Modifying Nano-RK to support EDF with SRP

UBC | CICS 525

In this assignment you will extend a small real-time operating system kernel (Nano-RK) and you will also work with (and learn) a tool chain for embedded systems development. You can use a Windows-based system and Cygwin for this mini-project/assignment.

There are several goals to this assignment:

- Understand the architecture and design choices behind a real-time operating system kernel;
- Extend the capabilities of the RTOS kernel;
- Develop an understanding of the tool chain for embedded systems development;
- Gain an appreciation for the challenges of embedded software development;
- Improve upon work of others by examining prior engineering choices;
- Design and implement techniques for testing your modifications to the kernel.

1 Tasks

T0: Nano-RK. Nano-RK's source is available at the Nano-RK website.

To install Nano-RK on the MicaZ motes, you should start with the instructions on the Nano-RK website: MicaZ quick start. We will, however, use a slightly "hacked" version of Nano-RK that is available on the course web page.

The source code for Nano-RK is accompanied by several examples that will help you develop your own application easily. You should read the code for the example applications because they represent many of the features of the sensor node platform and Nano-RK. Read the code for Nano-RK itself. For this mini-project/assignment, you can initially restrict your attentions to a few files in the kernel directory of the source tree. The main source files that you will need to become familiar with, and modify, are `nrk.c`, `nrk_task.c` and `nrk_scheduler.c`, but you will also need to examine, and probably alter, some other functionality.

T1: EDF support. Nano-RK has been designed to support static priority scheduling. Extend the scheduler to support EDF. You may assume that tasks have relative deadlines that are equal to their periods. How will you test your implementation? Develop suitable tests to demonstrate that you have, indeed, implemented EDF. Modify the task structure in Nano-RK appropriately.

T2: Stack-based resource access protocol. To support resource sharing with the EDF scheduler, extend the semaphore implementation of Nano-RK to use SRP. (By default Nano-RK supports the Priority Ceiling Protocol.) Again, develop suitable tests to demonstrate that your implementation of SRP is correct.

We will use Avrora to simulate the serial port and leds of a micaZ-based WSN that has Nano-RK running upon. Please use the following instructions that supplement those on the Nano-RK web page (in particular, Step 3 of the instructions on the Nano-RK support page):

1. Download and Install Cygwin with the following modules:
   - `make`
   - `svn` (subversion)
   - `gcc-core`
   - `bison` (required to compile `avrdude`)
   - `openssl`
   - `openssh`
   - `inetutils` (required to `telnet` with Avrora)
2. Download and install WinAVR.
3. Download the `Nano-RK` tarball from the course web page. One option to unzip the tarball is to use 7-zip.
4. `cd` into `$NANORK_ROOT/projects/examples/basic_tasks`.
5. In `makefile`, make sure that `platform` is set to `micaZ`
   `PLATFORM = micaZ`.
6. Type "`make clean all`".
7. Install Java on your machine.
8. Download the precompiled Avrora tarball from the course web page.
9. Open a second shell window. `cd` into `$AVRORA_ROOT/avrora/bin`, and run Avrora using the command
   "`java avrora.Main -platform=micaz -monitors=serial,leds -action=simulate -program $NANORK_ROOT/projects/examples/basic_-tasks/main.elf`".
10. You should see the message "`Waiting for serial connection on port 2390...`". Avrora will be waiting for a `telnet` client to listen to the serial port. Open a third shell window and type "`telnet localhost 2390`". Here you should be able to see the output of the serial port.

2 Documentation and submission

Write a clear report for each task. Describe your implementation for the tasks. The report should be submitted as a PDF file and should include the names of the group members. Notice that we do not provide any tests for correctness. In engineering practice, you should consider testing mechanisms in conjunction with the design. You should *describe clearly* the methods you used to verify the correctness of your implementation.

Use `handin` to submit your work. For your implementation effort, you will (and should) only submit the files from the Nano-RK source code that were changed. Invoke `handin` from `remote.mss.icics.ubc.ca` as

follows after making sure that all the necessary files are in a directory named `nrk-edf`: `handin cics525 nrk-edf`