Curitiba, 22, março de 2022.

**Disciplina**: Sistemas Computacionais

**Professor:** Jhonatan Geremias

**Curso: Engenharia de Computação**

**Nomes dos Estudantes:**

* **Ricardo Godoi Kurashiki**
* **Carlos Eduardo Marques Assunção Torres**
* **Milena Heloisa de Amorim Silvério**

**Atividade Prática / Relatório**

**Tarefas FreeRTOS**

# **Descrição da Atividade:**

Esta atividade é composta por duas etapas, primeiramente responder as questões do item 1, na sequência implementar o código especificado no item. A atividade compreende o conceito de tarefas e o seu uso no FreeRTOS.

**Entrega:**

Esta atividade deverá ser entregue até o dia **05/04/2022** no Canvas.

O estudante deverá entregar um arquivo “.pdf” contendo as respostas do roteiro de atividade item 1.

O item 2 é a implementação do código no FreeRTOS, seguindo a especificação do roteiro. O estudante deverá implementar o seu código apenas dentro do arquivo “example.c”. Entregar apenas o arquivo “example.c” onde foi codificado, deve conter o nome do estudante e curso adicionados no cabeçalho do arquivo como comentário.

Ambos os arquivos devem estar compactados no formato zip e postado no Canvas até a data limite.

**Roteiro da Atividade:**

## **Com apoio ao material fornecido responda:**

* 1. Descreva o uso da função xTaskCreate(), para que é utilizada essa função? Quais os parâmetros da função e para que são utilizados?

**O xTaskCreate é responsável por criar tarefas para o RTOS, com funcionalidade parecida das Threads de um sistema operacional padrão.**

**1º parâmetro – Função que será rodada na tarefa em questão;**

**2º parâmetro – Passa uma String que determina o nome da tarefa;**

**3º parâmetro – Tamanho da pilha da tarefa;**

**4º parâmetro – Parâmetros que serão passadas para a função da tarefa, sendo sempre necessário fazer cast tanto para passar o parâmetro como também para receber na função;**

**5º parâmetro – Prioridade da tarefa;**

**6º parâmetro – Task Handle relacionada à tarefa, sendo sempre necessário passar o endereço do Handle. Utilizado para alterar as propriedades da tarefa dentro da função, como por exemplo alterar a prioridade da tarefa programaticamente.**

* 1. Descreva para que são utilizadas as funções vTaskDelay() e vTaskDelete()?

**vTaskDelay – A função libera o processamento de uma tarefa até que o tempo termine, para permitir que outras tarefas rodem enquanto o tempo ainda não acabou.**

**vTaskDelete – A função é responsável por deletar uma tarefa específica. É utilizado quando uma tarefa não é mais necessária, assim sendo preciso excluí-la explicitamente.**

* 1. O que faz a função vTaskStartScheduler()?

**vTaskStartScheduler – Essa função é responsável por iniciar o escalonador e executar as tarefas criadas anteriormente com o xTaskCreate.**

1. **Implemente o programa no FreeRTOS conforme a especificação:**

Contexto: Para auxiliar no combate da Pandemia do Covid-19, o centro pesquisa de equipamentos médicos está convidando você a auxiliar no desenvolvimento de um monitor dos dados vitais dos pacientes. O monitor deve registrar os batimentos cardíacos do paciente (considerar entre 20 e 140 batimentos cardíacos - abaixo de 50 mensagem de batimento cardíaco alto, acima de 90 apresentar mensagem batimento cardíaco alto), nível saturação do oxigênio (considerar oxigenação entre 80% e 100% - abaixo de 90% mensagem de saturação baixa) no sangue e temperatura (considerar temperatura de 34° a 41° - abaixo de 35° apresentar mensagem de hipotermia, acima de 37,5° mensagem de febre).

* 1. Implementar um programa no FreeRTOS destinado a equipamentos médicos que deve medir os batimentos cardíacos, saturação de oxigênio e febre;
  2. Deverá ser criado três tarefas um para monitorar cada um dos dados vitais;
  3. A criação das três tarefas deve ser realizada no main\_;
  4. Para simular os dados vitais do paciente deverá ser utilizado funções randômicas para gerar cada um dos dados vitais;
  5. Os dados vitais (aleatórios) devem ser gerados dentro de cada tarefa;
  6. Todas as tarefas criadas devem ter a mesma prioridade;
  7. Efetuar a passagem de parâmetros para cada tarefa (sendo “Batimentos:”,” Saturação:” e “Temperatura:”);
  8. Fornecer os dados vitais na saída do console (batimento cardíaco, temperatura, saturação) – utilizar a função vPrintStringAndNumber();
  9. As tarefas devem gerar alertas quando os dados vitais dos pacientes estiverem alterados (mensagem no console).
  10. Utilizar a função vTaskDelay() configurando um tempo de um segundo para cada tarefa.
  11. Todas tarefas devem definir sua exclusão explicita utilizando a tarefa vTaskDelete();
  12. O código deve ser documentado, utilizar os comentários em toda a extensão do programa.

