Vitor Amadeu Souza

RTOS

para medir

GLP

Programado no Arduino

Abril de 2021

Direitos reservados por:
Cerne Tecnologia e Treinamento Ltda
Produção: Cerne Tecnologia e Treinamento
E-mail da Empresa: cerne@cerne-tec.com.br
Home Page: www.cerne-tec.com.br.com.br
Atendimento ao Consumidor: sac@cerne-tec.com.br
Contato com o Autor: vitor@cerne-tec.com.br

FEITO NO BRASIL

"A felicidade não deve limitar-se a um pequeno número de pessoas; ela pertence a todos."

Antoine L. Lavoisier

Kits Didáticos e Gravadores da Cerne Tecnologia

A Cerne tecnologia tem uma linha completa de aprendizado para os microcontroladores da família ARDUINO, 8051, Holtek, dsPIC, ARM e etc. Veja os detalhes de cada um nas figuras abaixo:

Kit Arduino

- Gravação On-Board
- Comunicação Serial RS232
- Entrada de 12 V

Uma linha completa de componentes para o desenvolvimento de seus projetos eletrônicos como displays, PICs, botões, leds, cristais etc. Visite a nossa página na Internet e conheça melhor nossos serviços e produtos.

Cerne
Conhecimento para o Desenvolvimento

www.cerne-tec.com.br

Sumário

I. Metodologia de desenvolvimento **6**

II. Acionando Saídas **8**

III. Pisca-Pisca **14**

IV. Botão e Led **23**

V. Display de 7 segmentos **26**

VI. Conversor AD, Display LCD e Serial **31**

VII. Medindo a concentração de GLP **35**

Capítulo I
Metodologia de desenvolvimento

1. Introdução ao RTOS

Esta literatura é uma continuação da obra *Arduino – Prático e Objetivo* (2011) e *Programação para Arduino – Avançado* (2014) do mesmo autor e editora, onde o objetivo é estudar como implementar um RTOS no Arduino para em seguida ler o sensor título da obra como um processo usando para isso o recurso do RTOS.

O RTOS permite a um microcontrolador tratar concorrentemente diversas tarefas. Desta forma, pode-se obter melhor desempenho e eficiência, já que diversas tarefas são executadas em uma fração do tempo, dando a impressão que todos estão sendo processados simultaneamente da mesma forma que ocorre nos atuais sistemas operacionais.

Neste livro, a biblioteca FREE RTOS foi utilizado na programação para o Arduino, onde diversos exemplos práticos utilizando recursos corriqueiros como I/Os, display, serial, ADC são mostrados de modo a facilitar o entendimento do leitor. O FREE RTOS pode ser baixado através do link abaixo.

https://github.com/greiman/FreeRTOS-Arduino

Após este passo, descompacte o arquivo e mova o conteúdo da pasta libraries para a pasta de mesmo nome do caminho em que está instalado o Arduino.

Reinicialize o Arduino e vá até o menu Sketch->Include library e observe que três novas bibliotecas foram adicionadas, sendo estas identificadas por FreeRTOS_ARM, FreeRTOS_AVR e Sdfat. Ao longo desta obra, a biblioteca FreeRTOS_AVR sempre deverá ser incluída em seus projetos.

A placa didática utilizada foi a Arduino UNO R3. Tal kit e acessórios estão à venda no site www.cerne-tec.com.br.

Capítulo II

Acionando Saídas

1. Introdução

Um processador de um núcleo como o Arduino processa sequencialmente um código gravado no mesmo. Desta forma, para executar mais de uma tarefa ao mesmo tempo é necessário a implementação de um RTOS (Real Time Operating System ou Sistema Operacional de Tempo Real) que possibilitará o tratamento de vários processos (thread) simultaneamente, pois o tempo de processamento de uma tarefa será dividido para vários processos, dando a impressão que todos eles são tratados simultaneamente.

Esta mesma lógica é implementada em um sistema operacional como o Linux ou Windows, onde um processador sequencial processa várias *threads* por vez como o editor de texto, navegador de internet, planilha e faz com que o usuário tenha a impressão de todos serem executados ao mesmo tempo.

2. Programa básico com RTOS

A estrutura básica de um programa com 2 tarefas, processos ou threads terá a estrutura definida abaixo.

```
#include <FreeRTOS_AVR.h>

static void Task1(void* arg)
{
  while(1)
  {

  }
}

static void Task2(void* arg)
{
  while(1)
  {

  }
}

void setup()
{
   xTaskCreate(Task1,NULL,configMINIMAL_STACK_SIZE,NULL,1,NULL);
   xTaskCreate(Task2,NULL,configMINIMAL_STACK_SIZE,NULL,1,NULL);
   vTaskStartScheduler();
   while(1);
}

void loop()
{

}
```

A função `vTaskStartScheduler()` inicia o sistema operacional e as funções `xTaskCreate(função,NULL,`

`configMINIMAL_STACK_SIZE,NULL,prioridade,NULL)` permite definir a prioridade de 0 (maior) a 31 (menor) e o nome da função.

Dentro das funções Task1 e Task2 será feito o código que se deseja executar dentro do bloco `while(1)`. Note que no RTOS a função `loop()` nunca é tratada, pois após inicializar o RTOS através da função `vTaskStartScheduler()` há um loop de repetição infinito na linha `while(1)`.

Obs: *Outros nomes de função assim como prioridades podem ser definidos.*

3. Esquema elétrico

Para este exemplo o seguinte esquema elétrico foi adotado.