Laboratorio 1: Introducción a Contiki OS

Redes de Sensores Inalámbricos
IIE, Facultad de Ingeniería, UDELAR

Tabla de Contenidos.

1 Introducción

Este laboratorio es introductorio al sistema operativo Contiki. El énfasis estará en comprender el funcionamiento básico del sistema operativo y el manejo de sensores.

El laboratorio debe ser completado **por cada participante de la clase**. Si bien se recomienda el intercambio de ideas entre compañeros, es obligatorio que cada uno realice los trabajos en forma individual en su computadora.

Cada grupo de tres estudiantes contará con dos nodos RE-Mote. Para facilitar el desarrollo de las actividades, se podrá utilizar la herramienta Cooja para simular (disponible en el repositorio de Contiki). La simulación con Cooja se realizará utilizando los nodos z1, ya que los RE-Mote no están soportados.

2 Objetivo

El objetivo de este laboratorio es familiarizarse con el funcionamiento básico de Contiki, realizando lecturas desde los sensores internos de un nodo, controlando LEDs y experimentando con la comunicación entre procesos concurrentes mediante *post* de eventos.

2.1 Objetivos de aprendizaje

Se espera que el estudiante logre como mínimo:

- Entender la estructura de declaración y definición de procesos en Contiki.
- Integrar bibliotecas y comprender la utilización básica de sensores.
- Utilizar los timers de Contiki adecuadamente.
- Comprender la comunicación entre procesos mediante *post* de eventos.

3 Lecturas y actividades previas

Para la realización de las prácticas de laboratorio el estudiante cuenta con una máquina virtual provista por los docentes, incluyendo los programas y las configuraciones necesarias para el curso. Para ejecutar la máquina virtual, se deberá instalar el software VMware disponible en: http://www.vmware.com/go/downloadplayer/.

El usuario de la máquina virtual es ‘user’ y la contraseña es ‘user’.

Para el laboratorio deberán leer el capítulo 3 "Introduction to Contiki" de “Internet of Things in five days” [2] y resolver los ejercicios de la clase de Contiki [1].

Para profundizar sobre el sistema operativo Contiki y su funcionamiento, se sugiere consultar el Wiki de Github [3].

4 Fundamentos

A continuación se presentan los fundamentos básicos para el manejo de los LEDs y sensores disponibles. Se hace referencia a directorios en el árbol de Contiki, que siempre están referidos al directorio raíz del árbol.

4.1 LEDs

Para utilizar los LEDs se debe incluir la biblioteca correspondiente:

```
#include "dev/leds.h"
```

Se puede observar que algunas de las funciones para trabajar con los LEDs son:

```
void leds_set(unsigned char leds);
void leds_on(unsigned char leds);
void leds_off(unsigned char leds);
void leds_toggle(unsigned char leds);
```

Investigar qué colores están disponibles en la plataforma RE-Mote (ver el archivo *platform/zoul/remote-revb/board.h*).

4.2 Sensores

Las características de todos los sensores se pueden encontrar en *core/lib/sensors.h*.
Allí se define una estructura general a todos los sensores, ya sea el botón de usuario, el sensor interno de temperatura u otros sensores.
Para poder atender eventos de cierto sensor, éste debe estar activado. Para ello se definen las siguientes macros:

```
#define SENSORS_ACTIVATE(sensor) (sensor).configure(SENSORS_ACTIVE, 1)
#define SENSORS_DEACTIVATE(sensor) (sensor).configure(SENSORS_ACTIVE, 0)
```

Existen particularidades de los sensores en cada plataforma, debido a diferencias de hardware. Esto implica que distintas plataformas tienen archivos específicos para redefinir algunas características de ciertos sensores. Estos detalles se ven más adelante para cada uno de los sensores utilizados.

