Pontifícia Universidade Católica do Paraná
Escola Politécnica
Engenharia Elétrica
Engenharia da Computação

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:

1. Com apoio ao material fornecido responda:
    a. 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.

b. 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.

c. O que faz a função vTaskStartScheduler()?

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

2. Implemente o programa no FreeRTOS conforme a especificação:

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Engenharia Elétrica
Engenharia da Computação

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).

a. Implementar um programa no FreeRTOS destinado a equipamentos médicos que deve medir os batimentos cardíacos, saturação de oxigênio e febre;
b. Deverá ser criado três tarefas um para monitorar cada um dos dados vitais;
c. A criação das três tarefas deve ser realizada no main_;
d. Para simular os dados vitais do paciente deverá ser utilizado funções randômicas para gerar cada um dos dados vitais;
e. Os dados vitais (aleatórios) devem ser gerados dentro de cada tarefa;
f. Todas as tarefas criadas devem ter a mesma prioridade;
g. Efetuar a passagem de parâmetros para cada tarefa (sendo “Batimentos:”,” Saturação:” e “Temperatura:”);
h. Fornecer os dados vitais na saída do console (batimento cardíaco, temperatura, saturação) – utilizar a função vPrintStringAndNumber();
i. As tarefas devem gerar alertas quando os dados vitais dos pacientes estiverem alterados (mensagem no console).
j. Utilizar a função vTaskDelay() configurando um tempo de um segundo para cada tarefa.
k. Todas tarefas devem definir sua exclusão explicita utilizando a tarefa vTaskDelete();
l. O código deve ser documentado, utilizar os comentários em toda a extensão do programa.

```c
#include "FreeRTOS.h"
#include "task.h"
#include "basic_io.h"
#include <stdlib.h>

// Prototipando a funcao da task que ira monitorar os batimentos cardiacos.
void monitBatimentos(void *pvParameters);
// Prototipando a funcao da task que ira monitorar a saturacao.
void monitSaturacao(void* pvParameters);
// Prototipando a funcao da task que ira monitorar a temperatura.
void monitTemperatura(void* pvParameters);

// Criacao das variaveis que irao simular o batimento, a saturacao e a temperatura.
volatile int batimentos = 0;
volatile double saturacao = 0;
volatile double temperatura = 0;

int main_(void)
{
  // Criando processo de monitoramento de batimentos passando a funcao de monitoramento de batimentos, pilha de 1000 bytes, e prioridade 1.
  xTaskCreate(monitBatimentos, "Monitoramento de Batimentos", 1000, NULL, 1, NULL);
  // Criando processo de monitoramento de saturacao passando a funcao de monitoramento de saturacao, pilha de 1000 bytes, e prioridade 1.
  xTaskCreate(monitSaturacao, "Monitoramento de Saturacao", 1000, NULL, 1, NULL);
  // Criando processo de monitoramento de temperatura passando a funcao de monitoramento de temperatura, pilha de 1000 bytes, e prioridade 1.
  xTaskCreate(monitTemperatura, "Monitoramento de Temperatura", 1000, NULL, 1, NULL);

  // Iniciando o escalonador para gerenciamento dos processos.
  vTaskStartScheduler();

  for (;;);

  return 0;
}

void monitBatimentos(void *pvParameters)
{
  for (;;)
  {
            // Gera um numero aleatorio para batimentos entre 0 e 140.
            batimentos = rand() % 141;

            vPrintStringAndNumber("\n\nBATIMENTOS: ", batimentos);

            /*
            Caso os batimentos sejam menores que 50, informara o usuario que esta com
            batimentos baixos, caso seja maior que 90, informara o usuario que esta com
            batimentos altos.
            */
            if (batimentos < 50)
                      vPrintString("!! Batimentos cardiacos baixos. !!");
            else if (batimentos > 90)
```

```
52.                     vPrintString("!! Batimentos cardiacos altos. !!");
53.
54.             vTaskDelay(500);
55.     }
56.
57.     vTaskDelete(NULL);
58. }
59.
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;
66.
67.             vPrintStringAndNumber("\n\nSATURACAO: ", saturacao);
68.
69.             // Caso a saturacao seja menor que 90%, informara ao usuario que esta
    com saturacao baixa.
70.             if (saturacao < 90)
71.                     vPrintString("!! Saturacao baixa. !!");
72.
73.             vTaskDelay(500);
74.     }
75.
76.     vTaskDelete(NULL);
77. }
78.
79. void monitTemperatura(void* pvParameters)
80. {
81.     for (;;)
82.     {
83.             // Gera um numero aleatorio para saturacao entre 0 e 41.
84.             temperatura = rand() % 42;
85.
86.             vPrintStringAndNumber("\n\nTEMPERATURA: ", temperatura);
87.
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. !!");
97.
98.             vTaskDelay(500);
99.     }
100.
101.            vTaskDelete(NULL);
102.    }
103.
```