é verificado se o resto da divisão inteira entre o *TMAN_TICK* e o período da tarefa é igual à soma entre o deadline e a fase da tarefa. Neste caso, é imprimido uma frase assinalando que houve um *deadline miss* e o atributo *deadline misses* da tarefa é incrementado.

Testes e resultados obtidos:

Ao longo da sua implementação, a plataforma foi testada para verificar o seu correto funcionamento. Para facilitar os testes e a perceção, cada tarefa ao executar, imprime o seu nome e o tick em que executou. (ex: "A, 20")



Nesta execução da framework foram criadas 6 tarefas com períodos diferentes, A - 5, B - 10, C - 3, D - 6, E - 7, F - 3. As tarefas tinham as seguintes prioridades: A - 0, B - 1, C - 2, D - 3, E - 3, F - 3. Como podemos verificar na imagem 1 tanto a prioridade como a periodicidade se encontram a funcionar corretamente, visto que Tarefas com períodos iguais, como no caso da tarefa C e F, são executadas por ordem de prioridade, sendo que a F imprime sempre primeiro que a C.

Figura 1: Teste de periodicidade e prioridade

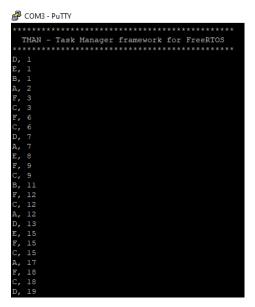
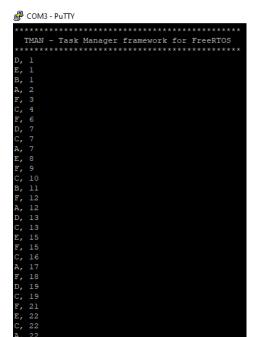


Figura 2: Teste de fase

Com as mesmas configurações anteriores foi adicionada uma fase a algumas tarefas: A - 2, B - 1, D - 1, E - 1. E como é possível verificar na figura 2, a tarefa A por exemplo corre em *TICKs* iguais a período + fase (7, 12 e 17).



Continuando com as configurações anteriores, foram adicionadas as seguintes precedências simples: B depende de C e C depende de F. Também foram adicionados deadlines por exemplo a tarefa B tem um deadline de 2 *TICKs*. Como é possível ver na figura 3, quando a tarefa C e F querem executar no mesmo *TICK*, a tarefa C espera pela execução da tarefa F, e apenas executa no *TICK* a seguir. (F, 3 -> C, 4). No *TICK* 21 as tarefas B, C e F pretendem executar devido aos períodos e fases. Assim, visto que a tarefa B apenas tem um deadline de 2 e depende da tarefa C que por sua vez depende da tarefa F, apenas executa dois *TICKS* depois da sua ativação, o que resulta num deadline miss.

Figura 3: Teste precedência simples e deadline

```
B, 23
TASK (A) NUMBER OF ACTIVATIONS = (5)
TASK (A) DEADLINE MISSES = (0)
TASK (B) NUMBER OF ACTIVATIONS = (3)
TASK (B) DEADLINE MISSES = (1)
TASK (C) NUMBER OF ACTIVATIONS = (7)
TASK (C) DEADLINE MISSES = (0)
TASK (D) NUMBER OF ACTIVATIONS = (4)
TASK (D) DEADLINE MISSES = (0)
TASK (D) DEADLINE MISSES = (0)
TASK (E) NUMBER OF ACTIVATIONS = (4)
TASK (E) NUMBER OF ACTIVATIONS = (4)
TASK (E) DEADLINE MISSES = (0)
TASK (F) NUMBER OF ACTIVATIONS = (7)
TASK (F) DEADLINE MISSES = (0)
```

Seguindo o teste anterior, para garantir que a contagem de ativações e deadline misses se encontra correta, foi impresso as estatísticas das tarefas com recurso à função TMAN_TaskStats(), e podemos ver que após o deadline miss este é apresentado nas estatísticas da tarefa B, bem como o número de ativações de cada uma das tarefas.

Figura 4: Teste à contabilidade de estatísticas

Figura 5: Teste precedência múltipla

Para testar a precedência múltipla, colocamos a fase da tarefa B a 0, a tarefa C depende agora da tarefa B e F e a tarefa B não tem dependências. De acordo com o esperado, a tarefa C que normalmente executaria no TICK 9, executa apenas 2 TICKs depois visto que depende da tarefa F que executa no TICK 9 e da tarefa B que executa no TICK 10.