UNIVERSIDAD POLITÉCNICA DE VALENCIA

Departamento de Comunicaciones

Simulador web de redes de sensores para la automatización industrial

Autor: Pablo Giménez Salazar

Director: Dr. Carlos Palau Salvador

Fecha comienzo: 09/2011

Lugar de trabajo: Dpto. de Comunicaciones

Objetivos

- Diseñar y desarrollar un simulador capaz de representar el funcionamiento real de varios sensores, tanto fijos como móviles, y los datos que miden.
- Usar el SOS como base de datos de observaciones de los sensores, para lo cual es necesario enviar y recibir mensajes siguiendo el estándar O&M.
- Aplicar SWE a un sistema de prevención de riesgos laborales.

- Buscar simuladores que pudieran realizar la funcionalidad necesaria.
- Colaborar con otros equipos como parte de un mismo proyecto.
- Publicar un artículo como parte del importante de un proyecto de investigación.

Metodología

Se han llevado a cabo tres partes importantes a la hora de desarrollar el simulador. La primera de ellas ha sido conocer el funcionamiento del SOS, ya que ha sido necesario registrar nuevos sensores, insertar las mediciones que iban generando, y actualizar las propiedades de un sensor. La segunda parte ha sido el propio simulador, que a partir de un fichero de configuración genera todos los sensores necesarios y empieza a enviar datos al SOS. Por último para que sea más sencillo de utilizar se ha desarrollado un interfaz gráfico que facilite todas las funcionalidades.

Resultados

El simulador desarrollado es una herramienta útil para facilitar la planificación y desarrollo de escenarios de seguridad para los trabajadores en entornos industriales, ya que evita la implementación de muchos sensores reales dentro de una fábrica para identificar situaciones de riesgo.

El simulador es capaz de generar los datos en tiempo real de sensores móviles y fijos, y enviar todas las mediciones simuladas a un Servicio de Observación del sensor (SOS) que actúa como sumidero común. Como la interfaz con los sensores de SOS se ofrece como una interfaz web, cualquier sensor real conectado a una red IP puede interactuar con el SOS, para ello únicamente es necesario registrarse primero y luego el enviar las mediciones y las actualizaciones. Esto es posible gracias a que se han seguido los estándares propuestos por el SWE para permitir la integración de cualquier tipo de sensor.

Los datos generados por el simulador de cambian en función de la elección del modos de simulación entre los definidos. Es posible introducir anomalías, que representen un evento temporal o un riesgo que se quiere detectar. También se puede especificar una probabilidad de error, que representa el mal funcionamiento de un sensor en un instante de tiempo para entrenar al sistema a descartar ese tipo de fallos.

Líneas futuras

- Añadir nuevos modelos de simulación de sensores que se adapten mejor a la realidad física de los sensores y añadir nuevos modelos de sensores.
- Mejorar el diseño de la interfaz gráfica para que sea más interesante visualmente para los usuarios.
- Integrar el simulador en otros proyectos.

Publicaciones

Semantic Sensor Web Simulator for Factory Automation. Pablo Giménez, Benjamín Molina, Carlos E. Palau, Manuel Esteve. URSI 2012

Abstract

La simulación es una importante herramienta en el disseño, con el finde realizar las pruebas necesarias con mayor rapidez, sin riesgos y con un coste mucho menor. Las redes de sensores se utilizan para muchas aplicaciones, como el caso particular de la seguridad en entornos industriales, las redes de sensores se pueden utilizar para prevenir los accidentes y los riesgos a los trabajadores. El proyecto CENIT FASyS (Fabrica Absolutamente Segura y Saludable) tiene como reto fundamental la vigilancia continua de los trabajadores con el fin de detectar de forma proactiva los riesgos. Por todo esto, hemos desarrollado un simulador para simular redes de sensores que representen cualquier situación, con el fin de planificar cualquier escenario potencialmente peligroso. El simulador incluye también una interfaz gráfica sencilla.

Índice

Capítulo I

Introducción

I. Introducción

La presente tesina ha sido realizada por **Pablo Giménez Salazar** en el grupo de investigación de Sistemas y Aplicaciones de Tiempo Real Distribuidos (**SATDR**) del Departamento de Comunicaciones de la UPV, emplazado en la Escuela Técnica Superior de Ingenieros de Telecomunicaciones (ETSIT) de la UPV. Ha estado bajo la supervisión del doctor Carlos Palau Salvador, director de la presente tesina.

I.1 Motivación

Los avances tecnológicos en seguridad en entornos industriales han evolucionado mucho en los últimos años, pero aún existen riesgos relacionados con el bienestar y la salud de los trabajadores. Por tanto se deben desarrollar nuevos conocimientos y tecnologías para garantizar de forma integrada, la seguridad y el bienestar continuo del trabajador en las fábricas de manipulación, mecanizado y montaje, permitiendo que los trabajadores se conviertan en factores clave de competitividad y diferenciación del nuevo modelo productivo.

El logro de este objetivo requiere una plataforma completa para la detección generalizada de riesgos, capacidades avanzadas de razonamiento y una respuesta más autónoma hacia la mitigación del riesgo, así como la obtención de información y el lanzamiento de programas de formación de los trabajadores. Esta plataforma de sensores debe monitorizar en tiempo real y de manera detallada todos los riesgos y mostrarlos de forma jerárquica y sectorizada. Todo lo que sucede en la fábrica, desde los niveles ambientales a la posición de todos los elementos, deben ser controlados y los riesgos potenciales se deben anticipar a través de acciones preventivas automatizadas.

El proyecto FASyS (Fábrica Absolutamente Segura y Saludable) se dirige a este objetivo principal desde un marco multidisciplinar. Es un proyecto con un presupuesto de 23,3 millones de euros que forma parte de los 18 grandes proyectos estratégicos nacionales apoyados por el CDTI dentro de la convocatoria CENIT 2009. Tiene el objetivo de mejorar la competitividad empresarial a través del desarrollo de nuevos niveles en seguridad industrial y confort en el trabajo.

FASyS identifica 13 situaciones de peligro en el entorno de una fábrica de manipulación, mecanizado y montaje, tales como atrapamientos, caídas a nivel, posturas forzadas o movimientos repetitivos, entre otras (Fig. 1), y desarrolla la tecnología necesaria para evitar y responder proactivamente y de forma integral a los riesgos asociados a las mismas [1].

Fig. 1 – Riesgos planteados en FASYS

I.2 Prestaciones de la simulación

Con el fin de comprobar la viabilidad (y escalabilidad) de un sistema como el de FASYS en el que son necesarios muchos recursos, es necesario introducir herramientas de simulación, ya que sería muy costoso desplegar cientos de sensores para reproducir cualquier escenario de riesgo potencial. Hay que tener en cuenta que un sistema simulado permite evaluar todo tipo de situaciones como casos excepcionales o el comportamiento a largo plazo, mientras que una fábrica real sólo puede requerir algunas de las redes de sensores destinados a aquellos riesgos que son relevantes para ellos. El uso de herramientas de simulación de entornos de prueba permite una línea de tiempo de despliegue rápido, barato y sin riesgo [2] [3].

Debido a la importancia de la simulación, ya existen varias herramientas que facilitan el estudio de las redes de sensores, tales como TOSSIM [4], que es un simulador de eventos discretos para redes de sensores basados en el sistema operativo de sensores TinyOS. Otras herramientas para la simulación de eventos discretos son NS-2 [5] (ahora también avanza hacia NS-3) y OMNeT ++ [6], que son simuladores de red generales, de alcance totalmente programable y modular, y se han diseñado para soportar el modelado de las grandes redes construidas a partir de los componentes del modelo reutilizables. Otro simulador relevante es Globosim, que es específico para redes inalámbricas móviles.

Además de las herramientas de simulación, existen normas (protocolos y especificaciones) que tratan de dirigirse a la interoperabilidad entre las redes de sensores. En la actualidad las redes de sensores reales son específicas de cada fabricante, así que son necesarias aplicaciones propietarias y adaptaciones si se necesita interoperar con redes de sensores de diferentes fabricantes. La iniciativa OGC (Open Geospatial Consortium) creó SWE (Sensor Web Enablement) para promover la interoperabilidad y la definición de los distintos servicios y componentes. Esta tesina se centra en el SOS (Sensor Observation Service), que apoya un modelo de datos común de cualquier sensor, así como una consulta de datos común y un modelo de recuperación de cualquier entidad del proceso.

