UNIVERSIDAD CARLOS III DE MADRID

Departamento de Ingeniería de Sistemas y Automática

TESIS DOCTORAL

INTERACCIÓN REMOTA CON ROBOTS MÓVILES BASADA EN INTERNET

Autor

Alaa Mohamed Khamis Rashwan

Ingeniero Industrial

Directores

Francisco José Rodríguez Urbano

Doctor Ingeniero Industrial

Miguel Ángel Salichs

Doctor Ingeniero Industrial

Madrid, 2003

TESIS DOCTORAL

INTERACCIÓN REMOTA CON ROBOTS MÓVILES BASADA EN INTERNET

Autor: **Alaa Mohamed Khamis Rashwan**

Directores: **Dr. Francisco José Rodríguez Urbano**
Dr. Miguel Ángel Salichs

Tribunal Calificador:

Presidente: D. Sebastián Dormido Becomo

Vocales: D. Carlos Balaguer Bernaldo de Quiros
D. Fernando Torres Medina
D. José María Sebastián y Zúñiga

Vocal Secretario: D. José Maria Armengol Moreno

Calificación: Sobresaliente cum laude por unanimidad

Leganés, 14 de Noviembre de 2003

A la memoria de mi madre
A mi esposa Nermein y a mi hija Renad
A toda mi familia y a todos mis amigos

Agradecimiento

Quisiera expresar aquí mi gratitud a todas aquellas personas que de un modo u otro me han facilitado el camino para la realización de este trabajo.

En primer lugar a mis directores de tesis Miguel Ángel Salichs y Francisco José Rodríguez Urbano, con quienes de alguna manera inicié mi andadura en la investigación sobre la robótica móvil. Muchas gracias por su inestimable ayuda y buenos consejos durante la realización de esta tesis.

También quiero mostrar mi agradecimiento a Carlos Balaguer y Luis Moreno por su gran apoyo. Agradecemos el soporte y la financiación de este trabajo y de las estancias realizadas en el extranjero al gobierno Español y la embajada de Egipto en Madrid, la Universidad Carlos III de Madrid, los proyectos DIP2002-188 de la CICYT, y el IECAT de la Comisión Europea.

También quisiera expresar mi gratitud a los amigos y compañeros de departamento quienes me ayudaron en la fase final de esta tesis: Ramiro, Dulce Milagro, Maria Jesús, Beatriz, Maria de los Ángeles, Ramón, Mohamed y Alberto.

A los amigos y compañeros que son, o han sido miembros del departamento: Antonio, Dolores, Santiago, Arturo, José María, José Manuel, Vicente, Mario I, Cristina I, Salah, Verónica, Ángela, Carlos, Cristina II, Mario II, Eva, Eladio, Samuel, Elena, Dmitry, Raquel I, Raquel II, Javier, Julio Cesar, Pavel, Marcelo, Felipe, Juan Manuel, Raúl, Ramiro II, Luis Maria y Diego.

También quiero agradecer a mis proyectantes espacialmente a Jorge Flores, Javier Pastor y Olli Knuuttila.

También quiero mostrar mi agradecimiento a Sebastián Dormido de la UNED por invitarnos a muchas de las conferencias relacionadas con el tema de la tesis que ha organizado. Además quiero agradecer a Klaus Schilling, quien me admitió en el laboratorio de sistemas de robots autónomos de la Universidad de Ciencias Aplicadas FH-Weingarten, Ravensburg, Alemania. También a Gerard Mckee de la Universidad de Reading, Inglaterra por su confianza en mi y su gran ayuda.

Quisiera expresar aquí mi gratitud a todas aquellas personas que de un modo u otro me han ayudado durante el desarrollo de este trabajo. A nuestra familia en España, Pedro, Belén, Pedro Miguel, Vanesa, Carmen, Maria Belén y Aroeste. También a mis amigos Abdel Razak y Hassan Mustafa.

Finalmente, quiero expresar mi cariño y gratitud a mi esposa Nermien por su constante impulso a mi trabajo; y a mi hija Renad por su silenciosa cooperación durante el desarrollo de esta tesis.

Resumen

El objetivo de esta tesis es el desarrollo de una arquitectura software para un laboratorio remoto en el campo de robótica móvil para facilitar la interacción remota con robots móviles utilizando varios dispositivos de interacción.

Se ha realizado esta tesis en el marco del proyecto IECAT, que pretende desarrollar una red de laboratorios remotos entre seis universidades en Europa y EE.UU. en el campo de los sistemas autónomos y teleoperados. Este laboratorio distribuido internacional proporciona un conjunto coordinado de experimentos a los estudiantes, con recursos hardware físicamente distribuidos en varios lugares pero accesibles vía Internet, ayudando así a intercambiar recursos hardware y materiales educativos entre los participantes.

En la tesis se estudian las ventajas y los inconvenientes de utilizar la red Internet como medio de comunicación en los sistemas de interacción remota con robots móviles. Se ha realizado un estudio para evaluar el rendimiento de la comunicación entre la Universidad Carlos III de Madrid y la Universidad de Ciencias Aplicadas FH-Weingarten en Alemania, que son miembros del proyecto IECAT. Se ha hecho hincapié en los inconvenientes más destacadas de Internet como el retraso temporal, el ancho de banda y el *jitter*.

Se ha utilizado el paradigma del control supervisado en el desarrollo del sistema propuesto para disminuir el ancho de banda necesario y la sensibilidad del sistema al retraso temporal.

En la tesis se propone la utilización de los patrones de software para desarrollar interfaces hombre-robot flexibles y reutilizables. Las interfaces de usuario se han desarrollado utilizando el patrón *proxy* visual. Este patrón proporciona interfaces de usuario flexibles con relaciones de acoplamiento mínimas entre los subsistemas. La generación de la interfaz de usuario está completamente separada de los objetos de la capa de abstracción proporcionando así facilidad de reutilización y extensibilidad.

En esta tesis se estudia también el uso de los dispositivos móviles tales como las PDAs y los teléfonos móviles como elementos de interacción con el robot. Los experimentos desarrollados han demostrado que estas tecnologías mejoran el alcance y la funcionalidad de los entornos de interacción remota.

Finalmente, se propone una metodología para construir entornos educativos innovadores para la robótica móvil. Dicha metodología combina varias actividades para resolver los problemas del paradigma tradicional de instrucción y para permitir a los estudiantes tener un papel activo y creativo en el sistema.

Abstract

The objective of this thesis is to develop software architecture for building a remote laboratory in the field of mobile robotics to facilitate the remote interaction with mobile robots using varies interaction devices.

This thesis has been developed as a part of the IECAT project, which aims at developing a network of remote laboratories between six universities in Europe and United States in the field of autonomous and teleoperated systems. This international distributed laboratory provides a coordinated set of experiments for students with hardware facilities physically spread over different locations, but accessible via the Internet. It permits the exchange of existing hardware resources and educational materials between the partners.

In this thesis, the advantages and disadvantages of using the Internet as a communication medium in remote interaction systems with mobile robot are studied. A survey was conducted to evaluate the network performance between the University Carlos III in Madrid (UC3M), and the University of Applied Sciences (FH-Weingarten) in Germany, which are partners in the IECAT project. This study has focused on the most common problems of the Internet such as time delay, bandwidth and jitter.

Supervisory control paradigm has been used to develop the proposed system in order to alleviate high communication data rates and system sensitivity to network delays.

In the thesis, the use of software patterns is proposed for developing flexible and reusable human-robot interfaces. The user interfaces have been developed using visual proxy pattern. This pattern provides flexible user interfaces with minimal coupling relationships between subsystems. The generation of the user interface is entirely separated from the abstraction layer objects to provide the reusability and extensibility facilitates.

In this thesis, the use of mobile devices such as PDAs and cellular phones as interaction devices with the robot is studied. The conducted experiments through this work have shown that these technologies improve the reachability and the functionality of remote interaction environments.

Finally, an approach for building innovative educational environments for mobile robotics is proposed. This approach combines varies activities to solve the problems of the traditional paradigm of education and to permit the students to have an active and creative role in the system.

Índice

Lista de Figuras

Lista de Tablas

Capítulo

1

INTRODUCCIÓN

Capítulo 1 INTRODUCCIÓN

1.1 MOTIVACIÓN Y MARCO DE LA TESIS

Las nuevas tendencias en robótica han ayudado a desarrollar muchos sistemas de robótica, que pretenden ampliar el uso de los robots en la vida diaria. La mayoría de estas nuevas tendencias se ha hecho posible gracias a la evolución de las tecnologías telemáticas.

Internet proporciona una infraestructura global de comunicación que habilita la implementación fácil de sistemas distribuidos. Aunque la red Internet mantiene un medio de comunicación barato y disponible, existen todavía muchos problemas por resolver antes de desarrollar aplicaciones verdaderamente fiables. Estos problemas incluyen el ancho de banda limitado y el retraso de la transmisión que varía arbitrariamente e influye en el rendimiento de los sistemas basados en Internet.

Recientemente, esfuerzos considerables de investigación en el campo de robótica móvil están dirigiéndose hacia el uso de Internet como medio de comunicación para facilitar la interacción remota con los robots móviles.

La interacción hombre-robot juega un papel importante en cualquier sistema de robótica teniendo en cuenta que todavía no existe un robot con capacidad totalmente autónoma. Incluso si esta meta de la autonomía completa se alcanzara, el papel humano y el nivel de interacción variarán pero el hombre seguirá siendo una parte del sistema.

La interacción remota es un tipo especial de la interacción hombre-robot, donde el hombre y el robot están separados por barreras físicas pero se comunican vía las tecnologías telemáticas. Se puede usar este tipo de interacción en muchas aplicaciones útiles como experimentación remota, teleoperación, telepercepción, teleprogramación, etc.

La experimentación remota tiene la ventaja de proporcionar la posibilidad de compartir un experimento o varios experimentos entre varios operadores localizados en lugares distintos. De esta manera, podrían compartirse fácilmente experimentos reales entre varios laboratorios y, por lo tanto, se podrían reducir los costes.

Los laboratorios remotos se consideran como entornos innovadores que pueden usarse para facilitar la experimentación remota. Se pueden definir los laboratorios remotos como laboratorios basados en red donde el usuario y los equipos del laboratorio real están geográficamente separados y donde se usan tecnologías de la telecomunicación para dar a los usuarios acceso a los equipos del laboratorio. Dichos laboratorios tienen la ventaja de no estar restringidos a la asistencia sincronizada por instructores y estudiantes, así tienen la capacidad de proporcionar acceso constante siempre y cuando sea necesario.

Se pueden reunir varios laboratorios remotos para formar una red o un laboratorio distribuido que puede usarse para proporcionar a los usuarios un conjunto coordinado de experimentos con recursos físicamente distribuidos en diferentes lugares pero accesibles vía Internet. Tales laboratorios pretenden ser un entorno de trabajo electrónico para la colaboración y la experimentación a distancia en la investigación o en otras actividades creativas. Ayudan a intercambiar recursos de hardware y materiales educativos entre los participantes. El proyecto IECAT (*Innovative Educational Concepts for Autonomous and Teleoperated Systems* - Conceptos Educativos Innovadores para Sistemas Teleoperados y Autónomos) en lo que participa la Universidad Carlos III de Madrid es un ejemplo de un laboratorio distribuido en el campo de mecatrónica [IECAT,03].

Se ha realizado esta tesis en el marco de este proyecto, que pretende desarrollar una red de laboratorios remotos entre seis universidades en Europa y EE.UU. en el campo de los sistemas autónomos y teleoperados. Este laboratorio distribuido internacional permite a los estudiantes llevar a cabo experimentos con equipamiento real y simulado perteneciente a las universidades participantes.

Para desarrollar este tipo de laboratorios basados en Internet para la robótica móvil, existen varios requerimientos que deben tenerse en cuenta para garantizar el alto rendimiento y la funcionalidad. Entre los cuales cabe citar la infraestructura telemática que da acceso a los experimentos, así como las interfaces de usuario que proporcionan la interactividad necesaria y realimentación adecuada a los usuarios.

Interfaces intuitivas son necesarias para dar al usuario la habilidad de interactuar remotamente con el robot de manera correcta e interactiva. También las interfaces de usuario deben garantizar un envolvimiento continuo y activo del usuario en el sistema, proporcionándole realimentación suficiente sobre todas las tareas que el robot está llevando a cabo en el laboratorio real. Existe también la necesidad de facilitar la adaptación de las interfaces hacia los cambios, como la necesidad de modificar o extender las interfaces para que cubran nuevas tareas o para que se adapten con nuevos dispositivos de interacción.

Se recomienda también diseñar las interfaces basándose en componentes reutilizables, de forma que se puedan construir interfaces para nuevas tareas de una manera más rápida y eficaz. Los patrones de software ayudan a construir software basado en la reutilización. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

En esta tesis se ha querido desarrollar una arquitectura software para un laboratorio remoto de robótica móvil teniendo en cuenta todos los requisitos expuestos anteriormente.

1.2 OBJETIVOS DE LA TESIS

La presente tesis está centrada en los sistemas de interacción remota con robots móviles basados en Internet. Los objetivos principales que se pretenden conseguir en esta tesis son:

- Estudiar las ventajas y los inconvenientes de utilizar Internet como medio de comunicación en los sistemas de interacción remota con robots móviles.

- Estudiar las estrategias de control que se pueden aplicar para desarrollar sistemas de interacción remota basados en Internet.

- Estudiar el uso de los patrones de software para desarrollar interfaces hombre-robot intuitivas, extensibles y reutilizables.

- Desarrollar una arquitectura software utilizando el modelo cliente-servidor de tres capas para construir un laboratorio remoto en el campo de la robótica móvil.

- Desarrollo e implementación de interfaces para las habilidades automáticas tanto simples como complejas.

- Estudiar el uso de los dispositivos móviles tales como las PDAs y los teléfonos móviles como elementos de interacción remota con el robot.

- Proponer una metodología que se puede usar para construir entornos educativos innovadores para la robótica móvil.

- Implementar los módulos educativos de la metodología propuesta para desarrollar un curso fundamental de robótica móvil y probar el sistema desarrollado con estudiantes reales.

1.3 ESTRUCTURA DE LA TESIS

La tesis se divide en ocho capítulos, los cuales se describirán a continuación:

- En el capítulo 2 se describen las últimas tendencias en el campo de la interacción hombre-robot, planteando las diferentes relaciones entre el hombre y los robots, los niveles de control, los papeles del usuario durante el proceso de la interacción, las lecciones aprendidas de los campos relacionados, los elementos de interacción y los diferentes métodos de realimentación.

- En el capítulo 3 se discute el concepto de la interacción remota hombre-robot basada en Internet haciendo hincapié en las tareas que se pueden llevar a cabo utilizando este tipo de sistemas, en las ventajas e inconvenientes de utilizar Internet como medio de comunicación entre el sitio local y el sitio remoto y en la estrategia de control adecuada para desarrollar sistemas basados en Internet.

- En el capítulo 4 se analiza detalladamente el diseño y el funcionamiento de los laboratorios remotos con un enfoque especial sobre los laboratorios de robótica móvil. Se explica qué es un laboratorio remoto y qué es un laboratorio distribuido y para qué sirven. Se presenta el estado del arte de los laboratorios remotos y distribuidos y las distintas arquitecturas y tecnologías utilizadas para implementar estos laboratorios. Además se plantea una metodología por medio de la cual se puedan construir entornos educativos innovadores para la robótica móvil.

- El capítulo 5 se concentra en describir los patrones de software, destacando los patrones de software utilizados para desarrollar interfaces hombre-robot.

- En el capítulo 6 se realiza la descripción de la arquitectura software desarrollada en la presente tesis. Se presentarán las tres capas que forman la arquitectura propuesta y se explicará la interacción entre las capas de la misma.

- En el capítulo 7 se analiza detalladamente el diseño y la implementación de las interfaces desarrolladas utilizando la arquitectura propuesta y se describe la integración de dichas interfaces en un entorno educativo de robótica móvil.

- Por último, el capítulo 8 resume las principales aportaciones de la tesis así como las conclusiones más importantes que de ellas se derivan. También se incluye una lista de los posibles futuros desarrollos a partir de las investigaciones realizadas.

Al final de la tesis se encuentran cuatro apéndices, los cuales se describirán a continuación:

- En el apéndice A se describen las características básicas de los dispositivos móviles y las tecnologías de desarrollo disponibles para las aplicaciones inalámbricas.

- El apéndice B presenta la arquitectura híbrida de control AD (Automática-Deliberativa) utilizada en el desarrollo de la tesis.

- En el apéndice C se discuten los sistemas distribuidos y la arquitectura cliente-servidor utilizada para desarrollar laboratorios remotos. Se discuten también las tecnologías disponibles para implementar sistemas basadas en Web.

- En el apéndice D se describen el robot B21 empleado en el desarrollo de esta tesis y su *framework* Mobility.

- Finalmente, el apéndice E es una lista de los acrónimos utilizados en los capítulos de la presente tesis.

Capítulo

2

INTERACCIÓN HOMBRE-ROBOT

Capítulo 2

INTERACCIÓN HOMBRE-ROBOT

2.1 INTRODUCCIÓN

La interacción hombre-robot (IHR) juega un papel importante en cualquier sistema de robótica teniendo en cuenta que todavía no existe un robot con capacidad totalmente autónoma. Incluso si esta meta de la autonomía completa se alcanzara, el papel humano y el nivel de interacción variarán pero el hombre seguirá siendo una parte del sistema. Por ejemplo, un robot personal interactuara regular y frecuente con un humano, mientras que un robot planetario de exploración recibiría comandos supervisados únicamente en casos ocasionales.

La IHR puede definirse como el estudio de los seres humanos, los robots, y la forma en que uno influye en el otro [Fong,02]. Existe una tendencia a pensar que los robots han de volverse más "humanos", que necesitan interactuar con el ser humano (y quizás entre sí) por la manera en que los seres humanos interactúan entre sí, y que, finalmente, ellos pueden sobrepasarlos por completo. Esta metodología, a veces se denomina "robótica centrada al hombre", y da énfasis al estudio de los seres humanos como modelos para los robots, o incluso al estudio de los robots como modelos para los seres humanos [Rogers,01]. Sheridan ha mencionado que uno de los desafíos de la interacción hombre-robot es proveer a los seres humanos y a los robots modelos de uno al otro [Sheridan,97]. En particular, él defendió que la idea sería análoga a la de dos personas que se conocen bien y que pueden colaborar para llevar a cabo ciertas tareas (p. ej., dos músicos tocando un dúo). Debido a la naturaleza de la inteligencia necesitada por los robots modernos, la IHR debe combinar el sueño de inteligencia artificial (IA) de crear un ser inteligente, criatura, o programa con la visión de la interacción persona-ordenador (IPO) de construir sistemas que se centran alrededor de las necesidades y los deseos del usuario.

En este capítulo se describen las últimas tendencias en el campo de la interacción hombre-robot. En primer lugar, se describen las relaciones entre el hombre y los robots desde diferentes puntos de vista como la familiaridad con la robótica, las diferentes taxonomías de relaciones y los

niveles de control y los papeles de usuario. A continuación, se intenta contestar la pregunta "¿cómo podrían interactuar los seres humanos con los robots"? discutiendo lecciones aprendidas de las disciplinas relacionadas como la IPO, automatización, ciencia de ficción y ciencia cognoscitiva, etología, emoción y personalidad. En la sección 2.4 se clasifican los diferentes tipos de interacción según la naturaleza de la interacción. Y en la sección 2.5 el tema de interacción hombre-robot se clasifica según la accesibilidad a interacción directa e interacción remota. A continuación, los diferentes elementos de interacción se plantean en la sección 2.6. Y finalmente los diferentes métodos de realimentación se presentan en la sección 2.7.

2.2 RELACIONES HOMBRE-ROBOT

La interacción humana y la autonomía del robot son las funciones claves que pueden ampliar el uso del robot en la vida diaria. Hoy en día, la mayoría de los robots disponibles pueden interactuar únicamente con sus creadores o con un grupo pequeño de individuos especialmente entrenados. El objetivo final de las investigaciones en robótica es desarrollar un robot que pueda interactuar con seres humanos y participar en la sociedad humana. Este tipo de robot debe tener interfaces efectivas, intuitivas y transparentes con nivel alto de autonomía por lo cual el robot será capaz de sobrevivir en situaciones diferentes.

2.2.1 Familiaridad con la Robótica

Desde el punto de vista de la familiaridad con los robots, se pueden clasificar los usuarios en distintos niveles, como se muestra en la figura 2.1.

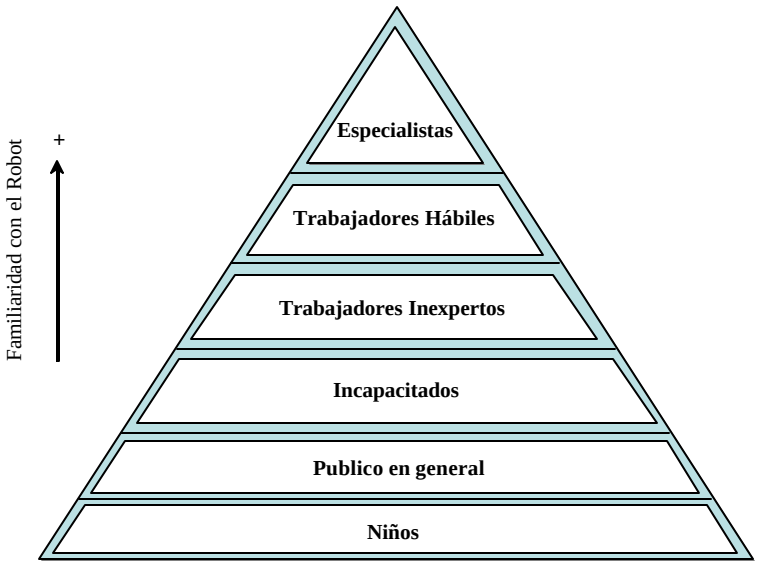


Fig. 2.1: Familiaridad con los Robots

Al nivel más alto de la familiaridad con los robots y especialización están los expertos, los investigadores en robótica, los especialistas y los técnicos en robótica. En el siguiente nivel se sitúan los trabajadores hábiles en otras áreas (poca o ninguna experiencia de robótica o informática), p. ej., los trabajadores de búsqueda y rescate, mineros, trabajadores de fabricación, etc. Estos trabajadores tienen una relación de colaboración con el robot. En el próximo nivel de

familiaridad con los robots estarían los trabajadores inexpertos, quienes pueden encontrar los robots como parte de su entrono de trabajo, aunque no trabajen directamente con los robots. Cuando encuentran el robot, tienen que establecer una relación con él. En otro nivel de usuario están los discapacitados, que tienen poca o ninguna experiencia de ordenadores o robots. En este caso los robots son robots personales de servicio que ayudan a los discapacitados hacer varias tareas. En el próximo nivel estaría el público o la gente normal que tiene poca o ninguna experiencia de informática o robótica y que usa los robots personales de servicio como aparatos de limpieza, cortacéspedes, etc. En nivel más baja de capacidad existen los niños que operan robots de entretenimiento.

2.2.2 Taxonomías de Relaciones

Los seres humanos pueden jugar distintos papeles en el sistema con diferente limitación de uso y autoridades. Rogers et al. han clasificado la relación hombre-robot basándose en tres taxonomías: relación numérica, relación espacial, y relaciones de autoridad [Rogers,01].

- **Relaciones Numéricas**

La relación numérica determina la escalabilidad del sistema como se muestra en la tabla 2.1.

Tabla 2.1: Relación Numérica

Hombres	Robots
un usuario	un robot
un usuario	varios robots
varios usuarios	un robot
grupos de usuarios	grupos de robots

Un tema importante que afecta a la escalabilidad de una metodología es cómo se integran los usuarios con el sistema durante la interacción. Esto involucra tanto el acceso como el control de los recursos del sistema. Los sistemas de un usuario único pueden permitir acceso total a sus recursos sin ninguna restricción. Agregar más usuarios, significa que estos recursos tienen que estar compartidos y por lo tanto algunos tipos de interacción pueden estar perjudicados o prohibidos, particularmente aquéllos que requieren un ancho de banda alto y especializado.

Se puede usar la metodología de *round robin* para manejar el acceso de los usuarios como se explica en el capítulo 6. Cuando sólo haya un usuario conectado, no se da ninguna restricción en el uso. Sin embargo, si se conectan varios usuarios simultáneamente, sólo uno tendrá el acceso durante una cantidad especificada de tiempo. Una vez el tiempo haya expirado, el próximo usuario en línea tendrá la oportunidad de controlar el sistema y el primer usuario se moverá al extremo de la lista de espera. Este mecanismo tiene una limitación en caso de que el número de usuarios crezca [Khamis,02]. Existen situaciones donde se necesitan más de un robot para realizar una tarea asignada. La interacción de los seres humanos con sistemas multi-robot incluye sistemas robóticos distribuidos para aplicaciones prácticas, como se puede ver en [Habib,92].

- **Relaciones Espaciales**

Según la distancia entre el usuario y el robot, se puede clasificar la interacción hombre-robot en interacción directa o local e interacción remota. La sección 2.5 discute la interacción directa y remota con los robots en más detalle.

- **Relaciones de Autoridad**

Las relaciones de autoridad tienen un estilo piramidal por lo que los privilegios de control, se pueden determinar. La tabla 2.2 muestra dichas relaciones.

Tabla 2.2: Autoridad: niveles de control

Autoridad	Función
Supervisor	Comandos "qué"
Operador	Comandos "cómo"
Cooperador	Cooperación
Espectador	Interactúa

El nivel de control se refiere a la naturaleza de responsabilidad humana para el funcionamiento del robot y determina la importancia relativa y la frecuencia de las tareas realizadas por el hombre. En el apartado siguiente, se discuten los distintos niveles de control con más detalles.

2.2.3 Niveles de Control

Generalmente los papeles del usuario en su interacción con la máquina se pueden describir mediante el concepto de niveles de control [Draper,94]. El nivel de control determina la importancia relativa y la frecuencia de las tareas realizadas por el usuario. Se refiere a la naturaleza de la responsabilidad humana con la funcionalidad de la máquina y se varía de control total a control estratégico como se muestra en la figura 2.2.

Durante el control total, el usuario es responsable de todas las decisiones, desde la planificación estratégica al control de las trayectorias. Por otra parte, el usuario es sólo responsable de los planes relativamente a largo plazo; por lo menos mientras la máquina está realizando la tarea (la programación tiene lugar antes de la ejecución de la tarea). Como se muestra en la figura 2.2, los niveles de control son los siguientes:

- **Control Manual**

A este nivel de control, el usuario debe controlar el rango entero de la funcionalidad del sistema, desde el seguimiento de trayectorias hasta la planificación. La tarea de la máquina es monitorizar la información del entrono de trabajo y actuar según las entradas del usuario.

- **Control Manual con Asistencia Inteligente**

A medida que las maquinas son más inteligentes, el usuario puede enseñar la máquina información rudimentaria sobre el sitio de trabajo, como definir regiones en las que no debe entrar. La máquina puede modificar las pantallas de información para reforzar la información de video disponible y modificar las entradas del usuario para mejorar el guiado, quizás en forma de restricciones de movimiento [Vertut,85]. La teleoperación asistida por ordenador [Vertut,85] y el control compartido [Hannaford,89] se encuentran en este área. Cuando la máquina se vuelve más inteligente, el usuario puede delegar más responsabilidad en ella.

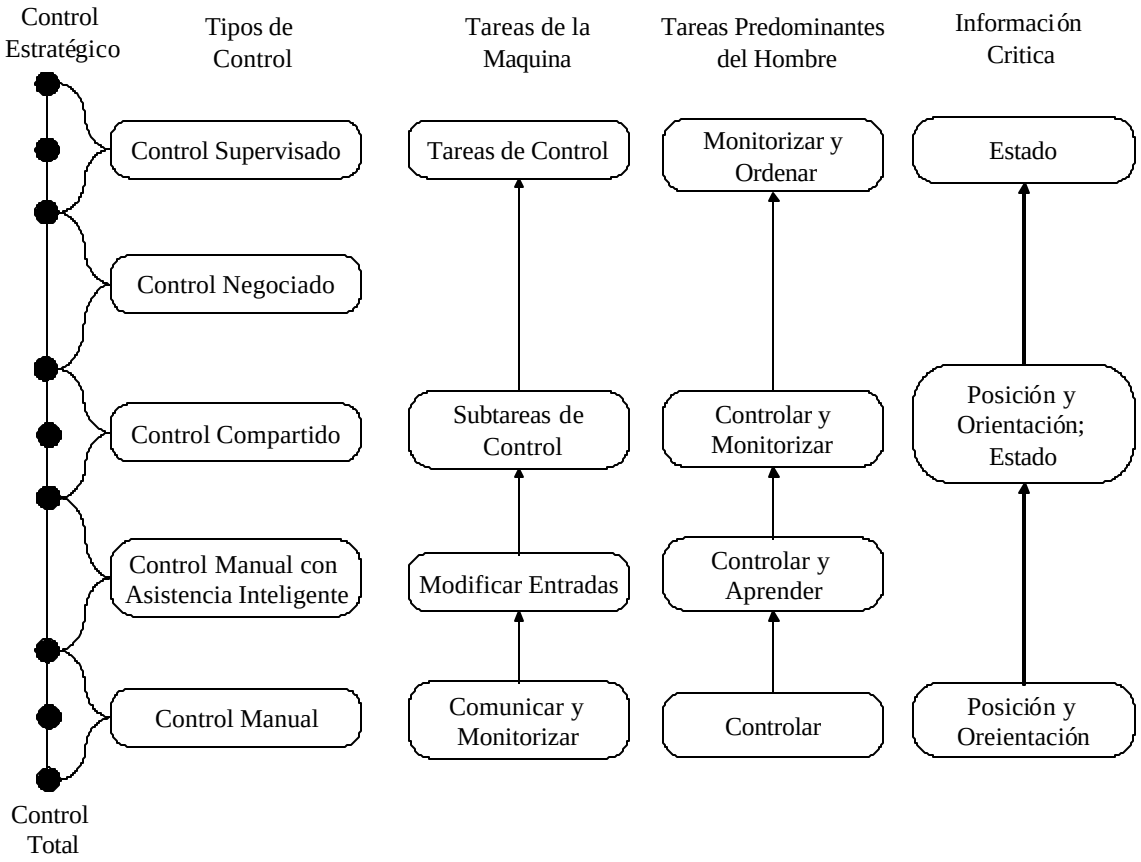


Fig. 2.2: Niveles de control con los tipos de control, las tareas primarias de la máquina y del hombre, y la información crítica para el operador humano.

- **Control Compartido**

En este nivel, el usuario es responsable de controlar algunas subtareas mientras que la máquina controla otras simultáneamente. Un ejemplo es el control humano de velocidad horizontal y el control por parte de la máquina de la orientación y la velocidad vertical de un submarino [Yoerger,90].

- **Control Negociado**

  La máquina y el hombre son consecutivamente responsables de las subtareas, es decir, algunas veces la máquina tiene control completo y a veces el hombre lo tiene. A este nivel, el usuario puede estar controlando, ordenando, o supervisando (con programación ocasional o enseñanza) dependiendo de la subtarea.

- **Control Supervisado**

El nivel siguiente, denominado control supervisado, se describió por primera vez en [Sheridan,78] y se define como un modo de control en el que uno o más de los operadores están intermitentemente programando y recibiendo información continuamente de un ordenador que controla un teleoperador [Sheridan,92]. La máquina es responsable de controlar tareas y el ser humano está supervisando interviniendo ocasionalmente para ordenar, operar o programar. La interacción humana con el sistema es simbólica, es decir, involucra la selección de tareas del teleoperador y metas pero no involucra control directo de las acciones del teleoperador. El usuario entrará en el bucle de control sólo si aparecen situaciones anormales.

La figura 2.3 ilustra la naturaleza del papel humano en el bucle de control para cada nivel de control. En esta figura, las flechas oscuras indican responsabilidad primaria, las flechas sombreadas indican responsabilidad intermitente, y las líneas finas indican monitorización o sólo modificación.

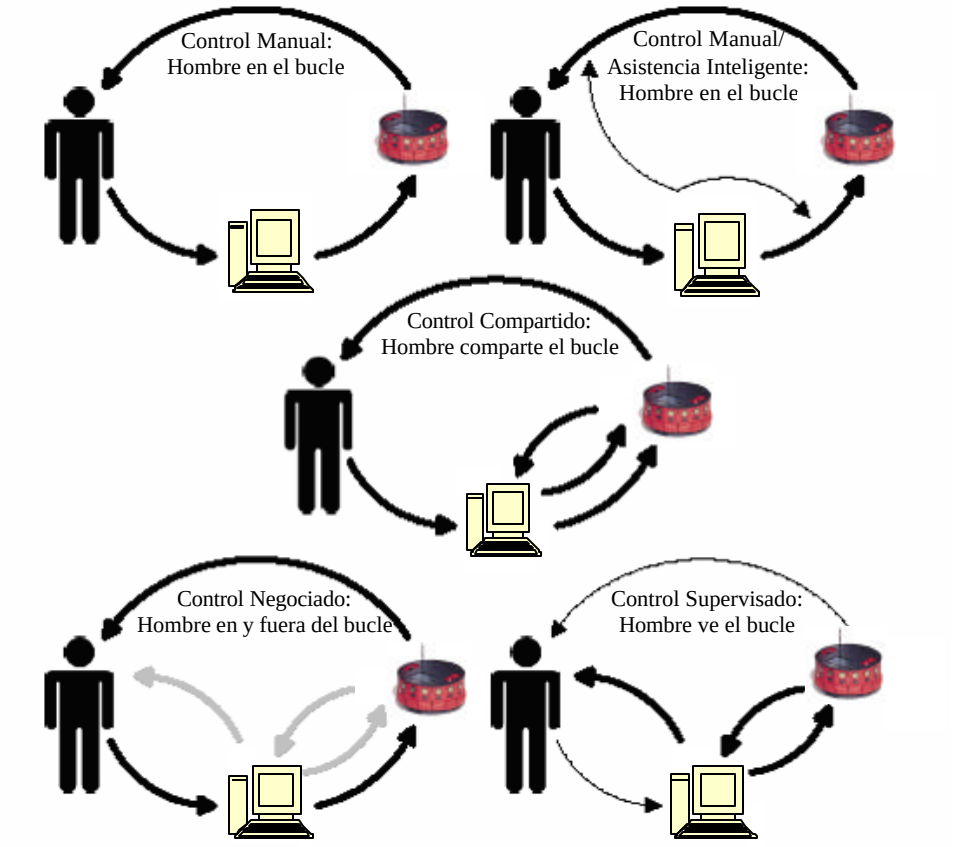


Fig. 2.3: Papeles del humano y de la maquina dentro de cada tipo de control.

Las interfaces de usuario para un teleoperador deben ser capaces de acomodar interacciones hombre-máquina a los niveles apropiados de control para el sistema. Es importante notar que los futuros teleoperadores no tendrán siempre el mismo papel, sino que este cambiará según los requisitos de la misión y la capacidad de la máquina. Sin embargo, no es verdad que un teleoperador capaz de operar a un nivel de control, sea automáticamente capaz de operar a niveles más bajos. Las interfaces del usuario deben ser bastante flexibles para acomodar el rango de tareas humanas que corresponden a los niveles de control que el teleoperador exhibirá durante una misión.

Las interfaces graficas proporcionan maneras útiles para ordenar y controlar un robot remotamente, y permiten al robot presentar información útil sobre su estado. La complejidad de las interfaces debe variar de interfaces de bajo nivel muy sofisticadas (en el caso de ingenieros de la robótica) a interfaces de alto nivel muy intuitivas y atractivas (en el caso de los robots de entretenimiento). En el capítulo 5, se discuten las interfaces hombre-robot con más detalles.

2.3 ¿CÓMO PODRÍAN INTERACTUAR LOS SERES HUMANOS CON LOS ROBOTS?

Esta sección intenta contestar la pregunta "¿cómo podrían interactuar los seres humanos con los robots?" [Rogers,01]. En el desarrollo de intentar entender cómo interactúan los seres humanos con los robots, se pueden deducir varias lecciones de las disciplinas siguientes:

- La Interacción Persona-Ordenador (IPO)
- La automatización
- La ciencia ficción
- Trabajo Cooperativo apoyado por ordenador
- La Ciencia cognoscitiva, Etología, Emoción y Personalidad

Todas estas áreas son valiosas, pero cada una está enfocando en un aspecto diferente de interacción por lo que se deben deducir lecciones de todas.

2.3.1 Interacción Persona-Ordenador (IPO)

El campo de la interacción persona-ordenador ha crecido firmemente en los años recientes. Muchas de las lecciones aprendidas en IPO deben ser útiles en la interacción hombre-robot. Por supuesto, existen diferencias particulares, entre los ordenadores y los robots. Entre estas diferencias, se encuentra la habilidad de los robots de moverse a través del mundo físico. No obstante, un descubrimiento importante de investigadores de IPO fue que el método más rápido de avanzar técnicamente era llevar a cabo nuevos tipos de interacciones y evaluar su actuación. El paradigma general en IPO es el de un humano controla y una máquina bastante simple, que recibe órdenes de la persona. La interacción centrada en el hombre y la interacción centrada en la máquina son dos metodologías en la interacción persona-ordenador.

2.2.3.1 Sistemas Centrados en el Hombre

El término centrado en el hombre se ha llamado de varias formas "centrado en el usuario", "centrado en el uso", "amigable a usuario", "centrado en el hombre ", y más recientemente, "centrado en la práctica". En tales sistemas, los siguientes principios deben lograrse [Billings,97]:

- Los operadores humanos son completamente responsables de los resultados de los procesos dirigidos por seres humanos y máquinas.
- Los seres humanos deben estar a cargo de todos los componentes de los sistemas que emprenden esos procesos. Deben tener autoridad total encima de los sistemas, es decir, que deben tener los medios para intervenir constructivamente en los procesos.
- Muchos sistemas hombre-máquina distancian al operador de los procesos en curso, algunos intencionadamente, y otros por defectos. Sin una participación activa de manera continua en el proceso, el operador humano no podrá entender el problema y tomar el control en caso de fallo de la máquina.
- Sin una buena información acerca del proceso en curso, un operador humano no puede permanecer activamente involucrado en ese proceso. En este caso, la máquina y no el hombre, es la que está al mando.
- Como las máquinas han progresado desde tareas simples de control a la gestión de información y más recientemente a la gestión de todo el proceso, se ha vuelto más difícil seguir lo que están haciendo. Esto lleva a la necesidad de informar al hombre de qué tales máquinas están funcionando debidamente, en lugar de simplemente cuando fallan.
- A menos que una máquina se comporte predeciblemente, un hombre no puede formarse un modelo interior de cómo funciona, y así no puede permanecer involucrando en el proceso.
- Los seres humanos también fallan. Las máquinas saben bastante sobre el proceso de interacción, y este conocimiento puede permitir a las máquinas supervisar la actuación humana para los errores, así como los humanos deben poder supervisar la actuación de la máquina ante errores o averías.
- Para entender qué resultado se desea, cualquier agente en un sistema hombre-máquina debe entender lo que están intentando lograr otros componentes del sistema. Esto requiere el conocimiento de las intenciones de cada uno de los agentes.

2.2.3.2 Sistemas Centrados en la Máquina

La tabla 2.3 muestra una comparación entre las características de la máquina y del hombre en los sistemas centrados en el hombre y en la máquina [Norman,93].

Tabla 2.3: Tecnología Centrada en la Maquina y Centrada en el Hombre

Vista de Sistemas Centrados en la Maquina		**Vista de Sistemas Centrados en el Hombre**	
Persona	***Maquina***	***Persona***	***Maquina***
Vaga	Precisa	Creativa	Muda
Desorganizada	Ordenada	Flexible	Rígida
Distraible	No distraible	Atenta al cambio	Insensible al cambio
Emocional	Impasible	Lista	Falta de imaginación
Ilógica	Lógica	Las decisiones son flexibles y están basados en métodos cualitativos y cuantitativos	Las decisiones son consistentes porque están basados en variables cuantitativas

Como seres humanos, nos guiamos por patrones y eventos, emociones, aprendiendo de errores pasados, razonamiento, interpretación, y comprensión. Además tenemos habilidades intelectuales y comportamiento social que no tienen las maquinas. Las maquinas pueden realizar funcionamientos complejos por orden del ser humano, y trabajar bien cuando se combinan con la inteligencia artificial. La máquina también debe proyectar una imagen de su funcionamiento. La idea es que las personas suelen desarrollar modelos interiores, mentales, y conceptuales de la manera en que el dispositivo funciona, y forman esos modelos a partir de sus expectativas y experiencia con el propio dispositivo. Por esta razón, el dispositivo debe proyectar una imagen que pueda ser útil para desarrollar esta conceptualización.

Implicaciones para los robots: En la interacción persona-maquina, es esencial tener un buen modelo de cómo opera el otro, por lo que se entiende lo que está a punto de hacer, y el progreso de lo que está haciendo. En los sistemas de ordenadores, la necesidad de realimentación es bien conocida. Sin embargo, la información exigida para obtener un modelo conceptual bueno y coherente sobre su funcionamiento no es tan bien conocida.

2.3.2 Automatización

Con la automatización tradicional, la máquina cada vez asume más funciones inicialmente realizadas por humanos. Por eso, aunque la persona podría darle inicialmente instrucciones, desde ese momento, la máquina funciona de manera relativamente autónoma. Si el comportamiento autónomo es inesperado por parte de un operador humano, se percibe a menudo como "animado"; la máquina parece tener "mente propia". El humano debe decidir si el comportamiento percibido es apropiado, o si representa un fallo de un componente del sistema. Esta decisión puede ser bastante difícil, sobre todo si el sistema no está bien documentado o no proporciona realimentación.

En muchos dominios, el uso considerable de la automatización ha simplificado el trabajo del ser humano y ha mejorado la seguridad de los sistemas. La filosofía general entre muchos diseñadores es automatizar lo más posible y dejar que el ser humano asuma al resto. Ésta es una

manera de diseñar verdaderamente mala. La manera correcta es entender totalmente las tareas a realizar y los puntos fuertes y débiles de las personas y las máquinas, y se diseña el sistema con un esfuerzo cooperativo, donde las personas hacen lo que pueden hacer mejor que las máquinas y las máquinas hacen lo que pueden hacer mejor que el hombre. La compenetración entre los dos debe ser alta y debe estar diseñada teniendo en cuenta las necesidades y las capacidades humanas.

Implicaciones para los Robots: No intentar que el robot haga una tarea para la cual no es adecuado y por tanto requiere monitorización humana continua. El humano se aburrirá que cuando el robot trabaje correctamente y no sea capaz de tomar el control en las situaciones criticas. Es mejor hacer la tarea completa, o no hacer nada, o al menos hacerla cooperativamente.

2.3.3 Ciencia Ficción

La ciencia ficción puede ser una fuente útil de ideas e información que proporciona escenarios sobre cómo un dispositivo como el robot, podría encajar dentro del trabajo y las actividades cotidianas, aunque algunos escenarios son más útiles que otros, por supuesto. Las leyes de la robótica de Asimov [Roger,93] resultan ser más pertinentes de lo que uno podría pensar. Asimov desarrolló una serie de novelas para analizar las dificultades cuando los robots autónomos poblaron la tierra. Como resultado, él postuló las tres leyes de la robótica añadiéndose posteriormente una cuarta ley, denominada la ley Cero.

Las Leyes de Asimov son algo prematuras para el estado del arte actual, pero aún así es recomendable seguirlas. De hecho, la mayoría de los robots tienen muchos aspectos claves de las leyes implementados. Por eso, aunque la ley cero está más allá de la capacidad actual, la primera ley (no dañar) se lleva a cabo parcialmente a través de los sensores de proximidad y colisión que salvaguardan a cualquier persona con la que el robot podría entrar en contacto. Así un dispositivo simple como un ascensor o la puerta de un garaje tiene sensores que inmediatamente detienen el cierre de la puerta si hay una persona. De forma análoga, los robots intentan evitar colisionar con personas u objetos. La segunda ley (seguir órdenes) también está integrada en los robots. La pregunta es ¿desobedecerá un robot la segunda ley para proteger la primera ley? quizá. En este caso, cuando un sensor descubre a una persona en su camino, ¿avanzará el robot tal y como se ha ordenado, o anulará el sensor la orden de avance?

Todavía no existe el caso de órdenes en conflicto, pero pronto aparecerán robots interactivos, donde las peticiones de un robot podrían entrar en conflicto con las peticiones de los supervisores humanos en este caso, determinar la precedencia y la prioridad será más importante. La tercera ley (proteger su propia existencia) también se integra en muchos robots. Así, los sensores para evitar caerse de escaleras u otros riesgos conocidos se integran. Además, muchos robots monitorizan el estado de energía y bien entran en modo "*sleep*" o regresan a una estación de carga cuando su nivel de energía baja mucho.

La aplicación total de las Leyes de Asimov no se puede realizar salvo que el robot tenga conciencia de su propio conocimiento, y auto conciencia de su estado, actividades, e intenciones. Puede analizar sus acciones actuales con respecto a las leyes y puede modificar las acciones cuando sea necesario.

Con los dispositivos bastante primitivos de hoy, tener algunas de estas capacidades sería útil. En casos de conflicto, podría ignorar algunos comandos. Incluso el perro Aibo de Sony (figura

2.4) tiene conciencia. Su funcionamiento está controlado por su "deseo" de jugar con su usuario, y para indicar su estado emocional, mostrado cuando está aburrido. Aibo regresa a su estación de carga cuando se le está acabando la energía, aunque los usuarios deseen jugar más con él. Esto parece anteponer la tercera ley de Asimov de autoprotegerse sobre la segunda ley de seguir órdenes.

Fig. 2.4: El Aibo de Sony

Las Leyes de Asimov, tienen ciertas suposiciones que quizás no se puedan aplicar en los sistemas actuales.

