Proyecto Fin de Carrera
Ingeniería en Informática

Plataforma para la realización de simulaciones híbridas en entornos de computación móvil

Gorka Guerrero Ruiz

Directores: Roberto Yus Peirote
Eduardo Mena Nieto

Septiembre de 2012

Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

RESUMEN

Durante el 2011 se vendieron 491 millones de smartphones a nivel mundial y 115 millones durante el primer trimestre de 2012. Todos estos dispositivos integran cada vez más sensores como giroscopios o acelerómetros y mecanismos de comunicación como Wi-Fi o Bluetooth, así como GPS, por lo que pueden ser utilizados como *sensores móviles* y obtener remotamente información sensorial del entorno. Por ello, los investigadores en campos como la *computación móvil* tienen en consideración estos dispositivos para desarrollar sistemas que consideren información sensorial obtenida inalámbricamente. Sin embargo, a la hora de validar las distintas propuestas de estos grupos de investigación surge el problema de que utilizar pruebas reales no siempre es posible (por ejemplo, debido a los elevados costes tanto materiales como de usuarios implicados). Por otro lado, la complejidad del mundo real puede ser simplificada de acuerdo a los requisitos de dichas propuestas, de tal forma que es posible considerar abstracciones de los objetos reales involucrados. Así pues, el uso de simulaciones software está muy extendido ya que permiten obtener resultados aproximados con un coste muy reducido. Sin embargo, es difícil desarrollar un modelo fiel del mundo real con el que simular ciertas condiciones del entorno (por ejemplo, éste puede afectar al comportamiento de los sensores, a las comunicaciones, etc.). Una posible solución es utilizar *simulaciones híbridas* en las que intervengan componentes reales (objetos móviles equipados con sensores y mecanismos de comunicación reales) junto con elementos simulados (escenarios reales a escala).

Se plantea para este proyecto el análisis, estudio y desarrollo de una plataforma que permita realizar simulaciones híbridas de escenarios donde objetos móviles equipados con sensores capturen información del entorno. Esta plataforma hace uso de pequeños robots de coste asequible para simular objetos móviles, y permite la validación de sistemas de acceso a datos sensoriales con pruebas que ofrecen mayor realismo que simples simulaciones software ya que involucra: errores y retrasos reales en las comunicaciones (puesto que utiliza comunicaciones inalámbricas reales), lecturas reales de los sensores (ya que se usan sensores reales), etc. La plataforma permite el control de los dispositivos móviles de forma remota, realizando los movimientos definidos por el usuario y obteniendo datos del entorno mediante los sensores configurados. También muestra de forma visual (con tablas, gráficas, mapas, etc.) los datos que los robots envían de forma inalámbrica. Además, la plataforma almacena los datos recibidos en una base de datos de forma que, en caso de querer acceder a los datos tras una simulación, podemos realizar un análisis de los mismos.

Para la realización de este proyecto se utilizan diferentes tecnologías como: robots *Lego Mindstorms* con firmware *leJOS*, comunicaciones inalámbricas mediante *Bluetooth*, *Java* y diferentes librerías externas para el desarrollo de la aplicación, *Google Earth* como utilidad donde representar la localización de los robots y recrear vistas de cámaras, *MySQL* para el almacenamiento de la información, etc. Además, como resultado se ha publicado un artículo en las *Jornadas de Ingeniería del Software y Bases de Datos* de 2012 donde se explica el análisis y desarrollo de la plataforma.

Agradecimientos

En primer lugar quiero agradecer a Roberto y a Eduardo la posibilidad de acabar con ellos la carrera al ofrecerme este proyecto y también agradecer su ayuda a lo largo del mismo y a todos aquellos profesores que también han estado ahí durante estos últimos años.

También agradecer a todos mis amigos, a todos los compañeros de fatigas con los que he compartido duros momentos de estudio y sesiones de prácticas, así como de momentos de diversión y entretenimiento y, en general, a todas aquellas personas con las que he compartido conocimientos y dudas a lo largo de estos años en la Universidad y fuera de ella, ya que también tengo que acordarme de todos aquellos compañeros que he conocido durante mis viajes y congresos gracias a la asociación ISC. Si os mencionara a todos vosotros, necesitaría de un anexo para no dejarme a nadie.

Y para acabar, he de agradecer a mis padres Carmen y Vicente, y en especial a mi hermano Ander, sin olvidarme del resto de mi familia, la comprensión, el apoyo y la ayuda que me han prestado a lo largo de toda la carrera, en particular durante los primeros años de carrera, ya que todos los comienzos son difíciles y con su ayuda pude llegar hasta donde estoy ahora. Muchas gracias.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El número de teléfonos móviles vendidos en España durante el pasado año 2011 alcanzó los 20 millones de unidades, de los cuales 9.8 millones corresponden a dispositivos smartphones[1]. Las ventas mundiales de smartphones durante el primer trimestre del 2012 ascienden a 144.9 millones, mientras que durante el 2011 se vendieron 491.4 millones de unidades, según datos de IDC[2]. Además de smartphones, la venta de tablets ha crecido considerablemente con respecto al año anterior, experimentando un incremento del 182 % frente a los decrementos en las ventas del resto de componentes electrónicos relacionados con la informática[3]. Todos estos dispositivos tienen integrados cada vez más sensores (por ejemplo, giroscopios, acelerómetros, cámaras, etc.) y suelen disponer de mecanismos de comunicación como Wi-Fi, Bluetooth o 3G, así como mecanismos de posicionamiento como GPS. Estas características los convierten en candidatos idóneos para ser utilizados como *sensores móviles* y obtener remotamente información sensorial del entorno. Por ello, la investigación en campos como la *computación móvil* o la *minería de datos* tienen en consideración estos dispositivos para desarrollar sistemas que consideren información sensorial obtenida mediante comunicación inalámbrica.

Para validar estos sistemas los investigadores deben realizar distintos tipos de tests, siendo los más recomendados los test en condiciones reales, pero probar el sistema de este modo no siempre es posible. Por ejemplo, evaluar un sistema en un escenario donde podemos encontrar un gran número de coches y personas moviéndose en un área determinada de tamaño considerable puede implicar grandes costes tanto materiales como humanos. Sin embargo, desde el punto de vista de un sistema que procesa datos obtenidos del entorno, la complejidad del mundo real puede ser simplificada de acuerdo a la información que queremos obtener. En el ejemplo

[1] Nota de prensa 7 de Febrero de 2012, Asociación de Empresas de Electrónica, Tecnologías de la Información, Telecomunicaciones y Contenidos Digitales (AMETIC)

[2] `http://www.idc.com/getdoc.jsp?containerId=prUS23299912` - Última consulta 30/08/2012

[3] "Las Tecnologías de la Información" - Informe AMETIC 2011.

anterior, si el sistema está centrado en obtener la localización de los coches y las personas, éstos pueden ser simplificados como simples abstracciones de dichos objetos móviles y que realizan los mismos recorridos. Por esta razón, los investigadores usan *simulaciones software* para probar sus propuestas, reduciendo considerablemente el coste y obteniendo resultados aproximados. Sin embargo, obtener un modelo real del entorno para simular condiciones ambientales realistas (por ejemplo los retrasos o desconexiones en comunicaciones Wi-Fi, el fallo en un sensor, etc.) es un gran desafío.

Una posible solución a este problema es la utilización de *simulaciones híbridas* que permiten recrear ciertos escenarios reales de forma más precisa que las simulaciones software. Consideramos que una simulación híbrida consiste en el uso de elementos reales (objetos móviles equipados con sensores y mecanismos de comunicación reales) como abstracción de objetos que podrían encontrarse en una prueba real del sistema, y elementos simulados (escenarios reales a escala). Por ejemplo, consideremos un sistema que analiza en tiempo real la información sensorial adquirida en un evento deportivo en el que objetos móviles (barcas) están equipados con sensores de localización, cámaras de vídeo y comunicación inalámbrica; generar un modelo real para simular dicho escenario es una tarea compleja (hay que simular lo que verían las cámaras, recrear cómo las condiciones ambientales afectarían a los distintos sensores, etc.), sin embargo, utilizando una simulación híbrida, ciertos elementos serán simulados (por ejemplo, la escala del escenario) a la vez que se utilizan elementos hardware reales (cámaras, elementos móviles, GPS, Wi-Fi, etc.).

La Tabla 1.1 muestra una comparación de tres aproximaciones diferentes para realizar pruebas de sistemas de acceso a información sensorial del entorno (pruebas reales, simulaciones software y simulaciones híbridas). Comparada con una simulación software, la simulación híbrida maneja retrasos reales en la comunicación (debido a que usa mecanismos reales de comunicación), sensores reales (que obtienen información real) y condiciones medioambientales reales. Sin embargo, la duración de un test en una simulación híbrida es mayor que en un simulador software (por ejemplo, debido al intervalo máximo de adquisición de muestras por parte de los sensores software), y el coste del despliegue puede ser también mayor. Comparado con una prueba real, la simulación híbrida requiere un escenario e infraestructuras menores (al poder ser escalado), es más fácil el repetir la prueba (por ejemplo, obtener nuevos datos con diferentes configuraciones), la duración de las pruebas pueden ser reducidas y el coste es bastante inferior.

Por lo tanto, consideramos que la realización de simulaciones híbridas puede suponer una mejora con respecto al tradicional uso de simulaciones software. Sin embargo, así como disponemos de multitud de herramientas para realizar simulaciones software, no disponemos en la actualidad de una plataforma que permita configurar los elementos que intervienen en nuestro escenario para realizar una simulación híbrida. Así pues, se pretende con este trabajo el análisis, estudio y desarrollo de una plataforma para realizar simulaciones híbridas en entornos de computación móvil.

Aspectos a comparar	Test real	Simulación software	Simulación híbrida
Retrasos realistas en la comunicación	✓✓	✗	✓
Sensores usados	✓✓	✗	✓
Escenario	✗	✓✓	✓
Repetitividad	✗	✓	✓
Interacción con el entorno	✓	✗✗	✓
Intervalo de tiempo	✗	✓✓	✓
Costes	✗	✓✓	✓

Tabla 1.1: Comparando las diferentes aproximaciones: test reales, simulaciones software y simulaciones híbridas

1.1. Objetivos

El objetivo principal de este PFC consiste en desarrollar una plataforma que, haciendo uso de dispositivos móviles equipados con sensores, permita realizar simulaciones de escenarios reales gracias a los datos obtenidos por dichos dispositivos y enviados de forma inalámbrica. Para ello, nuestra plataforma permitirá:

- Recrear escenarios reales que serán usados en nuestra simulación y sobre los que representaremos las abstracciones de los dispositivos móviles reales usados.
- Buscar dispositivos móviles reales de forma inalámbrica, encargados de recopilar los datos sensoriales del entorno que se mostrarán al usuario.
- Definir las configuraciones (sensores, motores, movimientos, etc.) de dichos dispositivos de forma remota de acuerdo con las necesidades del usuario.
- Representar gráficamente los datos obtenidos por los sensores de los dispositivos móviles, permitiendo al usuario acceder a ellos de forma sencilla.
- Almacenar los datos para un posterior análisis de los mismos ya que durante el transcurso de la prueba es posible no detectar determinados fallos en la comunicación o lecturas erróneas desde los sensores.
- Tratar con sensores muy heterogéneos, por ejemplo de localización o cámaras (permitiendo poder controlar estas últimas).

El diseño e implementación de la plataforma desarrollada ha sido realizado atendiendo a los requerimientos y directrices de los directores del proyecto. Además. debido a que la plataforma va a ser utilizada en el grupo de Sistemas de Información Distribuidos (SID)[4] de la Universidad de Zaragoza para evaluar algunos de los sistemas

[4] `http://sid.cps.unizar.es/`

que desarrollen, se ha tenido en cuenta la usabilidad y mantenimiento de la plataforma, intentando que pueda ser modificada o ampliada con nuevas funcionalidades en un futuro de forma sencilla.

1.2. Estructura de la memoria

El resto de la memoria sigue la siguiente estructura:

- **Capítulo 2, *Contexto tecnológico***: describe las características de las tecnologías y herramientas utilizadas en el desarrollo del proyecto.
- **Capítulo 3, *Estado del arte***: describe los trabajos relacionados analizando sus diferencias con nuestra propuesta.
- **Capítulo 4, *Análisis y diseño***: describe el proceso de desarrollo de las aplicaciones de escritorio y para lose dispositivos móviles.
- **Capítulo 5, *Evaluación experimental***: se muestran las pruebas realizadas a lo largo del proceso de desarrollo con los resultados de las mismas.
- **Capítulo 6, *Conclusiones***: se analizan los objetivos cumplidos, se describen las posibles líneas de trabajo futuro para la plataforma y se muestran las conclusiones y opiniones personales del autor sobre el trabajo realizado.

Además, se incluyen los siguientes anexos que sirven como ampliación de la información de la memoria:

- **Anexo A, *Análisis y diseño***: describe de forma más detallada el proceso seguido para la realización del proyecto junto con los hitos y fases más importantes. Se detallan además las fases de análisis, diseño e implementación del proceso de desarrollo tanto de la aplicación de escritorio como de la aplicación para los dispositivos.
- **Anexo B, *Manual de usuario***: explica la instalación de la plataforma además de las pautas generales de uso.
- **Anexo C, *Casos de uso***: se indican de forma detallada varios ejemplos de uso completos de las partes más importantes de la aplicación.
- **Anexo D, *Artículo aceptado en JISBD'12***: se adjunta el artículo titulado "*Using Small Affordable Robots for Hybrid Simulation of Wireless Data Access Systems*", que ha sido aceptado en la decimoséptima edición de las "Jornadas de Ingeniería del Software y Bases de Datos".

Capítulo 2

Contexto tecnológico

En este capítulo se detallan las diferentes tecnologías que han sido utilizadas en el PFC. En primer lugar se muestra el análisis de los distintos candidatos disponibles para ser usados como dispositivo móvil en nuestra plataforma junto a sus características. Posteriormente analizamos los distintos tipos de firmwares existentes para el dispositivo elegido (*Lego Mindstorms*), concluyendo el capítulo con el lenguaje de programación escogido para la implementación de la aplicación de escritorio así como el software y el resto de librerías usadas durante la fase de desarrollo de la plataforma.

