Timers

Ramona Marfievici
(ramona.marfievici@cit.ie)

Some slides originally by Carlo Boano

1st Laboratory

- Verify that your system works using `hello-world` example
- Run `hello-world` in Cooja
- Run `hello-world` on a TMoteSky node

```
murphy@murphy-Latitude-E5450:~/Desktop/contiki/contiki/examples/hello-w
ello-world.native
Contiki-2.6-2443-g63bb46f started with IPV6, RPL
Rime started with address 1.2.3.4.5.6.7.8
MAC nullmac RDC nullrdc NETWORK sicslowpan
Tentative link-local IPv6 address fe80:0000:0000:0000:0302:0304:0506:07
Hello, world
```

Let's add complexity to the code!

Timers in Contiki

Clock module

- Handle system time in clock ticks
- Number of clock ticks per second is platform dependent and is specified using CLOCK_SECOND
- Converts seconds into the tick resolution of the platform
  - TMoteSky: 64 ticks per second, 1 tick = 15.625 ms

Timer modules

- Timer, stimer, etimer, ctimer, rtimer
- Timer, stimer
  - Simplest form of timers
  - Check if a time period has passed
  - Resolution: timer-ticks, stimer-seconds
- rtimer
  - Scheduling of real-time tasks

Timers in Contiki

ETimer

- Active timer, sends an **event** when it expires
- Declaration of timer

```
static struct etimer et;
```

- Activate and deactivate the timer

```
etimer_set (&et, AMOUNT_OF_TICKS);
etimer_stop (&et);
```

- Set `AMOUNT_OF_TICKS` as a function of `CLOCK_SECOND`
- Keep track of expirations

```
etimer_pending(); //is there a non-expired event?
clock_time_t next_expiration_time();
```

- `/contiki/core/sys/etimer.h`

Periodic Hello World! with etimer

```
#include "contiki.h"
#include <stdio.h>
PROCESS (hello_world_proc, "Hello world process");
AUTOSTART_PROCESSES (&hello_world_proc);
PROCESS_THREAD(hello_world_proc, ev, data)
{
      PROCESS_BEGIN();
      static struct etimer et;
      while(1){
      etimer_set (&et, CLOCK_SECOND*2);
      PROCESS_WAIT_EVENT_UNTIL (etimer_expired(&et));
      printf("Hello World!\n");
      }
      PROCESS_END();
}
```

We add an etimer that fires every second

We wait for the timer to expire

Timers in Contiki

CTimer

- Active timer, calls a function when it expires
- Declaration of timer and callbacks

```
static struct ctimer timer1;
static void ctimer1_callback(void *ptr) {…}
```

- Activate and deactivate the timer

```
ctimer_set (&timer1, AMOUNT_OF_TICKS, ctimer1_callback, NULL);
etimer_stop (&timer1);
```

- Set `AMOUNT_OF_TICKS` as a function of `CLOCK_SECOND`
- `/contiki/core/sys/ctimer.h`

Periodic Hello World! with ctimer

```
#include "contiki.h"
#include <stdio.h>
static struct ctimer timer;
static void tout_cback(void *ptr) {
  printf("%s", (char *)ptr);
  ctimer_set (&timer, EXPIRATION, tout_cback, ptr);
}
PROCESS (hello_world_proc, "Hello world process");
AUTOSTART_PROCESSES (&hello_world_proc);
PROCESS_THREAD(hello_world_proc, ev, data)
{
  PROCESS_BEGIN();
  ctimer_set (&timer, CLOCK_SECOND*2, tout_cback, "Hello World!\n");
  while(1){
      PROCESS_WAIT_EVENT();
  }
  PROCESS_END();
}
```

ctimer declaration

ctimer callback

ctimer activation

Exercise

- Run `hello-world` with etimer and ctimer

Blink the LED!

```
#include “contiki.h”
#include “dev/leds.h” //necessary for the LEDS
#include <stdio.h>
PROCESS (blink_process, “blink process”);
AUTOSTART_PROCESSES (&blink_process);
PROCESS_THREAD(blink_process, ev, data)
{
      PROCESS_BEGIN();
      leds_on(4);     // red = 4, yellow = 1, blue = 2
      PROCESS_END();
}
```

Exercises

- Blink the LED
- Modify the program to blink all the LEDs (with no time constraints)
- Write a program that blinks the LEDs every 4 seconds

Change the state of LEDs using the button

```
...#include "dev/button-sensor.h" //necessary for the buttons
...PROCESS (blink_process, "blink process");
PROCESS_THREAD(blink_process, ev, data)
{
    PROCESS_BEGIN();
        SENSORS_ACTIVATE(button_sensor);
        leds_on(LEDS_ALL); //at boot time all leds are on
        while(1){
        static unint8_t push = 0; //counter for pushes
        PROCESS_WAIT_EVENT_UNTIL((ev==sensors_event) && (data == &button_sensor));
        if (push % 2 == 0){
        leds_toggle(LEDS_ALL);
        printf("[%d] Turning OFF all leds…[DONE]\n", push);
        push++;
        } else {
        leds_toggle(LEDS_ALL);
        printf("[%d] Turning ON all leds…[DONE]\n", push);
        push++;
        }
        if (push = 255) {push =0;} //prevents overflowing
}
PROCESS_END();}
```

Exercises

Exercise 1.

Write a program that loops indefinitely, waits for an event, then checks if the timer has expired, and if so, toggles the LEDs and outputs a message.
Test the program on your mote.

Exercise 2.

Write a program that loops indefinitely, waits for an event, then checks whether this event is a button press and if the timer has expired. Upon button press, toggle the LEDs and print out “Button press!”. When the timer expires, simply toggle the LEDs and print out “Timer!”.
Test the program on your mote.