Suposición 1: Funcionamiento autónomo. Los robots de Asimov parecían ser totalmente autónomos, es decir, al darles una orden, pasarían automáticamente a su ejecución. Análogamente, los robots razonarían entre ellos y determinarían un curso de acción, que de nuevo se lleva a cabo autónomamente. Probablemente, se necesitan más robots cooperativos, sistemas en lo que hombre y robot o equipos de robots trabajan juntos. En el caso de cooperación, las leyes no tienen mucho sentido, y además necesitan estar complementadas con otras (como la necesidad de comunicación total de intenciones, estado actual, y progreso).

Suposición 2: Control central. La misma formulación de las leyes implica una estructura de control central, que siga una estructura jerárquica, priorizada. En realidad, se desarrollan sistemas de control local, con estructuras de control cooperativo e interactivo. La cognición distribuida y el control distribuido son probablemente candidatos para otros sistemas. En otras palabras, las redes neuronales y las estructuras interactivas de las células del autómata de la vida artificial en lugar de una lógica central basada en reglas. En estos casos, las leyes se vuelven a ser estructuras de pesos en lugar de la prioridización.

2.3.4 Trabajo Cooperativo apoyado por Ordenadores

Cuando se desarrollan sistemas de comportamiento cooperativo, una de las cosas más críticas es el requisito de comunicación completa y total. Así, en algunos accidentes de aviones que involucran automatización, los sistemas automatizados mantuvieron el control del avión, incluso cuando la automatización estaba acercándose al límite de su control. Cuando se llega al límite, el sistema falla de repente por lo que el piloto se da cuenta de que algo está saliendo mal, y se enfrenta de repente con un avión inestable.

Si el sistema hubiese sido más comunicativo, podría haber dicho que "El avión no está equilibrado uniformemente, pero lo estoy compensando". Entonces, después, podría haber dicho que "el desequilibrio es cada vez mayor. Quizás el piloto deba intentar determinar la causa". Y finalmente, podría haber dicho que "estoy alcanzando el extremo de mi habilidad de compensar - yo alcanzaré ese límite en 1 minuto, si la velocidad de cambio de las variables continúa." Teniendo un humano volando en el avión cuando el desequilibrio inicial se descubrió, entonces la conversación anterior probablemente habría tenido lugar entre los miembros de la tripulación [Rogers,01].

Implicaciones para los Robots: No se quieren robots demasiado habladores. Por otro lado, el robot debe supervisar sus acciones continuamente y asegurar que los humanos sepan sobre su estado y qué predicciones pueden hacerse.

2.3.5 Ciencia Cognoscitiva, Etología, Emoción y Personalidad

2.3.5.1 Ciencia Cognoscitiva

El estudio de los mecanismos que permiten a un individuo que adquiera información o habilidades de otro individuo ha sido tema de debate en muchas áreas de la ciencia cognoscitiva [Breazeal,02-a]. Hauser ha descrito el siguiente grupo de definiciones simplificadas como un ejemplo del rango de comportamientos considerados bajo el aprendizaje social entre "A" y "B" como dos individuos o subconjunto de poblaciones de individuos [Hauser,96]:

- **Imitación:** "A" aprende un comportamiento realizado por "B", que es nuevo en el repertorio de comportamientos de "A". "A" es capaz de realizar el comportamiento en la ausencia de "B".
- **Emulación de la meta:** después de observar las acciones de "B", "A" produce el mismo que "B". La forma de comportamiento de "A" es distinta que la de "B".
- **Perfeccionamiento del estímulo:** se desvía la atención de "A" a un objeto o situación como resultado del comportamiento de "B".
- **Apoyo social:** es más probable que "A" aprende el comportamiento de "B" debido a que la actuación de "B" produce un estado motivacional similar en "A".
- **Exposición:** como resultado de la asociación de "A" con "B", los dos se exponen a entornos comparables y por lo tanto adquieren comportamientos comparables.
- **Facilitación social:** un comportamiento innato de "A" parece como resultado de la actuación de "B".

La investigación en robótica se ha centrado en el aprendizaje social por muchas razones como el interés comercial en construir robots que puedan ser usados por personas normales en sus casas, su trabajo, y en espacios públicos como hospitales y museos. Este interés invoca el aprendizaje social como un mecanismo que permite a los usuarios personalizar sistemas a entornos particulares o preferencias de usuario. Por otra parte, la investigación en inteligencia artificial se ha enfocado en el aprendizaje social como posible medio para construir máquinas que pueden adquirir nuevo conocimiento autónomamente, y aumentan su complejidad y capacidad sin requerir esfuerzo adicional de los diseñadores [Breazeal,02-b].

2.3.5.2 Etología y Emoción

La etología es el estudio de los comportamientos de los animales, en especial de animales domésticos. Los estudios de las emociones en los seres humanos y su ocurrencia similar como motivacional en animales pueden proporcionar un apoyo para el desarrollo de una interacción hombre-robot eficaz [Breazeal,02-b].

La emoción humana es crítica para nuestro poder intelectual. Además, la emoción es un dispositivo comunicativo, tanto dentro de la persona como entre la gente. Así, una función de la emoción es comunicar las propiedades autónomas, subconscientes del cuerpo con la mente consciente. Así la sensación de hambre es una manera para el cuerpo para señalar a la mente la necesidad de comer. Cuando esa necesidad se hace bastante urgente, la sensación de hambre puede requerir la atención humana y llevar a la búsqueda urgente de comida. Dado que las emociones reflejan un conjunto multidimensional complejo de disparos, su reflexión es un conjunto similarmente complejo de comportamientos (por ejemplo, en el humano, las expresiones faciales, son una manera rica y sofisticada para comunicar mensajes complejos). Los robots podrían hacer igual para emular tal complejidad.

La etología y las emociones se pueden usar como base para desarrollar varias aplicaciones prometedoras como robots interactivos de entretenimiento, los robots de *edutainment* (educación y entretenimiento), los humanoides, etc.

Se ha desarrollado un robot de auto preservación para estudiar la comunicación emocional entre los seres humanos y los robots [Ogata,99]. Según Kanda et al, un robot da impresiones inteligentes, no sólo poseyendo funciones, pero también expresando las funciones [Kanda,01]. Una arquitectura software basada en la etología y la emoción ha sido propuesta por Arkin et al [Arkin,02].

Esta arquitectura se dirigió a las necesidades fundamentales propuestas por este dominio y está basada en los dos aspectos siguientes:

- La incorporación de modelos etológicos de comportamientos como una base para proporcionar a las personas la habilidad de relacionarse de manera predecible con un artefacto robotizado.

- La generación de comportamientos motivacionales (emociones por ejemplo) que apoya los conceptos humanos de las criaturas vivas, y por lo tanto anima una unión natural entre el ser humano y el robot.

Se ha usado esta arquitectura con éxito en robots de entretenimiento como el perro AIBO de Sony y el humanoide SDR que aparece en la figura 2.5.

Fig. 2.5: El Humanoide SDR de Sony

Un entorno de trabajo –altamente inspirado en el trabajo en etología, psicología, y el desarrollo cognoscitivo- se ha presentado para diseñar sistemas motivadores específicamente para los robots autónomos para regular la interacción hombre-robot [Breazeal,98]. Este sistema implementa los motores, emociones, y expresiones faciales que interactúan entre si para mantener una interacción mutuamente regulada con el ser humano a un nivel apropiado de intensidad.

2.3.5.3 Personalidad

La personalidad es una forma de modelo conceptual. Para ello encauza comportamientos, creencias, e intenciones en un comportamiento cohesivo y consistente. Ésta es una simplificación bastante drástica del campo complejo de la personalidad humana y de los muchos debates científicos que tienen lugar dentro de ese campo. Proporcionar deliberadamente una personalidad a un robot se puede dar a los seres humanos buenos modelos y un buen entendimiento del comportamiento.

Así, el robot Aibo de Sony tiene la personalidad de un cachorro que significa que si no entiende no obedece, o si intenta ejecutar una orden y falla, los mismos fracasos agregan a su experiencia que es justo como los perritos reaccionan. De forma similar, el robot Kismet de MIT (figura 2.6) tiene la personalidad de un niño y por lo tanto cometer errores y equivocaciones parece normal.

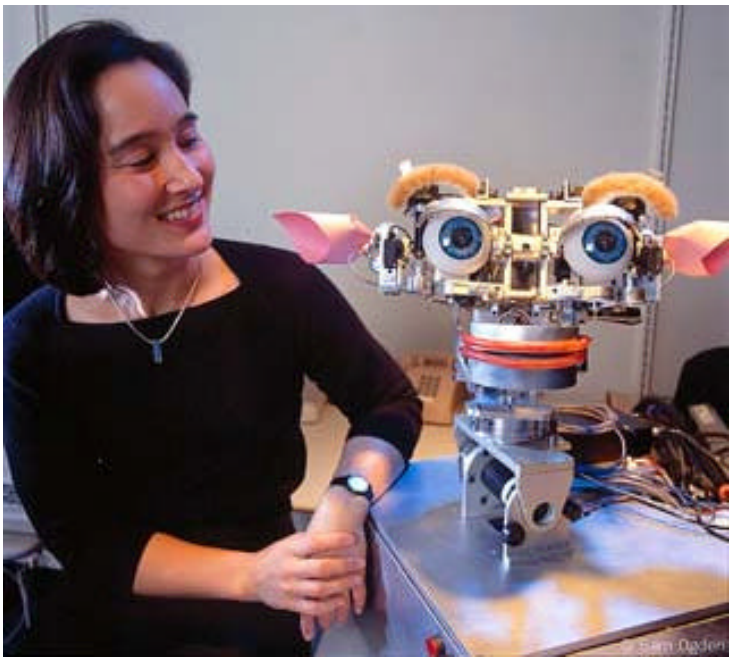


Fig. 2.6: El robot Kismet de MIT

La personalidad es una herramienta potente de diseño, porque puede proporcionar a los seres humanos un buen modelo conceptual para entender e interpretar el comportamiento del robot y para entender como deben comportarse en la interacción y en dar órdenes.

2.4 NATURALEZA DE LA INTERACCIÓN

Esta sección discute los tipos de interacción hombre-robot. Como se muestra en los apartados siguientes, estos tipos han sido clasificados basándose en la naturaleza de la interacción como interacción con un robot real, simulación, realidad virtual y realidad aumentada.

2.4.1 Interacción con un Robot Real

Para interactuar con un robot real, hay que utilizar un dispositivo de interacción por medio del cual el usuario pueda mandar sus peticiones al sistema robotizado y también hay que facilitar medios de realimentación para proveer información sobre el sistema al usuario. Los distintos elementos de interacción y medios de realimentación se discuten en la sección 2.6 y 2.7 respectivamente.

La interacción es necesaria en los sistemas modernos mecatrónicos, sobre todo en los robots, para llevar a cabo tareas específicas. En cuanto a las tareas a ejecutar, los robots pueden hacer las tareas para las cuales el diseñador los ha creado aunque los robots industriales o de servicio y personales a veces crean la impresión de actuar bastante inteligentemente. Se diseñan con ciertas capacidades de percepción, como entender el idioma y computación, y habilidades para cumplir tareas que les permita funcionar según las expectativas del usuario. Así, se pueden construir robots para entretenimiento como el perrito Aibo de Sony que básicamente se comporta como un perro, y también está entrenado para aprender el idioma [Kaplan,00]. Por otra parte, se pueden

diseñar otros robots para servicio, para la industria, para la ayuda médica, o para la ayuda a personas discapacitadas [Röfer,98].

2.4.2 Simulación

Se puede usar la simulación para proporcionar visualización gráfica y animación del movimiento de los sistemas robotizados. Recientemente, las necesidades de simuladores para robots están aumentando para proporcionar herramientas *off-line* eficaces y interfaces hombre-robots intuitivas [Tach,88][Kotoku,91]. En el modelo de simulación descrito en [Rooy,02], mediante el uso de un ojo simulado y una mano, se interactúa con una tarea teleoperada fija y simple de maniobrar en un entorno evitando otros objetos móviles. Mejorando el sistema de simulación para los componentes cinemáticas mediante interfaces hombre-máquina modernas y estableciendo una comunicación en tiempo real, llevan a un sistema de realidad virtual.

2.4.3 Realidad Virtual

La realidad virtual (RV) es una representación de las cosas a través de medios electrónicos, que nos da la sensación de estar en una situación real en la que podemos interactuar con lo que nos rodea. Es una presentación de imágenes virtuales interactivos reforzados por un proceso especial y por modalidades de presentación no visuales, como audio y hápticos, para convencer a los usuarios de que se sumerjan en un espacio sintético [Ellis,91]. También se puede considerar la RV como una interfaz hombre-ordenador de alto-nivel que permite la interacción del usuario con entornos simulados en tiempo real y a través de múltiples canales sensoriales [Burdea,94].

En los sistemas de teleoperación, la realidad virtual ofrece la capacidad de desplegar gran cantidad de información al usuario. Dicha información se organiza en cierto modo de forma que sea importante y no sea excesiva, dando al teleoperador la oportunidad de tomar decisiones rápidas, así como para acortar dramáticamente el tiempo necesario para extraer resultados y conclusiones de ciertos experimentos [Vernet,01][Schilling,99].

En los sistemas basados en Internet, la limitación del ancho de banda limita la velocidad de refrescar las imágenes de video que hace imposible experimentos en los que la escena podría variar a velocidades altas ocurriendo fallos abruptos y saltos en la visualización de los eventos [Khamis,02]. Las imágenes de realidad virtual de una simulación dada tienen necesidades mucho más bajas en cuanto al ancho de banda, por lo tanto, es una mejor alternativa a las imágenes reales de la cámara en este tipo de situaciones. En el robot TOM (TeleOperated Machine), un modelo VRML-*Virtual Reality Modeling Language* (Lenguaje para Modelado de Realidad Virtual) se ha desarrollado como se muestra en figura 2.7 a través del cual el teleoperador puede lograr una vista local del estado del experimento de manera rápida y eficaz [Khamis,02].

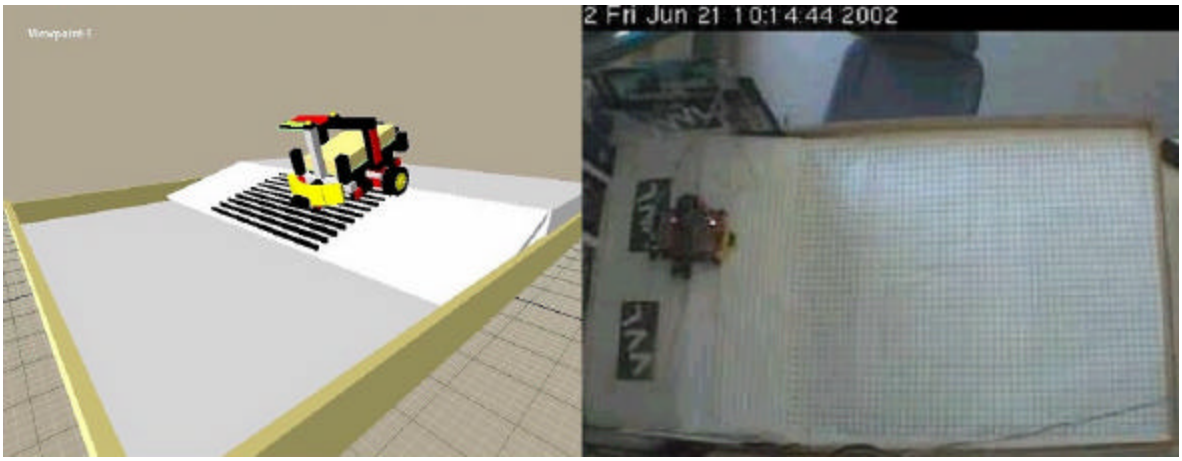


Fig. 2.7: Modelo VRML de TOM y la imagen de la Cámara

VRML provee un conjunto básico de primitivas para el modelado geométrico tridimensional y tiene la capacidad de dar comportamiento a los objetos y asignar diferentes animaciones que pueden ser activadas por eventos generados por diferentes usuarios. Existen ahora otros lenguajes de programación que tratan de desarrollar entornos de realidad virtual como X-VRML (*VRML basada en XML*), Java 3D y X3D.

Con la realidad virtual, la interfaz de usuario se puede aumentar con otra ventana que muestra la simulación virtual del experimento real, con la opción de superimposición de imágenes virtuales y las de la cámara, una técnica conocida como realidad aumentada que sirve para mostrar las diferencias entre los eventos reales y simulados.

2.4.4 Realidad Aumentada

La realidad aumentada se refiere a la combinación de lo real y lo virtual para ayudar al usuario en su entorno. Las aplicaciones incluyen la telemedicina, la arquitectura, la construcción, y muchas otras. Los objetos relacionados con el entorno o el trabajo en ejecución se pueden representar en la vista del operador. La telepresencia con esta propiedad se llama realidad aumentada [Halme,99]. Si los objetos pueden crearse interactivamente y remotamente con la ayuda de una herramienta fácil de usar, da una sensación extra y una fuente de información al operador al operar la máquina en un entorno desconocido o parcialmente conocido.

2.5 ACCESIBILIDAD

Las nuevas tendencias en la robótica y en los sistemas de automatización ayudaron a desarrollar muchos sistemas robotizados para extender las aplicaciones del robot a varias tareas útiles en la vida diaria. La mayoría de estas nuevas tendencias se ha hecho posible gracias a la evolución de los ordenadores personales (en términos de coste, potencia, y robustez) e Internet (en términos de seguridad, velocidad, y fiabilidad) [Brugali,02]. Desde el punto de vista espacial, la interacción hombre-robot se puede clasificar en interacción directa o local e interacción remota.

2.5.1 Interacción Directa

En la interacción directa, no existe ninguna barrera física entre el ser humano y el robot. Se suele aplicar este tipo de interacción en la mayoría de los robots como los robots de servicio y los robots personales. La próxima sección discute los diferentes métodos de interacción por los que se puede lograr la interacción directa entre el ser humano y el robot. Desde el punto de vista de la arquitectura software, el modelo cliente/servidor de dos capas se puede usar para llevar a cabo tal tipo de interacción como aplicaciones locales. En estos modelos, el lado del cliente es responsable del acceso a los datos, la implementación de la lógica, estructuración de los datos, la interfaz de usuario y la entrada de los datos. El servidor maneja ambos datos y procesos. Si el cliente y el servidor son parte del mismo programa, residen en el mismo espacio de direcciones y el cliente sostiene una referencia al servidor (su dirección de memoria) para invocar sus funcionamientos. Debido a que todos los detalles involucraran en el lado del cliente (clientes complejos), los programas son muy difíciles de adaptar a las aplicaciones remotas. Por eso este modelo es más conveniente para las aplicaciones locales.

2.5.2 Interacción Remota

En la interacción remota, el hombre y el robot están separados por barreras físicas. Hoy en día, la red de Internet se usa normalmente como un medio de comunicación para la interacción remota. En el siguiente capítulo de la presente tesis, se discute detalladamente la interacción remota basada en Internet.

Una tendencia innovadora en el campo de interacción remota es el desarrollo de laboratorios remotos y distribuidos. Un laboratorio remoto puede definirse como un laboratorio basado en red donde el usuario y los equipos del laboratorio real están geográficamente separados y donde las tecnologías de la telecomunicación se usan para dar a los usuarios acceso a los equipos del laboratorio. Estos laboratorios no están restringidos a la asistencia sincronizada por instructores y estudiantes; ellos tienen la potencia de proporcionar acceso constante siempre y cuando sea necesario. Se pueden reunir varios laboratorios remotos formando un laboratorio distribuido que se puede considerar como una red de laboratorios remotos. Tal laboratorio es un entorno de trabajo que puede usarse para proporcionar un conjunto de experimentos para los estudiantes con hardware físicamente distribuido en diferentes sitios, pero accesible vía Internet. El proyecto IECAT es un ejemplo de un laboratorio distribuido en el campo de la mecatrónica y la telemática [IECAT,03]. En capítulo 4, se discuten los laboratorios remotos y distribuidos en más detalle.

2.6 ELEMENTOS DE INTERACCIÓN

Los avances recientes en informática y robótica hacen los robots más aplicables en la vida diaria. Los robots y las personas co-existirán para compartir y cooperar en tareas de varias maneras. Se necesitan medios naturales por los cuales las personas puedan comunicarse e interactuar con los robots en cualquier sitio y en cualquier momento. Para llevar a cabo la interacción hombre-robot se utilizan varios instrumentos como los ordenadores o las recientes tecnologías como la comunicación hablada o las basadas en gestos. Esta sección discute los dispositivos principales de interacción por los que el hombre puede interactuar con el robot para realizar ciertas tareas.

2.6.1 Ordenadores

Los dispositivos tradicionales de entrada de los ordenadores como el teclado, el ratón y las pantallas táctiles se usan normalmente para realizar la interacción hombre-robot. Un área para la aplicación de interacciones hombre-robot avanzadas está en diseñar nuevos dispositivos de entrada y salida para los ordenadores. Estas aplicaciones incluyen un ratón modificado para presentar al usuario con pantallas táctiles las señales visuales por el movimiento del puntero del ratón, y diseñar dispositivos de entrada y salida para el concepto de un ordenador de hombro [Agah,01][Kobayashi,95-a].

El uso de pantallas táctiles puede mejorar la accesibilidad a los sistemas de interacción hombre-robot de una serie de usuarios que presentan restricciones de movimientos. Ello requiere un diseño de la interfaz de entrada diferente a la habitual en este tipo de pantallas. En [Gardeazabal,02] se propusieron unas disposiciones, en las que se utiliza para entrada de datos una parte de la superficie táctil que no coincide con la pantalla, que pueden ser útiles para las personas con leves discapacidades motoras, sino también para usuarios sin discapacidad.

2.6.2 Fuerza

El operador humano puede generar órdenes apropiadas aplicando fuerzas al sistema robotizado. Agah en [Agah,01] ha resumido los proyectos de investigación empleando el uso de la fuerza como un medio de interacción hombre-sistema en lo siguiente: el uso de un brazo maestro de siete grados de libertad (GDL) para controlar un manipulador remoto y un robot en un entorno virtual, control manual reflejando fuerza de un telemanipulador que usa un joystick isotónico de 6 GDL, un brazo de maestro controlado por un operador en el sitio local para manejar un robot esclavo en un sitio remoto, un operador humano que mueve un manipulador maestro para la teleoperación, el uso de dos maestros mecánicos para controlar herramientas quirúrgicas en micro-cirugía telecontrolada de ojos, un operador humano que cambia la forma de una mano de exoesqueleto usada como un manipulador maestro, el control del operador humano de un conjunto de antebrazo antropomórfico y una mano mecánica, control de un par de macro-brazos para teleoperar un par de micro-brazos, brazos maestros y esclavos con la misma estructura cinemática, y operar un brazo maestro para manipular objetos virtuales.

2.6.3 Comunicación Hablada

Los sistemas de voz admiten diferentes niveles de complejidad, desde manejar un menú mediante voz, hasta establecer un diálogo con el usuario referente a la tarea a realizar [Perez,97]. El reconocimiento de voz es generalmente un proceso de dos pasos: procesar la señal (para transformar una señal de audio a vectores de rasgos) seguido por la búsqueda de diagrama (para equiparar expresiones al vocabulario). La mayoría de los sistemas actuales usan modelos ocultos de Markov para determinar e*stocásticamente* el patrón más probable.

Para desarrollar un sistema robotizado con habilidad de entendimiento del idioma natural, deben tenerse en cuenta muchos problemas como tratar con la naturaleza no gramatical de muchas pronunciaciones habladas, la detección de errores en reconocimiento de voz y en la interpretación, y el diseño de diálogos claros e inteligentes.

2.6.4 Control de Mirada

La mirada fija es un buen indicador de qué está mirando una persona y a qué presta atención. Una dirección de mirada de una persona se determina por dos factores: la orientación de la cabeza y la orientación del ojo. Este tipo de control está basado en el procesamiento de imágenes utilizando un sistema de visión. Las aplicaciones de procesar imagen para la IHR son: procesamiento de imagen en tiempo real y reconocimiento de expresiones faciales básicas [Kobayashi,95-b], cámaras de monitorización para adquirir información visual de un objeto indicado [Kobayashi,94], y el seguimiento automatizado y el reconocimiento de objetos móviles no rígidos [Huang,95]. Aunque numerosos sistemas de visión se han desarrollado para investigar la orientación de la cabeza (generalmente con base en caras frontales), pocos investigadores han intentado investigar la mirada del ojo mediante el uso de sólo visión pasiva. Además, estos sistemas de seguimiento no han resultado fiables ni altamente precisos [Stiefelhagen,97].

El control de mirada fija (*Gaze Control*) puede jugar un papel importante en la interacción hombre- robot. Kanda et al. han definido este tipo de control como mecanismo de control para mover la dirección de la cámara del robot en su trayectoria, obstáculos, la persona que encuentra, etc. [Kanda,01]. Han desarrollado un robot que tiene dos sistemas de visión: un sistema omnidireccional para adquirir la información visual necesaria y un sistema binocular estereo de visión. El sistema de visión binocular indica qué está mirando el robot y no se usa para la locomoción. En este sistema, el robot obtiene toda la información visual necesaria desde el sistema omnidireccional o antes de comenzar los movimientos de la mirada, y entonces el robot deja de tener acceso a información visual durante el control de la mirada. El robot se comunica con el ser humano usando el control de mirada, que expresa sus estados internos, y parece para los observadores que el robot expresa sus intenciones. Un intérprete visual para monitorizar y generar acciones en el entorno de la interacción se ha descrito en [Colombo,97]. La idea clave es desarrollar un puntero basado en el ojo para saber donde está mirando el usuario realmente en la pantalla, y permite utilizar la pupila del ojo como un ratón 2D. Atienza y Zelinky en [Atienza,02] han propuesto un sistema de seguimiento activo de mirada que puede medir la dirección de la mirada de una persona en tiempo real como se muestra en figura 2.8.

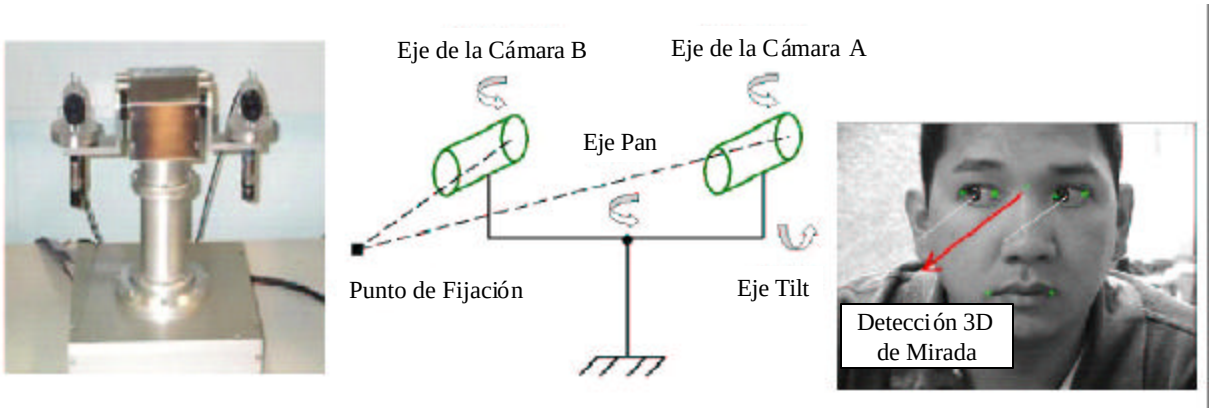


Fig. 2.8: Seguimiento Activo de Mirada

2.6.5 Gestos

La interacción basada en gestos es un proceso de interacción en tiempo real en el que el sistema de visión y el control del robot se combinan para extender la capacidad de percepción y reforzar el nivel inteligente del robot [Guo,99] [Krueger,91]. Gesto literalmente significa "un movimiento expresivo de una parte del cuerpo o la mano para traer intenciones y actitud" [Ashutosh,98]. El reconocimiento del gesto, o la identificación y clasificación de movimientos humanos, se está estudiando ampliamente como un mecanismo de entrada de datos en un ordenador. Mientras hay muchas herramientas de análisis del gesto humano basadas en varias técnicas, existen pocos sistemas robustos de gestos visuales. El seguimiento fiable tridimensional de los hombres en tiempo real (cabeza, manos, cuerpo, etc.) es una tarea difícil, y muchos sistemas de reconocimiento de gestos basados en visión restringen su aplicación a un entorno restringido [Ashutosh,98]. La figura 2.9 muestra los pasos de control remoto del robot basado en gestos descrito en [Ashutosh,98].

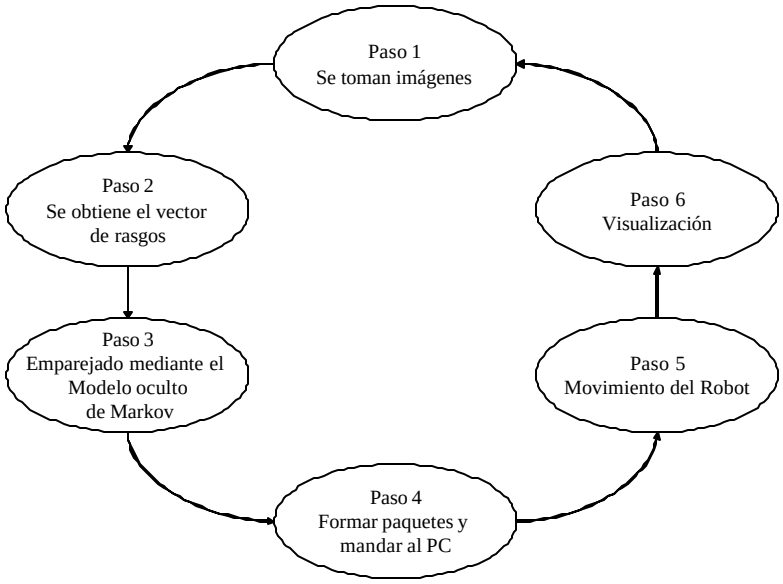


Fig. 2.9: Control basado en Gestos

2.6.6 Interacción Facial

La cara es la parte más importante para la interacción hombre-hombre. La comunicación cara-a-cara es la mejor manera para establecer comunicación entre los seres humanos. Eso significa que nuestras caras incluyen información no verbal importante. Las expresiones faciales no solamente pueden usarse en comunicación entre los seres humanos o comunicación no verbales para los sordos, sino también en sistemas de automatización avanzados, como robots de servicio para los que la interfaz hombre-máquina es un componente íntegro [Biena,03]. Así, la cara puede ser una herramienta necesaria para comunicar con los robots. Bruce et al han supuesto que la interacción cara-a-cara es el mejor modelo para obtener interfaces transparentes del robot que personas normales pueden utilizar [Bruce,02].

Varias técnicas inteligentes se pueden usar para resolver el problema del reconocimiento de expresiones faciales. Estas técnicas incluyen redes artificiales neuronales [Hara,95], metodología

de modelado fuzzy que usa rasgos basados en distancia manualmente extraídos [Ralescu,97], metodología basada en movimiento usando flujo óptico [Yacoob,96], y metodología basada en unidad de acción (*Action Unit - AU*) utilizando un modelo gráfico de ordenador [Essa,97]. Pero todavía existen varios dificultades o limitaciones en llevar a cabo un sistema práctico de reconocimiento que sea capaz de extraer rasgos fiables en tiempo real [Biena,03].

2.6.7 Dispositivos Móviles

La independencia del lugar es la ventaja más grande de los dispositivos móviles comparada a otras interfaces de control. Estos dispositivos móviles pueden ser útiles para las aplicaciones que requieren control remoto como las aplicaciones de los robots u otros equipos de automatización. Otra ventaja es el uso de dispositivos que ya existen y que los usuarios son familiares con su uso. Las posibles aplicaciones incluyen los robots personales y la automatización de la casa. Un robot de limpieza, tanto en casa como en una zona exterior como los centros comerciales o las estaciones, se puede dirigir y controlar por el operador desde cualquier sitio mediante el uso de un teléfono móvil.

Aunque no son muy abundantes, existen algunas aplicaciones de dispositivos móviles interactuando con robots móviles. Nikitas el al. han desarrollado un sistema basado en WAP para teleoperar un robot móvil [Nikitas,02]. Este sistema permite al operador controlar los movimientos del robot a través de un teléfono móvil. El robot está equipado con una cámara de video. El usuario periódicamente recibe en la pantalla del móvil las tramas de video desde la cámara del robot, también se pueden recibir imágenes de otras cámaras diferentes a la del robot. Kubik et al. han estudiado el control de un robot móvil a través de la voz (un menú de comandos sencillos como *Go*, *Stop*, *Turn*) utilizando un teléfono móvil [Kubik,01]. Se ha desarrollado un interfaz PDA *Palm Pilot* para el control remoto de vehículo en entornos peligrosos [Fong,00].

Se utiliza un teléfono *Motorola i85s* con J2ME interfaz para controlar el robot *Lego Flashbot* (ver figura 2.10) [Knudsen,01]. Este dispositivo realiza una conexión HTTP vía Internet a un servidor. Un servlet maneja los comandos que envía el teléfono y utiliza *Java Communications API* para enviar comandos por el puerto serie a una torre de infrarrojos. Esta torre transmite los comandos hacia el robot.

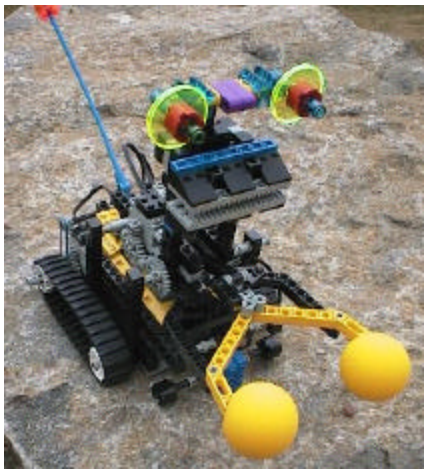


Fig. 2.10: Robot Flashbot

Los laboratorios de Fujitsu S.A. ha anunciado el desarrollo de un nuevo robot (MARON-1) para casa que se puede controlar remotamente mediante un teléfono móvil para operar en aparatos electrónicos domésticos o para la vigilancia como se muestra en la figura 2.11 [MARON,03]. Se ha diseñado este prototipo con una gama amplia de funciones, incluso el teléfono, cámara, telemando, cronómetro y equipo de vigilancia. Con estas características, por ejemplo, se podrá utilizar MARON-1 para supervisar casas u oficinas por la noche o para vigilar personas que requieren cuido especial y monitorización.

Fig. 2.11: MARON-1

En el apéndice A, se explica el desarrollo de aplicaciones para dispositivos móviles. En el capítulo 6 se describen el diseño y la implementación de interfaces para dispositivos móviles desarrolladas en la presente tesis.

2.7 REALIMENTACIÓN

Realimentación inadecuada, o opacidad, denota una situación en la que un robot no comunica, o comunica pobremente o ambiguamente, lo que está haciendo, o por qué está haciéndolo, o en algunos casos, por qué está a punto de cambiar, o acaba de cambiar, lo que está haciendo. Sin esta realimentación, el hombre debe entender, de memoria o de un modelo mental de comportamiento del robot, la razón del comportamiento observado. En esta sección se plantean los diversos métodos que se pueden utilizar para proveer realimentación.

2.7.1 Realimentación Visual

El uso de dispositivos visuales es el método más común para proveer realimentación del sistema robotizado al usuario. Las imágenes visuales proporcionadas al usuario podrían ser de cuatro tipos posibles: (1) imágenes reales, (2) imágenes virtuales, (3) imágenes combinadas (4) datos. El primer tipo de demostración visual presenta las imágenes reales de un entorno al hombre, tomadas de escenas reales. El segundo tipo implica la presentación de un entorno virtual al hombre, como los entornos generados por ordenadores. En el tercer tipo de demostración visual las imágenes reales y las virtuales se combinan y se presentan al usuario para determinar las diferencias entre dos. Tales metodologías son necesarias para ver en qué se diferencian la predicción del sistema del entorno de tarea, del entorno real [Agah, 01].

Para proporcionar en tiempo real imágenes de sitios remotos, el sistema debería reaccionar dinámicamente a los cambios del ancho de banda y los recursos computacionales y sólo transmitir aquellos píxeles que en realidad se necesitan utilizando técnicas inteligentes (la fragmentación inteligente, velocidad de escenas inteligente, velocidad de tarea inteligente y la compresión bruta de fuerza) descritas en [Sayers,99]. La figura 2.12 muestra diferentes métodos de realimentación visual usados en el laboratorio remoto de robots móviles desarrollado en la Universidad Carlos III de Madrid.

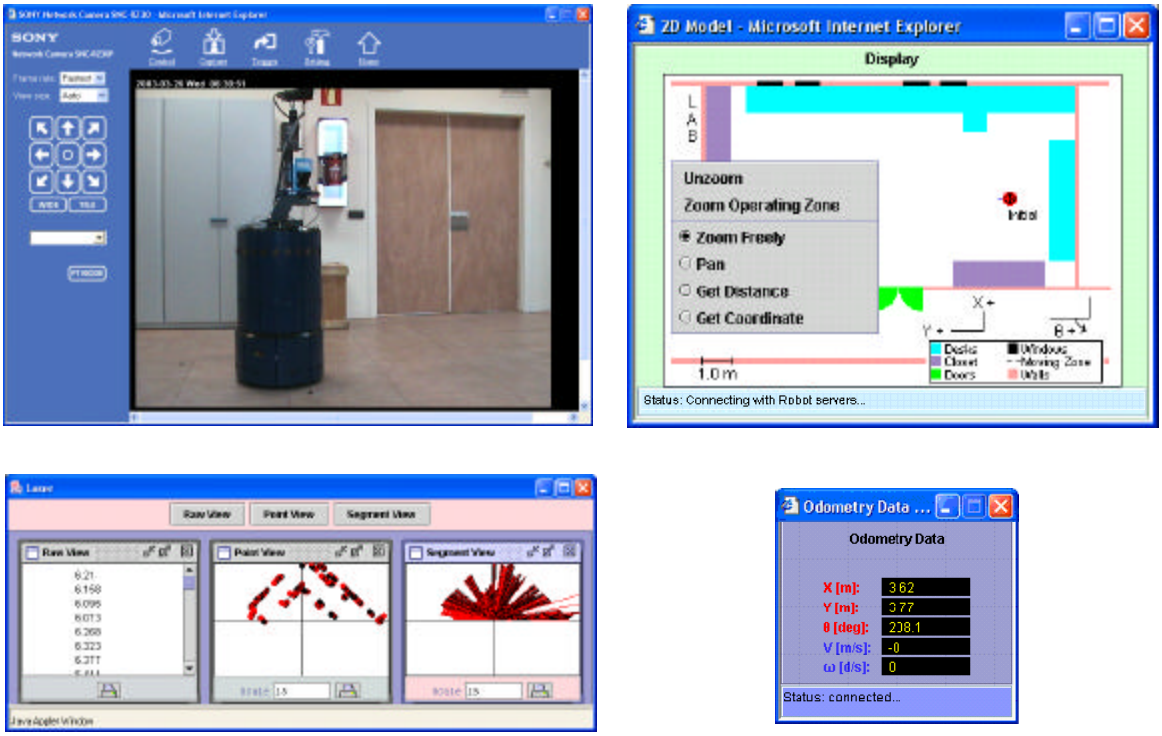


Fig. 2.12: Realimentación Visual

2.7.2 Realimentación de Audio

El sonido es esencial para realizar la experiencia visual y la interacción hombre-robot, pero por lo general, la mayor parte de los esfuerzos de investigación y desarrollo se concentra principalmente en la generación de sonido, la síntesis de voz y el reconocimiento de voz. La razón por la que sólo existe una pequeña atención sobre el análisis auditivo es que la percepción en tiempo real de una mezcla de sonidos es difícil. Los sonidos presentados al usuario a través de los dispositivos de audio pueden ser: (1) sonidos reales capturados del entorno; (2) sonido virtual generado por el ordenador; o 3) dispositivos de audio de información y datos que se convierten en formas de sonido. La información no auditiva presentada como sonidos incluye el estado de sistema interno, o la realimentación desde un sitio remoto, como la transformación de datos de realimentación de fuerza en sonidos.

2.7.3 Realimentación de Fuerza

Es posible para un sistema proporcionar fuerzas al hombre, donde estas fuerzas de realimentación permiten al hombre percibir mejor el entorno de la tarea. Estas fuerzas son de dos tipos: (1) fuerzas transmitidas por un entorno real (sitio local o remoto); o 2) fuerzas de un entorno virtual, como las fuerzas generadas por un ordenador. En ambos casos las fuerzas deben pasar por un proceso de escalado para proveer al usuario una realimentación de fuerza realista y segura. En [Agah, 01] se presentan varios ejemplos de sistemas con realimentación de fuerza.

2.7.4 Realimentación Táctil

Se usa la información táctil para proporcionar al usuario una percepción mejor del entorno de la tarea. La realimentación táctil consiste en textura, sensación de agarre, y otra información similar. Los casos de la utilización de realimentación táctil son: un monitor vibro-táctil para el dedo índice y pulgar de un operador para sustituir la realimentación de fuerza de la tarea de teleoperación [Massimino,93], realimentación táctil generando la sensación de geometría del objeto virtual a los niveles macroscópicos y microscópicos [Bergamasco,95], realimentación táctil que se proporciona por un ratón de ordenador modificado usando pulsaciones ligeras con el dedo en el botón del ratón [Akamatsu,94-a] [Akamatsu,94-b], la textura y realimentación de sensación de agarre proporcionadas en el guante del operador usando efectos de vibración de pulsos piezoeléctricos [Caldwell,94-a] [Caldwell,94-b], y pulsación de luz de la cabeza de un piloto para el dispositivo táctil de información a través de los cascos [Gilliland,94].

2.7.5 Expresiones

La realimentación emocional no verbal por medio de las expresiones faciales y el movimiento de cabeza/cuerpo juega un papel muy importante en la interacción humana. Movimientos pequeños de la cabeza, la boca, los ojos, o las cejas son a menudo suficientes para indicar el estado del proceso de interacción como distraído, disfrutando, entendiendo, etc. [Breemen,03]. Bruce et al intentaron estudiar la influencia de las personas para reconocer las sutilezas de la expresión como un mecanismo de realimentación. Esta expresión se lleva a través de muchos cauces: el discurso, expresión facial, gesto, y postura [Bruce,02]. Estos usaron un modelo animado 3D para desplegar la cara del robot. El discurso y los fonemas acompañados que se usan para sincronizar los labios, se generan usando un software especial texto-a-discurso. De este experimento, concluyeron que teniendo una cara expresiva e indicando la atención con movimientos hacen un robot más irresistible para interactuar con él.

El robot Kismet es un robot cuyo propósito es la interacción social cara-a-cara [Bruce,02]. Este robot usa expresiones faciales y vocalizaciones para indicar sus emociones y guiar la interacción de personas con él. Se ha diseñado este robot con dos sistemas estereofónicos activos, visión y audio, adornados con rasgos faciales para la expresión emotiva [Breazeal,01]. Estos rasgos faciales incluyen cejas, orejas, globos del ojo, y párpados. Este robot puede mostrar expresiones reconocibles como enojo, fatiga, temor, hastío, la excitación, felicidad, interés, tristeza, y sorpresa. Otro ejemplo es el robot CERO (*Co-operative Robot Operator*), que trabaja como un punto focal para el usuario. Usando cuatro motores RC-servo, el robot puede cabecear, agitar su cabeza y mover sus brazos. Su diseño sin ojos pretende señalar al usuario que no hay

capacidades de visión [Green,00]. En el proyecto Lino, la cabeza del robot con una apariencia buena y lista y realimentación emocional puede configurarse de forma que el usuario humano disfruta de la interacción y aceptará fácilmente posibles equivocaciones. La figura 2.13 muestra las expresiones emocionales de este robot [Breemen,03].

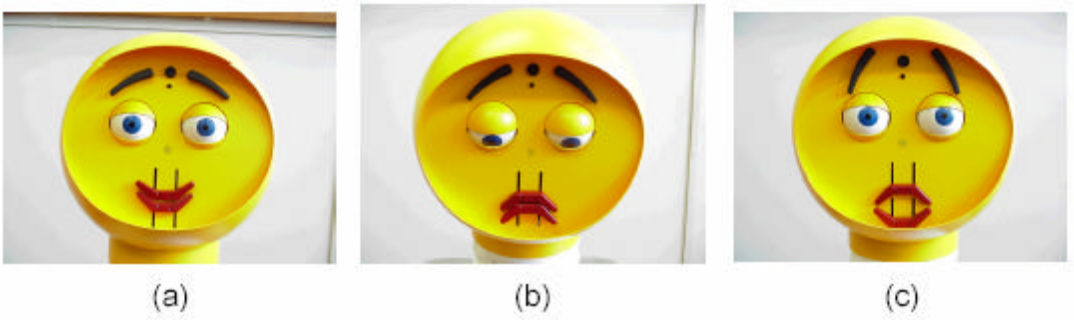


Fig. 2.13: Cabeza mecánica 3D que expresa emociones del robot *Lino*
a) feliz, b) triste, c) sorprendido.

En lugar de usar actuación mecánica, otra metodología para proporcionar expresiones faciales es utilizar gráficos de ordenador y técnicas de animación. Por ejemplo, Vikia tiene una cara 3D de mujer [Bruce,02]. Como la cara de Vikia (véase la figura 2.14) se muestra gráficamente, están disponibles muchos grados de libertad para generar expresiones [Fong,02].