2.1. Dispositivos móviles

Para desarrollar nuestra plataforma necesitamos un dispositivo que sea fácil de programar, modificable y configurable, ya que va a ser la representación simplificada del objeto que queramos simular. Es necesario que el dispositivo pueda ser controlado remotamente y que realice una serie de movimientos definidos por el usuario, a la vez que recopile información sensorial (posición GPS, dirección de la brújula, velocidad de giro, etc.) para ser enviada a la aplicación. Es por ello que necesitamos que dicho dispositivo tenga la posibilidad de comunicarse de forma inalámbrica, que pueda ser configurado con distintos sensores y motores en función de los datos a medir, y que el tamaño y el coste del mismo sea reducido.

Atendiendo a nuestras necesidades, en el mercado encontramos varios dispositivos que sirven para nuestro propósito. Como uno de los requisitos previos es que los dispositivos tengan un coste asequible, descartamos en primer lugar dispositivos tales como robots complejos. Por ello, los dispositivos que más se adaptan a nuestros requisitos son el robot *Lego Mindstorms NXT* y la placa *Arduino*. Posteriormente ha aparecido otro dispositivo que también podría haberse usado para nuestros fines, la placa *Raspberry PI*, aunque no se ha tenido en cuenta debido a la fecha de lanzamiento de la misma y su similitud con la placa Arduino. A continuación procedemos

a analizar los citados dispositivos, haciendo especial hincapié en aquel que hemos seleccionado para la realización del proyecto.

2.1.1. Arduino

Arduino [4] es una plataforma de hardware libre muy popular, formado por una placa con un micro controlador (generalmente un *Atmel AVR*) y un entorno de desarrollo. Debido a la sencillez de los diseños y a la variedad de placas disponibles, es posible encontrar varios tipos de micro controladores, siendo los más usados el *Atmega168*, *Atmega328* y *Atmega1280*. Todos ellos comparten características similares en cuanto a frecuencia de reloj (16 MHz), voltaje operativo (5 V), intensidad de corriente (40 mA), y voltajes de entrada recomendados y límite (7-12 V y 6-20 V), pero difieren en el resto de sus especificaciones, como por ejemplo la memoria FLASH disponible (16 KB, 32 KB y 128 KB, respectivamente), y en la SRAM y EEPROM disponibles (1 KB / 512 bytes, 2 KB / 1 KB y 8 KB / 4 KB, respectivamente).

Las citadas placas de Arduino tienen en común el uso del puerto USB que tienen integrado para la transmisión de datos y la posibilidad de ampliación mediante placas adicionales conocidas como *shields*. Los shields mas extendidos son el *Shield Xbee*, *Shield Motores,* y *Shield Ethernet*. Principalmente, las funcionalidades de los shields son las de proporcionar al Arduino mecanismos de comunicación inalámbrica debido a que, salvo por el modelo de Arduino dotado de un dispositivo Bluetooth, las demás placas carecen de esta funcionalidad.

2.1.2. Lego Mindstorms

Lego Mindstorms [5] es un kit de robótica fabricado por la empresa Lego y desarrollado en colaboración con el MIT a finales de 1990. Aunque está orientado al ámbito comercial, también se vende como herramienta educacional bajo el nombre de *Lego Mindstorms for Schools*. El kit de Lego Mindstorms se compone principalmente de un bloque programable, un conjunto de piezas para el montaje de diseños, una colección de sensores y motores, y un software de programación gráfica bastante intuitivo ya que el público objetivo son los niños de edades comprendidas entre los 10 y 14 años. Tras el lanzamiento inicial, la comunidad de aficionados a la robótica acogió con interés este nuevo producto, apareciendo las primeras comunidades de usuarios que ampliaban las posibilidades del producto original mediante la creación de entornos de programación alternativos y nuevos sistemas operativos o firmwares para el bloque de control.

Posteriormente apareció en el mercado una nueva revisión, conocida como *NXT* (ver Figura 2.1(a)) y que es la que vamos a analizar. El bloque NXT es la evolución del anterior bloque, el *RXC* y que es comercializado en dos versiones: *Retail Version* y *Education Base Set*. Además, Lego liberó varios kits para desarrolladores:

Software Developer Kit (SDK), con los controladores del puerto USB, archivos ejecutables y referencia a bytecodes, *Hardware Developer Kit* (HDK), con documentación y esquemas para los sensores del NXT, y *Bluetooth Developer Kit* (BDK), con documentos de los protocolos usados para la comunicación Bluetooth, convirtiéndose en un sistema *open source* tanto en el hardware como en el software.

En cuanto a las especificaciones hardware, el bloque NXT posee un procesador principal *Atmel ARM* de 32 bits con 256 KB de memoria FLASH, 64 KB de memoria RAM y con una velocidad de reloj de 48 MHz, y un co-procesador *Atmel AVR* de 8 bits Atmega48, con 4 Kb de memoria FLASH, 512 Bytes de memoria RAM y a una frecuencia de 8 MHz. Como comunicaciones disponibles, el bloque NXT dispone de un puerto USB e incluye además un chip *CSR BlueCore4* para comunicaciones Bluetooth, además de varios puertos E/S a los que poder conectar diferentes motores o sensores, en una disposición que podemos ver en la Figura 2.1(b).

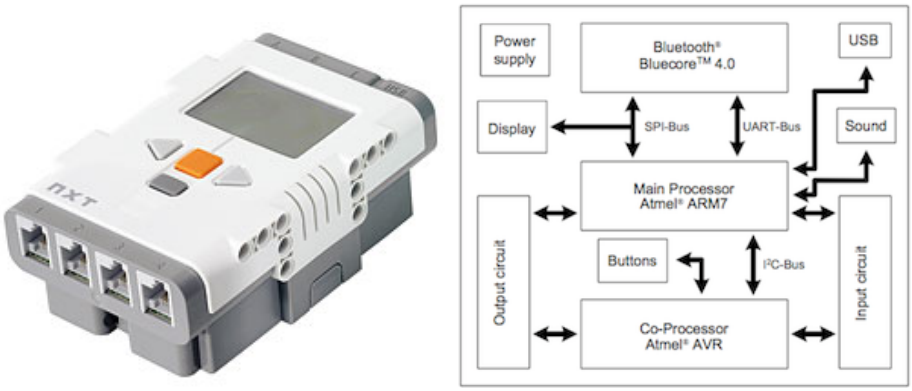

Figura 2.1: Lego Mindstorms NXT (a) y su diagrama hardware (b)

Hemos mencionado que al bloque NXT se le pueden acoplar distintos tipos de sensores. Al ser la plataforma Lego Mindstorms un sistema hardware *open source*, no sólo tenemos disponibles los sensores que comercializa Lego, sino que podemos encontrar sensores de terceros que funcionan en el bloque sin incompatibilidades, aumentando las funcionalidades del mismo. Los sensores iniciales incluidos en el kit son los siguientes:

- *Ultrasonidos*, devuelve la distancia hasta un obstáculo.
- *Luz / Color*, detecta diferentes niveles de luz y colores bien en escala de grises o de colores.
- *Presión*, detecta colisiones.
- *Sonido*, mide los niveles ambientales de sonido.

pero también podemos encontrar sensores de terceras partes. Por ejemplo, Hi-Technics[1], Mindsensors[2] y Dexter Industries[3] son algunas de las empresas que distribuyen sensores tales como:

- *Acelerómetro*, mide la aceleración e inclinación en tres ejes.
- *Brújula*, mide la dirección en la que está orientado el sensor.
- *Presión barométrica*, mide la presión atmosférica y la temperatura, así como la altitud.
- *Campos magnéticos*, capaz de detectar campos magnéticos presentes en el entorno del sensor.

La Figura 2.2 muestra algunos de los sensores disponibles. Además de los citados sensores, también tenemos disponibles mecanismos de posicionamiento como GPS, que pueden ser integrados en los dispositivos NXT, bien de forma inalámbrica o por cable. Estos fabricantes también comercializan dispositivos de comunicación para los NXT como por ejemplo infrarrojos, Wi-Fi o *XBee* [10].

Figura 2.2: Varios de los sensores para dispositivos NXT

2.1.3. Comparativa

A modo de resumen, la Tabla 2.1 muestra las principales características de los dispositivos analizados, que procedemos a comparar a continuación.

Atendiendo a las características, podemos observar que ambos dispositivos son bastante sencillos en cuanto a hardware (procesador y memoria), aunque el bloque NXT es ligeramente superior ya que la frecuencia de reloj es mayor, dispone de más memoria y tiene un segundo procesador que trabaja en colaboración con el procesador principal (controlando los botones y puertos del dispositivo). Aunque la placa Arduino es un dispositivo económico (con un precio orientativo de 20 €[4]

[1] `http://www.hitechnic.com/`
[2] `http://mindsensors.com/`
[3] `http://dexterindustries.com/`
[4] Precio extraído de la tienda online Cooking Hacks Store (a fecha 30/08/2012) - `http://cooking-hacks.com`

	Procesador Ppal. / Sec.	Frecuencia Proc. Ppal. / Sec.	Memoria Flash Ppal. / Sec.	Memoria RAM Ppal. / Sec.	Comunicaciones (por defecto)	Conexiones E/S disponibles	Hardware Adicional incluido	Comunicaciones soportadas	Sensores adicionales	Contenido adicional	Precio base
Arduino (placa Mega)	Atmega328 / -	16 MHz / -	256 KB / -	8 Kb / -	USB	14 E/S digit. 6 Entr. analog.	-	Wi-Fi, 3G, ZigBee, Bluetooth, etc.	Si	-	46€ *
Kit Lego Mindstorms	Atmel ARM /Atmel AVR	48 MHz / 8MHz	256 KB / 4 KB	64 KB / 512 B	USB, Bluetooth	4 Entr. digit 3 Entr. Analog.	3 motores Sensores varios (sonido, luz, distancia, presión)	Wi-Fi, 3G, ZigBee, etc.	Si	Piezas de montaje	299.95 €

* La versión básica incluye sólo la placa. De incluir los sensores y módulos necesarios para equipararse al kit Lego, el precio asciende a 225-250€ (shield Bluetooth, motores, sensores varios)

Tabla 2.1: Comparativa de dispositivos para simular objetos móviles

para la placa en su modelo Uno), es necesario adquirir shields adicionales para las comunicaciones, además de los sensores y motores que pudiéramos necesitar (lo que hace que el precio total oscile entre los 225 - 250 €). Por contra, el kit NXT ya incluye comunicación por Bluetooth además de un conjunto de sensores básicos y motores, aunque el desembolso inicial es mayor (con un precio orientativo del kit de 299.95 €[5]). Por lo tanto, debido a que las características y precio de los dos productos analizados son bastante similares y siendo que tanto el propio grupo SID como otros grupos del departamento ya disponían de varios kits de Lego Mindstorms, se ha decidido seleccionarlos para simular objetos móviles en nuestra plataforma.

2.2. Firmwares disponibles

El NXT viene equipado de fábrica con un firmware diseñado para soportar los programas desarrollados con el software de programación *NXT-G*. Dicho software está basado en el entorno de programación gráfico *LabVIEW* que se asemeja a la creación de diagramas de flujo. De esta forma, el NXT puede ser fácilmente programado por usuarios que carecen de nociones de programación (por ejemplo estudiantes de primaria y secundaria). Sin embargo, para explotar la potencia del dispositivo utilizando programas complejos, suele ser necesario utilizar otros lenguajes de programación que abarcan desde Ada hasta Python, pasando por Lua, Perl, MatLab, LISP, Objective-C, o C++, e incluso lenguajes menos habituales como OCaml, Forth, o Haskell. Para ello disponemos de diferentes firmwares alternativos fácilmente instalables y que soportan dichos lenguajes de programación.

Vamos a analizar los entornos de programación más utilizados por la comunidad de desarrolladores.

[5]Precio extraído de la tienda online Elecbricks (a fecha 30/08/2012) - `http://www.electricbricks.com/lego-mindstorms-8547-lego-mindstorms-nxt-v20-p-3301.html`

2.2.1. NBC / NXC

NBC (NeXT Byte Codes) [24] es un lenguaje similar al lenguaje ensamblador Lego NXT compuesto por dos partes: el lenguaje *NBC* que describe la sintaxis a usar para escribir programas y el *API NBC* que describe las funciones del sistema, constantes, etc. que pueden ser usadas por los programas. Los programas escritos con este lenguaje pueden ejecutarse en el dispositivo NXT mediante la traducción a bytecode NXT utilizando el firmware por defecto. Por otra parte, *NXC (Not eXactly C)* [24] es un lenguaje de alto nivel similar a C desarrollado inicialmente para el primer RCX aunque actualizado para el dispositivo NXT. Los programas escritos en NXC son traducidos también a bytecode NXT, por lo que podemos usar el firmware original. Sin embargo, para tener acceso a todas las características que nos brindan NBC y NXC existe un firmware alternativo, mediante el cual tendremos acceso a todas las operaciones y macros disponibles. Además, NBC / NXC carecen de entornos de desarrollo, pero pueden usarse en el *IDE BricxCC*, mediante el cual también podremos modificar el firmware del dispositivo.

2.2.2. RobotC

RobotC [15] es un entorno de desarrollo para NXT desarrollado por la Universidad *Carnegie Mellon* que usa un lenguaje de programación basado en el lenguaje C estándar. RobotC es ampliamente utilizado en el ámbito de la enseñanza [34]. Los programas desarrollados con RobotC han de ser ejecutados en un firmware especial incluido que ha de ser cargado previamente en el dispositivos NXT. Además, es necesario adquirir una licencia para acceder a la versión del software RobotC sin restricciones (con un precio de 49 $ al año).

2.2.3. LeJOS

LeJOS [14] es un firmware alternativo que permite la creación de programas usando el lenguaje de programación Java (en realidad se trata de una versión reducida de las clases disponibles debido a las limitaciones de espacio existentes). Como Java no es un lenguaje soportado de forma nativa por el dispositivo NXT, es necesario sustituir el firmware, por lo que, junto a la API de leJOS, se incluye un firmware que instalará una reducida máquina virtual de Java sobre la que ejecutaremos los programas. Una característica interesante es que los desarrolladores de leJOS han ocultado los detalles de la implementación de los sensores, permitiendo al usuario final trabajar con un alto nivel de abstracción sin tener que preocuparse por detalles como los componentes hardware o direcciones hexadecimales. Al ser un firmware creado por una comunidad de usuarios, la versión actual (0.9.1) todavía se considera *beta* debido a las nuevas funcionalidades que se van añadiendo constantemente y a los fallos del firmware que se van solucionando. Al igual que RobotC, leJOS también es ampliamente usado en el ámbito de la enseñanza (sobre todo universitaria) [35].

