Exercise in Embedded Computing: Contiki Basics

Sivan Toledo, Tel-Aviv University

November 26, 2009

This exercise covers the basic facilities of Contiki, an operating system for embedded systems. The exercise refers to to the installation of Contiki and the compiler on our servers, but you can also download a virtual machine image[1] that allows you to experiment with Contiki on your own computer.

1. Plug the tmote sky board (which is called simply *sky* in the Contiki code and documentation) to a USB port. Copy the directory `/users/courses/embedded/exercises/contiki-basics` and its contents to your home directory. In this directory give the command `make exercise.upload`. The command should build a binary containing the code in `exercise.c` and the operating system itself and upload it to the board. The sources of the operating system, along with many useful example programs, are stored in `/users/courses/embedded/contiki-2.3`. The makefile that you copied into your own directory points to this location. The documentation for Contiki is at `http://www.sics.se/contiki` (click on "documentation").

2. Run the command `make login`. It connects to the board through a USB serial port. Press the reset button on the board. You should see a Contiki startup message, followed by "`Hello`!", which is a string that our program prints. Press control-C to exit from the terminal emulation program.

3. Copy the source file `exercise.c` to `solution.c`, and modify `Makefile` so that it builds from it, rather than from `exercise.c`. Add your name to the welcome message, build the binary and upload, and test it.

4. The main loop in the program waits for events but does not do anything. Modify it so that it prints a message every 4 seconds. You will need to use an event timer (under "Modules->Contiki system->Event timers" in the documentation). Time is specified using the constant `CLOCK_SECOND`. You will need the macro `PROCESS_WAIT_EVENT_UNTIL` to wait for the timer expiration (or for any other specific event). Test the code.

5. Add another process that blinks the green LED once per second, and add it to the list of automatically started processes. You will need to include

[1] Download from `http://www.sics.se/contiki/instant-contiki.html`.

"dev/leds.h"; the relevant functions are leds_on, leds_off, leds_toggle, and the argument should be LEDS_GREEN.

6. Add a third process that blinks the red LED 10 times per second, but start it only 4 seconds after first two (that is, after the hello process finishes its first wait). See "Modules->Contiki System->Contiki processes" for the API to start a process from another process.

7. Add a process that prints a message every time the user button on the board is pressed. See examples/sky/test-button.c for a useful example. The message that is printed must indicate the number of times the button was pressed so far.

8. Processes can post events to each other. Modify the process that prints a welcome messages every 4 seconds so that it allocates a new event type and posts it to a new process every time it prints a message. There are several types of events in contiki, to denote message arrival, timer expiration, and so on; they are represented by integers. We need a new event type (a new number to represent our events), which we allocate using process_alloc_event. This needs to be done once. We post events to the new process using process_post. The new process should print a message every time it receives an event of the new type.

9. Contiki supports both event timers, which we have already used, and real-time timers. A real-time timer does not post an event when it expires, but invokes a function (from within the hardware timer's interrupt service routine). This provides better control over scheduling than an event timer. Use one of your existing processes to schedule a real-time task for a second later. The task should toggle the blue LED and re-schedule itself for a second later. This should cause blinking of the blue LED.

10. Optional: The Contiki build system allows you to override operating system sources simply by copying them into your working directory and modifying them. The build process will use the modified local versions. There is no need to modify the makefile. Copy the file cpu/msp430/watchdog.c into your working directory. The file contain functions that start, stop, and pacify the watchdog timer (a timer whose role is to reset the processor unless the program pacifies it periodically). Modify the code so that Contiki does not use the watchdog timer at all. You still need to stop it in the initialization function. This might be useful because on the MSP430 you can also use the watchdog timer to generate periodic interrupts.