Fig. 2.14: Cara de Vikia generada por ordenador

2.7.6 Interacción Multimodal

Existen varios robots de guía de turismo en los museos diseñados para interactuar con personas utilizando varios modelos de interacción. El robot Minerva usa el aprendizaje reforzado para aprender cómo atraer a las personas para interactuar con él, usando una recompense proporcional a la proximidad y a la densidad de personas alrededor de él [Thurn,00]. Las acciones que el robot podría emplear para esta tarea incluyeron movimientos de cabeza, expresiones faciales, y actos de discurso. Sus resultados experimentales no mostraron que ciertas acciones tuvieron más éxito que otras con cualquier importancia estadística más de que las expresiones amistosas tuvieron más éxito en atraer a las personas [Breazeal,99].

En Minerva, se ha implementado una máquina de estado emocional [Schulz,00]. Esta máquina codifica el comportamiento de interacción del recorrido completo en un total de cuatro estados, como se muestra en la figura 2.15. Minerva empieza con un estado "feliz", sonriendo mientras viajando entre las paradas del recorrido, hasta el primer enfrentamiento con un obstáculo humano que no puede esquivar trivialmente. A estas alturas, el robot anuncia

amablemente que está haciendo un recorrido y cambia de sonreír a la expresión neutra, mientras apuntando su cabeza hacía la dirección en la que necesita viajar. Si esto no tiene éxito, Minerva adopta una expresión triste, y pide a la persona de obstrucción que se ponga detrás de él. Si la persona todavía no se mueve, entonces Minerva se enfada y se vuelve más exigente.

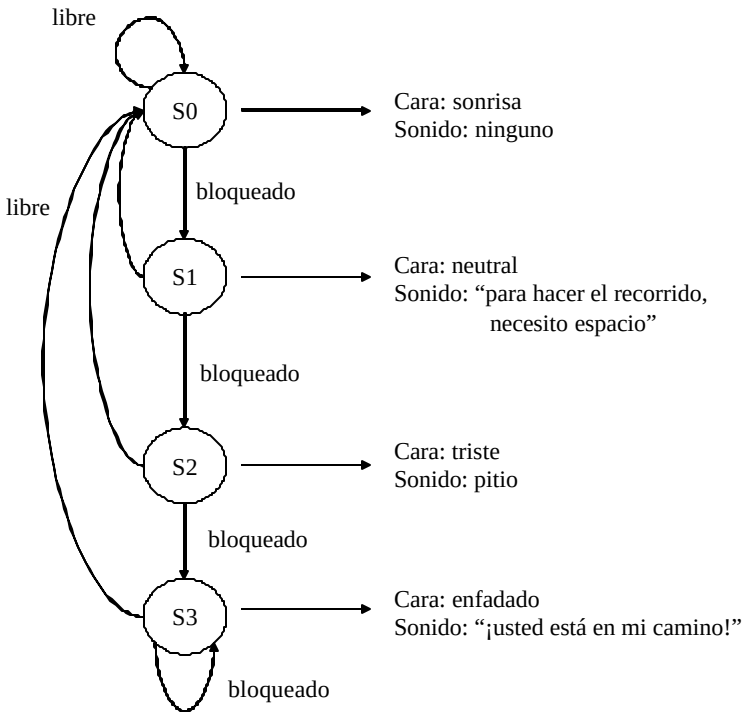


Fig. 2.15: Diagrama de estados de las emociones de Minerva durante el recorrido.

Un entorno de trabajo ha sido presentado en [Breazeal,02-a], para diseñar sistemas motivadores para robots autónomos expresamente engranados para regular la interacción hombre-robot. Este entorno de trabajo se inspira en la etología, la psicología, y el desarrollo cognoscitivo. Este sistema motivador implementa movimientos, emociones, y expresiones de cara. Ghidary et al. han desarrollado una interfaz para la interacción multimodal hombre-robot, que permite a la gente introducir a un robot recién llegado información sobre los diferentes atributos de objetos y sitios en la habitación utilizando órdenes de voz y gestos de la mano [Ghidary,01].

Una interacción multimodal entre agentes humanos y robotizados mediante la emoción artificial ha sido mostrado en [Suzuki,98]. El sistema desarrollado es un entorno robotizado basado en arquitectura de agentes, que es una de las herramientas software más útiles para aplicaciones multimedia en tiempo real como está descrito en [Camurri,97]. Kismet está diseñado especialmente como un niño, tratando a la gente de todas las formas que ocurren entre un niño y su canguro. REA [Breazeal,02-a] y Steve [Ricel,01] son caracteres humanoides que usan comunicación multimodal que imita el idioma y señales no verbales que las personas usan en las conversaciones cara-a-cara.

Capítulo

3

INTERACCIÓN REMOTA BASADA EN INTERNET

Capítulo 3

INTERACCIÓN REMOTA BASADA EN INTERNET

3.1 INTRODUCCIÓN

En este capítulo se discute el concepto de la interacción remota hombre-robot basada en Internet haciendo hincapié en las tareas que se pueden llevar a cabo utilizando este tipo de sistemas, en las ventajas e inconvenientes de utilizar Internet como medio de comunicación entre el sitio local y el sitio remoto y en la estrategia de control adecuada para desarrollar sistemas basados en Internet.

En el apartado 3.2 se describen los componentes básicos de cualquier sistema de interacción remota y se muestran algunas aplicaciones de estos sistemas. En el apartado 3.3, se discuten las ventajas y los inconvenientes de utilizar Internet como medio de comunicación, enfocando a los problemas tradicionales de Internet como el retraso temporal y el ancho de banda. En el apartado 3.4, se describen las estrategias de control que se pueden utilizar para desarrollar sistemas basados en Internet.

3.2 INTERACCIÓN REMOTA

Como se ha mencionado en el capítulo anterior, en la interacción remota hombre-robot existen barreras físicas que separan el ser humano del robot. La interacción remota incorpora varias tareas y procesos que se pueden llevar a cabo, como la teleoperación, la telepercepción, la teleprogramación, la experimentación remota, etc. Sin embargo, sin tener en cuenta la aplicación, todos los sistemas de interacción remotos con robots móviles suelen constar de los componentes básicos siguientes como se muestra en la figura 3.1:

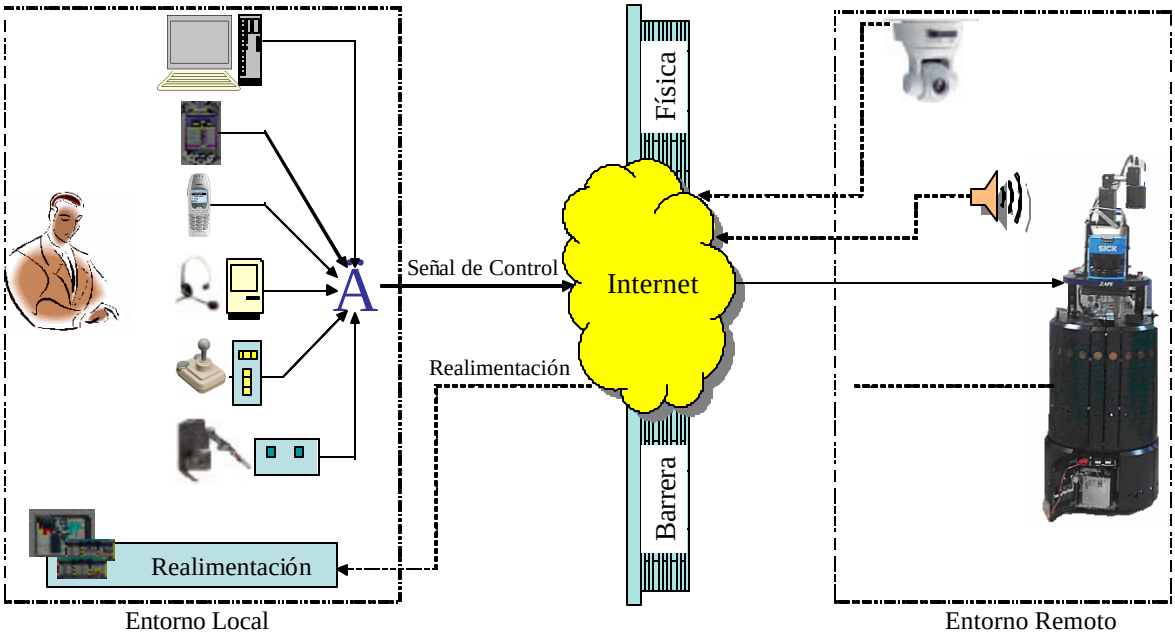


Fig. 3.1: Interacción Remota basada en Internet

- **Una interfaz de control:** Incorporando un dispositivo de interacción que el operador usa para mandar comandos al robot. Como se ha mencionado en el capítulo anterior, existen diferentes dispositivos de interacción con los que el usuario puede interactuar remotamente con el robot como con el uso de ordenadores personales, el uso de dispositivos móviles como las PDAs y los teléfonos móviles, mediante comunicación hablada, el uso de joysticks y dispositivos de interfaz háptica.
- **Un robot móvil:** Que realiza las acciones ordenadas por el operador en el sitio remoto.
- **Un esquema de comunicación entre los dos sitios**: En la interacción remota basada en Internet, se usa Internet como medio de comunicación por medio del cual se transmiten las señales de control del operador al sistema del robot y luego se proporciona realimentación al operador desde el entorno del robot.
- **Interfaces de realimentación:** Se pueden usar varias herramientas para proporcionar realimentación al usuario como se ha discutido en el capítulo anterior. La realimentación más frecuente es la realimentación visual mediante la transmisión de vídeo. La transmisión de vídeo exige disponibilidad de un ancho de banda alto por eso se pueden utilizar los modelos gráficos y los modelos de realidad virtual como una alternativa para proporcionar la realimentación visual. Otro tipo es la realimentación audio que no exige ancho de banda elevado pero tiene la desventaja de ser sensible al retraso temporal de la comunicación. También las ayudas kinestéticas, como la realimentación háptica, pueden usarse para proporcionar una realimentación física adicionalmente a los otros tipos de realimentación. Utilizando la realimentación háptica, el usuario puede tener sensación táctil que expresa la secuencia de sus comandos directamente en la interfaz de control.

En las subsecciones siguientes se presentan algunas de las aplicaciones de la interacción remota hombre-robot basada en Internet.

3.2.1 Teleoperación

La teleoperación, como herramienta para la realización de tareas en lugares remotos, ha estado presente siempre en el desarrollo de la robótica. De hecho, las primeras realizaciones en robótica se centraron en el desarrollo de robots teleoperados para la manipulación de sustancias peligrosas. La teleoperación se define como un conjunto de tecnologías que comprenden la operación o gobierno a distancia de un dispositivo por un ser humano [Aracil,02]. Se han clasificado los sistemas de teleoperación en tres niveles según la distancia entre el ser humano y el dispositivo [Sayers,99].

- **Teleoperación de Rango Corto:** En estos sistemas, la distancia entre el operador y el sitio remoto está restringida por la necesidad del operador de ver el entrono remoto directamente. En este caso, no hay restricción en el flujo de información entre los dos sitios, y no existe retraso de comunicaciones.
- **Teleoperación de Rango Medio:** En los sistemas de teleoperación de rango medio, el sistema eléctrico de teleoperación de rango corto se combina con unos medios para permitir ver el sitio remoto a distancia. La adición de cámaras y monitores significa que la separación entre el operador y el entorno remoto podría aumentarse considerablemente. En tales sistemas, la conexión entre los sitios es completamente eléctrica y no hay retraso perceptible de comunicaciones. El operador puede ver lo que está pasando vía una cámara y un monitor de televisión, puede escuchar lo que pasa vía un micrófono y altavoces, y puede sentir lo que está pasando vía una interfaz háptica o un manipulador con reflexión de fuerza.
- **Teleoperación de Rango Largo:** Con el aumento de la distancia entre los dos sitios, el retraso de comunicaciones aumenta hasta un punto donde los sistemas convencionales de teleoperación con reflexión de fuerza fallan. En estos sistemas, la realimentación juega un papel importante en el funcionamiento del sistema. En la sección 3.4 se discute con más detalles el control remoto basado en Internet como un caso especial de este tipo de sistemas.

3.2.2 Telepercepción

Debido al retraso de las comunicaciones, no se puede enviar realimentación inmediata y detallada desde el sitio remoto. En cambio debe generarse o utilizarse la información sensorial disponible en la percepción del entorno remoto en el sitio local del operador. La idea es aislar al operador del retraso de comunicaciones mediante el procesamiento local de los datos sensoriales enviados desde el entorno remoto como se muestra en figura 3.2.

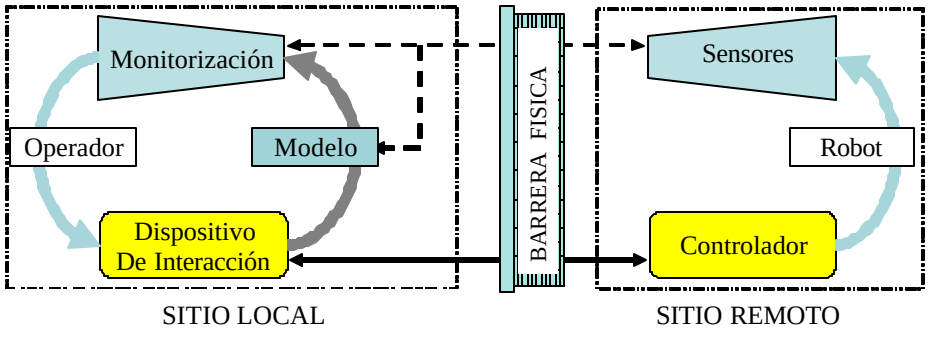


Fig. 3.2: Telepercepción

Por ejemplo, construir mapas del entorno utilizando datos sensoriales es un aspecto importante de navegación de los robots móviles, particularmente para esas aplicaciones en las que los robots deben funcionar en entornos no estructurados. Se pueden llevar a cabo diferentes algoritmos en el modelo local para generar mapas de entorno usando los datos de los sensores como una manera por la que se puede percibir el entorno remoto. Los mapas construidos también pueden usarse para localizar al robot en el entorno remoto y comparar el resultado del algoritmo de la localización con los datos de la odometría. En el capítulo 7, se puede encontrar un ejemplo sobre como se usan los datos sensoriales del sonar para construir mapas del entorno remoto y utilizar el mapa construido para localizar al robot remotamente.

3.2.3 Teleprogramación

Gracias a las tecnologías de telemática, se pueden desarrollar entornos virtuales para teleprogramar y probar distintos algoritmos para los robots móviles. Como se muestra en la figura 3.3, en los sistemas de teleprogramación se suele combinar una representación de realidad virtual del entorno remoto con un nivel bajo de inteligencia del sitio remoto. Esta representación virtual puede contemplar el modelo de los sensores de un robot real: sonares, sensores de contacto, cámara, etc. De esta forma, es posible reproducir y visualizar los movimientos, desplazamientos, giros, etc. realizados por el robot móvil virtual, para posteriormente ejecutar los programas en un robot real.

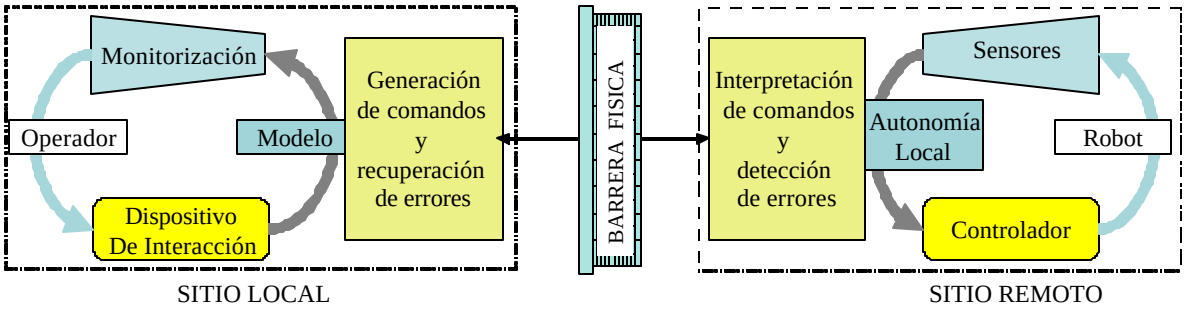


Fig. 3.3: Teleprogramación

En la teleprogramación, se suele utilizar un editor o un generador de comandos por medio del cual el usuario puede introducir un conjunto de comandos y parámetros, para posteriormente traducirlos a un *script*. Las instrucciones disponibles suelen ser un conjunto reducido de comandos que se envían por la red y se interpreten por un robot móvil. Las mismas instrucciones se utilizan para programar el robot real y el simulado. Si ocurre un error, el controlador puede hacer una pausa y señalarlo al operador que entonces debe diagnosticar y corregir el error y generar nuevos comandos para el robot remoto.

3.2.4 Experimentación Remota

La experimentación remota tiene la ventaja de proporcionar la posibilidad de compartir un experimento o varios experimentos entre varios operadores localizados en lugares distintos. De esta manera, podrían compartirse fácilmente experimentos reales entre varios laboratorios y, por lo tanto, se podrían reducir los costes. El capítulo siguiente discute en detalle los laboratorios remotos y distribuidos como entornos para la experimentación remota.

3.3 INTERNET COMO MEDIO DE COMUNICACIÓN

Recientemente, muchos proyectos de investigación y desarrollo (I+D) se están dirigiendo hacía la integración de Internet como medio de comunicación para desarrollar aplicaciones distribuidas en distintos campos. Como se ha mencionado en la sección anterior, se puede utilizar Internet para establecer la comunicación entre el sitio remoto y local en la interacción remota hombre-robot, aprovechando así las ventajas de Internet como un medio de comunicación barato y accesible. En las subsecciones siguientes se platean las ventajas y los inconvenientes de Internet como medio de comunicación. A continuación, se discuten en detalle los parámetros de la calidad de servicios como el retraso temporal, el ancho de banda, la pérdida de paquetes y la variabilidad instantánea (*jitter*).

3.3.1 Ventajas e Inconvenientes

El rendimiento de cualquier sistema basado en Internet depende del rendimiento actual de Internet. El rendimiento de una conexión depende de la velocidad y la fiabilidad con las que se transmiten los datos en esa conexión. En la actualidad no se pueden medir la velocidad y la fiabilidad total de Internet. Muy pocos datos cuantitativos de rendimiento de Internet están disponibles, pero recientemente se han creado varios proyectos para analizar el rendimiento. Los proyectos ITR (*Internet Traffic Report)* [ITR,03] y IWR (*Internet Weather Report)* [IWR,03] proporcionan una medida aproximada del rendimiento de Internet midiendo el rendimiento de las conexiones entre un grupo pequeño de sitios supervisados y distribuidos en varios lugares del mundo. Se monitorizan el retraso y las pérdidas de paquetes como medidas de la velocidad de la conexión y la fiabilidad respectivamente. Otros proyectos como IPMA (*Internet Performance Measurement and Analysis*) [IPMA,03] y IPPM (*Internet Protocol Performance Metrics*) [IPPM,03] pretenden desarrollar indicadores para analizar el retraso de Internet y la pérdida de paquetes. En cuanto a las ventajas e inconvenientes de Internet se pueden destacar los siguientes:

- **Ventajas**

Actualmente, las redes se están usando cada vez más para la comunicación en sistemas de interacción remota con los robots, debido al hecho de que las redes tienen varias ventajas como la fácil accesibilidad, la disponibilidad, la alta flexibilidad y el bajo coste. En particular esto es verdad para Internet. Aunque los requisitos para interacción remota con realimentación adecuada todavía no se han logrado debido al retraso temporal, el ancho de banda limitado y la pérdida de paquetes de datos. Estos parámetros dependen principalmente de las características de la red y su carga.

- **Inconvenientes**

Aunque la red de Internet mantiene un medio de comunicación barato y disponible, existen todavía muchos problemas por resolver antes de desarrollar aplicaciones verdaderamente fiables. Estos problemas incluyen el ancho de banda limitado y el retraso de la transmisión que varía arbitrariamente e influye en el rendimiento de los sistemas de interacción remota basados en Internet.

El retraso temporal de Internet es muy impredecible e inevitable, y diferente de los sistemas de teleoperación tradicionales donde se usa un medio de comunicación dedicado y, por lo tanto, se garantiza el valor del retraso. Este retraso de la transmisión afecta la fiabilidad del funcionamiento remoto. Más allá de un cierto valor de retraso, el control manual de un vehículo puede causar varios errores y en la mayoría de los casos se vuelve erróneo e impráctico [McGovern,90]. Por ejemplo, suponiendo que una función y(t)=sin(t)+3, se envía como información o señal de control desde el sitio local al sitio remoto, y teniendo en cuenta la influencia del retraso temporal de Internet en la información de control, la señal enviada se recibirá torcida como se muestra en la figura 3.4.

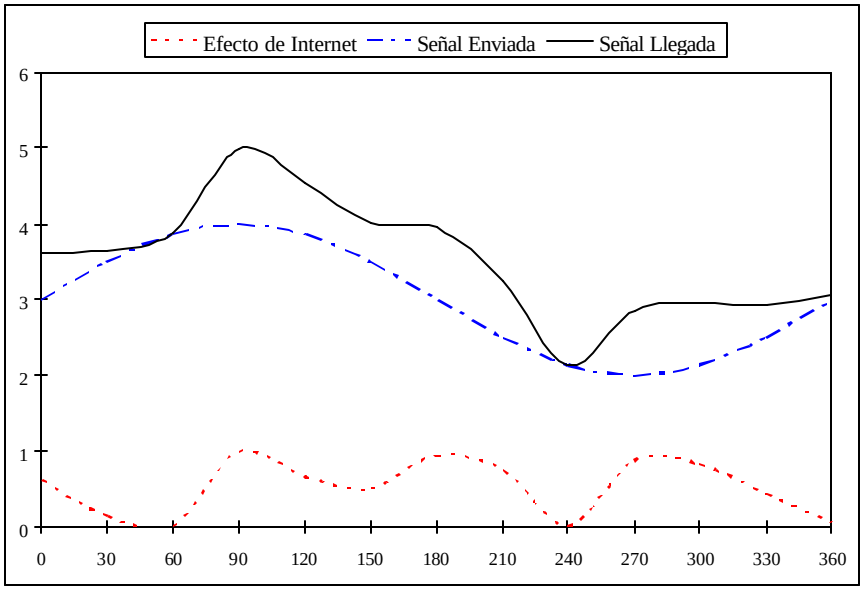


Fig. 3.4: Influencia de Internet

La figura 3.5 muestra la pérdida de información de las señales de control cuando el retraso temporal es T_d, y un comando se envía al robot remoto vía Internet cada tiempo T. Al sitio remoto, van a llegar N señales de control (donde N=entero[T_d/T]) al mismo tiempo después del retraso T_d, por eso la información de las señales se pierde [Han,01].

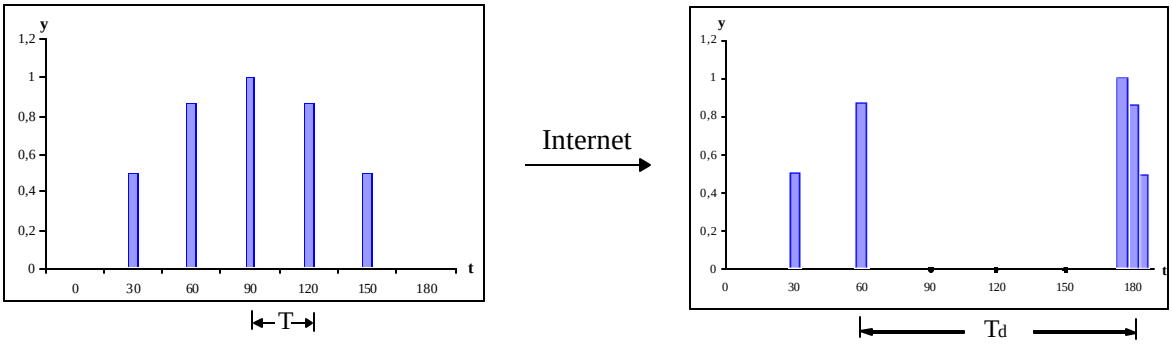


Fig. 3.5: Pérdida de Información de los señales de Control

Otros efectos del retraso temporal de Internet han sido estudiados desde la perspectiva del usuario. En [Ando, 99] se ha presentado una evaluación cuantitativa de la operabilidad con respecto al retraso de la comunicación en los sistemas de teleoperación y las percepciones de tiempo del operador humano. Los resultados experimentales de este estudio mostraron la tendencia de que, de 0,6 segundo a 1 segundo de retraso, había algún cambio en el tiempo de operación del operador. Sin embargo, no había resultados claros que pudieran ser significativos para diseñar un método de conexión de red más conveniente y una interfaz más utilizable para el sistema de teleoperación. En [Johnsen,71] se ha concluido que el rendimiento del operador disminuye significativamente con el aumento del retraso para más de 200 mseg. Borella et al. han intentado estudiar el efecto de la latencia de Internet sobre la percepción de usuario de la calidad de la información [Borella,97]. Han modelado tres condiciones de retraso controlando un instrumento de simulación en tres sitios Web populares (uno lento, uno rápido, y un tercero en medio entre los otros dos), asegurando así que los retrasos eran representativos de condiciones reales. La tabla 3.1 muestra el promedio y la mediana de estos retrasos.

Tabla 3.1: Mediana y Promedio de los Tres Retrasos

Retraso	Corto	Medio	Largo
Mediana (ms)	385	2210	3600
Promedio (ms)	575	3500	6750

La figura 3.6 muestra el promedio de las respuestas de 120 estudiantes a la pregunta "la información era de alta calidad". La línea sólida indica el promedio de las respuestas de los usuarios quienes navegaron en sitio Web basado en textos, mientras la línea discontinua indica el promedio de las respuestas de usuarios quienes utilizaron un sitio Web basado en gráficos y textos [Sears,97].

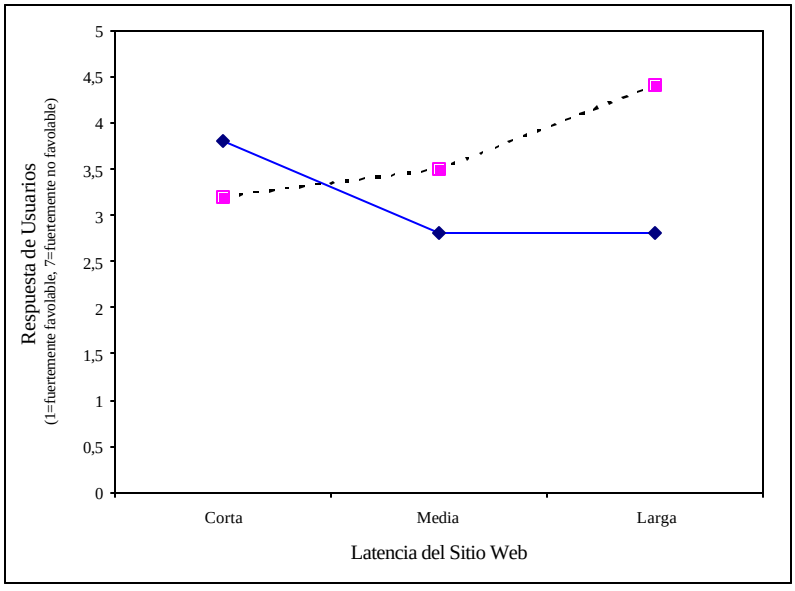


Fig. 3.6: Interacción entre Latencia de Internet y Percepción de Usuario de la Calidad de Información

Esto indica que con el aumento del retraso, los usuarios responden más negativamente a los sitios basados en gráficos y textos pero acerca lo mismo a los sitios de sólo texto, y que prefieren sitios de sólo texto con largos retrasos. Botella et al. supusieron una hipótesis, que cuándo los usuarios acceden a un sitio de gráficos y textos, consideran que los gráficos son la mayor fuente de retraso. Sin embargo, si el sitio es texto sólo, los usuarios parecen ser más tolerantes.

¿Pero cómo sienten los seres humanos el retraso de la comunicación? Es importante estudiar este asunto porque en algunos casos se puede pensar que en la interacción remota con retraso temporal de comunicación, la inestabilidad de la operación se puede deber a la percepción de tiempo del ser humano. La respuesta de esta pregunta se puede obtener de los conocimientos en el campo de la psicología. En psicología, se han presentado diferentes estudios sobre la percepción del tiempo en los seres humanos como son: el concepto de umbral temporal y la presencia psicológica [Ando,99]. El umbral temporal es el tiempo más corto que se puede percibir. El tiempo más corto en el que los estímulos se pueden distinguir. La tabla 3.2 muestra algunos umbrales de percepción de tiempo en la sensación visual, auditiva y táctil.

Tabla 3.2: Límites de Percepción de Tiempo en los Seres Humanos

Sensación	Limite
Sensación Visual	40 mseg
Sensación Audio	2 mseg
Sensación Táctil	30 mseg

La presencia psicológica se define como el rango de tiempo, durante el que se percibe un fenómeno que se produce continuamente. Se dice que la presencia psicológica está comprendida entre 5-6 segundos ó 2-3 segundos y cambia según la condición mental [Ando,99].

3.3.2 Calidad de Servicios

El concepto de calidad de servicios (*Quality of Service - QoS*) es muy importante en el área de las redes de ordenadores y se discute ampliamente en las aplicaciones multimedia distribuidas. QoS se define como un conjunto de requisitos necesarios de calidad en el rendimiento de transmisión de los datos para lograr la funcionalidad requerida de una aplicación. La calidad de transmisión de los datos vía Internet se refleja por un conjunto de parámetros según se muestra en la figura 3.7.

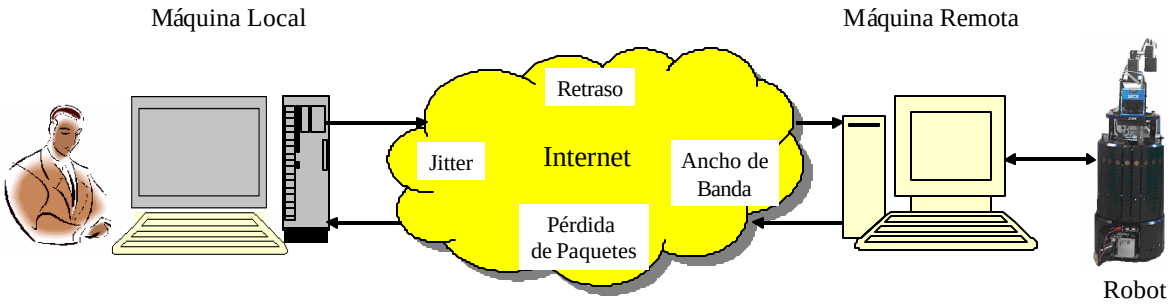


Fig. 3.7: Parámetros de QoS

Para valorar el impacto de los parámetros de la red en los sistemas de interacción remota, cualquier parámetro que describe el rendimiento de la red necesita ser definido desde la perspectiva de los usuarios, y entre los dos sitios que forman el sistema. Según la literatura en redes, los siguientes cuatro parámetros, ilustrados en la figura 3.7, describen suficientemente el rendimiento de la red en términos del modelo de QoS [Kurose,01].

3.3.2.1 Retraso Temporal

Los datos transmitidos sobre una red como Internet sufren retraso temporal causado por el retraso de encolamiento, el retraso de procesamiento, el retraso de transmisión en los interruptores y el retraso de propagación en las conexiones. El retraso de encolamiento define el tiempo que un paquete espera en el búfer de un interruptor hasta que se transmite en la próxima conexión. Por lo tanto su valor varía con la carga de la red. El retraso de la propagación depende de la distancia física debido a la velocidad de la luz. El retraso de encolamiento representa la mayor parte del retraso total de la comunicación.

El retraso temporal, como un parámetro de QoS, representa el tiempo medio requerido por un paquete para viajar de un emisor a un receptor. En general, la mayoría de las aplicaciones basadas en Internet utiliza el protocolo HTTP como protocolo de comunicación entre el sitio local y el sitio remoto. HTTP se implementa en dos programas: un programa cliente y un programa servidor que se ejecutan en los dos sitios que forman el sistema, cambiando mensajes HTTP como se muestra en la figura 3.8.

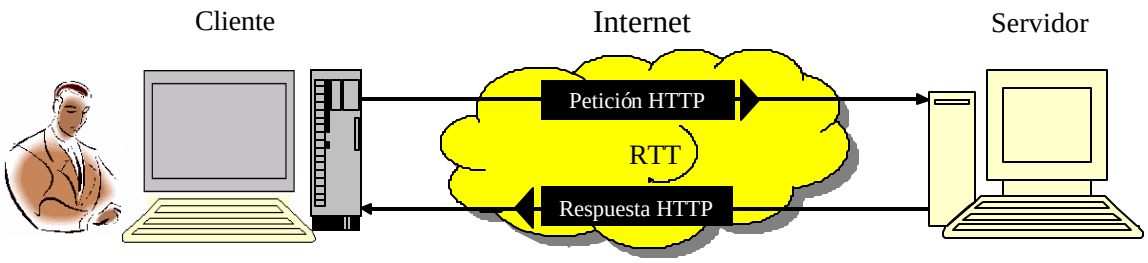


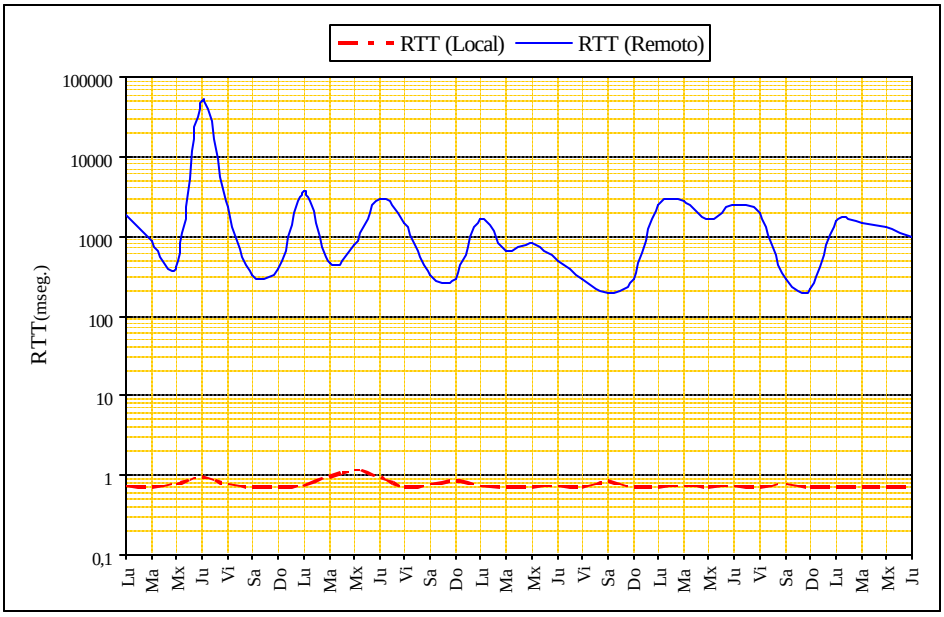
Fig. 3.8: Petición-Respuesta HTTP

Una página Web consiste en objetos. Un objeto simplemente es un archivo como un archivo HTML, una imagen JPEG, una imagen GIF, un applet de Java, un clip de audio o vídeo, etc. que se referencian por una URL. La mayoría de las páginas Web consisten en un archivo base HTML y varios objetos referenciados. Por ejemplo, si una página Web contiene texto HTML, una imagen GIF y un applet de Java, entonces esta página tiene tres objetos. El tiempo de ciclo (*Round Trip Time – RTT*) se define como la cantidad de tiempo desde que un cliente pide el fichero base HTML hasta que el cliente recibe este archivo. Este retraso incluye los retardos de propagación de paquetes, el retardo de encolamiento en los enrutadores intermedios y los interruptores, y los retardos de procesamiento de paquete.

Se ha desarrollado un estudio, durante el periodo de 7/1/02 a 5/2/02, para evaluar el rendimiento de la red entre la Universidad de Ciencias Aplicadas FH-Weingarten en Alemania, y la Universidad Carlos III de Madrid que son miembros del proyecto IECAT [IECAT,03]. Existen herramientas especiales por las que se puede medir el rendimiento de la red como *ttcp* [ttcp,03],

pingER [PingER,03] o *hp netperf* [Netperf,03]. En este estudio se ha utilizado el netperf que consiste de dos ficheros ejecutables: el cliente del netperf en una máquina Linux de 200MHz situada en la Universidad de Ciencias Aplicadas, y el servidor del netperf en una máquina Linux de 300MHz situada en la Universidad Carlos III de Madrid. Las pruebas se realizaron varias veces utilizando distintas duraciones de pruebas (5, 10, 30, 60, y 120 seg.) y se ejecutaron dos veces al día (a las 11:00H y a las 20:00H) con el objetivo de disminuir los errores.

Con el netperf, se pueden ejecutar varios tipos de pruebas. Se han concentrado en las pruebas que utilizan el protocolo TCP como la *TCP Stream*, la *TCP Response/Request* y *TCP Connect/Request/Response*. La prueba TCP Connect/Request/Response se ha usado para medir el tiempo de ciclo entre los dos sitios. Esta prueba imita el protocolo HTTP usado por la mayoría de los navegadores Web. En lugar de medir simplemente el rendimiento de petición/respuesta como en el caso de la prueba *TCP Request/Response*, en esta prueba se establece una nueva conexión para cada par petición/respuesta. La figura 3.9 muestra una comparación entre el tiempo del ciclo remoto (entre UC3M y FH-Weingarten) y el tiempo de ciclo local (desde la Intranet de FH-Weingarten).



Host	Min.	Av.	Max.
Local	0,698	0,758	1,16
Remoto	192	2726	5000

Fig. 3.9: Tiempo de Ciclo en mseg.

Como se puede observar, el tiempo de ciclo medido desde el host local (Intranet de FH-Weingarten) es mucho más bajo que el tiempo de ciclo proporcionado por la red entera entre el cliente y el servidor. Esto muestra que Internet impone retrasos temporales serios que hay que tener en cuenta al desarrollar sistemas basados en Internet.

Los resultados de esta prueba han sido utilizados para calcular la latencia que puede definirse como el tiempo desde que el cliente inicia una conexión TCP hasta que recibe el objeto pedido en su totalidad. La latencia de Internet ha sido estudiada por un número pequeño de investigadores.

En [Mukherjee,94] y [Fasbender,95] se desarrollaron modelos analíticos para la distribución del tiempo de ciclo entre dos sitios. El modelo usado para calcular la latencia está basado en el análisis del protocolo TCP [Kurose,01]. En este modelo, se han tenido en cuenta las siguientes suposiciones:

- La cantidad de datos que el remitente puede transmitir está limitada por la ventana de congestión del remitente. La ventana es un rango de sucesión de números permisibles para paquetes transmitidos pero todavía no reconocidos.
- Los paquetes no se pierden ni se corrompen, por eso no hay ninguna retransmisión.
- Todas las cabeceras de los protocolos –incluso las cabeceras TCP, IP, y la capa de conexión– son despreciables e ignoradas.
- El objeto (un archivo) a transferir, consiste en un número entero de segmentos de tamaño MSS (Tamaño Máximo de Segmento – *Maximum Segment Size*).
- El MSS es S bits. Según el RFC879, el tamaño máximo de segmento TCP es el tamaño máximo de datagrama IP menos cuarenta [RFC879,83]. Es decir que S = 576 – 40 = 536 byte.
- Los únicos paquetes que no tienen tiempos de transmisión despreciables son paquetes que llevan segmentos TCP de tamaño máximo. Los mensajes de petición, reconocimientos, y los segmentos de establecimiento de conexión TCP son pequeños y tienen tiempos de transmisión despreciables.
- El tamaño del objeto a transferir es O-bits.
- La velocidad de transmisión de la conexión es R bps.
- El tiempo de ciclo es RTT.

Según el diagrama de cronometraje de la figura 3.10 para una conexión TCP, primero el servidor comienza con una ventana de congestión de un segmento y envía un segmento al cliente. Cuando recibe un reconocimiento para el segmento, aumenta su ventana de congestión a dos y envía dos segmentos al cliente (espaciados por S/R segundos). Cuando recibe los reconocimientos por los dos segmentos, aumenta la ventana de congestión a cuatro segmentos y envía cuatro segmentos al cliente. El proceso continúa, con el doblamiento de las ventanas de congestión cada RTT.

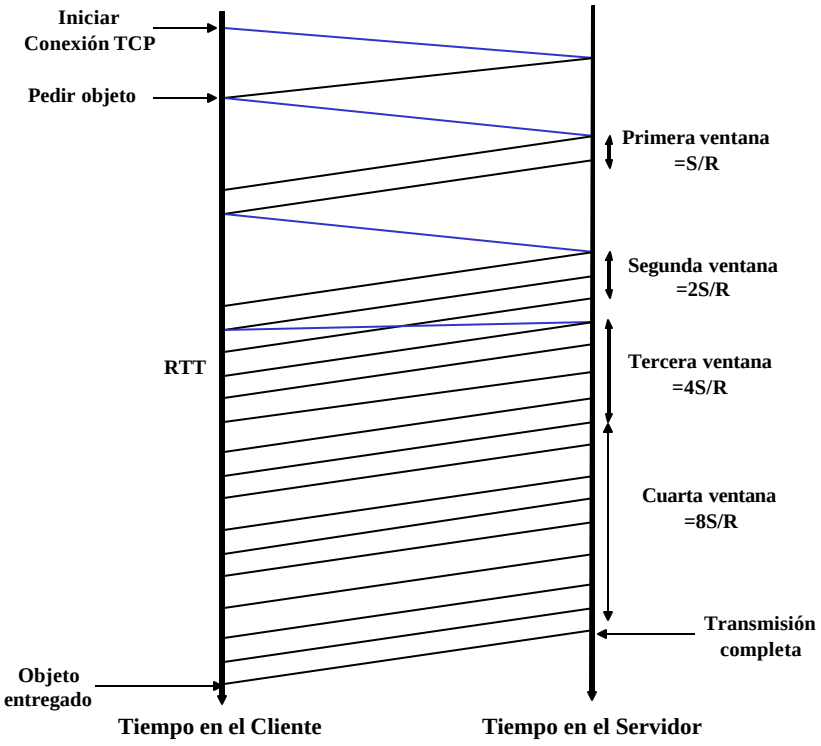


Fig. 3.10: Diagrama de Cronometraje de TCP

Este diagrama muestra que la primera ventana contiene un segmento, la segunda dos segmentos, y la tercera cuatro segmentos. Generalmente, la ventana K^{th} contiene 2^{K-1} segmentos.

Se supone que:

K = el número de ventanas que cubra el objeto
O = el tamaño de objeto a transmitir.
S = el tamaño máximo de segmento, MSS.
O/S = número de segmentos en el objeto.

K puede expresarse en términos de O/S como:

$$K = \min\{ K : 2^0 + 2^1 + ... + 2^{k-1} \geq \frac{O}{S}\}$$ Ecuación 3.1

$$K = \min\{ K : 2^K - 1 \geq \frac{O}{S}\}$$ Ecuación 3.2

$$K = \min\{ K : K \geq \log_2(\frac{O}{S} - 1)\}$$ Ecuación 3.3

$$K = \left\lceil \log_2(\frac{O}{S} + 1) \right\rceil$$ Ecuación 3.4

Después de transmitir una ventana de datos, el servidor puede estar en espera (es decir, deja de transmitir) mientras espera por un reconocimiento. En el diagrama, el servidor está en espera

después de transmitir la primera y la segunda ventana, pero no después de transmitir la tercera. El tiempo en el que el servidor empieza a transmitir la ventana K^{th} hasta el momento en lo que el servidor recibe un reconocimiento para el primer segmento en la ventana es S/R + RTT. El tiempo de la transmisión de la ventana K^{th} es $(S/R)2^{K-1}$. El tiempo de espera es la diferencia de estas dos cantidades, que es $\left[\frac{S}{R} + RTT - 2^{K-1}(\frac{S}{R})\right]$.

Ahora la latencia en transferir un archivo puede calcularse. La latencia tiene tres componentes:

- 2RTT para establecer la conexión TCP y pedir el archivo,
- Tiempo de transmisión del objeto,
- La suma de todos los tiempos de espera.

$$Latencia = 2RTT + \frac{O}{R} + \sum_{K=1}^{K-1}\left[\frac{S}{R} + RTT - 2^{K-1}\frac{S}{R}\right] \qquad \text{Ecuación 3.5}$$

Siendo que Q igual al número de veces en las que el servidor se encontraría en estado de espera si el objeto contuviera un número infinito de segmentos.

$$Q = \max\{K : RTT + \frac{S}{R} - \frac{S}{R}2^{k-1} \geq 0\} \qquad \text{Ecuación 3.6}$$

$$Q = \max\{K : 2^{k-1} \leq 1 + \frac{RTT}{S/R}\} \qquad \text{Ecuación 3.7}$$

$$Q = \max\{K : K \leq \log_2(1 + \frac{RTT}{S/R}) + 1\} \qquad \text{Ecuación 3.8}$$

$$Q = \left\lceil \log_2(1 + \frac{RTT}{S/R}) + 1 \right\rceil \qquad \text{Ecuación 3.9}$$

Siendo que:

P = el número de veces que el servidor se atasca
P=min{Q, K-1}
En el diagrama, P=Q=2 donde K=4

Combinándose las ecuaciones anteriores da,

$$Latencia = 2RTT + \frac{O}{R} + \sum_{K=1}^{P}\left[\frac{S}{R} + RTT - 2^{K-1}\frac{S}{R}\right] \qquad \text{Ecuación 3.10}$$

Para más simplificación extensa, teniendo que $\sum_{K=1}^{P} 2^{K-1} = 2^P - 1$

$$Latencia = 2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R} \qquad \text{Ecuación 3.11}$$

Es interesante comparar la latencia TCP con la latencia que ocurriría si no hubiera ningún control de congestión (es decir, ninguna restricción de ventana de congestión). Sin el control de congestión, la latencia es 2RTT + O/S que se denomina la latencia mínima.