2.2.4. Comparativa

A modo de resumen, la Tabla 2.2 muestra los diferentes lenguajes junto con sus principales características.

	Licencia	Precio	Versión (fecha)	IDE	Lenguaje	Firmware alternativo necesario
RobotC	Propietary Comercial Software	49$*	3.0 (09/2011)	Incluido	C estándar	Si
NXB / NXC	Open source	-	1.2.1 r5 (09/2011)	No **	Ensamblador / C estándar	Si
leJOS	Open source	-	0.9.1 beta (02/2012)	No ***	Java	Si

* Licencia de 1 año
** Disponible IDE de terceras partes, BricxCC
*** Disponible IDE de terceras partes, Eclipse o NetBeans mediante uso de plugins

Tabla 2.2: Comparativa de lenguajes de programación para Lego Mindstorms

Al ser RobotC un software en el que es necesario adquirir una licencia para su uso y que ofrece unas características muy similares a las de leJOS, se descartó en favor de los otros dos lenguajes. Debido además a que el API y la ayuda existente de leJOS es bastante amplia, habiendo disponibles libros [30, 38], foros de comunicación activos y ejemplos de ayuda, se optó por este lenguaje. También, al tratarse del lenguaje Java, la familiaridad con él es mayor al haberse usado a lo largo de la carrera en diferentes asignaturas.

2.3. Mecanismos de comunicación

Ya que los NXT han de enviar información de forma inalámbrica a la aplicación de escritorio que estará ejecutándose en un ordenador, hemos analizado los distintos mecanismos de comunicación que tenemos disponibles para ellos.

2.3.1. Wi-Fi

El estándar *IEEE 802.11* en cualquiera de sus variantes (a, b, g, o n), conocido como *Wi-Fi* [13], es también un mecanismo de conexión de dispositivos de forma inalámbrica definidos dentro de una *Red de Área Local Inalámbrica* (*WLAN*). El protocolo Wi-Fi está pensado para desplegar redes con un mayor alcance, por lo que en función del estándar empleado, tendremos unas velocidades u otras, pudiendo

alcanzar también distancias mayores. Además, es posible dotar a los NXT de comunicación vía Wi-Fi con la compra de adaptadores cuyo precio se sitúa en torno a los 100 $.

2.3.2. ZigBee

ZigBee [11] está basado en el estándar *IEEE 802.15.4* de *Redes Inalámbricas de Área Personal* (*WPAN*) y su uso principal está pensado en el ámbito de la domótica debido a su bajo consumo, su fácil integración y la topología en red de malla. Una red ZigBee puede constar de un máximo de 65535 nodos, distribuidos en subredes con 255 nodos cada una, no siendo necesaria la visión directa entre los dispositivos. Los dispositivos ZigBee también tienen un alcance limitado, con un rango de transmisión teórico de entre 10 a 75 metros para la versión básica y de hasta 1500 metros para la versión PRO, aunque estos rangos son muy dependientes del entorno. La potencia máxima es de 1 mW en la versión básica, aumentando hasta los 100 mW para la versión PRO, lo que le permite alcanzar las distancias comentadas. Debido a que el consumo eléctrico en los dispositivos ZigBee es bastante reducido (al permanecer el dispositivo la mayor parte del tiempo en reposo, y disponer de una velocidad de transmisión de unos 250 Kbps), este sistema está bastante extendido en campos como la telemedicina, la domótica o en aquellos productos dependientes de la batería.

2.3.3. Bluetooth

Se denomina *Bluetooth* [12] al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, con una cobertura baja y basados en transceptores de bajo costo. Se ha considerado una tecnología usada para redes *WPAN* realizándose las comunicaciones por radiofrecuencia, por lo que no es necesario que los dispositivos tengan visión directa el uno del otro, pudiendo incluso estar en habitaciones separadas si la potencia de transmisión lo permite. Atendiendo a la potencia de transmisión, los dispositivos se clasifican por "clases" (ver Tabla 2.3), siendo compatibles los dispositivos de distintas clases entre sí.

Clase	Potencia máxima permitida		Rango (aprox.)
	(mW)	(dBm)	
Clase I	100 mW	20 dBm	~100 metros
Clase II	2.5 mW	4 dBm	~10 metros
Clase III	1 mW	0 dBm	~1 metro

Tabla 2.3: Diferencias entre las clases Bluetooth

El número de dispositivos que podemos tener conectado de forma simultánea utilizando Bluetooth es reducido. En una configuración maestro/esclavo, son 7 los dispositivos esclavos que puede manejar un dispositivo maestro como máximo al

mismo tiempo en una red conocida como *piconet*. El número de dispositivos puede aumentar en una configuración *scatternet*, que es una red basada en un número determinado de redes piconet. Una scatternet se forma cuando un dispositivo Bluetooth (maestro o esclavo) es escogido para participar como esclavo en otra piconet separada. El dispositivo que participa en ambas piconets puede compartir información entre ellas, aunque el protocolo básico no soporta dicha configuración, por lo que es necesario que el software de cada dispositivo gestione las conexiones.

La versión de Bluetooth incluida en el dispositivo NXT es *2.0 + EDR* (*Enhanced Data Rate* o "mayor velocidad de transmisión de datos"), que acelera la transferencia de datos sobre la versión anterior de Bluetooth. La tasa nominal de dicha versión es de 3 Mbit/s, aunque si consideramos la tasa de transferencia de datos efectiva, se reduce a 2.1 Mbit/s. Para reducir el consumo eléctrico usado por el Bluetooth, éste ha sido implementado como un dispositivo Bluetooth de Clase II.

2.3.4. Comparativa

En la tabla de la Tabla 2.4 se puede ver a modo de resumen cada una de las características anteriormente comentadas para los mecanismos de comunicación analizados.

	Número de dispositivos (red)	Velocidad teórica	Alcance máximo	Potencia máxima	Precio
ZigBee (normal / PRO)	255	250 Kbps	75 / 1500 m **	1 / 100 mW	78$
Wi-Fi (b / g / n)	255	11 / 54 / 300 Mbps	140 / 140 / 250 m	Según legislación local	99.99$
Bluetooth (v2.0 + EDR, Clase II)	8 *	3 Mbps	10 m	2.5 mW	0***

* 8 dispositivos en una red Piconet. El número de dispositivos aumenta en redes scatternet
** Al aumentar la potencia, se alcanzan distancias superiores
*** Incluido en el NXT

Tabla 2.4: Comparativa de mecanismos de comunicación

Podemos ver que, tanto las redes Wi-Fi como ZigBee soportan un mayor número de dispositivos que una piconet Bluetooth, sin necesidad de recurrir a configuración de redes de tipo scatternet. El sistema ZigBee tiene una velocidad menor, superado en velocidad por el Bluetooth y encontrando las mayores velocidades en las redes

Wi-Fi, sobre todo en aquellas de más reciente aparición. Las distancias que podemos alcanzar también son mayores en estos dos últimos casos, pudiendo posicionar dispositivos hasta incluso más de un kilómetro en el caso de ZigBee (el modelo Pro es capaz de alcanzar dicha distancia debido a un incremento en la potencia empleada), aunque siempre con la limitación de que las distancias son bastante dependientes del entorno.

Por el contrario, para usar tanto Wi-Fi como ZigBee como mecanismo de comunicación entre los dispositivos NXT y la estación base, es necesario adquirir sensores adicionales al no ser un sensor estándar incluido por defecto en el kit (el precio de estos elementos se sitúa en torno a los 100$). Por ello hay que recurrir a fabricantes externos, teniendo en cuenta que es necesario adquirir tantos sensores como dispositivos tengamos y que necesitaremos conectarlos a uno de los puertos disponibles del robot NXT, por lo que tendremos menos puertos libres para el resto de los sensores. Además, en el momento del análisis de estas tecnologías, no había soporte oficial de leJOS por parte del fabricante de los sensores.

Bluetooth ha sido ampliamente utilizado en proyectos en los que intervienen NXT y se dispone de una API con las librerías necesarias para utilizarlo. Además, tanto el alcance máximo como el número de dispositivos conectados son suficientes para la realización de simulaciones. Aun así, no se descarta que en un futuro se pudieran usar Wi-Fi o ZigBee en el caso de necesitar aumentar la superficie sobre la que desplegamos nuestros dispositivos y por lo tanto se desarrollará la plataforma facilitando el posible cambio en el mecanismo de comunicación.

2.4. Lenguaje de programación

En esta sección se explica el lenguaje de programación usado así como las librerías auxiliares que han sido necesarias para el desarrollo de la aplicación de escritorio.

2.4.1. Java

Java es un lenguaje de programación orientado a objetos con el que se puede realizar programas que pueden ejecutarse en ordenadores independientemente del sistema operativo que estén ejecutando, ya que el único requisito es que tenga instalada una máquina virtual. Gracias a ello se consiguen programas multiplataforma sin tener que preocuparse por las librerías propias de cada sistema operativo. Java es un lenguaje que además ha sido usado a lo largo de la carrera de Ingeniería en Informática, por lo que ya se tienen conocimientos previos tanto en el lenguaje como en el desarrollo de interfaces gráficas, manejo de bases de datos, etc. Además, hay disponibles multitud de librerías externas que facilitan el desarrollo de ciertos módulos de la plataforma y la integración con otros sistemas.

Debido a que hemos escogido leJOS como el firmware que tendrán instalados nuestros dispositivos NXT, la integración con Java es total al estar basado en dicho lenguaje de programación, pudiendo hacer uso de los mismos entornos de desarrollo disponibles. Además, leJOS proporciona un completo API para acceder completamente a las funcionalidades del firmware, así como de ayuda bastante extensa, accesible tanto vía web como integrada en el IDE usado para desarrollar los programas.

2.4.2. Librerías auxiliares

Para el desarrollo de la aplicación de escritorio se han usado diferentes librerías y frameworks ya existentes como complementos al software desarrollado, a continuación mostramos aquellas más importantes:

- **JFreeChart**: era un requisito de la plataforma mostrar toda la información recopilada de los sensores en forma de gráficas, por lo que para ello hemos usado *JFreeChart* [16]. Esta librería, distribuida bajo licencia *LGPL* (*Licencia Pública General Reducida de GNU*, o más conocida por su nombre en Inglés *GNU Lesser General Public License*), nos permite crear una serie de gráficas diferentes seleccionando las escalas de los ejes y leyendas, y además permite tanto la manipulación de las mismas mediante el uso del ratón como la actualización en tiempo real con los datos que indiquemos.

- **JWebBrowser**: en nuestra aplicación queremos mostrar la localización de los NXT, por lo que necesitaremos de un soporte para poder realizarlo. Ya que las principales aplicaciones de mapas necesitan de un navegador web para poder ejecutarse, vamos a usar *JWebBrowser*, que es uno de los componentes que incluye la librería *NativeSwing* [17] bajo licencia *LGPL*. Dicho componente permite insertar un navegador web completamente funcional en nuestra aplicación Java, pudiendo mostrar cualquier página web y soportando además ejecución de código *JavaScript*.

- **Bluecove**: es una librería Java para Bluetooth [18] que sirve como interfaz de la pila de Bluetooth de Microsoft que podemos encontrar en Windows XP. Bluecove viene incluida junto con el firmware de leJOS y se distribuye bajo licencia *Apache Software, Versión 2.0*.

2.4.3. Google Earth

Ya que consideramos que los robots pueden estar equipados de un dispositivo GPS queremos mostrar la localización de los mismos en un mapa. En este caso hemos decidido utilizar *Google Earth* [27], una herramienta muy potente similar a un *Sistema de Información Geográfica* (GIS en sus siglas en inglés) que nos permite

visualizar imágenes del planeta combinando imágenes de satélite y de mapas y sobre el que podemos representar nuestros dispositivos gracias a modelos 3D. Además, tenemos acceso al API de Google Earth, mediante la cual podremos manipular el mapa según las necesidades del usuario y alterar la posición de los elementos que representemos en él. También se han analizado otros GIS como *OpenStreetMaps* o *Google Maps*, pero se ha seleccionado finalmente Google Earth ya que, además de ofrecer mayor cantidad de información geográfica que OpenStreetMap, puede utilizarse para recrear en un entorno virtual la visión de las videocámaras como proponen en [29].

2.4.4. Otro software

Además, durante el desarrollo del proyecto se utilizaron las siguientes aplicaciones:

- *Eclipse Índigo* [19], IDE usado para el desarrollo de las aplicaciones de escritorio y para los dispositivos NXT.
- *WindowBuilder* [23], herramienta usada para desarrollar interfaces gráficas en Java mediante un editor visual.
- *Gantt Project 2.5.4* [22], aplicación para la elaboración de diagramas de Gantt.
- *Dia 0.97.1* [21], usada para la elaboración de diagramas de casos de uso, diagramas de clases y trazas de eventos durante la fase de diseño.
- *InkScape 0.48.3 [20],* utilizado para el diseño de imágenes.
- *LaTeX* [1] ,utilizado para la elaboración de la documentación. Los componentes utilizados fueron:
  - *MiKTeX 2.9* [2], distribución TeX / LaTeX para Microsoft Windows.
  - *LyX 2.0.3* [3], entorno gráfico para escribir documentos en LaTeX.

Capítulo 3

Estado del arte

Los robots NXT son cada vez más populares ya que, además de ser usados en el ámbito de la enseñanza escolar, educación especial, y enseñanza universitaria [36, 37], son usados también por programadores y aficionados a la robótica en general. A consecuencia de ello, los NXT son utilizados generalmente para simular comportamientos autónomos [39, 40], no existiendo muchas aplicaciones cuyo objetivo sea controlarlos remotamente. Por lo tanto, nos vamos a centrar en aquellas herramientas que permitan buscar, configurar y controlar los robots NXT, así como herramientas que recopilen y muestren información sensorial gracias a sensores acoplados en robots NXT.