Los simuladores anteriores no se adaptan a nuestras necesidades, porque ninguno de ellos ha sido diseñado para introducir de forma nativa los datos en el SOS. Actualmente no existe ninguna implementación disponible que emplee el servicio SOS en cualquier entorno de simulación de sensores. Además, algunos sensores modelados en el proyecto FASyS pueden requerir un tratamiento especial, por lo que ha sido necesario desarrollar nuestro propio simulador. El simulador cumple con los requisitos FASyS, ya que es capaz de simular todos los sensores que se plantean en los casos de uso y puede simular todos los escenarios posibles a través de propiedades específicas. El SOS se utiliza como sumidero de todas las medidas proporcionadas por sensores, cumpliendo con las normas establecidas por el SWE. De esta manera se puede añadir cualquier tipo de sensor en el sistema y enviar las mediciones fácilmente sobre una red IP o Internet.

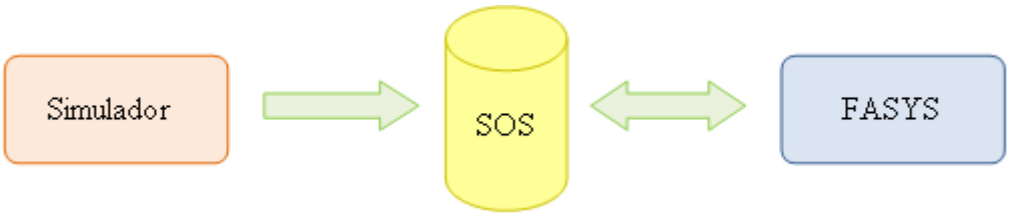

Fig. 2 – Esquema general de funcionamiento

Capítulo II

Objetivos de la Tesina

II. Objetivos de la Tesina

A continuación se describen de manera esquemática las metas que se pretenden alcanzar con el desarrollo de esta tesina, para lograr la mejora en seguridad en entornos industriales antes mencionados.

Se intentan conseguir una herramienta eficaz que permita mejorar y optimizar aplicaciones que ayuden la seguridad laboral. Para ello se establecen un objetivo principal:

II. 1 Objetivos principales

- Diseñar y desarrollar un simulador capaz de representar el funcionamiento real de varios sensores, tanto fijos como móviles, y los datos que miden.
- Usar el SOS como base de datos de observaciones de los sensores, para lo cual es necesario enviar y recibir mensajes siguiendo el estándar O&M.
- Aplicar SWE a un sistema de prevención de riesgos laborales.

II.2 Objetivos secundarios

Para cumplir con los objetivos principales es necesario cumplir una serie de objetivos de carácter secundario:

- Aprender el ciclo de desarrollo un proyecto completo, desde la búsqueda de información, estudio de las necesidades y especificación de requisitos, pasando por el diseño de hasta la programación y validación del mismo.
- Buscar simuladores que pudieran realizar la funcionalidad necesaria.
- Colaborar con otros equipos como parte de un mismo proyecto.
- Realizar una programación modular que permita una fácil modificación y ampliación, tanto durante desarrollo como en el futuro.
- Publicar un artículo como parte del importante de un proyecto de investigación.

Capítulo III

Herramientas

III. Herramientas

En este apartado se centra en definir el entorno de desarrollo elegido para gestionar los sensores, así como en los aspectos técnicos de las tecnologías empleadas de carácter innovador que han sido objeto de estudio en la realización de la presente tesina, y que son necesarios para el cumplimiento de los objetivos del punto anterior.

Las tres herramientas principales utilizadas son los estándares desarrollados por SWE para enviar y recibir información con el SOS, Java y Eclipse para el desarrollo del simulador, y Java y Netbeans para el entorno gráfico llevado a cabo.

III.1 Sensor Web Enablement (SWE)

Dado que el SDI (Spatial Data Infrastructures) no disponía de un framework genérico para integrar datos de sensores, resultó necesaria la ampliación de las especificaciones de SDI para contemplar esta situación. Es por ello por lo que el OGC (Open Geospatial Consortium) fundó la iniciativa Sensor Web Enablement (SWE) encargada de desarrollar estándares para acceder y controlar sensores y redes de sensores a través de Internet.

La arquitectura SWE consta de dos bloques principales: el modelo de información y el modelo de servicio. El primero de ellos consiste en los modelos conceptuales y las codificaciones, mientras que el segundo estriba en la especificación de servicios. La separación en estos dos bloques representa un punto de vista lógico sobre la arquitectura SWE, aunque ello no quiere decir que no existan enlaces entre ambos. [7]

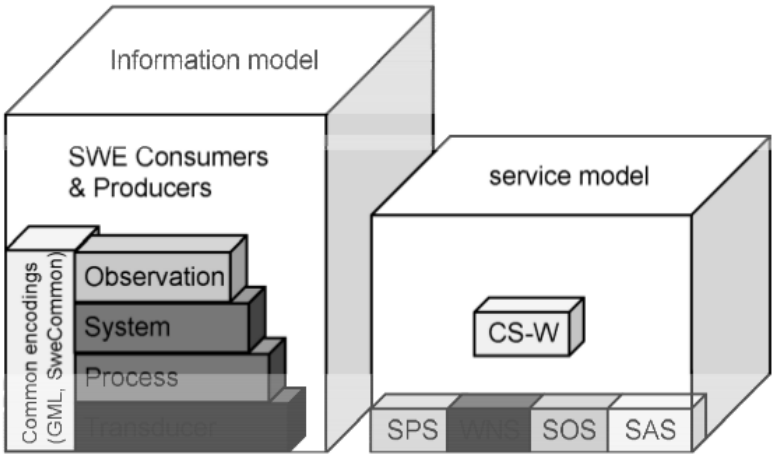

Fig. 3 - Bloques principales de la arquitectura SWE [8]

A continuación se describen las especificaciones que forman la base del modelo de información:

- **Observation & Measurements** (**O&M**): la especificación proporciona un modelo estándar para representar e intercambiar resultados de observación. O&M es principalmente un modelo conceptual que describe la relación entre diferentes aspectos del proceso de captura de datos. Se puede incluir información adicional como metadatos para un mayor análisis e interpretación de los datos. El esquema O&M no sólo permite la definición de observaciones sino también de fenómenos. Basándose en estas definiciones se pueden diseñar diccionarios que definan los fenómenos a emplear un cierto dominio de aplicación. Este tipo de diccionarios forman la base para un conocimiento general de los datos de sensores.

- **Sensor Model Language (SML)**: este lenguaje describe un formato común para la descripción de sensores y sistemas de sensores lo que facilita el descubrimiento de sensores así como el análisis y proceso de datos de sensores. La idea principal en SensorML es que los sensores pueden ser modelados como procesos. Adicionalmente, se pueden introducir metadatos en un proceso. Estos metadatos contienen, por ejemplo, información general empleada en la identificación y clasificación de un proceso (y por ello de un sensor), y también contiene información acerca de las propiedades y capacidades de un proceso. Un proceso describe o bien el procedimiento de medida actual o un método que analiza los datos y genera información adicional.

  En términos generales, SensorML permite:

  - La descripción de la información que es requerida para la exploración de sensores, sistemas de sensores y procesos.
  - El archivo de parámetros del sensor relevantes para una medida.
  - El procesado y análisis de datos de sensores bajo demanda. A través de los procesos SensorML se pueden realizar los pasos necesarios de procesado en tiempo real, basándose en la descripción del sensor y definiciones adicionales del proceso.

- **Transducer Markup Language (TML)**: este lenguaje proporciona la descripción de los datos del sensor, así como la información necesaria para comprender el proceso de obtención de datos. También se emplea en archivar e intercambiar datos de sensores. La especificación define un modelo a partir del cual se puede enviar (en formato streaming), archivar, agregar y analizar de manera eficiente los datos de sensores y de una forma común. TML está especialmente indicado para transmitir streams de datos, por ejemplo streams de vídeo. Los

datos pueden ser enviados (en formato streaming) desde un archivo o directamente desde el sensor. No obstante, en el contexto de SWE, TML se usa principalmente para enviar datos en vivo (live data) desde el sensor al cliente.