Como una aplicación del análisis de la latencia, se pretende calcular el tiempo de respuesta para una página Web enviada a través del protocolo HTTP. Este tiempo se puede considerar como el tiempo total que tarda el sistema en cargar la página.

Con HTTP no persistente, cada objeto se transfiere independientemente, uno después del otro. El tiempo de respuesta de la página Web es por consiguiente la suma de las latencias de los objetos individuales.

$$T = (M+1)\left\{2RTT + \frac{O}{R} + P\left[RTT + \frac{S}{R}\right] - (2^P - 1)\frac{S}{R}\right\} \qquad \text{Ecuación 3.12}$$

Siendo que:
T = el tiempo de respuesta,
M = el número de los objetos referenciados en la página Web,
M+1 = la suma de la página base HTML y los objetos referenciados.

Ahora se supone que se quiere calcular el tiempo de respuesta para cargar una página Web guardada en un servidor localizado en la Universidad Carlos III de Madrid desde un sitio cliente que es, en este caso, la Universidad de Ciencias Aplicadas FH-Weingarten, Alemania. Se asume que esta página tiene un objeto referenciado de tamaño 10 kB como se muestra en la figura 3.11.

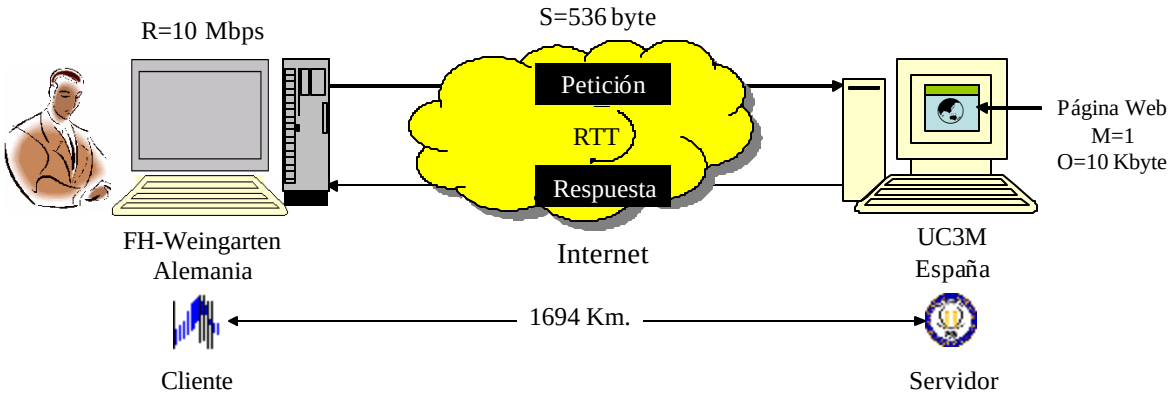
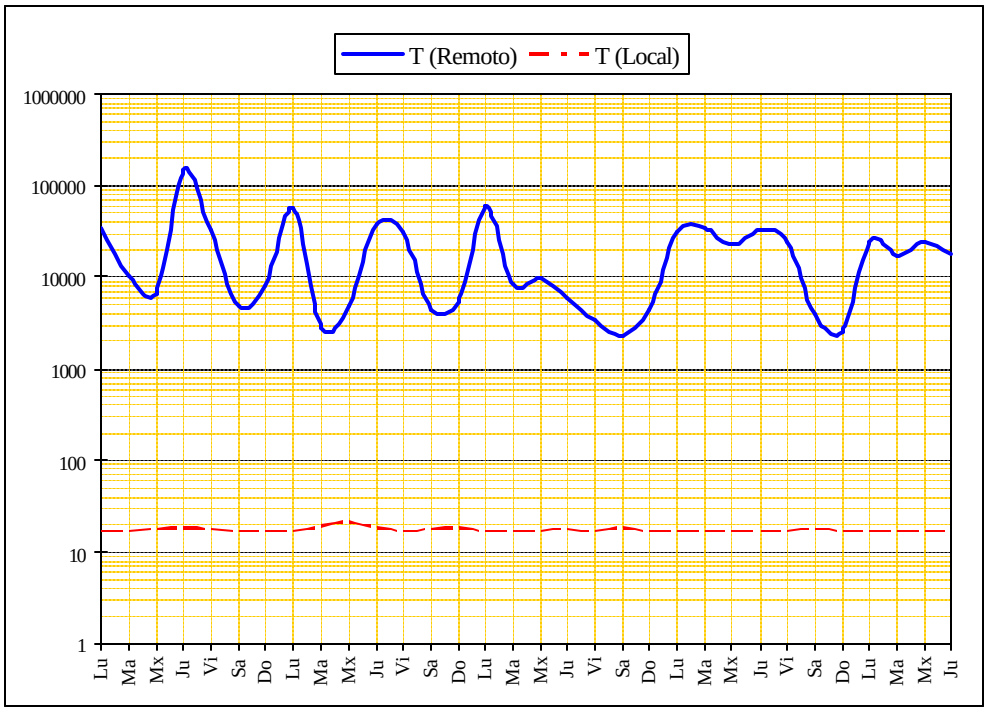


Fig. 3.11: Tiempo de Respuesta entre FH-Weingarten y UC3M

Como se muestra en la figura 3.12, el retraso temporal de Internet es muy impredecible e inevitable a diferencia de los sistemas tradicionales de teleoperación donde se usa un medio de comunicación dedicado que garantiza los retrasos.



Host	Mín.	Medio	Máx.
Local	17	17,5	21,6
Remoto	2319,6	22364	150001

Fig. 3.12: Tiempo de Respuesta en mseg.

Se ha llevado a cabo otro estudio para comparar el tiempo de respuesta desde distintos sitios de algunas universidades del proyecto IECAT y el proyecto TEAM [TEAM,03]. La figura 3.13 muestra las ubicaciones geográficas de estas universidades. En la figura 3.14 se puede ver una comparación entre el tiempo medio de respuesta calculado desde FH-Weingarten (Alemania) como host local y cuatro hosts remotos que son la Universidad Carlos III de Madrid (España), la Universidad de Aalborg (Dinamarca), la Universidad de Tecnología de Helsinki (Finlandia) y Universidad de Estado de UTA (EE.UU.). En el modelo usado se supone que un fichero HTML de 10 kB se transmite desde el host remoto hasta el host local.

Fig. 3.13: Ubicaciones Geográficas de la Universidades Examinadas

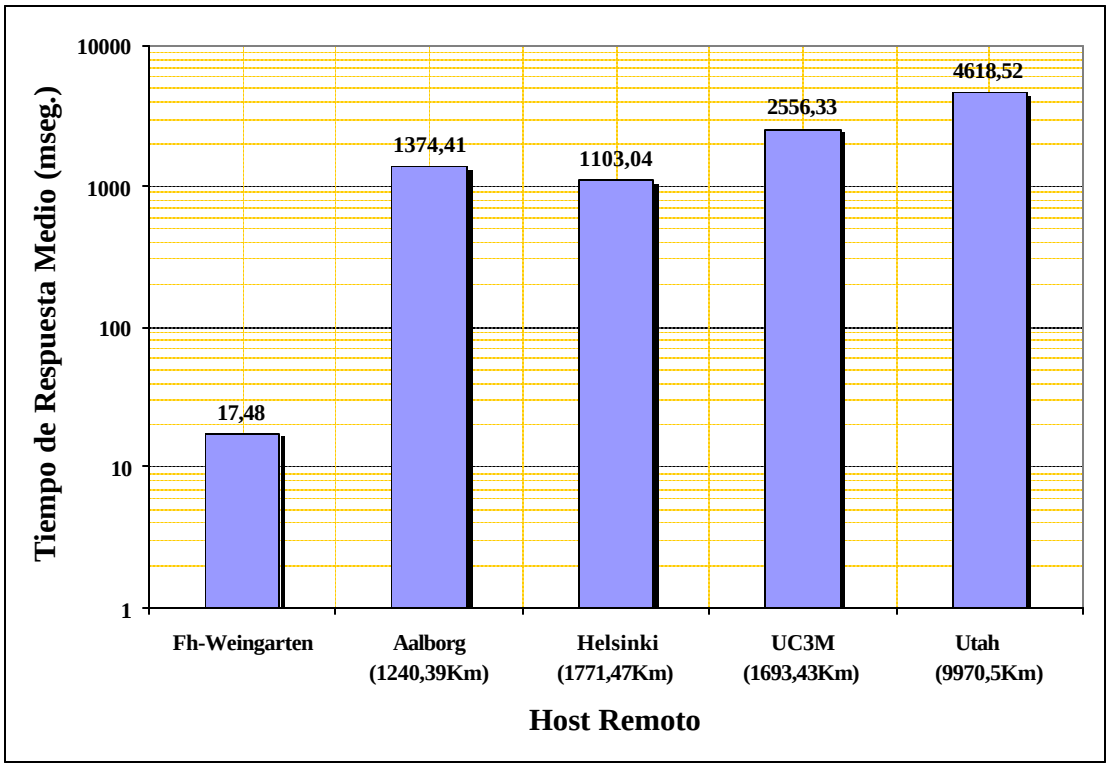


Fig. 3.14: Tiempo de Respuestas de Varios Hosts

Como se ha mencionado anteriormente, el retraso temporal de Internet se debe al retraso de encolamiento, al retraso de procesamiento, al retraso de transmisión en los interruptores y al retraso de la propagación en las conexiones. Como se muestra en la figura 3.14, las distancias

físicas entre los hosts no tienen efecto dominante en el retraso de respuesta debido a que el retraso de encolamiento representa la mayor parte.

La implementación del modelo de QoS en las redes actuales todavía está en proceso, así se da la oportunidad de proponer nuevos parámetros dinámicos de sintonía [Hirche,02]. Los sistemas de interacción remotos se van a beneficiar de la implementación de este modelo, no solamente, debido a los comportamientos garantizados de los dos sitios que forman el sistema, sino también, por el control de los parámetros de red a través de un criterio de rendimiento para adaptar el rendimiento a los requisitos de la tarea. En los puntos siguientes se resumen algunos comentarios sobre el retraso temporal de Internet:

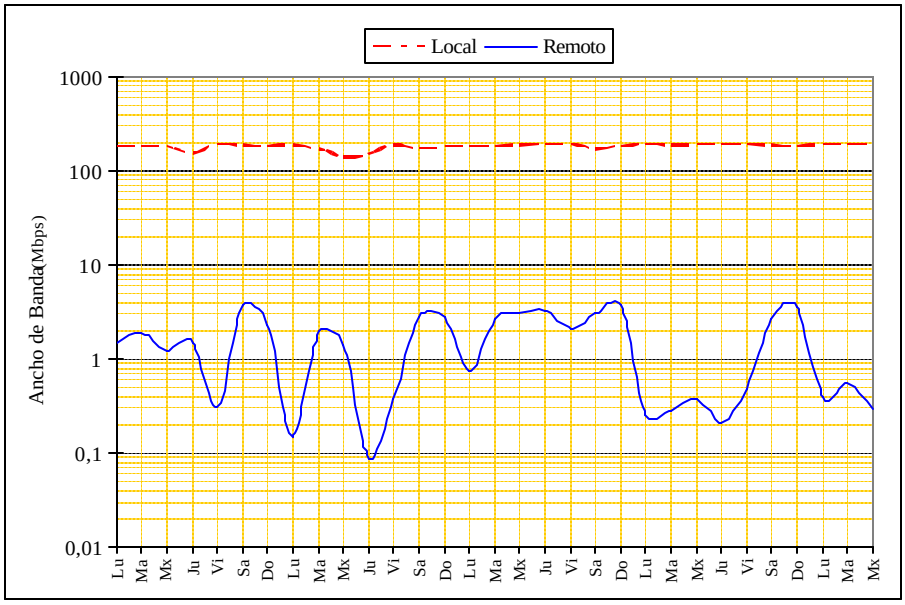
- El retraso temporal de Internet es muy impredecible e inevitable a diferencia de los sistemas tradicionales de teleoperación. Este retraso de transmisión afecta a la fiabilidad del funcionamiento remoto. Más allá de un cierto retraso, el control manual de un vehículo puede volverse erróneo o impráctico [McGovern,90].
- Las distancias físicas entre los hosts no tienen efecto dominante en el tiempo de respuesta.
- La variación del retraso de transmisión en la comunicación de Internet es el problema más importante que hay que manejar para llevar a cabo sistemas de interacción remotos. Dependiendo de cómo se transmiten los paquetes de información, este retraso puede variar de unos milisegundos a centenares.
- El retraso temporal es especialmente crítico debido a que valor afecta directamente a la realimentación sensorial y a las capacidades de control de bucle cerrado.
- El retraso de transmisión tiene que ser aceptable o una conexión dedicada se tiene que utilizar. El rendimiento del operador disminuye significativamente con el aumento del retraso más allá de 200 mseg. [Johnsen,71]. Otros estudios cualitativos muestran que las personas pueden compensar retrasos agregados pequeños, pero no pueden ignorar retrasos grandes (>100 mseg.) [Rogers,01].
- Debido a la variación de arquitecturas de teleoperación, no se puede dar ninguna formulación general de los requisitos mínimos. Normalmente, un retraso de comunicación de un segundo como máximo, se tiene como referencia para garantizar la operabilidad del sistema.

3.3.2.2 Ancho de Banda

El ancho de banda significa generalmente la cantidad de información que puede transmitirse en un periodo de tiempo dado (normalmente un segundo) a través de una conexión. En la transmisión de datos, el *throughput* es la cantidad de datos trasmitidos con éxito de un lugar a otro en un periodo de tiempo dado [Whatis,03]. En caso de que no haya pérdida de información, se puede utilizar el concepto del *throughput* para medir el ancho de banda.

Para evaluar el ancho de banda entre la Universidad Carlos III de Madrid y FH-Weingarten, se ha utilizado la prueba *TCP Stream* del netperf. En esta prueba, el cliente del netperf (situado en FH-Weingarten) envía mensajes de tamaño seleccionado a un servidor (situado en la UC3M) que los recibe. No hay ningún protocolo por encima del protocolo TCP/IP, pero todos los mensajes que se envían también se reciben correctamente, por eso, el término *throughput* se puede usar

como ancho de banda. Un cronómetro detiene la transmisión según la duración de la prueba y se calcula el *throughput* en ambos lados utilizando el tamaño del mensaje, número de mensajes y el tiempo de la transmisión. La prueba se ejecuta varias veces utilizando distintos tiempos de ejecución y se calcula el promedio. La figura 3.15 muestra los resultados de esta prueba.



Host	Mín.	Medio	Máx.
Local	143	182,88	195
Remoto	0,089	1,61	3,768

Fig. 3.15: Ancho de Banda en Mbps

En los sistemas de interacción remota, se necesita ancho de banda adecuado para la transmisión de señales de control, información sensorial e imágenes en tiempo real. En [Bapna,98] se ha presentado un sistema de teleoperación para el robot móvil Nomad que permite controlar el Nomad con seguridad desde distantes centros de control. Las imágenes y los datos del Nómada también estaban inmediatamente disponibles en Internet. Bapna et al. han demostrado que un ancho de banda de 1,4 Mbps es suficiente para transportar imagines reales desde el robot a una estación de control local y después a los sitios de control remotos. Los resultados obtenidos, como se muestra en la figura 3.15, pueden ser considerados como resultados satisfactorios.

Las conclusiones siguientes pueden ser útiles para manejar el problema del ancho de banda limitado.

- En los sistemas basados en Internet, los comandos de alto nivel son ideales para interactuar con los robots remotos porque requieren menos ancho de banda. También la idea de superar las restricciones de la comunicación utilizando interacción de nivel más abstracto y aumentando la autonomía del robot es una idea fundamental para el control remoto vía Internet.

- Diferentes tipos de datos tienen niveles diferentes de importancia para el operador. Se recomienda transmitir con resolución óptima aquellos datos que se necesitan realmente.
- La limitación del ancho de banda de Internet limita la velocidad de refrescar las imágenes de videos por eso es recomendable prohibir experimentos en los que la escena podría variar a velocidades altas para evitar los fallos abruptos y saltos en la visualización [Khamis,02]. Los modelos gráficos y las imágenes de realidad virtual de una simulación dada tienen necesidades mucho más bajas en cuanto al ancho de banda, por lo tanto se pueden considerar como una mejor alternativa a las imágenes reales de la cámara en este tipo de situaciones.
- Para proporcionar en tiempo real imágenes de sitios remotos, el sistema debería reaccionar dinámicamente a los cambios del ancho de banda y a los recursos computacionales, y sólo transmitir aquellos píxeles que en realidad se necesitan utilizando técnicas inteligentes (la fragmentación inteligente, velocidad de escenas inteligente, velocidad de tarea inteligente y la compresión bruta de fuerza) descritas en [Sayers,99].
- Permitir a varios usuarios utilizar el sistema simultáneamente significa que los recursos del sistema tienen que estar compartidos y, por lo tanto, algunos tipos de interacción pueden estar perjudicados o prohibidos, particularmente aquéllos que requieren un ancho de banda alto y especializado

3.3.2.3 Pérdida de Paquetes

Probablemente la preocupación más grande de los sistemas de interacción remota basados en Internet es el comportamiento no determinista del sistema que resultaría durante la pérdida de paquetes o la caída total de la comunicación entre los sitios del sistema.

La pérdida de paquetes se origina por exceder la capacidad de la red causando que un dispositivo de la red deja caer un paquete. Este parámetro depende de la carga de la red y el mecanismo de encolamiento utilizado en el nodo de la red.

Una posibilidad para prevenir la pérdida de paquetes está implementada en TCP. En este protocolo, cuando se descubre una pérdida de paquetes, se pide un reenvío por el receptor. Esto produce una latencia más alta con el protocolo TCP comparándolo con el UDP, por eso existe una compensación entre porción de pérdida de paquetes y el retraso temporal. Generalmente, el parámetro del retraso temporal es más crucial que la pérdida de paquetes.

3.3.2.3 Jitter

Un componente crucial del retraso temporal son los retrasos arbitrarios de encolamiento en los dispositivos de la red. Debido a estos retrasos variantes dentro de la red, el tiempo desde la generación de un paquete hasta que se recibe, puede fluctuar de un paquete a otro. Este fenómeno se llama variabilidad instantánea o *jitter*. El *jitter* es la variación en la latencia en una ruta de conexión.

Se puede calcular el jitter enviando y recibiendo paquetes consecutivos. Como se muestra en la figura 3.16, dos paquetes se están enviando de un remitente a un contestador.

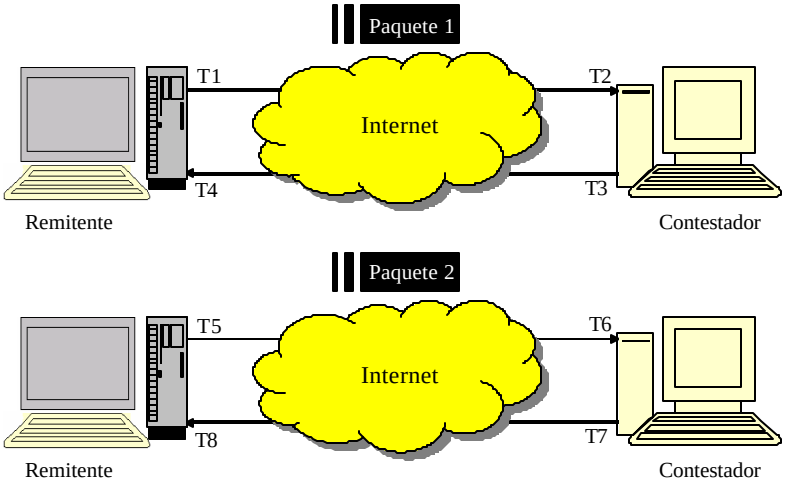


Fig. 3.16: *Jitter*

Teniendo que:

T1: Tiempo de envío del paquete 1
T2: Tiempo de recepción del paquete 1
T3: Tiempo de envío de la contestación del paquete 1
T4: Tiempo de recepción de la contestación del paquete 1
T5: Tiempo de envío del paquete 2
T6: Tiempo de recepción del paquete 2
T7: Tiempo de envío de la contestación del paquete 2
T8: Tiempo de recepción de la contestación del paquete 2

Para estos dos paquetes:

Jitter de la fuente al destino = (T6-T2) - (T5-T1)
Jitter del destino a la fuente = (T8-T4) - (T7-T3)

El jitter se calcula para cada dos paquetes consecutivos. Para medir el jitter usando las medidas del tiempo de ciclo, se supone que la medida i-th del tiempo de ciclo es R_i, entonces el *jitter* se calcula como el rango intercuartil (*Inter Quartile Range- IQR)* de la distribución de frecuencia de R [Cottrell,01]. La tabla 3.3 muestra las estadísticas descriptivas del tiempo de ciclo (RTT) entre UC3M y FH-Weingarten que se pueden utilizar para calcular el jitter.

Tabla 3.3: Estadísticas Descriptivas del RTT

	N	Promedio	Mediano	Rang.Tot Promedio	Dev. Est.	Err.Est. Promedio	Min.	Máx	Q1	Q3
RTT (mseg.)	32	2727	901	1188	8677	1534	192	50000	340	1911

Jitter = rango intercuartil (IQR)= tercer cuartil (Q3)- primer cuartel (Q1)
= 1911-340= 1571 mseg.

Se ha clasificado la degradación de redes en cuatro categorías en base del jitter según se muestra en la tabla 3.4 [TIPHON,98].

Tabla 3.4: Niveles de Degradación de Redes

Categoría de Degradación	Jitter Máximo
Perfecta	0 mseg.
Bien	75 mseg.
Mediana	125 mseg.
Baja	225 mseg.

El valor obtenido del jitter según las medidas del retraso temporal (1571 mseg.), indica que la comunicación entre el sitio local y el sitio remoto tiene categoría muy baja, por lo tanto, puede afectar al rendimiento del sistema. Como se puede ver en la figura 3.17, las aplicaciones más sensibles al retraso son las aplicaciones de voz [Mason,94]. Aunque utilizar la comunicación hablada como herramienta de interacción remota requiere transmitir poca cantidad de datos, el *jitter* afecta a la calidad de la voz llegada al sitio remoto donde se encuentra el robot.

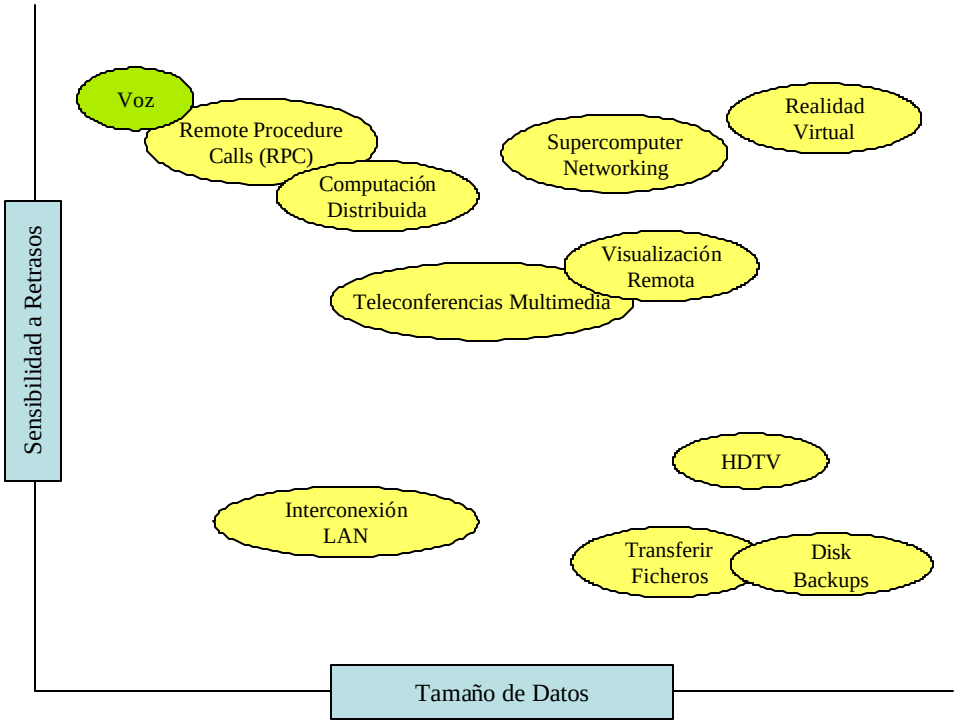


Fig. 3.17: Aplicaciones de Internet

Se pueden utilizar algunas pasarelas de voz para manejar el problema del *jitter* como *Cisco IOS H.323*, que puede manejar el retraso causado por *jitter* hasta 240 mseg. En [Kurose,01] se presentan varios mecanismos que se pueden usar conjuntamente para eliminar el *jitter* en algunas aplicaciones especiales que son sensibles a él, como son las aplicaciones de voz o telefonía de Internet.

3.4 CONTROL REMOTO BASADO EN INTERNET

Muchos investigadores han discutido los sistemas basados en Internet que permiten teleoperar un vehículo y la estrategia de control adecuada para desarrollar este tipo de sistemas. Se ha discutido la teleoperación remota basada en Internet, como un nuevo método para controlar un telerobot a largas distancias, donde los retrasos en la comunicación son significantes. Se ha probado que tales retrasos desestabilizan potencialmente el sistema y degradan la intuición y el rendimiento del operador humano [Brady,98].

Los protocolos de red para sistema de robótica móvil se han discutido en [Dogulas,97]. Se ha desarrollado un protocolo de gestión de comunicaciones que puede funcionar encima del protocolo estándar de Internet orientado a datagramas (User Datagram Protocol-UDP). Este protocolo proporciona el mejor balance posible de la transmisión de datos y tiempo de respuesta, al adaptarse su comportamiento de comunicaciones a los recursos de canales de comunicaciones realmente disponibles. La clave está en diseñar la interfaz, de forma que, los procesos de usuario de alto nivel accedan a esta capa de servicio de transporte (*Transport Layer Service*) y puedan evaluar la reducción de las comunicaciones en tiempo real.

Han et al. han propuesto un sistema de control que garantiza que un robot personal pueda evitar obstáculos y pueda reducir los errores de trayectoria y la diferencia de tiempo entre un robot virtual en local y un robot real en el sitio remoto [Han,01]. En el sistema propuesto se controlaba el robot personal a través de un simulador que se controlaba en el sitio local del operador, por eso este sistema ha sido insensible al retraso temporal. Según sus medidas, el retraso de Internet aumentaba con la distancia física entre el sitio local y el remoto, y también dependía del número de nodos cruzados entre el sitio local y remoto. También el retraso dependía fuertemente de la carga de Internet, por eso no se podía modelar.

Se ha propuesto en [Liu,00] usar la teoría de control basado en eventos como un algoritmo de programación, en el lado del servidor, para un sistema de teleoperación vía Web. Este sistema proporciona un control estable en presencia de incertidumbre en el tiempo de transmisión de la red y un ancho de banda limitado.

En las subsecciones siguientes se presentan dos estrategias de control que pueden ser útiles para desarrollar sistemas de control remoto basado en Internet.

3.4.1 Control Remoto Manual en Lazo Cerrado

Mediante un examen de los diferentes sistemas de teleoperación basados en Internet, como se puede ver en el capítulo siguiente, se descubre que todos están basados en una arquitectura similar. Estos sistemas toleran los retardos, ya que, existe un operador humano como enlace inteligente al sistema de control. Cuando se produce un error en el posicionamiento del dispositivo remoto, se visualiza el caso mediante realimentación visual y el operador se encarga de enviar comandos al dispositivo para compensar. Este modelo tiene los componentes que se muestran en la figura 3.18. A continuación se describe el papel de cada componente.

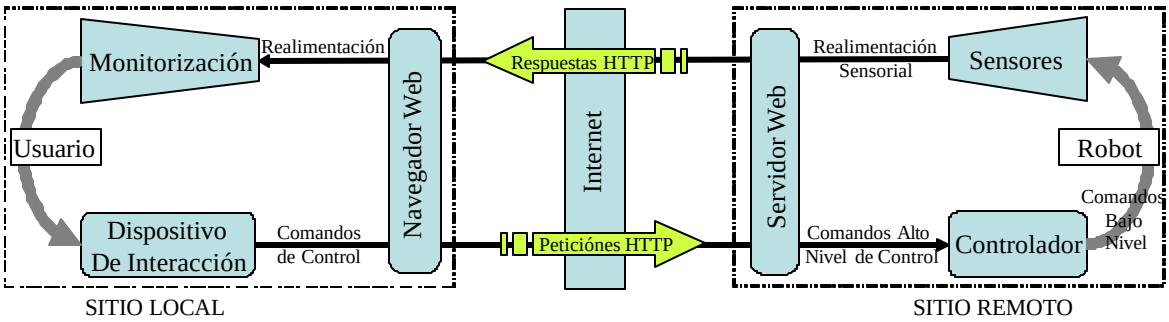


Fig. 3.18: Control Remoto Manual en Lazo Cerrado

El modelo de control remoto manual en lazo cerrado está basado en un modelo simple de programación distribuida y el protocolo "Petición-Respuesta" de la arquitectura cliente-servidor. El cliente interactúa con el sistema usando cualquier navegador Web que le muestre la interfaz de usuario. La interfaz reúne los datos necesarios y genera las peticiones que el navegador Web se encarga de transportar como peticiones HTTP. Estas son atendidas por el servidor Web, que se encarga de procesar los datos y las peticiones enviadas por el cliente para generar peticiones de alto nivel. Los servidores de control del robot se encargan de interpretar las órdenes de alto nivel y de generar comandos de bajo nivel que entenderán los actuadores, ejecutándose así las tareas demandadas.

La realimentación de la información sensorial proporciona al usuario información acerca de las acciones emprendidas por el robot dentro de su entorno, pudiendo analizar las consecuencias que tienen sus peticiones sobre el robot.

Si bien el sistema de control manual en lazo cerrado no es ciertamente óptimo, si es un sistema simple y que funciona. Se pueden utilizar pantallas de predicción para mejorar este modelo. Debido al retardo de las comunicaciones una realimentación inmediata no puede venir de lugar remoto y debe ser generada en la estación del operador [Sayers, 99].

3.4.2 Control Supervisado

En los sistemas basados en Internet y con la existencia de las restricciones de comunicación discutidas previamente, es recomendable intentar lograr un equilibrio entre la autonomía y la intervención del usuario para que éste sólo pueda ayudar al robot en casos excepcionales de emergencia. Sin embargo, el robot debe manejar la mayoría de operaciones locales automáticamente para disminuir la sensibilidad del sistema a los retrasos en la comunicación y su dependencia en el ancho de banda. Kress el al. han estudiado la selección de la estrategia de control teniendo en cuenta las consideraciones del retraso temporal [Kress,01]. Se ha dado un ejemplo sobre cómo desarrollar un sistema de asamblea basado en Internet.

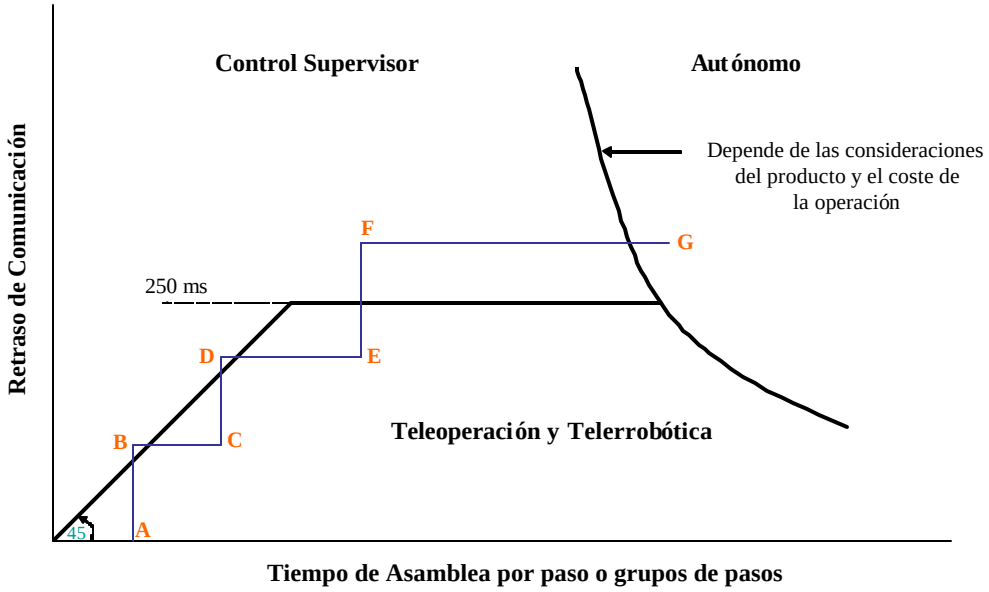


Fig. 3.19: Estrategia de Control de un Sistema de Asamblea.

Como se muestra en la figura 3.19, empezando en el punto A, con el aumento del retraso de comunicación de A a B (debido al tamaño del paquete, al tráfico de Internet, a la pérdida de paquetes, etc.) la teleoperación ya no es deseable cuando el retraso de comunicación iguala al tiempo de paso de asamblea. Se puede usar un controlador supervisor en este punto o formar nuevos grupos de pasos y automatizar ciertos pasos de asamblea hasta que la asamblea se pueda llevar a cabo telerrobóticamente (es decir, mover de B a C). En el punto C, como el tiempo de retraso se aumenta de nuevo (por el aumento del tráfico de Internet, transmisión de información sensorial adicional, etc.), se mueve hasta el punto D donde se requiere de nuevo el control supervisor. Como antes, la opción es quedarse con el control supervisor o formar grupos más largos de pasos y automatizar ciertas operaciones hasta que la asamblea pueda hacerse telerrobóticamente (punto E). Como se aumenta más el retraso a lo largo del camino de E a F (de nuevo debido al tráfico aumentado, las transmisiones adicionales, etc.), el control de la teleoperación y la telerrobótica se vuelve inviable como resultado de exceder el retraso a los 250 mseg. A estas alturas, el control supervisor o el control autónomo vuelven a ser la única opción.

El control autónomo también podría seleccionarse debajo de la línea de 250 mseg. en base a la consideración del coste de la asamblea, los costes operacionales y el tipo y el coste de los pasos del proceso de la asamblea. Esto se puede determinar usando la información disponible de la experiencia en la operación y el conocimiento de la asamblea y el proceso de la asamblea. Este ejemplo da directrices generales que pueden ser útiles para elegir la estrategia de control conveniente en base a las consideraciones del retraso temporal.

Un esquema para reducir la cantidad de datos que se transmiten entre los dos sitios del sistema es mediante la incorporación del control supervisado y la autonomía en el sistema remoto [Brady,98]. De esta manera, el operador interactúa con el sistema remoto a un nivel más alto que requiere interacciones de comandos comparativamente bajos. Esto significa que los sensores, las capacidades de control y las capacidades cognoscitivas del sistema remoto deben ser adecuados para realizar las tareas requeridas.

El control supervisado mejora la intervención del operador y evita los problemas de inestabilidad, donde el control del robot está compartido entre un lazo de control remoto y el operador [Sheridan,92]. Este paradigma de control no pretende que el robot realice todas las operaciones autónomamente, pero habilita al robot a realizar operaciones simples que el operador puede secuenciar. Mediante el control supervisado el usuario se puede comunicar con un sistema remoto a un nivel más abstracto enviando comandos de alto nivel al robot, el cual, al incrementar su nivel de autonomía permite disminuir la cantidad de datos a enviar. Limitar la interacción remota a comandos de alto nivel ayuda a disminuir el ancho de banda necesario, e incrementar la autonomía del robot ayuda a reducir la sensibilidad al retraso.

La figura 3.20 muestra los componentes principales del modelo utilizado de control para desarrollar el laboratorio remoto. En el laboratorio remoto propuesto, el usuario interactúa con el sistema remoto usando comandos de alto nivel que requieren bajo ancho de banda. El nivel de autonomía del robot remoto ha sido aumentado encapsulando todos los comportamientos del robot en diferentes tipos de habilidades basadas en la arquitectura de control AD propuesta por R. Barber (ver Apéndice B).

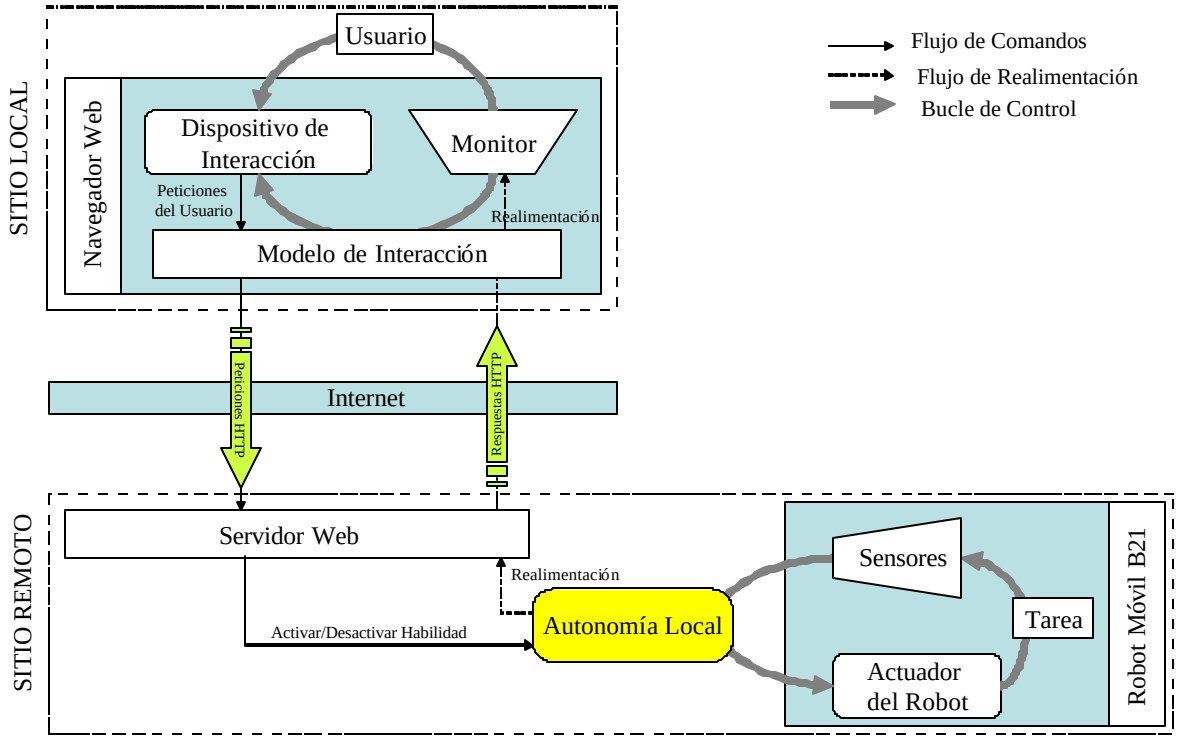


Fig. 3.20: Control Supervisado

Las subdivisiones siguientes describen el modelo desarrollado en más detalle:

- **Autonomía Local**

Para aumentar la autonomía local del robot, se ha utilizado el concepto de habilidad. Una habilidad representa la capacidad del robot de realizar una tarea particular. Las habilidades son todas las capacidades integradas de acción y de percepción del robot [Alami,98].

Según la estructura de la habilidad propuesta por Mª Jesús López, las habilidades son módulos que actúan tanto como clientes como servidores [Boada,02-a]. Son servidores porque proporcionan recursos a otras habilidades y son clientes porque solicitan servicios de otras habilidades o servidores. Así por ejemplo, la habilidad "detectar obstáculo" actúa como cliente de otros servidores al solicitar información sensorial del sonar (servidor de la base), o del láser (servidor del láser). Pero también actúa como servidor proporcionando a otra habilidad la presencia o no de obstáculos frente al robot dentro de una distancia determinada que se indica en la activación.

La figura 3.21 muestra la estructura genérica de una habilidad. Cada módulo o habilidad contiene un objeto activo, un objeto manejador de eventos y objetos de datos. El objeto activo es el encargado de llevar a cabo la ejecución de la tarea de la habilidad. A diferencia de los sensores y actuadores virtuales que están activos continuamente, las habilidades pueden ser activadas y desactivadas por otros módulos. En la estructura genérica este hecho se representa por un círculo.

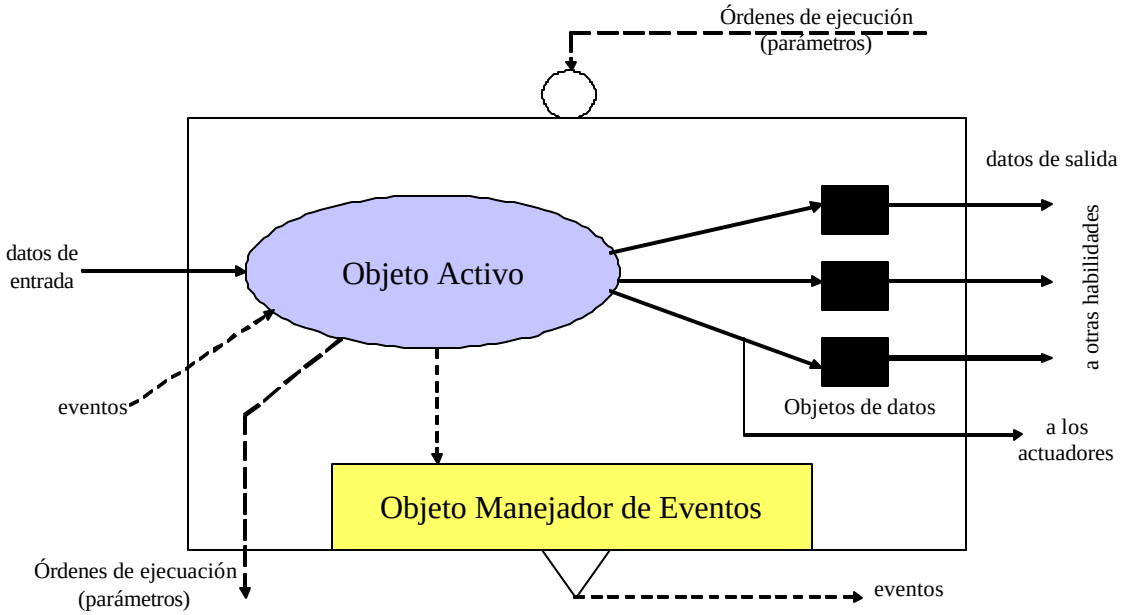


Fig. 3.21: Estructura Genérica de una Habilidad

Durante el proceso de activación (a través de las órdenes de ejecución) se pueden enviar parámetros que permitan a la habilidad ejecutarse de una manera determinada. Por ejemplo, en la habilidad "Seguir Contorno" el parámetro de activación enviado es la distancia a la que el robot debe seguir dicho contorno. Una vez activado, en cada ciclo de ejecución, el objeto activo recibe datos de entrada que procesa, almacenando los resultados obtenidos en los objetos de datos. Estos objetos contienen estructuras de datos distintas en función de los tipos de datos que se quieran guardar pero mantienen interfaces similares.

Un objeto activo puede recibir, de manera asíncrona, eventos que pueden hacer cambiar el comportamiento de la habilidad. Durante el procesamiento de los datos de entrada el objeto activo puede generar eventos que son enviados al objeto manejador de eventos que se encarga de notificarlos a todas las habilidades que se han registrado en este último. No sólo se notifica el evento producido sino también la habilidad que lo ha originado.

Cuando la habilidad es desactivada puede enviar un informe de su estado. Por ejemplo, la habilidad "Ir a un Punto" puede informar sobre el error cometido entre la posición del robot y la meta [Boada,02-b].

- **Intervención del Usuario**

En este modelo, el usuario tiene un papel supervisor por lo que él puede activar o desactivar las habilidades del robot. En el capítulo 6, se explica detalladamente el diseño y la implementación del modelo de interacción, que se usa para controlar remotamente las habilidades del robot utilizando diferentes dispositivos de interacción como PC o dispositivos móviles.

Se proporcionan varios tipos de realimentación visual, como se puede ver en el capítulo 6, para que el usuario tenga sensación visual que expresa la secuencia de sus comandos directamente en la interfaz de control. Con toda esta información el cliente bien automáticamente o por la supervisión humana, generará nuevas órdenes de control que eviten situaciones no deseadas o corrijan las acciones emprendidas. En el modelo de interacción se puede combinar o secuenciar distintas acciones o habilidades del robot, decidiéndose en el lado del cliente la activación, desactivación o prioridad de ejecución de cada una de ellas. En el capítulo 6, se presenta como se secuencian las habilidades tanto en el lado del cliente como en el lado del servidor utilizando un secuenciador.

- **Detección y Recuperación de Errores**

En este modelo de control, los errores se pueden manejar mediante tres procesos: detección autónoma, diagnóstico compartido, y recuperación manual. Estos errores se detectan usando los medios de visualización como *streaming* videos, gráficos, datos sensoriales o paneles de estados. La tarea del diagnóstico se comparte entre el usuario y el sistema. En los sistemas basados en Internet, el usuario puede telecolaborar con un hombre en el sitio remoto o puede pedir los privilegios necesarios para ser capaz de hacer *telnet* a los servidores remotos para reiniciarlos.

- **Tiempo de Retraso**

Limitando la interacción remota a comandos de alto nivel y aumentando la autonomía local del robot mediante la utilización del concepto de habilidades, se ayuda a disminuir la sensibilidad del sistema al retraso de comunicación.

De esta manera, se pretende calcular el tiempo total de retraso en la comunicación, que es el tiempo entre tomar la decisión de activar una habilidad hasta el comienzo de la ejecución de esta habilidad. Como se muestra en la figura 3.22, el tiempo de retraso consta de dos componentes principales:

t_1: tiempo de ciclo entre el cliente y el servidor middleware.
t_2: tiempo de ciclo entre el middleware y el servidor remoto.