4.2.1 Botón de usuario

El nodo RE-Mote está equipado con dos botones: un botón de reset y un botón de usuario.
El botón de usuario es un sensor.
La interfaz pública del botón de usuario se encuentra en el directorio *core/dev/button-sensor.h*.
Para poder usar el botón debemos incluir la biblioteca de la siguiente forma:

```
#include "dev/button-sensor.h"
```

Dado que el botón es un sensor, debe ser activado:

```
SENSORS_ACTIVATE(button_sensor);
```

Luego para leerlo:

```
if (ev == sensors_event) {
      if (data == &button_sensor) {
            if (button_sensor.value(BUTTON_SENSOR_VALUE_TYPE_LEVEL) ==
                  BUTTON_SENSOR_PRESSED_LEVEL) {
                  printf("Button pressed\n");
            } else {
                  printf("...and released!\n");
            }
      }
}
```

Consideraciones para la simulación con Cooja de LEDs y botón de usuario

El nodo RE-Mote no está soportado en Cooja. Por tanto, se recomienda utilizar el nodo z1 para simulación. El nodo z1 también está equipado con LEDs y botón de usuario. Las diferencias con RE-Mote se deben encontrar comparando lo explicado para RE-Mote con los contenidos de los archivos correspondientes a la plataforma z1. Estos son:
platform/z1/board.h
platform/z1/dev/button-sensor.h y platform/z1/dev/button-sensor.c

4.2.2 Sensor de temperatura

El microcontrolador del nodo RE-Mote cuenta con un sensor interno de temperatura. Para utilizarlo se debe tener en cuenta que el valor analógico de la temperatura interna se convierte con un ADC. Esto es similar para cualquier sensor analógico que se quiera implementar. Los dos archivos que implementan el driver para el sensor interno están en *platform/zoul/dev/* y son *zoul-sensors.h y zoul-sensors.c*
Se debe incluir en el programa el archivo *zoul-sensors.h*, análogamente a como se incluyeron las bibliotecas anteriores, y además se debe agregar el archivo externo *zoul-sensors.c* en el Makefile del proyecto (ver [1]):

```
CONTIKI_SOURCEFILES += zoul-sensors.c
```

Antes de usar este sensor, hay que activarlo:

```
SENSORS_ACTIVATE(cc2538_temp_sensor);
```

Después es posible leer el valor reportado de la temperatura en mili grados Celsius:

```
val = cc2538_temp_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED));
```

Consideraciones para la simulación con Cooja del sensor de temperatura

Para simular se utiliza el nodo z1. Análogamente a como se explicó para RE-Mote, se debe incluir *platform/z1/dev/temperature-sensor.h*. No es necesario agregar al Makefile:

```
CONTIKI_SOURCEFILES += "temperature-sensor.c"
```

La activación del sensor y la lectura de la temperatura, son más simples:

```
SENSORS_ACTIVATE(temperature_sensor);
temperature_sensor.value(0);
```

Observaciones:

- `temperature_sensor.value(0)` en realidad no usa el int que se le pasa como parámetro, en este caso 0.
- Los resultados numéricos que se obtienen de simular un sensor de temperatura en Cooja no son representativos. Esta simulación nos permite probar sólo en parte el código desarrollado.

4.2.3 Sensor de tensión de alimentación

El microcontrolador cuenta con un sensor del nivel de tensión a la que está alimentado. Para comprobar el valor reportado de esa tensión, se debe consultar el valor en mili Volts dado por el sensor:

```
val = vdd3_sensor.value(CC2538_SENSORS_VALUE_TYPE_CONVERTED);
```

Nota: este sensor no está en la plataforma z1, por tanto se omite la simulación opcional.

4.3 Comunicación entre procesos mediante eventos

Desde el proceso que inicia la comunicación se realiza el pasaje de un puntero a los datos con `process_post`, teniendo en cuenta la forma en que se deben recibir los datos en el otro proceso, que por ejemplo incluirá una sección `PROCESS_WAIT_EVENT` o `PROCESS_WAIT_EVENT_UNTIL` (repasar los ejemplos de clase en caso de tener dudas [1]).

4.4 Cargar el programa en el RE-Mote

Para comprobar que el RE-Mote está conectado correctamente al PC y a la máquina virtual, ir al directorio *examples/hello-world/* y ejecutar el comando:

```
$ make TARGET=zoul motelist
```

Si el nodo está conectado correctamente se verá en el listado generado en la terminal.

Compilar y cargar el programa de ejemplo:

```
$ make hello-world.upload TARGET=zoul BOARD=remote-revb
```

Abra una terminal serial para ver la salida del nodo:

```
$ make TARGET=zoul login
```

Si no logra ver nada, presione el botón de reset del nodo para reiniciar el programa.

