Anexos

Apéndice A

Análisis y diseño

En este capítulo se presenta la fase de análisis y diseño que incluye aspectos tales como el análisis de requisitos, metodología empleada, el prototipado de las ventanas, casos de uso y trazas de eventos, diagramas de clases o el estudio de traslación de coordenadas de GPS.

A.1. Análisis de requisitos

El análisis de requisitos se ha realizado tanto para la aplicación de escritorio como para la aplicación que incluirán los NXT, además los requisitos han sido divididos en funcionales y no funcionales. Los requisitos funcionales del dispositivo NXT (ver Tabla A.1) y de la aplicación de escritorio (ver Tabla A.2) permiten dar una visión de todas las funcionalidades que presentan, mientras que los requisitos no funcionales (ver Tabla A.3 y Tabla A.4 para la aplicación del dispositivo y de escritorio, respectivamente) especifican ciertas exigencias a nivel de rendimiento, nivel tecnológico, entorno de desarrollo, de consistencia, etc.

Código	Descripción
RF-1	Soporte de los distintos sensores disponibles, tanto cableados como inalámbricos.
RF-2	Captura de información ambiental de los sensores.
RF-3	Envío de dicha información a la aplicación de forma inalámbrica.
RF-4	Realización de los los movimientos definidos por el usuario.

Tabla A.1: Requisitos funcionales de los dispositivos móviles

Código	Descripción
RF-1	Uso de dispositivos móviles para capturar datos del entorno mediante sensores.
RF-2	Búsqueda de dichos dispositivos de forma dinámica de forma inalámbrica.
RF-3	Configuración remota de los dispositivos detectados.
RF-3a	Configuración de los distintos sensores del dispositivo.
RF-3b	Configuración de los motores del dispositivo.
RF-4	Control de los movimientos de los dispositivos de forma remota.
RF-4a	Control de los movimientos mediante teclas previamente configuradas.
RF-4b	Configuración de trayectorias para ser realizadas por el dispositivo.
RF-5	Representación de la información enviada por los sensores.
RF-5a	Posibilidad de representar los datos en modo texto.
RF-5b	Posibilidad de representar los datos en gráficas.
RF-6	Representación de la posición de los dispositivos.
RF-7	Visualización en tiempo real del vídeo de las cámaras disponibles en el sistema.
RF-7a	Posibilidad de controlar remotamente las cámaras disponibles.
RF-8	Introducción de nuevos sensores en la aplicación.
RF-9	Guardado de la información obtenida en un SGDB externo para un análisis posterior.
RF-10	Guardado / carga de las configuraciones de los dispositivos para posteriores pruebas.
RF-11	Guardado / carga de los movimientos de los dispositivos para posteriores pruebas.

Tabla A.2: Requisitos funcionales de la aplicación

Código	Descripción
RNF-1	El dispositivo elegido será un robot NXT.
RNF-2	El robot NXT tendrá cargada la versión 0.9.1 del firmware leJOS.
RNF-3	El robot NXT tendrá que estar pareado previamente con el ordenador usado.

Tabla A.3: Requisitos no funcionales de los dispositivos móviles

Código	Descripción
RNF-1	La aplicación será una aplicación de escritorio.
RNF-2	La aplicación podrá necesitar de librerías y programas externos.
RNF-3	La aplicación tendrá una interfaz de usuario amigable.

Tabla A.4: Requisitos no funcionales de la aplicación

A.2. Arquitectura del sistema

El sistema desarrollado, bautizado como SkyNXT, se compone de dos partes diferenciadas: la aplicación de escritorio, y la aplicación para los robots NXT. La aplicación de escritorio tiene la misión de controlar los robots, procesar los datos que éstos envían y servir como interfaz para el usuario. Por otra parte, la aplicación para los robots se encarga de recibir las ordenes enviadas por el usuario y transmitírselas a los robots, así como capturar la información sensorial. La Figura A.1 muestra una visión general del sistema con todos los módulos que intervienen en él.

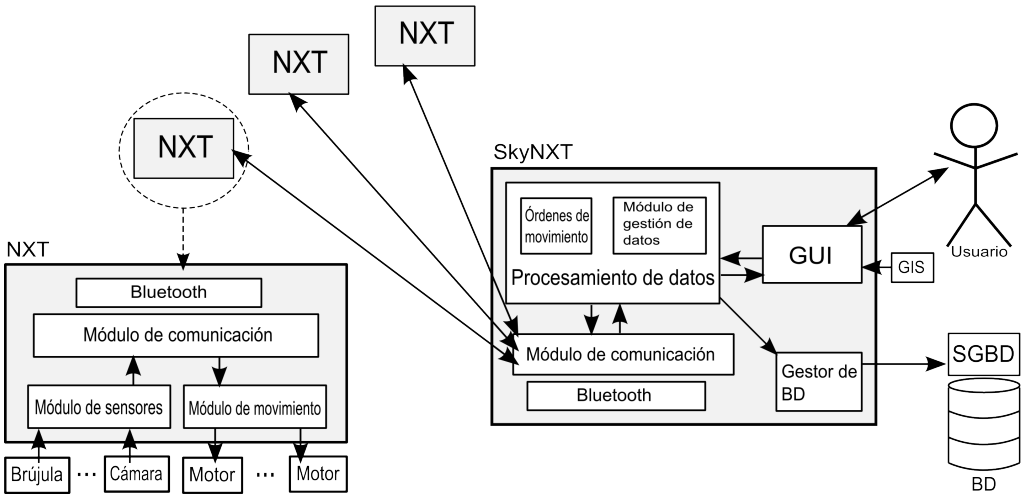


Figura A.1: Arquitectura general de la plataforma

Observamos varios módulos principales en la aplicación cargada en el robot NXT (la Figura A.2 incluye el esquema gráfico de los módulos), cada uno encargado de ciertos elementos diferenciados, como son los distintos sensores y los motores encargados del movimiento, que se comunican con el módulo de comunicación que se encargará de enviar y recibir los mensajes mediante comunicación inalámbrica. Al haber optado por comunicarnos vía Bluetooth, dicho módulo está construido sobre una capa gestionada por la librería Bluetooth disponible por defecto y que se encargará de crear y gestionar las comunicaciones con la aplicación.

El otro elemento del sistema, la aplicación de escritorio, tiene una arquitectura bastante similar a la de la aplicación para los NXT (ver Figura A.3). Observamos

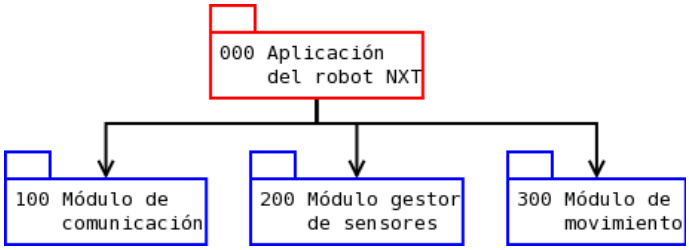


Figura A.2: Arquitectura general de la aplicación para el robot NXT

que tenemos también un módulo de comunicación construido sobre una capa Bluetooth gestionada por la librería instalada en el Sistema Operativo. Dicho módulo se comunica con el módulo denominado "*Procesamiento de datos*", encargado tanto de los movimientos como de recibir los datos de los distintos dispositivos móviles. Al ser una aplicación de escritorio controlada por el usuario, dicho módulo se comunica directamente con la interfaz gráfica (GUI) que mostrará los datos al usuario y recibirá los comandos para controlar remotamente a los dispositivos. Ya que queremos mostrar la posición de los dispositivos en un mapa, la interfaz se comunica con el GIS integrado en la aplicación, Google Earth en nuestro caso. Debido a que queremos almacenar la información para su posterior análisis, el módulo de procesamiento de datos se comunica además con un *Sistema Gestor de Bases de Datos* externo a la aplicación mediante un módulo que se encargará de todo lo relacionado con el almacenamiento de los datos en dicho *SGBD* externo.

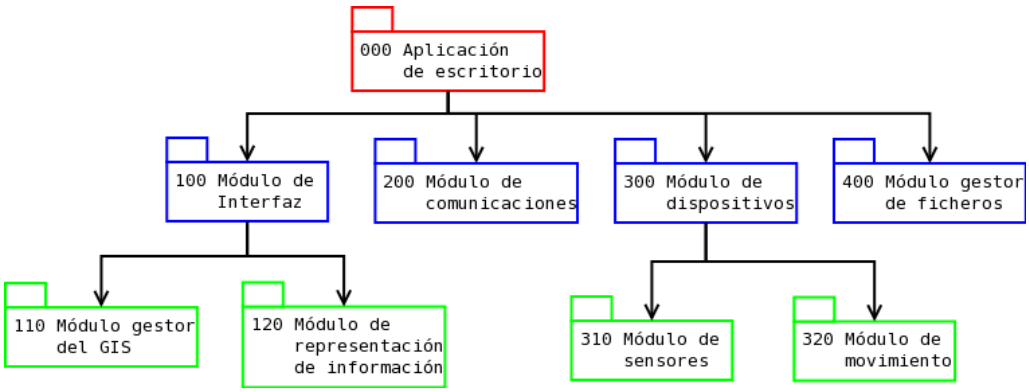


Figura A.3: Arquitectura general de la aplicación de escritorio

A.3. Metodología

Para el desarrollo del proyecto se ha decidido por seguir una metodología iterativa e incremental o creciente, la cual, para cada fase de desarrollo del sistema, se analizan los requisitos y se lleva a cabo el análisis y diseño del mismo para comenzar con la fase de implementación y obtener un producto con el que el usuario pueda interactuar.

Tras una fase de pruebas y *feedback*, se procederá a iterar nuevamente el proceso añadiendo nuevas funcionalidades y capacidades al sistema y realizando cambios en el diseño, atendiendo siempre al *feedback* del usuario y a la lista de control del proyecto, que contendrá un historial de las tareas a ser realizadas.

A.4. Prototipado del GUI

Durante la fase de análisis y diseño se han elaborado varios prototipos para la interfaz gráfica de la aplicación de escritorio. El primero de los prototipos presentado (ver Figura A.4) muestra una zona destinada a la representación de los robots y las cámaras, así como otra zona destinada al control de los movimientos de los robots. Tras revisar el prototipo, se presentó una segunda versión, con algunas mejoras sobre la primera aproximación (ver Figura A.5). Mientras que en el primer prototipo se consideró presentar la información sensorial al usuario utilizando ventanas externas, consideramos que sería más usable que está información estuviera integrada en la ventana principal y que el usuario pudiera personalizar la forma en la que se muestra por medio de botones para el minimizado, maximizado y cierre de cada una de las partes de la interfaz gráfica principal.

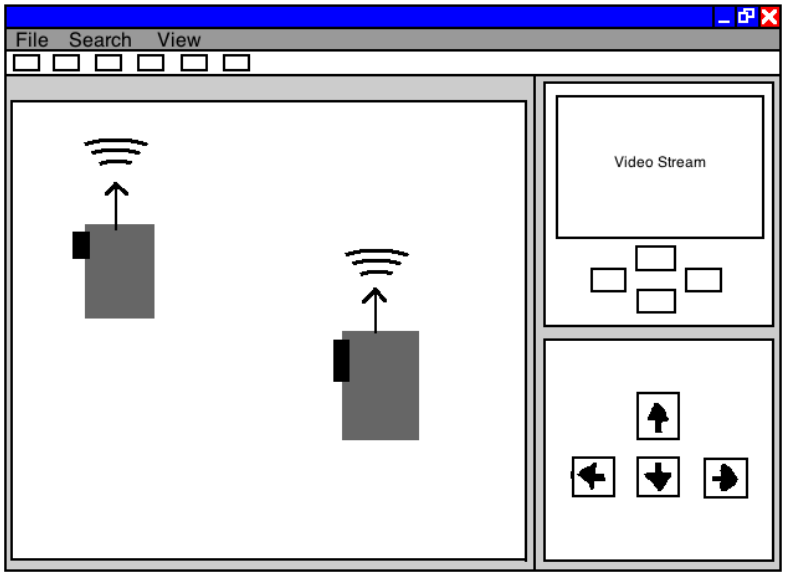


Figura A.4: Primera versión del prototipo de la interfaz gráfica de la aplicación de escritorio

Finalmente se optó por eliminar de la interfaz gráfica la zona destinada al control de los robots, puesto que se sobrecargaba el contenido mostrado, y se decidió que esa parte sería mostrada en una ventana externa. Así pues, el prototipo final para la interfaz gráfica de la aplicación de escritorio (ver Figura A.6) recoge las ideas

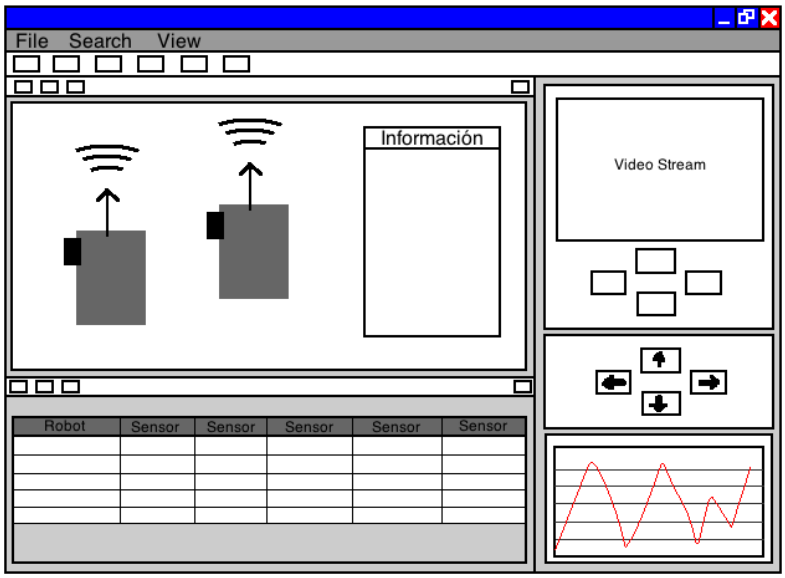


Figura A.5: Segunda versión del prototipo de la interfaz gráfica

comentadas previamente para contribuir a que la información pueda ser obtenida por el usuario de una forma rápida y permitiendo la flexibilidad de la configuración de cada una de las distintas partes. En la Figura A.7 mostramos el resultado final tras implementar las ideas del prototipo final en Java.

A.5. Casos de uso y trazas de eventos

En esta sección vamos a analizar los posibles usos de la herramienta mediante algunos diagramas de casos de uso y trazas de eventos. Los diagramas de casos de uso se crean para representar las operaciones que realiza el usuario dentro del sistema, tal y como podemos ver en la Figura A.8.

Tras la visión general de los casos de uso, veamos ahora ejemplos más concretos de algunos de estos casos de uso o de situaciones concretas dentro de algún caso de uso determinado. La Figura A.9 muestra la traza de eventos perteneciente al caso de uso de configurar la aplicación. En este ejemplo, vemos cómo el usuario ha de configurar un nuevo sensor y posteriormente una nueva cámara en la aplicación. En primer lugar, vemos como la aplicación devuelve la ventana solicitada por el usuario (ventana de configuración) y posteriormente, la ventana de inserción de un nuevo sensor, sobre la que inserta los datos necesarios hasta guardar el sensor. El usuario repite la operación solicitando esta vez la ventana para insertar una nueva cámara. Cuando ya ha insertado ambos, cierra la ventana de configuración, no sin antes salvarse la información insertada en el fichero de configuración.

La segunda traza de eventos que analizamos (ver Figura A.10) muestra la rea-

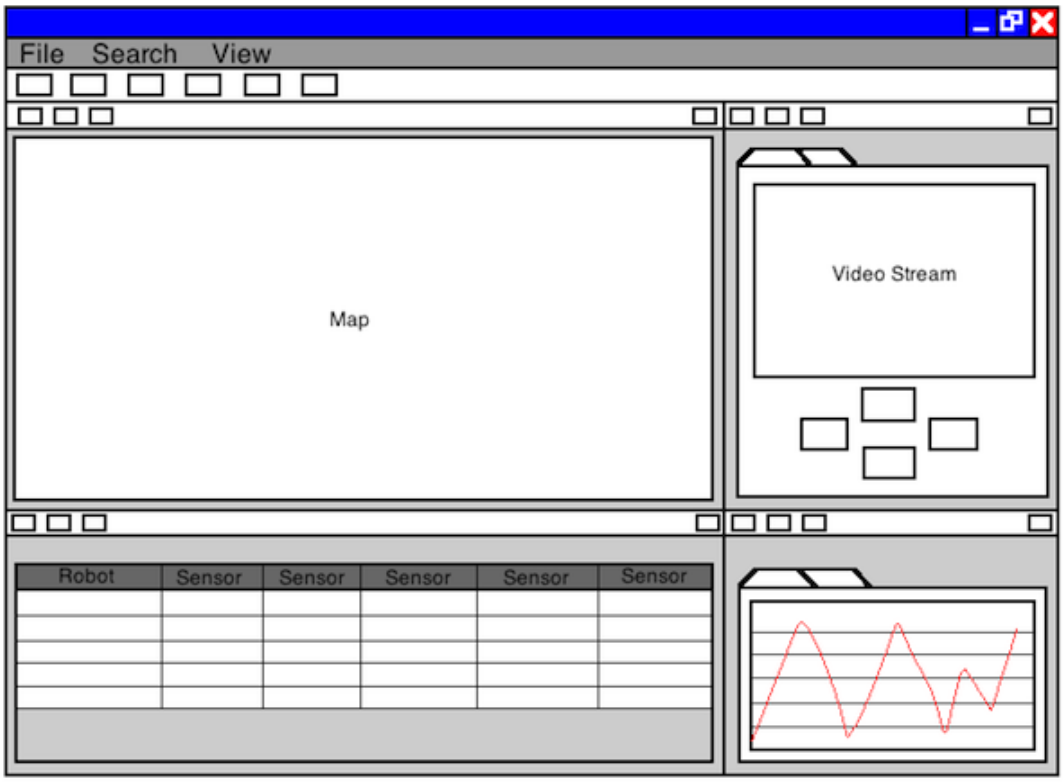


Figura A.6: Prototipo final de la interfaz gráfica de la aplicación de escritorio

lización de todos los pasos necesarios que hay que realizar para poner la ejecución de una simulación. Comenzamos con la búsqueda de robots para continuar con la configuración de los mismos, compuesta por la configuración de los sensores y la configuración de los movimientos, además de cargar dicha configuración al robot. Estos pasos los tendremos que repetir tantas veces como NXT hayamos seleccionado y tengamos disponibles. Tras ello, se manda la orden de comienzo a los NXT, que a su vez comienzan los procesos asociados de lectura de sus sensores y movimiento. Para cada uno de los sensores disponibles en el dispositivo, se leerán los datos adecuados y se enviarán a la aplicación, que los mostrará a través del GUI, en un proceso que repetirá para cada uno de los robots. Una vez finalizada la prueba, se manda el comando de parada a cada uno de los NXT, los cuales finalizarán los procesos de lectura y movimiento en ejecución.

La última traza de eventos que vamos a comentar (ver Figura A.11) se centra en el envío de los datos que se han capturado por los sensores desde cada uno los dispositivos a la aplicación y el tratamiento de éstos para ser mostrados adecuadamente al usuario, según el destino de los mismos (gráficas, tablas, o mapa). Como podemos ver, cuando un dispositivo envía un dato a la aplicación, el módulo de lectura se encarga de recibir dicha información y parsearla para tratarla adecuadamente. En función del tipo de dato recibido, éste será mostrado de varias maneras. Si el dato

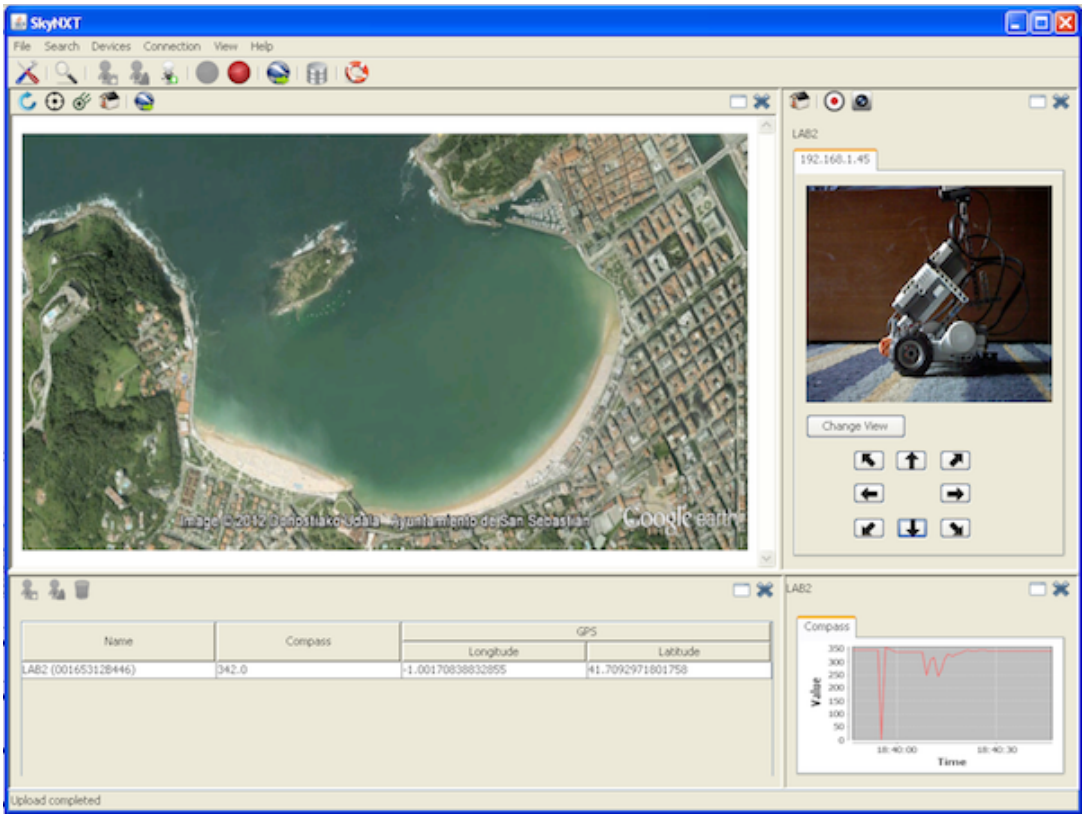


Figura A.7: Interfaz gráfica final de la aplicación de escritorio

pertenece al sensor GPS, entonces enviaremos los datos al módulo de mapas, que se encargará de actualizar la posición en el mapa acorde a los nuevos datos recibidos. Si por el contrario, el dato puede mostrarse en una de las gráficas creadas, el dato se enviará al módulo de gráficas, el cual se encargará de actualizar la gráfica adecuada al sensor del robot con los nuevos datos. En todos los casos, los datos leídos se envían además al módulo de tablas, el cual se encargará de actualizar la tabla insertando en la celda adecuada los datos leídos. También enviamos los datos al módulo de bases de datos para almacenarlos en la base de datos correspondiente.

A.6. Diagramas de clases

En este apartado vamos a comentar tanto el diagrama de clases general como diagramas de algunas de las clases principales de la aplicación de escritorio y de la aplicación para los NXT, detallando los principales atributos y métodos que contiene.

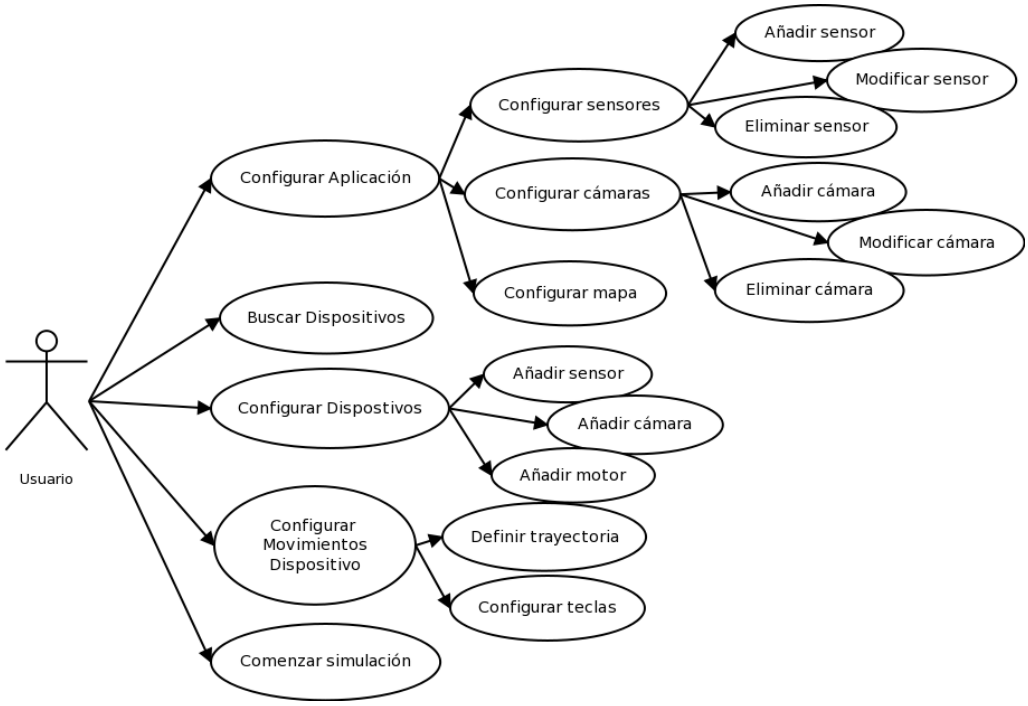


Figura A.8: Diagrama de casos de uso de la aplicación SkyNXT

A.6.1. Diagramas de clases para la aplicación de escritorio

Comenzamos por el diagrama de clases de la aplicación que hemos visto en la memoria, esta vez con los principales métodos y atributos de cada clase (ver Figura A.12).

Como vemos, el módulo *GUI* es la clase principal de la aplicación, que contiene los distintos paneles de la interfaz gráfica y que se relaciona con la mayoría de las otras clases. Más adelante comentaremos las distintas clases que componen a la interfaz gráfica, pero ahora nos vamos a centrar en el resto de las clases de la aplicación. Una de las clases más importantes es la clase *Dispositivo*, ya que hace referencia a cada uno de los dispositivos móviles que va a ser usado en la aplicación y que está compuesto de sensores y de cámaras. Cada sensor tiene una serie de atributos que nos permite conocer cada una de las propiedades del mismo junto con los métodos que nos permiten manipular los atributos de la clase. Además tenemos disponibles las clases *Sensor* y *Cámara*, que contienen los atributos y métodos necesarios para crear y editar cada una de las propiedades de dichas clases. Estas clases también se relacionan con las clases *GestionSensores* y *GestionCamaras*, que permiten añadir nuevos sensores y cámaras respectivamente en la aplicación.