3.1. NXJ Control Center

Al sustituir el firmware original por el firmware leJOS disponemos de una serie de aplicaciones que vienen incluidas junto con el firmware para interactuar con los dispositivos. Una de ellas es el programa *NXJ Control Center* (ver Figura 3.1), que nos permite configurar, controlar y monitorizar dispositivos NXT de forma remota. Para ello, permite configurar un dispositivo NXT emparejado mediante Bluetooth con el ordenador, con la limitación de que el programa no permite el control simultaneo de varios dispositivos NXT. Entre las distintas opciones disponibles, podemos configurar los sensores disponibles en el dispositivo indicando los puertos a los que están conectados. El programa representa los datos que obtiene de los sensores sobre unos monitores específicos para cada sensor configurado, teniendo que actualizar manualmente los monitores para obtener la información más reciente. Además, nos permite configurar los motores del dispositivo así como variar sus velocidades para moverlo remotamente.

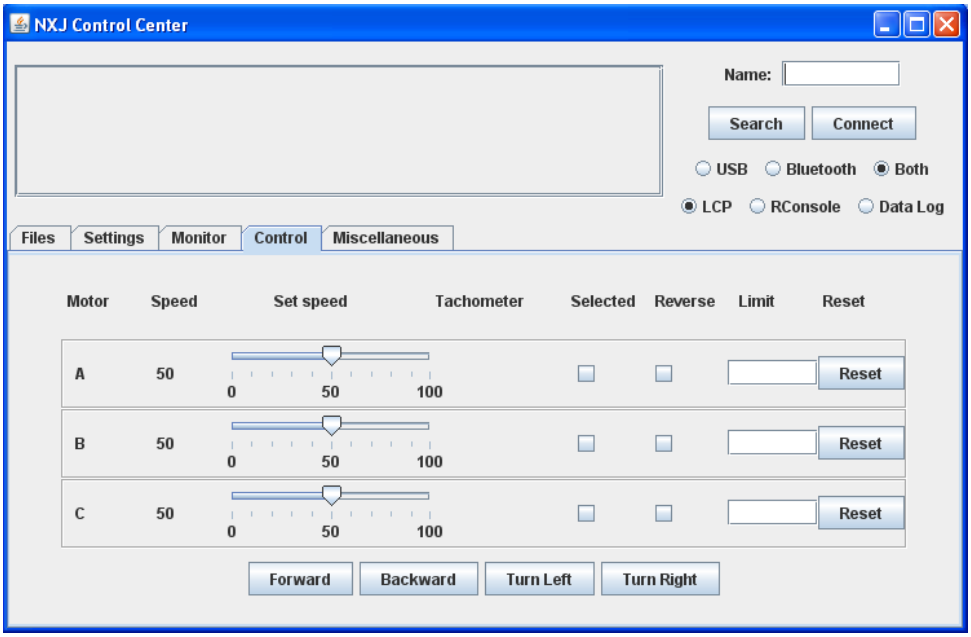

Figura 3.1: Aplicación NXJ Control Center

3.2. Remote Control Application

La herramienta *Remote Control Software Application* [6] (ver Figura 3.2) es un software desarrollado como proyecto de final de año del curso 2007/2008 en el *School of Computing, Dublin Institute of Technology*. Dicha herramienta hace uso de una librería externa de NXT (*iCommand*) para el control remoto de un único dispositivo mediante Bluetooth con el firmware leJOS instalado, pudiendo controlar los movimientos del robot y obtener la posición del mismo en una representación del mapa en un sistema de coordenadas X-Y (aunque para ello no hace uso de ningún mecanismo de posicionamiento, simplemente indica los movimientos realizados sobre dicho sistema de coordenadas). Sin embargo, la herramienta no considera la existencia de sensores (ni por lo tanto la obtención de datos sensoriales y su configuración). Al hacer uso de una librería externa no es necesario tener instalado un programa en el dispositivo NXT.

3.3. NXT Vehicle Remote

La herramienta *NXT Vehicle Remote* [7] (ver Figura 3.3), permite el control de forma remota de un dispositivo NXT con el firmware original instalado. Dicha herramienta, disponible para sistemas operativos Windows, permite buscar y controlar el dispositivo NXT mediante Bluetooth, aunque hemos de indicarle el puerto de comunicación al que ha de conectarse. Una vez conectado, la aplicación nos permite controlar al robot NXT bien mediante teclado o ratón, además de permitir

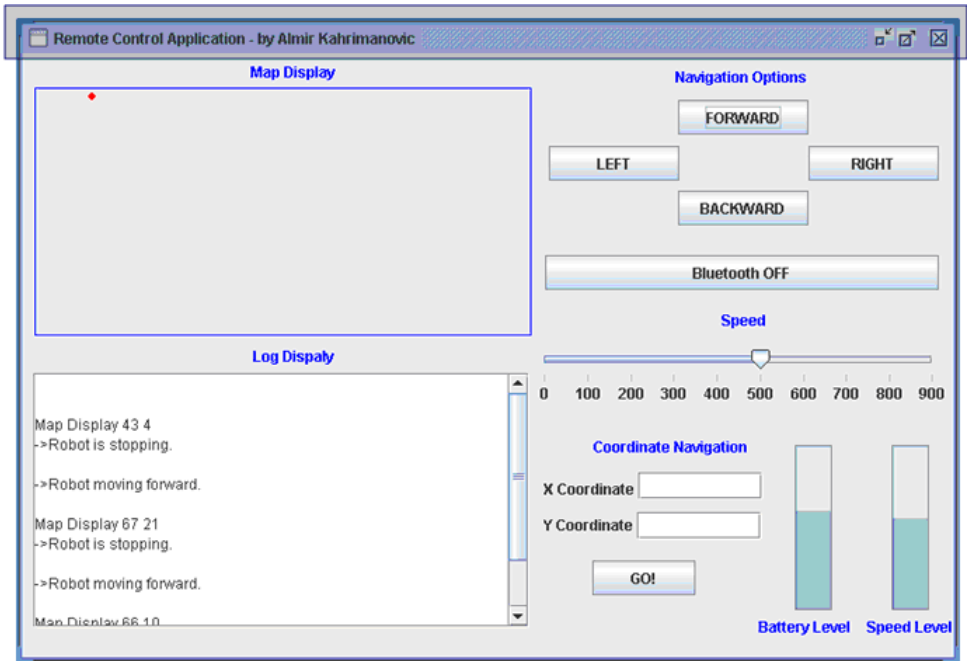

Figura 3.2: Aplicación Remote Control Application

configurar los sensores conectados a los puertos del mismo para obtener los datos que capturen, pero sin tener la posibilidad de añadir nuevos sensores a la aplicación. La herramienta tampoco muestra la representación gráfica de los sensores ni de la posición del dispositivo en un mapa o similar.

3.4. Robótica Móvil con Lego Mindstorms

En el Proyecto de Fin de Carrera de David Pellicer Martín [9] (Universidad de Zaragoza, Noviembre de 2010), se hace uso también de los robots Lego Mindstorms para "analizar las posibilidades de dichos robots para el aprendizaje y/o investigación práctica sobre robots móviles autónomos". En dicho proyecto se analizan varios vehículos construidos para estimar su posición y movimiento, se estudia la posibilidad de generar un modelo del entorno mediante el uso de los sensores disponibles o de determinar el movimiento de un objeto. También se analiza la búsqueda de rutas óptimas y la navegación por ellas tomando como base un mapa del entorno. En dicho proyecto se analizan además las comunicaciones PC - NXT mediante el protocolo Bluetooth, estableciendo el autor un protocolo de comunicación para el envío de los mensajes en ambas direcciones.

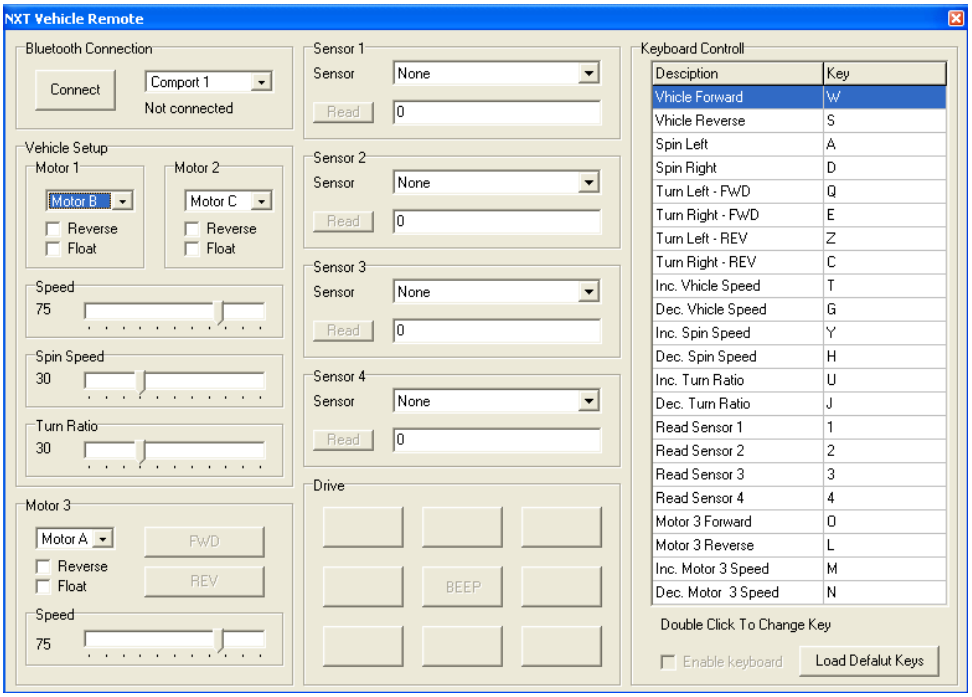

Figura 3.3: Aplicación NXT Vehicle Remote

3.5. CoLego Track (Group 7)

Este proyecto [8], desarrollado por alumnos de la asignatura de *Sistemas de Control Embebido* de la *Universidad de Uppsala* en Suecia, hace uso de varios dispositivos NXT los cuales, mediante una cámara, son capaces de determinar la posición de un objeto haciendo uso del algoritmo *DEFK (Distributed Extended Kalman Filter)*. Cada robot es capaz de enviar información a un programa principal que analizará dicha información por cada uno de los robots conectados y será capaz de determinar la posición y velocidad de dicho objeto. Si fuera necesario, el programa tiene la posibilidad de indicar a los dispositivos que se desplacen de forma autónoma a otra posición desde la que sea mejor analizar al objeto. En el documento se incluye el análisis de distintos escenarios con distintas configuraciones, junto con los resultados obtenidos. Se puede observar que un programa de escritorio es capaz de trabajar conjuntamente con varios dispositivos NXT conectados por Bluetooth a él. Además, para la comunicación Bluetooth, los alumnos hacen uso de una librería creada por el mismo autor del programa visto en la Sección 3.3.

3.6. Comparativa

Se incluye en la Tabla 3.1 una comparativa de las características principales de las aplicaciones analizadas con respecto a la que se ha desarrollado fruto de este

proyecto (SkyNXT). En primer lugar, aunque las aplicaciones principalmente están pensadas para manejar un único dispositivo simultáneamente, tanto las aplicaciones CoLego Track como SkyNXT, pueden controlar varios dispositivos a la vez. Nótese que mientras CoLego Track puede controlar 3 robots, SkyNXT permite controlar hasta 7 NXT[1] (no confundir con Skynet, que podía controlar miles de T-800 Terminator[2]). Todas las aplicaciones permiten buscar dispositivos de forma inalámbrica por defecto, aunque no todas permiten controlar y configurar los dispositivos de forma remota.

	NXJ Control Center	Remote Control Application	NXT Vehicle Remote	Robótica móvil	CoLEGO Track	SKyNXT
Número de dispositivos soportados	1	1	1	1	3	7
Conexión inalámbrica	✓	✓	✓	✓	✓	✓
Configuración de sensores / motores	✓	✗	✓	✗	✗	✓
Control remoto del dispositivo	✓	✓	✓	✗	✗	✓
Representación de los datos sensoriales	✓	✗	✓	✗	✗	✓
Representación de la posición en un mapa	✗	✓	✗	✓	✓	✓
Es necesario un programa específico para funcionar	✗	✗	✗	✓	✓	✓
Comportamiento autónomo del dispositivo	✗	✗	✗	✓	✓	✗

Tabla 3.1: Comparativa de las aplicaciones analizadas

En las aplicaciones que permiten configurar los sensores de forma remota se representan también los datos, aunque de distinta forma entre ellas. En la aplicación NXJ Control Center se representa en unos monitores circulares, que además hay que actualizar manualmente, mientras que la aplicación NXT Vehicle Remote se indica el valor del dato leído. SkyNXT, por el contrario, muestra tanto el último valor leído en una tabla como un histórico de los datos agrupados en una gráfica.

[1]El límite de conexiones Bluetooth explicado en la Sección 2.3.3.

[2]Ficha de la película en IMDB `http://www.imdb.com/title/tt0088247/` - Última consulta 30/08/2012

Las aplicaciones que permiten el control remoto del NXT (NXJ Control Center, Remote Control Application y NXT Vehicle Remote) realizan los movimientos en base a la pulsación de los botones dispuestos en la interfaz gráfica, mientras que la aplicación SkyNXT permite la realización de movimientos mediante diferentes opciones (pulsación de teclado o definición de trayectorias). Asociado al movimiento de los dispositivos, no todas las aplicaciones permiten representar sobre un sistema de coordenadas, escenario o mapa la posición de los mismos. Las aplicaciones Remote Control Application y CoLego Track permiten la representación de la posición del dispositivo en un mapa representado por el eje de coordenadas X-Y, mientras que las aplicaciones Robótica Móvil y SkyNXT permiten representar la posición sobre un mapa manual del entorno o sobre un GIS como Google Earth, respectivamente.

Debido a que varias de las aplicaciones analizadas (NXJ Control Center, Remote Control Application y NXT Vehicle Remote) están orientadas al público en general, no es necesario que los dispositivos tengan cargada ninguna aplicación específica en la memoria ya que es la aplicación de escritorio la que se encarga de las comunicaciones y movimientos de los dispositivos. En cambio, el resto de las aplicaciones analizadas si que necesitan una aplicación cargada en memoria y ejecutándose para recibir las órdenes de la aplicación principal. Salvo las aplicaciones de Robótica Móvil y CoLego Track, cuyos dispositivos realizan tareas de forma autónoma sin intervención del usuario, los robots del resto de las aplicaciones no pueden considerarse que tengan un comportamiento autónomo al necesitar intervención del usuario para funcionar.

