O uso de um sistema operacional em tempo real (RTOS) para processamento de multitarefas é uma realidade cada vez mais presente nos projetos de sistemas embarcados. A ferramenta computacional SanUSB implementa um RTOS livre, desenvolvido pelo russo Victor Timofeev, através dos compiladores MPLABX C18 e CCS, baseado em interrupção de temporizadores.

Uma das principais características de um RTOS é a capacidade de processar tarefas concorrentes, ou seja, tarefas paralelas. Dessa forma, o RTOS torna a programação de projetos reais mais simples, pois basta descrever cada tarefa em uma função *task* do firmware, que o RTOS se encarrega do gerenciamento do processo. Dessa forma, O RTOS é baseado na ideia de multitarefas (*multithread*), onde cada tarefa é uma função em C do firmware em laço infinito.

Em um sistema multitarefa, inúmeras tarefas exigem tempo da CPU, e uma vez que existe apenas uma CPU, é necessária alguma forma de organização e coordenação pelo RTOS para que cada tarefa tenha o tempo que necessita. Na prática, cada tarefa tem um intervalo de tempo muito curto, assim parece que as tarefas são executadas de forma paralela e simultânea.

Como exemplo, estão disponíveis uma vídeo-aula em http://www.youtube.com/watch?v=s6BG8ZN0aDk e os programas em https://dl.dropbox.com/u/101922388/RTOSB.zip para os compiladores CCS e MPLABX C18. As duas práticas descritas abaixo de RTOS foram desenvolvidas com a placa SanUSB, que pode ser construída seguindo os tutoriais disponíveis em https://dl.dropbox.com/u/101922388/121007SanUSBOrig.zip . O compilador livre C18 e a plataforma MPLABX estão disponíveis para baixar em https://drive.google.com/open?id=0B5332OAhnMe2N3czQWxVX0JVSkE&authuser=0.

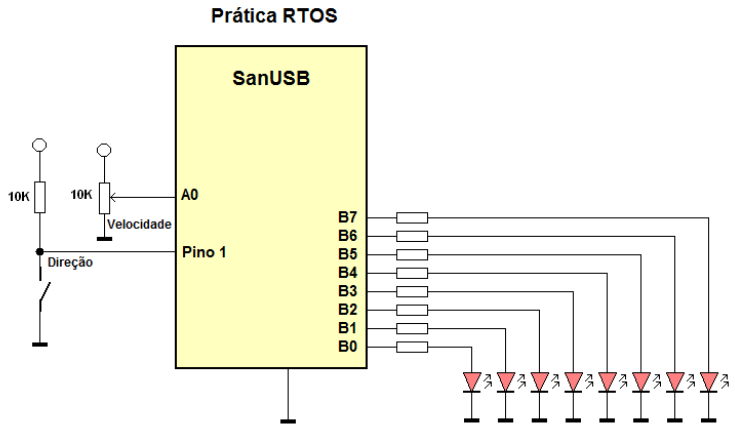
Prática 1: Nesta prática o RTOS executa em paralelo 3 tarefas concorrentes e paralelas em loop infinito para acionar 3 leds, conectados nos pinos B7, B6 e B5, de forma independente.

Prática 2: Nesta prática o RTOS executa em paralelo:

1- Tarefa de leitura do AD com potenciômetro para modificar a velocidade de brilho dos leds;

2- Tarefa de rotação de oito leds na porta B; e

3- Tarefa que inverte o sentido de rotação dos leds por botão no pino 1 (PIN_E3) utilizando display de sete segmentos como leds.



OBS: Dentro de cada projeto , necessário inserir o cabeçalho OSAcfg.h, como descrito abaixo, que deve indicar a quantidade de tarefas e as características do projeto como a prioridade das tarefas.

```
#ifndef _OSACFG_H
#define _OSACFG_H

#define OS_TASKS                3
#define OS_DISABLE_PRIORITY
#define OS_ENABLE_TTIMERS
```

A placa PIC SanUSB pode ser construída seguindo o tutorial e os programas disponíveis em https://dl.dropbox.com/u/101922388/121007SanUSBOrig.zip ou adquirida em http://lista.mercadolivre.com.br/sanusb .