La interfaz está relacionada también con las clases de *ConfigurarDisp* y *ConfigurarMovs*. La primera de ellas permite configurar un dispositivo con los sensores y cámaras que hemos descrito mediante una ventana gráfica, mientras que la segunda permite configurar los movimientos que realizará el dispositivo. Como uno de los

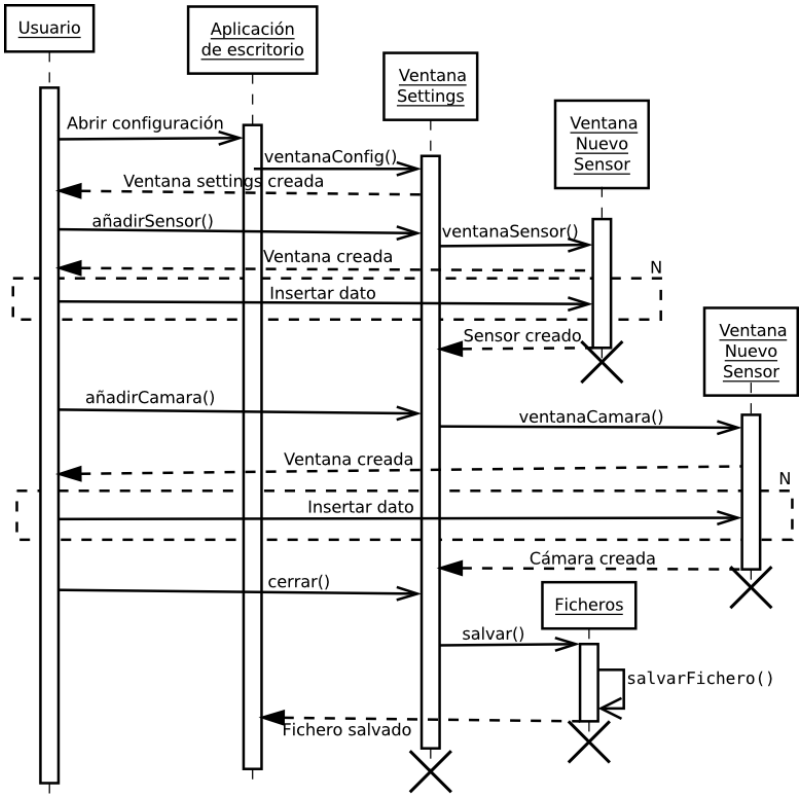


Figura A.9: Traza de eventos para la inserción de un nuevo sensor y una nueva cámara

movimientos disponibles es el de seguir trayectorias definidas por el usuario, tenemos además la clase *Trayectoria*, con los atributos y métodos que permiten definirla y manipularla. Como cada trayectoria se compone de una serie de puntos, tenemos la clase *Punto* con los atributos que lo definen y los métodos para acceder a ellos.

El resto de las clases añaden funcionalidades a la aplicación. La clase *Ficheros* se encarga de la creación y acceso a cada uno de los ficheros de configuración empleados, así como a los ficheros de trayectorias que se generen para ser analizados a posteriori. La clase *Búsqueda* se encarga de la búsqueda de los dispositivos disponibles en el entorno mediante una ventana gráfica, mientras que la clase *Comunicación* se encarga de enviar los comandos a los dispositivos y de leer los datos que ellos envíen a la aplicación, todo ello de forma inalámbrica.

A continuación vamos a analizar en detalle las clases que componen en módulo de la interfaz gráfica, necesarias para el manejo de la aplicación de escritorio. En primer lugar tenemos el diagrama de clases del panel que muestra la información de cada sensor en forma de tabla, que mostrará los datos de cada sensor en las celdas correspondientes de la tabla (ver Figura A.13).

La clase principal, *PanelTablas*, tiene como atributo principal la tabla sobre la que se mostrarán los datos que se reciban, mientras que los atributos de la clase

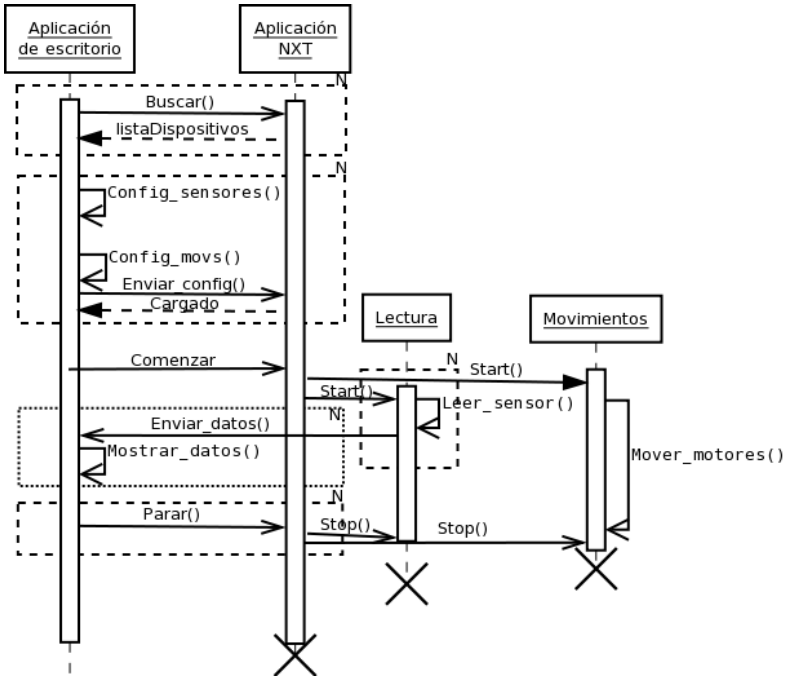


Figura A.10: Traza de eventos de una simulación

permiten tanto la inserción de datos en la tabla como la propia creación y manipulación de la misma, además de permitir configurar los dispositivos con los sensores implicados y los movimientos que realizarán. Estas clases ya han sido comentadas previamente, por lo que no las comentaremos de nuevo, así como la creación de los ficheros de configuración, que emplean la clase ya comentada. Podemos ver que la clase principal se relaciona también con la clase ajena, *JXTableExtendida* que extiende de *JXTable* (esta clase pertenece a la librería externa *JXTable*).

El siguiente diagrama de clases que vamos a comentar pertenece a la ventana del mapa (Figura A.14), sobre la que posicionaremos cada uno de los dispositivos móviles cuando actualicen su posición La clase principal es el *PanelMapa*, que contiene el atributo *browser* que contendrá el mapa donde posicionaremos los dispositivos, y los métodos que permitirán manipular el mapa, permitiendo centrar el mapa sobre los dispositivos, seguirlos automáticamente o activar funcionalidades del mapa. La clase *Browser* extiende de *JWebBrowser* (esta clase pertenece a la librería externa *NativeSwing*) y se compone de un atributo *webBrowser* y un método que permite ejecutar funciones pertenecientes a la página web que ejecutemos en el navegador. A su vez, la clase *PanelMapa* se relaciona también con la clase *Ficheros* anteriormente comentada.

La ventana de cámaras se encarga de mostrar el vídeo que emitan cada cámara situada en los robots, la Figura A.12 contiene el diagrama de clases de esta ventana. Observamos que la clase principal, *PanelCamaras*, contiene dos atributos, el panel que contendrá cada uno de los visores de vídeo (*panelVisores*) y el propio vídeo (*vi-*

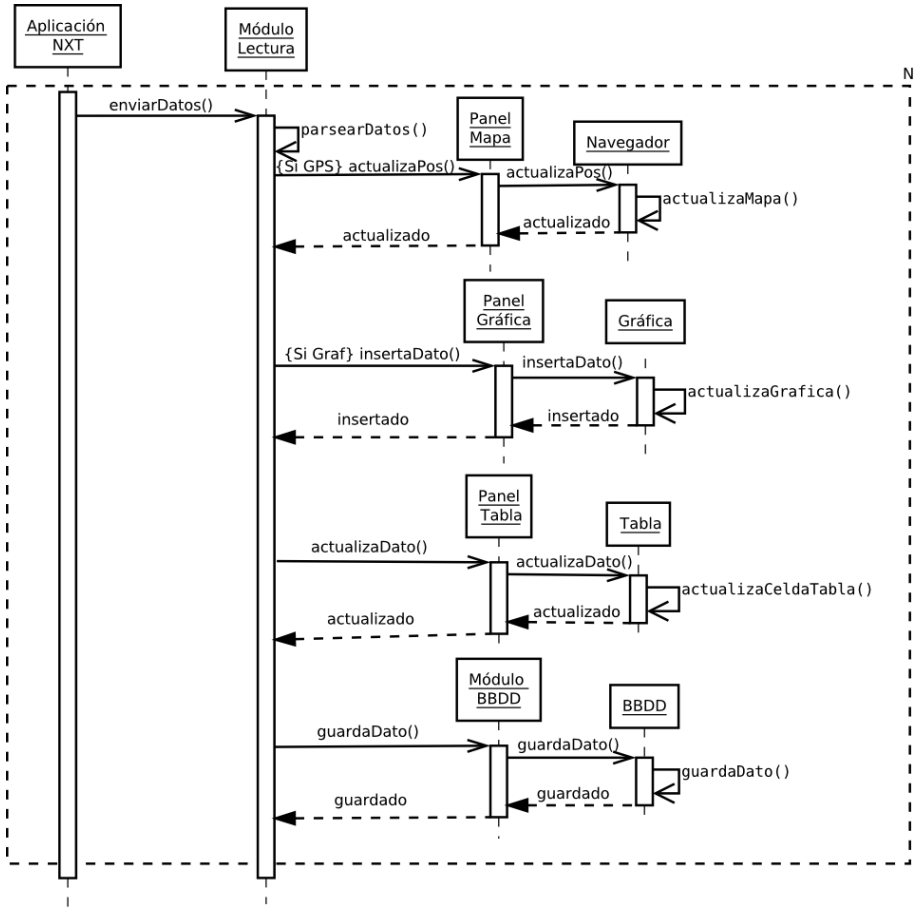


Figura A.11: Traza de eventos del tratamiento de los datos

sor). Los métodos disponibles permiten cambiar el formato de vídeo que se mostrará entre el vídeo real de las cámaras y el vídeo simulado mediante el uso del plugin de Google Earth. Para mostrar el vídeo de las cámaras, empleamos la clase *Visor* que se encargará de conectar a una cámara con los datos de los atributos para obtener una imagen que se refrescará para cada nueva recepción de la misma. Para simular el vídeo volvemos a utilizar la clase *Browser* usada también en la ventana de mapas.

La última de las ventanas de la que vamos a mostrar el diagrama de clases es la ventana de gráficas (ver Figura A.16), encargada de mostrar las gráficas que contendrán el histórico de los datos leídos de cada uno de los sensores de los robots. Al igual que en el diagrama de clases anterior, la clase principal, *PanelGraficas*, contiene dos atributos: el panel que contendrá cada uno de las pestañas con las gráficas (*panelGraficas*) y las propias pestañas (*tabs*). La clase *GraficasTabs* contiene las gráficas pertenecientes a cada uno de los sensores configurados en el NXT, que se crean gracias a la clase *Grafica* (que extiende de la librería externa *JFreeChart*). Para interactuar con ellas tenemos disponibles los métodos de creación y manipulación de las gráficas.

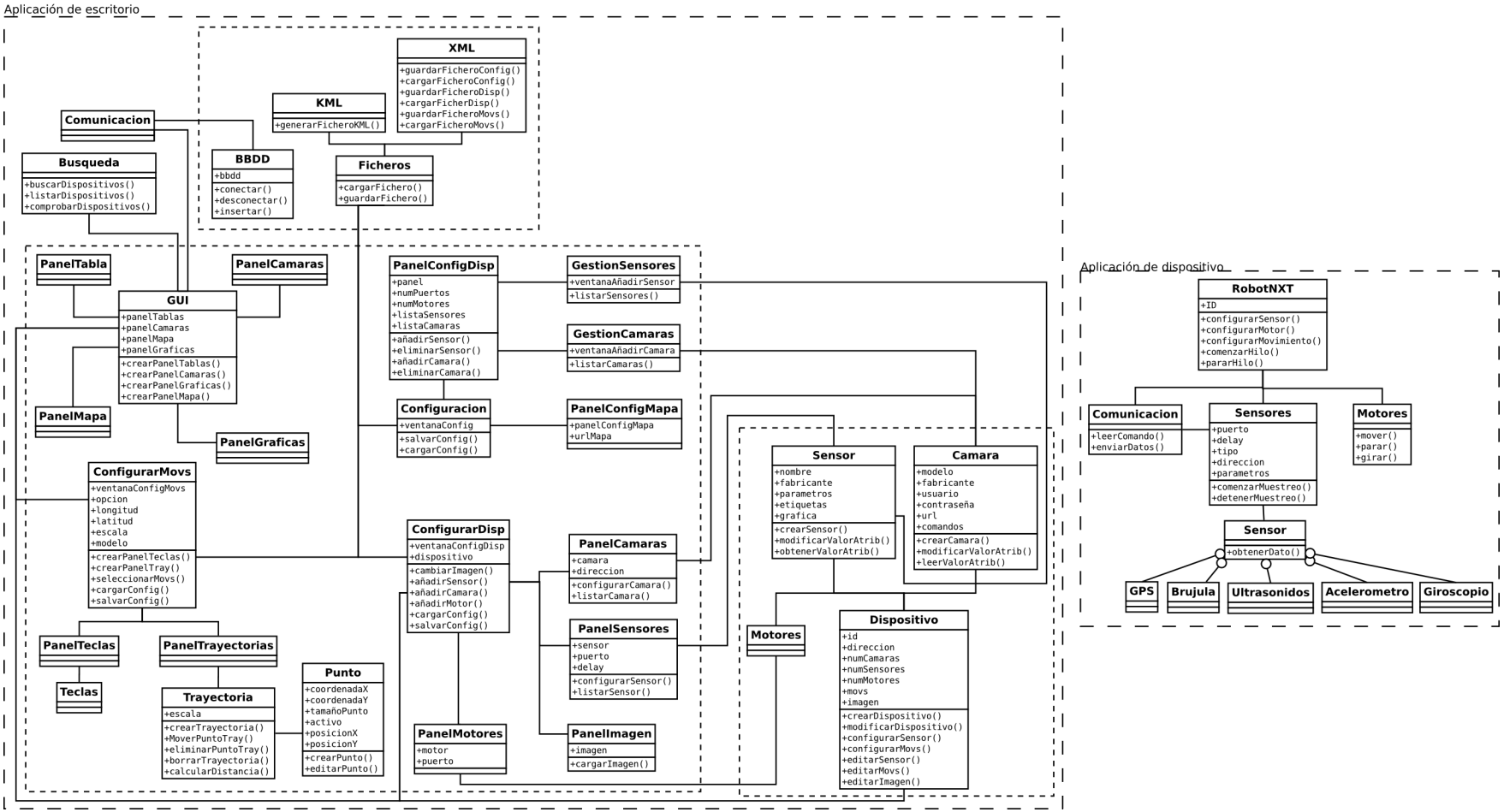


Figura A.12: Diagrama de clases general del sistema

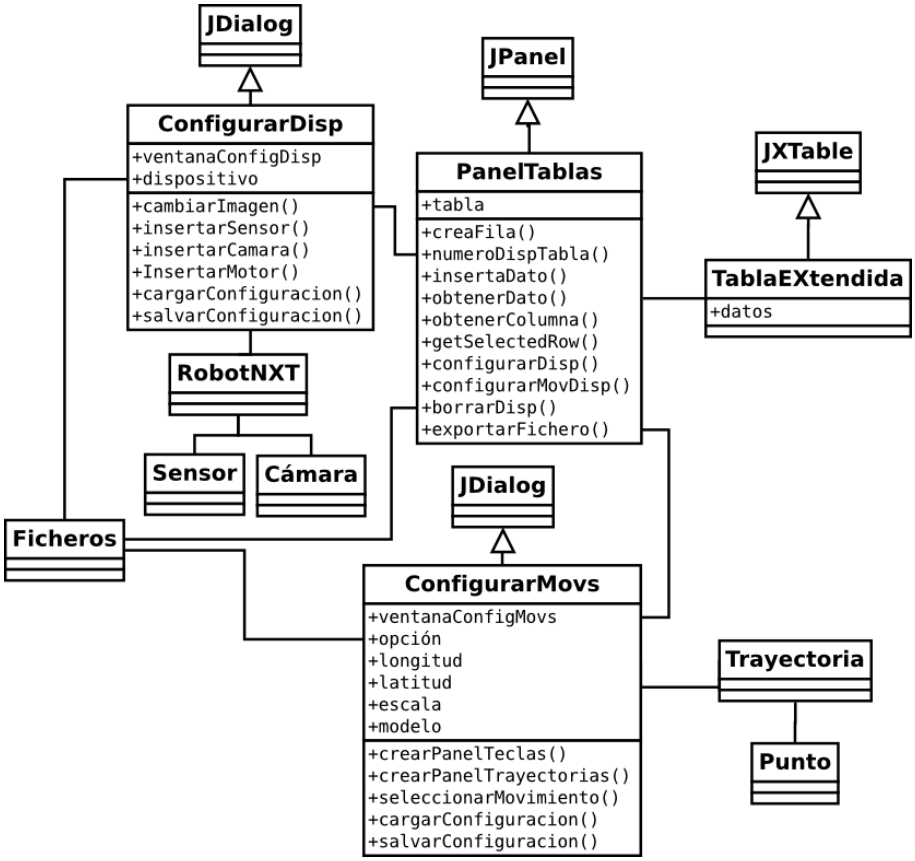


Figura A.13: Diagrama de clases del panel de la tabla de sensores

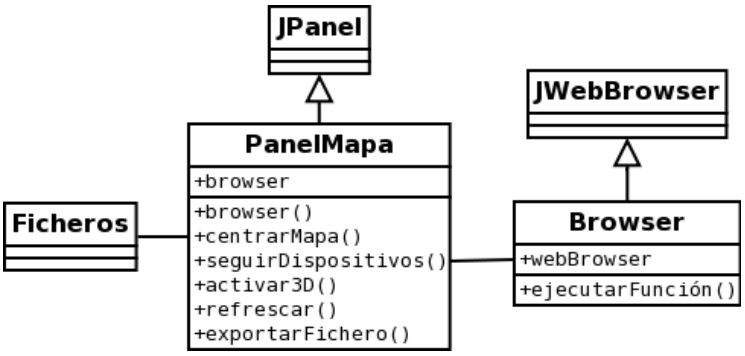


Figura A.14: Diagrama de clases de la ventana de mapa

A.6.2. Diagrama de clases de la aplicación para los robots

En este apartado vamos a comentar el diagrama de clases de la aplicación desarrollada para los robots (ver Figura A.17). La clase principal *RobotNXT*, está compuesta por el atributo *ID* del NXT y por varios métodos encargados de configurar los distintos sensores, motores y opciones de movimiento. Debido a que el robot

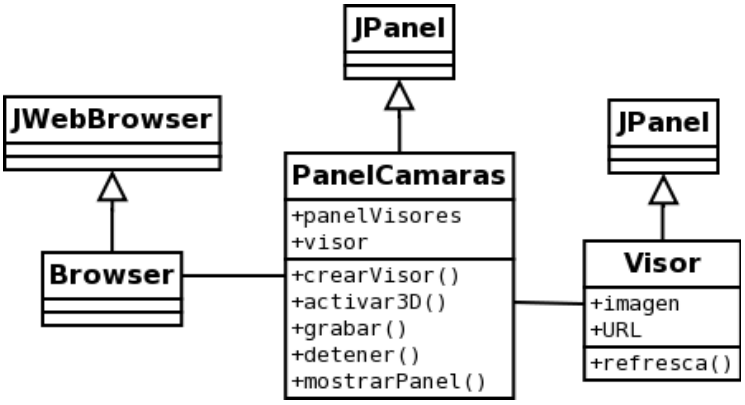


Figura A.15: Diagrama de clases de la ventana de cámaras

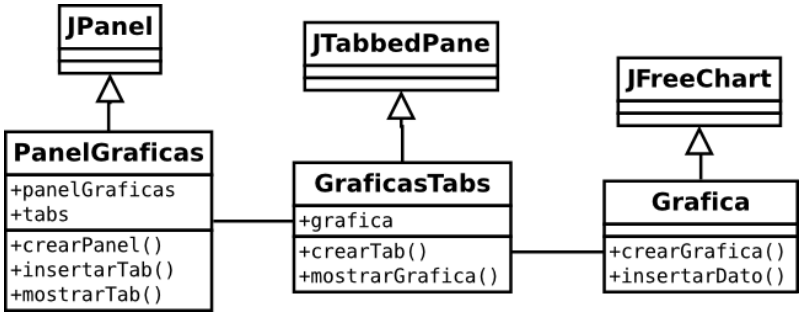


Figura A.16: Diagrama de clases de la ventana de gráficas

va a tener que realizar varias tareas a la vez (leer la información de los sensores, controlar los motores, etc.), se crea un hilo de ejecución por cada una de ellas y los métodos *parar()* y *comenzar()* son los encargados de su puesta en funcionamiento y pausa, respectivamente. La clase *Sensores* contiene los atributos que controlan cada sensor y cuya configuración ha sido enviada desde la aplicación de escritorio (el puerto al que hemos conectado el sensor o su dirección, el tipo del sensor, la frecuencia de muestreo, sus parámetros, etc.). Los métodos *comenzarMuestreo()* y *DetenerMuestreo()* permiten iniciar o parar la lectura de los datos, respectivamente. La clase *Sensor* actúa como interfaz para los distintos tipos de sensores disponibles definiendo las operaciones y atributos comunes a todos ellos. De este modo, las clases *GPS*, *Brujula*, *Ultrasonidos*, etc. implementan a la clase *Sensor* a la vez que extienden a las clases encargadas de la gestión de dichos sensores disponibles en el firmware leJOS. Finalmente, la clase *Motores*, está encargada de los movimientos que ejecutará el robot y la clase *Comunicación*, se encargará de leer los comandos enviados desde la aplicación de escritorio y de los datos leídos por cada uno de los sensores.

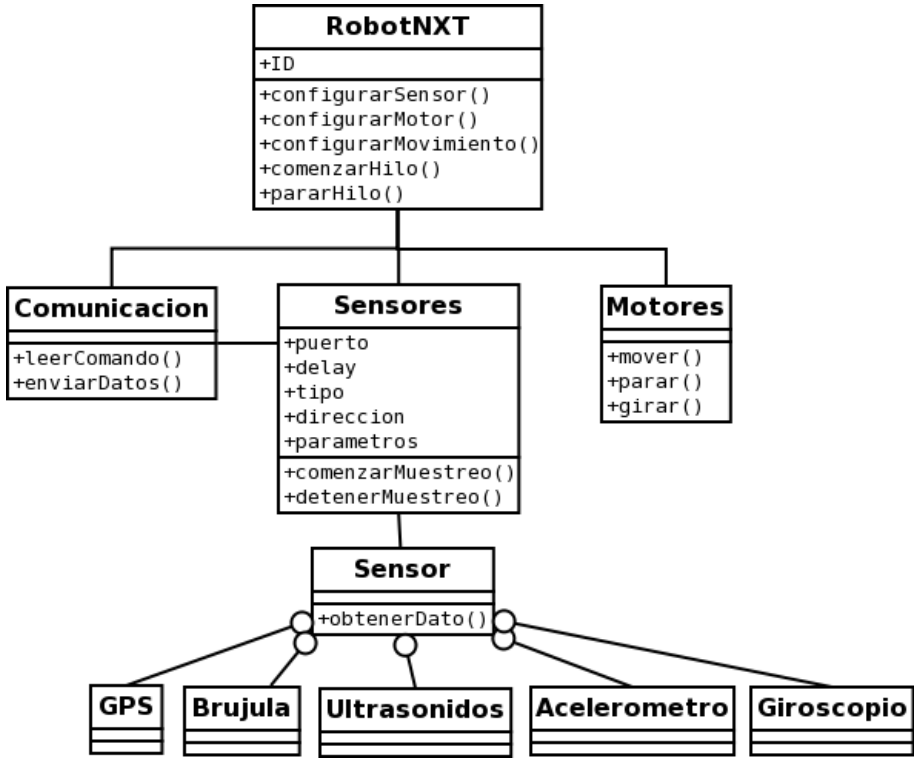


Figura A.17: Diagrama de clases de la aplicación de los NXT

A.7. Diseño de la base de datos

Es un requisito del sistema que se pueda almacenar la información obtenida durante una simulación en una base de datos para su posterior análisis. Por lo tanto, se diseñó un esquema *Entidad/Relación* en el que aparecen todos los elementos involucrados en una simulación (ver Figura A.18).

El diseño de la base de datos contiene tres entidades principales que son las encargadas de almacenar tanto los robots como los sensores que usaremos en una simulación. De la simulación (entidad *Simulación*) queremos almacenar la fecha y hora en la que comenzó y termino (*fechaInicio* y *fechaFin*, respectivamente), el nombre que le asignemos (*nombre*) y un breve comentario donde explicamos sus objetivos (*comentario*). De los NXT (entidad *Robot*) queremos almacenar además de su nombre (*nombre*) la dirección de su dispositivo Bluetooth (*dirección*). De los sensores (entidad *Sensor*) queremos almacenar su nombre (*nombre*), el tipo de datos que captura (*tipoDato*) puesto que consideramos muchos sensores heterogéneo, y finalmente la dirección Bluetooth en caso de que la tuviera. Además, para cada sensor vamos a almacenar en la entidad débil *Muestra* la información de todos los datos que ha capturado durante una simulación (*dato*) guardando también la fecha en la que la capturó (*fecha*). Este diseño E/R lo plasmamos también en el esquema de relaciones de la Figura A.19 en el que podemos ver cada una de las tablas principales con los

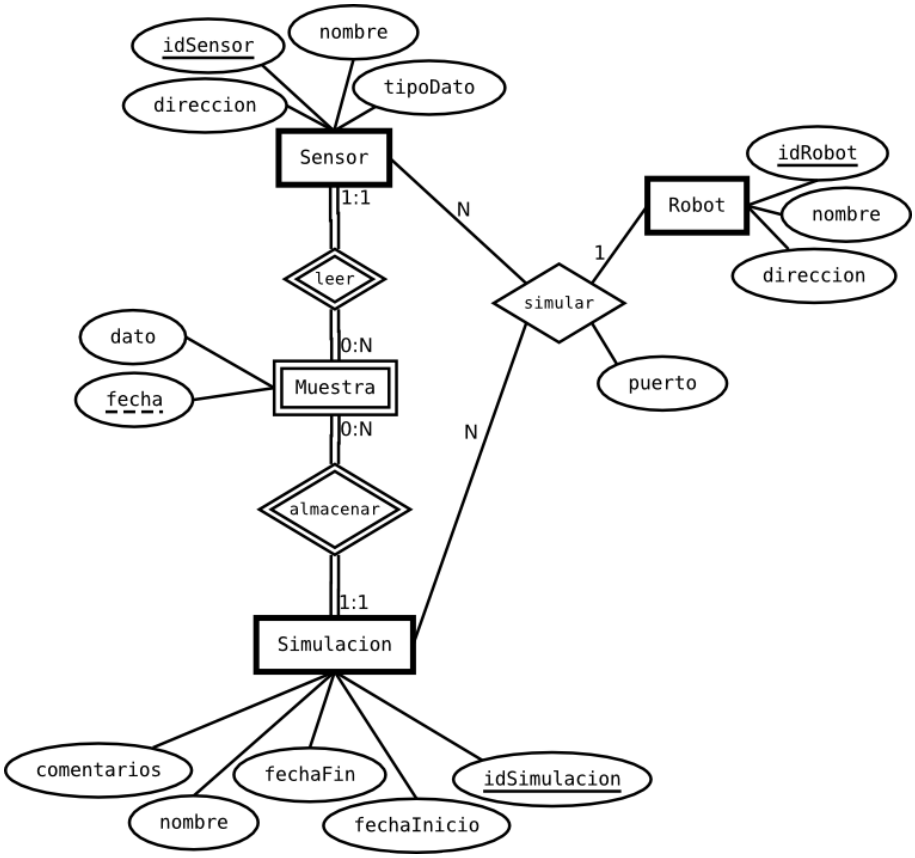


Figura A.18: Esquema E/R de la base de datos

atributos primarios y como están relacionados entre sí. Por último, el Algoritmo A.1 muestra el código SQL para la creación de la base de datos con el diseño de tablas realizado.