Capítulo 4

Análisis y diseño

En este capítulo se presenta brevemente la fase de análisis y diseño que incluye aspectos tales como el análisis de requisitos, prototipado de las ventanas y diagramas de clases. Para el desarrollo del proyecto se ha decidido por seguir una metodología *iterativa e incremental* siguiendo el desarrollo del sistema mediante la repetición de las distintas fases que a su vez se incrementan gradualmente, permitiendo tomar ventaja de lo aprendido durante el desarrollo de las partes anteriores del sistema. Para obtener información adicional acerca de los contenidos de este capítulo, el Anexo A contiene el *documento de Análisis y Diseño*.

4.1. 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 4.1) y de la aplicación de escritorio (ver Tabla 4.2) permiten dar una visión de todas las funcionalidades que presenta el sistema, mientras que los requisitos no funcionales (ver Tabla 4.3 y Tabla 4.4 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 4.1: 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 4.2: 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 4.3: 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 4.4: Requisitos no funcionales de la aplicación

4.2. 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 4.1 muestra una visión general del sistema con todos los módulos que intervienen en él.

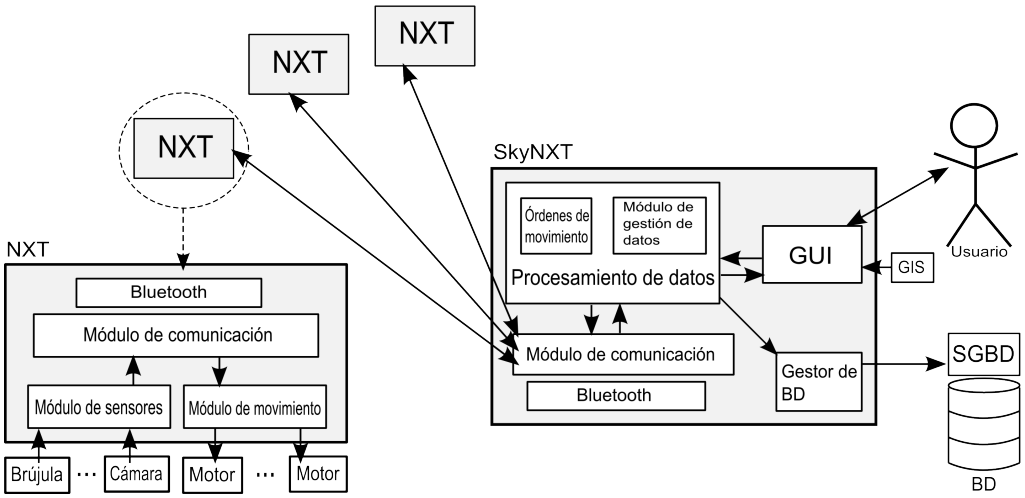

Figura 4.1: Arquitectura general de la plataforma

Observamos varios módulos principales en la aplicación cargada en el robot NXT (la Figura 4.2 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 4.3). Observamos

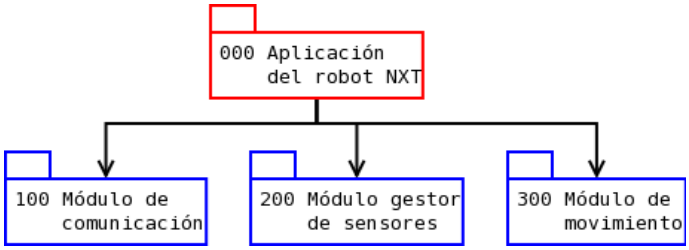

Figura 4.2: Arquitectura general de la aplicación para el robot NXT

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.

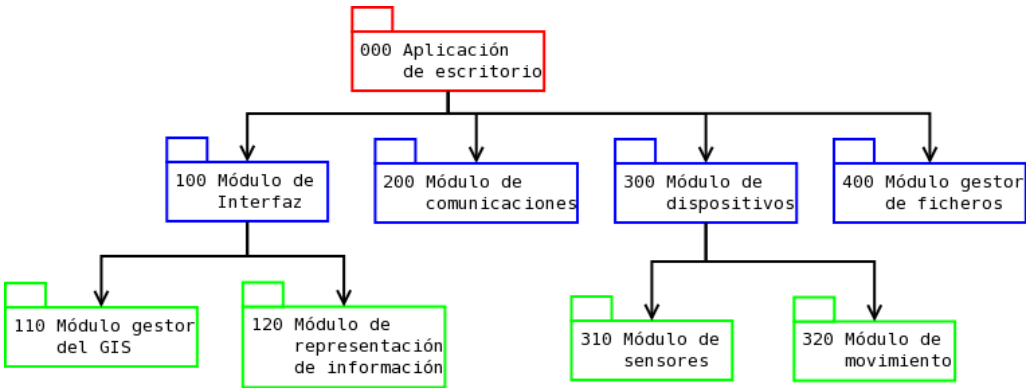

Figura 4.3: Arquitectura general de la aplicación de escritorio

4.3. Prototipado del GUI

Durante el proceso de análisis y diseño se han elaborado varios prototipos de la interfaz gráfica de la aplicación con diferentes propuestas. El prototipo final elegido para la interfaz gráfica puede observarse en la Figura 4.4 (para consultar el proceso seguido en el desarrollo de la interfaz gráfica con sus distintos prototipos véase la

Sección A.4). Del mismo modo, la Figura 4.5 muestra una imagen de la interfaz gráfica final de la aplicación donde se puede observar como se han implementado las ideas del prototipo.

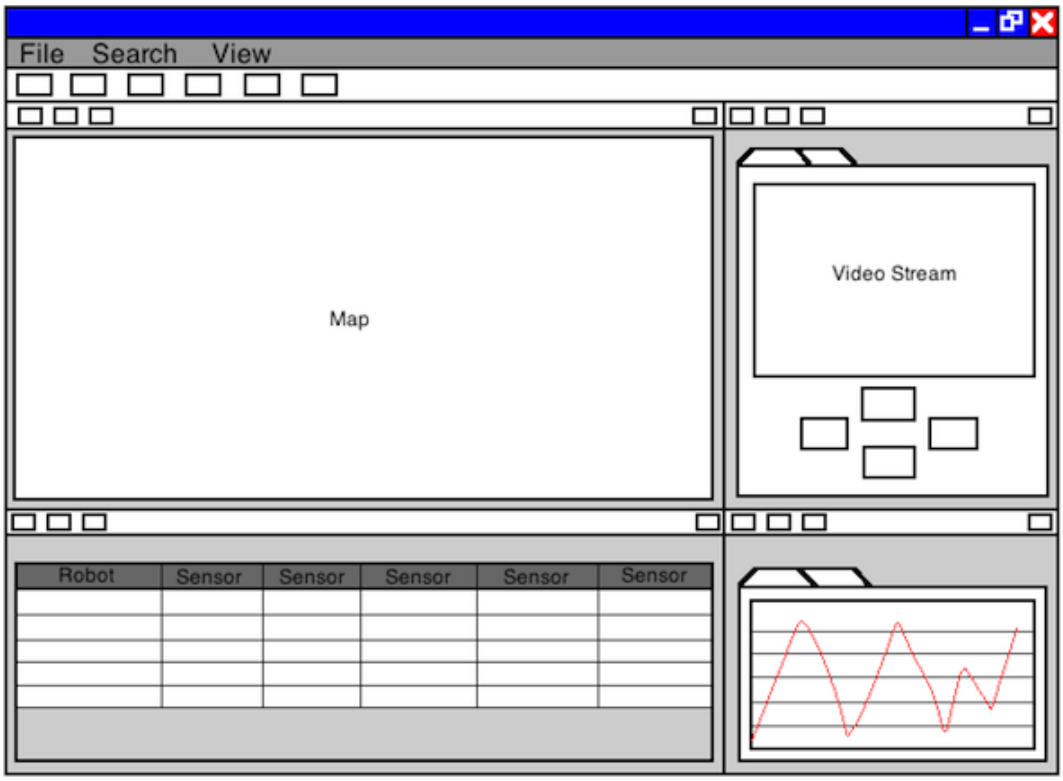

Figura 4.4: Prototipo final de la interfaz gráfica de la aplicación de escritorio

4.4. Diagrama de clases

A continuación mostramos en la Figura 4.6 un diagrama de clases simplificado de la plataforma SkyNXT desarrollada. En el podemos ver las clases principales de la aplicación de escritorio y la aplicación para los NXT (incluyendo un breve resumen de cada una de ellas) así como las relaciones existentes entre ellas. En el Anexo A se encuentra una explicación más detallada del presente diagrama así como otros diagramas de más bajo nivel de las distintas clases de las aplicaciones.

- *GUI*: interfaz gráfica de la aplicación de escritorio que engloba a todas las ventanas y paneles que componen la aplicación.
- *Configuración*: se encarga de la configuración de la aplicación, permitiendo añadir nuevos sensores y cámaras.

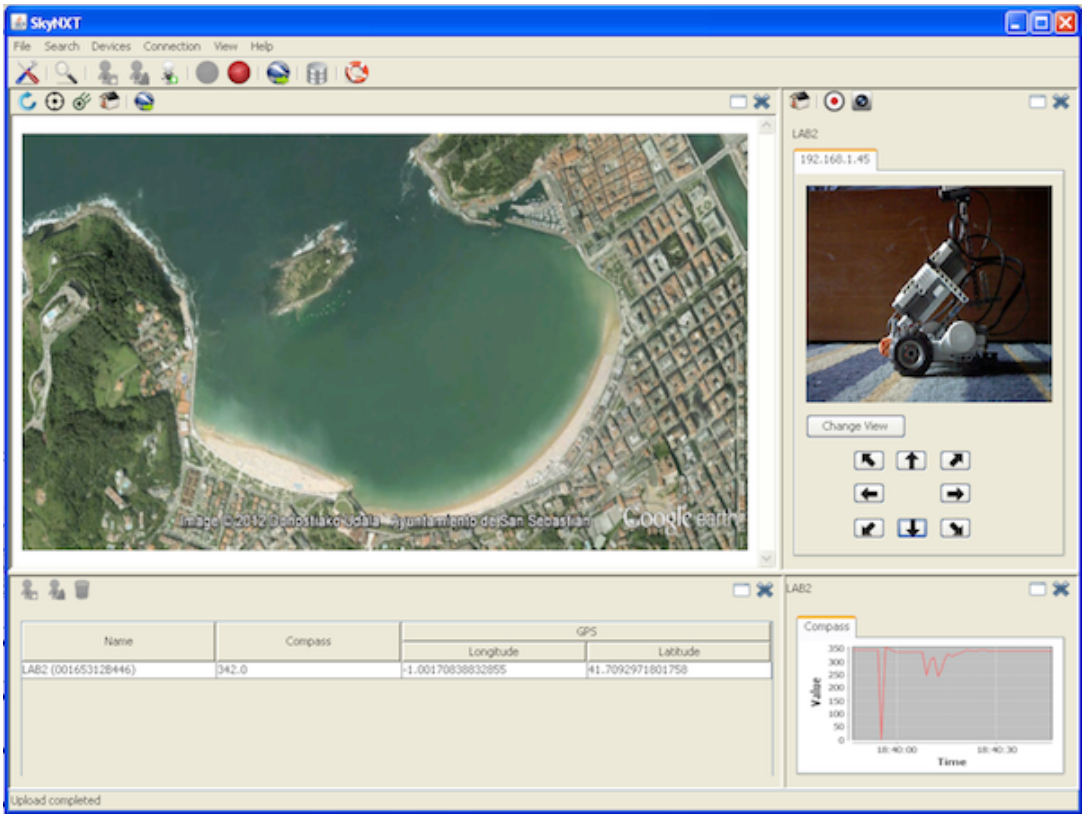

Figura 4.5: Interfaz gráfica final de la aplicación de escritorio

- *ConfigurarDisp*: se encarga de la configuración de un dispositivo con los sensores y cámaras añadidos en la aplicación.
- *ConfigurarMovs*: se encarga de la configuración de los movimientos de un dispositivo.
- *Dispositivo*: define un dispositivo que representa al dispositivo NXT y que será configurado para ser usado en la simulación.
- *Cámara*: define una cámara con todas sus propiedades necesarias que podrá ser usada en un dispositivo.
- *Sensor*: define un sensor con todas sus propiedades necesarias que podrá ser usado en un dispositivo.
- *Trayectoria*: define una trayectoria que puede ser seguida por un dispositivo como uno de los movimientos posibles que puede realizar.
- *Puntos*: define los puntos que componen las trayectorias que pueden ser definidas.
- *Búsqueda*: se encarga de la búsqueda y localización de los dispositivos NXT.

