João Martins (93183); Raquel Milheiro (92948)

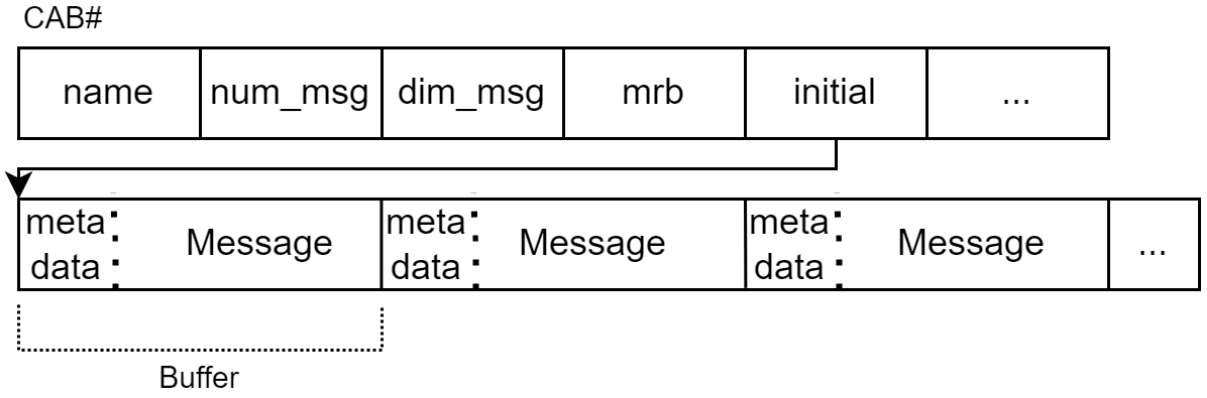
SOTR Projeto final - Relatório

CAB

Mecanismo de troca de mensagens que permite comunicação entre tarefas periódicas de forma assíncrona, portanto sem bloqueios. Outros requisitos do CAB são as tarefas encontrarem sempre a mensagem mais recente tarefas terem sempre um buffer disponível.

De forma a cumprir com os requisitos um CAB ao inicializar aloca em memória buffers suficientes (dependente de introdução de um parâmetro correto) de forma que todas as tarefas que necessitem de acesso a mensagem como um buffer para escrita tenham sempre um disponível. O CAB guarda também o endereço do buffer com a mensagem mais recente.

Um CAB é descrito por duas estruturas, uma com os metadados gerais do CAB e a outra para metadados relevantes, ref counting, aos buffers sob o domínio do CAB (o número de buffers é um parâmetro de inicialização).



Uma operação de escrita num CAB começa por uma tarefa reservar um buffer, *cab_reserve*. Para encontrar um buffer disponível são, primeiro, desligadas as interrupções, são percorridos todos os buffers até encontrar um em que *ref_count* seja 0, ou seja, não tem qualquer tarefa a usá-lo. Após encontrar um buffer livre este é marcado como em uso e voltam-se a ligar as interrupções. O pointer para este buffer é então devolvido à tarefa, sendo da responsabilidade desta sinalizar que o buffer está pronto, *cab_putmes*, onde se volta a desligar as interrupções, marca-se o buffer como livre, e atualiza-se o pointer para a mensagem mais recente, *mrb*, finalizando com o ligar das interrupções.

Uma operação de leitura de uma mensagem num CAB,*cab_getmes,* começa por desligar as interrupções, atualizar o buffer com a mensagem mais recente, *mrb*, como estado e uso(aumentado o número de referências para aquele buffer, *ref_count*), liga-se às interrupções, e finaliza-se retornando o pointer para o buffer. Cabe a tarefa libertar o buffer quando deixa de necessitar dele, *cab_unget*, ao reduzir o número de referências para aquele buffer.

Image Processing Tasks

As imagens seguem uma estrutura dada por:

Foram implementadas 3 tarefas para aplicar cada um dos 3 algoritmos de processamento de imagem, baseados no código exemplo dado:

- Guideline Search: pesquisa pelo início e fim da guideline nas correspondentes linhas da imagem, definidas por GN_ROW e GF_ROW respetivamente, retorna o ângulo e a posição do começo, na forma de percentagem, da guideline;
- Near Obstacle Search: pesquisa pela existência de objetos na CSA returnando no momento de detecção do primeiro objeto;
- Object Counting: percorre toda a imagem de forma a contar todos os objetos presentes.