A.8. Estudio de traslación de coordenadas

Como hemos podido ver en el análisis de requisitos, una de las características que tenía que cumplir nuestra aplicación era que a posición de los robots tenía que ser mostrada en un *Sistema de Información Geográfica*. Uno de los sensores que pueden equiparse a los robots es el GPS, el cual permite obtener la posición del robot mediante su latitud y longitud. Esta información se envía a la aplicación de escritorio que, por medio de los módulos correspondientes, muestra al usuario la posición de cada robot en un mapa. Esta característica podría ser suficiente para una aplicación que simplemente mostrará la posición de los robots, pero SkyNXT, al estar destinada a realizar simulaciones en las que podrán definirse distintos tipos de escenarios, ha de permitir situar el movimiento que realizan los robots en cualquier punto del mapa. Por ello, en algunas ocasiones será necesario que traslademos la posición actual

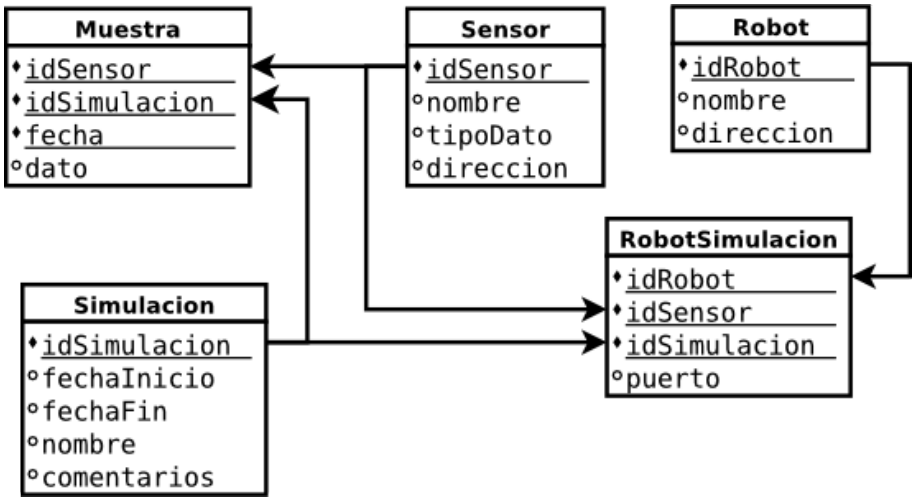


Figura A.19: Esquema de relaciones de la base de datos

Algoritmo A.1 Código SQL para la creación de la base de datos

```
CREATE TABLE Robot (
   idRobot number(6) PRIMARY KEY,
   nombre char(30) NOT NULL,
   direccion char(12) NOT NULL);

CREATE TABLE Sensor (
   idSensor number(6) PRIMARY KEY,
   nombre char(30) NOT NULL,
   tipoDato char(10) NOT NULL,
   direccion char(12) );

CREATE TABLE Simulacion (
   idSimulacion number(6) PRIMARY KEY,
   fechaInicio date NOT NULL,
   fechaFin date NOT NULL,
   nombre char(40),
   comentarios char(100) );

CREATE TABLE Muestra (
   idSensor  number(6) NOT NULL,
   idSimulacion number(6) NOT NULL,
   fecha date UNIQUE,
   dato char(30) NOT NULL,
  PRIMARY KEY (idSensor, idSimulacion, fecha),
  FOREIGN KEY (idSensor) REFERENCES Sensor(idSensor),
  FOREIGN KEY (idSimulacion) REFERENCES Simulacion(idSimulacion) );

CREATE TABLE RobotSimulacion (
   idSensor  number(6) NOT NULL,
   idSimulacion number(6) NOT NULL,
   idRobot number(6) NOT NULL,
   puerto number(6) NOT NULL,
  PRIMARY KEY (idSensor, idSimulacion, idRobot),
  FOREIGN KEY (idRobot) REFERENCES Robot(idRobot),
  FOREIGN KEY (idSensor) REFERENCES Sensor(idSensor),
  FOREIGN KEY (idSimulacion) REFERENCES Simulacion(idSimulacion) );
```

del robot obtenida por el GPS a la posición del escenario que estemos simulando, teniendo en cuenta que puede ser necesario modificar la escala de los movimientos de los robots.

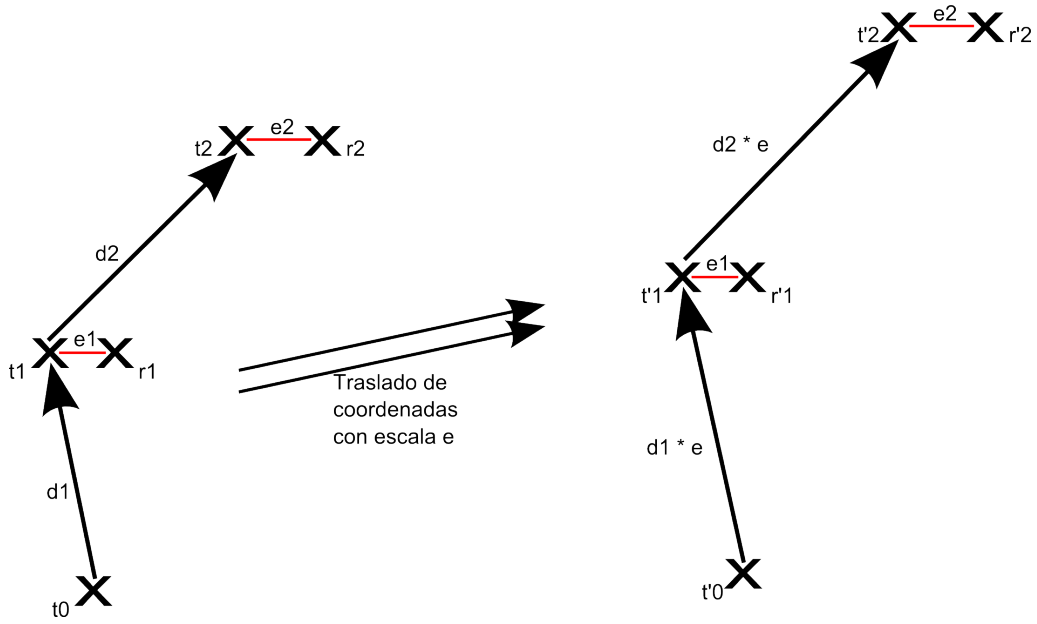


Figura A.20: Concepto de trasladar la posición real a otra definida por el usuario

Podemos pensar que lo más intuitivo es, dado una coordenada inicial obtenida con el GPS (c_0) y la coordenada inicial correspondiente a la posición deseada de la nueva ubicación (c'_0), una vez el robot se desplace y obtenga una nueva coordenada GPS (c_1), realizando la diferencia entre ambas coordenadas:

$$\begin{aligned} \triangle lat &= (c_1.latitud - c_0.latitud) \\ \triangle lon &= (c_1.longitud - c_0.longitud) \end{aligned}$$

y aplicando esa diferencia a la coordenada c'_0 podemos obtener la posición deseada c'_1. Pero hay que tener en cuenta que la precisión del GPS no es tan exacta como se piensa y las mediciones tienen asociadas un error (de entre 1 y 5 metros), por lo que esta aproximación no es válida para calcular la posición en el lugar que nosotros queramos. Además, si queremos que el dispositivo se represente con otra escala (1:10, por ejemplo, es decir, para cada unidad de medida en la realidad, nos movemos 10 unidades al trasladar la posición), entonces un error en la lectura de la posición también será ampliado las mismas unidades que una lectura correcta.

Por lo tanto, si simplemente escalamos las posiciones GPS que obtienen los robots, estaremos escalando también ese error (eso quiere decir que un error de 1

metro sería multiplicado por 100 en el ejemplo anterior). Dado a que es interesante conservar esos errores puesto que pueden ser útiles a la hora de evaluar la robustez de un sistema, hemos propuesto una posible solución a este problema. De forma resumida, nuestra idea se basa en los siguientes puntos:

1. Calcular la posición "teórica" (en adelante p_t) en la que se encuentra el robot en cada instante. Esto es posible puesto que conocemos el punto de partida, la dirección, la velocidad, y el tiempo transcurrido.

2. Obtener el error GPS restando la posición GPS obtenida por el robot a la posición p_t.

3. Trasladar la posición p_t (debidamente escalada) al punto que estemos simulando.

4. Sumarle el error GPS calculado en el paso 2.

Este método nos permitirá mantener las simulaciones a escala mientras que consideraremos el error GPS cometido. A continuación detallamos la solución obtenida para conseguir nuestro objetivo.

Para trabajar con posiciones GPS y obtener, por ejemplo, la distancia entre dos coordenadas GPS, es necesario tener en cuenta que la tierra no es plana y por lo tanto no basta con calcular la resta de las coordenadas. Aunque la tierra es en realidad ligeramente elipsoidal, normalmente se considera una tierra esférica para simplificar los cálculos y la precisión suele ser suficiente en la mayoría de los casos (los errores típicos cometidos con esta aproximación no sobrepasan el 0.3 %).

En nuestro caso necesitamos calcular en primer lugar la coordenada final t_1 en la que nos situaríamos partiendo de una coordenada inicial r_0 ($lat_1 = r_0.latitud$ y $lon_1 = r_0.longitud$), con una cierta orientación ϑ (expresada en radianes, en sentido horario desde el norte) y recorriendo una cierta distancia d (en realidad usamos la distancia angular en radianes d/R, siendo R el radio de la Tierra). Para ello utilizamos la siguiente fórmula:

$$lat_2 = \arcsin(\sin(lat_1) * \cos(d/R) + \cos(lat_1) * \sin(d/R) * \cos(\vartheta)) \quad \text{(A.1)}$$

$$lon_2 = lon_1 + \arctan 2(\sin(\vartheta) * \sin(d/R) * \cos(lat_1), \cos(d/R) - \sin(lat_1) * \sin(lat_2)) \quad \text{(A.2)}$$

Con estas fórmulas ya tenemos calculadas la latitud y la longitud de la coordenada t_1 por lo que procedemos a calcular el error existente entre las coordenadas t_1 y r_1 para calcular el error existente entre ambas (d_e). Para ello, primero calculamos el error con la fórmula del *haversine*:

$$
\begin{aligned}
a &= \sin^2(\triangle lat/2) + \cos(lat_1) * \cos(lat_2) * \sin^2(\triangle long/2) \\
c &= 2 * \arctan 2(\sqrt{a}, \sqrt{(1-a)}) \\
d_e &= R * c
\end{aligned} \tag{A.3}
$$

tomando valores $lat_1 = t_1.latitud$, $lat_2 = r_1.latitud$, Δlat = *(t_1.latitud - r_1.latitud)* y Δlon = *(t_1.longitud - r_1.longitud)*, obteniendo el error existente en kilómetros. Para el cálculo del ángulo del error ϑ_r, usamos la fórmula siguiente:

$$
\begin{aligned}
\vartheta_r &= \arctan 2(\sin(\triangle long) * \cos(lat_2), \\
&\quad \cos(lat_1) * \sin(lat_2) - \sin(lat_1) * \cos(lat_2) * \cos(\triangle long))
\end{aligned} \tag{A.4}
$$

con los valores anteriores de lat_1, lat_2, Δlat y Δlon.

Ya tenemos todos los datos necesarios para poder trasladar las coordenadas a la posición que nosotros queramos. Primero necesitamos una coordenada de origen (denotada como t'_0), que proporcionaremos cuando definamos los movimientos del robot, la misma distancia d y ángulo ϑ iniciales y, en caso de querer representar los movimientos en una escala diferente a la original, el valor de ésta (denotado por s). Empleando de nuevo las fórmulas A.1 y A.2 tomando ahora como distancia angular el valor $(d * s)/R$ al aplicar el factor de escala, obtenemos la posición teórica t'_1 de la posición en nuestra nueva ubicación. Empleando otra vez las fórmulas A.1 y A.2, sustituyendo de nuevo d y ϑ por los valores d_e y ϑ_r calculados previamente, obtendremos la posición real, r'_1, que será la que empleemos para posicionar el dispositivo en el mapa en la simulación de un escenario diferente al real. Para una nueva iteración, las coordenadas t_1 y t'_1 pasan a ser t_0 y t'_0 respectivamente. Con estos cálculos hemos conseguido escalar el movimiento real del dispositivo pero manteniendo el error real original ocasionado por el GPS.

A.9. Estructura de los ficheros

SkyNXT utiliza ficheros de texto para almacenar tanto la configuración de la propia aplicación como las configuraciones y movimientos de los dispositivos NXT. Para ello, utilizamos unos ficheros de texto basados en el lenguaje *XML*. Se ha escogido el lenguaje XML para almacenar los datos ya que dichos ficheros podrán ser leídos por un editor externo o por un analizador estándar, sin necesidad de recurrir a un programa propio para acceder a la información que contiene.

A.9.1. Ficheros de configuración de la aplicación

En primer lugar vamos a analizar la estructura del fichero de configuración de la aplicación de escritorio, de la cual mostramos un ejemplo en el Algoritmo A.2. Observamos que primero tenemos las características de los robots NXT (número de puertos disponibles para sensores y motores), a continuación encontramos la información referente a cada uno de los sensores cargados en la aplicación y que serán los que configuremos en el robot, además de los datos referentes a cada una de las cámaras. Finalmente tenemos la información referente a la pestaña *Map* de la ventana de configuración, que almacena cada una de las opciones que se emplearán en los mapas, así como la ruta a los mismos.

A.9.2. Ficheros de configuración del robot NXT

El siguiente fichero del que analizamos la estructura empleada es el fichero de configuración del robot NXT. Estos ficheros son los que almacenan la configuración de los distintos sensores conectados al dispositivo, así como los parámetros de los mismos para que, en caso de necesitar repetir alguna configuración, tener accesible la información y poder cargarla en la aplicación fácilmente. El ejemplo de fichero del Algoritmo A.3 muestra la estructura de uno de estos ficheros y que analizamos a continuación. En primer lugar tenemos la información asociada al dispositivo configurado (*nombre*, *dirección Bluetooth*, *fecha*, *imagen del dispositivo,* y *numero de sensores*, *cámaras* y *motores*), para tener después la información asociada a cada uno de los sensores y motores que tenga configurado. Para cada uno de los sensores almacenaremos el nombre del mismo (si tiene), el tipo de conexión que posee (cable o inalámbrica) y el puerto (o dirección) al que lo hemos conectado, así como la frecuencia con la que enviará la información a la aplicación. Además, almacenamos también la posición relativa del sensor que será superpuesto en la imagen que represente al dispositivo. Finalmente el fichero contiene la información asociada a los motores configurados en el caso de que queramos mover al robot.

A.9.3. Ficheros de configuración de movimientos

El último fichero usado por la aplicación es aquel en el que almacenamos los movimientos que han de realizar los robots en las pruebas que realicemos. La estructura es bastante similar a los ficheros ya comentados, tal y como vemos en el ejemplo del fichero A.4. Al tener distintas opciones de configuración de los movimientos, obviamente el fichero será diferente en cada uno de los casos, aunque la estructura es bastante similar en todos los casos. En primer lugar tenemos la información asociada al posicionamiento de los robots sobre el mapa (*latitud* y *longitud*, en el caso de que sea diferente a la obtenida, escala del mapa, modelo del objeto y opción de movimiento). En función de la opción de movimiento elegida, el contenido que sigue

Algoritmo A.2 Estructura del fichero de configuración de la aplicación de escritorio

```
<?xml version="1.0" encoding="UTF-8"?>
<Settings>
  <Sensors>
    <sensor type="ports"><Ports_number>4</Ports_number></sensor>
    <sensor type="motors"><Ports_number>3</Ports_number></sensor>
    <sensor type="sensor">
      <Name>Gyro</Name>
      <Manufacturer>Hi-Technics</Manufacturer>
      <Parameters>1</Parameters>
      <ParameterNames><Name>gyro</Name></ParameterNames>
      <Chart enable="false"/>
    </sensor>
    ......
    <sensor type="camera">
      <Manufacturer>Edimax</Manufacturer>
      <Model>Plata</Model>
      <User>admin</User>
      <Password>1234</Password>
      <Video_url>snapshot.jpg</Video_url>
      <URL_type>1</URL_type>
      <PreCommand>camera-cgi/com/ptz.cgi?</PreCommand>
      <PostCommand>move=</PostCommand>
      <Command_type>1</Command_type>
      <Up>up</Up>
      <Down>down</Down>
      <Right>right</Right>
      <Left>left</Left>
      <UpRight>upright</UpRight>
      <UpLeft>upleft</UpLeft>
      <DownRight>downright</DownRight>
      <DownLeft>downleft</DownLeft>
    </sensor>
    ......
  </Sensors>
  <Map>
    <URL>C:\xampp\htdocs\ge\index.html</URL>
    <AutoCenter>false</AutoCenter>
    <Enable3D_Map>false</Enable3D_Map>
    <Enable3D_Cam>false</Enable3D_Cam>
  </Map>
</Settings>
```

será diferente. En el caso del fichero de ejemplo, observamos como almacenamos las trayectorias definidas por el usuario, para cada una se almacena cada uno de los puntos definidos por el usuario mediante ratón, compuesto por las coordenadas X e Y del plano donde se dibuja, el tamaño del punto y las coordenadas X e Y del punto relativas al punto de inicio (que estará definido por el par de coordenadas (0,0)).

Algoritmo A.3 Estructura del fichero de configuración de un robot NXT

```
<?xml version="1.0" encoding="UTF-8"?>
<Config>
  <Information>
    <Name>LAB1</Name>
    <Address>0016530D8DC9</Address>
    <Date>Fri Aug 03 12:43:09 CEST 2012</Date>
    <Path>\img\nxt.jpg</Path>
    <Num_sensors>4</Num_sensors>
    <Num_cams>0</Num_cams>
    <Num_motors>2</Num_motors>
  </Information>
  <sensor type="sensor">
    <Name/>
    <Comm_type>0</Comm_type>
    <Port>1</Port>
    <Address/>
    <delay>1</delay>
    <sensor_type>0</sensor_type>
    <sensor_draw>
      <x>10</x>
      <y>10</y>
    </sensor_draw>
  </sensor>
  ......
  <sensor type="motor">
    <Name/>
    <Port>2</Port>
  </sensor>
  ......
</Config>
```

A.9.4. Ficheros KML de trayectorias

La aplicación también es capaz de almacenar las trayectorias que han sido representadas en el mapa tras la realización de la simulación en un fichero que podrá ser cargado en una aplicación compatible. Para este tipo de ficheros, también basados en lenguaje XML, se ha decidido usar la estructura de un fichero *KML* compatible con Google Earth, siguiendo la estructura que especifica la documentación disponible asociada al programa. Dichos ficheros se van actualizando constantemente cada vez que hay cambios en la posición de los robots, por lo que el plugin de Google Earth actualizará dicha información en el mapa. Podemos ver parte de la estructura del fichero, en concreto la estructura de un *placemark* o marca de posición, en el ejemplo de fichero del Algoritmo A.5.

Algoritmo A.4 Estructura del fichero de movimientos de un robot NXT

```
<?xml version="1.0" encoding="UTF-8"?>
<Movements>
  <Information>
    <X_pos>43.317925</X_pos>
    <Y_pos>-1.98981156</Y_pos>
    <Scale>10</Scale>
    <URLModel>http://localhost/models/rowboat.dae</URLModel>
    <Option>1</Option>
  </Information>
  <Draw_path Scale="1">
    <Point>
      <X>205</X>
      <Y>307</Y>
      <X_size>10</X_size>
      <Y_size>10</Y_size>
      <X_coord>0</X_coord>
      <Y_coord>0</Y_coord>
    </Point>
    ......
    <Point>
      <X>171</X>
      <Y>311</Y>
      <X_size>10</X_size>
      <Y_size>10</Y_size>
      <X_coord>34</X_coord>
      <Y_coord>-4</Y_coord>
    </Point>
  </Draw_path>
</Movements>
```

Algoritmo A.5 Estructura de un Placemark del fichero KML

```
<Placemark>
  <name>LAB2(00165312B446)</name>
  <Model>
    <altitudeMode>relativeToGround</altitudeMode>
    <Location>
      <longitude>-1.9893280298347982</longitude>
      <latitude>43.317325421459124</latitude>
    </Location>
    <Orientation>
      <heading>333</heading>
      <tilt>0</tilt>
      <roll>0</roll>
    </Orientation>
    <Scale>
      <x>1</x>
      <y>1</y>
      <z>1</z>
    </Scale>
    <Link>
      <href>http://localhost/models/rowboat.dae</href>
    </Link>
  </Model>
</Placemark>
```

Apéndice B

Manual de usuario

B.1. Introducción

El Manual de Usuario proporciona una descripción detallada de las funcionalidades de la aplicación SkyNXT. También contiene toda la información e indicaciones necesarias para una correcta utilización y un total aprovechamiento de todas estas funcionalidades. Está destinado a todos aquellos usuarios de la aplicación. Se pretende que tras su lectura, todo usuario tenga un concepto claro de cómo funciona y cómo se maneja el sistema. Se utilizarán muchos gráficos y también un lenguaje conciso y claro a lo largo de todo el manual.

Visión general del manual

A continuación se resumen las distintas partes de que consta el manual de usuario:

- Descripción general del sistema: incluye una descripción general del programa, de los dispositivos Lego Mindstorms y los requisitos necesarios para hacerlo funcionar.
- Guía rápida de instalación y puesta en marcha: describe los pasos básicos a seguir para una correcta instalación tanto de la aplicación de escritorio como de los dispositivos NXT y puesta en marcha de la aplicación.
- Manual de uso: detalla todas las funcionalidades que ofrece la aplicación.

B.2. Descripción general del sistema

Aplicación

La aplicación SkyNXT es una aplicación gráfica que permite realizar simulaciones híbridas sobre sistemas de acceso a datos. Para ello, permitimos la búsqueda, configuración y control de dispositivos móviles que serán los encargados de recopilar datos y enviarlos a la aplicación para representarlos gráficamente.

Dispositivos móviles

El dispositivo móvil es una abstracción de un elemento real que será empleado como una simplificación del mismo en nuestra herramienta de simulación híbrida. Estos dispositivos tendrán la capacidad de ser configurados y controlados remotamente y de enviar todos los datos que recopilen mediante sus sensores a la aplicación.

Requisitos

Los requisitos de la aplicación de escritorio son los siguientes:

- Un PC con sistema operativo Windows.
- Versión 1.6.0_29 de la máquina virtual de Java (*JVM*) instalada.
- Servidor *XAMPP* [25] (o similar) instalado en el ordenador con el servicio *Apache* [26] ejecutándose.
- Plugin de Google Earth [27] instalado.
- LeJOS [14], instalada en el ordenador.
- Conexión Bluetooth en el ordenador.

En cuanto a los robots, en primer lugar es necesario haberlos pareado con el PC a utilizar, y sus requisitos técnicos son:

- Firmware leJOS, versión 0.9.1.
- Programa de control instalado.
- Sensores que vayamos a configurar conectados al dispositivo.

B.3. Guía rápida de instalación

En este apartado se detallan las instrucciones para la instalación tanto de la aplicación de escritorio como de la aplicación para los robots.

Instalación y puesta en marcha de la aplicación de escritorio

A continuación se detallan los pasos necesarios para la instalación de la aplicación de escritorio y su puesta en marcha:

1. Descomprima el archivo SkyNXT_PC.zip que contiene la aplicación en la carpeta deseada.
2. Asegúrese de que todos los archivos han sido descomprimidos adecuadamente.

Una vez descomprimida la aplicación, busque el archivo "SkyNXT.jar" en la ubicación donde descomprimió la aplicación y ejecútelo.

Instalación y puesta en marcha del dispositivo móvil y puesta en marcha.

Como ya se ha comentado, el dispositivo NXT (Figura B.1) ha de tener cargado el firmware leJOS 0.9.1, por lo que tendremos disponibles todas las herramientas que vienen incluidas con dicho firmware, con ellas, hemos de cargar la aplicación específica para el NXT. A continuación se detallan los pasos necesarios para su instalación y puesta en marcha:

Figura B.1: Dispositivo NXT

1. Descomprima el archivo SkyNXT_NXT.zip que contiene el archivo que cargaremos en cada uno de los dispositivos NXT.
2. Vaya a la carpeta "/bin" que se encontrará en la carpeta donde haya instalado previamente leJOS en su ordenador.

3. Con el dispositivo encendido y conectado (bien mediante USB o Bluetooth), ejecute el archivo "nxjbrowse.bat" y seleccione el dispositivo al que quiera cargar el programa de la lista (Figura B.2).

4. Haga click sobre el botón *Upload File* y cargue el archivo "SkyNXT_NXT.nxj" que ha descomprimido previamente en el paso 1 (Figura B.3).

5. Cargue el archivo.

6. Si todo ha ido bien, el dispositivo emitirá un sonido (si el volumen está habilitado en el dispositivo) y aparecerá un mensaje de aviso.

7. Desconecte el dispositivo.

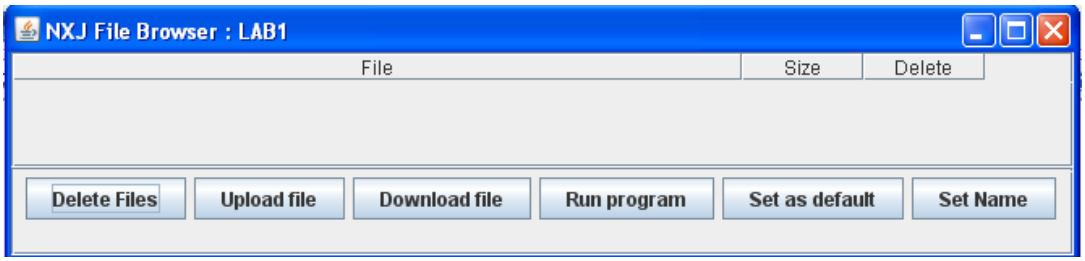


Figura B.2: Programa *NXJ File Browser*

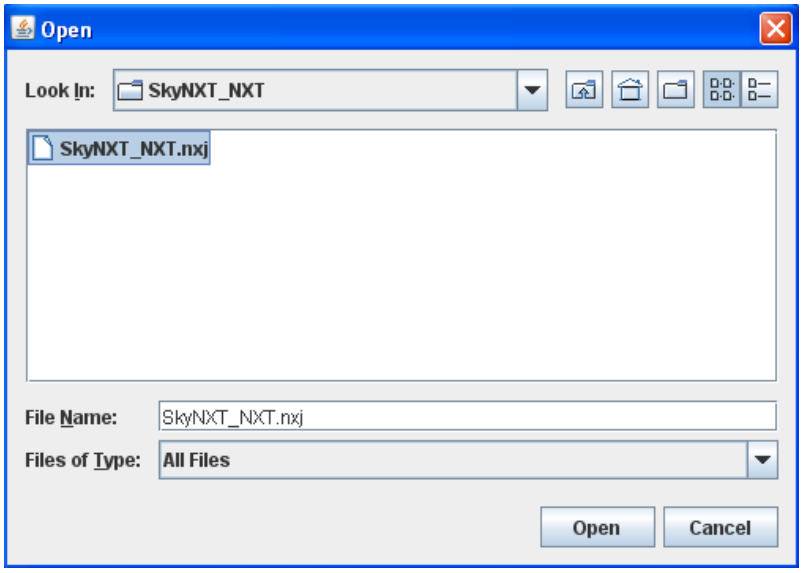


Figura B.3: Seleccionar archivo para cargar al dispositivo

Una vez cargado el programa en el dispositivo, se detallan los pasos para lanzar dicho programa:

1. Desde el menú general de leJOS, nos desplazamos con los botones  o  hasta seleccionar la opción *Files*.

2. Con el botón  accedemos a la lista de los programas cargados en la memoria del dispositivo.

3. Nos desplazamos con los botones  o  hasta posicionarnos sobre el programa cargado.

4. Pulsamos el botón  para acceder a dicho programa.

5. Pulsamos nuevamente el botón  para lanzar el programa (*Execute Program*).

Opcionalmente podemos establecer nuestro programa como defecto, por lo que para ejecutarlo no tendríamos que repetir los pasos previamente descritos. Para ello, sobre las instrucciones anteriores, repetimos los pasos 1 a 4 y a continuación realizamos los siguientes pasos:

1. Pulsamos los botones  o  hasta la opción *Set as Default*.

2. Pulsamos el botón  para hacer que nuestro programa se cargue por defecto.

3. Pulsamos el botón  hasta regresar al menú principal de leJOS.

4. Nos desplazamos con los botones  o  hasta seleccionar la opción *Run Default* (dicha opción es la primera que vemos al encender el dispositivo).