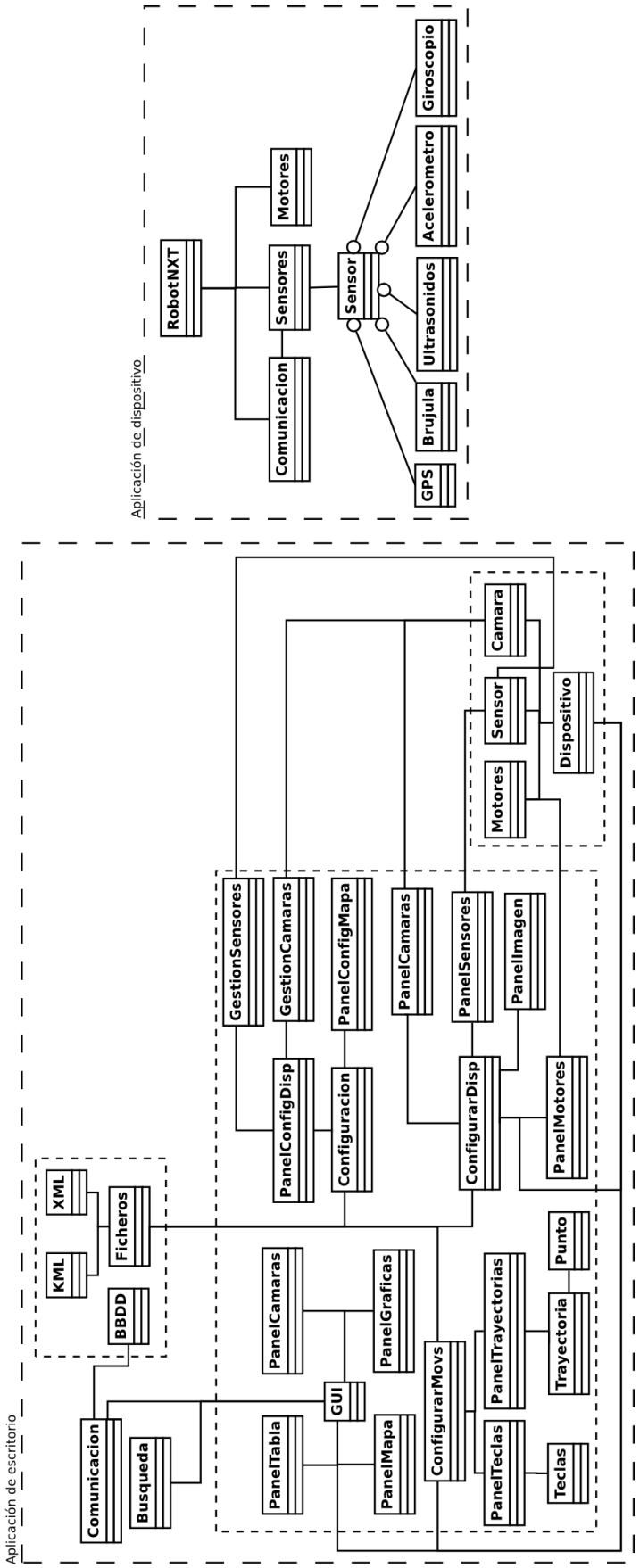

Figura 4.6: Diagrama de clases simplificado

- *Ficheros*: se encarga de la creación y lectura de los distintos ficheros usados por la aplicación. Incluye las clases para el tratamiento de los distintos ficheros empleados.

- *Comunicación*: se encarga de las comunicaciones entre el dispositivo y la aplicación de escritorio.

Además también tenemos las clases de la aplicación del dispositivo móvil, que comentamos brevemente:

- *RobotNXT*: es la clase principal de la aplicación y define un robot NXT.

- *Comunicación*: se encarga de las comunicaciones entre el dispositivo y la aplicación de escritorio.

- *Sensores*: se encarga de comenzar el muestreo de datos de cada uno de los sensores con los parámetros definidos.

- *Sensor*: clase que extiende al resto de sensores existentes y se encarga de la lectura de los datos desde ellos.

- *Motores*: se encarga del movimiento de los dispositivos.

4.5. Estudio de traslación de coordenadas

Una de las características que tiene la plataforma SkyNXT es que permite situar el movimiento que realizan los robots en cualquier punto del mapa. Por ello, en algunas ocasiones será necesario que traslademos la posición actual 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. Por ejemplo, si queremos simular una trayectoria que en la realidad transcurre a lo largo de un kilómetro en un espacio de 10 metros, deberemos trasladar las coordenadas GPS obtenidas por los robots escalándolas con un factor de 100 (ver Figura 4.7).

Sin embargo, el posicionamiento GPS no es cien por cien preciso pudiendo obtener errores de entre 1 a 5 metros. 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 (el proceso se explica con más detalle en la Sección A.8), nuestra idea se basa en los siguientes puntos:

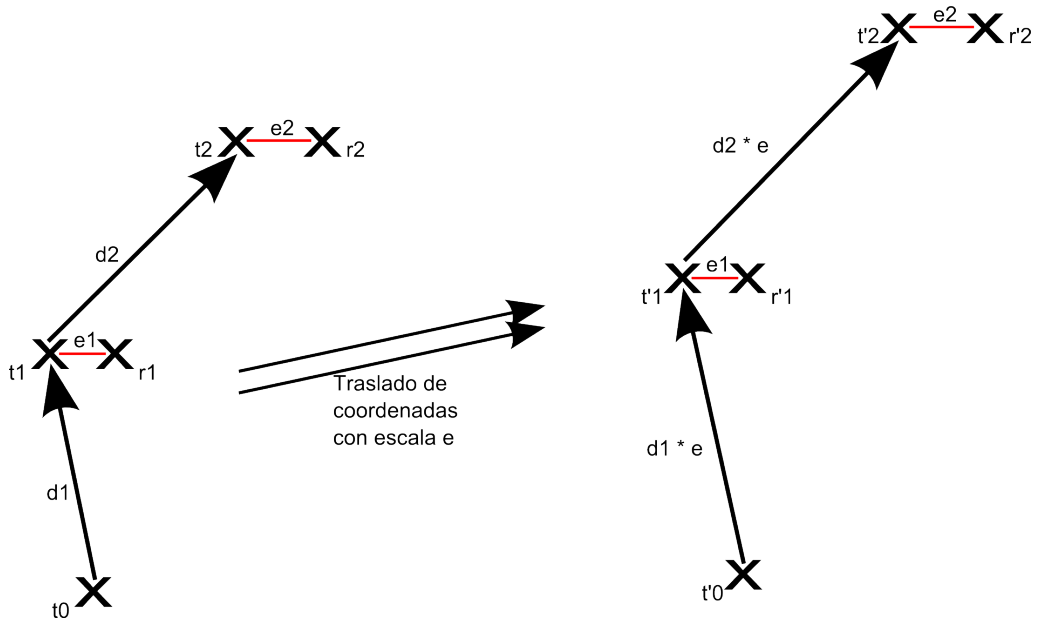

Figura 4.7: Concepto de trasladar la posición real a otra definida por el usuario

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.

Capítulo 5

Evaluación experimental

En este capítulo vamos a comentar las pruebas realizadas a lo largo del desarrollo del proyecto. En primer lugar mencionamos las pruebas realizadas durante la fase de análisis a los distintos sensores disponibles para el dispositivo NXT, posteriormente nos centramos en las pruebas individuales realizadas a las distintas librerías usadas para integrarlas en nuestro sistema y verificar que cumplían los requisitos y finalmente detallamos las pruebas finales realizadas a la plataforma.

5.1. Pruebas previas

La realización de estas pruebas tenía como finalidad tanto probar el comportamiento del NXT con el firmware elegido como los distintos sensores que habíamos adquirido. En esta fase se probaron todos los sensores disponibles de forma individual, con programas de prueba específicos que fueron desarrollados para el dispositivo NXT, además de que también se probaron los dispositivos GPS que se usarían en el proyecto, ya que al tener que comunicarnos vía Bluetooth con ellos, teníamos que ver que el dispositivo NXT era capaz de mantener la comunicación en todo momento, obteniendo los resultados que se resumen en la Tabla 5.1. Probar los distintos sensores ha servido para conocer previamente los valores que recogen y han de enviar a la aplicación de escritorio.

Se probó también el funcionamiento de las cámaras, tanto de forma individual como de forma global en una red Wi-Fi local, para verificar el correcto funcionamiento de las mismas y sus características. La última prueba antes de proceder a la fase de desarrollo consistió en comprobar que el dispositivo pudiera ser controlado remotamente desde una aplicación externa. Para ello utilizaron librerías de control remoto y se enviaron comandos vía Bluetooth.

Para conocer el límite máximo de dispositivos soportados, contactamos con Ana Cristina Murillo, profesora de la asignatura de “Robótica de servicio” en la que se

Sensor	Lecturas obtenidas
Brújula	Ángulo con respecto al Norte (0 - 359)
Giroscopio	Velocidad de giro (grados/seg) del sensor
Acelerómetro	Velocidad (en mg) en las coordenadas X, Y y Z
Ultrasonido	Distancia hasta el objetivo (0 - 255) en centímetros
Luz B/N	Intensidad de luz ambiental (0 (oscuridad) - 1300 (claridad))
Luz Color	Color leído por el sensor (valores RGB)
GPS	Devuelve distintos valores (latitud, longitud, altura, etc.)

Tabla 5.1: Resultados de las pruebas realizadas a los distintos sensores

usan también dispositivos NXT, que nos cedió temporalmente 4 robots NXT adicionales para nuestras pruebas (ver Figura 5.1). Gracias a ellos, pudimos comprobar como, efectivamente, el número máximo de dispositivos que pueden ser controlados simultáneamente con nuestra aplicación es siete, coincidiendo con las especificaciones de Bluetooth que hemos mencionado anteriormente.

Figura 5.1: Dispositivos NXT usados para las pruebas previas

5.2. Pruebas de integración

Al usar distintas librerías externas en nuestro proyecto, éstas tenían que ser probadas de forma independiente antes de ser integradas en la aplicación. En esta sección vamos a comentar brevemente las pruebas realizadas a las principales librerías externas:

- Pruebas con Bluetooth. Se realizó una sencilla prueba consistente en la búsqueda dinámica de dispositivos disponibles en el entorno.

- Pruebas con JFreeChart. Debido a que junto a la librería se incluye también bastante documentación y ejemplos de uso, las pruebas se centraron en integrar las gráficas en la aplicación.

- Pruebas con JXTable. Las pruebas tenían como finalidad la creación de tablas de forma dinámica con distintas cabeceras, agrupando casillas comunes bajo una cabecera común.

- Pruebas con cámaras. Se realizaron programas de prueba con el objetivo de integrar las cámaras mediante los comandos CGI disponibles, mostrando el streaming de vídeo en tiempo real y controladas remotamente mediante los comandos CGI adecuados.

- Pruebas con JWebBrowser. Partiendo de varios ejemplos se consiguió disponer de un navegador web funcional integrado en la aplicación sobre el que ejecutaríamos el SIG seleccionado.

- Pruebas con Google Earth. Hemos realizado pruebas partiendo de los ejemplos proporcionados por Google, de la documentación on-line disponible y de programas previos que ya habían sido desarrollados en el grupo para integrar el plugin en el navegador creado.

5.3. Pruebas del sistema

En este apartado vamos a comentar las pruebas realizadas al sistema en su totalidad, incluyendo tanto los dispositivos NXT como la aplicación de escritorio. A la hora de la realización de dichas pruebas, destacamos dos hitos importantes, el primero de ellos coincide con las pruebas intermedias realizadas para la redacción del artículo enviado a las JISBD, mientras que el segundo de ellos coincide con las pruebas finales realizadas al final del proyecto.

Para realizar las pruebas, se ha decidido usar como caso de uso un ejemplo en concreto, las carreras de traineras que tienen lugar en la playa de La Concha en San Sebastián. Dicho caso de uso ha sido seleccionado ya que ha sido usado en un sistema desarrollado por el grupo que considera parámetros como la localización de las traineras y las cámaras de TV para ayudar a seleccionar las mejores cámaras a un realizador durante la retransmisión de la carrera [41].

Pruebas intermedias

En esta etapa se probó el sistema con cuatro robots equipados con una cámara, un GPS, una brújula, dos ultrasonidos, y un giroscopio (ver Figura 5.2(a)). El objetivo de esta prueba era, en primer lugar, comprobar que era posible controlar a los

4 robots para que siguieran una trayectoria marcada. También queríamos obtener los datos de los sensores para poder estudiar si se producían errores y en ese caso cuál era su frecuencia.

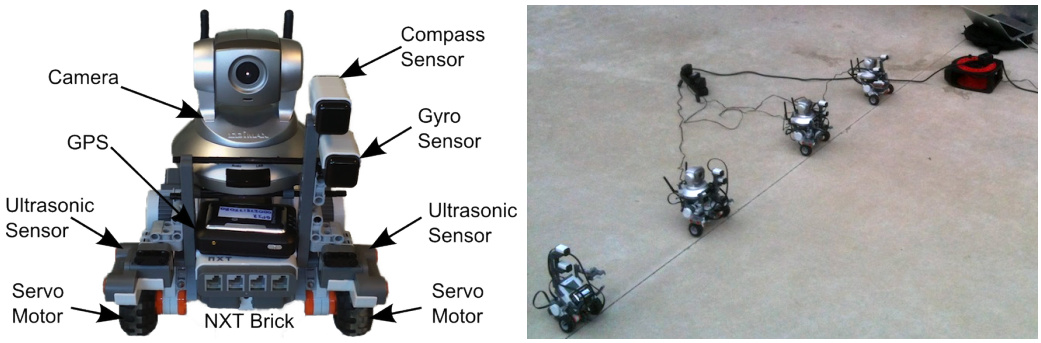

Figura 5.2: Configuración de un robot NXT (a) y escenario de prueba (b)

Como resultado de esta prueba se almacenó la información sensorial para su posterior análisis. En la Figura 5.3, se muestran todas las lecturas obtenidas por las brújulas de los robots NXT durante la simulación del escenario de las traineras. En ella, podemos ver cómo los robots NXT realizan todos prácticamente la misma trayectoria, dirigirse en línea recta hasta un punto determinado en el cual realizan un giro de 180º para regresar al punto de origen, pudiendo ver que se ha cumplido en todos los dispositivos. Otro ejemplo del análisis de los resultados obtenidos es la comprobación de la distancia existente entre dos robots (uno a continuación del otro) utilizando la información de los sensores de ultrasonidos (ver Figura 5.4). En esta gráfica podemos observar en verde la lectura del ultrasonidos izquierdo de un robot y en morado la del ultrasonidos derecho del otro robot. Puesto que estos ultrasonidos están apuntándose entre sí, los resultados que teóricamente deberíamos obtener (la distancia entre ellos) son los mismos. Sin embargo, podemos observar como aparecen algunos datos atípicos en las gráficas, posiblemente debido a errores en la lectura de los sensores puesto que la frecuencia de muestreo era de un segundo.

De la realización de estas pruebas obtuvimos además información útil para corregir fallos en la aplicación y añadir nuevas funcionalidades. Por ejemplo, comprobamos que en función del diseño realizado de los robots (número de ruedas, distancia entre ellas, etc.) el seguimiento de la trayectoria, que en este caso constaba de dos tramos de línea recta, no era del todo correcto. Por lo tanto, se decidió incluir un nuevo método de control de los robots en el cual colocando un sensor de color en la parte inferior y pintando una línea negra en el suelo, los robots corrigen su trayectoria en caso de desviarse para seguir esa línea.