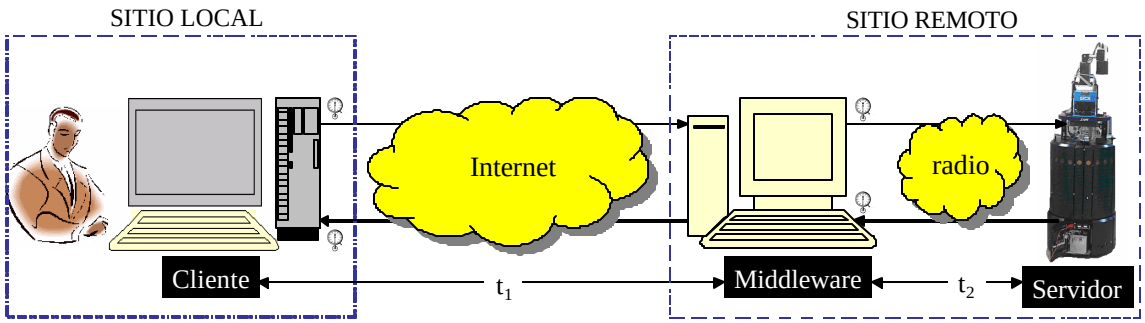


Fig. 3.22: Tiempo de Retraso entre el Cliente y el Servidor

El tiempo total de retraso es la suma de estos dos componentes, $T = t_1 + t_2$.

En las subsecciones siguientes se presentan los resultados obtenidos de una habilidad motora (Control Directo). Está habilidad es una habilidad automática muy simple que permite mover el robot hacia delante o hacia atrás o girarlo en sentido horario o antihorario.

➢ *Tiempo de Ciclo entre el Cliente el Servidor Middleware (t_1)*

Para medir el retraso temporal entre el cliente y el middleware, se han insertado dos relojes en el programa del cliente. Estos dos relojes miden la diferencia de tiempo entre enviar una petición desde el cliente al servidor de middleware hasta que el cliente recibe la respuesta. Se puede considerar esta diferencia como retraso de ciclo entre el cliente y el middleware. La figura 3.23 muestra la interfaz gráfica de usuario de la habilidad motora "control directo" que tiene una opción para mostrar la variación del retraso entre el cliente y el middleware.

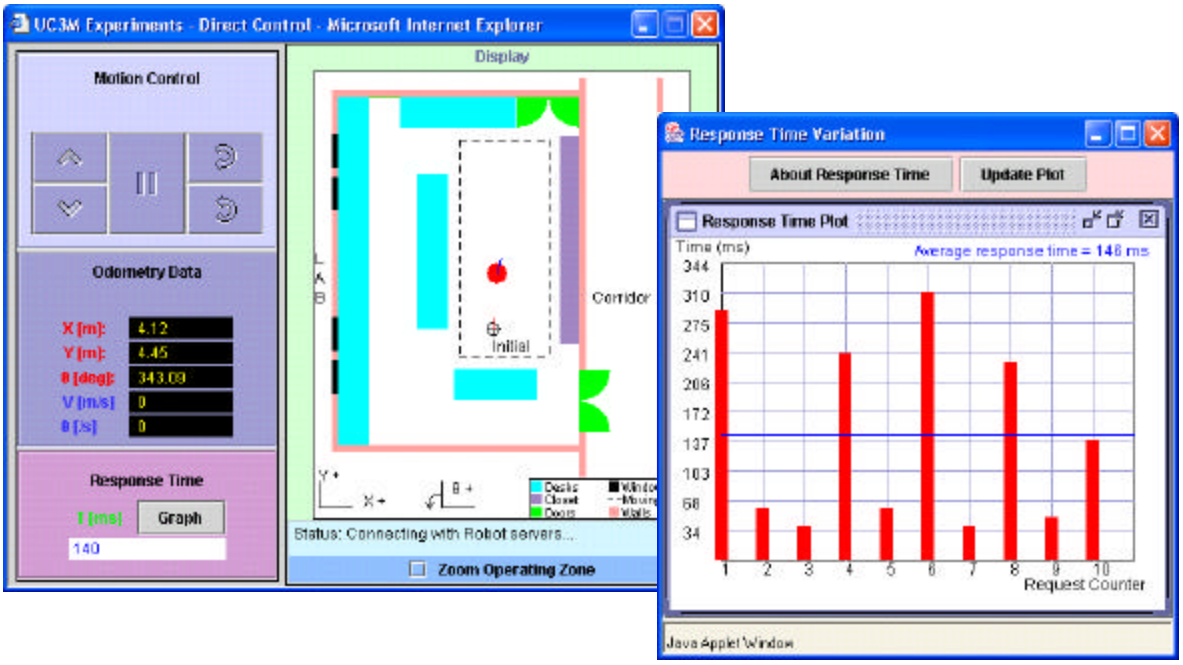
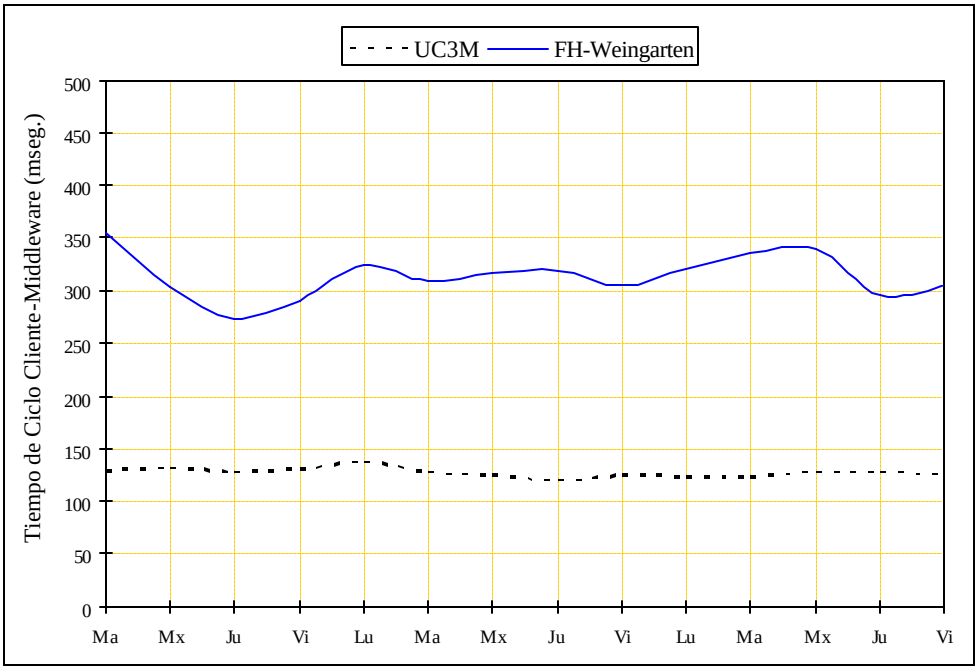


Fig. 3.23: Control Directo

Para conseguir una estimación sobre este retraso, se ha activado esta habilidad desde sitios diferentes. La figura 3.24 muestra una comparación entre el retraso de ciclo entre el cliente y middleware medido desde FH-Weingarten, Alemania y desde la UC3M como sitios de cliente. En ambos casos, el servidor de middleware se encuentra en la UC3M, por eso se nota que el retraso medido desde la UC3M es más bajo que el retraso medido desde FH-Weingarten. El retraso medido desde FH-Weingarten (medio 313,4 mseg.) se notaba por el usuario pero se puede aceptar para las aplicaciones educativas de laboratorios remotos.

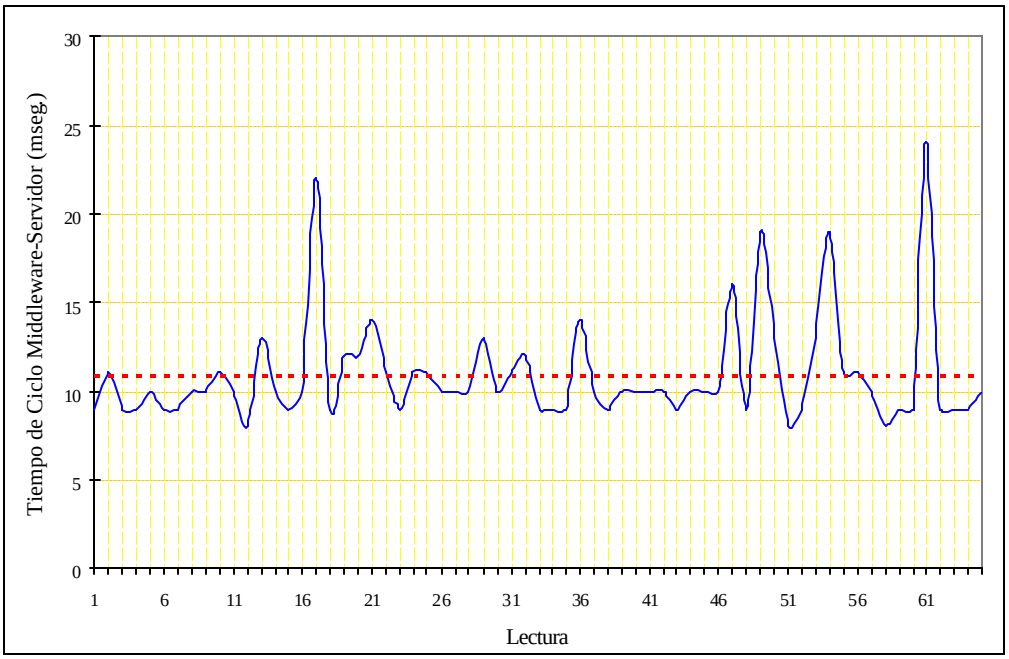


Host	Mín.	Medio	Máx.
UC3M	121	127,2	137,8
FH-Weingarten	273	313,4	354

Fig. 3.24: Tiempo de Ciclo Cliente-Middleware desde FH-Weingarten & UC3M en msg.

- ***Tiempo de Ciclo entre Middleware y el Servidor Remoto (t_2)***

De la misma manera, dos relojes se han insertado en el servidor de middleware para medir la diferencia entre enviar la petición al servidor remoto y recibir la respuesta. Esta diferencia de tiempo se puede considerar como retraso de ciclo entre el middleware y el servidor. La figura 3.25 muestra los resultados de las medidas. Se toma una medida cada 10 minutos.

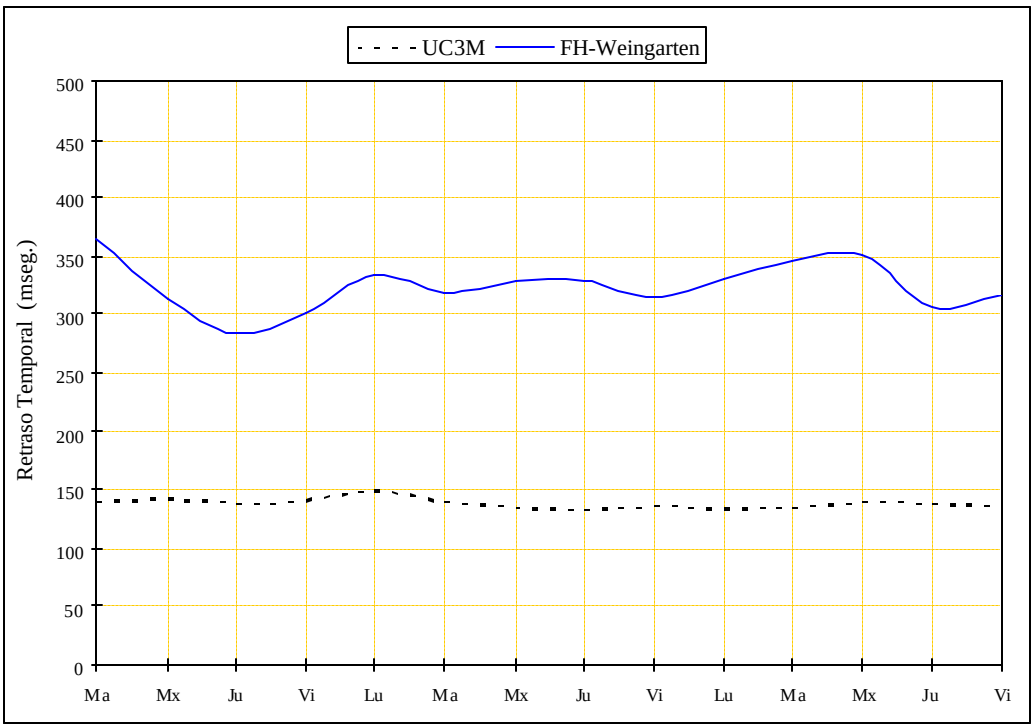


Mín.	Medio	Máx.
8	10,89	24

Fig. 3.25: Tiempo de Ciclo Middleware-Servidor en mseg.

Estos resultados muestran que la conexión entre el middleware y el servidor (conexión radio de 1Mbits/s) no impone un retraso significante. El valor medio de este retraso (10,89 mseg.) se puede considerar como un retraso de ciclo constante entre el middleware y el servidor.

El retraso de tiempo total es la suma de los dos retrasos considerando que el tiempo de ciclo entre el middleware y el servidor es constante como se indica en la figura 3.26. En esta figura se muestra una comparación entre el retraso de tiempo total medido desde FH-Weingarten y desde la UC3M.



Host	Mín.	Media	Máx.
UC3M	131,89	138	148,65
FH-Weingarten	283,97	324,3	364,89

Fig. 3.26: Retraso Total en mseg.

Las medidas del retraso de comunicación dan una buena estimación sobre el efecto de Internet. Como se ha mencionado anteriormente, estos retrasos son inevitables y cambian aleatoriamente según la carga de la red. Sin embargo, los valores medios (138 mseg. desde UC3M y 324,3 mseg. desde Weingarten) representan valores aceptables para una aplicación como los laboratorios remotos teniendo en cuenta que normalmente se puede considerar un segundo como una referencia de operabilidad para el retraso máximo de comunicación en los sistemas de teleoperación.

Capítulo

4

LABORATORIOS REMOTOS

Capítulo 4

LABORATORIOS REMOTOS

4.1 INTRODUCCIÓN

Hoy en día el avance de las tecnologías de la información es imparable, los rápidos progresos tanto en la informática como en las comunicaciones están provocando una revolución en la sociedad moderna. Este nuevo entorno tecnológico está cambiando tanto la manera de afrontar los problemas como la forma de resolverlos.

Toda esta revolución se ha visto acentuada con el rápido crecimiento de Internet, y la enorme aceptación que ha tenido en todos los sentidos. En España, el uso de Internet ha crecido un 11% entre el 2000 y 2002 según un estudio de *Taylor Nelson Sofres Interactive* [TNSofres,03] como se muestra en la figura 4.1.

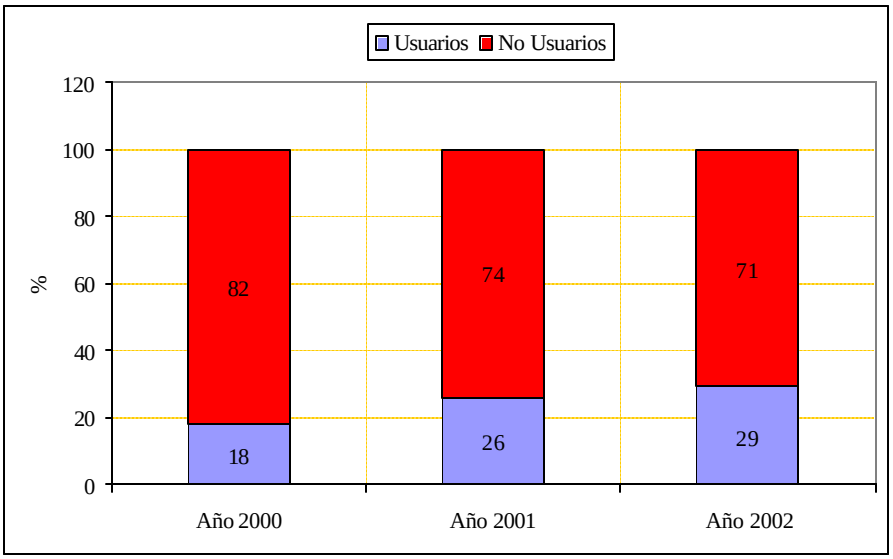


Fig. 4.1: Porcentaje de Población de Usuarios de Internet en España

La red de redes facilita la interconexión de personas, empresas e instituciones de cualquier lugar del mundo, con la eliminación de las barreras (administrativas, políticas, sociales,...) que eso conlleva. Junto con otros aspectos, Internet permite un rápido acceso a la información más reciente, y a los avances realizados por los investigadores más destacados de cualquier campo académico, profesional o de investigación.

Pero es dentro del campo educativo donde Internet debe tener gran importancia y una tarea que cumplir, puesto que no todas las instituciones de educación superior cuentan con recursos humanos ni materiales suficientes para afrontar de manera eficiente la formación de nuevos ingenieros e investigadores. El espacio virtual que Internet nos ofrece, permite aprovechar de modo más eficaz los recursos con los que cuentan unos pocos laboratorios al hacerlos accesibles al resto de la comunidad investigadora. Por eso en el ámbito de la docencia el papel de Internet puede ir mucho más allá que el de simple distribuidor de información, ya que las posibilidades que nos ofrece para comunicarnos de forma bidireccional y acceder a infraestructuras remotas la dotan de un valor añadido que sirve de complemento a las metodologías de enseñanza tradicionales. Sin embargo, sólo unos pocos ejemplos de interacción real con entornos remotos están disponibles hoy en día.

En el presente capítulo se analiza detalladamente el diseño y el funcionamiento de los laboratorios remotos con un enfoque especial sobre los laboratorios de robótica móvil. En primer lugar, se explica qué es un laboratorio remoto y qué es un laboratorio distribuido y para qué sirven. En la sección 4.3, se presenta el estado del arte de los laboratorios remotos y distribuidos y las distintas arquitecturas utilizadas para implementar estos laboratorios. A continuación, en la sección 4.4, se plantea una metodología por medio de la cual se puede construir entornos educativos innovadores para la robótica móvil.

4.2 LABORATORIOS REMOTOS Y DISTRIBUIDOS

Los laboratorios remotos pueden ser una posible solución a los problemas que existen en los laboratorios experimentales universitarios. Un laboratorio remoto es un laboratorio basado en una red de comunicaciones, donde el usuario y los equipos del laboratorio están geográficamente separados y donde las tecnologías de las telecomunicaciones se usan para permitir a los usuarios acceder a los equipos del laboratorio. Este tipo de laboratorios requiere una conexión permanente entre el usuario y el sitio Web durante el experimento y tiene la ventaja de no estar restringido a la asistencia sincronizada por instructores y estudiantes, es decir, tienen la potencia de proporcionar acceso constante siempre y cuando sea necesario.

Se pueden reunir muchos laboratorios remotos formando un entorno de trabajo llamado laboratorio distribuido que se puede considerar como una red de laboratorios remotos. Un laboratorio distribuido es un entorno heterogéneo de resolución de problemas que permite a un grupo de investigadores, diseminados por todo el mundo, trabajar juntos en un conjunto común de proyectos mediante las tecnologías de información. Para ello se sirven, como en cualquier otro laboratorio, de las herramientas y técnicas específicas del dominio de investigación, pero los requisitos de infraestructura básica se comparten entre las distintas disciplinas. El proyecto IECAT en que participa la Universidad Carlos III de Madrid es un ejemplo de una red de laboratorios remotos en el campo de la mecatrónica aplicada a la robótica y las tecnologías aeroespaciales [IECAT,03].

4.2.1 Ventajas

Los sistemas basados en Internet tienen cada vez mayor aceptación, sobre todo debido a las grandes ventajas que aportan de cara al usuario. Entre las más importantes cabe citar su accesibilidad desde cualquier computador conectado a Internet en cualquier momento.

En [Moreno,01], se han resumido las características que presenta el acceso remoto a través de Internet en los siguientes puntos:

- Estos laboratorios pueden estar accesibles las 24 horas del día, todos los días del año. Como se ha mencionado estos laboratorios tienen la ventaja de no estar restringidos a la asistencia sincronizada por instructores y estudiantes, es decir, ellos pueden proporcionar acceso constante siempre y cuando sea necesario. El estudiante dispone de su tiempo de una forma más personal y libre, ajustando su ritmo de estudio a su vida laboral y familiar y no al contrario, tal y como sucede habitualmente.
- No hay barreras físicas. Los estudiantes no tienen que desplazarse al centro para la realización de las actividades prácticas. Pensemos principalmente en los modelos educativos basados en la enseñanza a distancia, en los que los desplazamientos al centro de práctica son de varios cientos de kilómetros y por varios días. Si recordamos que uno de los objetivos de los estudios a distancia es posibilitar a los colectivos que realizan actividades laborales y que, por lo tanto, no disponen de excesivo tiempo libre asistir a unas sesiones presénciales, los laboratorios remotos y distribuidos constituyen la solución idónea para la enseñanza a distancia con experimentalidad.
- Optimización en el aprovechamiento de los recursos. Entender la accesibilidad temporal con independencia de la ubicación del operador deriva en un mejor aprovechamiento de los elementos implicados en el laboratorio, con lo que la relación dinero invertido/tiempo de utilización mejora ostensiblemente. Por otra parte, conviene observar que la construcción de laboratorios remotos y distribuidos no requiere de grandes superficies y locales ya que no es necesario ubicar a los estudiantes; sólo es necesario el espacio del instrumental que compone el experimento y las distancias de seguridad que marquen las normas de prevención de accidentes.
- Accesibilidad a diferentes tipos de experimentos con independencia de que los recursos de un centro sean escasos. Un símil relativo a este punto es la consulta de los fondos bibliográficos o documentales de otras universidades o centros de investigación que se hace vía Web o el acceso a los recursos de los centros de supercomputación mediante Telnet o terminales X para la ejecución de programas con gran carga computacional.
- Preparación previa por parte del alumno de sus experimentos en caso de que el acto de presencia en el laboratorio sea ineludible. No sólo los laboratorios remotos constituyen una magnífica herramienta para la preparación de las actividades prácticas, sino también posibilita la realización de una fase de preparación en aquellas situaciones en las que el estudiante debe acudir inexcusablemente al laboratorio para demostrar sus conocimientos. Por otra parte, el tiempo de preparación o familiarización que se emplea en el laboratorio se reduce notoriamente al haber realizado de antemano cierta parte de las actividades prácticas.

- El proceso de aprendizaje se potencia al poder establecer un nexo constante entre experimentalidad y teoría ya que el estudiante puede poner en práctica los conocimientos que son transmitidos en clase sin esperar al período de práctica. No tiene que limitarse a la realización de las actividades prácticas fijadas por el equipo docente sino que puede innovar libremente.
- Adaptación y personalización del entorno del laboratorio a alumnos con discapacidades. Es viable que personas con minusvalías accedan a los laboratorios desde su hogar con interfaces de experimentación especialmente adaptadas a su problemática particular.

4.2.2 Inconvenientes

Existen todavía ciertos inconvenientes y desafíos que están limitando la utilización masiva de los laboratorios remotos y distribuidos en la actualidad. Entre los cuales cabe destacar los siguientes:

- Estos laboratorios suelen requerir una conexión permanente entre el usuario y el sitio Web durante el experimento, y por eso su rendimiento depende del rendimiento de la comunicación que no es predecible y depende de la carga de la red. Además el retraso temporal es inevitable y aleatorio. También el ancho de banda limitado no permite transmitir gran cantidad de información y limita la emisión de audio y vídeo.
- Falta de un contacto físico con el experimento. La separación espacial del estudiante con el laboratorio puede llevarle a una pérdida de la sensación de realismo práctico. Las soluciones a este factor desmotivador pasan por una correcta utilización de los componentes multimedia como el vídeo y el audio en directo o los escenarios 3D de forma que se incremente la sensación de telepresencia [Moreno,01]. En el desarrollo de nuestro laboratorio remoto se ha propuesto una solución a este problema. Esta solución está basada en una metodología que intenta combinar las actividades de instrucción con las actividades de construcción para construir entornos educativos innovadores como se va a explicar con más detalles en la sección 4.4.
- La plataforma de experimentación debería tener la flexibilidad suficiente para permitir a los usuarios diseñar sus propios experimentos.
- El equipo debe ser más robusto de lo necesario en los experimentos normales.
- Para asegurar alta disponibilidad, el equipo básico debería estar dedicado exclusivamente al uso remoto. Equipos caros muy especiales para usuarios avanzados tendrán que ser compartidos.
- Un esfuerzo considerable debe dirigirse a la interfaz entre hombre/máquina para facilitar una interacción eficiente. El usuario debe superar una fase de entrenamiento previa, siendo necesario la implementación de un control de acceso.
- El sistema debe tener mecanismos extensivos de seguridad. Incluso un usuario entrenado, tiene poca capacidad para detectar situaciones de peligro y sus consecuencias.
- Necesidad de un cambio de mentalidad en el docente y en discente [Astroza,00]. Pese a la creciente implementación en la sociedad de nuevas formas de interacción y comunicación (teléfonos móviles, Internet, televisión interactiva, video conferencia, etc.), todavía un

porcentaje elevado de la sociedad permanece de espaldas a esta inevitable realidad. Una parte de ella es el sector docente universitario que, pese al notable esfuerzo que está realizando, todavía está muy por debajo en el campo de estas tecnologías en comparación con otros campos, como el bancario [Moreno,01]. Este cambio de mentalidad, que es necesario, se hace más patente en el profesor que en el alumno, el cual, por razón de su juventud, está más habituado al empleo de estos nuevos medios. Según un estudio de *Taylor Nelson Sofres Interactive* [TNSofres,03] sobre los grupos de usuarios de Internet en España y sus sexos, el crecimiento más grande en el uso de Internet como se ve en la figura 4.2 ha sido entre los grupos de usuarios de edad menor que 20 años reflejando el futuro brillante de Internet.

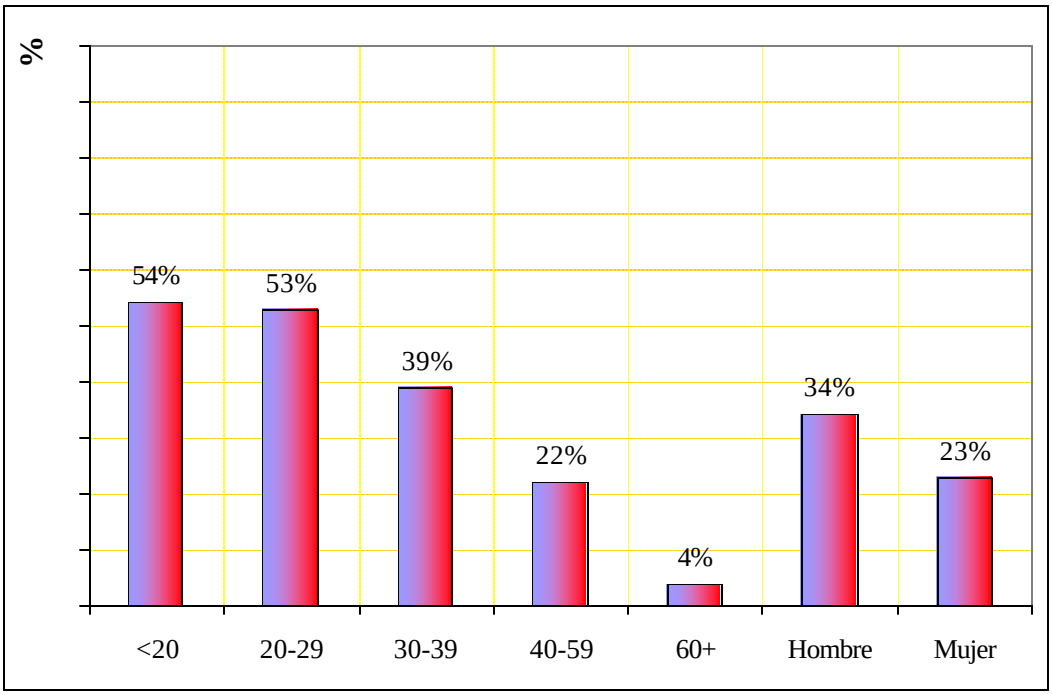


Fig. 4.2: Porcentaje de Grupos de Edad Específica y Sexos de los Usuarios de Internet en España (2002)

4.3 ROBÓTICA EN LÍNEA

Los robots remotamente controlados se desarrollaron en los años cuarenta y fueron usados por expertos especializados en espacio, submarinos, y en el sector nuclear. Una nueva clase de telerobots es ahora accesible vía Internet. Los robots en línea (*Online Robots*) permiten a cualquiera en Internet controlar un robot a distancia, por ejemplo para visitar un museo, cuidar un jardín, moverse bajo el mar, viajar en globo, o manipular cristales de proteínas. Los robots en línea apoyan la educación y el conocimiento público haciendo el hardware valioso del robot accesible al público. Están apareciendo nuevas aplicaciones para la investigación y la educación, así como para el entretenimiento.

Los robots en línea tienen unos desafíos técnicos. Además de los problemas asociados con el retraso temporal y la estabilidad, los robots en línea deben diseñarse para ser manejados por personas no especialistas utilizando interfaces de usuario intuitivos y accesibles 24 horas a día.

Métodos nuevos son necesarios para la coordinación de usuarios simultáneos, enfrentándose con variaciones grandes en la demanda y el retraso temporal, y para la detección y la recuperación de errores.

En los últimos años, las aplicaciones robóticas basadas en Web [Taylor,00-a] han crecido de tal manera, que la mayoría de los laboratorios importantes tienen aplicaciones para acceder a sus robots desde Internet. Actualmente existen muchas tecnologías diferentes que permiten a un robot ser teleoperado a través de la red. El término telerrobótica Web normalmente se refiere a aplicaciones de telerrobótica basadas en Web, es decir, aplicaciones de teleoperación que se pueden ejecutar dentro, o se lanzan, desde un navegador Web. Esta sección describe las tecnologías involucradas en algunas aplicaciones basadas en Internet con un enfoque especial sobre la robótica móvil.

4.3.1 Laboratorios Remotos

Los laboratorios remotos ponen al alcance de los usuarios una serie de experimentos que físicamente se encuentran alejados, haciéndolos accesibles vía Internet. El campo de la robótica siempre ha estado al alcance de unos pocos, en su mayoría investigadores, debido al alto coste del equipamiento. En la actualidad, gracias a la aparición de los laboratorios remotos, cualquier persona que cuente con una conexión a Internet puede operar un robot, por medio de un ordenador personal, un teléfono móvil, una PDA, etc. algo que hace algunos años era impensable. En las subsecciones siguientes, se hace un repaso sobre el estado del arte del tema de los laboratorios remotos de robótica en general y la robótica móvil con un enfoque especial.

4.3.1.1 Robótica

En el 24 de Abril de 1997, en la Conferencia Internacional sobre Robótica y Automatización en Alburquerque, Kevin Brady, un estudiante de doctorado del profesor T.J. Tarn de la Universidad de Washington, usó un joystick en Alburquerque para controlar los movimientos de un robot Puma en el laboratorio de Tarn a más de 1,500 kilómetros de distancia [Brady,98]. Estando en pie sobre la tarima, Brady trabajó con el robot frente a centenares de ingenieros que lo miraban mediante un monitor de vídeo en la conferencia de la Sociedad de Automatización y Robótica del IEEE. El experimento de tres minutos era una tarea difícil para un robot. Involucraba el transporte de una pieza de un sitio a otro evitando una caja. Mientras que el movimiento de una pieza es algo rutinario en robótica, evitar la caja y mandar los comandos desde una red pública remota fue extraordinario. "Esto no era un experimento de realidad virtual, sino una demostración en vivo, que muestra una aproximación de la robótica a Internet y otros sistemas remotos de comunicaciones", dice Tarn [Fitzpatrick,99]. Este experimento ha abierto la puerta a nuevas posibilidades en el aprendizaje a distancia, oportunidades de investigación y experimentación en-línea vía Internet. Entre las aportaciones más conocidas en el campo de robótica basada en Internet cabe citar las siguientes aplicaciones:

- **Telerobot de UWA**

El telerobot de la Universidad del Oeste de Australia, es un brazo robótico ABB IRB 1400. Los usuarios operadores del robot envían los movimientos solicitados rellenando un formulario HTML, seleccionando puntos en imágenes del entorno de trabajo, o especificando varios

movimientos en forma de fichero de comandos. Además, existen usuarios observadores que no controlan el robot, pero observan lo que hace [Taylor, 00-b].

Como se muestra en la figura 4.3 para controlar el robot hay que someter un formulario HTML a un servidor Web que recibe la petición y lanza un comando CGI (*Common Gateway Interface*). Se lanza un CGI para cada petición, y varias copias pueden ejecutarse simultáneamente para servir a un operador y muchos observadores. Sólo una persona a la vez puede controlar el robot. El guión CGI determina si la petición ha venido de un operador, y en tal caso, establece la comunicación con el servidor de robot y el servidor de imagen. En la figura 4.4 se muestra la interfaz de usuario de este sistema.

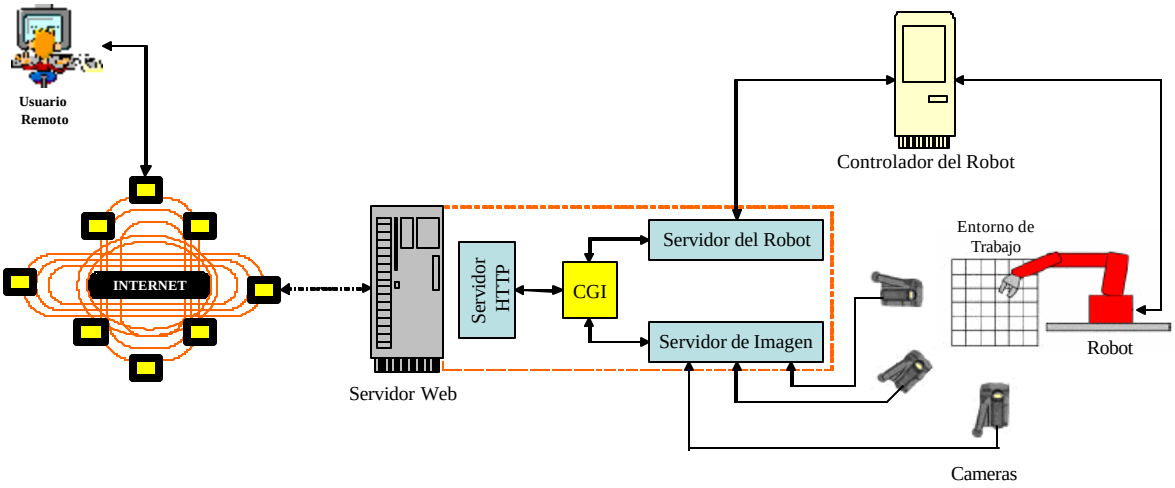


Fig. 4.3: Arquitectura de Sistema del Telerobot UWA

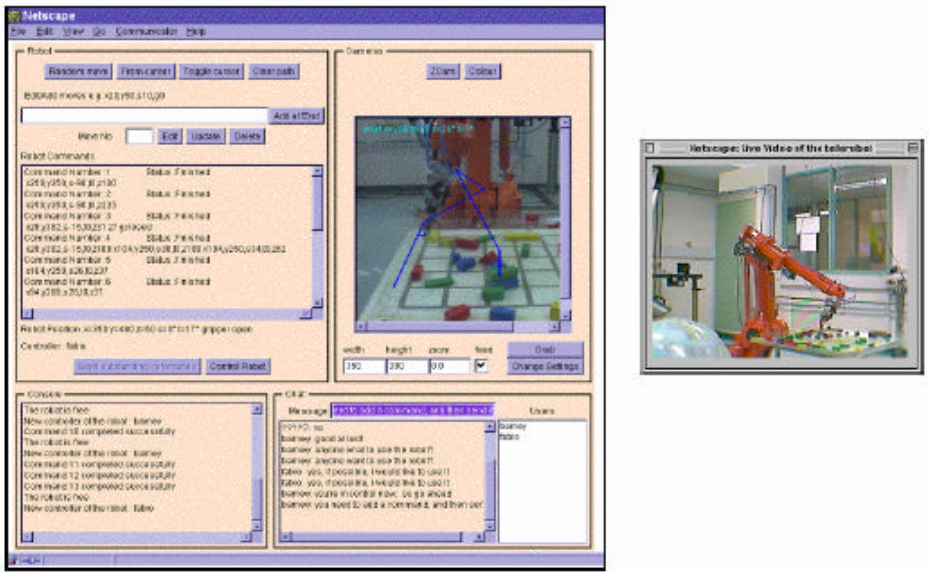


Fig. 4.4: Interfaz de Usuario del Telerobot UWA

Otras tecnologías como JavaScript y Java han sido usadas en este sistema. JavaScript proporciona la capacidad para el procesamiento local de los datos en vez de utilizar CGI para el manejo de la interfaz. Java se usa para producir un modelo que permite a los usuarios simular movimientos antes de enviarlos al robot real.

▪ Mercury Project

El Mercuy Project de la Universidad del Sur de California se considera como el primer sistema que permitía a cualquier usuario de Internet alterar un entorno lejano. La iniciativa tiene mucho éxito y, entre 1994 y 1995, 2,5 millones de usuarios se conectaron a este sistema, con el objetivo de buscar en la arena los misteriosos objetos y así descubrir el misterio oculto en un clásico de la literatura. Este proyecto combina la robótica y la arqueología en una instalación de arte interactivo. Los usuarios debían localizar los artefactos enterrados en un terrario, todos ellos relacionados con un texto anónimo del siglo diecinueve [Mercury,03]. Los usuarios fueron desafiados para identificar este texto describiendo sus conclusiones e hipótesis. La arquitectura del sistema (véase la figura 4.5) es similar a la del sistema anterior y su interfaz de usuario se puede ver en la figura 4.6.

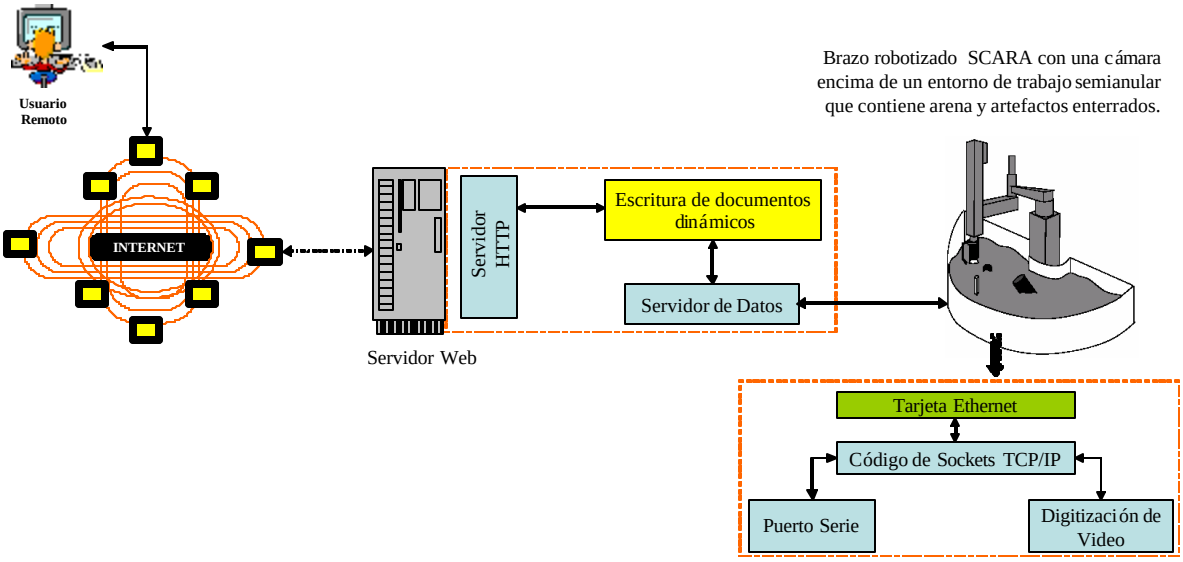


Fig. 4.5: Arquitectura del Mercury Project

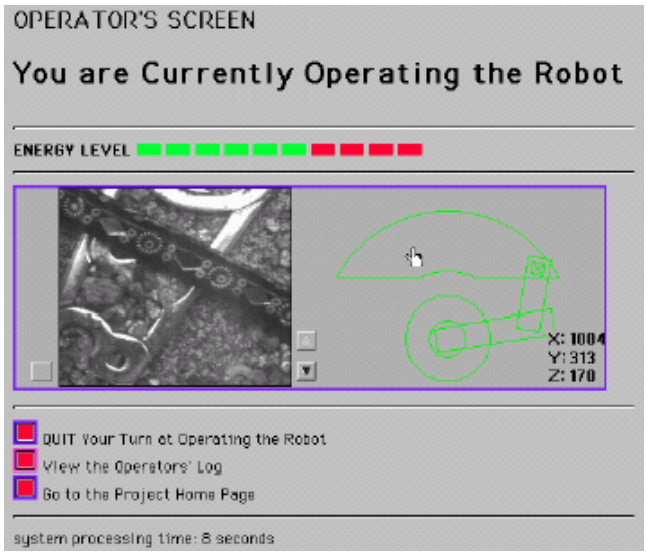


Fig. 4.6: Interfaz de Usuario del Mercury Project

- **Telegarden**

Basándose en la misa arquitectura del Mercury Project, se desarrolló en la misma universidad el Telegarden para permitir a los usuarios ver e interactuar con un jardín remoto lleno de plantas vivas a través de Web [Telegarden,03]. Los miembros del sitio pueden plantar, regar, y supervisar el progreso de los árboles usando los movimientos de un brazo de un robot industrial como se muestra en la figura 4.7.

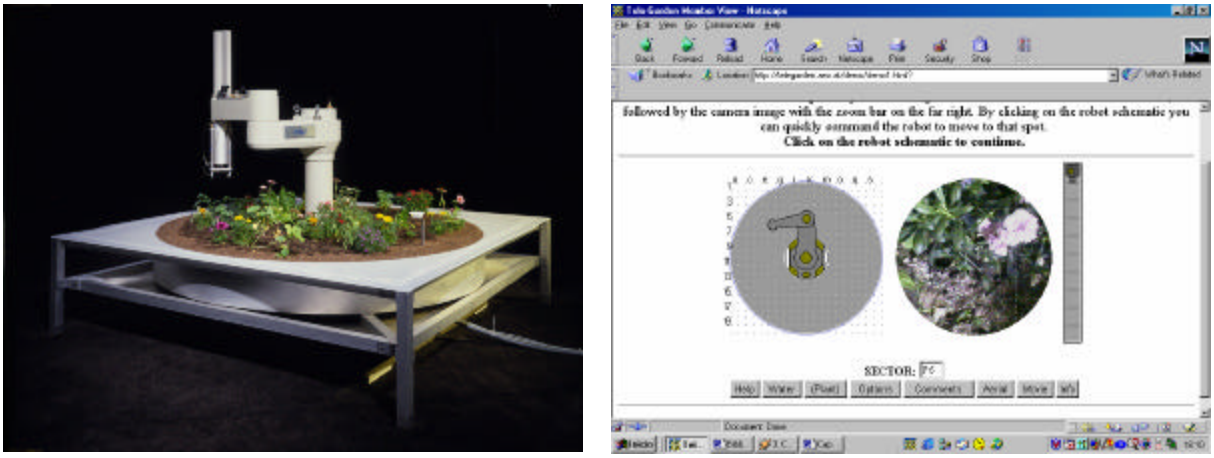


Fig. 4.7: Telegarden

- **RoboLab**

El laboratorio de la Universidad de Alicante (véase la figura 4.8) permite teleoperar el brazo robótico Scorbot ER-IX a través de Internet con la ayuda de modelos de realidad virtual. Un usuario puede acceder a todas las funciones del laboratorio virtual a través de una página Web con un applet Java. Además, para la simulación gráfica se emplea VRML, por ello, es necesario que el usuario disponga del software apropiado instalado en su ordenador, esto es, un navegador cliente con soporte para Java y VRML [Torres,02].

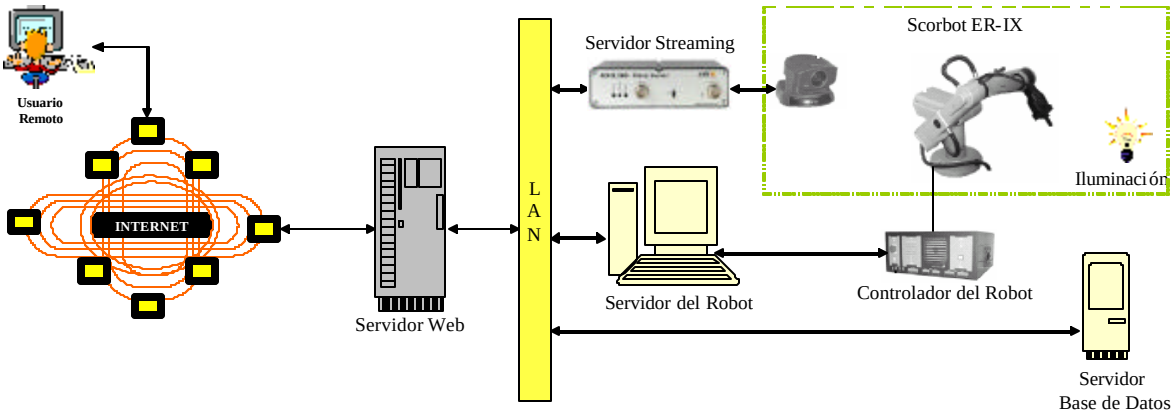


Fig. 4.8: Arquitectura del RoboLab

La interfaz de usuario (véase la figura 4.9) se compone de dos partes: el applet Java con las diferentes opciones para manejar la simulación local del robot, y una ventana VRML con el estado simulado del mismo.