El modelo de servicio describe los servicios del framework SWE. Dichos servicios son los siguientes:

- **Sensor Observation Service (SOS)**: El principal modelo de servicio del framework SWE es el SOS, cuyo objetivo principal es proporcionar acceso a las observaciones de los sensores y sistemas de sensores de una forma estándar que sea consistente para todos los sistemas sensores, incluyendo sensores in-situ, remotos, fijos y móviles. SOS cumple con la especificación O&M para modelar observaciones de sensores, así como con la especificación SensorML para modelar sensores y sistemas sensores. [9], [10]

- **Sensor Alert System (SAS)**: Define una interfaz para un servicio web estándar que posibilita la suscripción y publicación de alertas a los sensores. Es posible identificar varios tipos de eventos: una simple medida puede ser considerado como evento, así como la superación de un valor umbral definido por el usuario o un mensaje de estado procedente de un sensor

- **Sensor Planning Service (SPS)**: Define un interfaz para un servicio web estándar que acepta peticiones de usuarios para el acceso a medidas.

- **Web Notification System (WNS)**: Define una interfaz para un servicio web estándar que posibilita la entrega de mensajes o alertas debidas a los servicios web SAS o SPS.

III.2 Entorno de desarrollo

El lenguaje de programación elegido para el desarrollo del simulador ha sido de **Java** debido a su portabilidad y capacidad multiplataforma, lo que permite una fácil integración con otros desarrollos. Además gestiona eficientemente el uso de hilos de ejecución, lo cual es imprescindible, ya que la simulación de cada sensor se ejecuta, en tiempo real, en un hilo diferente con el fin de manera que son independientes entre ellos. Otra razón es que el SOS de 52north también está desarrollado en Java y por tanto la integración es más sencilla.

Se ha decidido usar **Eclipse** para el desarrollo del simulador, que es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar "Aplicaciones de Cliente Enriquecido", basadas en navegadores. Esta plataforma, típicamente ha sido usada para

desarrollar entornos de desarrollo integrados (IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse. [1]

III.3 Entorno de desarrollo gráfico

Para realizar un interfaz simple que permitiera gestionar el simulador y visualizar los datos recibidos, se ha decidido usar **NetBeans**, donde el lenguaje de programación también es **Java**. Es un entorno de desarrollo integrado libre, que permite escribir, compilar, depurar y ejecutar programas. Además es posible desarrollar aplicaciones a partir de un conjunto de componentes de software llamados *módulos*, que son archivos Java que contienen clases de java escritas para interactuar con las APIs de NetBeans. Con esta plataforma se ha podido representar fácilmente todos los sensores a simular en una tabla, así como mostrar todas sus características de simulación. También permite añadir menús y botones para gestionar de manera sencilla la aplicación.

Capítulo IV

Metodología

IV. Metodología

Con las herramientas descritas en el punto anterior se van a comentar en el presente capítulo los aspectos relacionados con la metodología de trabajo llevada a cabo durante la realización de la tesina, para cada una de sus partes.

Se han llevado a cabo tres partes importantes a la hora de desarrollar el simulador. La primera de ellas ha sido conocer el funcionamiento del SOS, ya que ha sido necesario registrar nuevos sensores, insertar las mediciones que iban generando, y actualizar las propiedades de un sensor. La segunda parte ha sido el propio simulador, que a partir de un fichero de configuración genera todos los sensores necesarios y empieza a enviar datos al SOS. Por último para que sea más sencillo de utilizar se ha desarrollado un interfaz gráfico que facilite todas las funcionalidades.

IV.1 Inserción de datos en el SOS

Todos los sensores, así como las observaciones que miden, van a ser almacenados en el SOS del SWE. La principal ventaja es que el interfaz que proporciona es vía web (HTTP) de modo que cualquier sensor puede comunicarse con el SOS independientemente del tipo de sensor.

El SOS es un servicio que ofrece básicamente dos niveles de interfaz:

- Una interfaz de comunicación con los sensores, donde estos sensores se registran y envían periódicamente (cada cierto tiempo, aunque asíncronamente) las medidas que van obteniendo. Desde este punto de vista, el sensor es simplemente un dispositivo sencillo que percibe al SOS como una base de datos donde quedan registradas las medidas de las magnitudes físicas observadas, así como la posición (en el caso de sensores móviles)
- Una interfaz estándar (web) de comunicación con el exterior, donde cualquier aplicación puede acceder a los datos históricos y en tiempo real del sensor sin necesidad de contactar directamente con el sensor. Téngase en cuenta que, al ser el SOS un servicio que centraliza todos los sensores, es posible realizar búsquedas sencillas y aplicar filtros espacio-temporales, como por ejemplo: Obtener todos los sensores que midan temperatura del SOS, u obtener todos los sensores que estén ubicados en una zona determinada.

Por otro lado, el servicio SOS registra los datos de los sensores de una forma semántica en cuanto a que cada valor incluye metadatos básicos:

- Metadatos espaciales: indica donde se produjo la medida (por ejemplo, la medida va asociada a una latitud, longitud y altitud)
- Metadatos temporales: indica cuando se produjo la medida (por ejemplo, en coordenadas UTC)
- Metadatos de medición: indica qué tipo de magnitud física representa (temperatura, presión, etc.) y en qué unidades (Celsius, bares, etc.)

Adicionalmente, dado que el formato de información es XML es posible introducir metadatos adicionales en el caso de resultar necesario.

Para los sensores simulados únicamente es necesario el uso de tres de los múltiples mensajes que ofrece el SOS: *registerSensor*, *insertObservation*, y *updateSensor*.

IV.1.1 RegisterSensor

Para que un sensor pueda enviar observaciones, el primer paso es dar a conocer los sensores al SOS. Para ello existe el mensaje *registerSensor*, que permite registrar un nuevo sensor, así como toda su información.

El mensaje XML siguiendo el estándar establecido por el SWE se encuentra desarrollado en el anexo 2. Los campos más importantes en el mensaje son: un identificador único del sensor, dentro del elemento *sml:identification*;

```
<sml:identification>
      <sml:IdentifierList>
        <sml:identifier>
          <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
            <sml:value>5</sml:value>
          </sml:Term>
        </sml:identifier>
      </sml:IdentifierList>
    </sml:identification>
```

El estado actual del sensor (activo o inactivo) y si es móvil (móvil o fijo), en el elemento *sml:capabilities*;

```
<sml:capabilities>
      <swe:SimpleDataRecord>
        <!-- status indicates, whether sensor is collecting data (true) or not (false) -->
        <swe:field name="status">
          <swe:Boolean>
            <swe:value>true</swe:value>
          </swe:Boolean>
```

```
        </swe:field>
        <!-- status indicates, whether sensor is mobile (true) or fixed (false) -->
        <swe:field name="mobile">
          <swe:Boolean>
            <swe:value>false</swe:value>
          </swe:Boolean>
        </swe:field>
      </swe:SimpleDataRecord>
    </sml:capabilities>
```

La posición representada por la latitud y longitud, en el elemento *sml:position*;

```
<sml:position name="sensorPosition">
      <swe:Position referenceFrame="urn:ogc:def:crs:EPSG::4326">
        <swe:location>
          <swe:Vector gml:id="STATION_LOCATION">
            <swe:coordinate name="easting">
              <swe:Quantity>
                <swe:uom code="degree"/>
                <swe:value> -0.343101 </swe:value>
              </swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="northing">
              <swe:Quantity>
                <swe:uom code="degree"/>
                <swe:value> 39.479161 </swe:value>
              </swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="altitude">
              <swe:Quantity>
                <swe:uom code="m"/>
                <swe:value>1</swe:value>
              </swe:Quantity>
            </swe:coordinate>
          </swe:Vector>
        </swe:location>
      </swe:Position>
    </sml:position>
```

Las medidas físicas que recibe el sensor, en el elemento *sml:inputs*;

```
<sml:inputs>
            <sml:InputList>
                <sml:input name="Temperatura">
                    <swe:ObservableProperty
definition="urn:ogc:def:phenomenon:FASYS:1.0.30:Temperatura"/>
                </sml:input>
            </sml:InputList>
```

```
        </sml:inputs>
```

Y las medidas que ofrece el sensor, en el elemento *sml:outputs*.

```
        <sml:outputs>
            <sml:OutputList>
                <sml:output name="Temperatura">
                    <swe:Quantity definition="urn:ogc:def:phenomenon:FASYS:1.0.30:Temperatura">
                        <gml:metaDataProperty>
                <offering>
                  <id>Termico</id>
                  <name>Termico</name>
                </offering>
               </gml:metaDataProperty>
                        <swe:uom code="ºC"/>
                    </swe:Quantity>
                </sml:output>
            </sml:OutputList>
        </sml:outputs>
```

IV.1.2 InsertObservation

Una vez el sensor ha sido registrado, ya puede empezar a enviar mediciones cada cierto intervalo. El tiempo entre medidas dependerá en cada caso de la magnitud física que se esté midiendo y de la necesidad de control exigida. El mensaje para hacerlo es *insertObservation*, que se detalla a continuación.