Estas 3 tarefas são periódicas com as seguintes prioridades (“Threads — Zephyr Documentation” 2022):

- Near Obstacle Search: -1, tornando esta tarefa numa tarefa cooperativa e portanto não preemtable devido a importância desta tarefa;
- Guideline Search: 2, tarefa com prioridade de segundo nível mas preemtable;
- Object Counting: 3, tarefa com menor importância e preemtable pois só é usada para fins estatísticos.

Todas estas tarefas consomem a imagem do CAB responsável pela imagem, cab_img, e colocam os seus resultados nos CABs respectivos: cab_guideline, cab_near_objs, cab_obj_count.

Temporal Characterization of the Tasks

A medição do WCET das várias tarefas é facilitada pelo facto de o nRF52840 ser um SoC unicore.

O scheduling do Zephyr OS é baseado em prioridades e em 2 tipos de tarefas (“Threads — Zephyr Documentation”)

- Tarefas cooperativas, prioridade < 0: assumem controlo do CPU durante a execução sendo responsáveis por libertar o CPU. São apenas suplantadas por ISRs
- Tarefas Preemtable prioridade >= 0: Podem ser suplantadas por tarefas cooperativas ou tarefas preemtable com maior prioridade.

Só foram medidos tempos para as tarefas de processamento de imagem e output, uma vez que a tarefa responsável pela comunicação serial apenas gera imagens localmente.

Metodologia

1. A medição do WCET foi feita tarefa a tarefa.
2. São usados dados (imagens) geradas localmente correspondentes ao worst case:
    a. Guideline: Guideline corresponde à última coluna na imagem.
    b. NearObjects: Não haver objetos na CSA, forçando o algoritmo a percorrer toda a CSA.
    c. Object Count: Qualquer imagem, pois este algoritmo percorre toda a imagem.
3. A prioridade da tarefa em análise é definida com o valor -15, menor valor possível, de forma a tornar a tarefa em uma tarefa cooperativa de maior prioridade.
4. São usadas as utilidades de timing.h.
5. É feito o timing do bloco principal da tarefa (loop while).
6. São recolhidos, no mínimo, 15 valores por tarefa.
7. Para cada tarefa é atribuído como WCET o pior valor obtido.

Resultados

Tarefa	Guideline	Near Objects	Object Count	Output*
Tempo(µs)	60,0	326,0	2068,0	11774,0/4,2

Ao medir o WCET da tarefa responsável pelo output inicialmente reportava um tempo de 11,774 ms, tempo demasiado alto para a função da tarefa, ler e escrever os valores resultantes da aplicação dos algoritmos de processamento de imagem, no entanto, ao comentar as chamadas de 'printf' o valor reduziu para 4,2µs, valor dentro do esperado. Esta diferença deve-se ao tempo de transmissão de todos os caracteres por parte da tarefa ao usar 'printf'.

Test Procedures

Tanto a funcionalidade base do CAB e dos algoritmos de processamento foram testadas no ambiente de desenvolvimento de forma a suportar uma iteração mais rápida e acesso a um melhor runtime debugger. Não foram executados testes formais e unidade, o teste destes componentes baseou-se em correr pequenas aplicações de forma a usar estes módulos em diferentes casos e observando os resultados tanto na consola como no debugger.

Como não foi feita a comunicação pela UART, foram criadas imagens 128*128 para teste geradas no dispositivo. O teste da aplicação final foi feito recorrendo a estas imagens, sendo alteradas entre runs do programa de forma a ter vários casos de teste.

Code Changes

- Depois da demonstração, foi alterado o código do CAB de maneira a desligar as interrupções, através da função irq_lock() quando as tarefas estão na zona crítica e voltar a ligá-las através da função irq_unlock() quando as tarefas saem da zona crítica.
- Foi detectado também um erro na rotina de inicialização dos CABs quando é passada uma mensagem inicial, o *ref_count* era aumentado e não colocado a 0.

- Foi também eliminado código no ficheiro *imageGenAuto.h* que não era utilizado (bloco switch).

References

Butazzo, Giorgio C. “HARTIK: A Real-Time Kernel for Robotics Applications.” Accessed 10 January 2023.

“Threads — Zephyr Documentation.” *Zephyr Project Documentation*, 30 September 2022, https://docs.zephyrproject.org/latest/kernel/services/threads/index.html#thread-priorities. Accessed 10 January 2023.