5. Pulsamos el botón  para lanzar el programa.

Si todo ha ido bien, nuestro programa se ejecutará y aparecerá en la pantalla LCD el mensaje "*Waiting...*".

B.4. Manual de uso

La ventana principal de la aplicación de escritorio (ver Figura B.4) contiene las siguientes partes:

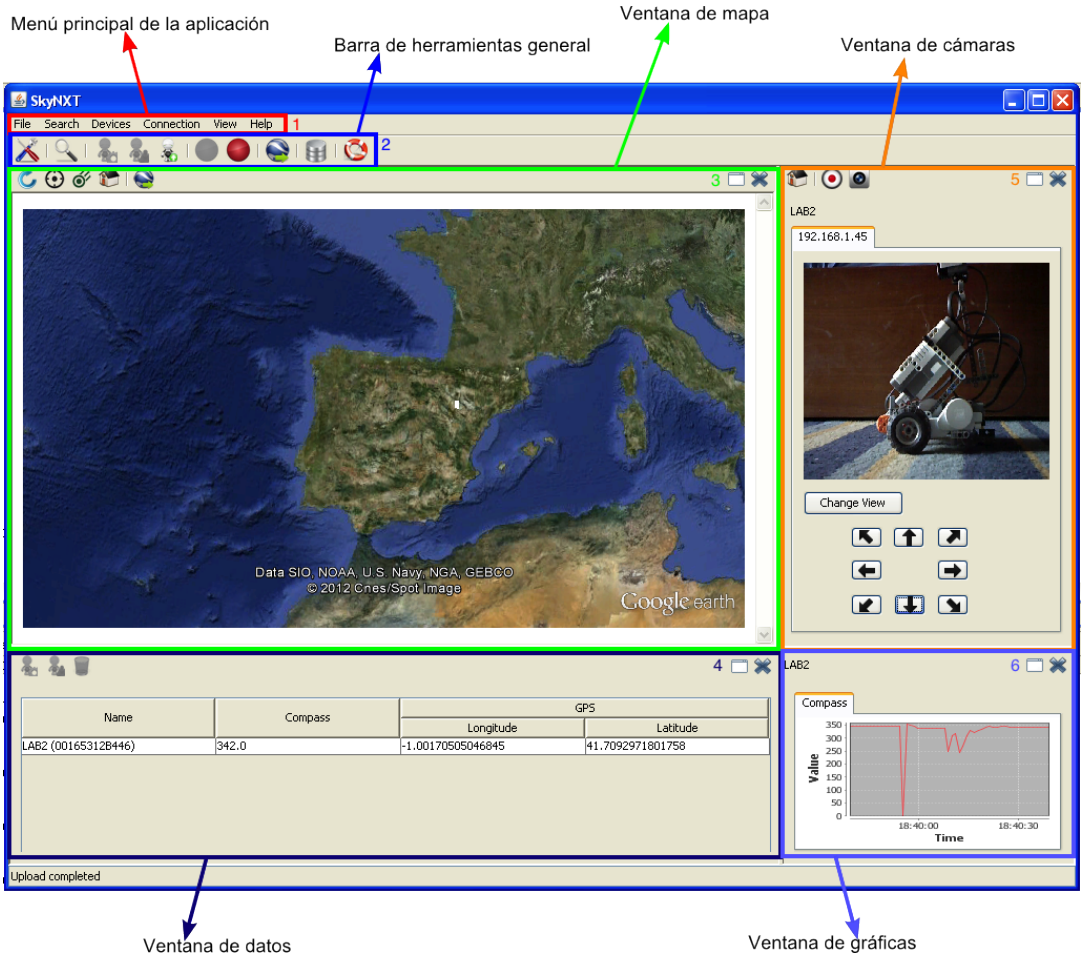


Figura B.4: GUI / Interfaz gráfica de la aplicación SkyNXT

B.4.1. Menú principal

El menú principal se compone de distintos submenús, que se detallan a continuación:

File

El submenú *File* incluye las siguientes funcionalidades (ver Figura B.5):

- *Settings*: abre la ventana que permite configurar las especificaciones de los dispositivos, así como los sensores que tenemos disponibles para configurar nuestros dispositivos, cada uno con sus especificaciones concretas. Esta operación es accesible también desde la barra de herramientas general.
- *Exit*: cierra la aplicación.

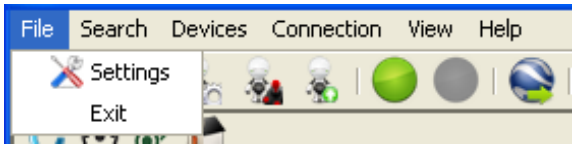


Figura B.5: Opciones del menú *File*

Search

El submenú *Search* incluye las siguientes funcionalidades (ver Figura B.6):

Figura B.6: Opciones del menú *Search*

- *Search Devices*: abre la ventana de búsqueda de dispositivos Bluetooth visibles por el ordenador. Esta operación es visible también desde la barra de herramientas general.

Devices

El submenú *Devices* incluye las siguientes funcionalidades (ver Figura B.7):

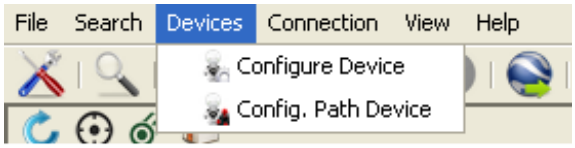


Figura B.7: Opciones del menú *Devices*

- *Configure Device*: abre una ventana que permitirá configurar el dispositivo seleccionado de la tabla de datos con los sensores que tengamos configurados en nuestro sistema. Esta operación es accesible también desde la barra de herramientas general y desde la barra de herramientas de la ventana de datos.
- *Configure Path Device*: abre una ventana que permitirá configurar los movimientos a efectuar por el dispositivo seleccionado de la tabla de datos. Esta operación es accesible también desde la barra de herramientas general y desde la barra de herramientas de la ventana de datos.

Connection

El submenú *Connection* incluye las siguientes funcionalidades (ver Figura B.8):

Figura B.8: Opciones del menú *Connection*

- *Upload Configuration*: una vez configurados todos los dispositivos, cargamos dichas configuraciones a cada uno de los dispositivos conectados. Esta operación es accesible también desde la barra de herramientas general y desde la barra de herramientas de la ventana de datos.
- *Start Devices*: tras subir la configuración a los dispositivos, manda la orden de comienzo a cada uno de los dispositivos. Esta operación es accesible también desde la barra de herramientas general.
- *Stop Devices*: detiene la ejecución de los dispositivos tras haber recibido la orden de comienzo. Esta operación es accesible también desde la barra de herramientas general.

View

El submenú *View* incluye las siguientes funcionalidades (ver Figura B.9):

Figura B.9: Opciones del menú *View*

- *Cameras*: en caso de estar seleccionado (denotado con el *tick* de la izquierda) muestra la ventana de cámaras. En caso de deseleccionarse, cierra dicha ventana.

- *Data*: en caso de estar seleccionado (denotado con el *tick* de la izquierda) muestra la ventana de datos. En caso de deseleccionarse, cierra dicha ventana.
- *Graphics*: en caso de estar seleccionado (denotado con el *tick* de la izquierda) muestra la ventana de gráficos. En caso de deseleccionarse, cierra dicha ventana.
- *Map*: en caso de estar seleccionado (denotado con el *tick* de la izquierda) muestra la ventana de mapa. En caso de deseleccionarse, cierra dicha ventana.

Help

El submenú *Help* incluye las siguientes funcionalidades (ver Figura B.10):

Figura B.10: Opciones del menú *Help*

- *Help*: abre el PDF de este mismo manual. Esta operación es accesible también desde la barra de herramientas general.
- *About*: muestra información sobre esta aplicación.

B.4.2. Barra de herramientas general

Contiene los siguientes botones (de izquierda a derecha, ver Figura B.11) con las funcionalidades que detallamos a continuación:

Figura B.11: Opciones de la barra de herramientas general

-  - *Settings*: ofrece la misma funcionalidad que el menú File/Settings ya explicado.
-  - *Search*: ofrece la misma funcionalidad que el menú Search/Search Devices ya explicado.

-  - *Configure Device*: ofrece la misma funcionalidad que el menú Devices/-Configure Device ya explicado.

-  - *Configure Path Device*: ofrece la misma funcionalidad que el menú Devices/Configure Device ya explicado.

-  - *Upload Configuration*: ofrece la misma funcionalidad que el menú Connection/Upload Configuration ya explicado. El botón quedará deshabilitado tras pulsarse el botón de comienzo hasta que se pulse el botón de parada.

-  - *Start Devices*: ofrece la misma funcionalidad que el menú Connection/Start Devices ya explicado. Posteriormente, el botón queda deshabilitado.

-  - *Stop Devices*: ofrece la misma funcionalidad que el menú Connection/Stop Devices ya explicado. Por defecto el botón está deshabilitado.

-  - *Export KML*: guarda un fichero KML con las trayectorias que han recorrido los dispositivos.

-  - *Help Contents*: abre el archivo PDF de este mismo manual. Esta operación es accesible también desde el submenú *Help*.

B.4.3. Configuración de la aplicación

La aplicación permite la configuración de ciertas opciones del programa así como de los dispositivos que vamos a utilizar. Para acceder a la ventana de configuración basta con hacer click en *Settings* en el menú *File* o bien hacer click en el botón  de la barra de herramientas general. Esta ventana está compuesta por dos pestañas, la primera de ellas, *Devices*, (Figura B.12) es la pestaña desde la que podremos configurar el número de puertos de los dispositivos que vamos a usar, así como de los sensores que tenemos disponibles o de aquellos que podamos tener próximamente.

- *Number of available ports*: aquí se puede establecer el número el número de puertos disponibles en nuestro robot para la conexión de sensores. En el caso del robot NXT, el número de puertos para sensores disponibles es de 4 puertos.

- *Number of available motors*: aquí se puede establecer el número el número de puertos disponibles en nuestro robot para la conexión de motores. En el caso del robot NXT, el número de puertos para motores disponibles es de 3 puertos.

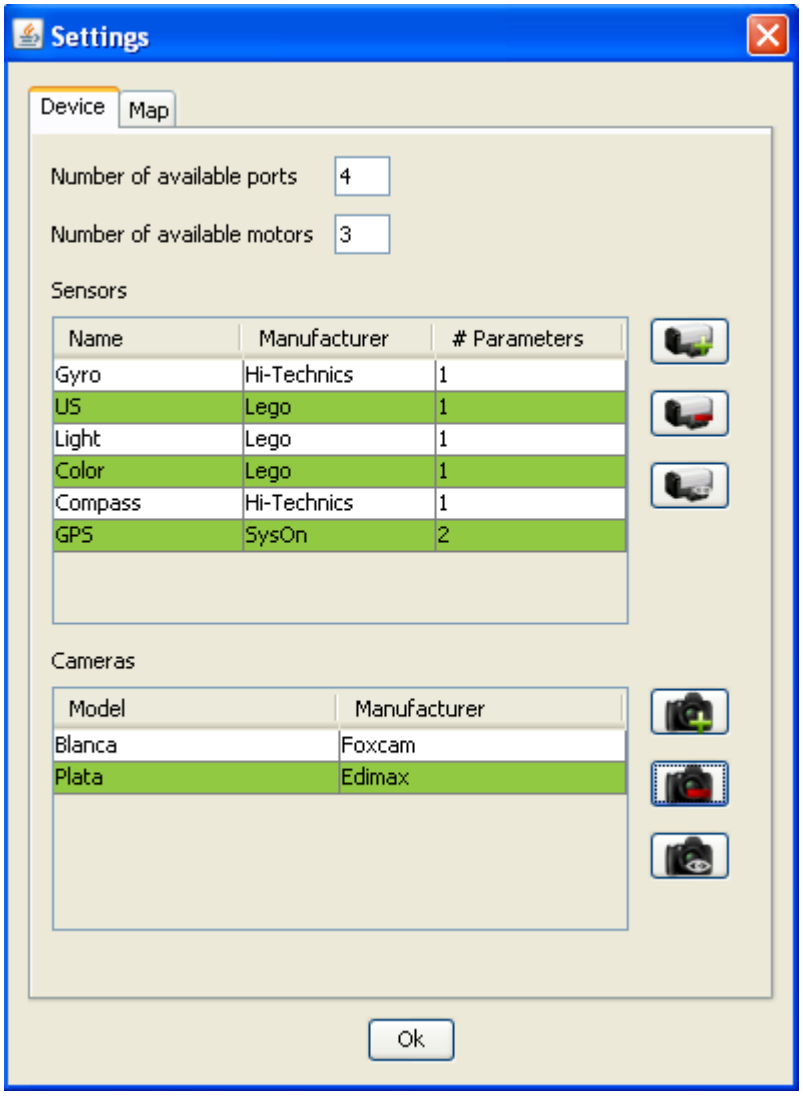


Figura B.12: Pestaña *Device* de la ventana *Setting*

- *Sensors*: en la sección de los sensores, tenemos una tabla en la que podemos ver los sensores que ya tenemos configurados en nuestra aplicación, además de poder configurar nuevos sensores, modificar los ya existentes o eliminar alguno de ellos.

  - *Add sensor*: si hacemos click en el botón  para añadir un nuevo sensor, aparece una ventana destinada para ello (Figura B.13). En ella, tenemos disponibles varios campos:

    - *Name*: especificamos un nombre para el sensor (por ejemplo, GPS).
    - *Manufacturer*: especificamos el fabricante del sensor (por ejemplo, SysOn).

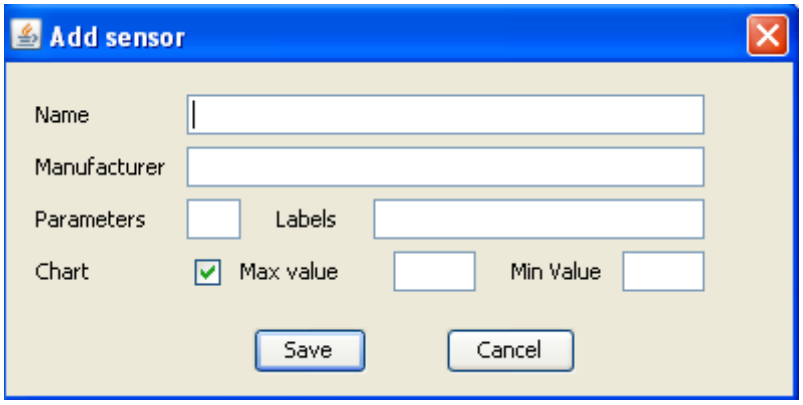


Figura B.13: Ventana *Add sensor* para añadir un nuevo sensor

- *Parameters*: especificamos el número de datos que enviará el sensor (por ejemplo, 2 -latitud y longitud-).
- *Labels*: especificamos qué etiquetas tienen los parámetros que hemos puesto en la casilla anterior. Ha de haber el mismo número de etiquetas que de parámetros. Las etiquetas han de ir separadas por "," (por ejemplo, longitude, latitude).
- *Chart*: marcamos si queremos que este sensor muestre los datos en una gráfica o no. Por defecto, esta casilla está habilitada, por lo que tendremos que añadir también los valores máximo (*Max value*) y mínimo (*Min value*) de la gráfica. Si deshabilitamos la casilla, los datos que envíe este sensor no se insertarán en ninguna gráfica, por lo que las casillas de máximo y mínimo no estarán disponibles.

Tras añadir todos los campos, si todos los datos son correctos, pulsamos sobre el botón *Save* y los datos de este sensor se añadirán al resto de los sensores disponibles. Si por el contrario no queremos almacenar este sensor, pulsamos sobre *Cancel* y el sensor no se añadirá.

- *Remove sensor*: si queremos eliminar un sensor de los que ya tenemos configurados, primero hemos de seleccionarlo de la tabla y posteriormente pulsar el botón  para eliminarlo. Aparecerá un mensaje de confirmación de la acción, evitando que eliminemos por error algún sensor (Figura B.14). Tras confirmar el mensaje, el sensor será eliminado definitivamente. Si no seleccionamos ningún sensor de la tabla y pulsamos el botón de eliminar sensor, la aplicación nos avisará que tenemos que seleccionar previamente un sensor.
- *See sensor*: si queremos ver alguno de los sensores ya configurados y acceder a todos sus parámetros y, en caso de ser necesario, editarlos, primero hemos de seleccionarlo de la tabla y posteriormente pulsar el botón  para visualizar el sensor. Nos aparecerá de nuevo la ventana

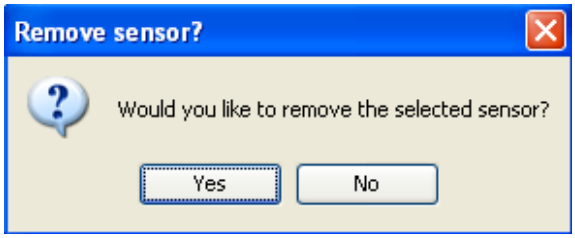


Figura B.14: Mensaje de confirmación de borrado de un sensor

que hemos visto al añadir un sensor (Figura B.13), salvo que esta vez estarán los datos que teníamos disponibles del sensor. Si necesitamos modificar algún dato, editaremos la casilla que queramos y guardamos los cambios (Figura B.15). Si no seleccionamos ningún sensor de la tabla y pulsamos el botón para visualizar el sensor, la aplicación nos avisará que tenemos que seleccionar previamente un sensor.

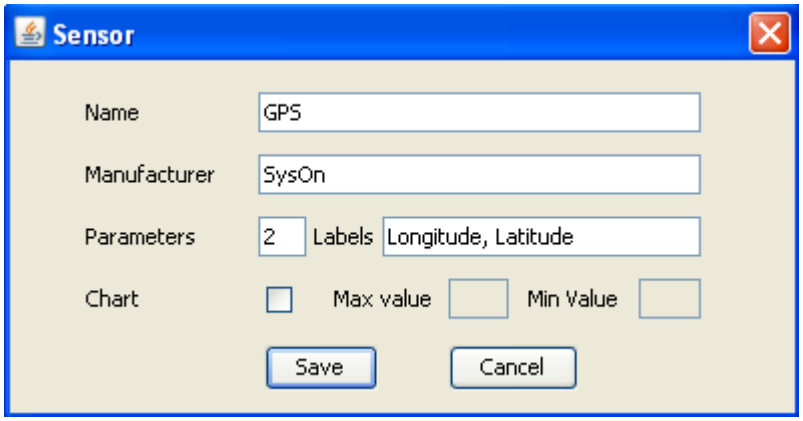


Figura B.15: Ventana *Sensor* para ver un sensor existente

- *Cameras*: en la sección referente a las cámaras, tenemos una tabla en la que podemos ver las cámaras que ya tenemos configuradas en nuestra aplicación, además de poder configurar nuevas, modificar las ya existentes o eliminar alguna de ellas.

  - *Add camera*: si hacemos click en el botón  para añadir una nueva cámara, aparece una ventana destinada para ello (Figura B.16). En ella, tenemos disponibles varios campos que rellenaremos con los datos adecuados:

    - *Manufacturer*: especificamos el fabricante de la cámara (por ejemplo, Edimax).
    - *Model*: especificamos un nombre para la cámara (por ejemplo, Plata).

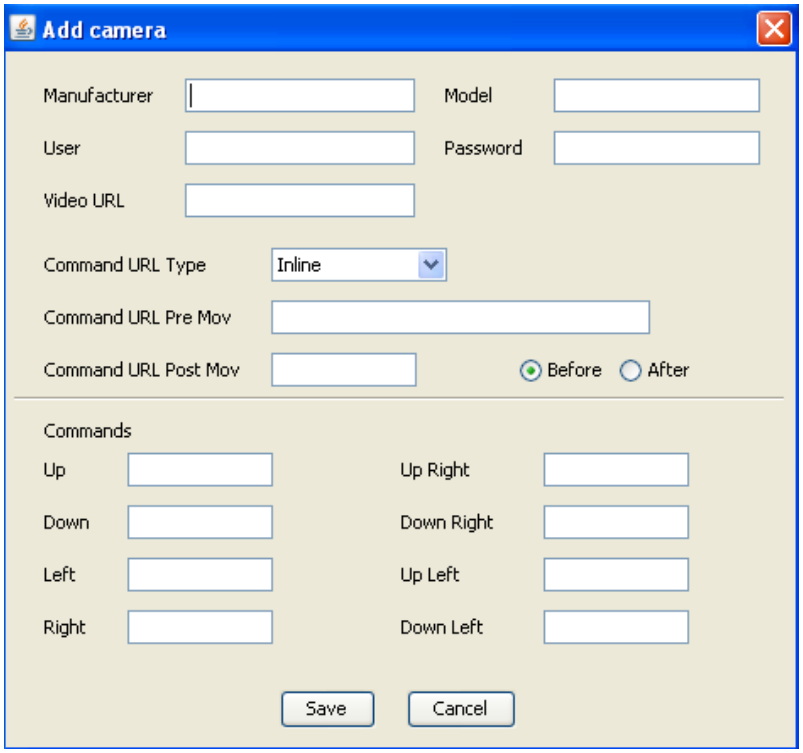


Figura B.16: Ventana *Add camera* para añadir una nueva cámara

- *User*: especificamos el usuario necesario para acceder a las opciones de la cámara. Este dato viene especificado en el manual de uso de la cámara.
- *Password*: especificamos la contraseña (si existe) para acceder a las opciones de la cámara. Este dato viene especificado en el manual de uso de la cámara.
- *Video URL*: especificamos la URL a la que hay que conectarse para ver el streaming de vídeo de la cámara. Dicha dirección viene dada en el manual de comandos CGI de la cámara. La dirección hay que especificarla a partir de la "/" posterior a la dirección IP de la cámara (por ejemplo, http://IP/snapshot.cgi, siendo *snapshot.cgi* es el valor que escribiríamos en el campo).
- *Command URL Type*: para esta opción hemos de saber si el comando para controlar remotamente nuestra cámara ha de mandarse en una única línea o por el contrario ha de ir en envíos diferentes. Si el comando va todo en la misma línea, marcaremos la opción *Inline*; si por el contrario va en envíos separados, seleccionaremos *Split Line*.
- *Command URL Pre Mov*: en este campo indicaremos la dirección a la que hemos de enviar el comando CGI que controla el movimiento de la cámara, teniendo en cuenta que el propio comando en sí no

ha de escribirse ahora, sino que se detallará más adelante ya que es independiente de cada movimiento. El resto de la dirección si que es común para todos los movimientos, que es el parámetro que estamos indicando en este campo. Para conocer la dirección exacta, tendremos que acudir al manual de comandos CGI de la cámara.

- *Command URL Post Mov*: en este campo especificaremos si además de la dirección del comando anteriormente escrito y que enviaremos a la cámara para ser controlada, tenemos que añadir alguna opción adicional junto al mismo, marcando además si ha de ir antes (*before*) o después (*after*) del comando.
- *Comandos Up / Down / Left / Right*: especificamos los comandos CGI que controlan el movimiento de la cámara en las direcciones indicadas. Dichos comandos se encuentran disponibles en el manual de comandos CGI de la cámara.
- *Comandos Up-Right / Down-Right / Up-Left / Down-Left*: especificamos los comandos CGI que controlan el movimiento de la cámara en las direcciones indicadas. Dichos comandos se encuentran disponibles en el manual de comandos CGI de la cámara, aunque es posible que estos comandos no estén disponibles en todos los modelos de cámaras.

Tras añadir todos los campos, si todos los datos son correctos, pulsamos sobre el botón *Save* y los datos de esta cámara se añadirán al resto de las cámaras disponibles. Si por el contrario no queremos tener esta cámara, pulsamos sobre *Cancel* y la cámara no se añadirá.

- *Remove camera*: si queremos eliminar una cámara de las que ya tenemos configuradas, primero hemos de seleccionarla de la tabla y posteriormente pulsar el botón  para eliminarla. Aparecerá un mensaje de confirmación de la acción, evitando que eliminemos por error alguna cámara (Figura B.17). Tras confirmar el mensaje, la cámara será eliminada definitivamente. Si no seleccionamos ninguna cámara de la tabla y pulsamos el botón de eliminar cámara, la aplicación nos avisará que tenemos que seleccionar previamente una cámara.

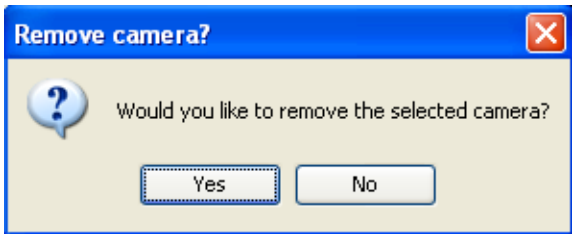


Figura B.17: Mensaje de confirmación de borrado de una cámara

- *See camera*: si queremos ver alguno de las cámaras ya configuradas y acceder a todos sus parámetros y, en caso de ser necesario, editarlos, primero hemos de seleccionarla de la tabla y posteriormente pulsar el botón  para visualizar la cámara. Nos aparecerá de nuevo la ventana que hemos visto al añadir una cámara (Figura B.16), salvo que esta vez estarán los datos que teníamos disponibles de la cámara. Si necesitamos modificar algún dato, editaremos la casilla que queramos y guardamos los cambios (Figura B.18). Si no seleccionamos ninguna cámara de la tabla y pulsamos el botón para visualizar la cámara, la aplicación nos avisará que tenemos que seleccionar previamente una cámara.

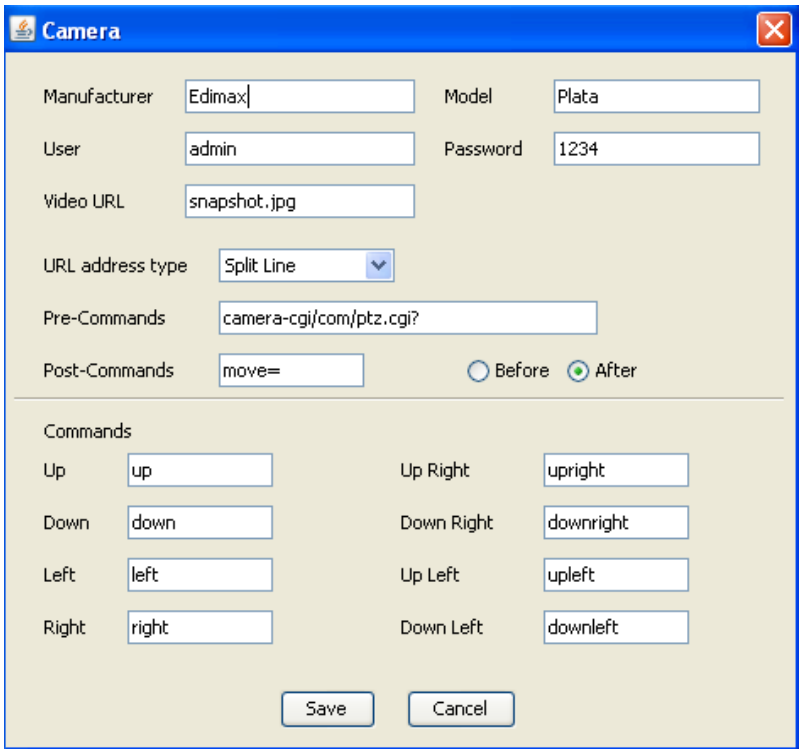


Figura B.18: Ventana *Cámera* para ver una cámara existente

La siguiente pestaña, *Map*, gestiona todo lo referente a los mapas que mostramos en la aplicación y que detallamos a continuación (Figura B.19).