El mensaje XML siguiendo el estándar establecido por el SWE se encuentra desarrollado en el anexo 3. Los campos más importantes en el mensaje son: el identificador único del sensor, en el elemento *AssignedSensorId*;

```
<AssignedSensorId>30</AssignedSensorId>
```

El instante temporal en el que se tomó la medida, en el elemento *om:samplingTime*;

```
<om:samplingTime>
        <gml:TimeInstant>
            <gml:timePosition>2011-09-06T17:44:15+02</gml:timePosition>
        </gml:TimeInstant>
    </om:samplingTime>
```

Y la medida junto con las unidades en las que se ha tomado, en el elemento *om:result*.

```
<om:result uom="C">25</om:result>
```

IV.1.3 UpdateSensor

Para no excluir ningún tipo de sensor, también se tiene que prever la posibilidad de que existan sensores que no sean fijos. En estos casos, los sensores además de enviar las mediciones, también deben de enviar periódicamente las posiciones por las que se van desplazando. Para tal cometido está el mensaje *updateSensor*, detallado a continuación.

El mensaje XML siguiendo el estándar establecido por el SWE se encuentra desarrollado en el anexo 4. Los campos más importantes en el mensaje son: el identificador único del sensor, en el elemento *SensorId*;

```
<SensorID>5</SensorID>
```

El instante temporal en el que se tomó la posición, en el elemento *timeStamp*;

```
<timeStamp>
  <gml:timePosition>2011-09-06T17:44:15+02</gml:timePosition>
 </timeStamp>
```

La latitud y longitud de la posición, en el elemento *position*;

```
<position referenceFrame="urn:ogc:def:crs:EPSG::4326">
  <swe:location>
   <swe:Vector>
    <swe:coordinate name="easting">
     <swe:Quantity>
      <swe:value>9.06667</swe:value>
     </swe:Quantity>
    </swe:coordinate>
    <swe:coordinate name="northing">
     <swe:Quantity>
      <swe:value>51.883906</swe:value>
     </swe:Quantity>
    </swe:coordinate>
   </swe:Vector>
  </swe:location>
 </position>
```

Y si el sensor está activo, en el elemento *isActive*;

```
<isActive>true</isActive>
```

IV.2 Programación funcionalidad simulador

Debido a que el diseño software y la programación tienen un peso importante en el proyecto la metodología usada para el trabajo, ha sido la de programación de código y la posterior depuración, en el caso de que su funcionamiento no fuera el correcto.

El simulador se configura a partir de un fichero XML llamado *conf.xml*, en el que existe un listado de todos los sensores a simular, con la descripción y configuración de cada uno de ellos. A continuación se muestra un ejemplo de *conf.xml* para un solo sensor:

```
<?xml version="1.0" encoding="UTF-8"?>
<sensors count="1">
<sensor>
   <!--  Identification of the sensor when registering into the SOS -->
   <id>6</id>

   <!--  Description of the sensor -->
   <description>Example sensor</description>

   <!--  Mode of the sensor: fixed or mobile -->
   <mode>fixed</mode>

   <!--  Offering  -->
   <offering>Termico</offering>

   <!--  Observed property  -->
   <observedProperty>Temperatura</observedProperty>

   <!--  Position where the sensor is located at startup -->
   <longitude>-0.343101</longitude>
   <latitude>39.479161</latitude>

   <!-- URL to invoke the SOS in order to insert observations -->
   <sos_url>http://158.42.188.157:8080/sos320/sos</sos_url>

   <simulation_options>
      <observedProperty>
         <!--  Minimum value of the observed property -->
         <value_min>20</value_min>

         <!--  Maximum value of the observed property -->
         <value_max>30</value_max>

         <!--  Indicates how to generate the values: random, linear or real -->
         <generation mode="real">
            <!--  linear -->
            <count>15</count> <!--  Number of steps -->

            <!--  real -->
            <distribution>sinus</distribution> <!--  sinus,normal,exponential,squared,gamma,random -->
            <distributionOption>0.03</distributionOption> <!-- f,sigma,lambda,k,k, -->
            <errorProb>5</errorProb> <!--  Probabilidad de error (%) -->
```

```
            <anomalyStart>20</anomalyStart> <!--  Anomaly start (s) -->
            <anomalyDuration>10</anomalyDuration> <!--  Anomaly duration (s) -->
            <anomalyValue>44</anomalyValue> <!--  Anomaly value -->
            <anomalyStart>60</anomalyStart> <!--  Anomaly start (s) -->
            <anomalyDuration>10</anomalyDuration> <!--  Anomaly duration (s) -->
            <anomalyValue>8</anomalyValue> <!--  Anomaly value -->
         </generation>

         <!-- Indicates the interval (in milliseconds) when the sensor sends new observations -->
         <refresh_interval>2000</refresh_interval>
      </observedProperty>

      <!--  Longitude and Latitude are targeted independently for greater flexibility -->
      <longitude>
         <!--  Minimum value of the longitude -->
         <value_min>-0.342627</value_min>

         <!--  Maximum value of the longitude -->
         <value_max>-0.342628</value_max>

         <!--  Indicates how to generate the values: random or linear -->
         <generation mode="random">
            <count></count>
         </generation>

         <!-- Indicates the interval (in seconds) when the sensor refreshes position updates -->
         <refresh_interval></refresh_interval>
      </longitude>

      <latitude>
         <!--  Minimum value of the latitude -->
         <value_min>39.479161</value_min>

         <!--  Maximum value of the latitude -->
         <value_max>39.479168</value_max>

         <!--  Indicates how to generate the values between: random or linear -->
         <generation mode="random">
            <count></count>
         </generation>

         <!-- Indicates the interval (in seconds) when the sensor refreshes position updates -->
         <refresh_interval></refresh_interval>
      </latitude>

   </simulation_options>

  </sensor>
</sensors>
```

En este fichero, en primer lugar, se encuentra el identificador con el que el sensor va a ser registrado en el SOS, en el caso que no exista con anterioridad. A continuación se tiene una

descripción del sensor, si es fijo o móvil, el *offering* o agrupación lógica de tipos de medidas, y las propias magnitudes físicas o *inputs*. También hay que definir la posición en la que se encuentra el sensor (latitud y longitud) y la URL de acceso web al SOS en el que se registra el dispositivo.

El resto del fichero se divide en tres partes, en las que se definen las tres posibles opciones de simulación del sensor. Por un lado las medidas que va obteniendo, y por otro el movimiento, en caso de que no sea fijo, separando la latitud y la longitud para tener total libertad.

Para cada uno de ellos se especifica el valor mínimo, el valor máximo, el tiempo cada cuanto se envía un dato, y el modo de simulación. El en caso de la latitud y la longitud el modo únicamente podrá ser lineal o aleatorio, pudiendo definir en número de pasos entre el mínimo y el máximo. Pero en el caso de las observaciones además tiene el modo real, en el que se puede simular errores de medidas de los sensores, anomalías en el ambiente, y seleccionar el modo normal de simulación entre senoidal, gaussiano, exponencial, distribución chi cuadrado, distribución gamma y aleatorio.

Por otra parte, el simulador está estructurado en 4 ficheros (.java). El primero de ellos es *SensorData.java*, en el que define todas las propiedades y métodos de un sensor. El segundo de ellos es *SensorSimulator.java*, el cual inicia el simulador, carga todos los dispositivos del fichero de configuración y por último ejecuta tantos hilos como sensores se hayan planificado. La simulación de cada una de estos hilos se encuentra en el fichero *SensorThread.java*. En él se ha creado un algoritmo que en función de las propiedades de simulación seleccionada (observaciones y/o posición) y la configuración de dichas propiedades (tiempo entre medidas, tipo de medidas) se van generando valores hasta que el usuario pare la simulación.

