Utilização do kernel RTKPIC18

Objectivos:

Tomar contacto com um *kernel* tempo-real para microprocessador PIC18 de 8 bit.

1. Observar as chamadas ao sistema típicas para gestão de tarefas.
2. Desenvolver um programa sobre um kernel deste tipo.

Procedimentos:

1. Compilação

O desenvolvimento de kernels tem uma forte dependência do compilador bem como das opções de compilação usadas. Por esse motivo **a compilação deve ser efectuada com o compilador PICC-18 versão 8.30**. Adicionalmente sugere-se que use como **base** para o desenvolvimento das aplicações **a *"makefile"* fornecida**.

Passos a seguir:

1. Descarregar o pacote "rtkpic-class-sources.zip" da página da disciplina e descompactar para a sua área de trabalho

2. Digitar o comando "**make**", para compilar. Observar eventuais mensagens de erro.

3. Carregar a aplicação no kit. Para tal digite o comando
   "sudo picload -p /dev/SERIAL_DEVICE -f aplicacao.hex"
   e de seguida fazer "reset" à placa. "SERIAL_DEVICE" poderá ser "/dev/ttyS{0,1,2,...}" para PCs com porta série embutida e "/dev/ttyUSB{0,1,2,...}" caso se usem conversores USB/RS-232.

4. Fazer novamente "reset" e a aplicação deve iniciar a sua execução.

2. Análise do código fornecido

Abrir com um editor de texto a aplicação-exemplo fornecida (sample.c).

1. Observar o respectivo código. Ver as chamadas ao sistema para definição, criação e activação de tarefas (consulte o manual resumido do RTKPIC18, rtkpic18-man.pdf).

2. Compile o programa referido e execute-o no kit, seguindo as instruções acima indicadas. Observar a execução independente das várias tarefas através do piscar dos vários leds às frequências correctas.

3. Tarefas a realizar:

Realize as tarefas abaixo indicadas. Documente todas as alterações ao código que efectuar. Deverá entregar um relatório (2 páginas, máximo) com a análise dos resultados observados e indicação das alterações ao software introduzidas.

1. Insira uma actividade em *background* (tarefa de *background),* que faça piscar o led mais à direita (porto RA5). A tarefa de *background* consegue manter periodicidade com baixo jitter? Justifique.

2. Reduza a carga computacional das tarefas 1 a 4 por um factor de 10. Como foi afectado o jitter da tarefa de *background*? Justifique.

3. Aumente a carga computacional das tarefas 1 a 4 por um factor de 10, em relação ao valor inicial. Consegue observar o efeito de preempção? Justifique.

4. Desactive a preempção. Que efeitos observa? Justifique.

5. Seleccione o escalonador EDF. Que efeitos observa? Justifique.

6. Modifique o código por forma a que as tarefas liguem o led que lhes está associado para sinalizar a violação de uma *deadline.* Seleccione o escalonador DM e aumente gradualmente a carga das tarefas aumentando o número de iterações do ciclo "for". Qual a primeira tarefa a sofrer violação de *deadline?* Para que valor do ciclo? Justifique o facto de ser esta tarefa?