- *URL Location*: desde el botón *Search Index* podemos navegar por el sistema de archivos en busca de la localización de nuestra carpeta donde tengamos el fichero *index.html* que usaremos como mapa. En ella podremos modificar la localización de la carpeta donde tenemos los archivos web, en caso de que decidamos mover la carpeta desde su ubicación original a la carpeta donde tengamos nuestra instalación del servidor Apache local.

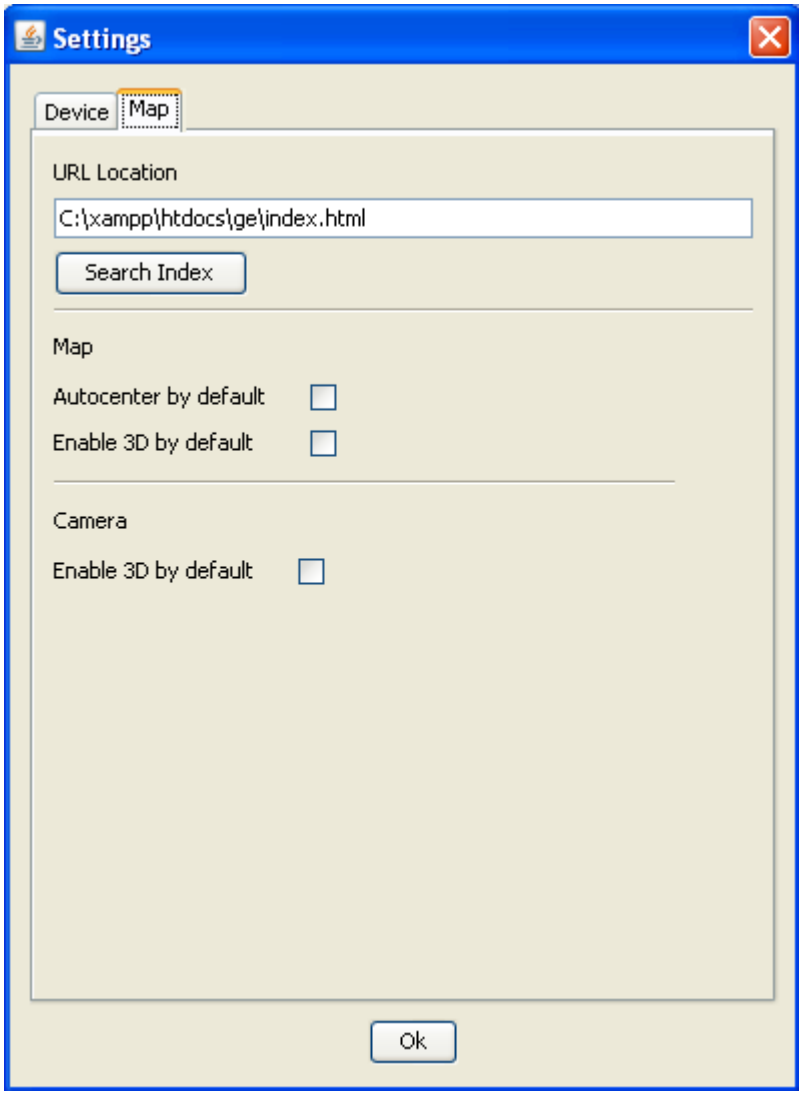


Figura B.19: Pestaña *Map* de la ventana *Settings*

- *Autocenter by default*: habilitando esta opción nos permite posicionar el mapa automáticamente sobre los dispositivos localizados en el mapa.

- *Enable 3D by default (Map)*: habilitando esta opción permite modelar en 3D aquellos edificios disponibles en el mapa que situemos los dispositivos en el mapa.

- *Enable 3D by default (Camera)*: habilitando esta opción permite modelar en 3D aquellos edificios disponibles en el mapa que situemos los dispositivos en la vista simulada de las cámaras.

B.4.4. Búsqueda de dispositivos

Para poder configurar los dispositivos que vamos a usar en nuestra aplicación, previamente tenemos que tener los dispositivos cargados en nuestra aplicación. Para ello, la aplicación permite realizar una búsqueda de los dispositivos Bluetooth disponibles y que sean visibles por la aplicación. Podemos accede a esta ventana bien desde la opción *Search devices* desde el menú *Search* o bien desde el botón  de la barra de herramientas.

Es importante que los dispositivos estén en el menú principal para que sean localizados, ya que si algún programa está en ejecución, los dispositivos no podrán ser localizados y no aparecerán en la tabla.

Una vez que estamos en dicha ventana, si presionamos el botón *Search*, la aplicación procederá a buscar los dispositivos visibles. Una vez finalice la búsqueda, aquellos dispositivos detectados aparecerán en la tabla. Como el programa hace una búsqueda de dispositivos Bluetooth visibles, es posible que además de los dispositivos NXT, aparezcan otros dispositivos tales como teléfonos móviles, GPS, etc. Podemos seleccionar aquel o aquellos dispositivos que vayamos a emplear seleccionándolos con el ratón. Si queremos seleccionar más de un dispositivo, podremos hacerlo manteniendo pulsada la tecla "*Ctrl*" del teclado mientras seleccionamos aquellos dispositivos (Figura B.20). Una vez tenemos seleccionados todos los dispositivos, confirmamos la búsqueda con el botón *OK* y los dispositivos seleccionados pasarán a la tabla de la ventana de datos. Si por el contrario pulsamos el botón *Cancel*, no se añadirá ningún dispositivo de los seleccionados.

La opción de búsqueda es destructiva con los dispositivos ya buscados previamente y que aparezcan en la tabla de datos, por lo que cada vez que pulsemos el botón de *OK*, aquellos dispositivos seleccionados para cada una de las búsquedas posteriores, reemplazarán a los dispositivos existentes en la tabla de datos.

B.4.5. Panel de datos

El panel de datos es uno de los paneles principales de la aplicación y, al iniciar el programa, es uno de los que tendremos disponibles inicialmente, aunque no tendremos ningún dato en él. Cuando realicemos la búsqueda de dispositivos, aquellos seleccionados pasarán a estar en la tabla de dicho panel y, para cada modificación de éstos, los cambios aparecerán reflejados en las columnas de la tabla. Además, cuando los dispositivos reciban la orden de comienzo y empiecen a transmitir datos, la última actualización de los mismos aparecerá reflejada en la celda correspondiente al sensor de cada robot. A continuación se explican las diferentes partes que componen esta ventana y sus funcionalidades (Figura B.21):

1. Barra de herramientas del panel de datos.

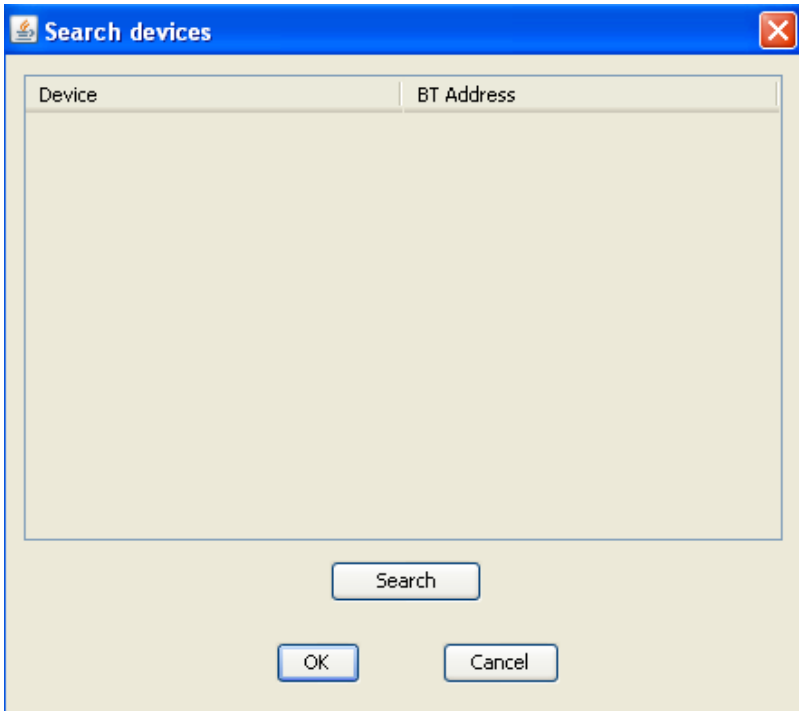


Figura B.20: Ventana para la búsqueda de dispositivos Bluetooth

2. Tabla de datos.

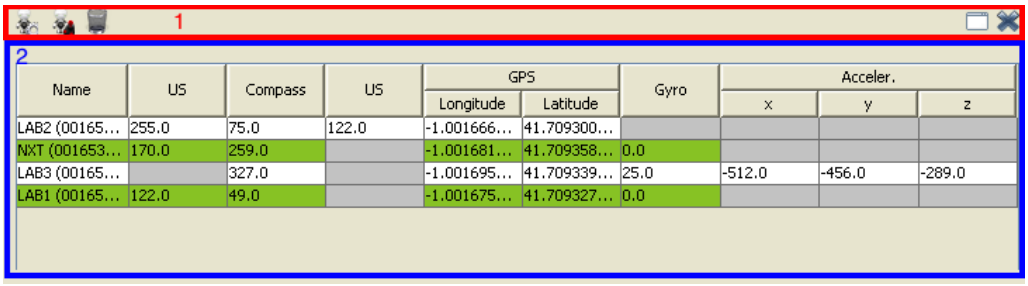


Figura B.21: Panel de datos

Barra de herramientas del panel de datos

Contienen los siguientes botones de izquierda a derecha (Figura B.22):

-  - *Configure Device*: ofrece la misma funcionalidad que el menú Devices/-Configure Device ya explicado.

Figura B.22: Opciones de la barra de herramientas de la ventana de datos

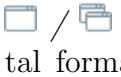
-  - *Configure Path Device*: ofrece la misma funcionalidad que el menú Devices/Configure Path Device ya explicado.
-  - *Remove device*: elimina el dispositivo seleccionado.
-  - *Maximize / Minimize*: maximiza o minimiza el panel de datos de tal forma que ocupe la totalidad de la ventana de la aplicación o la parte disponible por defecto.
-  - *Close*: cierra el panel de datos. Una vez cerrado es posible abrirlo nuevamente desde el menú *View*, seleccionando *Data* que, al estar cerrado, aparecerá desmarcado.

Tabla de datos

Como ya hemos comentado, la tabla de datos refleja toda la información referente a los dispositivos seleccionados así como la información que recibamos de los sensores que configuremos. Inicialmente dicha tabla permanece vacía ya que al cargar el programa no tenemos emparejado ningún dispositivo móvil, pero posteriormente los cambios aparecerán reflejados en ella. La primera actualización se produce al confirmar los dispositivos buscados tal y como hemos explicado en la Sección B.4.4. Todos aquellos dispositivos que seleccionemos aparecerán cada uno en una fila indicando, en primer lugar, el nombre del dispositivo para que podamos identificarlo fácilmente. Para configurar cada uno de los dispositivos, tendremos que seleccionar aquel que queramos modificar haciendo click con el botón izquierdo. La fila entera quedará marcada en color azul para distinguirla del resto de dispositivos y podremos pulsar los botones correspondientes para hacer la acción que deseemos. Si queremos seleccionar otro dispositivo, podemos hacer click de nuevo con el botón izquierdo sobre su fila y pasará a estar seleccionada dicha fila. Si por el contrario, no queremos tener ninguna fila seleccionada, podemos hacer click con el botón derecho sobre la fila marcada y dejará de estar seleccionada.

Cada vez que configuremos un dispositivo y añadamos diferentes sensores a los dispositivos, éstos aparecerán en la tabla al confirmar la configuración, un sensor en cada columna, agrupando en ella los posibles parámetros que tenga dicho sensor. Para cada fila de dispositivos, si éste no tiene configurado algún sensor que otro dispositivo si que tiene, la celda correspondiente aparecerá en un tono más oscuro para indicar que en dicha celda no se insertará ningún dato. Para el resto de celdas, cuando comencemos la prueba, los datos obtenidos de los sensores y que envíen

los dispositivos a la aplicación, se escribirán en las celdas correspondientes a cada sensor para cada uno de los dispositivos. En el ejemplo de la Figura B.23 podemos ver la combinación de colores de la tabla cuando hemos configurado varios robots con distintos sensores que están enviando datos a la aplicación.

Name	US	Compass	US	GPS		Gyro	Acceler.		
				Longitude	Latitude		x	y	z
LAB2 (00165...	255.0	75.0	122.0	-1.001666...	41.709300...				
NXT (001653...	170.0	259.0		-1.001681...	41.709358...	0.0			
LAB3 (00165...		327.0		-1.001695...	41.709339...	25.0	-512.0	-456.0	-289.0
LAB1 (00165...	122.0	49.0		-1.001675...	41.709327...	0.0			

Figura B.23: Tabla de datos

Configuración de un dispositivo

Antes de comenzar la prueba, hemos de configurar cada uno de los dispositivos con los sensores, cámaras y motores que vayamos a emplear en nuestra simulación con aquellos que ya tengamos cargados previamente en el sistema como hemos visto en el apartado B.4.3. Aquellos sensores y cámaras que ya tengamos cargados, podrán ser seleccionados. Además también hemos de tener en cuenta el número máximo de sensores y motores que podemos conectar físicamente y que hemos especificado en dicho apartado.

Con ello, si seleccionamos un dispositivo de la tabla y pulsamos el botón  de configurar dispositivo, nos aparecerá la ventana destinada para ello (ver Figura B.24). En dicha ventana podemos distinguir diferentes apartados. Primero podemos observar la imagen por defecto del dispositivo y sobre la que aparecerán de forma simplificada los sensores que añadamos al mismo. También tenemos los botones que nos permiten cargar una configuración previa, bastante útil en el caso de configurar de forma idéntica múltiples sensores o de salvar dicha configuración para posteriores usos, y para modificar la imagen que aparece por defecto por otra de nuestra elección. Finalmente, observamos en la parte inferior la sección destinada a los sensores, cámaras y motores del dispositivo y los botones que nos permiten añadir dichos elementos al mismo.

Procedemos a explicar cada una de estas partes con más detalle:

-  - *Cargar configuración*: si previamente ya hemos guardado la configuración de un dispositivo, podemos usarla en este nuevo dispositivo para, de forma rápida y sencilla, completar el proceso. Para ello, el navegador de archivos carga por defecto la ruta donde la aplicación guarda por defecto los archivos de configuración. Cuando hemos localizado el archivo que queramos usar, confirmamos la carga del archivo y el robot tendrá los datos que contenía el archivo seleccionado.

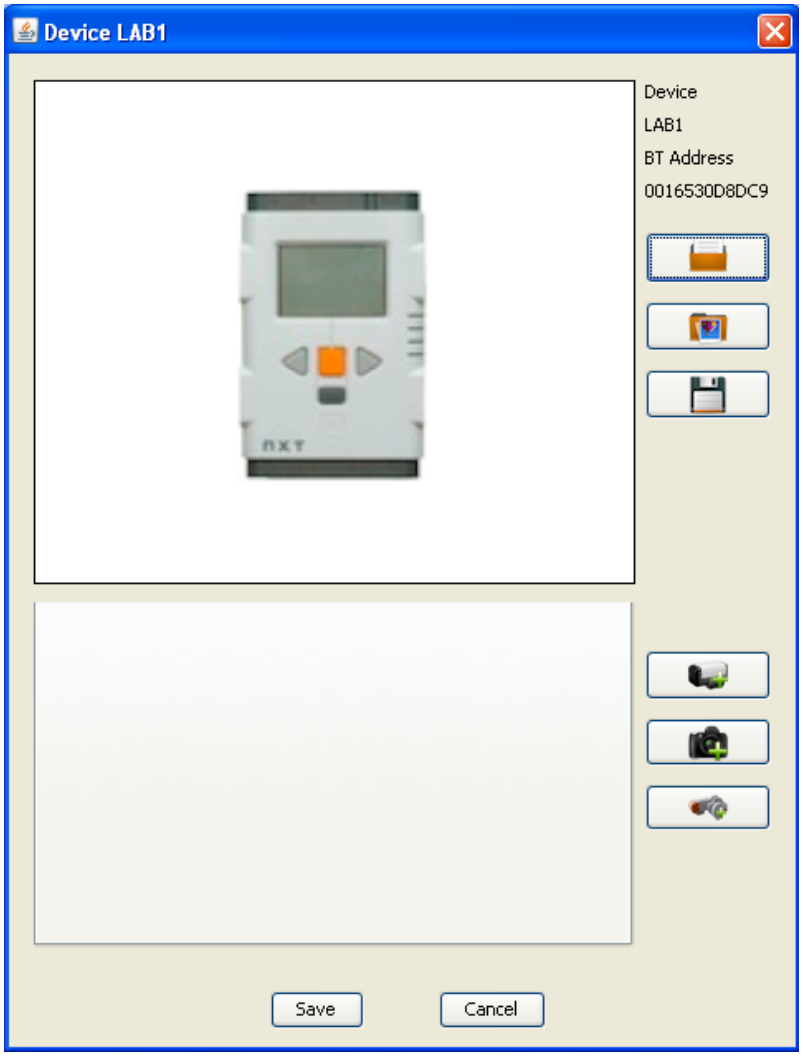


Figura B.24: Ventana para la configuración de un dispositivo

-  - *Guardar configuración*: si queremos exportar a un fichero todos los datos referentes a los sensores, cámaras y motores que hemos configurado en el dispositivo para, en pruebas posteriores usar dicha configuración o bien configurar varios dispositivos de la misma manera, esta opción genera un fichero con todos los datos que se guardará en la carpeta *saves* del programa. Al guardar el fichero, el programa nos avisa de ello para saber que el fichero ha sido creado con éxito.

-  - *Cargar imagen*: como podemos observar al configurar un dispositivo, en la ventana donde es representado aparece una imagen sobre la que los sensores que añadamos podrán sobre-impresionarse. Pero si la imagen no es similar a la de nuestro dispositivo o, simplemente, queremos usar otra que

sea más de nuestro agrado, con esta opción podemos escoger otra imagen buscándola en el sistema de ficheros del ordenador mediante el selector de ficheros que se abre. Tras confirmar la imagen seleccionada, ésta aparecerá en el panel sustituyendo a la que previamente aparecía en él.

- *Panel de imagen*: en dicha ventana aparecen tanto la imagen que aparece por defecto o la que hemos seleccionado a nuestro gusto y la representación simbólica de los sensores y cámaras que añadamos al dispositivo. Podemos mover libremente por el panel seleccionándolo con el ratón y moviéndolo a la posición que nosotros queramos. El sensor seleccionado aparecerá resaltado en otro color y además la pestaña asociada a dicho sensor pasará a ser la pestaña activa.
-  - *Añadir sensor*: para cada sensor que queramos añadir a nuestro dispositivo, podemos pulsar sobre este botón y aparecerán en el panel de los sensores en la pestaña correspondiente para que podamos configurarlos, además de crearse una representación del sensor en el panel que podremos mover hasta la posición simbólica que deseemos. Como observamos en la Figura B.25, se nos piden una serie de datos para cada sensor, por lo que podremos rellenarlo adecuadamente:

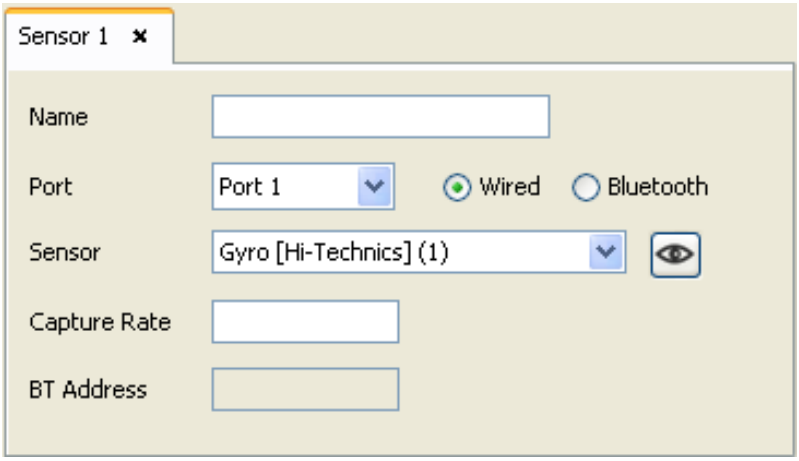


Figura B.25: Pestaña de configuración de sensor

  - *Name*: podemos darle un nombre simbólico al sensor.
  - *Port*: si el sensor que configuramos es un sensor con cable, marcaremos la opción *Wired* y seleccionaremos un puerto de la lista desplegable. Si por el contrario es un sensor inalámbrico, marcaremos la opción *Bluetooth*. Al marcar dicha opción observamos cómo la lista desplegable de puertos queda deshabilitada y podemos escribir en el campo *BT Address*, que estará habilitado. Si el sensor es con cable, la aplicación no permite que

distintos sensores estén configurados con el mismo puerto y, ante esta situación, nos avisará de ello para poder corregirlo.

- *Sensor*: como previamente hemos configurado los sensores disponibles en el apartado B.4.3, éstos aparecerán en la lista desplegable que nos aparece y podremos seleccionar aquel que queramos usar. Si tenemos dudas de las características de algún sensor, podemos ver los datos asociados al mismo con el botón que tenemos junto a la lista, apareciendo una ventana similar a la de la Figura B.13, salvo que los campos están deshabilitados.
- *Capture rate*: en este campo establecemos el intervalo en segundos entre envío y envío de los datos por parte del sensor desde el robot a la aplicación, siendo 1 segundo el tiempo mínimo.
- *BT Address*: como hemos indicado al hablar del campo *Port*, si seleccionamos la opción de conexión Bluetooth, dicha casilla estará disponible y tendremos que indicar en ella la dirección Bluetooth del dispositivo que vayamos a usar. Dicha dirección la podremos encontrar en el propio dispositivo, posiblemente escrita en alguna pegatina o distintivo, en el manual o en la caja del mismo.

Si hemos añadido algún sensor de más o queremos eliminar alguno de ellos, observamos que en la pestaña aparece una pequeña cruz. Si pulsamos dicha cruz el panel será eliminado y la representación del sensor en el panel desaparecerá.

-  - *Añadir cámara*: para cada cámara que queramos añadir a nuestro dispositivo, podemos pulsar sobre este botón y aparecerán en el panel de las cámaras en la pestaña correspondiente para que podamos configurarlas, además de crearse una representación de la cámara en el panel que podremos mover hasta la posición simbólica que deseemos. Como observamos en la Figura B.26, se nos piden una serie de datos para cada cámara, aunque menos que en la anterior pestaña:

  - *Camera*: como previamente hemos configurado las cámaras disponibles en el apartado B.4.3, éstas aparecerán en la lista desplegable que nos aparece y podremos seleccionar aquella que queramos usar. Si tenemos dudas de las características de alguna de las cámaras, podemos ver los datos asociados a la misma con el botón que tenemos junto a la lista, apareciendo una ventana similar a la de la Figura B.16, salvo que los campos están deshabilitados.
  - *Address*: al estar usando cámaras IP, necesitamos saber la dirección a la que hemos de conectarnos, como por ejemplo, una dirección IP pública o local desde la que la cámara estará emitiendo el vídeo para que lo visualicemos.

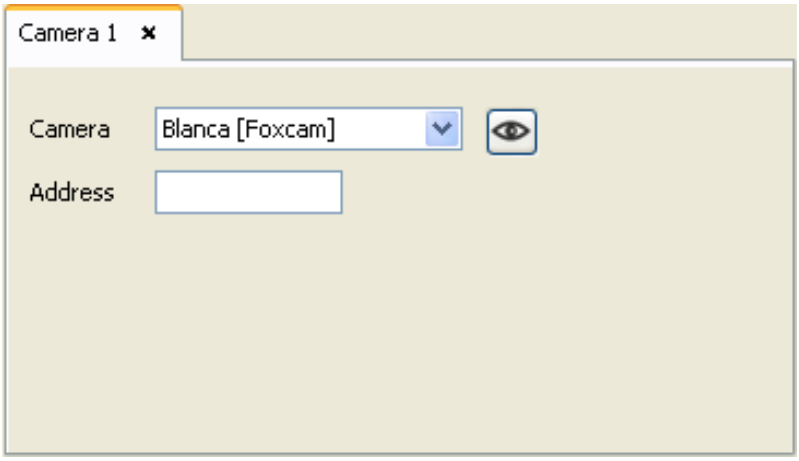


Figura B.26: Pestaña de configuración de cámara

Si hemos añadido alguna cámara de más o queremos eliminar alguna de ellas, observamos que en la pestaña aparece una pequeña cruz. Si pulsamos dicha cruz el panel será eliminado y la representación de la cámara en el panel desaparecerá.

-  - *Añadir motor*: para cada motor que queramos añadir a nuestro dispositivo, podemos pulsar sobre este botón y aparecerán en el panel de los motores en la pestaña correspondiente para que podamos configurarlo. Como observamos en la Figura B.27, se nos piden una serie de datos para cada motor:

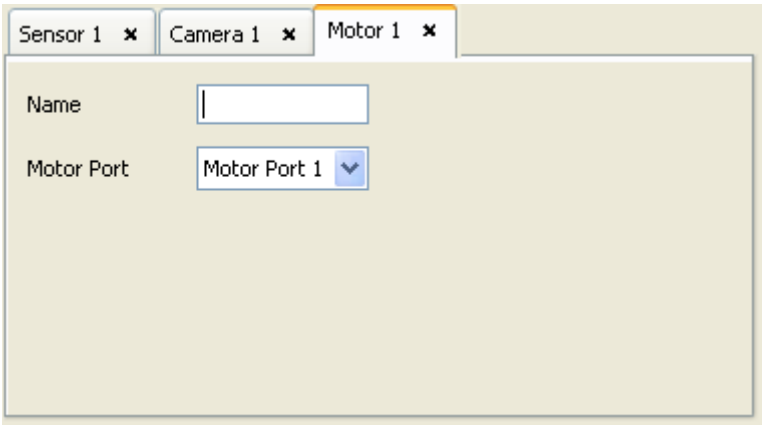


Figura B.27: Pestaña de configuración de motor

- *Name*: podemos darle un nombre simbólico al motor.
- *Port*: en la pestaña de configuración de dispositivos vista en el apartado B.4.3, hemos establecido el número de motores que nuestro dispositivo tiene disponible. El número que hemos indicado equivale al número de

puertos máximos que aparecerán en la lista desplegable. En ella seleccionaremos el puerto al que hemos conectado dicho motor. Como en el caso de los sensores, si configuramos varios motores, la aplicación nos indicara de ello y podremos corregir el error.

Si hemos añadido algún motor de más o queremos eliminar alguno de ellos, observamos que en la pestaña aparece una pequeña cruz. Si pulsamos dicha cruz el panel será eliminado.