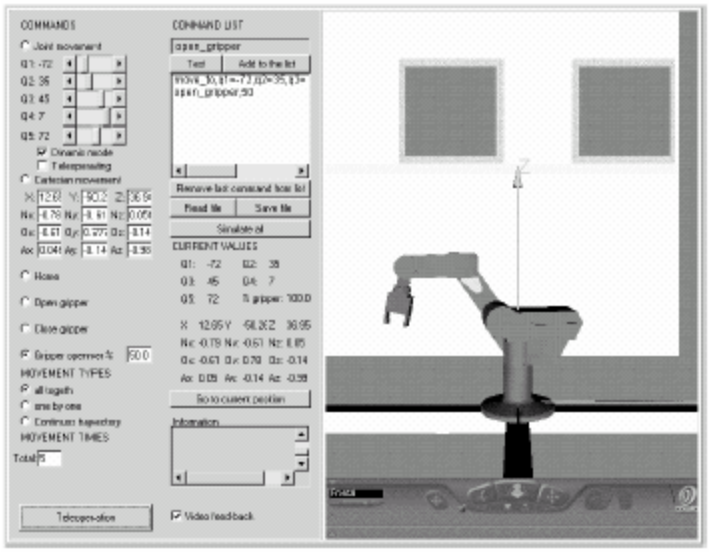


Fig. 4.9: Interfaz de Usuario del RoboLab

Con la simulación, un usuario puede determinar si la ejecución de una serie de comandos es válida y no crea problemas en el robot. Después de realizar una simulación y obtener una lista de comandos válidos, el usuario puede ejecutar la opción de teleoperación, solicitando al servidor Web que se ejecuten los movimientos de la lista en el robot real.

- **Augmented Reality Interface for Telerobotic Applications via Internet (ARITI)**

El sistema ARITI tiene la arquitectura mostrada en la figura 4.10. Un robot esclavo de 4 GDL que contiene una base, la cual puede moverse sobre dos ejes mediante una transmisión al suelo. Todos los movimientos del robot son posibles usando motores paso a paso. Se usa una tarjeta de adquisición de vídeo Matrox Meteor conectada a una cámara blanco y negro para obtener imágenes en tiempo real. El servidor de vídeo está escrito en C estándar y el servidor de control en C y ASM (Lenguaje ensamblador de Microprocesador) [ARITI,03].

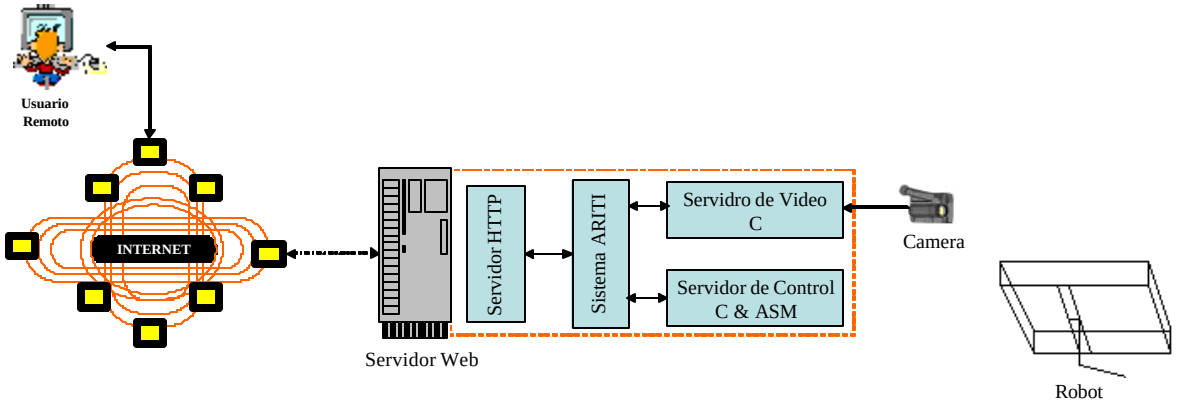


Fig. 4.10: Arquitectura de ARITI

La interfaz de usuario de ARITI (véase la figura 4.11) está escrita en Java y basada en el concepto de la realidad aumentada, para permitir a los usuarios controlar el robot remotamente, utilizando cualquier navegador Web.

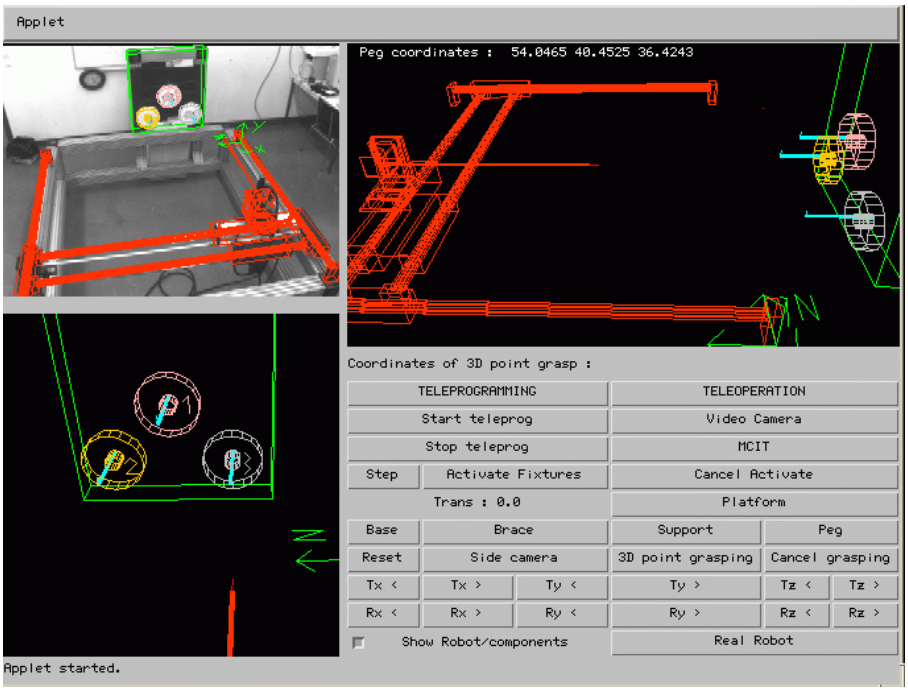


Fig. 4.11: Interfaz de Usuario de ARITI

- **El Proyecto PumaPaint**

El proyecto PumaPaint permite a cualquier usuario dibujar pinturas mediante el uso de un manipulador PUMA ubicado en la universidad Roger Williams [Stein,00]. Se ha utilizando el concepto de la teleprogramación para desarrollar este sistema. Los operadores utilizan una representación virtual del sitio remoto para enviar una serie de comandos de interacción. La figura 4.12 muestra el sitio remoto y la interfaz de usuario implementada en Java.

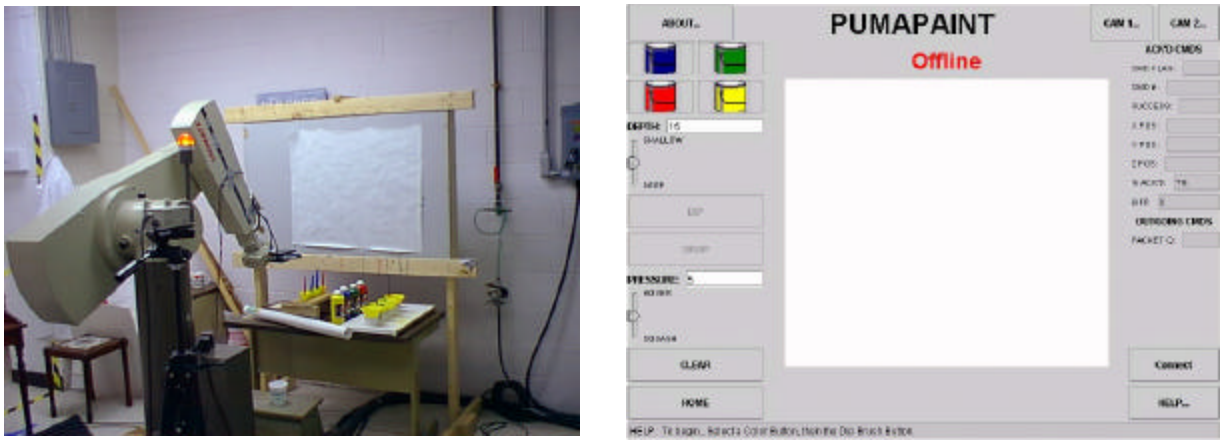


Fig. 4.12: El Proyecto PumaPaint

4.3.1.2 Robótica Móvil

Especial interés tienen los experimentos basados en robótica móvil, puesto que representan sistemas reales complejos, que reúnen una gran variedad de conceptos. Para analizar la dificultad que significa elaborar un laboratorio remoto para robótica móvil se estudia la dificultad que conlleva la realización de distintos experimentos de forma remota.

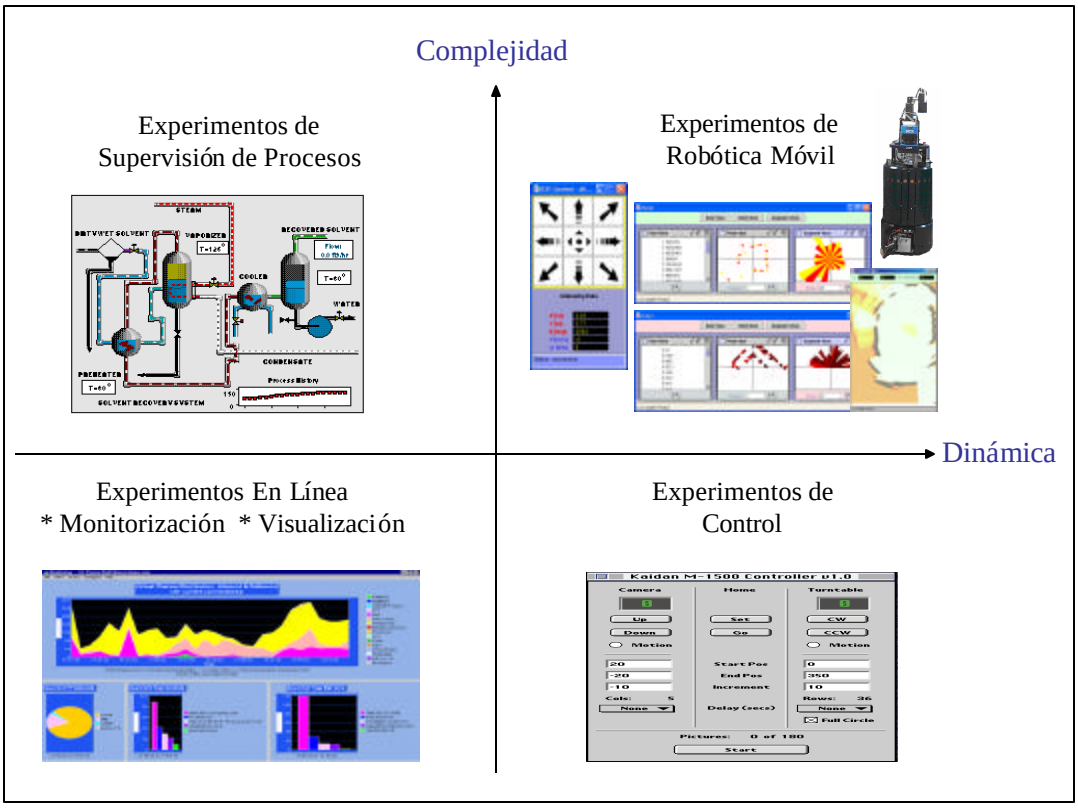


Fig. 4.13: Tipos de Experimentos remotos

Como se muestra en la figura 4.13, los experimentos remotos se pueden clasificar en los siguientes tipos:

- Experimentos en línea como monitorización o visualización. Son los más sencillos de implementar, puesto que son experimentos con escasa dinámica y baja complejidad.
- Experimentos de control de procesos, que aún careciendo de una dinámica importante, son mucho más complejos que los anteriores por el número de variables.
- Experimentos de control. Su dificultad viene dada por tener una alta dinámica, por lo que el retraso temporal que introduce Internet debe ser muy bien estudiado. El muestreo de las variables de salida para el posterior control de las variables de entrada ha de hacerse en tiempo real, para que el sistema no se desestabilice.
- Experimentos en robótica móvil que son los más complicados de desarrollar, por poseer una gran dinámica y tener una alta complejidad debido al alto numero de variables a controlar. Los robots móviles van acompañados de gran número de sensores y actuadores sobre los que se puede actuar.

Como resultado de la simplicidad y popularidad de la Web, y su independencia geográfica, temporal y de la plataforma, muchos investigadores han comenzado a desarrollar sistemas basados en Web para robots móviles. Entre las aplicaciones más destacadas en este campo cabe citar las siguientes:

- **Xavier de CMU**

El sistema Web del robot Xavier de la Universidad Carnegie Mellon, se desarrolló como una ayuda para probar algoritmos de navegación [Simmons,00]. Un sitio Web ha sido diseñado para que los usuarios pudieran enviar tareas al robot y tener alguna realimentación sobre el grado de éxito de la tarea. El experimento consiste en un robot autónomo móvil que se mueve en un edificio real. Es un robot móvil (0.6 m en diámetro) basado en una base de RWI (Real World Interface), que tiene un juego completo de sensores (parachoques, codificadores, láser y ultrasonidos) y una cámara CCD en color.

El método de control de Xavier (Figura 4.14) es muy simple y está basado en páginas HTML simples. Como en el sistema de telerobot de Australia, existen dos interfaces diferentes: la interfaz de operador (Figura 4.15-a) y la interfaz de observador (Figura 4.15-b). Cuando un observador se une al sistema, el servidor crea un proceso de CGI nuevo, que continuamente actualiza el vídeo en la interfaz de usuario. Además del vídeo se proporciona al usuario un mapa del edificio y la posición actual del robot. La interfaz de operador es solamente otro formulario HTML donde el usuario puede seleccionar la tarea requerida y luego enviar el comando a un CGI. Cuando la tarea se completa, el usuario es notificado mediante un correo electrónico. Hay tres tipos de tareas: gastar una broma, decir ¡hola!, o tomar una foto. En todos los casos, el usuario selecciona la habitación de destino donde la tarea debe ser realizada.

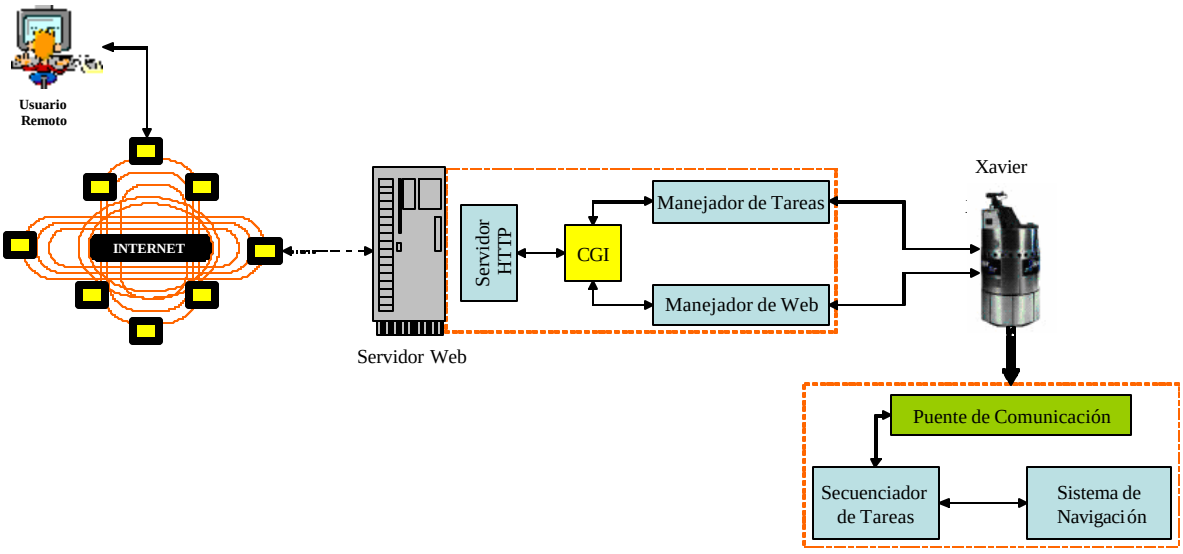
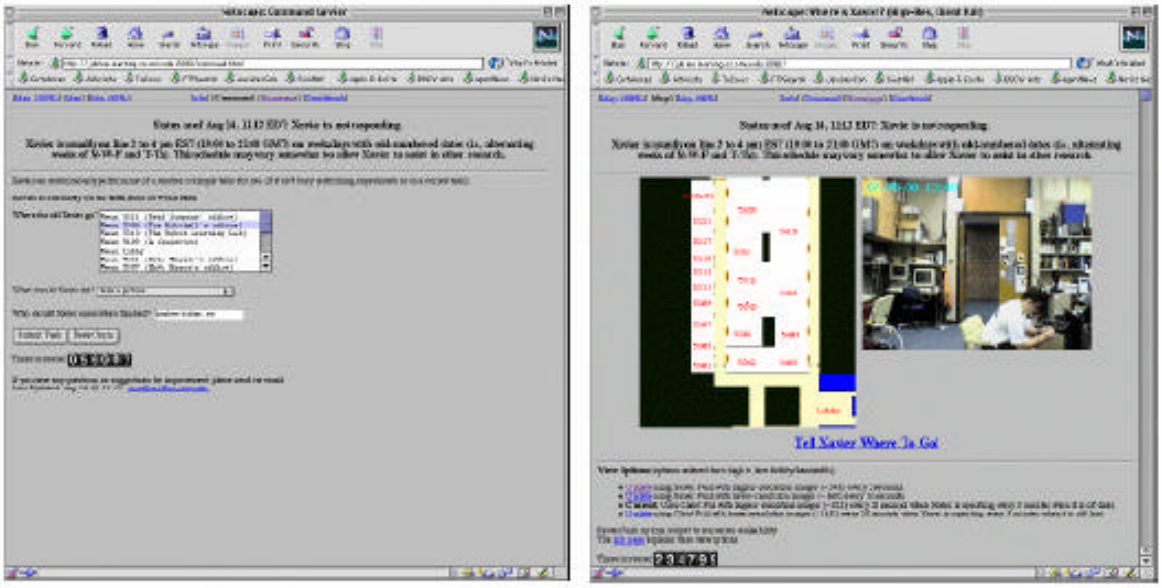


Fig. 4.14: Arquitectura del Sistema de Xavier



a) Interfaz del Operador b) Interfaz del Observador

Fig. 4.15: Interfaz de Usuario de Xavier

El experimento de Xavier tiene varios problemas: inexactitud de los sensores, obstáculos móviles y la autonomía. A pesar de estos hechos, el sistema funciona satisfactoriamente la mayor parte del tiempo, con una proporción de fracaso sumamente baja. Como el robot recibe comandos de alto nivel, no se necesita un ancho de banda alto y tanto el vídeo como la posición se actualizan con una velocidad muy baja. Aunque la comunicación se pierda completamente, Xavier todavía puede seguir alcanzando los objetivos de su tarea.

- **Minerva**

Otros sistemas basados en Web son muy similares al sistema Xavier tanto en la arquitectura como en el funcionamiento. Por ejemplo, los guías de museos Rhino y Minerva permiten a los usuarios seleccionar el área del museo que quieren ver. Una de las diferencias entre Minerva y Xavier es que Minerva usa una velocidad de actualización más alta para que la experiencia de usuario sea más interactiva. El objetivo del sistema Minerva es hacer que un museo sea accesible a visitantes remotos [Schulz,00].

La figura 4.16 muestra la interfaz de usuario de Minerva, que permite a los visitantes del museo dar comandos simples utilizando botones o tocando una pantalla táctil del robot. Además de mostrar gráficos y textos sobre las exposiciones en la pantalla, el robot reproduce sonidos grabados; como por ejemplo, para explicar exposiciones o para pedir autorizaciones.

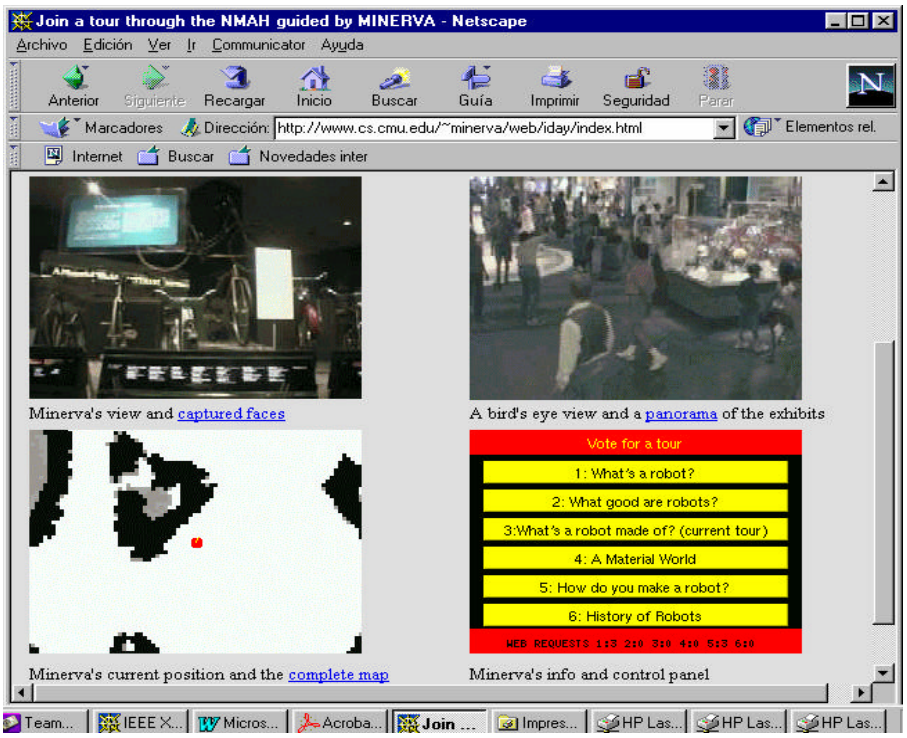


Fig. 4.16: Interfaz de Usuario de Minerva

- **KhepOnTheWeb de EPFL**

El objetivo del sistema KhepOnTheWeb del *Swiss Federal Institute of Technology of Lausanne*, es proporcionar el acceso a través de los medios de comunicación de red, a un robot móvil que funciona en un entorno complejo para llevar a cabo tareas de investigación en robótica móvil [Saucy,00]. El sistema consiste en un robot móvil que entra en un laberinto de madera. El robot es un robot móvil pequeño "Khepera" (55 mm en diámetro) que lleva una cámara CCD pequeña en color. Tiene instalada otra cámara panorámica adicional.

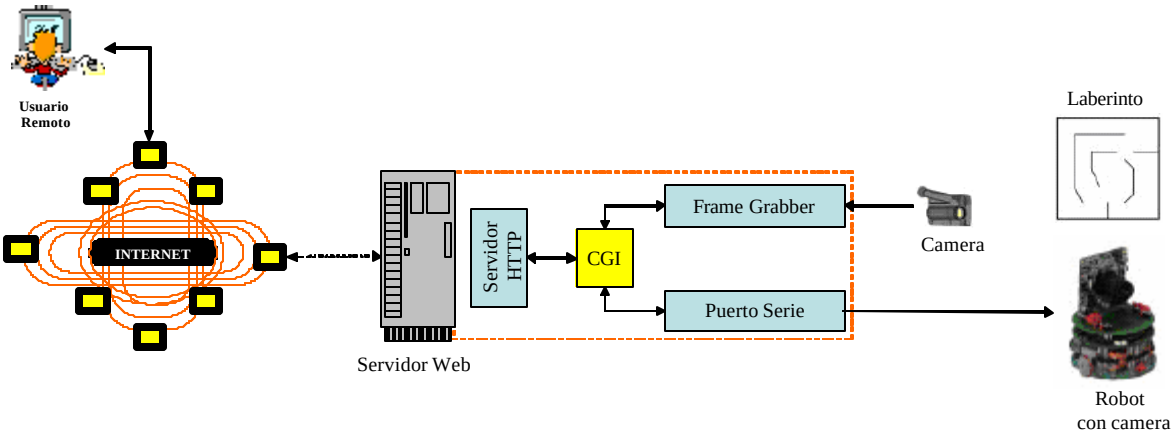


Fig. 4.17: Arquitectura de Sistema de KhepOnTheWeb

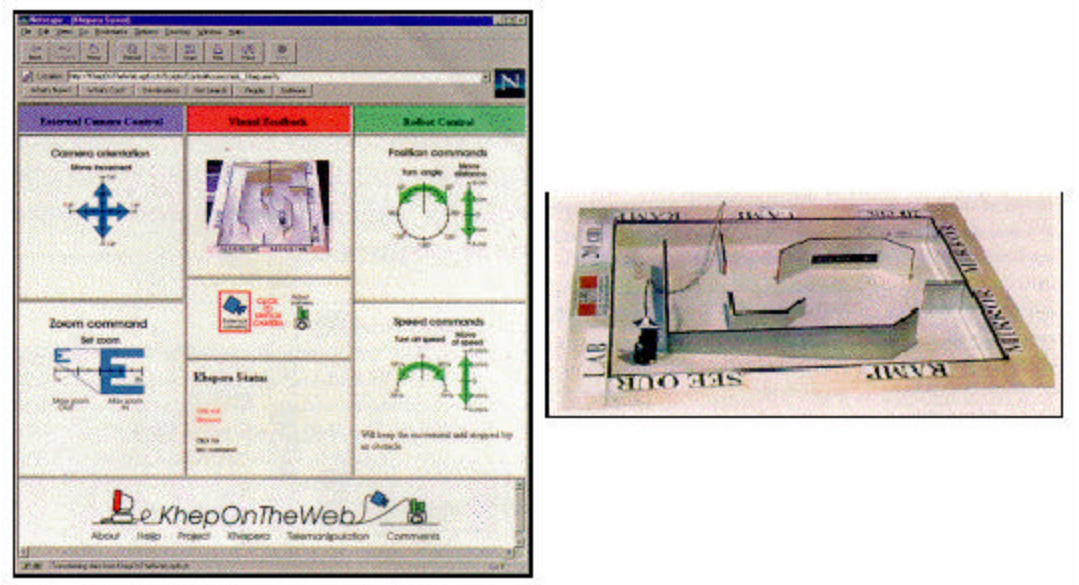


Fig. 4.18: Interfaz de Usuario de KhepOnTheWeb

El método de control de KhepOnTheWeb (Figura 4.17 y 4.18) es similar a los sistemas anteriores y está basado en rellenar formularios HTML y código CGI. A diferencia de Xavier, KhepOnTheWeb permite el control en bucle cerrado. No existe ninguna inteligencia local en el robot, como evitar obstáculos. Así, el operador realiza un control en bucle cerrado del robot usando el video como realimentación. En el caso de Khepera, no hay necesidad de comportamientos inteligentes debido al peso ligero del robot. Esto significa que no hay ningún riesgo de destruir una pared o el propio robot. Los diseñadores de KhepOnTheWeb escogieron esta opción porque teniendo control directo tiene la ventaja de que el usuario puede ver el resultado de su propia acción sin cualquier contribución externa. Esto tiene un inconveniente mayor: el control del robot es más difícil sin ayuda y sufre retrasos importantes. Para solucionar el problema del retraso temporal, un modelo virtual del experimento también se ha desarrollado usando VRML y Java. Este modelo proporciona al usuario la posibilidad de entrenamiento a través de controlar una simulación locamente en su máquina. Aunque la experiencia con KhepOnTheWeb es bastante satisfactoria para el usuario, la metodología no se puede adaptar en ambientes reales.

- **NASA WITS**

El sistema de NASA, WITS (*Web Interface for Telescience*) se ha desarrollado para proporcionar operaciones distribuidas basadas en el campo de la exploración planetaria y misiones de exploradores [Tso,98]. WITS proporciona un ambiente integrado para la colaboración con científicos geográficamente distribuidos. Hay dos versiones de WITS, las privadas para el uso de NASA, y las públicas que son gratis y disponibles para todo el mundo. Ambas versiones tienen funcionalidades similares, pero se diferencian en la cinemática de robot real y los comandos disponibles [WITS,03]. WITS ha sido usado y desplegado satisfactoriamente en muchas misiones en Marte de la NASA. Durante estas pruebas, los operadores usaron WITS para visualizar datos del enlace de descarga (*downlink*) y generar secuencias de comandos desde varios laboratorios y centros de control alrededor del mundo.

La arquitectura de sistema de WITS (Figura 4.19) incluye una base de datos, un servidor y clientes múltiples [Backes,00-a][Backes,00-b]. La base de datos es un sistema de archivos estructurado que sostiene datos del enlace de descarga y la información de secuencia de comandos. El servidor proporciona la comunicación entre los clientes y la base de datos. Los clientes están distribuidos y proveen la interfaz necesaria para ver los datos y generar secuencias de comandos. El sistema WITS ha sido implementada usando Java y algunas de sus extensiones, como Java3D (para el dibujo del modelo sintético de 3D) y la Extensión de Criptografía de Java (para establecer comunicaciones seguras).

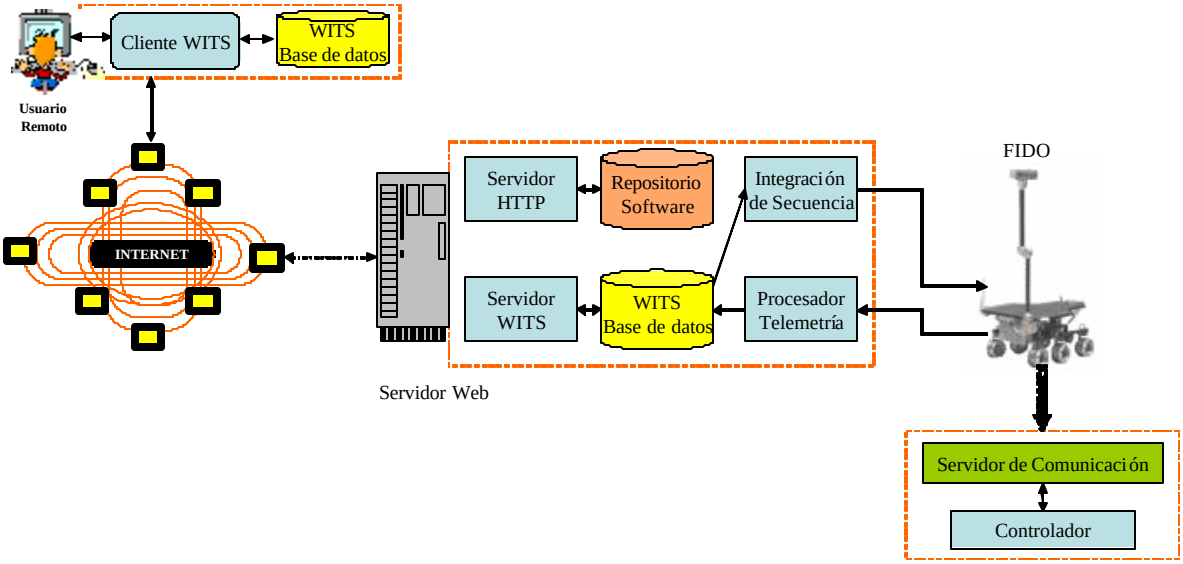


Fig. 4.19: Arquitectura de Sistema de NASA WITS

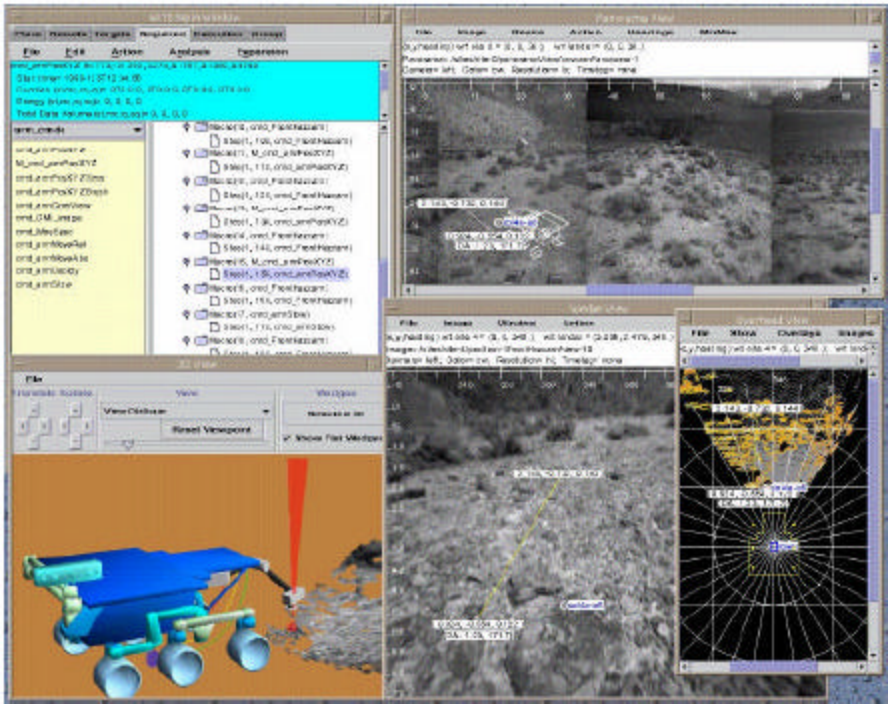


Fig. 4.20: Interfaz de Usuario de NASA WITS

En la interfaz de usuario (véase la figura 4.20), el usuario consigue el software a través de una página HTML. Entonces el applet de Java se encarga de acceder a una base de datos local y al servidor de WITS. El usuario genera una secuencia de acciones localmente, usando el simulador FIDO para comprobar los resultados. Cuando se termina, el usuario envía la secuencia al servidor de WITS. La secuencia de comandos se comprueba usando una integración de secuencia y el módulo de verificación (SEQ_GEN) y luego es enviada al explorador remoto. Además, el usuario tiene acceso a los datos recibidos por el enlace de descarga en la base de datos remota de WITS. Estos datos incluyen la posición del robot, imágenes de las cámaras de navegación, de las cámaras panorámicas y de otros sensores.

WITS, como pasa con Xavier, no permite el control en bucle cerrado del explorador remoto, aunque la razón es bastante diferente. Una de las razones es que WITS ha sido diseñado como asistencia en misiones espaciales donde los retrasos son muy considerables. Sin embargo, WITS, en contraste con Xavier, permite al usuario especificar tareas completas sin cualquier restricción. El usuario genera una secuencia de acciones que ejecuta el explorador al final del proceso. La secuencia incluye una prueba de integridad.

- **Sistema BGen-Web**

BGen-Web es una aplicación robótica de exploración basada en Web [Barberá,01]. El objetivo de esta aplicación es proporcionar un servicio basado en Web para un control supervisor altamente interactivo de un robot móvil con el objetivo de realizar una exploración de un entorno poco conocido o completamente desconocido.

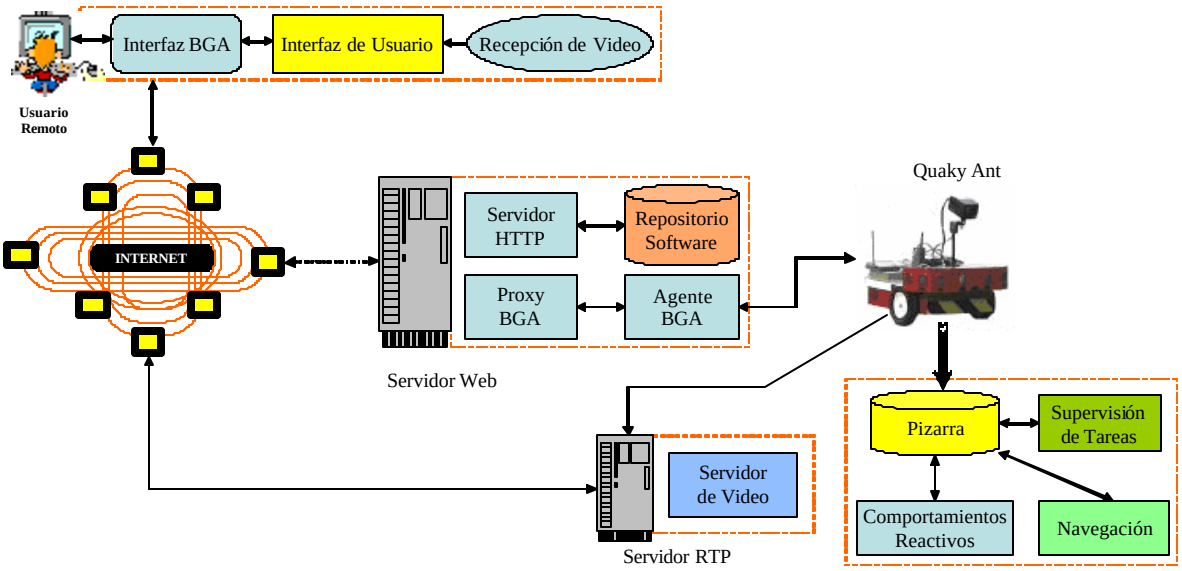


Fig. 4.21: Arquitectura del Sistema BGen-Web

En la arquitectura del sistema (véase la figura 4.21) los clientes proporcionan interfaces para datos sensoriales, realimentación visual, y control del robot. El servidor Web proporciona el repositorio de software y la pasarela para el protocolo de arquitectura distribuida de control (BGA) basada en una pizarra y agentes [Barberá,01]. El servidor de vídeo proporciona vídeo en tiempo real. Ambos servidores pueden ejecutarse físicamente en el mismo equipo. La pasarela BGA

actúa como un servicio proxy: recibe peticiones de los distintos clientes, las deposita en la pizarra, lee información sensorial de la pizarra y la envía a los distintos clientes. El vídeo en tiempo real lo proporciona el *Java Media Framework (JMF)* que ofrece interfaces de programación abstractas para las aplicaciones de video y audio en tiempo real. Así, permite a los programadores desarrollar software que presenta, captura y almacena dichos media, y también controla el tipo de procesamiento aplicado a esos flujos. Por último, la JMF ofrece soporte RTP [RFC1889] para ser usado con aplicaciones de vídeo o audio bajo demanda o aplicaciones de videoconferencia interactiva. En la implementación actual de BGen-Web el servidor de vídeo envía imágenes QCIF H.261 a 7.5 frames/s.

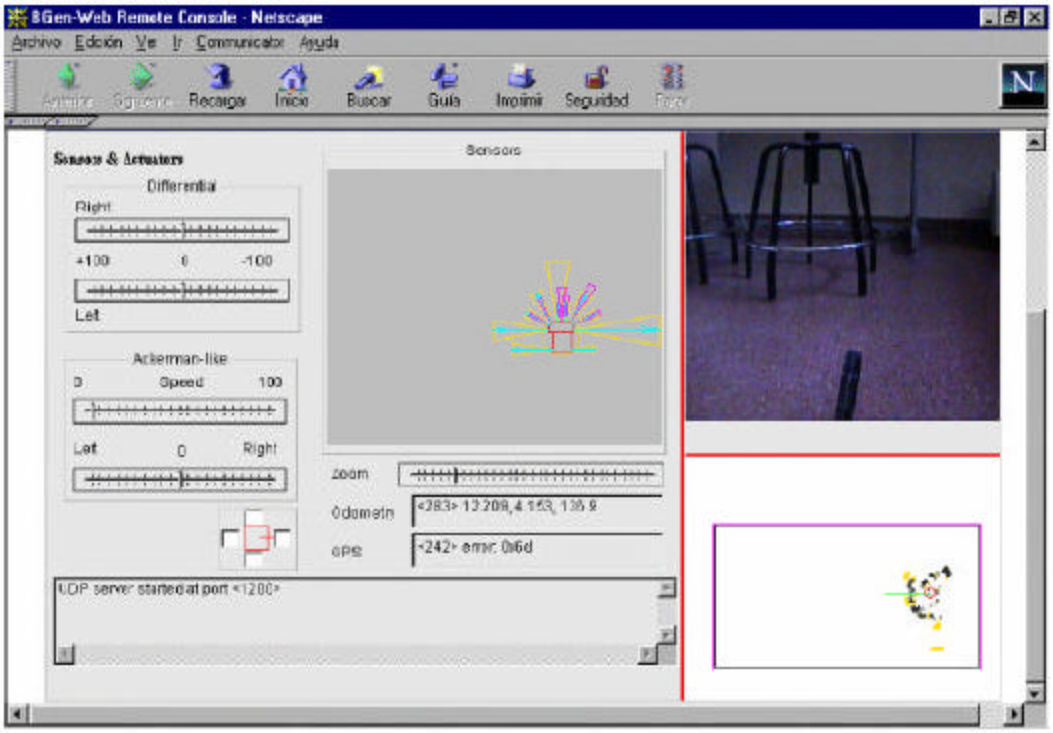


Fig. 4.22: Interfaz de Usuario del Sistema BGen-Web

La operación de BGen-Web es muy simple. Cuando el applet (Figura 4.22) arranca, se conecta tanto al proxy BGA (por medio de una dirección unicast) como al servidor de vídeo (por medio de una dirección multicast), y comienza a recibir datos de los sensores y realimentación de vídeo en tiempo real. El applet tiene unos componentes que muestran las lecturas de los sensores (ultrasonidos, infrarrojos, sensores virtuales, parachoques, brújula, odometría y GPS), el vídeo, el mapa de celdas difusas actual, y que permiten dirigir el robot.

- **TOM**

En la universidad de ciencias aplicadas Fh-Weingarten en Alemania, se están desarrollando varios experimentos sobre el robot TOM (TeleOperated Machine) utilizando la arquitectura mostrada en la figura 4.23 [Khamis,02].

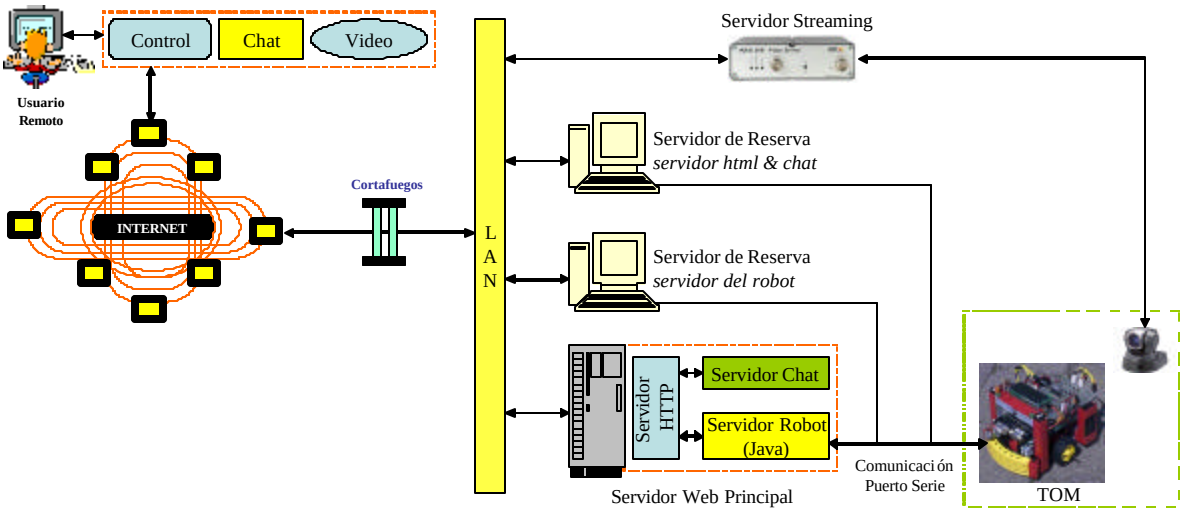


Fig. 4.23: Arquitectura del Sistema de TOM

En el lado servidor, hay cuatro equipos. El servidor principal del sistema, servidor de reserva del robot, el servidor de reserva html/chat y el servidor de streaming de video. El servidor principal del sistema es una máquina Linux que contiene el servidor del robot escrito en Java y el servidor html y chat. El servidor de reserva del robot y el servidor de reserva html/chat son dos máquinas Solaris, que funcionan como servidores de reserva para el servidor principal en caso de fallo del equipo o cualquier otro problema. El servidor de streaming video transforma el video analógico tradicional en imágenes digitales de alta calidad que pueden transmitirse a través de la red.

