Timer Programming Tutorial

Timers in Contiki

- Contiki OS provides 4 types of timers
  - Simple timers: Two basic timers for which the application should check itself if the timer has expired (see “core/sys/timer.h” and “core/sys/stimer.h”)
  - Callback timer: When this timer expires, it will callback a given function (see “core/sys/ctimer.h”)
  - Event timer: When this timer expires, instead of calling a function, it posts an event (see “core/sys/etimer.h”)
  - Real-time timer: Used to handle the scheduling and execution of real-time tasks; only one such timer is available (see “core/sys/ rtimer.h”)
- For more details, see the reference below
  - https://github.com/contiki-os/contiki/wiki/Timers

Timer Simulation Example

- The simplest way to open the simulation is to select "Timer Simulation" in the IoTrain-Sim interface
- Alternatively, you can open it manually as follows
  - Open Cooja
  - Click File > Open simulation > Browse…
  - Go to the folder "iotrain-sim/database/fundamental_training/single_node/actuation_control/timer/simulation"
  - Select the file "timer.csc"
  - Click Open
- Once the simulation control window appears, click the "Start" button
  - The mote will print a message every 3 seconds
  - This simulation runs in real time and will stop automatically after 20 seconds

Source Code Commentary

- Print a message at regular time intervals by using an event timer
  - Source code: iotrain-sim/database/fundamental_training/single_node/actuation_control/timer/simulation/timer-ex.c *NOTE: This file was not named “timer.c” in order to avoid a conflict with the name of the Contiki timer implementation*

```
#include "contiki.h"
#include ”sys/etimer.h"
#include <stdio.h>

#define SECONDS 3
/*---------------------------------------------------*/
PROCESS(timer_process, "timer process");
AUTOSTART_PROCESSES(&timer_process);
/*---------------------------------------------------*/
PROCESS_THREAD(timer_process, ev, data)
{
  PROCESS_BEGIN();
  static struct etimer et;
  while (1)
  {
    etimer_set(&et, CLOCK_SECOND *
SECONDS); ❶
    PROCESS_WAIT_EVENT(); ❷
    if (etimer_expired(&et))
    {
      printf(”Timer expired\n");
      etimer_reset(&et);
    }
  }
  PROCESS_END();
}
```

Source Code Commentary (cont.)

❶ The constant CLOCK_SECOND indicates the number of microcontroller ticks per second, and should be multiplied with the intended number of seconds in order to express the duration

- As Contiki runs on different hardware platforms, the value of the CLOCK_SECOND constant may also differ

❷ The function PROCESS_WAIT_EVENT() waits for any event to happen

- Once an event happens, we need to check if it is a "timer has expired" type of event for our timer

Exercise

- Write a program that blinks the blue LED once per second for a period of 10 seconds
- Verify the program by running it in Cooja and checking the status of the blue LED
- Hint
  - Remember to modify the Makefile by adding the new filename to “CONTIKI_PROJECT”