Configuración de los movimientos de un dispositivo

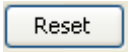
Tras configurar los sensores de un dispositivo, hemos de configurar también los movimientos que ha de realizar el dispositivo móvil. Para ello, procederemos de forma análoga a como hemos procedido para configurar los sensores de un dispositivo. Seleccionamos aquel que queramos configurar y esta vez presionamos el botón  , por lo que nos aparecerá la ventana mediante la cual configuraremos el movimiento (ver Figura B.28).

Observamos que nos aparecen varias opciones de movimientos, estando seleccionada por defecto la opción de que el dispositivo NXT no realice ningún movimiento (*None*), por lo que la ventana no mostrará ninguna opción de configuración.

Si seleccionamos la opción de dibujar manualmente la trayectoria que nosotros elijamos (*Draw Path*), la ventana nos mostrará el panel destinado para ello, que podemos observar en la Figura B.29. Para cada click que hagamos con el botón izquierdo del ratón en el panel destinado para ello, iremos añadiendo nuevos puntos a la trayectoria final que reproducirá el dispositivo. Además, para cada par de puntos, se nos muestra la distancia existente entre ambos, correspondiéndose a la distancia en centímetros existente entre dos puntos y que podemos modificar aumentando la escala existente en el mapa.

Observamos también que tenemos disponibles dos botones:

- 

, el cual nos permitirá eliminar aquel punto que esté seleccionado en color rojo de la trayectoria definida hasta el momento, sin alterar la situación del resto de los puntos.
- 

, mediante el cual borraremos la totalidad de los puntos que hayamos definido, pudiendo definir una nueva trayectoria.

El proceso de eliminar un punto de una trayectoria definida se puede observar en la Figura B.30, mediante la cual podemos observar el antes y el después de una

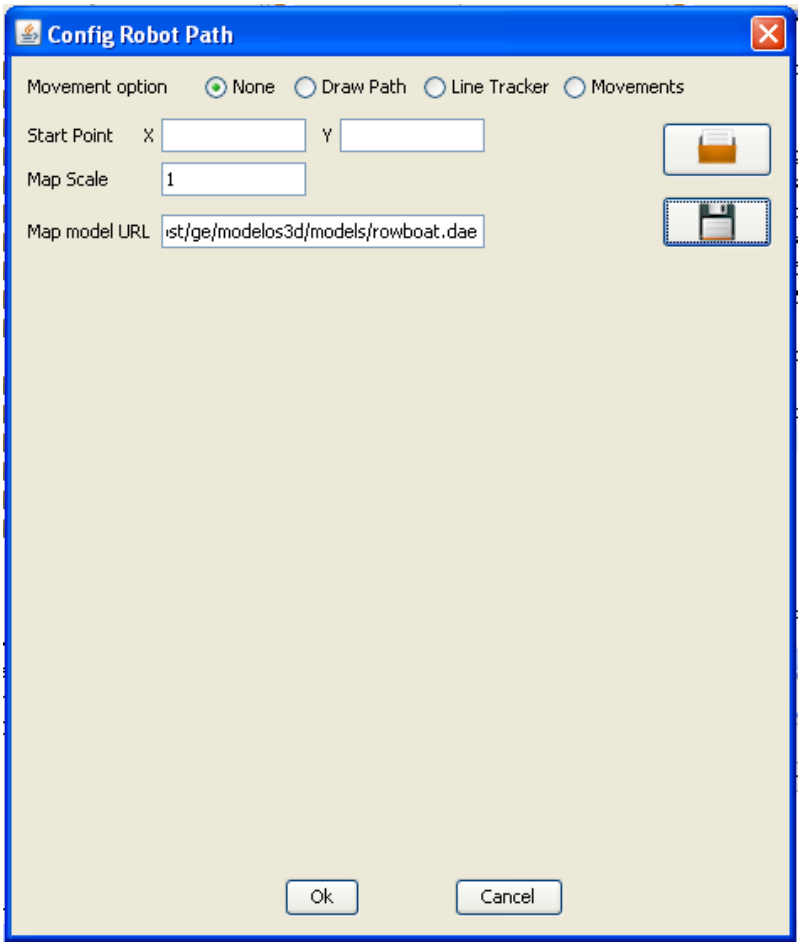


Figura B.28: Configuración de los movimientos de un dispositivo

trayectoria tras seleccionar uno de los puntos de la misma y eliminarlo de la manera que acabamos de explicar.

La tercera opción de movimiento es aquella denominada *Siguelíneas (Line Tracker)*. Seleccionándola, nos da la opción de que el dispositivo móvil siga una trayectoria predefinida mediante el dibujo de una línea de color negro que estará dibujada directamente sobre la superficie en donde situaremos a los dispositivos móviles.

Finalmente la última opción de control se basa en movimientos (*Movements*) realizados de forma manual mediante la pulsación selectiva de teclas del teclado. Como podemos ver en la Figura B.31, se nos permite configurar los movimientos del dispositivo con las teclas de nuestra selección. Para ello, cada una de las casillas nos permite asociar una dirección a una tecla en concreto.

Además de las opciones de configuración, podemos personalizar el aspecto y posición de los robots sobre el mapa, así como la distancia que recorren en cada actualización de los datos. Si equipamos al NXT con un sensor GPS, escribiendo en las casillas correspondientes la *Latitud* (x) y *Longitud* (y) de unas coordenadas GPS

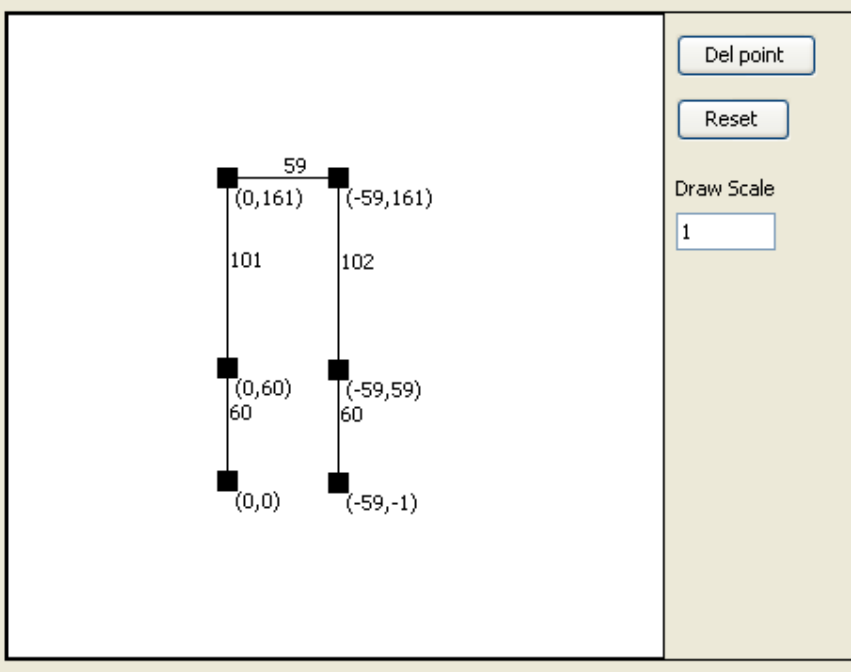


Figura B.29: Panel de definición de trayectorias (*Draw Path*)

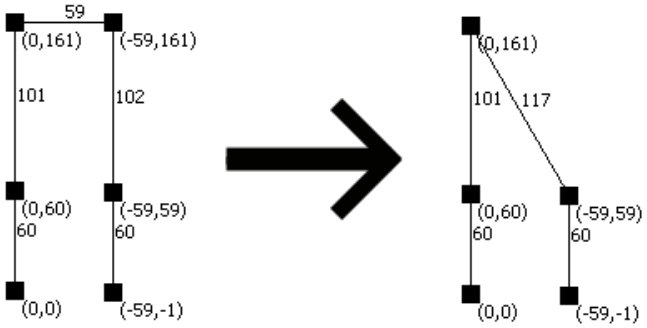


Figura B.30: Trayectoria antes y después de la eliminación de uno de sus puntos

válidas, podemos cambiar la localización sobre la que representaremos al dispositivo en el mapa, situando al dispositivo sobre las coordenadas dadas. En caso de dejar dichas casillas vacías, la localización será la real obtenida del GPS. De la misma forma que podemos posicionar al dispositivo sobre una localización determinada, también podemos escalar el movimiento que realice alterando la escala del movimiento realizado. Finalmente, la última opción nos permite modificar la representación del robot sobre el mapa, detallando la dirección URL donde tenemos disponible un fichero .dae

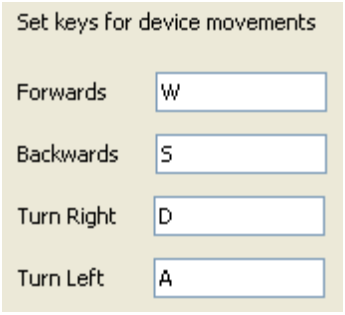


Figura B.31: Selección de teclas para los movimientos manuales

en el que tendremos un modelo 3D del dispositivo que queramos representar.

Todos estos cambios pueden ser guardados en un fichero externo que podremos además cargar en posteriores pruebas, evitando tener que rellenar cada una de las opciones nuevamente. Pulsando el botón  podremos cargar un fichero de movimientos que tengamos guardado de una prueba anterior, seleccionando el fichero mediante una ventana de selección y cargándose automáticamente el fichero con el contenido, mientras que pulsando  , procederemos a salvar los movimientos seleccionados en un fichero de movimientos que podremos cargar mediante el botón anteriormente comentado, recibiendo un mensaje por parte de la aplicación indicando el éxito o fallo de la acción. Es importante comentar que si queremos salvar los movimientos a un fichero externo, hemos de realizar dicho proceso antes de confirmar los movimientos y volver a la ventana principal.

Eliminación de un dispositivo

Al igual que con la configuración de un dispositivo y la configuración de los movimientos de un dispositivo de los disponibles en la tabla, el proceso para eliminar un dispositivo de la tabla es, en primer lugar, seleccionar el dispositivo deseado de la tabla para después pulsar el botón  , *Remove device,* y proceder a la eliminación del mismo de la aplicación, no teniendo ya disponible el dispositivo para interactuar con él. El proceso de eliminar un dispositivo únicamente está disponible mientras no tengamos en ejecución los dispositivos móviles, es decir, mientras estamos en el proceso de configuración de los dispositivos.

B.4.6. Panel de mapa

El panel de mapa es uno de los paneles principales de la aplicación y, al iniciar el programa, es uno de las que tendremos disponibles inicialmente, aunque no tendremos ningún dispositivo en el mapa que aparezca. Una vez que los robots hayan sido

configurados y hayamos comenzado la prueba, aparecerán posicionados en el mapa. Para cada actualización de la posición real del robot, el mapa también será actualizado. A continuación se explican las diferentes partes que componen esta ventana y sus funcionalidades (Figura B.32):

1. Barra de herramientas del panel de mapa.
2. Mapa.

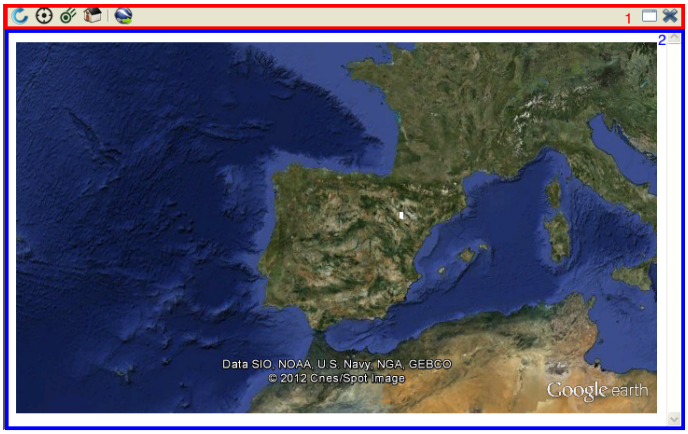


Figura B.32: Panel de mapa

Barra de herramientas del panel de mapa

Contienen los siguientes botones de izquierda a derecha (Figura B.33):

Figura B.33: Opciones de la barra de herramientas de la ventana de mapa

- - *Refresh*: refrescamos el mapa.
- - *Center*: posiciona la vista del mapa sobre los dispositivos una vez conectados.
- / - *Autocenter Enable* / *Disable*: permite activar o desactivar el seguimiento automático de los dispositivos una vez conectados.
- / - *3D Enable* / *Disable*: permite activar o desactivar la vista en 3D de los edificios existentes en el mapa.

-  - *Export KML*: guarda un fichero KML con las trayectorias que han recorrido los dispositivos.

-  /  - *Maximize* / *Minimize*: maximiza o minimiza el panel del mapa de tal forma que ocupe la totalidad de la ventana de la aplicación o la parte disponible por defecto.

-  - *Close*: cierra el panel del mapa. Una vez cerrado es posible abrirlo nuevamente desde el menú *View*, seleccionando *Map* que, al estar cerrado, aparecerá desmarcado.

Mapa

Como ya hemos comentado, el mapa usado en esta ventana es proporcionado por el plugin de Google Earth, que cargamos mediante una página alojada localmente. Para ello es necesario tener en ejecución el servicio Apache para que la página pueda ser abierta de forma correcta si hemos guardado la página en alguna ruta perteneciente al servidor Xampp (htdocs o similar). De ser así y tener el servicio detenido, podemos lanzarlo y actualizar la página sin necesidad de cerrar el programa y volverlo a abrir. Cuando ya tengamos el mapa cargado, podemos movernos libremente con el ratón por el, pudiendo además hacer zoom sobre las zonas que deseemos para ver con más nivel de detalle la posición actual.

Cuando en nuestra aplicación hayamos realizado la búsqueda y configuración de los dispositivos, si alguno de ellos tiene en su configuración un sensor GPS, podremos ver la situación de los mismos para cada actualización de los movimientos. Además, debido a que durante el proceso de configuración hemos podido seleccionar el modelo con el que representaremos al dispositivo (Sección B.4.5), dicha representación aparecerá en la posición actualizada del mapa.

Center

Con esta opción, una vez que hemos comenzado la simulación y los dispositivos han comenzado a enviar datos capturados por sus sensores, si no conocemos la localización de los dispositivos o no los encontramos desplazándonos por el mapa, tenemos disponible esta opción que nos permitirá situarnos de forma rápida sobre la localización exacta de los mismos. Hemos de tener en cuenta que los dispositivos no tienen por qué estar siempre en el mismo lugar, por lo que, en caso de salirse algún dispositivo del encuadre del mapa, tendremos que desplazarnos manualmente para seguir su movimiento o bien habilitar la opción de movimiento automático.

Autocenter

Con la opción de auto-centrado, una vez que hemos comenzado la prueba y los dispositivos móviles aparecen en el mapa, podemos centrarnos sobre ellos tal y como hemos visto en la Sección B.4.6. Pero ya hemos comentado que tenemos el inconveniente de que, en caso de que un dispositivo desaparezca del mapa, hemos de desplazarnos a mano por el mapa. Si habilitamos la opción de auto-centrado, el mapa seguirá de forma automática a los dispositivos, manteniéndolos en todo momento dentro del encuadre del mapa. Si queremos dejar de seguir a los dispositivos, volviendo a pulsar sobre el botón detendremos el movimiento del mapa y, en caso de querer encuadrar el mapa sobre un dispositivo, tendremos que hacerlo de forma manual.

3D

El mapa cargado nos muestra por defecto la vista Satélite del mapa y, gracias a ello podemos ver con detalle el lugar donde se posicionan los dispositivos móviles. Dicha vista nos muestra desde un plano aéreo el terreno con fotografías reales tomadas desde una cierta altura (ver Figura B.34(a)), que estarán más o menos actualizadas, en función de la fecha en que hayan sido tomadas por la empresa o empresas responsables. Además, para dotar de mayor realismo y detalle al mapa, es posible encontrarnos que, los elementos que han sido fotografiados, han sido modelados y representados en tres dimensiones, dotando de mayor calidad a los detalles del mapa (ver Figura B.34(b)). Pero si el número de elementos modelados en 3D es mayor, es posible que podamos observar ralentizaciones ocasionales o que el mapa tarde en cargar más de lo normal, por lo que ofrecemos la oportunidad de habilitar o deshabilitar la opción de visualizar dichos elementos en tres dimensiones en el mapa.

Figura B.34: Vista aérea normal (a) y con edificios en 3D (b)

Exportar un fichero KML

Al acabar la prueba, es interesante el tener un fichero con todos los movimientos que se han ido reproduciendo en el mapa de la aplicación. Por ello podemos exportar

dichos movimientos a un fichero kml que podremos guardar en la ruta que prefiramos y de esta manera podremos abrir el fichero con un programa externo, como por ejemplo Google Earth.

B.4.7. Panel de cámaras

El panel de cámaras es uno de los paneles que no están activos por defecto al ejecutar la aplicación pero que puede habilitarse mediante la opción de menú *View -> Camera*. Una vez abierto, cada uno de los dispositivos que tengamos detectados en la aplicación y que haya sido configurado con una cámara, cuando sea seleccionado de la tabla de datos, mostrará en esta ventana la emisión de vídeo en directo. A continuación se explican las diferentes partes que componen este panel y sus funcionalidades (Figura B.35).

1. Barra de herramientas del panel de cámaras.

2. Panel de streaming de video.

Barra de herramientas del panel de cámaras

Contienen los siguientes botones de izquierda a derecha (Figura B.36):

-  /  - *3D Enable / Disable*: permite activar o desactivar la vista en 3D de los edificios existentes en el mapa.
-  /  - *Start / Stop video recording*: permite iniciar la captura de video.
-  - *Snapshot*: permite realizar una captura de un fotograma del video.
-  /  - *Maximize / Minimize*: maximiza o minimiza el panel de vídeo de tal forma que ocupe la totalidad de la ventana de la aplicación o la parte disponible por defecto.
-  - *Close*: cierra el panel de vídeo. Una vez cerrado es posible abrirlo nuevamente desde el menú *View*, seleccionando *Cameras* que, al estar cerrado, aparecerá desmarcado.

Figura B.35: Panel de cámaras

Figura B.36: Opciones de la barra de herramientas de la ventana de cámaras

Panel de streaming de vídeo

Para poder acceder al panel, previamente hemos de habilitarlo ya que, al ejecutar el programa, permanece oculto. Para ello tenemos que ir al menú *View* y hacer click sobre *Cameras*, que permanecerá desmarcado, para que nos aparezca éste panel.

La forma de obtener el vídeo de una de las cámaras de los dispositivos es bastante simple. Una vez hemos configurado las cámaras de los dispositivos y teniendo una prueba en ejecución, hemos de seleccionar el dispositivo de la tabla de datos del que queramos obtener su streaming de vídeo y automáticamente veremos en el panel cada una de las cámaras del dispositivo (en el caso de tener varias acopladas) en su pestaña correspondiente. Como podemos apreciar en la Figura B.35, además de la zona dedicada al vídeo, tenemos también disponibles los botones para controlar

remotamente los movimientos de la cámara. Si la cámara tiene configurados todos los comandos para controlar la cámara en cualquiera de las direcciones, todos los botones estarán disponibles; en caso contrario, aquellos botones para los que no tengamos correspondencia para controlar la cámara no aparecerán, como bien hemos visto al configurar la cámara en la Sección B.4.3. Pulsando sobre cualquiera de los botones conseguimos mandar el comando correspondiente a la cámara, la cual recibirá la orden y efectuará el movimiento del objetivo adecuado, percibiendo el cambio en la emisión del vídeo. Nos llama la atención el botón 

que tenemos disponible para cada una de las cámaras. Ya que estamos realizando una simulación en la que intervienen elementos reales y elementos simulados, podemos hacer también que la emisión del vídeo sea la que recibiríamos si estuviéramos situados sobre otro punto diferente al que nos encontráramos, haciendo uso que las características que Google Earth ofrece. Pulsando sobre el citado botón, podremos intercalar entre la vista real de la cámara y la vista que obtendríamos de estar físicamente en la ubicación real sobre la que hemos posicionado el dispositivo, pudiendo manipular igualmente el movimiento de la cámara en esta vista (ver Figura B.37).

Figura B.37: Vídeo simulado mediante Google Earth

B.4.8. Panel de gráficas

El panel de gráficas es uno de los paneles que no están activos por defecto al ejecutar la aplicación pero que puede habilitarse mediante la opción de menú *View -> Charts*. Una vez abierto, para cada uno de los dispositivos que tengamos detectados en la aplicación, los sensores con los que hayan sido configurados y que tengan disponibles la representación gráfica de los datos, para cada actualización de los datos enviados, se mantendrá un histórico de los mismos en forma de gráficas. A continuación se explican las diferentes partes que componen este panel y sus funcionalidades (Figura B.38)

1. Barra de herramientas del panel de gráficas.

2. Pestañas de gráficas.

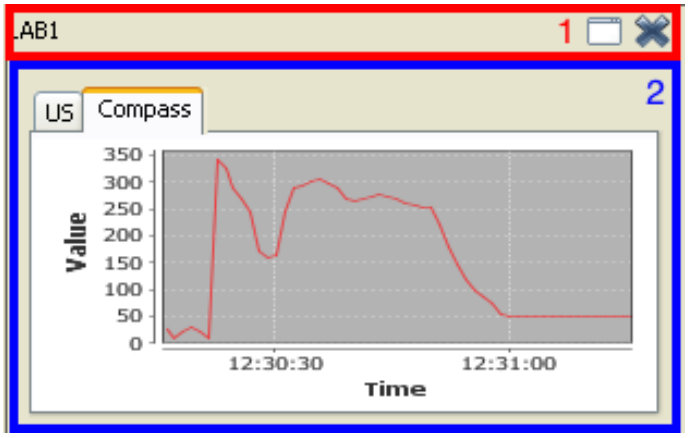


Figura B.38: Panel de gráficas

Barra de herramientas del panel de gráficas

Contienen los siguientes botones:

- / - *Maximize* / *Minimize*: maximiza o minimiza el panel de gráficas de tal forma que ocupe la totalidad de la ventana de la aplicación o la parte disponible por defecto.

- - *Close*: cierra el panel de gráficas. Una vez cerrado es posible abrirlo nuevamente desde el menú *View*, seleccionando *Chats* que, al estar cerrado, aparecerá desmarcado.

Pestañas de gráficas

Al igual que el panel de cámaras, este panel no está habilitado por defecto al ejecutar por primera vez el programa y, si queremos acceder a él, tendremos que ir nuevamente al menú *View*, y seleccionar la opción *Charts*, que también aparecerá desmarcada. Una vez lo hagamos, aparecerá el último de los paneles de la aplicación en el cual podremos ver un histórico de todos los datos capturados por los sensores de los dispositivos y que permitan su representación gráfica.

Para acceder a las gráficas correspondientes a un determinado sensor, procedemos de forma análoga a como hemos obtenido las imágenes de las cámaras. Simplemente hemos de seleccionar alguno de los dispositivos de la tabla de datos y automáticamente nos aparecerán todos los sensores disponibles en forma de pestañas en el panel. Seleccionando cualquiera de las pestañas, podremos ver cada una de las gráficas con todos los datos que han sido enviados por ese sensor desde el comienzo de la prueba hasta el momento actual.

Apéndice C

Ejemplos de uso

En este anexo se van a presentar distintos ejemplos de uso de la plataforma, detallando el funcionamiento de sus principales características, como son la carga de nuevos sensores o cámaras al sistema, la búsqueda y configuración de dispositivos y el inicio el acceso a los datos capturados por dichos dispositivos durante la ejecución de simulaciones.

C.1. Definición de un nuevo sensor.

Aunque es posible que tengamos en nuestra aplicación ya configurados los sensores disponibles, puede darse el caso que adquiramos un nuevo tipo de sensor para nuestros dispositivos, por lo que nuestro objetivo va a ser definir dicho sensor en la aplicación para que lo podamos configurar en los dispositivos. La Figura C.1 muestra un dispositivo NXT junto al nuevo sensor adquirido y que vamos a definir en la aplicación de escritorio.

El primer paso que seguiremos para ello es acceder a la opción de menú *File -> Settings*, donde nos aparecerá la ventana con la configuración del programa (Figura C.2).

Para introducir un nuevo sensor, pulsamos el botón  , apareciendo en pantalla la ventana para ello. Procedemos a introducir los datos que se nos pide. En nuestro caso, como se puede observar en la Figura C.3, ya hemos introducido los datos del sensor que vamos a cargar en el sistema:

- *Name*: establecemos el nombre como *Acceler*.
- *Manufacturer*: establecemos el fabricante como *Mindsensors*.
- *Parameters*: establecemos *3* como parámetros.

Figura C.1: Dispositivo NXT y el nuevo sensor adquirido

- *Labels*: establecemos *x*, *y*, y *z* como etiquetas.
- *Chart*: marcamos la casilla de gráfica y establecemos los valores máximo (*75*) y mínimo (*-75*).

Una vez que confirmemos los datos pulsando sobre el botón *OK*, el nuevo sensor será cargado y aparecerá en la tabla junto al resto de los sensores disponibles. Al introducir el sensor nos hemos percatado de que hemos marcado la casilla de gráficas pero, los datos que vamos a obtener de este sensor no es necesario que se representen gráficamente, por lo que procederemos a modificar el sensor que acabamos de introducir. Para ello, sobre la tabla de los sensores, seleccionamos la fila correspondiente al sensor introducido (que aparecerá en último lugar, como vemos en la Figura C.4) y presionamos el botón  , ver sensor.

Nos aparecerá de nuevo la ventana de antes, salvo que los datos del sensor elegido aparecerán en los campos correspondientes. Como bien hemos dicho, en la Figura C.3, la casilla *Chart* (Gráficas) aparece seleccionada, por lo que desmarcaremos la casilla y confirmaremos los cambios.

C.2. Definición de una nueva cámara

Además del modelo que ya teníamos disponible, hemos adquirido nuevas cámaras para nuestro sistema y vamos a tener que usarlas, pero previamente las tenemos que cargar en el sistema para que éste sepa como interactuar con ellas. La Figura C.5

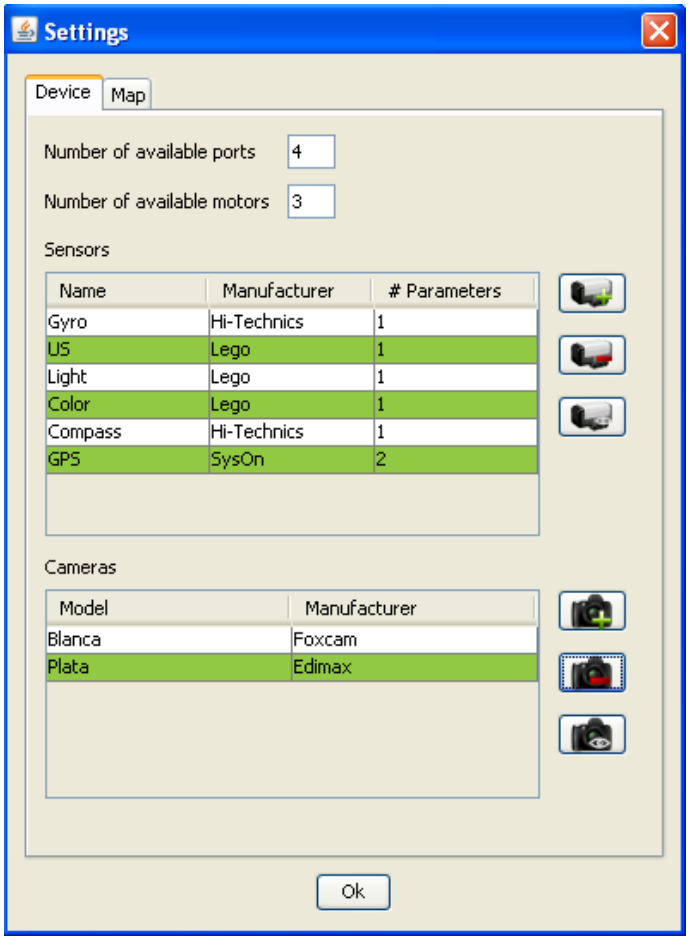


Figura C.2: Ventana *Settings*

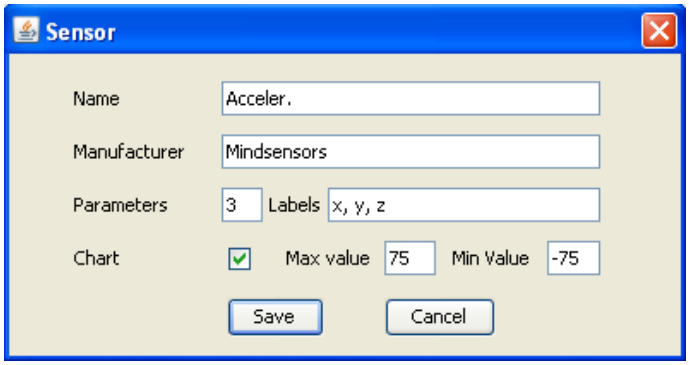


Figura C.3: Ventana *New sensor*

muestra un dispositivo NXT junto a la nueva cámara adquirida y que vamos a definir en la aplicación de escritorio.