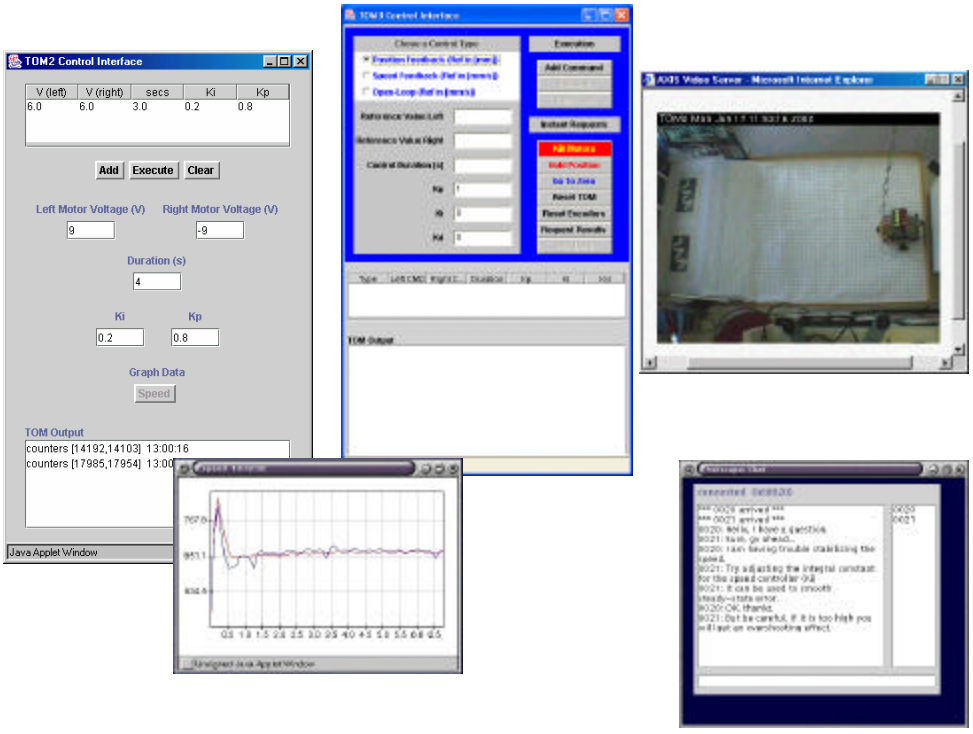


Fig. 4.24: La interfaz de usuario de TOM

La interfaz de usuario (véase la figura 4.24) incluye el control de posición y velocidad del robot, ajustando los voltajes de la rueda de izquierda y derecha y la duración del movimiento. También el cliente puede estudiar el efecto de los parámetros del controlador PI del robot. Una cola de espera se utiliza para manejar el acceso de los usuarios. Cuando sólo haya un usuario, no se da ninguna restricción en el uso, sin embargo, si se conectan varios usuarios simultáneamente, solo uno puede controlar el robot durante una cantidad especificada de tiempo. Una vez que el tiempo expira, el próximo usuario en línea tendrá la oportunidad de controlar el robot y el primer usuario se moverá al final de la lista de espera. El *chat* se proporciona como un mecanismo de comunicación entre los usuarios, también la cámara proporciona una vista real de los movimientos del robot.

- **RedRover**

En la misma universidad, se ha desarrollado el sistema RedRover (véase la figura 4.25) para la agencia espacial europea (ESA - *European Space Agency*). Este robot móvil puede controlarse para explorar un entorno de Marte en la universidad. Este sistema lleva disponible en Internet las 24 horas al día desde hace 6 años [RedRover,03].

Fig. 4.25: El robot RedRover y su interfaz de usuario

- **Ambiente Virtual para Programación de Robots Móviles (LabVir)**

En el Instituto tecnológico de Monterrey, México se ha desarrollado el sistema LabVir para la programación de robots móviles [LabVir,03]. En [Trinidad,02] se propone un modelo para el control remoto del robot Nomad 200 en este sistema, como se muestra en la figura 4.26. Este modelo consta de cuatro módulos.

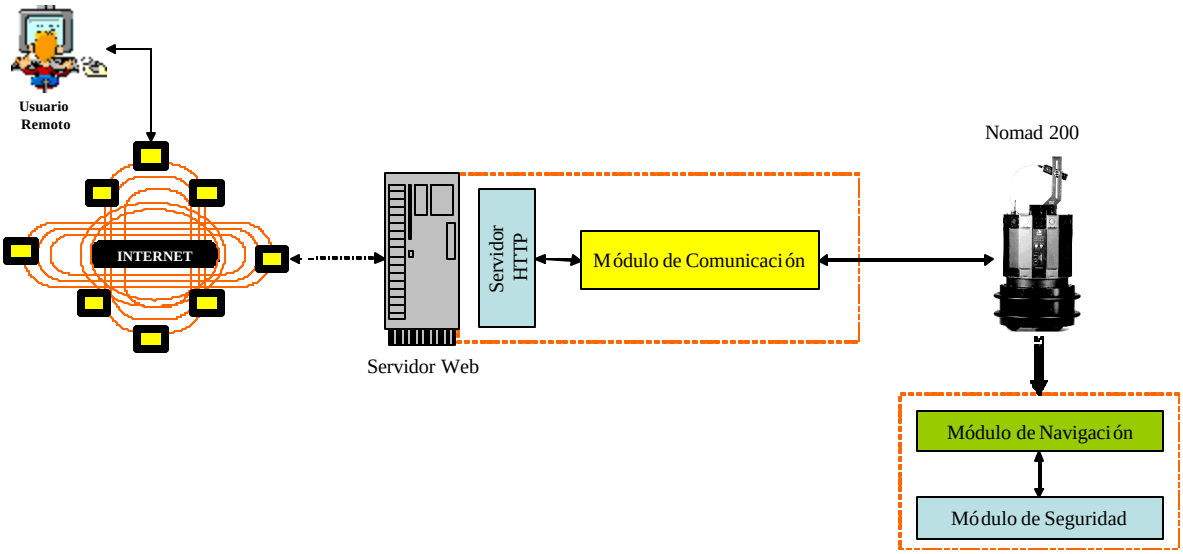


Fig. 4.26: Telecontrol de Nomad 200

El módulo de programación consta de cuatro subsecciones: a) introducción, mediante una interfaz gráfica, de los comandos y parámetros de un programa, b) validación de estos comandos con sus parámetros, c) codificación de los comandos y parámetros en un vector y d) envío por medio de la red de este vector. El módulo de comunicación esta compuesto de dos métodos que se encargan de enviar y recibir el vector que contiene el código de comandos y parámetros. Además se tiene otro método que se encarga de escribir el arreglo codificado para finalmente ejecutar un programa en "C" que interpreta y ejecuta las instrucciones en el robot móvil. El módulo de navegación esta programado en lenguaje "C" y se encarga de establecer la comunicación con el robot móvil, interpretar y ejecutar el conjunto de instrucciones que fue recibido por el módulo de comunicación. Dentro del programa de navegación se tiene un módulo de seguridad que se ejecuta de manera concurrente dentro del robot móvil. Este módulo toma continuamente lecturas de los sónares y de los sensores táctiles, para que en caso de posibles colisiones se detenga de inmediato el programa de navegación

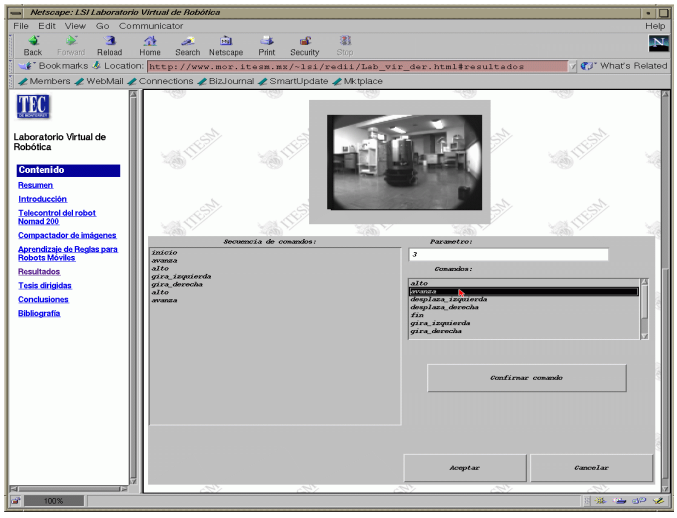


Fig. 4.27: Interfaz del Robot Móvil Nomad 200

Actualmente, con la interfaz (figura 4.27) es posible programar el robot Nomad 200, utilizando Internet. Para ello se ha definido un conjunto reducido de instrucciones que se envían por la red y se interpretan en el robot móvil. Las mismas instrucciones se utilizan para programar el robot real y el simulado.

4.3.2 Laboratorios Distribuidos

El acceso remoto mediante medios electrónicos, en particular a través de la red de Internet, a equipos o herramientas costosas, permite un mayor acceso a la comunidad para fines educativos o de investigación. De aquí viene la importancia de establecer entornos electrónicos de colaboración para intercambiar ideas y recursos de software y hardware costosos como se ha discutido en detalle en la sección 4.2. Se plantean en las siguientes subsecciones dos ejemplos de laboratorios distribuidos en el campo de la mecatrónica y la telemática.

▪ **El Proyecto IECAT**

Se ha realizado la presente tesis en el marco del proyecto IECAT [IECAT,03] que pretende desarrollar una red de laboratorios remotos entre seis universidades en Europa y EE.UU. en el campo de los sistemas autónomos y teleoperados. En esta sección se explica en detalle los objetivos de este proyecto, las universidades participantes y sus aportaciones.

- **Objetivos**

El objetivo principal del proyecto es implementar un laboratorio distribuido internacional, permitiendo a los estudiantes llevar a cabo experimentos de laboratorio con equipamiento real y simulado perteneciente a las universidades participantes. Debido a la larga distancia que separa unas universidades de otras, los estudiantes tendrán a su disposición todo lo necesario para la teleoperación de robots y otros experimentos mediante laboratorios remotos, accesibles vía Internet.

Este laboratorio distribuido permite formar y entrenar a estudiantes de diferentes nacionalidades en los campos de la mecatrónica y la telemática usando modernas tecnologías multimedia, áreas con amplias perspectivas de empleo en el futuro.

A su vez, las universidades participantes en el proyecto intercambian materiales educacionales y conocimientos, desarrollando y probando conjuntamente en clases practicas de laboratorio los experimentos diseñados.

- **Universidades participantes y sus aportaciones**

La figura 4.28 muestra las universidades participantes en el proyecto y sus principales aportaciones.

1- Universidad de Ciencias Aplicadas– Alemania ➢➢ Exploradores Teleoperados

Desarrollo de un experimento para control remoto de robots móviles en exteriores, considerando efectos relacionados con:

- Control con retardos en tareas de muestreo y reconocimiento.
- Control de reacciones basadas en diferentes configuraciones de sensores.
- Integración en el sistema de funcionalidades de control autónomo y teleoperado.
- Telediagnosis de situaciones de fallo.
- Técnicas telemáticas para control remoto basado en applets de Java, aplicaciones y CGI-Scripts.

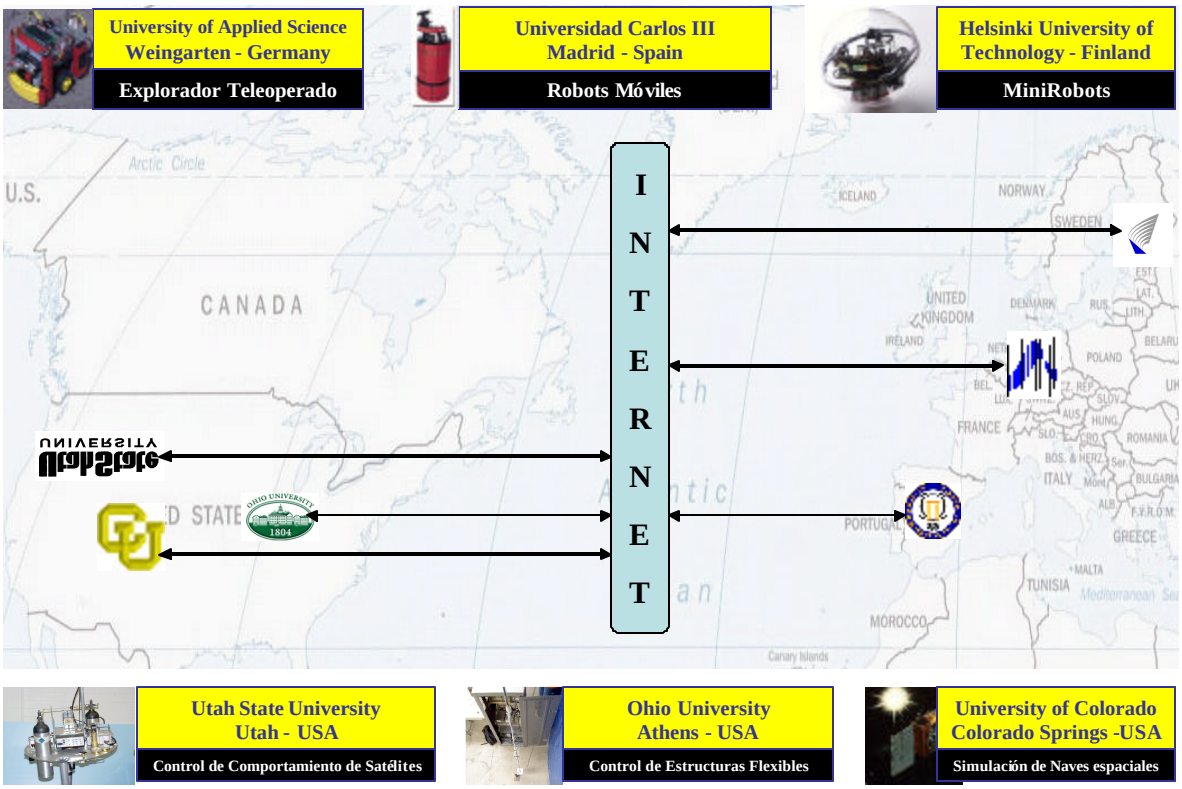


Fig. 4.28: Participantes y Experimentos del Proyecto IECAT

2- Universidad Carlos III de Madrid ➢➢ Robots Móviles

Implementación de un laboratorio remoto para robótica móvil. Además del estudio de la teleoperación de los robots, se analizan remotamente diferentes enfoques para resolver los principales problemas de los robots móviles como las habilidades de movimiento, la percepción y el modelado del entorno, etc. En el capítulo 7 se describen en detalle los experimentos en línea desarrollados en la Universidad Carlos III.

3- Universidad de Tecnología de Helsinki ➢➢ Minirobots

Teleoperación de un minirobot bola con caparazón transparente. La "bola" es capaz de moverse por el laboratorio llevando una cámara en su interior. Los estudiantes podrán trabajar sobre:

- Teleoperación de un robot con una cinemática y dinámica compleja.

- Identificación de parámetros de sistemas de locomoción.
- Localización del robot mediante un sistema de localización basado en un puntero láser.
- Inspeccionar el entorno y objetos mediante la cámara de interior.

4- Universidad de Utah State ➢➢ Control del Comportamiento de un Satélite

Se proporciona un pequeño simulador de control de comportamiento de satélites para control y operación remota a través de Internet. Éste incluye control de gases de potencia y control de momento. Los estudiantes serán capaces de escribir algoritmos de control para el simulador. La monitorización con video del simulador proporcionará a los estudiantes la posibilidad de ver la evolución del sistema en tiempo real

***5- Universidad de Ohio ➢➢* Control de Estructuras Flexibles**

Proporciona acceso remoto a través de Internet al laboratorio de control de estructuras flexibles para formación en modelado y control de estructuras flexibles, especialmente la simulación hardware de instrumentos ópticos orbítales como es el *Next Generation Space Telescope*. Los objetivos del experimento son:

- Determinación de los parámetros mecánicos del "*swinging rod*" (varilla flexible).
- Comparación de diversos algoritmos de control (PIDs, control adaptativo) para reducir la amplitud de la oscilación.
- Determinación de los efectos relacionados con la localización de los sensores en la controlabilidad y calidad del control.

6- Universidad de Colorado ➢➢ Simulación No Lineal de Naves Espaciales

Proporciona un entorno de simulación para la exploración de nuevas técnicas y algoritmos de control de vehículos espaciales. Se podrán utilizar las simulaciones y los resultados (tanto datos como imágenes) que se recibirán vía Internet. Estas simulaciones pretenden evaluar varios algoritmos de control así como técnicas de guiado y movimiento.

- **TEAM**

El proyecto TEAM (*TEleeducation in Aerospace and Mechatronics using a virtual international laboratory*) es un proyecto de colaboración entre universidades europeas (Universidad de Ciencias Aplicadas - Alemania, Universidad de Aalborg – Dinamarca y Universidad de Bologna - Italia) y canadienses (Universidad de Victoria, Universidad de Toronto y Universidad de Sherbrooke) cuyos experimentos son de características similares a los del Proyecto IECAT [TEAM,03].

4.4 METODOLOGÍA PARA CONSTRUIR ENTORNOS EDUCATIVOS EN ROBÓTICA MÓVIL

Las clases y los laboratorios son las formas de docencia tradicionalmente usadas en cualquier sistema de educación. El conocimiento teórico en la educación convencional, consiste en clases y ejercicios, complementados por libros y notas de clases. Por otro lado, el conocimiento práctico comprende cursos de laboratorio muy costosos. Para construir un entorno educativo para un tema como la robótica móvil, hay que incorporar varias herramientas y actividades por medio de las cuales se pueden mapear las actividades tradicionales en los sistemas de educación. En está sección, se plantea una metodología que se puede usar para construir entornos educativos en robótica móvil.

La metodología propuesta intenta combinar actividades educativas diferentes para desarrollar un entorno de enseñanza innovador. Como se muestra en la figura 4.29, esta metodología tiene dos partes principales, actividades de instrucción y actividades de construcción.

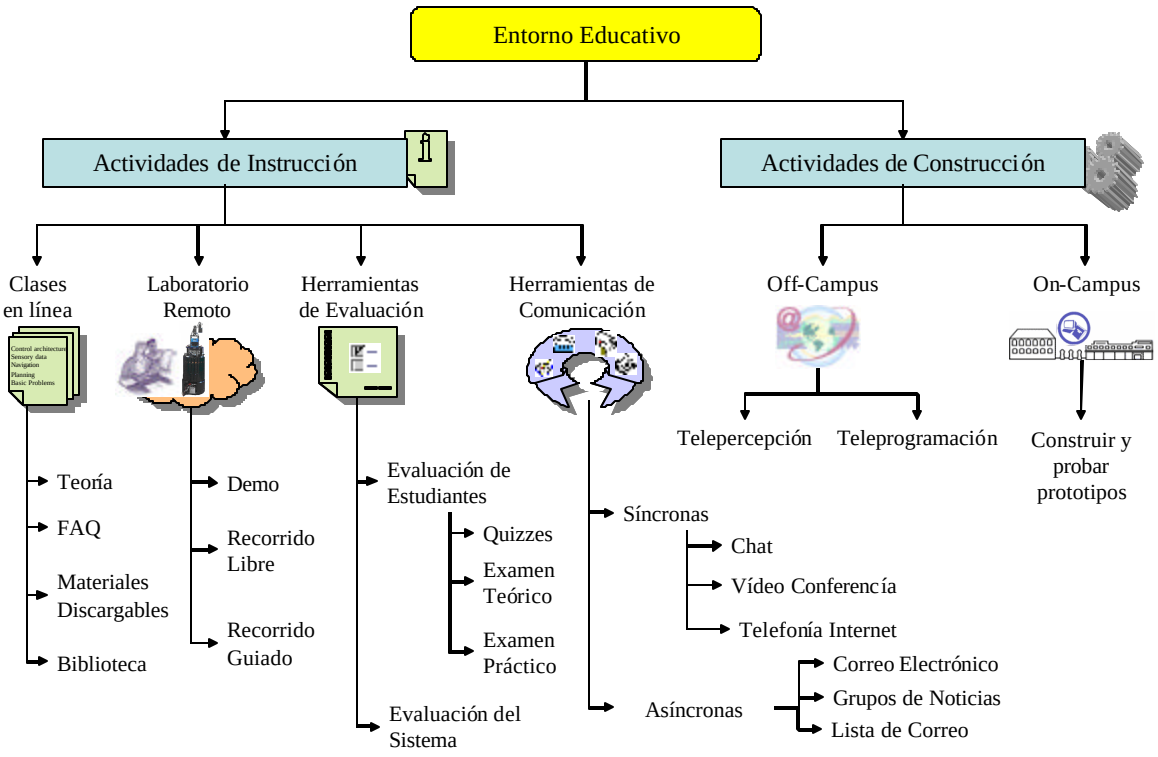


Fig. 4.29: Entorno Educativo para Robótica Móvil

Las actividades de instrucción basadas en Web se usan para mapear la enseñanza tradicional y para resolver los problemas del paradigma tradicional como el número elevado de estudiantes y los recursos limitados. Las actividades de instrucción deben complementarse con otras actividades o herramientas que permiten a los estudiantes tener un papel activo en el sistema y que pretenden mejorar la creatividad de los estudiantes. Esto se puede conseguir mediante el desarrollo de actividades de construcción, que dan a los estudiantes la oportunidad de construir físicamente y poner en práctica las ideas del curso. Estas actividades también pretenden proporcionar la comunicación cara a cara entre el estudiante y el tutor y la comunicación hombre-

máquina que es muy importante en un campo como la robótica móvil. Las subdivisiones siguientes describen en más detalle las actividades propuestas y en el capítulo 7, se presenta la implementación de estas actividades.

4.4.1 Actividades de Instrucción

El modelo tradicional de educación es bastante limitado. Los estudiantes están en las aulas con un profesor frente a ellos. La palabra "enseñar" hace al educador el responsable de esta transmisión de conocimiento. Este flujo de conocimiento es, en su mayor parte, unidireccional con la excepción de discusiones o preguntas ocasionales. La cantidad de las preguntas o de discusiones es inversamente proporcional con el tamaño de la clase, resultando que las clases grandes frecuentemente se parecen a ver un vídeo informativo.

Los sistemas de teleducación proporcionan una solución a los problemas de la enseñanza tradicional. El uso de las tecnologías de las comunicaciones y los computadores en la enseñanza tiene una historia de más de 30 años. En ese tiempo ha sido llamado por muchos nombres, incluyendo la comunicación mediante computador (*Computer-mediated Communication* - CMC), la conferencia vía computador, enseñanza en línea (*online learning*), enseñanza basada en Internet y telemática [McCormack,97]. En la pasada década muchos otros términos se han propuesto para los sistemas de educación basados en el apoyo de computadores como redes de aprendizaje asíncrono [Hiltz,97], teleeaprendizaje colaborativo [Alavi,95], educación a distancia [UNESCO,87], aprendizaje a distancia [Whalen,98], enseñanza a distancia [Moore,73], aprendizaje flexible [Ellington,95], aprendizaje a distancia mediante la tecnología [Webster,97], aprendizaje interactivo [Bourne,96] y teleeducación. Se esperan varios beneficios de desarrollar sistemas de instrucción basados en Web como se muestra en la tabla 4.1.

Tabla 4.1: Beneficios Esperados de la Teleeducación

	Beneficios		
Estudiantes	- Mayor Disponibilidad y flexibilidad al no haber barreras físicas. - Mejora en el acceso y búsqueda de materiales actualizados. - Aprendizaje para toda la vida.	- Acceso a expertos remotos. - Mayor Rendimiento - Mayor Promoción - Aprendizaje del conocimiento anecdótico almacenado.	- Realimentación inmediata y comparación con otros estudiantes.
Instructores	- Mayor Participación.	- Plazo de tiempo más. largo para entregar cursos.	- Informes inmediatos del rendimiento de la clase.
Instituciones	- Número aumentado de estudiantes. - Mayor variedad de estudiantes - Tiempo de entrenamiento más corto.	- Planificación más flexible. - Menos requerimientos para el aula.	- Mayor satisfacción de los empleados. - Mejor competitividad con otros centros.

Varias actividades de instrucción se pueden implementar utilizando el modelo de educación basado en Web para mapear las actividades de la enseñanza tradicional y solucionar los problemas asociados con dichos sistemas. En las subsecciones siguientes se plantean las actividades de instrucción basados en Web.

4.4.1.1 Clases en línea

Los cursos educativos convencionales se ofrecen según una planificación específica. Una tarea importante de cualquier aula es la distribución de información, tanto administrativa como educativa. En una clase de gran tamaño, los estudiantes tienen un papel pasivo en la clase. Como resultado los estudiantes absorben solamente alrededor de 30% de los conceptos presentados en la clase [McCormack,97]. Las desventajas de la clase se compensan con problemas de resolución de ejercicios en grupos, resolución de problemas de forma dirigida, y discusiones durante las tutorías.

Un procedimiento propuesto por J. Bourne [Bourne,96] para crear cursos en línea se ha presentado según las reglas siguientes:

- Analizar las necesidades y los resultados deseados del estudiante durante el curso, para determinar el contenido y tipos de materiales que se ajustan a los cursos y al estudiante.
- Diseñar las evaluaciones (p.ej., los tipos de ejercicios, los laboratorios, y pruebas escritos)
- Construir un índice de materias y una página Web para el curso, con una página principal que proporcione al usuario información completa sobre el curso.
- Determinar las estrategias y los tipos de componentes necesarios para preparar a los estudiantes para las evaluaciones. Estas estrategias incluyen actividades tanto síncronas como asíncronas.
- Crear ejercicios, laboratorios, gráficas y materiales de texto para cada uno de los temas del curso.
- Probar el esqueleto básico de los materiales del curso con estudiantes reales es muy útil para la realimentación
- Agregar demostraciones y enlaces al software del laboratorio
- Evaluar el curso con métricas conocidas. Por ejemplo, la reducción en el tiempo del tutor o del estudiante, o el aumento en la satisfacción y rendimiento del estudiante, que pueden determinarse usando encuestas.
- Revisar los materiales del curso.

Para organizar el contenido teórico de una clase en línea, es útil debatir sobre qué enseñamos en línea. Los materiales basados en Web actualmente son frecuentemente presentaciones de información que no se adaptan a un esquema intelectual adecuado para el aprendizaje. Si queremos crear materiales en línea, la creación y la presentación de estos materiales deberían conducirse desde un modelo robusto sobre ¿qué debería aprender un estudiante de robótica móvil? Esta no es una pregunta trivial. La respuesta a esa pregunta de gran parte de los estudiantes, suele ser "los fundamentos", seguidos estrechamente por "cómo resolver problemas".

Bourne propuso clasificar los tipos de cosas que estudiantes de ingeniería deberían aprender según dos taxonomías bien conocidas en la educación [Bourne,96]. La taxonomía de Barrett propuso que el aprendizaje debe dividirse en cuatro categorías: literal, inferencial, aplicativo y evaluativo. La taxonomía de Merrill usa conceptos a recordar, usar, o encontrar (crear). Los contenidos se clasifican como hechos, conceptos (clasificación), procedimientos o principios. La matriz de Merrill especifica el aprendizaje como una pareja actuación - contenido, como un procedimiento de usuario o un principio de búsqueda. Los hechos se emparejan solo con recuerdos. Las taxonomías parecen ser útiles para clasificar los resultados del aprendizaje en ingeniería y para seleccionar la tecnología requerida para la implementación en línea.

Las siguientes tablas muestran los resultados del aprendizaje en robótica móvil según las taxonomías de Barret y Merritt, que se pueden usar para organizar el contenido teórico de una clase en línea de robótica móvil.

Tabla 4.2: Conocimientos de Identificación

Qué es …(Identificación)	Barrett	Merrill
	Literal	Hechos a Recordar
¿Qué es un robot? ¿Cuales son los componentes principales de un robot móvil? ¿Cuales son los tipos de sensores que se usan con frecuencia en robots móviles? ¿Cual es la función del: planificador de trayectoria, el navegador y el piloto?		

Tabla 4.3: Conocimiento de Características Reales

Recordar Características	Barrett	Merrill
	Literal	Recordar Concepto
- Plantear las características de cada tipo de sensores que se usan normalmente en los robots móviles.		

Tabla 4.4: Describir el Conocimiento del Proceso

Describir proceso	Barrett	Merrill
	Literal	Recordar Proceso
- Describir cómo funcionan los sensores siguientes: sonar, táctil, láser, IR. - Describir el proceso siguiente: localización, detectar y evitar obstáculo.		

Tabla 4.5: Conocimiento de los Procedimientos

Cómo se hace algo (procedimiento)	Barrett	Merrill
	Literal	Recordar Procedimiento
- Cómo se usan los datos sensoriales en el proceso de percepción del entorno. - Cómo se puede corregir los errores sistemáticos de la odometria.		

Tabla 4.6: Estructurar un Problema

Establecer un problema usando directivas	Barrett	Merrill
	Literal	Recordar Fundamento
- Enumerar las directivas para obtener el comportamiento siguiente: seguir pared, evitar obstáculos.		

Tabla 4.7: Reconocer Conocimiento

Reconocer tipos de componentes	Barrett	Merrill
	Inferencial	Usar conceptos (clasificar cosas en categorías)
- Identificar la categoría de cada uno de estos robots. - Dados estos datos sensoriales, determinar cual es la fuente de estos datos.		

Tabla 4.8: Conocimiento de las Consecuencias de una Acción

Qué sucedería si	Barrett	Merrill
	Inferencial	Usar Principios
- En el método del campo de potencial para la planificación de trayectoria, qué sucedería si el manipulador entra en un mínimo local. - Qué sucederá si el haz del sonar encara bordes bruscos u objetos transparentes.		

Tabla 4.9: Aplicar Patrones

Aplicar un patrón aprendido o una secuencia de datos	Barrett	Merrill
	Aplicativo	Usar procedimiento
- Determinar la zona de los obstáculos y dibujar el mapa de entorno basado en los datos sensoriales adquiridos. - Usar un sistema de navegación topológico para obtener las habilidades siguientes: seguir pasillo, cruzar puerta …		

Tabla 4.10: Enlace

Enlace entre la teoría y práctica	Barrett	Merrill
	Aplicativo	Usar principio (leyes & heurístico para resolver un problema)
- Explicar por qué hay una diferencia entre la posición real del robot y los datos de la odometria.		

Tabla 4.11: Complejidad del Entorno

Enlace de Problemas a la complejidad de los entornos reales	Barrett	Merrill
	Aplicativo	Usar Principios
Esta figura indica que hay algunas medidas que salen fuera de las fronteras del laboratorio. ¿Cuál es su hipótesis para este resultado extraño?		

Tabla 4.12: Especificaciones

Crear especificaciones e implementarlas	Barrett	Merrill
	Evaluativo	Encontrar Principios
- Evaluar este problema de robótica, y diseñar y construir un circuito para implementar la estrategia de control.		

Tabla 4.13: Analizar

Averiguando (Analizando)	Barrett	Merrill
	Evaluativo	Encontrar Fundamentos
- Crear un conjunto de directivas para obtener el mapa del entorno utilizando los datos sensoriales.		

En el capítulo 7, se puede ver un ejemplo de un curso fundamental de robótica móvil que introduce los conceptos básicos de los robots móviles, sus componentes, los sensores, la arquitectura de control, etc. También se han añadido a este curso, páginas de preguntas más frecuentes (FAQ), una máquina de búsqueda, una biblioteca digital y varios materiales descargables.

4.4.1.2 Laboratorios Remotos

En las sesiones de laboratorio, los estudiantes ponen su conocimiento y habilidades en práctica. Fuera de las clases programadas, los instructores dan ayuda adicional a los estudiantes que tienen dificultades. Sin embargo, las limitaciones de tiempo y el gran número de estudiantes que requieren ayuda impiden el ofrecimiento de una alta calidad de asistencia. Para mejorar la eficacia en la enseñanza de grupos grandes de estudiantes, el laboratorio remoto puede usarse como una alternativa como se ha mencionado en la sección 4.2.

En el laboratorio remoto, se pueden presentar tres tipos de recorridos de instrucción al estudiante como herramientas educativas innovadoras para profundizar y aplicar el conocimiento sistemático de robótica móvil. Se clasifican según el nivel de apoyo, en un recorrido totalmente guiado (recorrido de demostración), un recorrido parcialmente guiado y un recorrido libre. El recorrido de demostración es un recorrido totalmente guiado, que presenta en un orden determinado todas las situaciones y tareas. El estudiante solamente será capaz de ejecutar una tarea cuando haya cumplido todas las órdenes anteriores. El recorrido guiado es más fácil de resolver y exige mucho más esfuerzo del autor para prepararlos didácticamente. Estos recorridos se pueden presentar al estudiante en forma de un curso en línea que tiene unas clases y unas prácticas, como en los métodos tradicionales de enseñanza y unas pruebas de evaluación en línea. El recorrido libre es un recorrido no guiado que no determina ninguna orden de situaciones o tareas. Este recorrido proporciona a los usuarios herramientas genéricas por medio de las cuales el usuario puede personalizar el experimento según sus necesidades. Estas herramientas genéricas pueden incluir un modelo 2D para el robot y el laboratorio, un panel para los datos de la odometría, los datos del sonar, los datos del láser, un controlador de movimiento y un editor para la teleprogramación.

4.4.1.3 Herramientas de Evaluación

Los métodos de la evaluación son aspectos importantes de cualquier curso. Tradicionalmente, corregir las evaluaciones manualmente y marcar los resultados es un procesamiento muy laborioso y necesita dedicar mucho tiempo. Con un entorno educativo basado en la telemática, será mucho más fácil construir herramientas que corrijan y manejen automáticamente los exámenes y los tests, lo que reduce el tiempo requerido para llevar a cabo estas tareas de evaluación. Las pruebas evaluativas se pueden utilizar para proporcionar las evaluaciones en línea. El estudiante accede primero a un examen y según el resultado se puede planificar y personalizar actividades de estudio que refuercen las áreas en las que tenga malos resultados. Además las pruebas prácticas se pueden diseñar para cubrir los temas experimentales como una simulación de laboratorios tradicionales.

En el curso de los fundamentos de robótica móvil, se han diseñado tres tipos de exámenes. Test en línea que se usan para evaluar los conocimientos de los estudiantes, exámenes teóricos con tiempo limitado que se usan para evaluar los conocimientos teóricos adquiridos en el curso y finalmente exámenes de práctica con el objetivo de ayudar a los estudiantes a entender bien los problemas prácticos.

4.4.1.4 Herramientas de Comunicación

Una parte esencial de cualquier experiencia de aprendizaje es la comunicación. Los sistemas de teleducación deben disponer de herramientas de comunicación por medio de las cuales los profesores y los estudiantes pueden estar en contacto. Los servicios de comunicación en los sistemas basados en Web se modelan típicamente a partir de las técnicas convencionales de interacción humana. Es decir, las conversaciones, las comunicaciones telefónicas, los libros y el correo postal se cambian por la videoconferencia, la telefonía Internet, las páginas Web y el correo electrónico respectivamente.

Las herramientas de comunicación síncronas, como el *chat*, la videoconferencia o la telefonía vía Internet, permiten una comunicación directa durante la sesión de aprendizaje entre los participantes. Las comunicaciones asíncronas, como el correo electrónico, las listas de noticias o las listas de distribución permiten la comunicación sin que todos los participantes estén conectados a la red al mismo tiempo.

4.4.2 Actividades de Construcción

El construccionismo es un proceso de aprendizaje activo en lo que los estudiantes construyen cosas que son personalmente significantes para ellos o para otros alrededor de ellos [Papert,91]. En lugar de recibir información de una manera unidireccional, los estudiantes desarrollan su propio conocimiento y comprensiones de un asunto a través de la construcción física y la implementación de sus ideas. Las actividades de la construcción les dan la oportunidad de construir físicamente e implementar las ideas sacadas del curso. Estas actividades son las siguientes:

4.4.2.1 Actividades Remotas

Las actividades de construcción remotas son actividades de aprendizaje activas tales como la teleprogramación, telemonitorización, telediagnosis, telemantenimiento, la telepercepción de entornos remotos, etc. que se puede incorporar en el laboratorio remoto. Una de las posibilidades

para implementar esta parte en la robótica móvil es desarrollar experimentos de percepción remota del entorno utilizando la información sensorial y construir mapas del entono y utilizarlos para la localización remota del robot y mostrar la diferencia entre las posiciones estimadas y las lecturas de la odometría. Otro experimento que se puede desarrollar es facilitar la programación remota del robot mediante un editor de comandos por medio del cual los usuarios pueden enviar comandos de bajo nivel al robot para llevar a cabo varias tareas como estudiar el efecto de variar los parámetros del controlador, cambiar la velocidad del robot, informar sobre el estado de la batería del robot, etc.

4.4.2.2 Actividades Locales

Las actividades de construcción locales, como diseñar y construir nuevos prototipos o aplicar la ingeniería inversa a sistemas existentes, se pueden usar para proporcionar la interacción cara-a-cara entre los estudiantes y los tutores y entre el estudiante y la máquina, y por consiguiente se aumenta la motivación de los estudiantes en la participación en el proceso educativo y se disminuye la sensación de aislamiento que algunos sienten al usar los sistemas de educación a distancia totalmente basados en Web. Estas actividades de construcción ayudan también a aumentar la creatividad de los estudiantes y el trabajo en equipo.

Para desarrollar estas actividades, se puede pedir a los estudiantes que construyan prototipos robóticos utilizando componentes hardware como Lego o Fischertechnik como componentes del hardware y los microcontralores como Motorola 88HC11 o 16-bit Siemens 80C167 y utilizar varios sensores disponibles para los robots móviles. Así varios robots móviles teleoperados pueden ser construidos y programados por los estudiantes.

Se esperan muchos beneficios de aplicar esta metodología para construir entornos educativos para la robótica móvil o para modernizar los cursos actuales con las nuevas tecnologías de telemática. Entre ellos cabe citar los siguientes:

- Aumentar la disponibilidad debida a que las actividades de instrucción no están restringidas por la asistencia simultanea de los estudiantes y los tutores.
- Aumentar la variedad y la flexibilidad debido a que esta metodología pretende hacer un equilibrio entre las actividades de instrucción y las de construcción.
- Comunicación aumentada gracias a la Web, que proporciona muchas herramientas de comunicación para que los estudiantes pueden hablar entre sí, individualmente o como un grupo, y enviar preguntas o mantener conversaciones orales o textuales con sus tutores.
- Aumentar la creatividad de los estudiantes donde las actividades de construcción ayudan que los estudiantes construyan cosas que son personalmente significantes para ellos o para otros alrededor de ellos.
- Aumentar las facilidades de control del proceso educativo por parte de los estudiantes, debido a que la combinación entre las actividades de instrucción y las de construcción ayudan a aumentar la sensación de tener control sobre el sistema y por lo tanto, se aumenta la motivación de los estudiantes para participar activamente en el proceso.

En el apéndice C, se describen en detalle las tecnologías más utilizadas para desarrollar laboratorios remotos basados en Internet.

Capítulo

5

PATRONES DE SOFTWARE PARA INTERFACES HOMBRE-ROBOT

Capítulo 5

PATRONES DE SOFTWARE PARA INTERFACES HOMBRE-ROBOT

5.1 INTRODUCCIÓN

La interfaz es la parte, tanto hardware como software del sistema informático, que facilita al usuario el acceso a los recursos del sistema. Como se muestra en la figura 5.1, existen tres tipos de interfaces que tienen que ser desarrolladas para establecer la comunicación remota basada en Internet entre el ser humano y el robot. A continuación, se explican los tres tipos de interfaces.

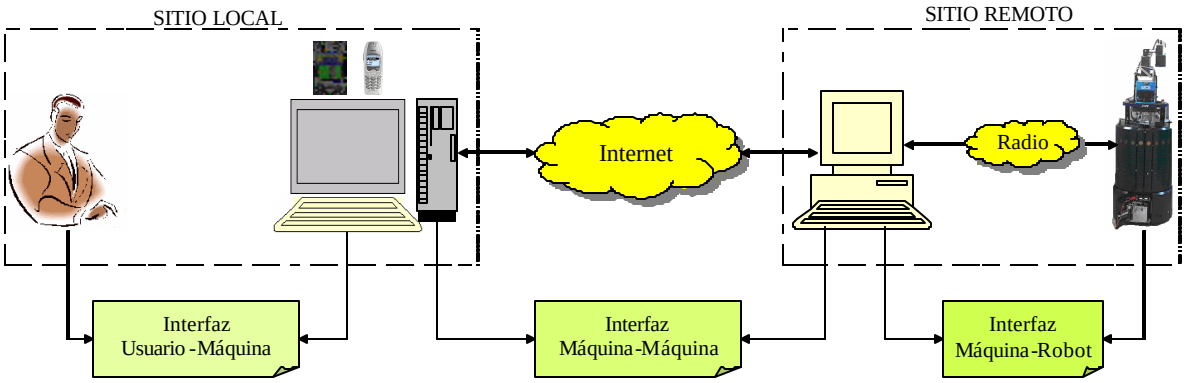


Fig. 5.1: Interfaces de un Sistema de Interacción Remota

- **Interfaz Usuario-Máquina**

La interfaz usuario-máquina o la interfaz gráfica de usuario es una parte fundamental en el proceso de desarrollo de los sistemas de interacción remota, por lo que desde el principio del desarrollo se debe tener en cuenta su diseño. La interfaz, como nexo de unión entre el ser humano

y el sistema, se caracteriza por su apariencia (nivel de presentación), su capacidad de gestión del diálogo (nivel de control) y su funcionamiento (nivel de abstracción). La interfaz determinará en gran medida la percepción e impresión que el usuario poseerá del sistema puesto que el usuario no suele estar interesado en la estructura y el funcionamiento interno de la interfaz. La tendencia actual a desarrollar interfaces intuitivas e independientes de la plataforma y del sistema operativo, está provocando que su diseño sea cada vez más complejo. En la sección 5.4 se discute en más detalle el desarrollo de interfaces de usuario utilizando patrones de software.

- **Interfaz Máquina-Máquina**

La interfaz máquina-máquina se refiere a los protocolos y los programas necesarios para intercambiar información a través de Internet entre dos máquinas ubicadas en lugares distintos. Se suele utilizar el modelo de programación distribuida cliente-servidor para llevar a cabo dicha comunicación. En este modelo la máquina del usuario funciona como un cliente que manda peticiones utilizando uno de los protocolos de Internet a otra máquina que funciona como servidor. En los sistemas de interacción remota basados en Internet, se suele utilizar un applet de Java como cliente y un servlet como un servidor y en otros casos como servidor intermedio que reenvía las peticiones del cliente a otro servidor remoto como los servidores del robot. Un servlet es un programa escrito en Java que se ejecuta en el marco de un servicio de red (un servidor Web, por ejemplo), y que recibe y responde a las peticiones de uno o más clientes. Un servlet puede manejar múltiples peticiones concurrentes y puede sincronizarlas (ver Apéndice C). Se usan peticiones/respuestas basadas en el protocolo HTTP para intercambiar información entre el cliente y el servidor. En la sección 5.5 se discute el modelo cliente-servidor en más detalle.

- **Interfaz Máquina-Robot**

La interfaz máquina-robot es un caso especial de la interfaz máquina-máquina. Las dos interfaces se pueden desarrollar utilizando el modelo de programación distribuida cliente-servidor, pero la diferencia viene del hecho de que los dos programas del cliente y el servidor, que en este caso son el servlet y el servidor remoto del robot, están escritos en diferentes lenguajes de programación (Java y C++). Además, se usa la comunicación radio como medio de comunicación entre el cliente y el servidor, lo que provoca la necesidad de cambiar el protocolo de comunicación. En este caso, se puede usar CORBA para comunicar los servlet de Java con los servidores C++ del robot utilizando un protocolo de interoperabilidad llamado IIOP (Internet Inter-ORB Protocol), que permite a los clientes usar productos CORBA de cualquier desarrollador que se comuniquen con objetos CORBA de cualquier otro desarrollador (ver Apéndice C). Este modo de comunicación entre objetos distribuidos está basado en la implementación de un patrón llamado broker o intermediario, que facilita la comunicación entre objetos remotos en los siguientes casos:

  - Los objetos deben acceder a servicios de otros objetos remotos pero sin preocuparse de su ubicación dentro de la red.
  - Se añaden, cambian o modifican objetos en tiempo de ejecución.
  - Los detalles de la implementación de los objetos deben de ocultarse a los clientes.

En la sección 5.6 se presenta el patrón broker con más detalles.

El presente capítulo se concentra en describir los patrones de software debido a que todo el diseño de las interfaces del sistema está basado en patrones de software como se puede ver en el capítulo siguiente. En el apartado siguiente se presenta el concepto de los patrones de software y sus características básicas. A continuación se describen varios tipos de patrones en el apartado 5.3. En el apartado 5.4 se hace hincapié sobre los patrones de software utilizados para desarrollar interfaces de usuario. El patrón cliente-servidor se describe en el apartado 5.5 como un patrón para las interfaces máquina-máquina. Y finalmente el apartado 5.6 se enfoca en el patrón *broker* o intermediario para las interfaces máquina-robot.

5.2 ¿QUÉ ES UN PATRÓN?

Los patrones proponen una forma de reutilizar la experiencia de los desarrolladores. Para ello clasifican y describen formas de solucionar problemas que ocurren de forma frecuente en el desarrollo. Por tanto, están basados en la recopilación del conocimiento de los expertos en desarrollo de software. Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser utilizada más de un millón de veces, sin hacerlo ni siquiera dos veces de la misma forma.

La idea de los patrones viene de una disciplina diferente a la informática, a la arquitectura. Fue un arquitecto [Alexander,77], el que a mediados de los setenta introdujo el concepto de patrón como un compendio de tres elementos. El primero, un contexto: condiciones bajo las que el patrón es útil. En segundo lugar un problema que se presenta repetidamente bajo ese contexto. Y el tercer elemento es la solución.

En programación orientada a objetos se entiende por patrón una solución probada que se puede aplicar con éxito a un determinado tipo de problemas que aparecen repetidamente en el desarrollo de sistemas software. Los patrones no son una biblioteca. Van más bien en la línea de un esqueleto básico que cada desarrollador luego adapta a sus necesidades y a las características peculiares de su aplicación. Se describen fundamentalmente en forma textual, acompañada de diagramas y de seudocódigo [Buschman,96].

Las características básicas de los patrones son las siguientes:

- **Soluciones Concretas**

  Los patrones de diseño proponen soluciones a problemas concretos, no son teorías genéricas.

- **Soluciones Técnicas**

  Los patrones indican resoluciones técnicas basadas en Programación Orientada a Objetos (POO). En ocasiones tienen más utilidad con algunos lenguajes de programación y en otras son aplicables a cualquier lenguaje.

- **Uso Frecuente**

  Los patrones de diseño se utilizan en situaciones frecuentes, ya que se basan en la experiencia acumulada al resolver problemas reiterativos.

- **Reutilización de Código**

  Ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar.