IV.3 Programación interfaz gráfico simulador

Para el interfaz gráfico se han utilizado dos ficheros principales. El primero de ellos *SensorSimulatorView.java*, para la pantalla principal del simulador en la que se visualizan la tabla con los sensores simulados y los valores que genera cada intervalo de tiempo, así como las opciones de de añadir más sensores o parar cada uno de ellos por separado o todos a la vez. El segundo fichero es *SensorSimulatorNewSensor.java*, en el que hay un formulario para añadir más sensores a la simulación y configurar todos sus parámetros.

Capítulo V

Resultados

V. Resultados

Siguiendo la metodología descrita, en esta tesina se ha conseguido realizar un simulador que permite representar el funcionamiento de una red de sensores que permite disminuirlos riesgos relacionados con la seguridad y la salud de los trabajadores en un ambiente industrial. Es capaz de simular sensores en tiempo real y enviarlos de manera estándar a una base de datos unificada con es el SOS.

V.1. Simulador

V.1.1 Inicio simulador

El simulador implementado es capaz de generar medidas de un gran número de sensores, incluyendo anomalías y errores. En la pantalla principal (Fig. 4) del simulador se puede visualizar un listado de todos los sensores que se están simulando en este momento. En la tabla se visualiza el identificador del sensor, el tipo, la posición inicial (longitud y latitud). Además se tiene la opción de añadir un nuevo sensor o de cargar un nuevo fichero de configuración con nuevos sensores.

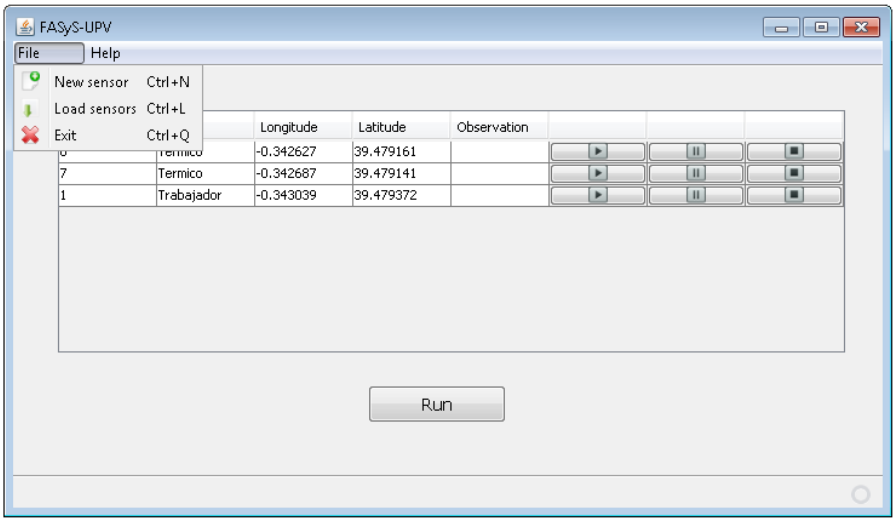

Fig. 4 - Vista principal del simulador

Una vez se empieza a ejecutar la simulación (Fig. 5), la posición del sensor, así como las observaciones a medida que se generan nuevos valores, en función del tiempo y especificaciones establecidas para cada uno de ellos.

Aunque en un principio se ejecuta todos los sensores de la tabla, puede interesar solo ejecutar algunos de ellos o pausar uso de ellos en algún momento, para lo que se han añadido

los botones de la derecha que permiten pausar, continuar y terminar cada uno de los sensores independientemente del resto.

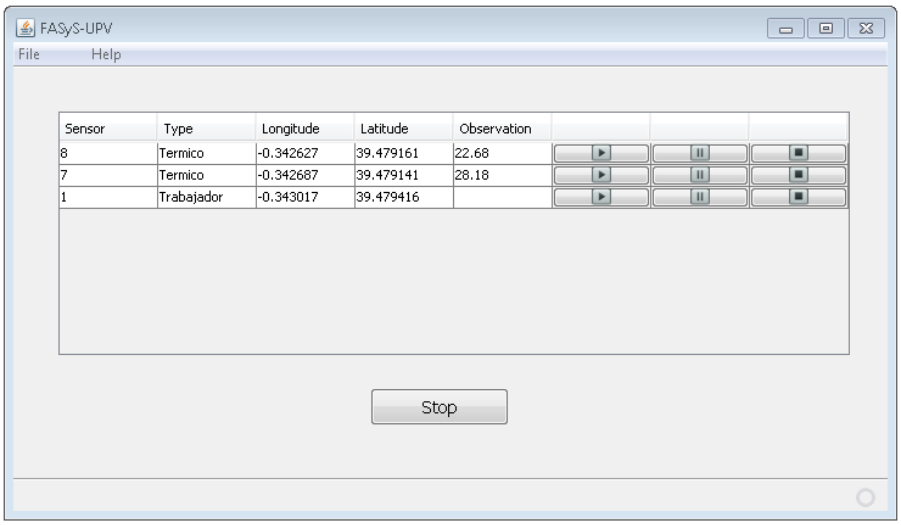

Fig. 5 - Vista principal des simulador (ejecución)

V.1.2 Añadir nuevo sensor

Cuando se necesita añadir un nuevo sensor, en la pestaña File se tiene la opción New sensor, desde el que se accede al formulario de creación de un nuevo sensor (Fig. 6). En él se pueden especificar todas sus características, así como las propiedades de simulación para cada una de las tres opciones: longitud, latitud y observación.

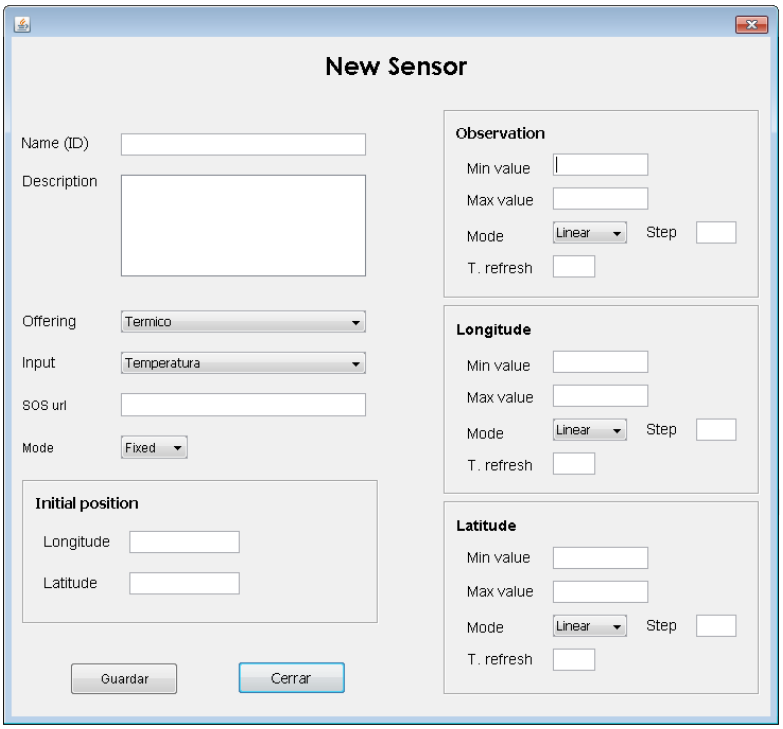

Fig. 6 - Vista del formulario de creación de un nuevo sensor

V.1.3 Datos del simulador

Los datos generados por el simulador de cambian en función del modo de simulación elegido. Como ya se ha comentado es posible introducir anomalías, que pueden representar en un evento temporal que se quiere detectar, como puede ser un incendio para un sensor de temperatura o un elevado ruido para un sensor de intensidad sonora. También se puede especificar una probabilidad de error, que representa el mal funcionamiento de un sensor en un instante de tiempo para que el sistema lo pueda detectar y no se tome ninguna medida.

En la Fig. 7 se pueden observar los datos de un sensor de temperatura con una distribución senoidal con una frecuencia de 0.01, una probabilidad de error de 4% y dos anomalías de 44 y 8 grados.