1. #include "FreeRTOS.h"
2. #include "task.h"
3. #include "basic\_io.h"
4. #include <stdlib.h>
6. // Prototipando a funcao da task que ira monitorar os batimentos cardiacos.
7. void monitBatimentos(void \*pvParameters);
8. // Prototipando a funcao da task que ira monitorar a saturacao.
9. void monitSaturacao(void\* pvParameters);
10. // Prototipando a funcao da task que ira monitorar a temperatura.
11. void monitTemperatura(void\* pvParameters);
13. // Criacao das variaveis que irao simular o batimento, a saturacao e a temperatura.
14. volatile int batimentos = 0;
15. volatile double saturacao = 0;
16. volatile double temperatura = 0;
18. int main\_(void)
19. {
20. // Criando processo de monitoramento de batimentos passando a funcao de monitoramento de batimentos, pilha de 1000 bytes, e prioridade 1.
21. xTaskCreate(monitBatimentos, "Monitoramento de Batimentos", 1000, NULL, 1, NULL);
22. // Criando processo de monitoramento de saturacao passando a funcao de monitoramento de saturacao, pilha de 1000 bytes, e prioridade 1.
23. xTaskCreate(monitSaturacao, "Monitoramento de Saturacao", 1000, NULL, 1, NULL);
24. // Criando processo de monitoramento de temperatura passando a funcao de monitoramento de temperatura, pilha de 1000 bytes, e prioridade 1.
25. xTaskCreate(monitTemperatura, "Monitoramento de Temperatura", 1000, NULL, 1, NULL);
27. // Iniciando o escalonador para gerenciamento dos processos.
28. vTaskStartScheduler();
30. for (;;);
32. return 0;
33. }
35. void monitBatimentos(void \*pvParameters)
36. {
37. for (;;)
38. {
39. // Gera um numero aleatorio para batimentos entre 0 e 140.
40. batimentos = rand() % 141;
42. vPrintStringAndNumber("\n\nBATIMENTOS: ", batimentos);
44. /\*
45. Caso os batimentos sejam menores que 50, informara o usuario que esta com
46. batimentos baixos, caso seja maior que 90, informara o usuario que esta com
47. batimentos altos.
48. \*/
49. if (batimentos < 50)
50. vPrintString("!! Batimentos cardiacos baixos. !!");
51. else if (batimentos > 90)
52. vPrintString("!! Batimentos cardiacos altos. !!");
54. vTaskDelay(500);
55. }
57. vTaskDelete(NULL);
58. }
60. void monitSaturacao(void\* pvParameters)
61. {
62. for (;;)
63. {
64. // Gera um numero aleatorio para saturacao entre 0 e 100 por cento.
65. saturacao = rand() % 101;
67. vPrintStringAndNumber("\n\nSATURACAO: ", saturacao);
69. // Caso a saturacao seja menor que 90%, informara ao usuario que esta com saturacao baixa.
70. if (saturacao < 90)
71. vPrintString("!! Saturacao baixa. !!");
73. vTaskDelay(500);
74. }
76. vTaskDelete(NULL);
77. }
79. void monitTemperatura(void\* pvParameters)
80. {
81. for (;;)
82. {
83. // Gera um numero aleatorio para saturacao entre 0 e 41.
84. temperatura = rand() % 42;
86. vPrintStringAndNumber("\n\nTEMPERATURA: ", temperatura);
88. /\*
89. Caso a temperatura seja menor que 35, informara o usuario que esta com
90. hipotermia, caso seja maior que 37.5, informara o usuario que esta com
91. febre.
92. \*/
93. if (temperatura < 35)
94. vPrintString("!! Voce esta com hipotermia. !!");
95. else if (temperatura > 37.5)
96. vPrintString("!! Voce esta febril. !!");
98. vTaskDelay(500);
99. }
101. vTaskDelete(NULL);
102. }