Pruebas finales

Para las pruebas finales, se ha decidido usar el mismo caso de uso que en las anteriores pruebas realizadas. Las pruebas finales tuvieron lugar casi al final de la

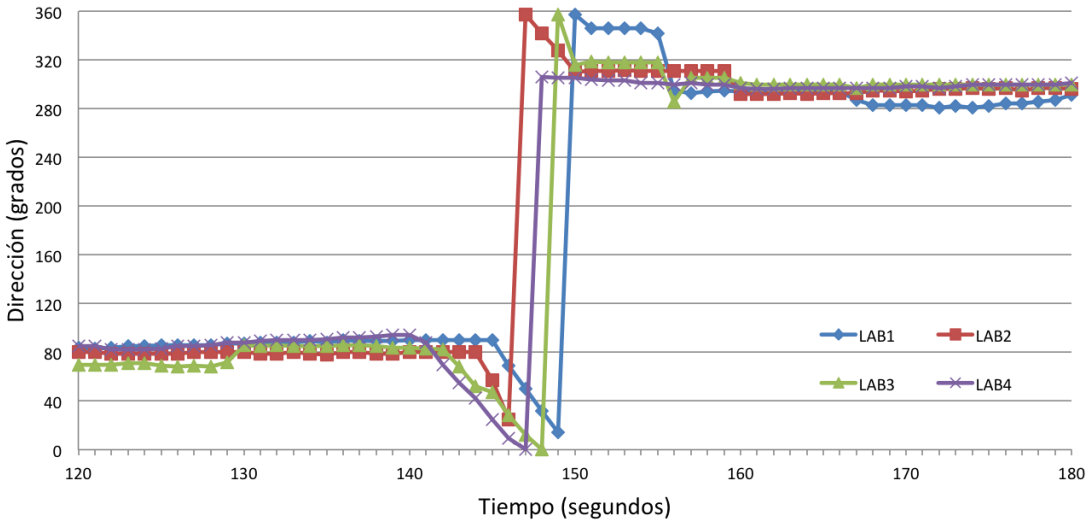

Figura 5.3: Lecturas de los sensores brújula de todos los dispositivos NXT

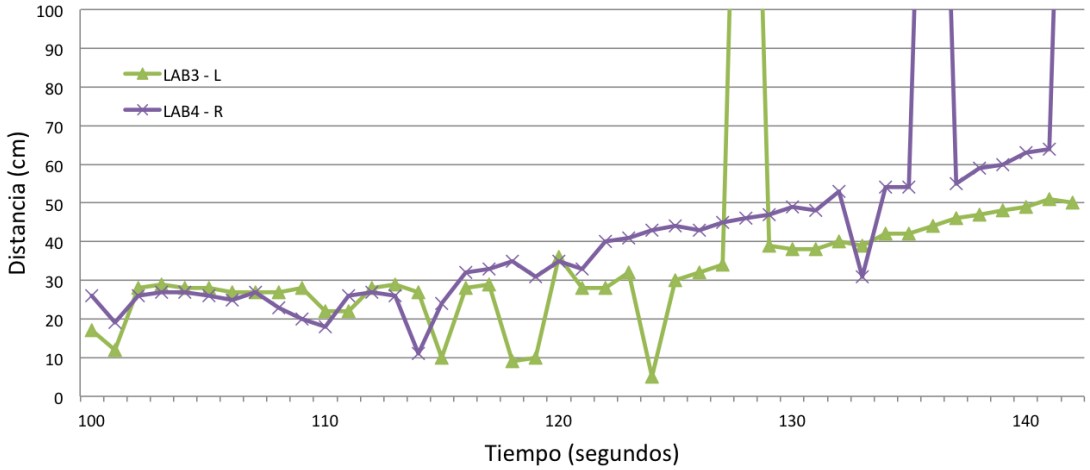

Figura 5.4: Lecturas de los sensores de ultrasonidos de dos dispositivos NXT

etapa de desarrollo de la aplicación de escritorio, ya que durante la realización de las mismas todavía podían aparecer determinados fallos en el programa que sólo podían ser detectados durante la ejecución de una simulación. Tras una serie de pruebas previas en interior para verificar el correcto funcionamiento de todos los dispositivos, procedimos a realizar las pruebas en el exterior con todos los sensores, incluyendo el sensor GPS. Como primera prueba usamos un único dispositivo NXT (ver Figura 5.5(a)) previamente emparejado con el ordenador y, tras ser detectado desde la aplicación, se procedió a configurarlo remotamente. Se seleccionaron en la aplicación los sensores conectados, indicando el puerto correspondiente y la frecuencia de envío de los datos y también se seleccionó una de las opciones de movimiento disponibles

(control mediante teclado). Tras realizar la prueba, se exportó a un fichero KML las trazas de las trayectorias seguidas para poder cargarlo en el programa Google Earth y comprobar si las trazas GPS obtenidas en Zaragoza habían sido correctamente trasladadas a la Playa de La Concha, en San Sebastián (ver Figura 5.5(b)).

Figura 5.5: Robot NXT (a) y trayectoria obtenida (b)

Para una segunda prueba con dos NXT, se configuraron los robots para seguir una trayectoria similar a la que realizan las traineras definida mediante su dibujo en la propia aplicación. Una vez completada la simulación, se exportó el fichero KML de nuevo a Google Earth para observar las trayectorias realizadas (ver Figura 5.6). Nótese como a pesar de que la trayectoria es la misma, los errores GPS y el hecho de que los robots tenían una configuración física diferentes hace que el resultado no sea exactamente igual.

Figura 5.6: Prueba con dos robots siguiendo la misma trayectoria

También probamos a realizar una captura de datos desde los sensores incluyendo distintas configuraciones en un escenario en el que estaban implicados hasta 4 dispositivos NXT. La finalidad de la prueba consistió en comprobar la perfecta comunicación de la aplicación de escritorio con la totalidad de dispositivos que pueden

estar implicados en el caso de uso en el que nos centraremos para la prueba final. En la Figura 5.7 se puede ver una captura de la aplicación en la que los dispositivos son representados en el mapa, los datos enviados aparecen reflejados en la tabla de datos y la gráfica asociada al sensor seleccionado contiene un histórico de los datos enviados desde el comienzo de la prueba.

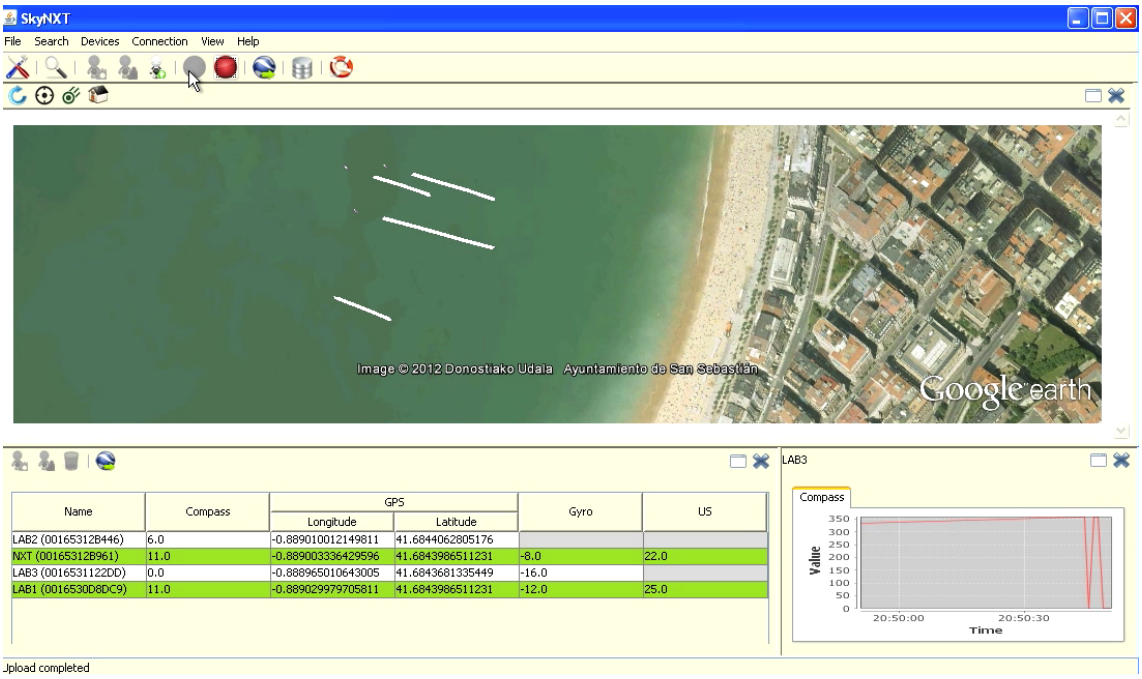

Figura 5.7: Prueba del escenario con 4 dispositivos NXT

La última prueba muestra el escenario en el que se han desplegado 4 dispositivos con una serie de sensores configurados. Podemos ver el escenario con los 4 dispositivos NXT situados en posición en la Figura 5.8 y varias capturas generadas a lo largo de la prueba con distintas vistas de las trayectorias que están recorriendo los dispositivos ya mencionados (ver Figura 5.9).

Figura 5.8: Robots NXT para prueba

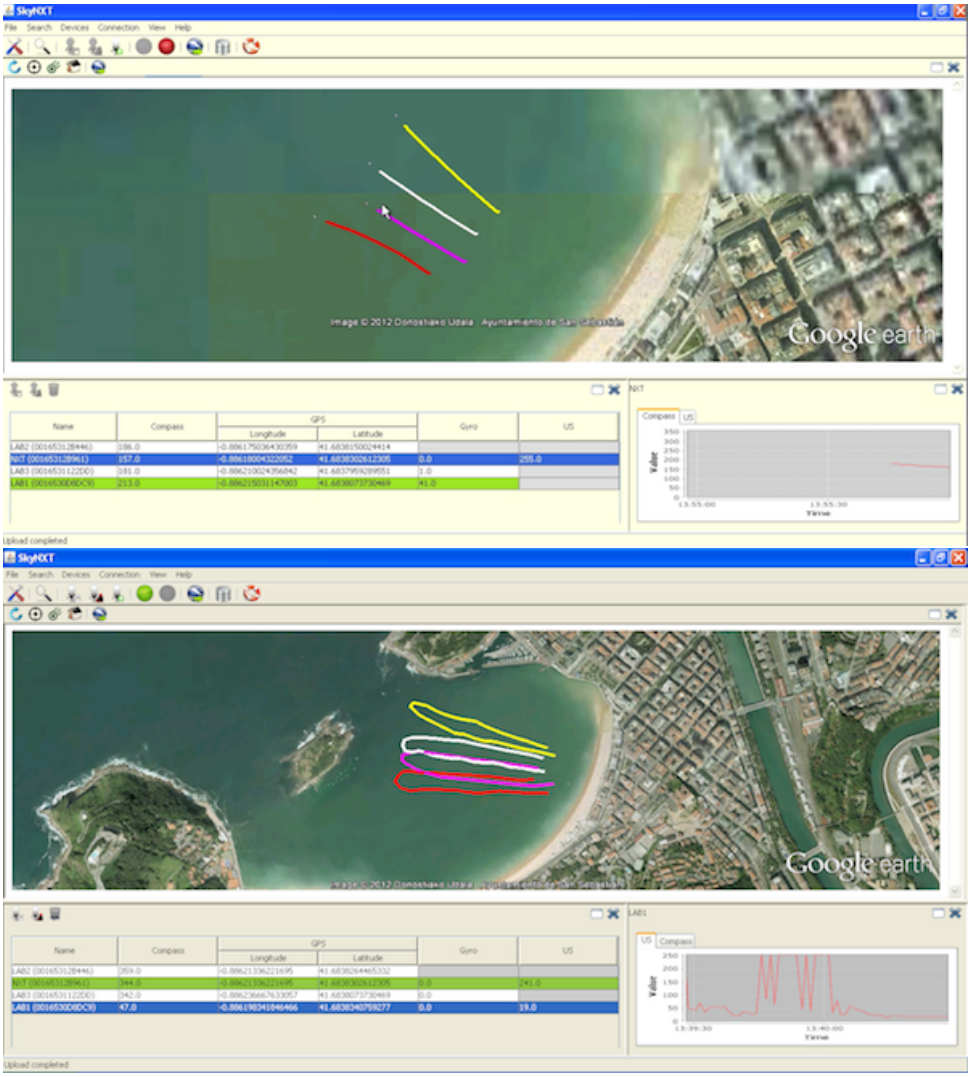

Figura 5.9: Distintas capturas de una misma simulación

Capítulo 6

Conclusiones

Como hemos visto, las simulaciones híbridas pueden ser de gran ayuda a la hora de probar sistemas de computación móvil o de acceso a datos sensoriales ambientales de forma remota. A lo largo de esta memoria hemos presentado nuestra propuesta de plataforma para la realización de simulaciones híbridas en entornos de computación móvil. Más en detalle, esta plataforma permite:

- Buscar dinámicamente vía Bluetooth dispositivos Lego Mindstorms que están dentro del radio de visibilidad de la estación base.
- Añadir a la aplicación los sensores y cámaras que se podrán configurar en los dispositivos NXT, configurando los parámetros de cada uno.
- Configurar los dispositivos detectados con sensores y motores seleccionando aquellos que están cargados en el sistema y definiendo parámetros tales como la frecuencia de lectura de información o el puerto al que se han conectado.
- Definir remotamente los movimientos de los dispositivos teniendo disponibles opciones tales como: 1) el control manual de los dispositivos mediante la pulsación de teclas definidas con anterioridad, 2) la realización de movimientos de acuerdo a trayectorias previamente definidas por el usuario desde la aplicación o 3) el seguimiento de líneas definidas sobre el escenario donde realizaremos las pruebas, de forma que puedan recrear de la forma más precisa los movimientos que realizarían los objetos que queremos representar en una situación real.
- Representar gráficamente la información que los dispositivos recopilan de los sensores y envían de forma inalámbrica a la aplicación (tanto el último dato enviado como un histórico de los mismos). Dependiendo de la configuración de los sensores, se podrán emplear gráficas para mostrar la información al usuario, además de usar tablas para indicar cual ha sido el último dato enviado por cada uno de los sensores de un dispositivo determinado.

