**EECE 494**
**Spring 2010**

# ASSIGNMENT 3
# DUE MARCH 17

This mini-project will provide you with some background in embedded systems programming and will expose you to cross-compilation and associated tool chains. You will use the MICAz sensor mote platform with the Nano-RK real-time operating system. You will be provided with two MICAz motes, one MIB520CB programming board and one MDA100CB prototype sensor board with light and temperature sensors. These devices should be returned intact at the end of the course.

The first research article describing Nano-RK is a good resource for understanding the architecture of this RTOS.

In this assignment you will explore aspects related to real-time operating system kernels and embedded systems application development. Specifically, you should accomplish the following tasks.

**Task 0A: Understand the hardware platform and the software tool chain.** Follow the instructions in the latter section of this assignment specification to install the required tools and to build and run applications on the MICAz mote using the Nano-RK operating system. Read the instructions carefully and follow the documentation on the Nano-RK website. 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, you can 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.

**Task 0B: Develop a project plan.** Divide the tasks/milestones among your group and establish due dates for the milestones. If multiple group members are going to participate in the same task then indicate the share of the responsibility each group member will take on. **The project plan is due on March 3 via email** to the instructor and the teaching assistant. If there are concerns about the distribution of work and the deadlines set, you will receive a response regarding the concerns. The project plan will be used for peer evaluation at the end of the assignment. **(5%)**

**Task 1: Get to know the wireless radio.** Measure the received signal strength (using the received signal strength indicator or RSSI) and the packet loss rate at different locations: a corridor of MacLeod Building, on Main Mall (outside MacLeod Building), and in a classroom of your choice. For these experiments, you will need to vary the distance between the two motes and measure the RSSI values. Plot a graph of the packet loss rate with respect to the RSSI values. What is the maximum observed distance for reliable communication between two nodes when indoors and when outdoors? Wireless communication is an important aspect of emerging embedded systems and this task is intended to make you familiar with issues regarding wireless channel reliability. **(10%)**

**Task 2: Earliest deadline first.** 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. **(30%)**

**Task 3: Stack-based resource access protocol**. To support resource sharing with the EDF scheduler, extend the semaphore implementation of Nano-RK to use SRP. Again, develop suitable tests to demonstrate that your implementation of SRP is correct. **(15%)**

**Task 4: Fixed length ticks vs. variable length ticks.** Nano-RK uses a variable length clock tick in its scheduling design. What are the advantages and disadvantages of this approach? What changes would you need to make to enable fixed length ticks? Do the changes depend on the scheduling policy chosen? You do not need to implement these changes but you should clearly describe the changes needed and the descriptions should refer to the appropriate source files and lines of code. **(10%)**

**Task 5: Documentation.** Write a clear report for Tasks 1-4. Describe your implementation for the tasks. For Tasks 2 and 3, you should include the modified data structures in the report. 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. **(15%)**

**Submission and evaluation:** Use `handin` to submit all your work before midnight on March 17. Group interviews will be scheduled for March 17 and March 22. (For instructions on using `handin`, see previous assignment specifications. Use `hw3` as the directory to hand in.). Your work will be evaluated on the basis of your report and a group interview, which may include a demonstration. **All members of the group are expected to be familiar with all aspects of the implementation. Questions will be asked on the implementation and on the relevant theory. The remaining 15% marks will be assigned based on the interview.**

## Getting started with MICAz and Nano-RK

This section describes the steps to build Nano-RK RTOS for MICAz sensor boards using Linux as the development platform. The content of this page is based on information provided on Nano-RK web site fine tuned for MICAz mote and the MIB520 programming board.

To build and run Nano-RK on a MICAz mote, follow the following five main steps. (Note: You will need root access to the machine you are using. The following steps have been tested on Fedora 9 and Fedora 12. You can use Linux within a virtual machine if you do not normally use Linux or follow suggestions on the Nano-RK page for development using Mac OS X or Cygwin but we will not provide any TA/instructor support for other environments.)

### 1. Install the Required Tools

To build Nano-RK the following need to be installed on your machine.

- Common tools: (These are standard tools, it is high likely that your machine already has these.)
  - flex-2.5.4a-34
  - bison-2.0-6

    - byacc-1.9-29
    - gcc-4.0.2-8
    - gcc-c++-4.0.2-8
    - make
- Install Atmel AVR Processor Family Tools:
    - binutils
    - avr-binutils
    - avr-gcc
    - avr-gcc-c++
    - avr-libc
    - uisp

  You can install the previous packages using `yum`. To install the packages using `yum` use the following command:

  ```
      yum install “package-name”
  ```

For example, to install the `uisp` package use the following command:

```
    yum install uisp
```

## 2. Getting the Nano-RK Code

Download the Nano-RK source code from the Nano-RK website and make changes as needed or as suggested to the build process.

## 3. Build the Basic Example

To test the Nano-RK on the MICAz mote try to compile and write the basic example “basic_tasks” provided in the Nano-RK source code. The basic_tasks source code is found in the directory `nano-RK/projects/basic_tasks`.

`basic_tasks` is a simple test program: the program creates four tasks, each task counting and turning on/off one of the LEDs at different frequencies.

To build the code basic example use the following commands:

```
    cd nano-RK/projects/basic_tasks
    make
```

The long output should end with something like

```
Errors: none
Platform: micaZ
-------- end --------
```

## 4. Write the Basic Example to the MICAz mote

This is easily done by the following command

```
make program
```

The output should be something like

```
Firmware Version: 49.56
Atmel AVR ATmega128 is found.
Firmware Version: 49.56
Uploading: flash
Fuse High Byte set to 0xd8
Fuse Extended Byte set to 0xff
```

The sensor board LEDs should start flashing.

## 5. Capturing the Sensor Board Output

The test program "basic_tasks " in addition to turning on/off the LEDs also prints the counts. The output is transmitted to the host machine through the USB connection.

To read the output we need to read the data transmitted at the USB connection. Reading the USB connection can be done using tools like `minicom`.

To use `minicom` to read the output follow these steps:

Run `minicom` in configuration mode by using the command

```
minicom -s
```

Select “Serial Port Settings”

Press A and set the “Serial Device” to “/dev/ttyUSB1”

If the baudrate (Bps/Par/Bits ) is not already set to 115200 8N1, press E and set it to 115200 8N1

Set both Hardware Flow Control and Software Flow Control to No, by pressing F and G.

Store the setup as default setup “Save setup as dfl”

Exit from `minicom`

The steps listed above need be done only once. The settings are stored as the default setting and will be in effect for future runs.

Run `minicom` to read the output by using the following command:

```
minicom -m
```

The output will be something like

```
Welcome to minicom 2.2
OPTIONS: I18n
Compiled on Sep 25 2007, 06:13:56.
Port /dev/ttyUSB1
Press ESC,Z for help on special keys
Task1 cnt=2414
Task2 signed cnt=-1208
Task1 cnt=2415
Task1 cnt=2416
Task4 cnt=303
Task3 cnt=605
```

To restart the test program on the board press the reset button on the board (this is the small button beside the USB connection on the board).  The output will be something like

```
Nano-RK Version 101
My node's address is 0
Task1 PID=1
Task1 cnt=0
Task1 cnt=1
Task1 cnt=2
Task2 signed cnt=-2
Task1 cnt=3
Task1 cnt=4
Task3 cnt=2
```

To exit `minicom` press `Esc` then `X` (do not press them together).