Para ello realizamos los mismos pasos que hemos seguido para introducir un nuevo sensor y accedemos a la opción de menú *File -> Settings*, apareciéndonos la

Name	Manufacturer	# Parameters
Gyro	Hi-Technics	1
US	Lego	1
Light	Lego	1
Color	Lego	1
Compass	Hi-Technics	1
GPS	SysOn	2
Acceler.	Mindsensors	1

Figura C.4: Tabla de sensores

Figura C.5: Dispositivo NXT y la nueva cámara adquirida

misma ventana de antes con la configuración del programa (Figura C.2). Esta vez, como vamos a introducir una cámara, pulsamos sobre el botón  para cargarla en el sistema, apareciéndonos en pantalla la ventana destinada para ello. Procedemos a introducir los datos de acuerdo a las indicaciones del fabricante según los manuales proporcionados junto con la cámara. En la Figura C.6 podemos ver el resultado con todos los parámetros ya introducidos y listos para ser cargados:

- *Manufacturer*: establecemos el fabricante como *Foxcam*.
- *Model*: establecemos el modelo como *Negra*.
- *User*: establecemos el usuario como *admin*.
- *Password*: establecemos el password como *1234*.
- *Video URL*: establecemos la URL como *snapshot*.cgi.
- *Command URL Type*: establecemos el tipo de URL, en nuestro caso, se compone de una única dirección, por lo que seleccionamos *Inline*.

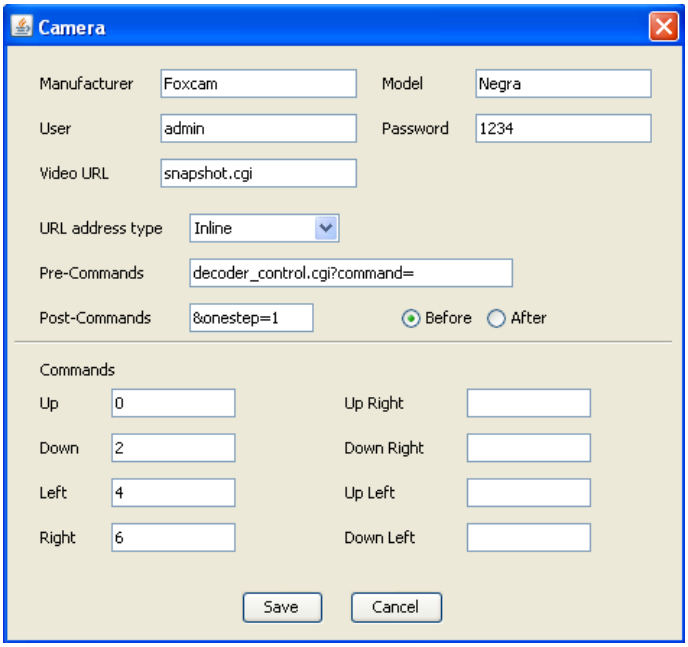


Figura C.6: Ventana *New camera*

- *Command URL Pre Mov*: establecemos la primera parte de la URL como *decoder_control.cgi?command=*.
- *Command URL Post Mov*: establecemos la última parte de la URL como *&onestep=1*, indicando que el comando va antes de esta parte (*Before*).
- *Comandos Up / Down / Left / Right*: establecemos los comandos de los movimientos arriba / abajo / izquierda / derecha como *0*, *2*, *4*, y *6* respectivamente.

Una vez que aceptemos, la nueva cámara será cargada y aparecerá junto al resto de las cámaras disponibles, tal y como podemos ver en la Figura C.7. De ser necesario, podríamos modificar esta cámara si hemos introducido algún dato de forma errónea de forma similar a como hemos procedido en el apartado anterior con el sensor. Seleccionaríamos en este caso la cámara deseada y pulsaríamos el botón  para modificar los datos.

Model	Manufacturer
Blanca	Foxcam
Plata	Edimax
Negra	Foxcam

Figura C.7: Tabla de cámaras

C.3. Búsqueda de dispositivos

Para la prueba que vamos a realizar, disponemos de una pareja de robots que ya han sido pareados previamente con el ordenador y que ya están encendidos. Como paso previo a comenzar el test, hemos de tener la aplicación cargados en los robots para ejecutarla una vez los dispositivos hayan sido detectados. Antes de comenzar la búsqueda, confirmaremos que los robots están todos encendidos y no tienen ningún programa en ejecución, es decir, están todos los dispositivos en el menú general de leJOS.

Tras esta comprobación previa, vamos a pulsar el botón  para realizar una búsqueda de los dispositivos (también podríamos hacerla mediante la opción de menú *Search -> Search devices*). Nos aparecerá la ventana en la que veremos los dispositivos que se encuentran dentro del alcance de nuestro Bluetooth. Nos aseguramos de que nuestro dispositivo Bluetooth está conectado y pulsamos sobre el botón *Search* para comenzar la búsqueda. Para conocer en todo momento el estado de la búsqueda nos aparecer una barra que nos va indicando del estado de la misma (ver Figura C.8), pudiendo pararla en caso de haber detectado algún problema en los dispositivos (por ejemplo, que estén apagados). Una vez que hemos detectado todos los dispositivos visibles, éstos aparecerán en la tabla y simplemente tendremos que seleccionar aquellos que vayamos a usar (ver Figura C.9).

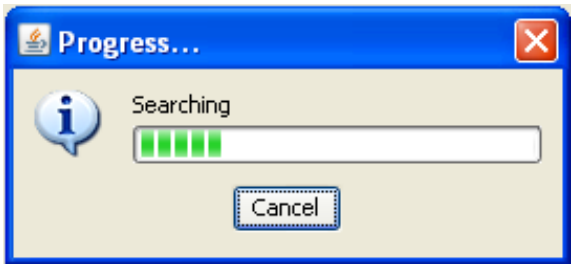


Figura C.8: Búsqueda de dispositivos

Device	BT Address
LAB1	0016530D8DC9
LAB2	00165312B446
LAB3	0016531122DD
LAB4	00165312B961
HOME	001653009996

Figura C.9: Tabla de dispositivos encontrados

Cuando ya hemos marcado los dispositivos, pulsamos el botón *OK* para volver a la ventana principal, en la que, como podremos ver en el panel de datos, habrán aparecido los dispositivos seleccionados (ver Figura C.10).

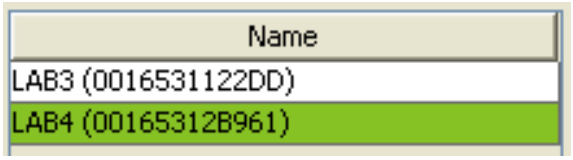


Figura C.10: Tabla de datos con los dispositivos seleccionados

C.4. Configuración de los NXT

Una vez seleccionados los dispositivos, vamos a configurar cada uno de ellos de la forma que más nos convenga de cara a las pruebas. Por ellos primero configuraremos los sensores de los dispositivos y posteriormente los movimientos que realizarán. Si intentamos comenzar en este punto una simulación, veremos que no es posible ya que previamente hemos de completar este paso para todos los dispositivos seleccionados en el paso anterior.

C.4.1. Sensores

Para configurar los sensores que irán en los dispositivos, primero seleccionaremos uno de los disponibles en la tabla de datos y pulsamos el botón  para configurar los sensores. Nos aparecerá la ventana desde la que podremos realizar todo el proceso. En ambos dispositivos vamos a tener una configuración bastante similar en cuanto a los sensores.

Primero pulsamos tantas veces como sensores queramos sobre el botón  e irán apareciendo los sensores en el panel de la izquierda, además de la representación gráfica simplificada de los mismos sobre la imagen. Posicionamos dicha representación en los lugares aproximados donde irán los sensores y rellenamos los campos de cada sensor. Al ser todos ellos sensores cableados, hemos de tener en cuenta que no podremos tener duplicado el puerto al que va a ir conectado. Seleccionamos el sensor de la lista e indicamos también cada cuantos segundos queremos que envíen la información. Como el dispositivo va a realizar movimientos, hemos de añadir también un par de motores al dispositivo pulsando el botón  . Indicamos los puertos a los que están conectados. Ya que hemos configurado antes una cámara nueva, podemos añadirla a este dispositivo; para ello pulsamos el botón  y aparecerá una nueva pestaña para configurar la cámara junto a la representación simbólica de la misma. Movemos dicha representación hasta la posición que queramos y en la pestaña seleccionamos el modelo de cámara e indicamos la dirección *IP* que tiene asignada la cámara. Antes de confirmar la configuración vamos a proceder a guardarla en un fichero externo, por lo que pulsaremos sobre el botón

 y, si todos los datos introducidos son correctos, el sistema nos avisará de que el fichero se ha creado con éxito. Ya podemos confirmar la configuración de este dispositivo. Como observaremos, en la tabla donde antes teníamos únicamente la columna con el nombre de los dispositivos, ahora han aparecido nuevas columnas con los nombre de los sensores que hemos seleccionado (Figura C.11). En el segundo dispositivo las celdas aparecen en otro color ya que todavía no tiene ningún sensor configurado.

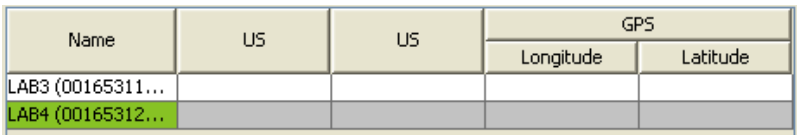


Figura C.11: Tabla de datos con los sensores de un robot configurados

A continuación procedemos a configurar el segundo de los dispositivos, por lo que lo seleccionamos de la tabla y volvemos a pulsar el botón  . Como hemos indicado, los dispositivos van a tener una configuración bastante similar, por lo que esta vez el proceso será bastante más rápido. Nos vuelve a aparecer la ventana de antes, pero ahora pulsaremos sobre el botón  para cargar una configuración previa. Seleccionamos el archivo guardado de antes (Figura C.12) y aceptamos.

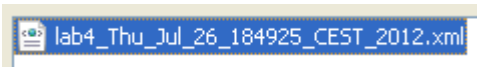


Figura C.12: Selección de un fichero de configuración existente

Automáticamente se cargarán los mismos datos que contiene el otro dispositivo, pero podemos realizar modificaciones sobre ellos. En este caso, como solo tenemos disponible una cámara, la eliminaremos pulsando el botón de cerrar, con forma de 'x', de la pestaña correspondiente. Aprovechando que hemos incluido un nuevo sensor y que tenemos todavía libre un puerto en el dispositivo, pulsamos nuevamente el botón  para cargar el sensor adicional en el puerto libre. Tras ello, confirmamos la configuración del dispositivo. Observamos el estado final de la tabla y vemos que, al ser una configuración bastante idéntica, las columnas de los sensores están disponibles para ambos dispositivos con la salvedad de este último sensor, que solo lo tiene disponible el segundo dispositivo (ver Figura C.13).

Todavía no podemos lanzar la aplicación ya que tenemos que configurar los movimientos de los dispositivos.

Name	US	US	GPS		Acceler.		
			Longitude	Latitude	x	y	z
LAB3 (0016...							
LAB4 (0016...							

Figura C.13: Tabla de datos, sensores configurados

C.4.2. Movimientos

Ahora vamos a configurar los movimientos para cada uno de los dispositivos. Para este caso, ya que únicamente queremos que se muevan por una zona, no es necesario que le configuremos unos movimientos demasiado complejos. Seleccionamos el primer dispositivo y pulsamos sobre el botón  , apareciendo la ventana destinada a configurar los movimientos. Para los movimientos que los dispositivos han de realizar, teneos diferentes alternativas, pudiendo controlar manualmente a los dispositivos, hacer que sigan una trayectoria definida mediante pantalla o bien pintada sobre la superficie donde serán colocados o, que permanezcan estáticos sin realizar movimiento alguno (ver Figura C.14).

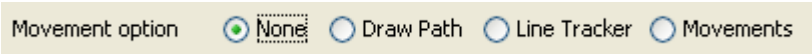


Figura C.14: Opciones de movimiento

En el caso de este dispositivo, vamos a seleccionar como opción de movimiento la definición de trayectorias, por lo que marcamos la opción *Draw Path* y aparecerá el panel sobre el que podremos dibujar la trayectoria (ver Figura C.15).

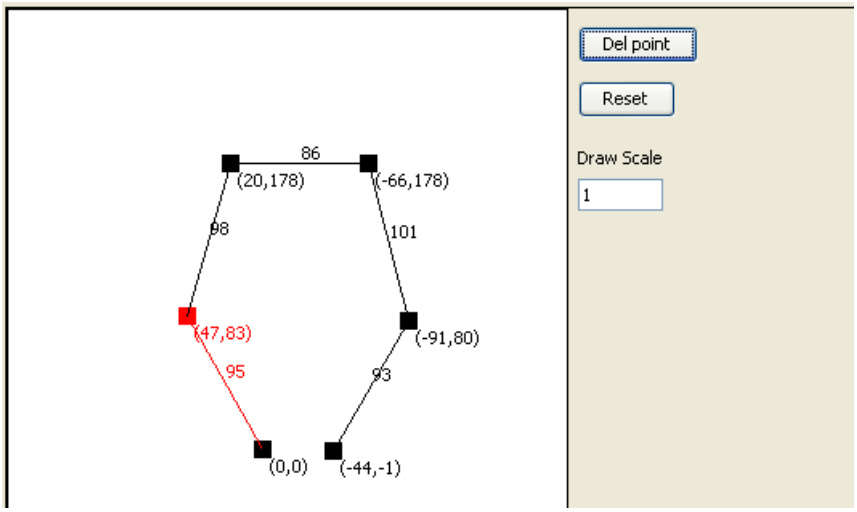
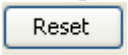


Figura C.15: Definición de trayectorias

El manejo es bastante simple, pulsamos sobre el panel blanco para añadir un

punto y éste aparecerá en el lugar marcado. Si pulsamos nuevamente en otro lugar, aparecerá un nuevo punto (en color rojo para indicar que es el último punto y por lo tanto el activo) unido al anterior junto con la distancia entre ambos y así sucesivamente. Si queremos modificar la posición de alguno de los puntos ya existentes, seleccionamos dicho punto y sin soltar el botón izquierdo del ratón, lo desplazamos hasta el lugar que queramos. si tenemos algún punto de más en el sistema y queremos eliminarlo, simplemente hemos de seleccionarlo (pasará a ser el punto activo, de color rojo) y pulsar sobre el botón 

. Al eliminarlo, el resto de los puntos permanecen igual con la salvedad de las distancias del punto eliminado con los adyacentes, que se recalcará. Si quisiéramos, podríamos eliminar la totalidad de los puntos dibujados presionando el botón 

, por lo que tendríamos que volver a dibujar la trayectoria. La escala por defecto para las distancias está multiplicada por 1 y tiene una equivalencia directa en centímetros (un valor de 50 en la trayectoria equivale a 50 cm), pero podríamos modificar dicha relación en caso de tener que abarcar distancias mayores. Si en el campo *Draw Scale* cambiamos el valor 1 que tenemos por defecto, por un valor 2 o cualquier otro entero positivo, observaremos como las distancias cambian automáticamente. Cuando tengamos la trayectoria deseada para que sea representada por el dispositivo, confirmamos con el botón de *OK* y procederemos a configurar el segundo dispositivo.

Antes de confirmar la trayectoria, podríamos exportarla a un fichero en el caso de querer usarla para posteriores dispositivos o pruebas. Para este ejemplo de uso, hemos decidido configurar los dispositivos de varias formas posibles, por lo que no exportaremos la trayectoria a un fichero.

Para el segundo dispositivo vamos a optar por un control manual del mismo mediante combinaciones de teclado. Para ello seleccionaremos la opción *Movements* en la ventana de configuración de movimientos (ver Figura C.14). Los movimientos del dispositivo son bastante simples, reduciéndose a 4 ordenes básicas: *adelante*, *atrás*, *izquierda*, y *derecha*, por lo que, para cada uno de los movimientos, tendremos que asignar una letra del teclado. Una configuración típica de teclas puede ser '*A*', '*W*', '*S*', y '*D*'. Confirmaremos la selección de teclas para volver a la ventana principal y poder comenzar la prueba.

C.5. Configuración remota y ejecución de las pruebas

Como ya tenemos configurados totalmente ambos dispositivos, llegados a este punto ya podemos comenzar la prueba, pero hemos de cargar la configuración para que los robots sepan los movimientos que tienen que realizar, los datos que tienen que capturar y enviar o las ordenes que tienen que recibir. Antes de nada, nos aseguramos de que todos los sensores que hemos indicado están conectados donde tienen que

estar y que ambos dispositivos están encendidos y con el programa en ejecución, apareciendo el mensaje de "*Waiting...*" en la pantalla del robot, además de estar los robots situados en una zona despejada y amplia, con el espacio suficiente para realizar sin ningún obstáculo las trayectorias indicadas. Pulsamos sobre el botón  para comenzar la carga de las configuraciones a los dispositivos. El proceso de carga de las distintas configuraciones a los dispositivos aparece reflejado en la esquina inferior izquierda del programa (ver Figura C.16), pasando a estar en blanco la pantalla del dispositivo hasta aparecer un mensaje con los datos cargados.

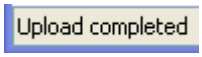


Figura C.16: Mensaje de subida de configuraciones

Cuando se hayan completado todas las subidas, para comenzar las pruebas simplemente tendremos que pulsar el botón  y los robots ejecutarán sus instrucciones. Ambos dispositivos enviarán, con el intervalo de tiempo indicado, los datos que recojan de sus sensores y se mostrarán en la celda correspondiente de la tabla. El dispositivo que ha sido configurado con una trayectoria preestablecida comenzará además a realizar los movimientos, mientras que el otro permanecerá en la misma posición hasta que pulsemos las teclas correspondientes. Para controlar remotamente el segundo dispositivo, observamos que nos ha aparecido una segunda ventana en la aplicación (ver Figura C.17), desde la que además podremos controlar también la velocidad de desplazamiento del propio dispositivo.

Figura C.17: Ventana de captura de teclas

Dicha ventana es la encargada de recoger las pulsaciones de teclado y, en caso de pulsar la tecla correspondiente a algún movimiento, capturarla para mandar la orden de desplazamiento adecuada al robot correcto. Si dicha ventana no está activa, las pulsaciones de teclado no podrán ser capturadas y por tanto no podremos mover el o los dispositivos. Esta ventana permite además controlar la velocidad de desplazamiento del dispositivo que controlamos por teclado, pudiendo disminuirla

o aumentarla en función de nuestras necesidades mediante el desplazamiento de la barra horizontal, viendo en todo momento la velocidad actual del dispositivo.

Podemos detener la prueba en cualquier momento pulsando el botón  . Esto hará que los dispositivos se detengan y dejen de enviar datos al programa. Para reanudar cualquier prueba, simplemente hemos de volver a ejecutar este último apartado, no teniendo que volver a configurar ni los sensores ni las trayectorias de los dispositivos a no ser que decidiéramos efectuar algún cambio, por lo que podríamos hacer las modificaciones oportunas según hemos visto a lo largo de este apéndice.

C.6. Visualización de los datos

Mientras tenemos en ejecución una prueba, hemos comentado que los dispositivos envían la información capturada por sus sensores hasta el programa y que ahí se muestra en la tabla destinada para ello. Pero además tenemos otros elementos mediante los cuales podremos ver la totalidad de los datos recibidos por los robots.

Para acceder a estos datos, hemos de desplegar los paneles que nos permitirán ver en detalle estos datos. Para acceder al histórico de todos los datos recibidos por los sensores, podemos ir al menú *View -> Charts* y nos aparecerá un nuevo panel en la ventana principal (ver Figura C.18). en este nuevo panel tenemos disponible de forma gráfica todos los datos enviados desde los dispositivos desde el comienzo de la prueba, siempre y cuando hayamos configurado en el sensor la opción de que sea representado de esta manera (tal y como hemos visto en la Sección C.1).

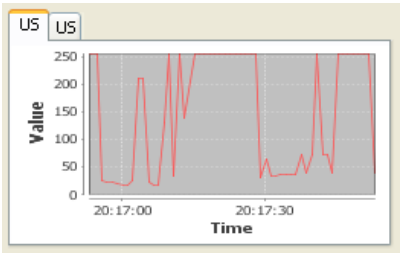


Figura C.18: Panel con las gráficas de los sensores

Para obtener las gráficas de todos los sensores de un determinado dispositivo, simplemente hemos de seleccionar de la tabla aquel en el que estemos interesados y aparecerán en este panel todas las gráficas de los sensores que tenga configurados en sus respectivas pestañas, que podremos visualizar fácilmente con el ratón. Si mantenemos abierto este panel mientras tenemos en ejecución alguna prueba, observaremos que, además de actualizarse las tablas con los últimos datos recibidos, las gráficas también se actualizan en tiempo real con el último dato recibido desde el sensor. Si queremos ver con más detalle la gráfica, tenemos la posibilidad de ampliar

dicho panel para que ocupe la totalidad de la aplicación, pudiendo regresar luego al estado original del panel.

El otro tipo de información que nos puede interesar ver en tiempo real es el streaming de video emitido desde las cámaras situadas en los robots y del que ya hemos hablado con anterioridad. Para acceder a este panel, también tenemos que acceder al menú *View* pero ahora seleccionaremos la opción *Cameras*, por el cual se nos abrirá el ultimo de los paneles que componen la aplicación. La forma de obtener el vídeo de una de las cámaras de los dispositivos es bastante similar a como obtenemos sus gráficas, simplemente hemos de seleccionar el dispositivo de la tabla y automáticamente veremos en el panel cada una de las cámaras (en el caso de tener varias acopladas) en su pestaña correspondiente. Como podemos apreciar, además de la zona dedicada al vídeo, tenemos también disponibles los botones para controlar remotamente los movimientos de la cámara. Si la cámara tiene configurados todos los comandos para controlar la cámara en cualquiera de las direcciones, todos los botones estarán disponibles; en caso contrario, aquellos botones para los que no tengamos equivalencia para controlar la cámara no aparecerán, como bien hemos visto al configurar la cámara en la Sección C.2. Pulsando sobre cualquiera de los botones conseguimos mandar el comando correspondiente a la cámara, la cual recibirá la orden y efectuará el movimiento del objetivo adecuado, percibiendo el cambio en la emisión del vídeo. Nos llama la atención el botón 

que tenemos disponible para cada una de las cámaras. Ya que estamos realizando una simulación en la que intervienen elementos reales y elementos simulados, podemos hacer también que la emisión del vídeo sea la que recibiríamos si estuviéramos situados sobre otro punto diferente al que nos encontráramos, haciendo uso que las características de Google Earth para el manejo de las vistas y las posibilidades que ofrece en el modelado de objetos representativos del mapa en ciertos lugares, como puede ser el caso de la ciudad de San Sebastián.

El último lugar de representación de los datos recibidos ya lo hemos ido viendo a lo largo del presente ejemplo de uso. Se trata de la tabla de datos que ya hemos comentado con anterioridad, en la que tenemos disponibles los dispositivos seleccionados junto con los sensores que tienen configurados. Para cada una de las actualizaciones, los datos que envíe cada uno de los dispositivos hasta la aplicación, serán mostrados en la celda de la tabla correspondiente, por lo que, de un simple vistazo, sabremos que sensores están enviando datos y cual es el último valor del mismo, tal y como podemos ver en la Figura C.19, con los últimos datos recibidos por todos los sensores que hemos configurado previamente.

Name	US	Compass	US	GPS		Gyro	Acceler.		
				Longitude	Latitude		x	y	z
LAB3 (00165...		327.0		-1.001695...	41.709339...	25.0	-512.0	-456.0	-289.0
LAB4(001653...	170.0	259.0		-1.001681...	41.709358...	0.0			

Figura C.19: Tabla de datos con información de los sensores

Apéndice D

Artículo aceptado en JISBD'12

A continuación se adjunta el artículo presentado y aceptado en las XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD) [28] enmarcadas dentro de las Jornadas SISTEDES 2012 que se celebrarán este año en la Universidad de Almería durante los días 17, 18 y 19 de Septiembre de 2012, organizado por el Grupo de Investigación de Informática Aplicada (TIC-211).

Using Small Affordable Robots for Hybrid Simulation of Wireless Data Access Systems

Gorka Guerrero, Roberto Yus, and Eduardo Mena

IIS Department, University of Zaragoza
María de Luna 1, 50018, Zaragoza, Spain
{524080,ryus,emena}@unizar.es

Abstract. During the last years, research on data processing in wireless environments has increased due to the emergence of mobile devices that are able to obtain real environmental sensory information (e.g., smartphones). Testing the different approaches in a real environment is not always possible due to high costs of deployment of hardware and users in many cases. However, the real world complexity can be simplified according to our needs as information systems always deal with simplified abstractions of real objects. For example, a system considering the location of a real car could simplify it as a certain entity with the same movement path. This can be achieved by using software simulations, which obtain approximated results reducing the costs. Nevertheless, it is difficult to develop an accurate real-world model to simulate the environmental conditions (e.g. uneven tracks, dynamic wireless network coverage, etc.).
We introduce in this paper a hybrid simulation platform that is able to recreate real-world scenarios more accurately than software simulations. For that, it uses small affordable robots equipped with sensors in controlled real environments as counterparts of real moving objects and the scenario where they are involved. This enables testing the system considering real communication delays, real sensor readings, etc., instead of having to simulate such events. Finally, we present the experimental evaluation of the system using LEGO Mindstorms robots to simulate a real rowing race at the San Sebastian bay.

Keywords: Hybrid Simulation, Wireless Data Access, LEGO Mindstorms

1 Introduction

Nowadays there exist more and more mobile devices, such as smartphones, tablets, or laptops with different types of sensors integrated. Because of their small size, they can be attached to moving objects (e.g. cars, people, etc.) and thanks to their communication mechanisms (Wi-Fi, Bluetooth, 3G, etc.), they are suitable for remote sensing. Thus, research in fields such as mobile computing or data mining are considering these devices to develop systems to manage sensory information obtained using wireless communications.

To validate these systems researchers must perform tests, but performing real tests is not always possible. For example, testing a scenario where there exist a great number of cars and people moving around a city have high costs of deployment. However, from the point of view of a data access system the complexity of the real world can be simplified according to the information it wants to obtain. In the previous example, if the data access system is interested in the location of the cars and people, they can be considered as simpler entities with the same movement path. For this reason, researchers use software simulation to test their approaches that dramatically reduce the testing costs. The main problem with software simulations is that the fidelity of their results has been a concern. Obtaining a real-world model to simulate environmental conditions as wireless communication delays or disconnections, sensor failures, etc., is a real challenge.