- Representar sobre un mapa (Google Earth) la posición de los dispositivos en el escenario que queremos simular, así como las trayectorias que han realizado en el caso de dotar de movimiento a los dispositivos. El usuario puede interactuar con el mapa, pudiendo realizar operaciones tales como zoom, movimiento de la zona visible, posicionamiento automático sobre los dispositivos simulados y seguimiento automático de los mismos.
- Mostrar en tiempo real el vídeo que los dispositivos retransmiten desde las cámaras que tienen configuradas, pudiendo controlarlas remotamente desde la aplicación, además de permitir cambiar la vista de las cámaras por una simulación virtual de la misma (utilizando Google Earth).
- Almacenar los datos obtenidos por cada uno de los dispositivos móviles en bases de datos externas gracias a la cual podremos recuperar posteriormente dicha información para ser analizada con detenimiento tras la realización de las pruebas.
- Permitir exportar a ficheros externos tanto las configuraciones de los dispositivos y de la aplicación como los movimientos que realizarán los dispositivos para poder usarse en pruebas posteriores. Además se permite exportar también a un fichero externo las trayectorias obtenidas durante una prueba para que puedan ser cargadas en una aplicación externa (como por ejemplo Google Earth) en caso de necesitar analizar con detalle los movimientos de la prueba.
- Añadir nuevas funcionalidades de forma sencilla tanta a la aplicación de escritorio como a la aplicación de los robots debido al desarrollo modular llevado a cabo.

En resumen, se han conseguido implementar todos los requisitos que se habían propuesto, además de otras características que han ido surgiendo durante el desarrollo del proyecto. Además, se ha publicado un artículo en una importante conferencia nacional (Jornadas de Ingeniería del Software y Bases de Datos), compartiendo el sistema desarrollado con la comunidad investigadora y haciendo posible la obtención de comentarios al respecto y posibles mejoras.

6.1. Problemas afrontados

Durante el desarrollo del proyecto han surgido algunos problemas que han sido resueltos satisfactoriamente. Por ejemplo, en la fase de análisis de los distintos sensores, mientras se realizaban pruebas individuales a cada uno de ellos, se encontraron ciertos *bugs* en la comunicación con dispositivos GPS Bluetooth y el tratamiento de los datos de los mismos, comprensible al estar el firmware en fase beta. Tras reportar los errores, se puso a disposición de la comunidad una nueva versión del firmware que corregía estos fallos además de otros previos y se dio soporte a nuevos sensores.

Otro ejemplo es que en un principio se consideró que trazando una trayectoria en línea recta los robots la seguirían adecuadamente. Sin embargo, durante las pruebas se comprobó que dependiendo de la estructura de robot que se montaba (número de ruedas y configuración), los robots terminaban por no seguir la trayectoria correctamente. La solución fue implementar otro método de control de los robots en el que utilizando un sensor de color los robots siguen las líneas que pintemos en el suelo. También la utilización de librerías externas ha conllevado algún problema (puesto que algunas de ellas son todavía versiones no completas).

6.2. Marco temporal

El PFC se ha desarrollado a lo largo de 9 meses con una dedicación parcial puesto que se ha compaginado con el trabajo. El diagrama de Gantt de la Figura 6.1 muestra la distribución temporal del proyecto dividido en cada una de sus fases, pudiendo ver cómo la fase de implementación ha requerido gran parte del tiempo del proyecto. Esto es debido a que se ha tenido que desarrollar una aplicación de escritorio y otra para los dispositivos móviles. La fase de documentación incluye tanto las horas invertidas para la elaboración de la memoria como para la elaboración del artículo escrito durante el desarrollo del proyecto. Además, en el gráfico de la Figura 6.2 mostramos la distribución de las horas invertidas en cada una de las fases en porcentajes con respecto a las *750 horas* invertidas en total en el proyecto.

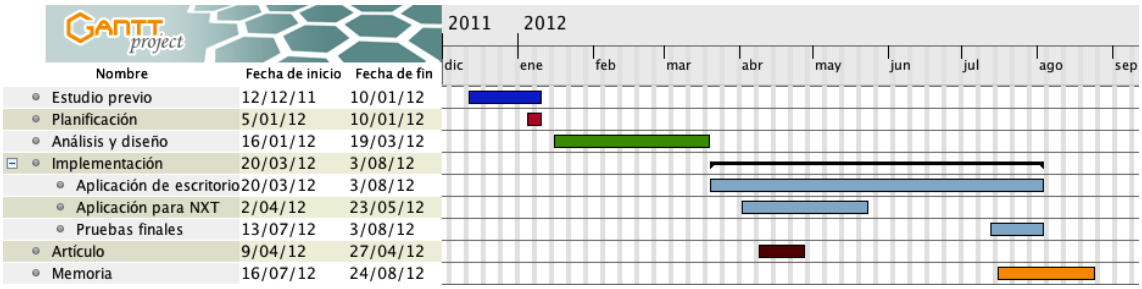

Figura 6.1: Diagrama de Gantt del desarrollo del proyecto

6.3. Trabajo futuro

En cuanto a líneas de trabajo futuro, es posible destacar varias mejoras y líneas de actuación:

- Localizar en interiores los dispositivos móviles haciendo uso de sensores varios y técnicas de localización basadas en el movimiento, o en sustitución de los sensores de posicionamiento GPS usados para la localización en exteriores.

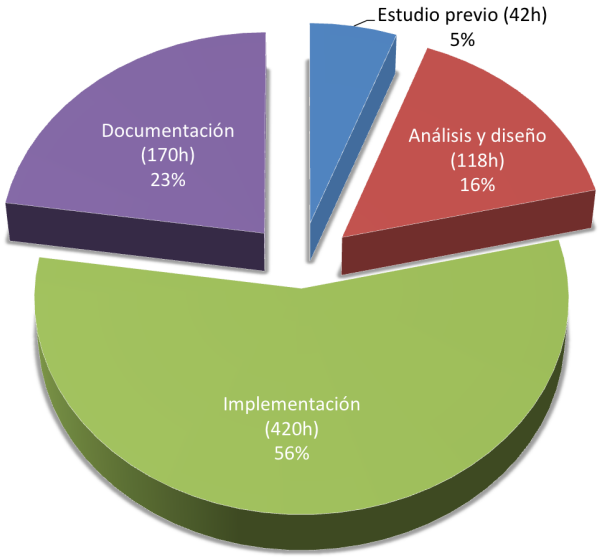

Figura 6.2: Distribución gráfica de las horas

- El uso de otros mecanismos de comunicación distintos al empleado (ZigBee, Wi-Fi, etc.). Ya se han analizado las posibles alternativas al Bluetooth, por lo que una línea de trabajo futuro podría ser el adaptar los programas desarrollados con dichos mecanismos de comunicación.
- Adaptar la plataforma al uso de otros dispositivos móviles. Al igual que los mecanismos de comunicación, se han analizado otras alternativas al dispositivo NXT y otra línea de trabajo futuro sería el poder configurar y usar dichos dispositivos.
- El uso de tablets o smartphones en la aplicación. Debido a que los tablets son dispositivos cada vez más populares, una posible línea de trabajo futuro sería la de poder controlar remotamente los dispositivos móviles mediante una tablet o smartphone y recibir en ellos la información de los mismos.

Gracias a que se tuvo en cuenta el realizar un diseño modular de la aplicación y se ha documentado el código desarrollado para futuros desarrolladores, el realizar estas modificaciones resultará una tarea sencilla.

6.4. Valoración personal

La realización de este proyecto me ha resultado bastante positiva y he de agradecer el apoyo y consejos aportados por los directores a lo largo del mismo, así como la ayuda recibida para solventar las dudas surgidas durante el proyecto.

El desarrollo de este proyecto me ha aportado tanto conocimientos técnicos como perfeccionamiento de mis habilidades como programador, además de haber tenido la oportunidad de poder usar diferentes tecnologías nuevas para mi. Desde el punto de vista técnico, el haber programado para los dispositivos NXT ha sido una experiencia bastante positiva ya que he tenido que tener en cuenta sus limitaciones y he podido trabajar con diferentes sensores, lo cual es bastante útil ya que actualmente dichos sensores se incluyen en smartphones y cada vez más aplicaciones hacen uso de ellos. El haber usado también tecnologías como Google Earth y dispositivos GPS, y emplear comunicaciones Bluetooth entre varios dispositivos también eran temas que tenía ganas de tratar y me han servido para ampliar mis conocimientos con respecto a ellos. Además, el haber desarrollado el proyecto en Java me ha servido para asentar y ampliar mis conocimientos con respecto a dicho lenguaje de programación y la programación orientada a objetos en general, ya que, aunque se usa durante la carrera en varias asignaturas, el tamaño y duración del proyecto a desarrollar requiere de cualidades como saber consultar adecuadamente la documentación, conseguir adaptar diferentes clases y que funcionen adecuadamente y gestionar un proyecto de software de tamaño medio, son cualidades que podrán ayudarme en un futuro cercano.

También quiero agradecer la posibilidad de haber podido escribir un artículo relacionado con el proyecto y que haya sido aceptado en las Jornadas de Ingeniería del Software y Bases de Datos (JISBD). Nunca me había planteado el redactar un texto de semejantes características, pero el haberlo hecho me ha ayudado a comprender la complejidad asociada a la correcta redacción de los mismos (en especial si el texto es en otro idioma como ha sido el caso), a saber destacar las partes más importantes, a revisar varias veces lo que se ha escrito y a experimentar la tranquilidad de entregar el manuscrito y esperar ansiosamente los resultados de la revisión.

En cuanto al ámbito personal, quiero valorar gratamente la experiencia y agradecer el tiempo que han invertido los directores en las dudas y preguntas que me hayan podido surgir a lo largo de estos meses. He aprendido también que es bastante importante una buena planificación del tiempo, máxime cuando la dedicación al proyecto es parcial y se ha de invertir el resto de las horas disponibles en el mundo laboral, aspecto que los directores han sabido comprender.

En definitiva, la experiencia ha sido muy enriquecedora y gratificante y la realización de este proyecto me ha servido para completar mi formación en cuanto a aptitudes y conocimientos y a mejorar profesionalmente como ingeniero.

Bibliografía

[1] LaTeX - http://www.latex-project.org/. Última consulta 30/08/2012.

[2] MiKTeX - http://miktex.org/. Última consulta 30/08/2012.

[3] LyX - http://www.lyx.org/. Última consulta 30/08/2012.

[4] Arduino - http://arduino.cc/es/. Última consulta 30/08/2012.

[5] Lego Mindstorms - http://mindstorms.lego.com/en-us/Default.aspx. Última consulta 30/08/2012.

[6] A. Kahrimanovic, "Remote Control Software Application", Dublin Institute of Technology, 2008.

[7] Aplicación software "Vehicle Remote Software" - http://www.norgesgade14.dk/bluetoothremote.php. Última consulta 30/08/2012.

[8] D. Jonsson , P. Babu, S. Zubair, C. Veibäck and D. Iich, "CoLEGO Track", Uppsala Universitet, 2010.

[9] D. Pellicer, "Robótica móvil con Lego Mindstorm", Universidad de Zaragoza, 2010.

[10] Sensor NXTBee, Dexter Industries- http://dexterindustries.com/NXTBee.html. Última consulta 30/08/2012.

[11] ZigBee Alliance - http://www.zigbee.org/Home.aspx. Última consulta 30/08/2012.

[12] Bluetooth - https://www.bluetooth.org/apps/content/. Última consulta 30/08/2012.

[13] Wi-Fi - http://www.wi-fi.org/. Última consulta 30/08/2012.

[14] leJOS - http://leJOS.sourceforge.net/. Última consulta 30/08/2012.

[15] RobotC - http://www.robotc.net/. Última consulta 30/08/2012.

[16] JFreeChart - http://www.jfree.org/jfreechart/. Última consulta 30/08/2012.

[17] JWebBrowser (Native Swing) - http://djproject.sourceforge.net/ns/index.html. Última consulta 30/08/2012.

[18] Bluecove - http://code.google.com/p/bluecove/. Última consulta 30/08/2012.

[19] Eclipse - http://www.eclipse.org/. Última consulta 30/08/2012.

[20] Inkscape - http://inkscape.org/. Última consulta 30/08/2012.

[21] Dia - http://dia-installer.de/index_en.html. Última consulta 30/08/2012.

[22] Gantt Project - http://www.ganttproject.biz/. Última consulta 30/08/2012.

[23] Java Developers Tools (Google Developers) - https://developers.google.com/java-dev-tools/. Última consulta 30/08/2012.

[24] NXC - http://bricxcc.sourceforge.net/nbc/. Última consulta 30/08/2012.

[25] Aplicación XAMPP - http://www.apachefriends.org/en/index.html. Última consulta 30/08/2012.

[26] Apache - http://httpd.apache.org/. Última consulta 30/08/2012.

[27] Google Earth plugin - http://www.google.com/earth/explore/products/plugin.html. Última consulta 30/08/2012.

[28] XVII Jornadas de Ingeniería del Software y Bases de Datos (JISBD) - http://sistedes2012.ual.es/sistedes/jisbd. Última consulta 30/08/2012.

[29] R. Yus, D. Anton, E. Mena, S. Ilarri and A. Illarramendi, "MultiCAMBA: A System to Assist in the Broadcasting of Sport Events", Eighth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2011), Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering (LNICST), Springer, volume 104, pp. 238-242, 2011.

[30] B. Bagnall. "Intelligence Unleashed: Creating LEGO NXT Robots With Java". Variant Press, 2011.

[31] P. De, A. Raniwala, S. Sharma, and T. Chiueh. "Mint: a miniaturized network testbed for mobile wireless research". In INFOCOM 2005. 24th Annual Joint Con- ference of the IEEE Computer and Communications Societies, volume 4, pages 2731 – 2742 vol. 4, March 2005.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] A. W. Schueller. “Programming with Robots”. Disponible online: `http://carrot.whitman.edu/Robots/notes.pdf`. Última consulta 18/08/2012.

[37] S. H. Kim y 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.

[38] D. Baum. “Dave Baum’s Definitive Guide to LEGO Mindstorms”. APress L. P., 1st edition, 1999.

[39] 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.

[40] F. Klassner and S. Anderson. “LEGO Mindstorms: not just for k-12 anymore”. Robotics Automation Magazine, IEEE, 10(2):12 – 18, June 2003.

[41] S. Ilarri, E. Mena, A. Illarramendi, R. Yus, M. Laka and Gorka Marcos, “A Friendly Location-Aware System to Facilitate the Work of Technical Directors When Broadcasting Sport Events“, Mobile Information Systems, ISSN 1574-017X, volume 8, number 1, pp. 17-43, IOS Press, February, 2012.