OPEN COMPONENTS FOR EMBEDDED REAL-TIME APPLICATIONS

Demo 1: Stand Alone RTLinux
Author: Vicente Esteve (UPVLC)

Goals:

This demo checks the correct behavior of the Stand Alone RTLinux component, which is a porting of the RTLinux-GPL executive to a bare machine. saRTL do not need Linux to operate.
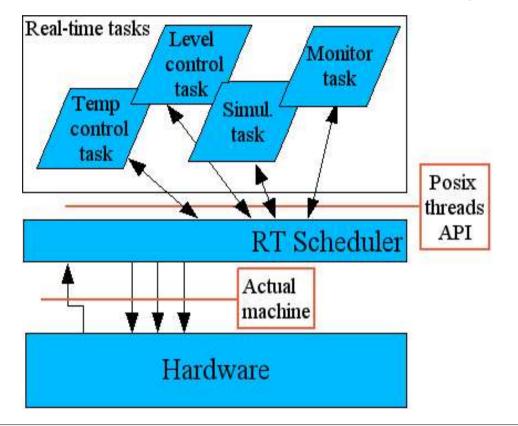The goal is to develop a water tank control aplication to run in a bare machine with saRTL. The evolution of the simulated system is monitored using RTTerminal component.

Tested components:

- saRTL (WP05)
- RTTerminal (WP05)

Description:

The aplication is a level and temperature control of a water tank. Level of water in the tank is controlled by a cold water input (Qn), while temperature is controlled by a warm water input (Qt), to compensate the output flow water (Qs). The control law implemented in the level and temperature control tasks is a propocional regulator.

The system has 4 tasks:

- Temperature control task. Calculates the degree of the warm water flow valve.
- Level control task. Calculates the degree of the cold water flow valve.
- Simulation task. Simulates the tank behaviour, since the computer is not connected to a real process.
- Monitoring task, that prints the state of the system in the console. This last task uses RTTerminal component to print messages and to draw the tank on the terminal, so the evolution of the system can be monitored in real-time.

Demo phases

Phase 1

Write the application code, in the host machine:
- File *sartl_directory/tasks/demo.c* contains the code of the four real-time tasks
- In *sartl_directory/tasks/tasks.c*
  - Create the *pthread_t* variables of each thread implemented in *demo.c*
  - In *init_tasks()* function create the threads using *pthread_create*

Phase 2

Compile and make the floppy image. Type:
make clean; make; cd arch/i386/boot; make disk
*If you want to debug the application then compile with the option „Enable GDB Agent" and „Start Breakpoint"

Phase 3

Boot the target from the floppy disk. If debugging is enabled, wait until the conection message with gdb appears.

Phase 4

Run ddd (or gdb) from the host machine, connected via RS232 with the target. You can even put breakpoints to stop the execution in the RTLinux scheduler function.

Conclusions and results:

The execution of the application shows how the level and the temperature of the tank is controlled as it was expected. Through gdb, the code can be executed step by step, which enables a complete control of the system. Interaction with the system, for instance, to change the reference of the controller parameters is not possible now, since the keyboard interaction is not implemented in the RTTerminal version for saRTL.