Laboratorio 1: Introducción a Contiki-NG

1 Introducción

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

El trabajo en este laboratorio es grupal, sin embargo sugerimos que cada participante trabaje con su propia computadora, implementando las funcionalidades discutidas dentro del grupo.

El grupo contará con al menos tres nodos, pudiendo consistir en launchpads (LP) y sensortags (ST) basados en el mismo SoC: CC1350 o CC2650. Es decir, tres nodos de tipo LP-CC1350 y ST-CC1350 o tres nodos de tipo LP-CC2650 y ST-CC2650.

2 Objetivos

2.1 Objetivos generales

- Familiarizarse con el funcionamiento básico de Contiki.
- Familiarizarse con el hardware proporcionado para los laboratorios.

2.2 Objetivos de específicos

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

3 Materiales

- Hoja de referencia cheat sheet.
- Tres nodos LP-CC1350/ST-CC1350 o LP-CC2650/ST-CC2650 (opcional J-Link EDU Mini, en caso de usar sensortag)

4 Fundamentos

A continuación se presenta Contiki-NG y 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 Primeros pasos

- Estudiar la siguiente documentación, y en caso que corresponda, seguir los instructivos (se sugiere tener a mano la hoja de referencia cheat sheet):

1. Las siguientes selecciones de Programming Contiki-NG:
   - Repository structure.
   - Processes and events.
2. Los Tutoriales:
   - Hello, World!.
   - En el paso "Running the example on a real device" tener en cuenta que tiene que modificar el TARGET y BOARD de acuerdo a la plataforma disponible. Se sugiere consultar la cheat sheet para ver cómo determinar los valores de TARGET y BOARD a utilizar.

   Nota: Para indentificar el puerto al cual está conectado el nodo (PORT) seguir las instrucciones de la guía.
3. Estudiar el código de `hello-world.c`

4.2 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_single_on(leds_num_t led);
void leds_single_off(leds_num_t led);
void leds_single_toggle(leds_num_t led);
void leds_on(leds_mask_t leds);
void leds_off(leds_mask_t leds);
void leds_toggle(leds_mask_t leds);
```

Investigar qué colores están disponibles en las plataformas LP-CC1350 y LP-CC2650 (ver el archivo `arch/platform/simplelink/cc13xx-cc26xx/launchpad/board-conf.h`).

4.3 Sensores

Las características de todos los sensores se pueden encontrar en `os/lib/sensors.h`.

Allí se define una estructura general a todos los sensores, ya sea el sensor interno de temperatura, voltaje 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.

4.3.1 Botones de usuario

Los nodos launchpad y sensortag tienen un botón de reset y dos botones de usuario. La interfaz pública del botón de usuario se encuentra en el directorio `core/dev/button-hal.h`.

Para poder usar el botón debemos incluir la biblioteca de la siguiente forma:

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

Dada esta implementación particular, para su uso no necesita ser activado como los demás sensores.

Para leerlo:

```
if(ev == button_hal_press_event) {
   btn = (button_hal_button_t *)data;
   printf("Press event (%s)\n", BUTTON_HAL_GET_DESCRIPTION(btn));
   if(btn == button_hal_get_by_id(BUTTON_HAL_ID_BUTTON_ZERO)) {
      printf("This was button 0, on pin %u\n", btn->pin);
   }
} else if(ev == button_hal_release_event) {
   btn = (button_hal_button_t *)data;
   printf("Release event (%s)\n", BUTTON_HAL_GET_DESCRIPTION(btn));
} else if(ev == button_hal_periodic_event) {
   btn = (button_hal_button_t *)data;
   printf("Periodic event, %u seconds (%s)\n", \
    btn->press_duration_seconds, \
    BUTTON_HAL_GET_DESCRIPTION(btn)\
   );
   if(btn->press_duration_seconds > 5) {
      printf("%s pressed for more than 5 secs. Do custom action\n",
         BUTTON_HAL_GET_DESCRIPTION(btn));
   }
}
```

4.3.2 Sensor de temperatura

Los microcontroladores de los nodos basados en CC1350 y CC2650 cuentan 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 archivos que implementan el driver para el sensor están en `arch/platform/simplelink/cc13xx-cc26xx/` y son `batmon-sensor.h` y `batmon-sensor.c`. `batmon-sensor.c` no es neccesario incluirlo en el `Makefile` porque se encuentra ya realizado en `Makefile.cc13xx-cc26xx`.

Para usarlo se debe activar mediante:

```
SENSORS_ACTIVATE(batmon_sensor);
```

Luego para leer el valor se realiza mediante:

```
val = batmon_sensor.value(BATMON_SENSOR_TYPE_TEMP);
```

Se obtiene el valor en °C.

4.3.3 Sensor de tensión de alimentación

El microcontrolador es capaz de medir el nivel de tensión de la que está alimentado. También se incluye en el archivo `batmon-sensor.h`.

Se activa mediante:

```
SENSORS_ACTIVATE(batmon_sensor);
```

Se debe activar una sola vez ya que es el mismo sensor que el de la temperatura.

Para leer el valor se realiza mediante:

```
Val = (batmon_sensor.value(BATMON_SENSOR_TYPE_VOLT) * 125) >> 5;
```

4.4 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 la documentación y los ejemplos de clase en caso de tener dudas).

5 Actividades

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. La aplicación se creará progresivamente realizando las siguientes tareas.

5.1 Tarea 1: medida y promedidado de temperatura

Realizar una aplicación que mida periódicamente la temperatura con el sensor interno del microcontrolador, 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: $y[n] = \alpha.x[n] + (1 - \alpha).y[n - 1]$.

5.2 Tarea 2: dos procesos

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 microcontrolador, 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 adquieran los datos.

5.3 Tarea 3: tres procesos

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.

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

7 Referencias

- Kurniawan, Agus. 2018. "Practical Contiki-NG". Pract. Contiki-NG. (disponible en https://foco.timbo.org.uy)
- Colina, Antonio Linan, Alvaro Vives, Antoine Bagula, Marco Zennaro, y Ermanno Pietrosemoli. 2016. IoT in five Days. E-Book. https://github.com/marcozennaro/IPv6-WSN-book/releases/.
- Contiki-NG documentación
- Contiki-NG cheat sheet
- "Clase Contiki-NG: Parte 1" [Online]. Disponible en: https://eva.fing.edu.uy/course/view.php?name=rsi

Los materiales de este curso fueron parcialmente financiados por:

NEON Network of Competence on Internet of Things

Co-funded by the Erasmus+ Programme of the European Union