REDES DE COMPUTADORES II – SEGUNDO SEMESTRE 2010
Tarea de SunSPOT: "Punch-o-Meter"
Pablo Ahumada Díaz, Rol: 2504052-k
Jorge Cápona González, Rol: 2521055-7

1. **Contexto**

   La tarea se desarrolla con dos grandes objetivos: globalmente se busca conocer la programación de dispositivos inalámbricos y, particularmente, familiarizarse con el ambiente de desarrollo del lenguaje Java para programar sensores inalámbricos, que en este caso son SunSPOT (motes para redes de sensores inalámbricos desarrollados por Sun Microsystems).

2. **Ambiente de Desarrollo**

   La tarea se desarrolló completamente en Ubuntu 10.10 32 Bits. En principio se intentó desarrollar en Ubuntu 10.04 64 Bits, se instalaron los software necesarios para trabajar con los SunSPOT, pero al momento de conectar el mote al computador, éste no lo reconocía.

   Para interactuar con los SunSPOT desde el computador, se necesita lo siguiente:

   - ✓ JVM de Java
   - ✓ IDE para Java; se sugiere Netbeans debido a que permite cargar los proyectos directamente (aunque también se puede hacer por consola)
   - ✓ JDK más SDK del SunSPOT

   SunSPOT viene con un programa de instalación – Sun Spot Manager – que señala todo lo necesario que se debe instalar, en caso de no estarlo, para poder trabajar con estos los nodos.
   En relación al hardware, sólo son necesarias las unidades físicas de SPOT (pues también hay virtuales).

3. **Desarrollo**
   I. Análisis Previo:
      Lo primero fue estudiar los códigos demos incorporados en la instalación del kit. Principalmente se analizaron los ejemplos sobre el uso del acelerómetro 3D, led, switch y la conexión inalámbrica. Posterior a ello se analizó la aplicación de muestra SpotBounce, en la que se utilizan todas las funciones que se debían utilizar para realizar la tarea. La solución a la tarea del año pasado también fue de mucha utilidad, pues en ella ya se había implementado la comunicación de los dos nodos con los mismos fines que se buscaban ahora.

II. Implementación:

Una vez estudiado todo lo anterior, no fue difícil realizar los cambios necesarios para conseguir nuestros objetivos, los que se detallan a continuación:

- Emisor:
  doKinematics (BallAnimator.java): se modificó para que enviara la máxima aceleración percibida en el eje X cada 200 muestras.
- Receptor:
  switchPressed (SPOTBounce.java): esta aplicación es la que detecta que switch ha sido presionado y qué hacer en cada caso. Fue modificada para que cuando se presione el switch 1 apague todos los leds que se encuentren encendidos.
  doKinematics (BallAnimator.java): se modificó para que prendiera los leds desde el cero al siete (izquierda a derecha, tres primeros leds verdes, los siguientes tres azules y los últimos dos rojos) a medida que la aceleración proveniente del emisor fuese mayor. Es decir, la aceleración entrante se compara según unos valores pre-establecidos para decidir la cantidad de leds que se encenderán.

4. **Resumen de la Ejecución (Instrucciones)**

El receptor se deberá poner como se muestra en la Fig.1. Al encenderlo el primer led – abajo – se encenderá de color azul. Se tendrá que presionar una vez el switch 1 (encerrado en un círculo rojo en la Fig.1), cuando el led 1 se torne verde, el receptor estará en condiciones de desplegar la información proveniente desde el emisor.

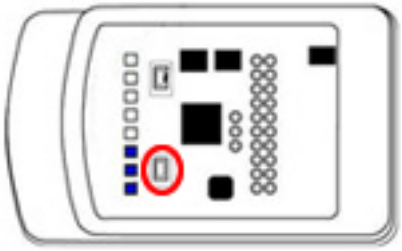

Figura 1. Receptor. Nodo Despliegue.

El emisor (nodo puño) enviará la máxima aceleración medida en el eje X cada 200 muestras. Una vez la medición de la aceleración del emisor haya sido desplegada por el receptor, se deberá presionar nuevamente el switch 1 para apagar los leds y recibir nuevamente otra medición para ser desplegada.

5. **Diagrama de Clases**