We propose in this paper a system to carry out hybrid simulations of data access systems, which is able to recreate some real-world scenarios more accurately than software simulations. Real objects, which are abstracted in software simulations, are simplified in our system by using real moving objects with similar characteristics. For this, we use small affordable robots equipped with real sensors in controlled real environments. Using these robots we are able to test the system considering real aspects that are difficult to simulate, such as communication delays, unexpected events, and sensor accuracy. For example, considering a system that analyzes in real-time the sensory data acquired in a sport event where involved moving objects are equipped with location sensors, video cameras, and wireless communications, generate a real-world model to simulate the environmental conditions is a challenging task. However, using our system some elements will be simulated (e.g., the scale of the scenario) while the real hardware employed leads to obtain real environmental conditions.

The rest of the paper is organized as follows. In Section 2 we present the motivation and technological context behind our system. In Section 3 we present the system architecture. In Section 4 and Section 5 we present how moving objects and the base station are modeled in our system, respectively. In Section 6 we present a prototype developed to test our system. In Section 7 we review some interesting related works. Finally, in Section 8 we present our conclusions and future work.

2 Context

In this section we explain a sample use case to motivate the development of a hybrid simulator. Then, we will explain the technological context of our system.

2.1 Motivation

The use case we are considering as example of wireless data access system [13] deals with the acquisition of sensory information (i.e. location, camera video,

etc.) of a sport event, the rowing race of San Sebastian (Spain). With this information, the system is able to help the technical director (in charge of the live TV broadcasting of the race) to select the best camera views.

A software simulator was used to test this system fed with the GPS locations of all the rowing boats obtained every second during a race. However, one of the problems of this test was that, even then the GPS locations were real, other sensor information (for example, the camera video streams) was simulated. Moreover, to test a different rowing boat configuration (for example, changing their movement path) new GPS traces are needed and obtaining them is difficult as the race takes place once a year.

In Table 1 we compare three different approaches to test a data access system (real test, software simulation, and our system), according to different criteria: a) realistic communication delay, how accurate is the behavior of the communications?, b) sensors used, how realistic are the sensors considered? , c) scenario, how much space is required to test the scenario?, d) repeatability, is easy to repeat the test?, e) interaction with the environment, how accurate are the environmental conditions?, f) time lapse, how much time is required to perform the test?, g) cost, how high is the cost to perform the test?.

Dealing with	**Real test**	**Simulation**	**Our system**
Realistic communication delay	✓✓	✗	✓
Sensors used	✓✓	✗	✓
Scenario	✗	✓✓	✓
Repeatability	✗	✓	✓
Interaction with the environment	✓	✗✗	✓
Time lapse	✗	✓✓	✓
Cost	✗	✓✓	✓

Table 1. Comparing the different approaches to test a data access system: using real tests, software simulations, and our system.

Compared with a software simulation, our hybrid simulation provides real communication delays (as it uses real communication mechanisms), real sensors (that obtain real data), and real environmental conditions. However, the duration of the test in our system is higher than in a software simulation (for example, because of the maximum sample acquisition rate of the sensors), and the cost of deployment could be also slightly higher. Compared with a real test, our hybrid simulation requires less space and infrastructure for the scenario (as it can be scaled), it is easier to repeat the tests (for example to obtain the results with new restrictions), the duration of the test could be reduced, and the cost is much smaller.

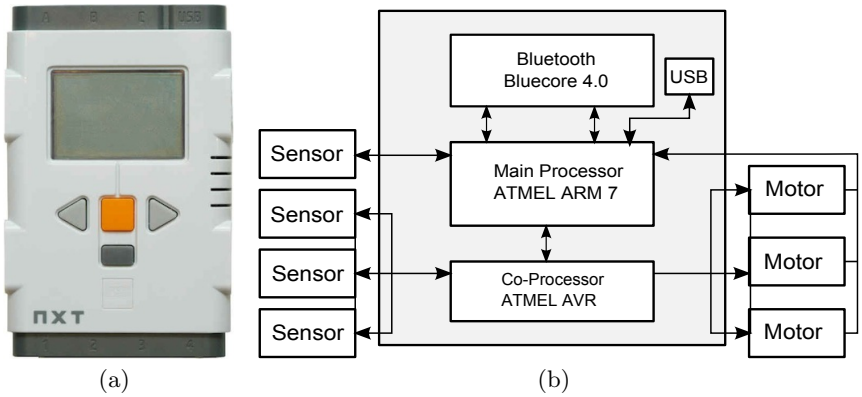
Therefore, our system allows to perform hybrid simulations of different scenarios where moving objects equipped with sensors capture and store data. To configure a specific scenario, the user selects its location and scale as well as the

amount of moving objects involved. For each object in the scenario, we have to specify its sensors and the path or movements that it is going to follow.

2.2 Technological Context

We have decided to use LEGO Mindstorms [2] robots in the prototype of our proposed system to represent real moving objects (see Section 6). LEGO Mindstorms is a basic but powerful, flexible, and affordable robot kit that was initially designed as an advanced toy, but is widely used to develop valuable designs [6] and in academic environments [4, 7, 9].

Architecture: As we can see in Figure 1, the NXT brick has a main processor and a co-processor. First, an Atmel 32-bits ARM processor is mainly used to control communication with Ultrasonic sensors, and USB/Bluetooth communication mechanisms. An Atmel 8-bits AVR processor controls the rest of sensors and servo motors connected to the NXT. Before starting to code programs for the NXT, we have to consider the limited capabilities of the robot's hardware, since we have only 256KB of FLASH memory and 64KB of RAM inside the 32-bit ARM processor, so the program has to be optimized to run on it. Each NXT can have attached up to 4 sensors and 3 motors to the available ports. In addition, the NXT is equipped with USB and Bluetooth 4.0 communications.



(a) (b)

Fig. 1. The NXT brick (a) and its main components (b).

Sensors: The Most popular LEGO Mindstorms sensors are the following:

- *Ultrasonic*, that returns the distance to an obstacle.
- *Touch*, that detects collisions.
- *Sound*, that measures sound levels.

– *Light/Color*, that detects different levels of grey or color scale.

A complete description of all sensors with their characteristics, schema and operations is available in the *LEGO MINDSTORMS NXT Hardware Developer Kit*[1]. However, the LEGO NXT brick is an Open Source hardware system and so we can easily find third-party sensors. For example, Hi-Technics[2], Mindsensors[3], and Dexter Industries[4] are some of the companies that distribute third-party sensors such as:

– *Compass*, that returns the direction.
– *3-Axis Accelerometer*, that measures the acceleration in three axis.
– *Gyro*, that measures the number of degrees per second of rotation.
– *Magnetic*, that can detect magnetic fields.

Figure 2 shows some of these sensors. All of them use the same interface to be connected to the NXT, a 6-position modular connector that features both analog and digital interfaces. However, the NXT brick is not limited to these sensors, for example, we can attach an external GPS or a camera via Bluetooth [11].

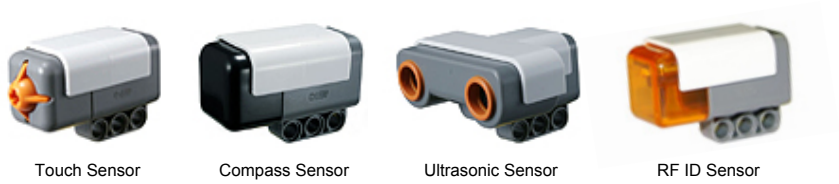


Fig. 2. Some of the available sensors for the NXT.

Software: NXT-G is the standard programming software for NXT and is included in the kit. NXT-G is based on LabVIEW graphical programming software, created by National Instruments and released in December 2006. Nevertheless, LEGO has released the firmware of the NXT as Open Source, so we can install other firmware on the NXT and create more complex programs for the robot using other popular programming languages. For example, RobotC [10], which is based on C, is distributed as Proprietary Commercial Software, and leJOS NXJ [1], a high-level Open Source programming language based on Java, that is the firmware we have chosen for NXTs in our prototype.

[1] `http://mindstorms.lego.com/eng/Overview/nxtreme.aspx`
[2] `http://www.hitechnic.com/`
[3] `http://mindsensors.com/`
[4] `http://dexterindustries.com/`

3 System Architecture

Our system consists of two main parts shown in Figure 3: 1) the selected moving object responsible for collecting data from all its attached sensors, and 2) the base station which is in charge of receiving data from all the moving objects and controlling them. This architecture allows the system to connect different moving objects, that are used to represent real objects (e.g., cars, rowing boats, people, etc.). Also the base station allows the user to access the data of the sensors in real-time and control the robots.

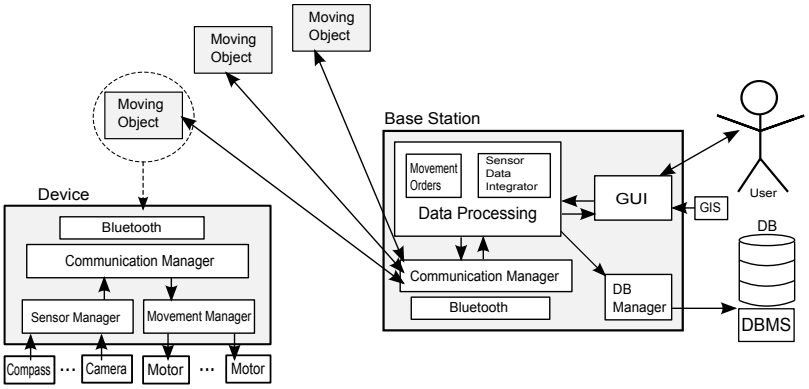


Fig. 3. System Architecture.

Therefore, the equivalence between performing tests in a real scenario and using our hybrid simulation system can be summarized as follows. We use LEGO Mindstorms robots as simplified abstractions of the real moving objects in a real scenario, as they can be controlled to perform similar movements. Instead of using certain real sensors, robots are equipped with (real) LEGO sensors that can perform nearly the same tasks and acquire the same type of information than we want to test. Finally, sometimes we do not need to test the system in the same scenario; instead we can deploy a simplified version of the real scenario (for example, at a smaller scale).

4 Simulating Moving Objects

The moving objects that our system manages are a simplified version of real objects to simulate, and the system controls them during the hybrid simulation to behave as the real ones. Thanks to this, the system is able to capture real data from the environment as real moving objects could do. To perform these tasks, our moving object has three main modules that we explain in the following.

4.1 Sensor and Movement Managers

As we have explained in Section 2.2, we can manage different types of sensors for our selected moving objects, so we need a module that can handle them and that allow us to add new sensors easily. Each sensor has different control options, different access functions, and provides different data, so the module has to deal with this heterogeneity.

We have analyzed several of the available sensors (e.g. sound, light, compass, accelerometer, etc.) to be integrated in our moving object, their purpose and the data they generate. We have noticed that, among the data provided by the sensors, we can get integer or float numbers (depending if we are measuring the distance to an obstacle or the angular velocity of a turn taken by the moving object). Also, we can receive different number of parameters from sensors too. For example GPS returns latitude and longitude data, an accelerometer returns the acceleration in the X, Y, and Z axis, and a compass only returns the direction.

The movement manager module receives orders from the base station and decodes them to send movement orders to the proper motor. Each of the motors are independent and can perform two different movements: backwards and frontwards. By combining these movements and by using different motors, the movement manager module is able to recreate the behavior of real-world moving objects.

4.2 Communication Manager

Once we have the data from the Sensor Manager, the Communication Manager module is in charge of sending it to the base station. We have developed a communication protocol to send all this different information from the sensors to the base station.

The protocol consists basically on a message generated by using the code of the NXT, the code of the sensor, and the data we want to send. The quantity of data depends on the sensor, so we have to separate each parameter while making the message easy to be parsed on the base station side (Table 2 shows some sample sensor messages of the communication protocol).

Compass Sensor	NXT code	Sensor Code	Direction		
GPS Sensor	NXT code	Sensor Code	Latitude	Longitude	
Accelerometer Sensor	NXT code	Sensor Code	X-Axis	Y-Axis	Z-Axis

Table 2. Sample of communication messages used by the system.

Once the message has been generated, the communication manager has to send it to the base station. The communications this module manages are built over a Bluetooth layer (Bluetooth is the default communication mechanism in

NXT brick and leJOS includes Bluetooth libraries in its API), but we could use other communication mechanisms as Wi-Fi or ZigBee [5].

Due to Bluetooth limitations, the maximum number of active slaved devices paired to the master device (base station) in a piconet is seven. This number may be higher if the implementation handles parked connections. If we have inactive or parked devices, we can connect up to 255 devices, which the master device can bring into active status at any time.

5 Base Station Side Application

The other main component of our system is the application on the base station side. The purpose of this program is to collect all data sent by moving objects connected to it and display this information in the Graphical User Interface (GUI). Also, it enables the user to control the moving objects. For all these tasks, the program uses different modules, explained in the following.

- *Graphical User Interface*
  The GUI is an important part of the hybrid simulator because it needs to ease the user interaction. All data that the moving objects capture and send to the base station have to be shown in the GUI according to their types. For example, for some data types such as the location, it is necessary not only to show the last data received, but also a history with all samples received from the beginning. Moreover, the GUI also has to be capable to show to the user in real time all the video streaming of the cameras. Therefore, the GUI module uses tables, graphs, and external Geographical Information Systems (GIS) to display the information easy and effectively.
- *Communication Manager*
  This module is very similar to the Communication Manager module explained in Section 4.2. It is in charge of receiving all the messages from the moving objects and also of sending them control messages. This module is built over a Bluetooth layer too, but this time the layer depends on the Operating System installed. In our prototype we have used Bluecove[5] as Bluetooth Java library that interfaces with Microsoft Bluetooth stack in Windows (XP and Vista) and also with Mac OS X.
- *Data Processing*
  This module processes data received from the Communication Manager. First, all data is parsed and processed to be treated and displayed according to the source. As in the communication protocol shown in Table 2, the message received contains the code of the moving object and the sensor, so when we decode it, we have to send the information to the GUI. Also, this information will be stored in a database to keep all the information received to be analyzed later if it is necessary.
  This module also manages all the robot movement orders they have to perform. From the GUI, we select the desired order and this module generates

[5] `http://bluecove.org/`

the message that the Communication Manager module will send to the moving object. In addition, this module controls also all the camera movements so, we can control remotely what each camera is viewing.

– *Database Manager*
Although all the information received from the moving objects is shown in the GUI, it is also interesting to store it to be analyzed later. This could be interesting because occasional anomalies can occur during the test (for example, a sensor could obtain a wrong measurement) and it is difficult to realize on them in real-time. So, storing the data in a database allow us to analyze this information.

6 Experimental Evaluation

We have developed a prototype of the system proposed to perform the hybrid simulation of the scenario explained in Section 2.1. Figure 4 shows the GUI of the prototype where three main areas can be observed:

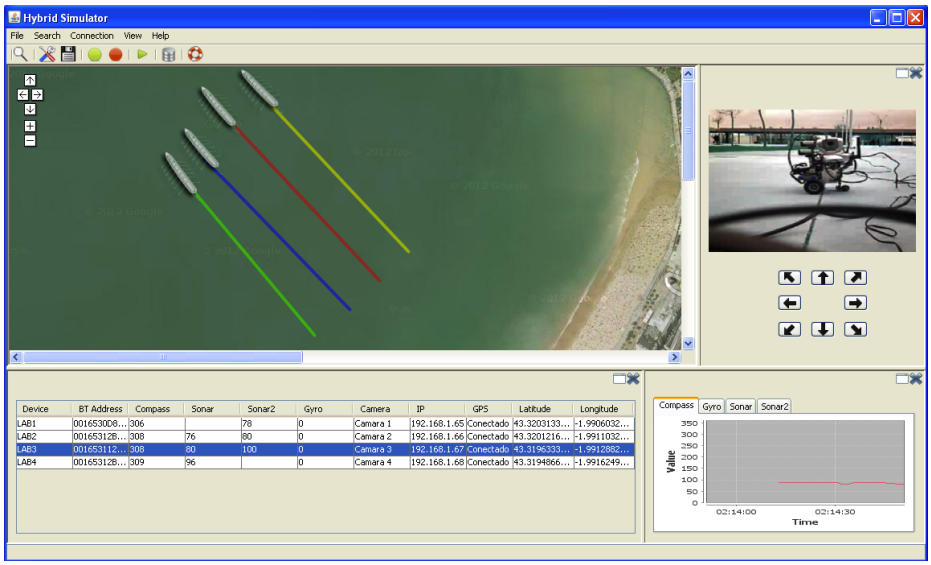


Fig. 4. Graphical User Interface of the prototype developed.

– *Map section*, where the real-time location of the simulated moving objects is shown using Google Maps[6] as GIS.

[6] `http://maps.google.com/`

– *Camera section*, that shows the video-stream of the cameras and enables the user to control them.
– *Sensory Information section*, where data obtained by sensors is shown both in a tabular way and using charts.

In our hybrid simulation we use four LEGO Mindstorms robots to simulate the rowing boats, with a software application (developed using leJOS v0.91[7]) that implements the ideas explained in Section 4. Each robot is equipped with a Bluetooth SysOnChip GPS and an Edimax IP camera. Moreover, we have attached also to the robots a compass, a gyro, and ultrasonic sensors to obtain additional information. The configuration of the robots is shown in Figure 5, notice that all the connection wires have been removed for the picture.

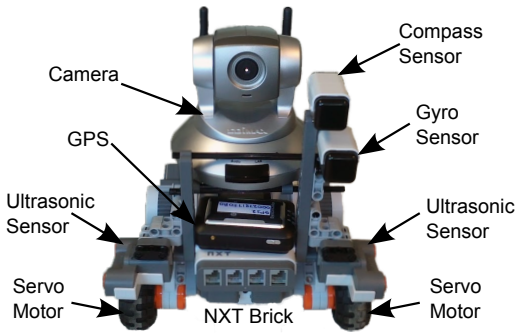


Fig. 5. LEGO Mindstorms configuration used in our test.

In the real rowing race there are four rowing boats involved, each of them is equipped with a GPS transmitter and a video camera. The boats leave from the start line, they sail 1.5 miles, and they come back after making a turn in the sea. To recreate the movement path of the rowing boats, each robot has been programmed to move straight and back to the beginning after performing a 180° turn. The different speeds of the boats have been simulated by adjusting the robots to experience random speed changes during the test (depending on the part of the race). With this configuration, Figure 6 shows a picture of the start of the real rowing race and the same moment in our test.

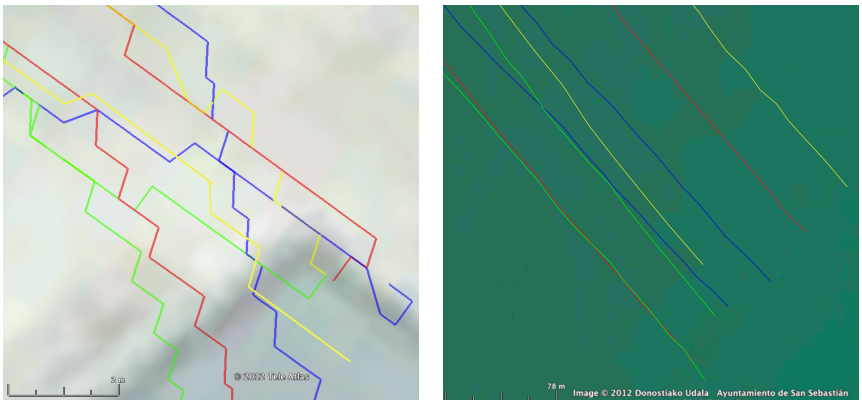
One of the most interesting sensory information in this scenario is the location of the boats, as it can be used for example to check the distance between them. In our tests, the robots obtain their location every second from the GPS and send this information to the base station. Then, the base station translates these locations to the simulated bay of San Sebastian by scaling it to be represented over the map. Figure 7 shows part of the path followed by the robots in our

[7] `http://lejos.sourceforge.net/`

(a) (b)

Fig. 6. Image from the real scenario (a) and image from our test (b).

scaled scenario and the translation to the rowing race scenario[8]. Notice that the GPS accuracy (around 2 meters) makes the paths in Figure 7(a) to cross each other and do not follow a straight line. This is because the distance between the robots was around 20 cm, and a imprecise measurement of the GPS (e.g., 1 meter) in the location of a robot makes it to appear outside its lane. To deal with this, in the translation of Figure 7(b) we have scaled the distance between the robots and the distance they travel each second. Then, we have applied the same error (at maximum was 2 meters) that the GPS obtained at each sample. Notice that the GPS error in Figure 7(b) seems smaller and that is because of the scale of the image.



(a) (b)

Fig. 7. Location obtained by the GPS of the robots during the test (a), and the translation to the simulated scenario (b).

[8] The represented part corresponds to the start/finish of the race; so, for each boat with the same color, it shows its leaving of the harbor and its returning.

Another interesting sensory information is the video stream that the cameras are capturing. We have recorded the whole event and we present in Figure 8 some interesting captures. The captures are obtained for the same time instants for the cameras on board "Robot2", Figure 8(a), and "Robot3", Figure 8(b). Notice that at the first time instant both cameras are recording their adjacent robots, so in the real scenario both cameras would be recording the adjacent boats. Camera from "Robot2" is viewing its adjacent robot, "Robot3", and camera from "Robot3" is viewing its adjacent robot, "Robot2". The cameras we used in our tests can be remotely controlled as the real cameras in the race; so, we rotated the camera on board "Robot2" horizontally to the left during the test, and the effect of this rotation and the different speeds of the robots can be observed in the rest of the captures. The camera on board "Robot3" was not rotated and so, it views "Robot2" during the rest of the captures.

(a)

(b)

Fig. 8. Real captures of Robot2 camera, (a), and Robot3 camera, (b), during a test.

Analyzing the results of the test, the prototype could be useful to train technical directors offline. A technical director must make quick decisions in a live broadcasting to select the camera whose view must be broadcasted. A trained technical director is able to make right decisions in such a scenario, but he/she have to be prepared for unexpected events (for example, a boat crash). Therefore, our prototype could be used to simulate a rowing race and the robots can be controlled to randomly stop or the cameras can be remotely turned off.

7 Related Work

The use of small robots to simulate the behavior of real moving objects has been also considered when testing vehicle scenarios. For example, in [12] they present a system to test Plug-in Hybrid Electric Vehicle scenarios using robots. Their testbed only take into account these scenarios, but we share some common points as the development of a GUI that easily helps the user to analyze the information captured by the robots. In [8] they also use small robots to test real scenarios.

This work is focus on RFID communications between vehicles, so, they use real RFID equipped robots. They program the robots to reproduce real scenarios as we do, but they are not interested in sensory information that the robots could provide.

The term hybrid simulation has been previously used in [3]. In this work, the authors remark that "*the fidelity of simulation results has been a concern*", and that a platform that improves software simulations is needed. Therefore, they presented a hybrid simulation platform used for wireless application/protocol testing, which combines software simulations with real hardware for communication. While we are concerned about moving objects and data access systems, we share the idea of using real hardware to increase the fidelity of the testing of systems.

As we previously commented, LEGO Mindstorms robots are widely used in academic environments thanks to their affordable costs and the ease of their programming. In [6] they explain the features that make LEGO Mindstorms a suitable platform for college students. Among the many works that use LEGO Mindstorms for teaching, in [7] the authors show the benefits of using these robots for teaching introductory programming. In [4], LEGO Mindstorms are used to teach C language in a more appealing way for students. Finally, in [9] the authors present a practical robotics engineering course using LEGO Mindstorms instead of advanced robots. They explain the results of their experience with the students showing that using this affordable robots is enough to introduce them to robotics typical challenges as computer vision, autonomous navigation, etc.

8 Conclusions and Future Work

We have presented an approach for hybrid simulation of wireless data access systems based on the use of small affordable robots. The combination of real hardware with the simulation of the movements or the scenarios, allows our system to increase the fidelity of the tests performed for a wireless data access system compared to the use of software simulations. Therefore our system presents the following benefits:

- It allows testing in simple but real scenarios as a scaled version of the real ones. Therefore, the scenarios will require less area to be deployed and less infrastructure.
- It uses real sensors and communication mechanisms, so wireless data access systems can be tested in environments with real failures, accuracy, and delays.

Our system allows to perform hybrid simulations of different scenarios where moving objects equipped with sensors capture and store data. To configure a specific scenario, the user selects its location and scale as well as the sensors and movement paths for each moving object involved. To improve Bluetooth limitations, as future work, we plan to extend the prototype considering Wi-Fi and ZigBee communications between the robots and the base station. We also plan

to improve the system by adapting the behavior of the robots to autonomously complete predefined tasks in a collaborative approach.

Acknowledgments. This research work has been supported by the CICYT project TIN2010-21387-C02-02 and DGA-FSE.

References

1. B. Bagnall. *Intelligence Unleashed: Creating LEGO NXT Robots With Java.* Variant Press, 2011.
2. D. Baum. *Dave Baum's Definitive Guide to LEGO Mindstorms.* APress L. P., 1st edition, 1999.
3. P. De, A. Raniwala, S. Sharma, and T. Chiueh. Mint: a miniaturized network testbed for mobile wireless research. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 4, pages 2731 – 2742 vol. 4, March 2005.
4. S. H. Kim and J. W. Jeon. Educating C language using LEGO Mindstorms robotic invention system 2.0. In *IEEE International Conference on Robotics and Automation. ICRA 2006*, pages 715 –720, May 2006.
5. P. Kinney. Zigbee technology: Wireless control that simply works. *ZigBee Alliance*, 2003.
6. F. Klassner and S. Anderson. LEGO Mindstorms: not just for k-12 anymore. *Robotics Automation Magazine, IEEE*, 10(2):12 – 18, June 2003.
7. P. B. Lawhead, M. E. Duncan, C. G. Bland, M. Goldweber, M. Schep, D. J. Barnes, and R. G. Hollingsworth. A road map for teaching introductory programming using LEGO Mindstorms robots. In *Working group reports from ITiCSE on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '02, pages 191–201, 2002.
8. J. Munilla, A. Ortiz, and A. Peinado. Robotic vehicles to simulate rfid-based vehicular ad hoc networks. In *3rd International ICST Conference on Simulation Tools and Techniques*, SIMUTools '10, pages 49:1–49:2, 2010.
9. A. C. Murillo, A. R. Mosteo, J. A. Castellanos, and L. Montano. A practical mobile robotics engineering course using LEGO Mindstorms. In *Research and Education in Robotics - EUROBOT 2011*, volume 161 of *Communications in Computer and Information Science*, pages 221–235. Springer Berlin Heidelberg, 2011.
10. A. W. Schueller. *Programming with Robots.* Available online: http://carrot.whitman.edu/Robots/notes.pdf.
11. D. E. Stevenson and J. D. Schwarzmeier. Building an autonomous vehicle by integrating LEGO Mindstorms and a web cam. In *38th SIGCSE technical symposium on Computer science education*, SIGCSE '07, pages 165–169, 2007.
12. W. Z. W. Su and M.-Y. Chow. A digital testbed for a PHEV/PEV enabled parking lot in a smart grid environment. In *Innovative Smart Grid Technologies (ISGT 2012)*, 2012.
13. R. Yus, E. Mena, J. Bernad, S. Ilarri, and A. Illarramendi. Location-aware system based on a dynamic 3D model to help in live broadcasting of sport events. In *19th ACM International Conference on Multimedia (ACMMM 2011)*, pages 1005–1008, November 2011.