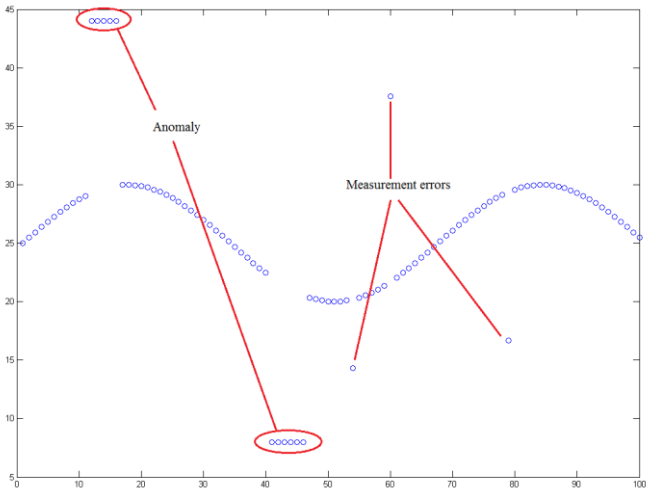

Fig. 7 - Ejemplo datos sensor de temperatura

Otro ejemplo de los datos simulados por el simulador, son las observaciones medidas por un sensor de intensidad sonora con una distribución aleatoria, y la misma probabilidad de error y anomalías.

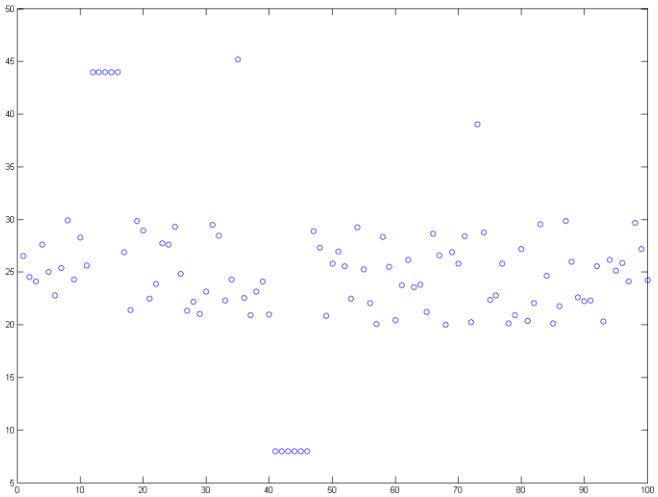

Fig. 8 - Ejemplo datos sensor de intensidad sonora

Capítulo VI

Discusión, conclusiones y líneas futuras

VI. Discusión, conclusiones y líneas futuras

VI.1 Discusión

En este apartado se va a hacer una valoración de los aspectos más importantes de la tesina, resumiendo las partes más importantes que se han encontrado durante la realización del mismo.

VI.1.1 Análisis de resultados

A la vista de los resultados vistos en el apartado anterior se puede concluir que el simulador desarrollado es una herramienta útil para facilitar la planificación y desarrollo de escenarios de seguridad para los trabajadores en entornos industriales, ya que evita la implementación de muchos sensores reales dentro de una fábrica para identificar situaciones de riesgo.

El simulador es capaz de generar los datos en tiempo real de sensores móviles y fijos, y enviar todas las mediciones simuladas a un Servicio de Observación del Sensor (SOS) que actúa como sumidero común. Mediante la interfaz de SOS, no sólo se almacenan mediciones, sino también se actualiza la ubicación del sensor en el caso de sensores móviles. Como la interfaz con los sensores de SOS se ofrece como una interfaz web, cualquier sensor real conectado a una red IP puede interactuar con el SOS, para ello únicamente es necesario registrarse primero y luego el enviar las mediciones y las actualizaciones. Esto es posible gracias a que se han seguido los estándares propuestos por el SWE para permitir la integración de cualquier tipo de sensor.

Los datos generados por el simulador de cambian en función de la variedad de modos de simulación a elegir. Es posible introducir anomalías, que representen un evento temporal o un riesgo que se quiere detectar. También se puede especificar una probabilidad de error, que representa el mal funcionamiento de un sensor en un instante de tiempo para entrenar al sistema a descartar ese tipo de fallos.

Con el fin de facilitar la interacción con el simulador, también se ha desarrollado una sencilla interfaz gráfica de usuario (GUI) donde los sensores se pueden configurar fácilmente. La interfaz gráfica permite al usuario agregar nuevos sensores, cambiar las opciones de simulación, y mostrar los resultados de la simulación.

VI.2 Conclusiones

Como finalización del proyecto se va a realizar una revisión de todo el trabajo realizado, comparando los resultados obtenidos con los objetivos planteados en un principio, comparando el grado de cumplimiento.

VI.2.1 Revisión de objetivos

Objetivos principales

- *Diseñar y desarrollar un simulador capaz de representar el funcionamiento real de varios sensores, tanto fijos como móviles, y los datos que miden.*

Este objetivo ha sido completado de forma totalmente satisfactoria. Se ha realizado un simulador capaz de generar los datos de multitud de sensores de forma realista.

- *Usar el SOS como base de datos de observaciones de los sensores, para lo cual es necesario enviar y recibir mensajes siguiendo el estándar O&M.*

Este objetivo ha sido alcanzado, ya que el simulador genera los datos y las posiciones y las introduce vía web en el SOS, de manera que la información de todos los sensores se encuentra disponible en un mismo sitio.

- *Aplicar SWE a un sistema de prevención de riesgos laborales.*

El uso de los estándares establecidos por el SWE y el SOS como sumidero de las mediciones de los sensores y su posterior análisis permite identificar muchos de los riesgos laborales que se pueden producir en un ambiente industrial

Objetivos secundarios

- *Aprender a desarrollar un proyecto completo, desde la búsqueda de información, estudio de las necesidades y especificación de requisitos, pasando por el diseño de hasta la programación y validación del mismo.*

Este objetivo también ha sido cubierto, ya que se ha partido de las necesidades iniciales de tener multitud de sensores desplegados, y haciendo un estudio de todo lo que ya existía y de los requisitos necesarios ha llevado al desarrollo del simulador.

- *Buscar simuladores que pudieran realizar la funcionalidad necesaria.*

Se ha realizado un estado del arte de los simuladores que existen en el mercado con el fin de encontrar alguno que se adapte a nuestras necesidades. Finalmente se a optado por desarrollar uno propio ya que ninguno cumplía los requerimientos.

- *Colaborar con otros equipos como parte de un mismo proyecto.*

Siendo esta tesina parte una parte del proyecto FASYS, ha sido necesario colaborar con otros equipos y grupos de investigación para establecer las bases y los requisitos del proyecto.

- *Realizar una programación modular que permita una fácil modificación y ampliación, tanto durante desarrollo como en el futuro.*

Durante el desarrollo del simulador se ha estructurado el código de manera que sea fácil de interpretar para la realización de futuros cambios.

- *Publicar un artículo como parte del importante de un proyecto de investigación.*

Dado que la tesina es parte de un proyecto de investigación se ha publicado un artículo sobre el simulador en el congreso URSI 2012.

VI.2.2 Difusión e implantación

El simulador ya está siendo utilizado como una de las herramientas dentro del proyecto FASyS.

El proyecto FASyS tiene como objetivo principal el desarrollo de un nuevo modelo de fábrica, que minimice por diseño los riesgos para la seguridad y la salud y a su vez, garantice el bienestar y el confort del trabajador en las fábricas de mecanizado. No es una fábrica exenta de riesgos, sino una fábrica que dispone de los medios técnicos, organizativos y humanos para identificar, detectar, monitorizar y gestionar de manera continua los riesgos relativos a la salud y seguridad a lo largo de todo el ciclo de vida de la fábrica.

En FASyS se han identificado un elevado número de casos de uso, asociados a 13 riesgos concretos definidos en el entorno de operación. Dada la complejidad de dichos casos de uso, se plantean algunos sencillos, a partir de los cuales se pueden conseguir más elaborados. El primero de ellos es la detección de colisiones, para el cual el simulador debe generar dos dispositivos móviles (un trabajador y maquinaria móvil) y simular un movimiento lógico, el cual permite el modelo de simulación lineal. Otro caso de uso sería la detección de anomalías

que pueden ser generadas por un sensor de temperatura que detecta la presencia potencial de un incendio en un cierto instante [11], [12].