4.5 Simular el programa en z1 utilizando Cooja (opcional)

Primero es importante familiarizarse con Cooja, la herramienta de simulación de Contiki. Para eso se recomienda:

1. Leer el material de referencia disponible en el wiki [4] (pueden ignorar lo referido a external tools de la sección "Getting started" y la sección entera "Configuration Wizard").
2. Hacer una simulación sencilla siguiendo los pasos de "Create a Hello World simulation" de [4].

Una vez que se tenga un programa que se quiere probar en simulación, se debe crear una nueva simulación y agregar un nodo z1 especificando el programa a cargar. Se corre la simulación.

Visualizar LEDs y pulsar botón en Cooja

En la simulación, se puede hacer click derecho sobre el nodo → Mote tools for Z1 → Mote Interface Viewer...
Esto despliega una ventana que permite observar los LEDs y pulsar el botón, entre otras funcionalidades.
Otra manera de visualizar los LEDs es ir a View en la ventana Network y seleccionar LEDs.

5 Procedimientos y Tareas

Compilar para la plataforma RE-Mote y ejecutar en el mismo cada una de las aplicaciones que se desarrollen durante el laboratorio. Se debe prestar atención a la salida del compilador de forma de solucionar los eventuales errores que surjan y entender los distintos warnings.

Se realizará una aplicación que realice mediciones periódicas con el sensor interno de temperatura, promediando estas mediciones e imprimiendo el valor obtenido (usando printf). Concurrentemente se monitorea el botón de usuario, llevando la cuenta de la cantidad de veces que el mismo es presionado e imprimiendo esta información.

Tarea 1

Realizar una aplicación que mida periódicamente la temperatura con el sensor interno del MCU, promedie las muestras e imprima el valor obtenido.
Además, se deberá medir periódicamente la tensión de alimentación del nodo e imprimir el valor obtenido.
El proceso deberá conmutar los LEDs cada vez que se adquieran los datos.
NOTA:

El promedio de las muestras podría implementarse de varias maneras. Una de ellas es, por ejemplo, utilizando un filtro IIR de primer orden: $y[n] = \alpha . x[n] + (1 - \alpha) . y[n - 1]$.

Tarea 2

Realizar una nueva versión de la aplicación de la Tarea 1 pero ahora con dos procesos:

1. Este proceso medirá periódicamente la temperatura con el sensor interno del MCU, promediará las muestras, medirá la tensión de alimentación y enviará los valores obtenidos realizando *post* de eventos al proceso número 2.
2. Este proceso se encargará de publicar los valores, obtenidos por el proceso número 1, imprimiéndolos. Además, encenderá y apagará los LEDs cada vez que se adquieren los datos.

Tarea 3

Agregar un tercer proceso, concurrente con los anteriores, que capture los eventos que se producen al apretar el botón de usuario, lleve la cuenta de la cantidad de veces que se ha presionado hasta el momento y envíe esta información al proceso encargado de publicar. Se debe modificar el segundo proceso (realizado en la Tarea 2) para que imprima el mensaje correspondiente según el evento que recibe (Botón, Temperatura o Tensión).

6 Entregables

Se deberá entregar un archivo comprimido que contenga lo siguiente:

1. carpeta Tarea 1 incluyendo Makefile, .c, .h,
2. carpeta Tarea 2 incluyendo Makefile, .c, .h,
3. carpeta Tarea 3 incluyendo Makefile, .c, .h.

Es imprescindible que el código esté comentado adecuadamente: cada línea o líneas consecutivas que sean agregadas, deberán ser acompañadas de un comentario que explique su propósito.

En caso de haber realizado simulaciones, los archivos de simulación generados se agregarán al archivo comprimido a entregar.

Las entregas se realizarán a través de la **plataforma EVA del curso**.

7 Referencias

[1] “Clase Contiki OS: Parte 1“ [Online]. Available: https://eva.fing.edu.uy/course/view.php?id=499

[2] A. L. Colina, A. Vives, A. Bagula, M. Zennaro, and E. Pietrosemoli, *IoT in five Days*. E-Book, Jun. 2016, rev 1.1. [Online]. Available: https://github.com/marcozennaro/IPv6-WSN-book/releases/

[3] “Contiki Wiki” [Online]. Available: https://github.com/contiki-os/contiki/wiki