La aplicación FASyS lee todos los datos introducidos en el SOS por el simulador y lo representa gráficamente en una interfaz gráfica. La Fig. 9 muestra una captura de la página principal, con todos los sensores de la fábrica. La GUI también proporciona un medio para acceder a toda la información almacenada en un sensor. Con sólo hacer clic en cada sensor se muestran los detalles del sensor e incluso se puede realizar un seguimiento de datos en tiempo real (Fig. 10).

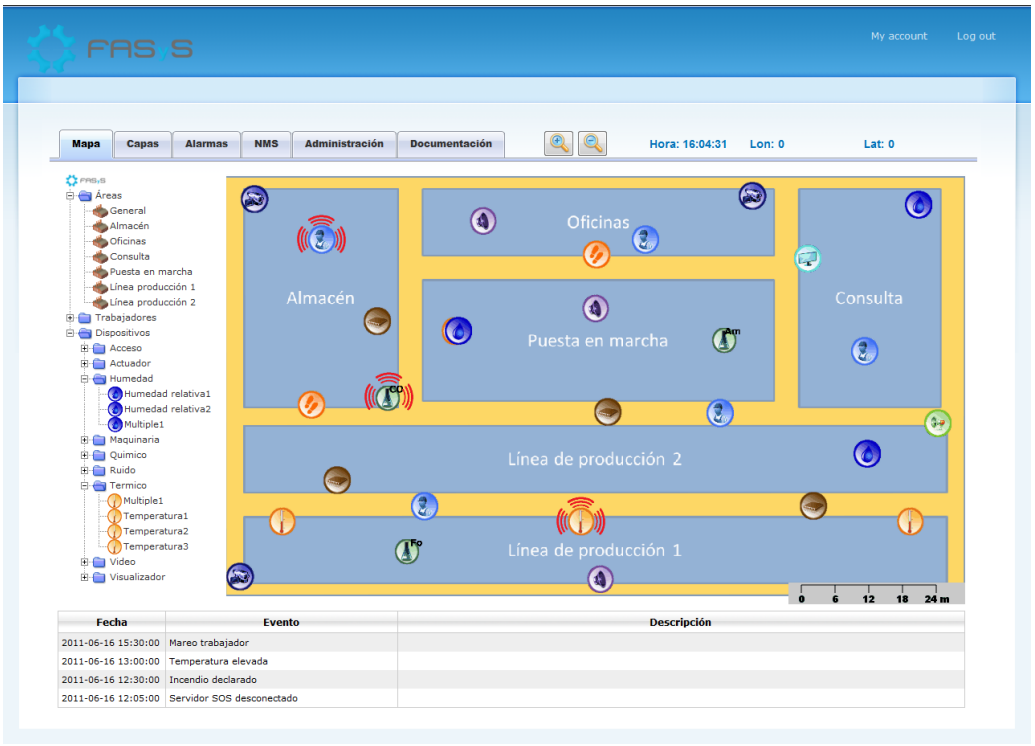

Fig. 9 - Pantalla principal de FASyS

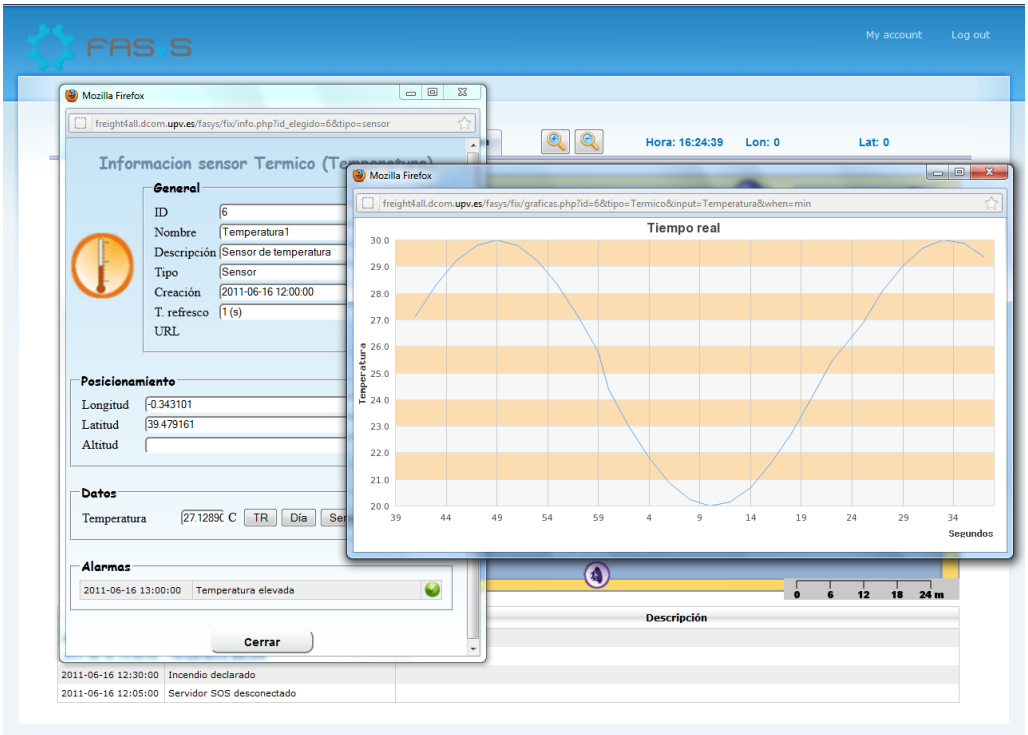

Fig. 10 - Pantalla de descripción de un sensor de FASyS

VI.3 Líneas futuras

Aunque el simulador ya está finalizado, existen diversos aspectos que pueden ser mejorados.

- Añadir nuevos modelos de simulación de sensores que se adapten mejor a la realidad física de los sensores y añadir nuevos modelos de sensores.
- Mejorar el diseño de la interfaz gráfica para que sea más interesante visualmente para los usuarios.
- Integrar el simulador en otros proyectos.

Agradecimientos

Quisiera dar las gracias al proyecto FASyS (CENIT 2009-1034) por apoyar esta investigación.

Referencias

[1] Fasys project website. http://www.fasys.es

[2] Jim Clark, The importance of simulation techniques in its research and analysis, 1997. http://www.informs-sim.org/wsc97papers/1236.PDF

[3] Hans Mielants, The importance of simulation as a mode of analysis, http://www.flw.ugent.be/btng-rbhc/pdf/BTNG-RBHC,%2027,%201997,%203-4,%20pp%20293-322.pdf

[4] Philip Levis and Nelson Lee, TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. http://www.cs.berkeley.edu/~pal/pubs/tossim-sensys03.pdf

[5] Ian Downard, Simulating sensor networks in NS-2. http://cs.itd.nrl.navy.mil/pubs/docs/nrlsensorsim04.pdf

[6] András Varga, The OMNET++ discrete event simulation system. http://www.omnetpp.org/download/docs/papers/esm2001-meth48.pdf

[7] 52north Sensor Web Community. http://52north.org/communities/sensorweb/index.html

[8] Michel Grothe and Jan Kooijman, Sensor Web Enablement. http://www.ncg.knaw.nl/Publicaties/Groen/pdf/45Grothe.pdf

[9] Cory A. Henson, SemSOS: Semantic Sensor Observation Service. http://svn6.assembla.com/svn/soray/user/Marcell/readings/Semantics/SemSOS%20Semantic%20Sensor%20Observation%20Service.pdf

[10] A. Witayangkurn, Real-time monitoring system using unmanned aerial vehicle integrated with sensor observation service.

http://www.isprs.org/proceedings/XXXVIII/1-C22/papers/witayangkurn.pdf

[11] Miguel Sepulcre, Wireless Connectivity for Mobile Sensing Applications in Industrial Environments.
http://www.uwicore.umh.es/files/paper/2011_international/uwicore_SIES2011_Wireless%20Connectivity%20for%20Mobile%20Sensing%20Applications%20in%20Industrial%20Environments.pdf

[12] Carlos Fernández, Semantic Process Choreography for Distributed Sensor Management. http://personales.upv.es/carferll/cv/PDF/SPCHORSSW2010.pdf

Anexo 1 - Artículo

Semantic Sensor Web Simulator for Factory Automation. Pablo Giménez, Benjamín Molina, Carlos E. Palau, Manuel Esteve. URSI 2012

http://www.ursi2012.org/programa-cientifico.html

Sesión VII – Telemática II

Anexo 2 – Registro de sensores en el SOS

El mensaje XML para registrar un sensor en el SOS es el siguiente:

```
<RegisterSensor service="SOS" version="1.0.0"
 xmlns="http://www.opengis.net/sos/1.0"
 xmlns:swe="http://www.opengis.net/swe/1.0.1"
 xmlns:ows="http://www.opengeospatial.net/ows"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:ogc="http://www.opengis.net/ogc"
 xmlns:om="http://www.opengis.net/om/1.0"
 xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.opengis.net/sos/1.0
 http://schemas.opengis.net/sos/1.0.0/sosRegisterSensor.xsd
 http://www.opengis.net/om/1.0
 http://schemas.opengis.net/om/1.0.0/extensions/observationSpecialization_override.xsd">

 <!-- Sensor Description parameter; Currently, this has to be a sml:System -->
 <SensorDescription>

  <sml:SensorML version="1.0.1">
  <sml:member>
   <sml:System xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"   >

   <!--sml:identification element must contain the ID of the sensor-->
   <sml:identification>
    <sml:IdentifierList>
     <sml:identifier>
      <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
       <sml:value>5</sml:value>
      </sml:Term>
     </sml:identifier>
    </sml:IdentifierList>
   </sml:identification>

   <!-- sml:capabilities element has to contain status and mobility information -->
   <sml:capabilities>
    <swe:SimpleDataRecord>
     <!-- status indicates, whether sensor is collecting data (true) or not (false) -->
     <swe:field name="status">
      <swe:Boolean>
       <swe:value>true</swe:value>
      </swe:Boolean>
     </swe:field>
     <!-- status indicates, whether sensor is mobile (true) or fixed (false) -->
     <swe:field name="mobile">
      <swe:Boolean>
       <swe:value>false</swe:value>
      </swe:Boolean>
     </swe:field>
    </swe:SimpleDataRecord>
   </sml:capabilities>

   <!-- last measured position of sensor -->
```

```
    <sml:position name="sensorPosition">
     <swe:Position referenceFrame="urn:ogc:def:crs:EPSG::4326">
      <swe:location>
       <swe:Vector gml:id="STATION_LOCATION">
        <swe:coordinate name="easting">
         <swe:Quantity>
          <swe:uom code="degree"/>
          <swe:value> -0.343101 </swe:value>
         </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="northing">
         <swe:Quantity>
          <swe:uom code="degree"/>
          <swe:value> 39.479161 </swe:value>
         </swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="altitude">
         <swe:Quantity>
          <swe:uom code="m"/>
          <swe:value>1</swe:value>
         </swe:Quantity>
        </swe:coordinate>
       </swe:Vector>
      </swe:location>
     </swe:Position>
    </sml:position>

        <!-- list containing the input phenomena for this sensor system -->
        <sml:inputs>
            <sml:InputList>
                <sml:input name="Temperatura">
                    <swe:ObservableProperty
definition="urn:ogc:def:phenomenon:FASYS:1.0.30:Temperatura"/>
                </sml:input>
            </sml:InputList>
        </sml:inputs>

        <!-- list containing the output phenomena of this sensor system; ATTENTION: these phenomena
are parsed and inserted into the database; they have to contain offering elements to determine the correct
offering for the sensors and measured phenomena -->
        <sml:outputs>
            <sml:OutputList>
                <sml:output name="Temperatura">
                    <swe:Quantity definition="urn:ogc:def:phenomenon:FASYS:1.0.30:Temperatura">
                        <gml:metaDataProperty>
                <offering>
                  <id>Termico</id>
                  <name>Termico</name>
                </offering>
              </gml:metaDataProperty>
                            <swe:uom code="ºC"/>
                        </swe:Quantity>
                </sml:output>
            </sml:OutputList>
        </sml:outputs>

            </sml:System>
    </sml:member>
```

```
      </sml:SensorML>
  </SensorDescription>

  <!-- ObservationTemplate parameter; this has to be an empty measurement at the moment, as the 52N
SOS only supports Measurements to be inserted -->
  <ObservationTemplate>
    <om:Measurement>
      <om:samplingTime/>
      <om:procedure/>
      <om:observedProperty/>
      <om:featureOfInterest></om:featureOfInterest>
      <om:result uom=""></om:result>
    </om:Measurement>
  </ObservationTemplate>

</RegisterSensor>
```

Anexo 3 - Inserción de observaciones en el SOS

El mensaje XML para insertar la medida de un sensor en el SOS es el siguiente:

```
<InsertObservation xmlns="http://www.opengis.net/sos/1.0"
  xmlns:ows="http://www.opengis.net/ows/1.1"
  xmlns:ogc="http://www.opengis.net/ogc"
  xmlns:om="http://www.opengis.net/om/1.0"
  xmlns:sos="http://www.opengis.net/sos/1.0"
  xmlns:sa="http://www.opengis.net/sampling/1.0"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:swe="http://www.opengis.net/swe/1.0.1"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sos/1.0
  http://schemas.opengis.net/sos/1.0.0/sosInsert.xsd
  http://www.opengis.net/sampling/1.0
  http://schemas.opengis.net/sampling/1.0.0/sampling.xsd
  http://www.opengis.net/om/1.0
  http://schemas.opengis.net/om/1.0.0/extensions/observationSpecialization_override.xsd"
  service="SOS" version="1.0.0">

  <AssignedSensorId>30</AssignedSensorId>

  <om:Measurement>

      <om:samplingTime>
          <gml:TimeInstant>
              <gml:timePosition>2011-09-06T17:44:15+02</gml:timePosition>
          </gml:TimeInstant>
      </om:samplingTime>

      <om:procedure xlink:href="30"/>
      <om:observedProperty xlink:href="urn:ogc:def:phenomenon:FASYS:1.0.30:Temperatura"/>

      <om:featureOfInterest>
          <sa:SamplingPoint gml:id="30">
              <gml:name>30</gml:name>
              <sa:sampledFeature xlink:href=""/>
              <sa:position>
                <gml:Point>
                  <gml:pos srsName="urn:ogc:def:crs:EPSG::4326">52.90 7.52</gml:pos>
                </gml:Point>
              </sa:position>
            </sa:SamplingPoint>
      </om:featureOfInterest>

      <om:result uom="C">25</om:result>
  </om:Measurement>

</InsertObservation>
```

Anexo 4 – Actualizaciones de sensores en el SOS

El mensaje XML para actualizar las propiedades de un sensor en el SOS es el siguiente:

```
<UpdateSensor service="SOS" version="1.0.0" mobileEnabled="true"
xmlns="http://www.opengis.net/sos/1.0"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:sos="http://www.opengis.net/sos/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xsi:schemaLocation="http://www.opengis.net/sos/1.0
http://mars.uni-muenster.de/SOSmobile/trunk/sos/1.0.0/sosUpdateSensor.xsd">
  <SensorID>5</SensorID>
  <timeStamp>
    <gml:timePosition>2011-09-06T17:44:15+02</gml:timePosition>
  </timeStamp>
  <position referenceFrame="urn:ogc:def:crs:EPSG::4326">
    <swe:location>
     <swe:Vector>
      <swe:coordinate name="easting">
       <swe:Quantity>
        <swe:value>9.06667</swe:value>
       </swe:Quantity>
      </swe:coordinate>
      <swe:coordinate name="northing">
       <swe:Quantity>
        <swe:value>51.883906</swe:value>
       </swe:Quantity>
      </swe:coordinate>
     </swe:Vector>
    </swe:location>
  </position>
  <domainFeature>
  <GenericDomainFeature gml:id="investigationArea1">
  <gml:description>Paderborn</gml:description>
  <gml:name>City of paderborn</gml:name>
  <gml:location>
    <gml:Polygon srsName="urn:ogc:def:crs:EPSG::4326" xsi:type="gml:PolygonType">
      <gml:exterior>
        <gml:LinearRing xsi:type="gml:LinearRingType">
          <gml:coordinates>51.7167 8.76667, 52.7167 8.76667, 52.7167 9.76667, 51.7167 9.76667,
51.7167 8.76667</gml:coordinates>
        </gml:LinearRing>
      </gml:exterior>
    </gml:Polygon>
  </gml:location>
</GenericDomainFeature>
  </domainFeature>
  <isMobile>false</isMobile>
  <isActive>true</isActive>
</UpdateSensor>
```