- **El uso de un patrón no se refleja en el código**

  Al aplicar un patrón, el código resultante no tiene por que delatar el patrón o patrones que lo inspiró. No obstante, últimamente hay múltiples esfuerzos enfocados a la construcción de herramientas de desarrollo basadas en los patrones y frecuentemente se incluye en los nombres de las clases el nombre del patrón en que se basan, facilitando así la comunicación entre desarrolladores.

- **Es difícil reutilizar la implementación de un patrón**

  Al aplicar un patrón aparecen clases concretas que solucionan un problema concreto y que no serán aplicables a otros problemas que requieran el mismo patrón: Una ventana es una solución para ventilar e iluminar un habitáculo, pero la ventana de mi casa no es útil en el camarote de un barco.

5.3 CLASIFICACIÓN DE LOS PATRONES

En este apartado se presenta una descripción sobre distintos patrones de software. En [Buschman,96], se han clasificando los patrones de software en los siguientes tipos:

5.3.1 Patrones de Arquitectura

Son esquemas fundamentales de organización de un sistema software. Especifican una serie de subsistemas y sus responsabilidades respectivas e incluyen las reglas y criterios para organizar las relaciones existentes entre ellos. Se recogen aquí ocho patrones estructurales, agrupados en cuatro categorías:

5.3.1.1 Del Caos a la Organización:

Niveles: Descompone una aplicación en un conjunto de capas independientes y ordenadas jerárquicamente. Cada nivel o capa usa los servicios de la capa inmediatamente inferior y ofrece servicios a la capa inmediatamente superior.

Tuberías y Filtros: Mantiene una estructura que procesa un flujo de datos. Cada paso del proceso se encapsula en un filtro. Los datos se pasan entre filtros adyacentes a través de las tuberías.

Pizarra: Este patrón es útil en problemas sin una solución determinística conocida. En la pizarra varios subsistemas trabajan conjuntamente para hallar una solución parcial o aproximada. Se puede imaginar este patrón como la traducción a software de una reunión de personas alrededor de una pizarra, dirigida por un moderador, discutiendo un tema y donde se van anotando las aportaciones que cada una de ellas realiza. Las posibles soluciones se escriben en la pizarra y se van debatiendo, hasta dar por finalizada la discusión.

5.3.1.2 Sistemas distribuidos

Intermediario o broker: Este patrón de arquitectura se usa para organizar sistemas distribuidos con componentes débilmente acoplados que interactúan entre sí invocando servicios remotos. El broker es responsable de coordinar la comunicación: cursa las peticiones de servicios remotos al servidor que corresponda en cada caso, y transmite a los usuarios los resultados de sus peticiones y las eventuales excepciones, si las hay. En la sección 5.6 se presenta una descripción más detallada sobre este patrón.

5.3.1.3 Sistemas Interactivos

MVC (Modelo-Vista-Controlador): Este patrón se usa en aplicaciones interactivas que requieren una interfaz de usuario flexible. El modelo vista-controlador se basa en delegar en el controlador la atención de los eventos y la modificación de los datos y a su vez se delega en la clase vista la representación de dicha información, es decir, la actualización de los cambios en el sistema de representación (sistema de ventanas habitualmente). Mientras que es la clase modelo la que contiene la funcionalidad del sistema y los datos.

PAC (Presentación-Abstracción-Control): También este patrón se usa en aplicaciones que requieren una interfaz de usuario flexible, usando una arquitectura de tres capas.

Los patrones de los sistemas interactivos se discuten con más detalles en el apartado 5.4.

5.3.1.4 Sistemas Adaptables

Microkernel: Este patrón se aplica en sistemas software cuyos requisitos deben de poder modificarse a lo largo del tiempo. Es apropiado cuando se necesita desarrollar una familia de aplicaciones similares que usan interfaces de programación parecidas para acceder a una funcionalidad básica común. La parte de servicios básicos debe de ser fácilmente configurable para poderla usar en todas las aplicaciones sin tener que cambiarla cuando se desarrolle una nueva aplicación, o se cambie la plataforma de hardware. Se separa una funcionalidad mínima que llama a otros servicios que están en paquetes software separados. Este núcleo o microkernel es lo más pequeño posible y consume pocos recursos del sistema (memoria, tiempo de CPU, etc.). La funcionalidad que no se encuentra en el microkernel se ubica en otros dos tipos de paquetes: servidores internos (a los que sólo el microkernel tiene acceso directo) y servidores externos, con los que se comunican los clientes.

Reflexión: Este patrón se usa para diseñar sistemas software muy flexibles y fácilmente modificables. Se trata de poder modificar la estructura o el comportamiento del sistema dinámicamente. Este patrón permite cambios en la estructura de los datos internos del sistema y en los mecanismos de llamada de las funciones. Hay dos niveles en el sistema, el nivel abstracto y el nivel básico. El nivel abstracto contiene los mecanismos de cambio del sistema: qué características se pueden cambiar y de qué manera. El nivel básico contiene la aplicación propiamente dicha. Los responsables del mantenimiento del software no hacen directamente los cambios en el código del sistema. Acceden al nivel abstracto y es el nivel abstracto el que modifica el nivel básico para que satisfaga los nuevos requisitos.

5.3.2 Patrones de Diseño

Los patrones de diseño especifican un esquema detallando los subsistemas o componentes del sistema del software, o las relaciones entre ellos. Describen una estructura de componentes de comunicación repetitiva que resuelven un problema del plan general dentro de un contexto particular. Son patrones de un nivel de abstracción menor que los patrones de arquitectura. Están por lo tanto más próximos a lo que sería el código fuente final. Su uso no se refleja en la estructura global del sistema. Se presentan aquí las descripciones de siete patrones, agrupados en cinco categorías:

5.3.2.1 Descomposición Estructural

Todo y Parte: Este patrón sirve para manejar conjuntos de objetos como una unidad. En todos los sistemas software hay objetos compuestos a su vez de otros. Las propiedades del objeto compuesto pueden ser diferentes a las de sus partes. La solución que implementa este patrón es introducir un componente (Todo) que engloba a los demás (Partes). El Todo es el componente que ve el usuario. Este Todo encapsula las partes que lo forman, organiza su colaboración y les da una interfaz de acceso homogénea. No se puede acceder directamente a las Partes.

5.3.2.2 Organización del Trabajo

Maestro Esclavo: Este patrón consiste en implementar un componente principal llamado maestro que distribuye trabajo a otros componentes secundarios, los esclavos. Cuando los esclavos terminan el trabajo que el maestro les ha asignado, le devuelven al maestro los resultados y el maestro los evalúa conjuntamente. Los esclavos son idénticos en cuanto a la funcionalidad que ofrecen.

5.3.2.3 Control de Acceso

Proxy: Este patrón consiste en comunicar a los clientes de un componente a través de una entidad intermedia. Introducir esta entidad intermedia puede servir para muchas cosas, como mejorar la eficiencia del sistema, facilitar el acceso o proteger usos no autorizados del sistema.

5.3.2.4 Gestión

Procesador de Mandos: Este patrón separa la petición de un servicio de su ejecución. El procesador de mandos guarda todas las peticiones como objetos diferentes, planifica su ejecución y ofrece servicios adicionales como el almacenamiento de los mandos ya realizados para poderlos deshacer eventualmente.

Gestor de Vistas: Este patrón se aplica en sistemas que ofrecen múltiples vistas de datos específicos de la aplicación, o que trabajan con varios documentos a la vez. Es útil para gestionar los diferentes tipos de vistas que ofrece una aplicación software. El gestor de vistas es el interlocutor que tienen los clientes para abrir, manipular y disponer las vistas. También organiza la actualización de las vistas de forma que todas presenten siempre los mismos datos.

5.3.2.5 Comunicación

Reenviador-Receptor: Este patrón se aplica en sistemas distribuidos cuyos nodos interlocutores actúan indistintamente como clientes o servidores (modelo de interacción *Peer To Peer*). Sirve para encapsular los detalles de los protocolos de comunicación de la red y ocultarlos a los nodos interlocutores. De esta manera los mecanismos de comunicación utilizados son transparentes a la comunicación entre procesos.

Cliente-Manejador-Servidor: Consiste en introducir una entidad intermedia entre los clientes y los servidores de un sistema distribuido, que es el manejador. El manejador ofrece a los clientes y a los servidores transparencia de ubicación por medio de un servicio de nombres y les oculta los detalles del establecimiento de las conexiones. Este patrón se usa en entornos distribuidos. En estos entornos es muy importante que un componente pueda acceder a un servicio sin tener en cuenta su ubicación en la red.

A continuación se explican detalladamente en las secciones siguientes los patrones de software que se han usado para desarrollar el sistema propuesto en esta tesis.

5.4 PATRONES PARA LAS INTERFACES DE USUARIO

Para desarrollar interfaces de usuario de un laboratorio remoto basado en Internet para la robótica móvil, existen unos requerimientos que deben tenerse en cuenta para garantizar el alto redimiendo y la funcionalidad. La tabla 5.1 resume los requerimientos más destacados y la solución propuesta para lograr cada requisito.

Tabla 5.1: Requerimientos y Soluciones

Requerimiento	Solución
Interfaces Intuitivas	Orientación a Usuario
Interfaces Portables	100% Java
Interfaces Extensibles	Orientación a Objetos
Interfaces Reutilizables	Patrones de Software

- **Interfaces Intuitivas:** Son necesarias para dar al usuario la habilidad de interactuar remotamente con el robot de manera correcta e interactiva. Así, las interfaces de usuario deben diseñarse según las necesidades de los usuarios y la complejidad de estas interfaces debe variar de interfaces de bajo nivel muy sofisticadas (por ejemplo como en el caso de ingenieros de la robótica) a interfaces de alto nivel muy intuitivas y atractivas (en el caso de los usuarios de los robots de entretenimiento). En todos los casos, las interfaces de usuario deben garantizar un envolvimiento continuo y activo del usuario en el sistema, proporcionándole realimentación suficiente sobre todas las tareas que el robot está llevando a cabo y notificándole continuamente acerca del estado de la conexión.

- **Portabilidad:** Esta característica garantiza el funcionamiento de las interfaces de igual manera bajo cualquier plataforma y sistema operativo. En los laboratorios remotos basados en Internet es importante tener en cuenta está característica para no obligar a los usuarios a tener ciertos requisitos como programas software adicionales o plug-ins para que puedan utilizar el sistema. De las características de Java como lenguaje de programación, se puede destacar la portabilidad del código gracias a la máquina virtual (WORA – Write Once and Run Anywhere) y la seguridad gracias al *SandBox* (ver apéndice C). Todas las interfaces de usuario en el laboratorio remoto propuesto se han implementado con java para garantizar la portabilidad de las interfaces y la seguridad de la aplicación.
- **Extensibilidad:** Es la facilidad de adaptación de las interfaces hacia los cambios, como la necesidad de modificar o extender las interfaces para que cubran nuevas tareas o para que se adapten con nuevos dispositivos de interacción. El paradigma de orientación a objetos pretende ser una red de objetos colaborativos, sin una clase principal desde la cual se controle todo desde arriba. Las diferentes partes de las interfaces se forman como objetos colaborativos, lo que facilita la modificación y la extensión del código de las interfaces. En el laboratorio remoto propuesto se han implementado las interfaces en forma de módulos, como el módulo de la odometría que proporciona información acerca de la posición y la velocidad actual del robot, módulo del sonar, módulo de láser, módulo del laboratorio, módulo de control, etc. Estos módulos se pueden colocar para formar una interfaz nueva de manera muy sencilla y rápida. También, se pueden agregar nuevos módulos al sistema.
- **Reutilización:** Se recomienda diseñar las interfaces de usuario basándose en componentes reutilizables, así se pueden construir nuevas interfaces para nuevas tareas de manera más rápida y eficaz. Los patrones de software ayudan a construir software basado en la reutilización, a construir clases reutilizables. Los propios patrones se reutilizan cada vez que se vuelven a aplicar. Por ejemplo en el laboratorio remoto propuesto se usa el patrón arquitecto "Broker" para establecer comunicación entre un servlet de java en la capa *middleware* con un servidor remoto de robot escrito en C++ a través de CORBA. Este patrón ofrece una solución del problema de comunicar dos objetos distribuidos, de tal manera que esta solución puede ser usada muchas veces con implementaciones distintas para diseñar nuevas interfaces.

En las subsecciones siguientes se discuten varios modelos que se pueden utilizar para desarrollar interfaces de usuario.

5.4.1 Modelo de Seeheim

En esta técnica de diseño como se ve en la figura 5.2, la comunicación entre el usuario y la aplicación se estructura en tres capas:

- **Capa de Presentación:** Es la parte estática y visible de la interfaz que se comunica con el usuario y se construye sobre sistemas de ventanas y cajas de herramientas. Supone el léxico de la interfaz.

- **Capa de Diálogo:** Es la parte dinámica que maneja los eventos o mensajes que se producen como consecuencia de las acciones del usuario sobre la interfaz y establece la comunicación entre el nivel de presentación y el nivel de aplicación.
- **Capa de Aplicación:** Es la parte de la aplicación que el usuario controla a través de la interfaz. Equivale a la semántica de la aplicación.

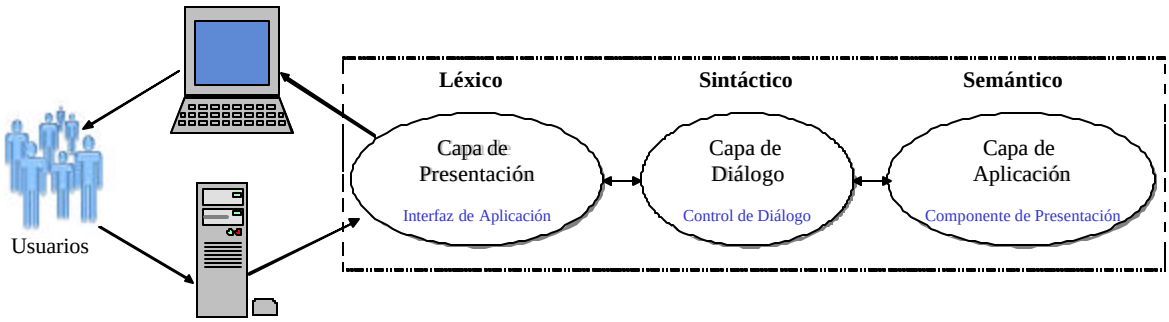


Fig. 5.2: Modelo de Seeheim

El problema principal de este modelo es la dependencia de las capas. Por ejemplo, al reemplazar un componente de la capa de presentación, se necesita volver a escribir la capa del diálogo para acomodar los nuevos rasgos del nuevo componente y viceversa, es decir, que con cualquier cambio en el diálogo, es necesario modificar el componente de la presentación.

Uno de los requerimientos importantes del diseño de interfaces de usuario es la manera en la que se pueden manejar los cambios en el sistema para minimizar los esfuerzos necesarios para modificarlo. En el modelo de Seeheim el manejo de los cambios se basa en el concepto de capas (*layering*), mientras el modelo MVC depende de encapsular la funcionalidad del sistema en distintos componentes por medio de los cuales se pueden manejar los cambios en el sistema.

5.4.2 Patrón "MVC"

Para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos, se usa el patrón modelo-vista-controlador (MVC) como en el caso del lenguaje de programación Smalltalk. En este patrón, el Modelo, las Vistas y los Controladores se tratan como entidades separadas; esto hace que cualquier cambio producido en el Modelo se refleje automáticamente en cada una de las Vistas.

Como se muestra en la figura 5.3, los componentes de la arquitectura MVC son los siguientes:

- ***Modelo*:** Refleja la estructura del modelo conceptual. El modelo es la información que manipula la aplicación, la representación de los objetos reales. Es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo.

- ***Vista*:** Es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.
- ***Controlador*:** Gestiona los eventos del usuario (dispositivos lógicos y eventos). Recibe los eventos, los interpreta y decide qué realizar con los mismos. Es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

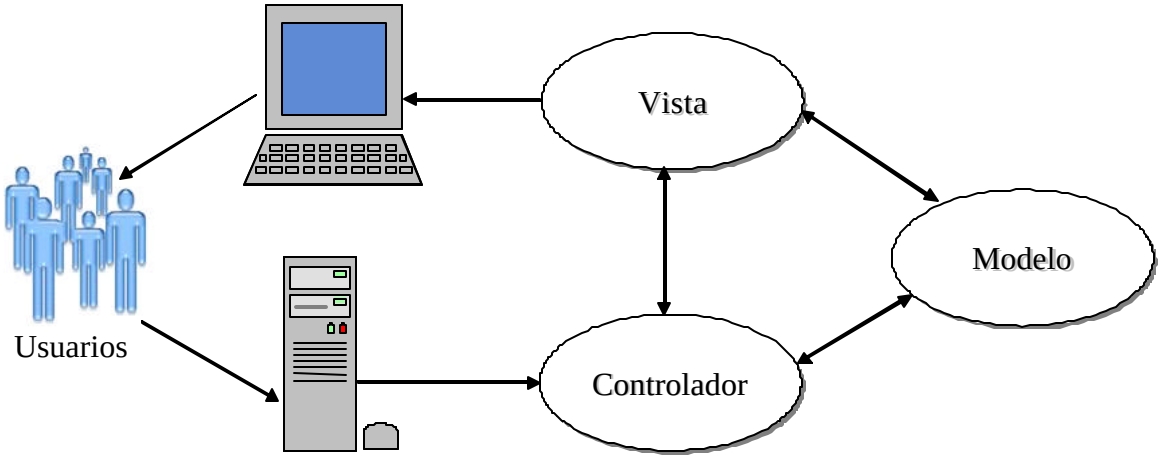


Fig. 5.3: Patrón MVC

Este patrón es adecuado para situaciones donde:

- Es necesario presentar los mismos datos de diferentes formas (gráficos circulares, gráficos de barras, etc.) o utilizando diferentes interfaces de usuario (Motif, Windows 95, etc.).
- Si cambia un dato, los datos se deben actualizar en todas las representaciones.
- La interfaz de usuario debe ser fácil de modificar para poderla adaptar a las distintas necesidades que vayan surgiendo.

Pensando en un sistema de adquisición remota de datos sensoriales, los datos sensoriales deben ser adquiridos y dibujados de distintos modos (vector, puntos, segmentos) como se muestra en la figura 5.4, en el caso de los sensores de ultrasonidos. Todos los diagramas deben reflejar en cada momento los últimos datos capturados del sonar. En este ejemplo, se supone que los datos se actualizan según un evento de actualización realizado por el usuario.

Para solucionar este problema, se puede utilizar el patrón Modelo-Vista-Controlador (MVC) que descompone la aplicación interactiva como se ha mencionado anteriormente en tres grandes bloques:

- El *modelo* contiene los datos y la funcionalidad de la aplicación. Es independiente de la representación de los datos.
- Las *vistas* muestran la información al usuario de una cierta forma.

- Cada *vista* tiene un *controlador* asociado. Los controladores reciben entradas en forma de eventos de actualización que responden a mandos realizados por el usuario a través del ratón o del teclado. El control traduce estos eventos a peticiones a la *vista* o al *modelo*.

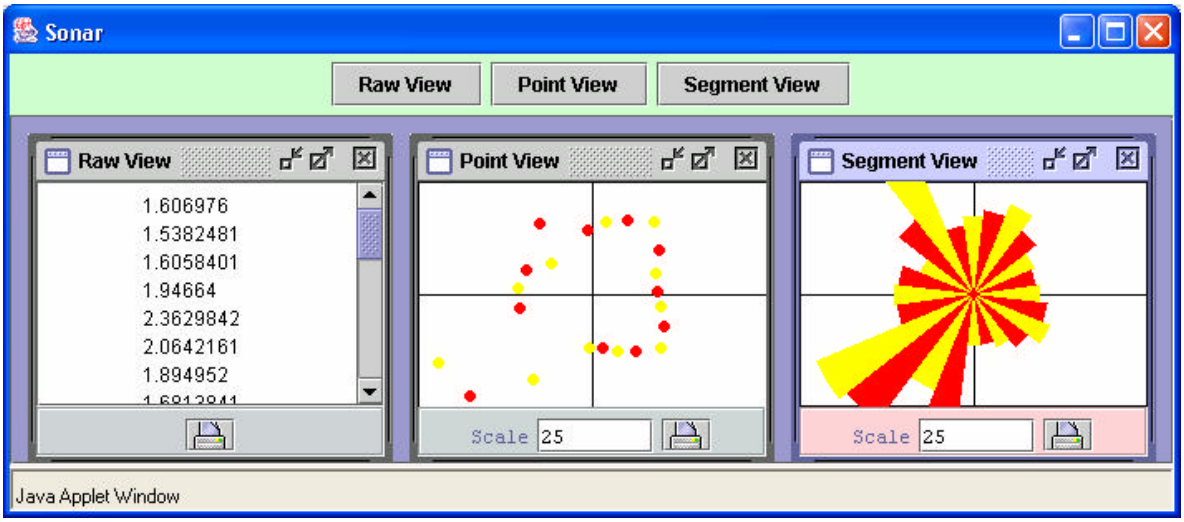


Fig. 5.4: Información Sensorial del Sonar

En las figura 5.5 y 5.6, se muestran el modelo estático y dinámico del patrón MVC. La llegada de un evento del usuario arranca el controlador que se comunica con el modelo para pedir la actualización de los datos. Las vistas piden los datos al modelo y actualizan la representación de la información. Cada controlador pide datos al modelo para permitir o no ciertas funciones. Por ejemplo, permitir imprimir automáticamente los datos actualizados puede ser consecuencia de haber modificado los datos del modelo.

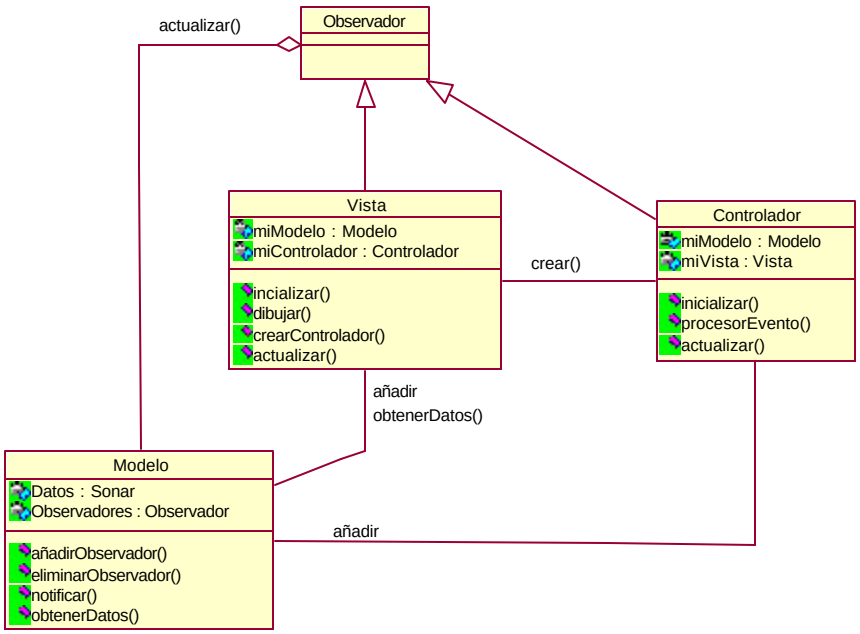


Fig. 5.5: Modelo Estático de MVC

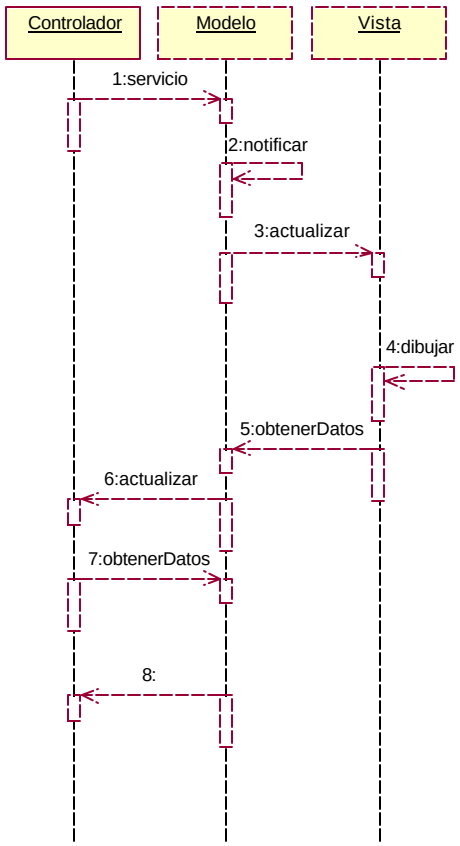


Fig. 5.6: Modelo Dinámico de MVC

La tabla 5.2 resume las ventajas y los inconvenientes del patrón MVC.

Tabla 5.2: Ventajas e Inconvenientes del Patrón MVC

Ventajas	Inconvenientes
• Múltiples vistas del mismo modelo • Vistas sincronizadas • Flexibilidad para cambiar las vistas y los controladores • La aplicación puede soportar distintos tipos de interfaz de usuario.	• Complejidad creciente. • Cambios innecesarios. Puede ser que no todas las vistas estén interesadas en todos los cambios. • Conexión entre la vista y el controlador. Hay que usar los dos a la vez. • Si cambia el interfaz del modelo, hay que cambiar todas las vistas y todos los controladores. • Acceso ineficiente a los datos en la vista. Puede necesitar varias llamadas al modelo para actualizar todos sus datos. • Tanto la vista como el controlador son específicos de una plataforma. • Algunas herramientas de diseño de interfaces de usuario incorporan parte del procesamiento de eventos entrada. El controlador deja de ser necesario.

5.4.3 Arquitectura Multiagente "PAC"

La arquitectura multiagente Presentación-Abstracción-Control (PAC) es bastante similar a la MVC, aunque existen diferencias entre ellas. Por ejemplo, PAC agrupa la entrada/salida en la presentación, mientras que en MVC la vista se ocupa de la salida y el controlador de la entrada. Como se muestra en la figura 5.7, los componentes de este patrón son:

- **Abstracción:** Representa la semántica de la aplicación
- **Presentación:** Gestiona las entradas/salidas
- **Control:** Gestiona el diálogo y la correspondencia entre la aplicación y la presentación

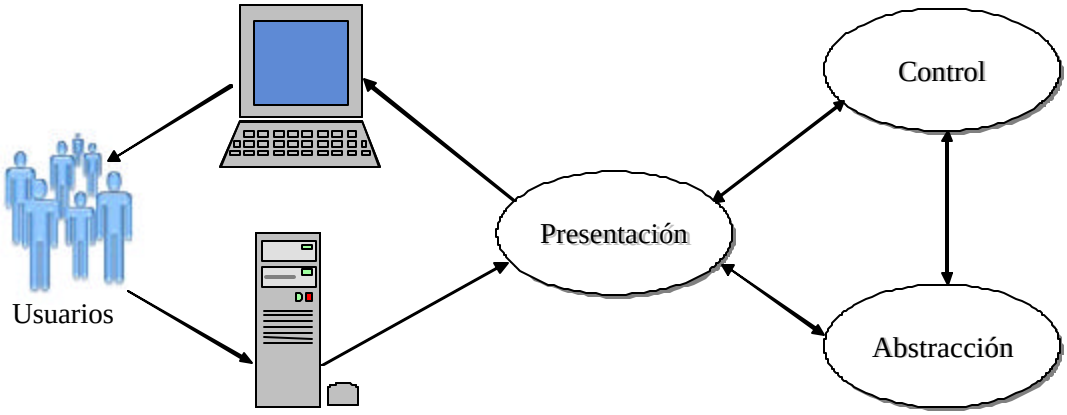


Fig. 5.7: Patrón PAC

El primer uso del patrón PAC en robótica ha sido en un sistema de control de un robot móvil descrito en [Crowley,85]. Este patrón es adecuado para situaciones donde:

- Es necesario presentar los mismos datos de diferentes formas utilizando diferentes interfaces de usuario.
- Se deben actualizar todas las representaciones cuando cambian los datos.
- El interfaz de usuario debe ser flexible, fácil de modificar, según las distintas necesidades que vayan surgiendo.

Existe por lo tanto un problema idéntico del sistema sensorial al que resuelve el patrón MVC. Sin embargo la solución es algo diferente. Se basa en el empleo de agentes cooperantes. Lo que se busca es un mayor grado de modularidad y de portabilidad. Un agente es un componente del sistema, que se responsabiliza de una funcionalidad concreta del sistema global. Un agente suele tener las siguientes características:

- Consta de tres partes: la presentación, la abstracción y el control.
- Es capaz de recibir y mandar eventos.
- Tiene estructuras de datos propias para mantener su estado.
- Tiene un procesador propio, que trata los eventos recibidos, actualiza el estado propio y que puede a su vez generar nuevos eventos.
- Su interfaz de usuario, si tiene, es propia.

Todas estas características permiten a los agentes la separación dentro de cada agente, de la interacción con el exterior *(presentación)* del núcleo funcional que contiene toda la información (*abstracción)* y la comunicación, tanto entre las dos capas anteriores como con otros agentes *(control)*.

La complejidad de un agente varía en función de la aplicación. Cada agente representa un concepto semántico diferente dentro del dominio de la aplicación. Esto facilita que la aplicación se pueda modificar con facilidad.

El sistema de presentación de los datos sensoriales del ejemplo, se estructura como una jerarquía en forma de árbol de agentes PAC, como se muestra en la figura 5.8. Cada agente es responsable de una parte específica de la funcionalidad de la aplicación.

Como se ha mencionado anteriormente, el agente PAC tiene tres partes: presentación, abstracción y control. La presentación es responsable de la interacción con el exterior del agente. La abstracción es la parte del agente que contiene el modelo de datos y que ofrece el acceso a los datos del agente. El control comunica la presentación con la abstracción y ofrece servicios de comunicación con otros agentes.

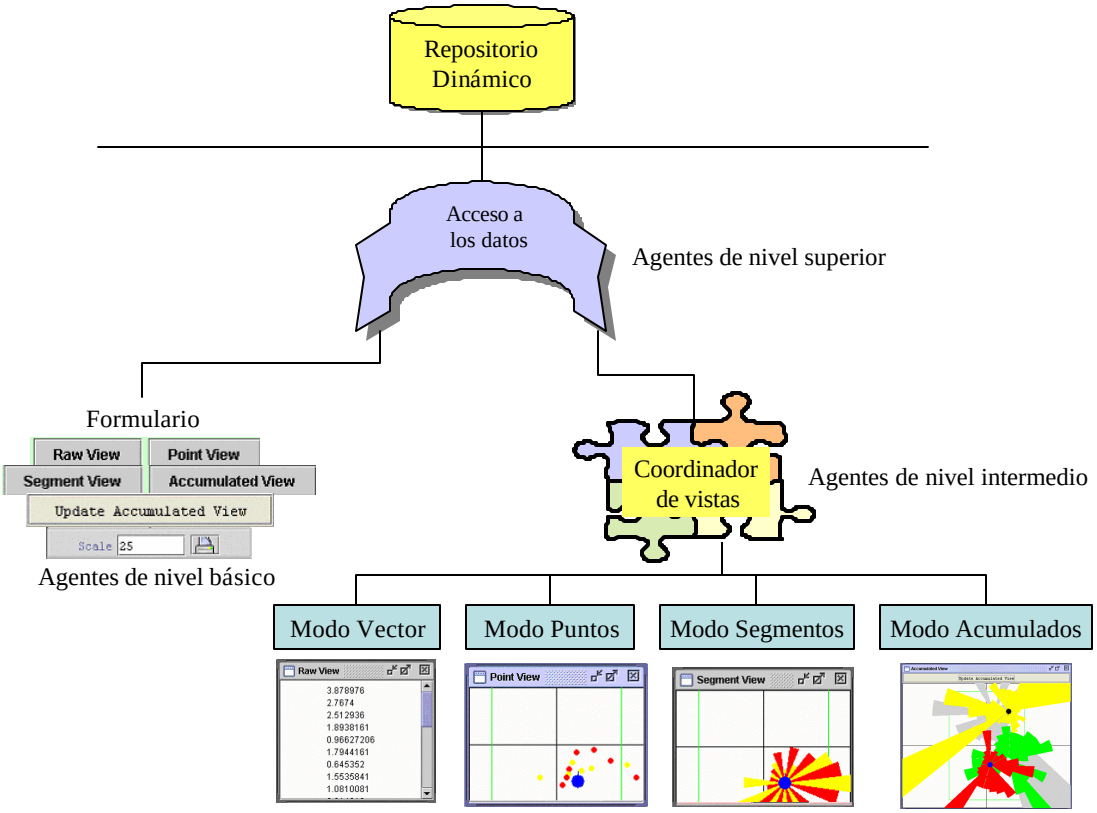


Fig. 5.8: Jerarquía de Agentes PAC

El agente PAC de nivel superior contiene el núcleo funcional del sistema. También incluye las partes de la interfaz de usuario que no pueden asignarse a subtareas particulares, como por ejemplo menús y cajas de diálogo con información sobre la aplicación. El agente PAC de nivel básico representa una entidad con la cual trabajan los usuarios del sistema como formularios o diagramas. El agente PAC de nivel intermedio representa una combinación o una relación de

agentes de nivel básico. A continuación, se muestra en la figura 5.9 el diagrama de clases del sistema ejemplo.

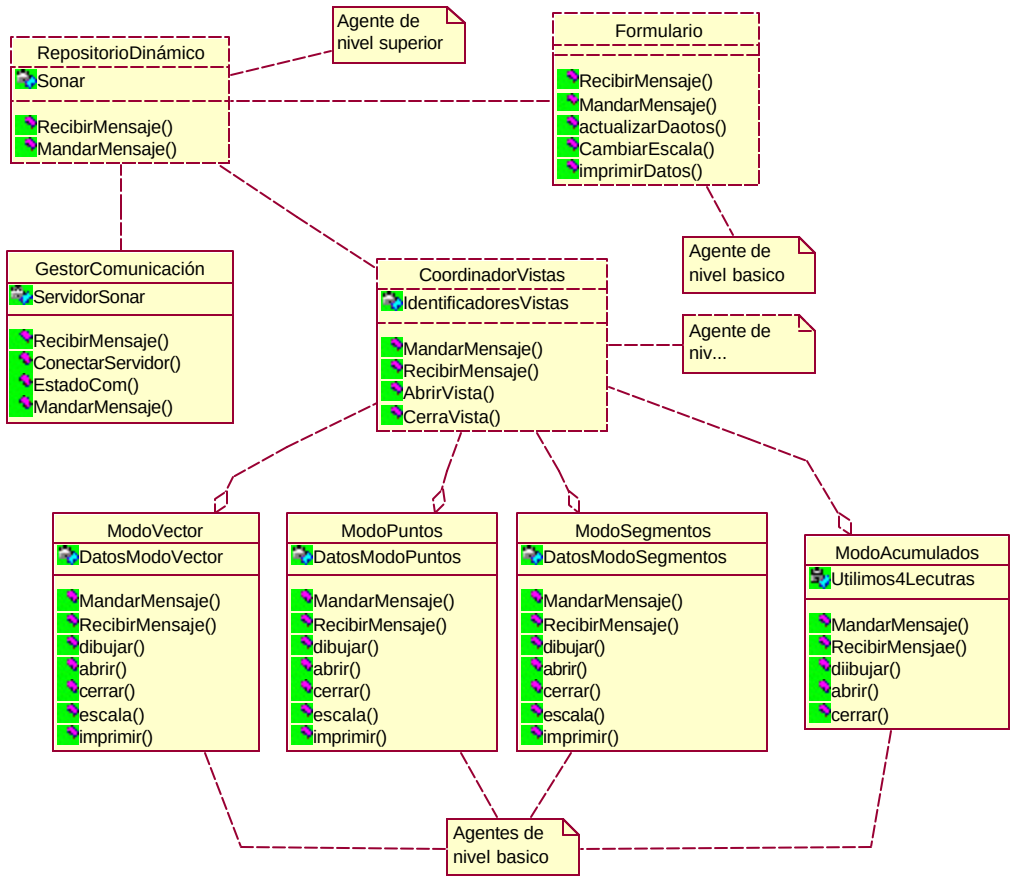


Fig. 5.9: Diagrama de Clases del Sistema Sensorial

La figura 5.10 muestra los agentes que forman el modo segmentos,.

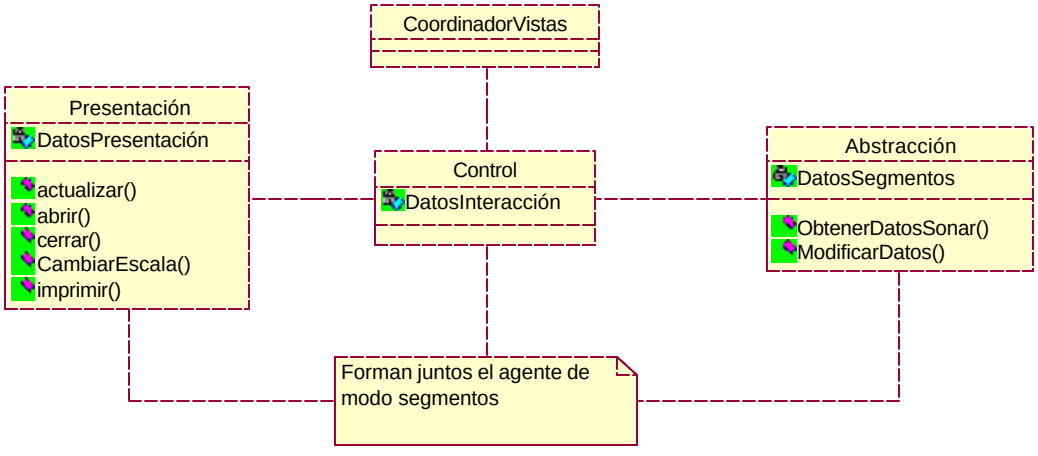


Fig. 5.10: Diagrama de Clases del Agente del Modo Segmentos

A continuación se presentan los dos escenarios siguientes como aplicaciones de este patrón:

- **Escenario I:** Apertura de un nuevo gráfico de modo segmentos

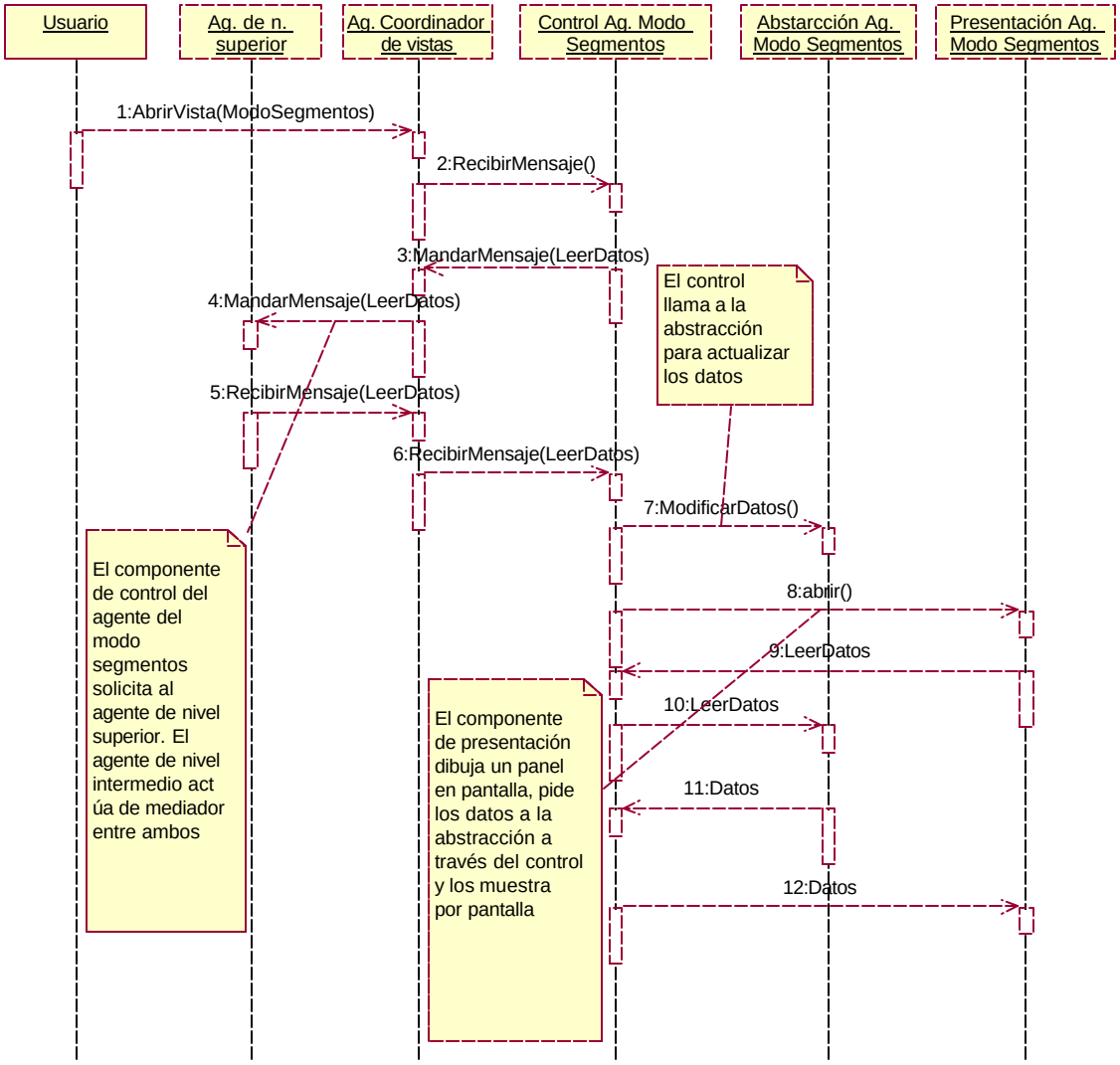


Fig. 5.11: Diagrama de Secuencia del Escenario I

- **Escenario II:** El usuario pide actualizar los datos sensoriales.

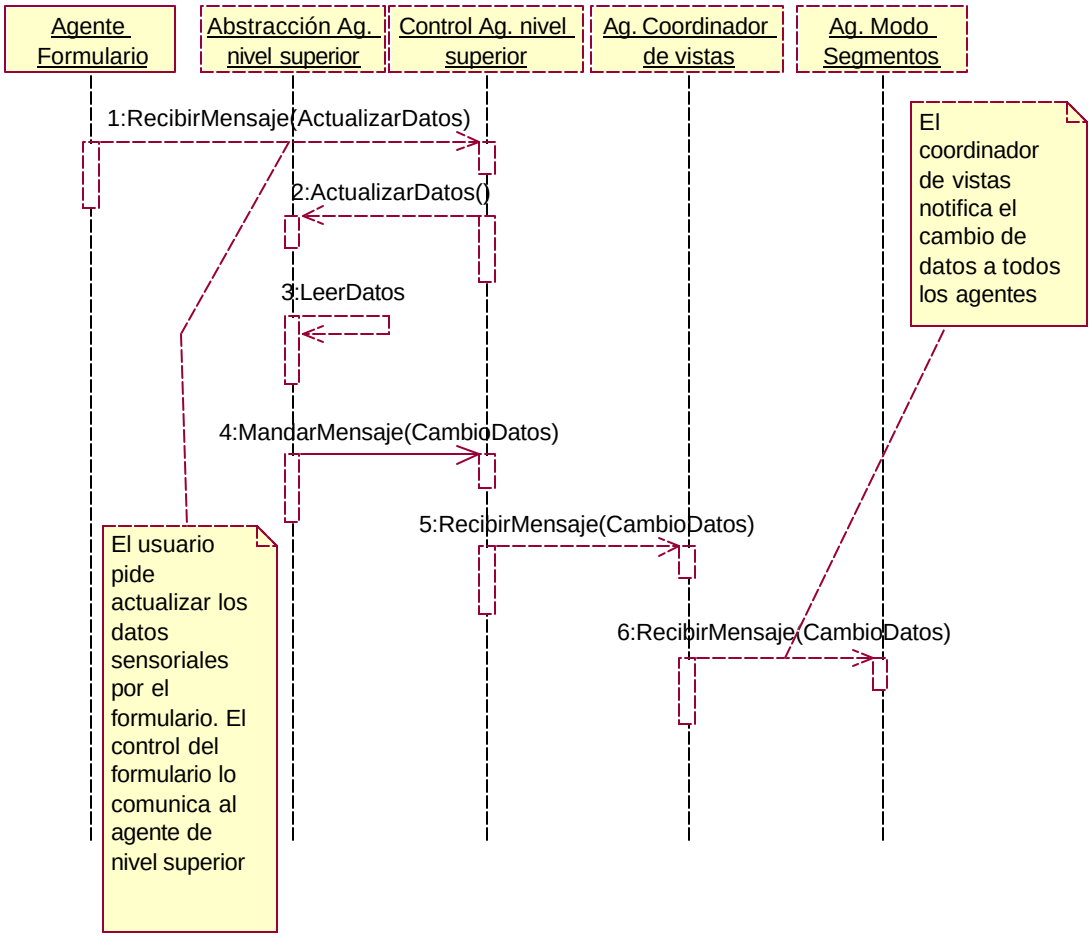


Fig. 5.12: Diagrama de Secuencia del Escenario II

En la tabla 5.2 se resumen las ventajas y los inconvenientes del patrón MVC.

Tabla 5.3: Ventajas e Inconvenientes del Patrón PAC

Ventajas	Inconvenientes
• Separación de conceptos. • Facilidad para cambiar y ampliar la aplicación • Soporte para multitarea.	• Mayor complejidad en el sistema. • Control de componentes complicado. • Baja eficiencia, por la sobrecarga que introduce la comunicación entre agentes. • Si los agentes son muy pequeños, las estructuras que hay que manejar son muy grandes. Por ejemplo un editor donde cada objeto de texto lo maneja un agente.

5.4.4 Arquitectura de "Proxy Visual"

Se ha presentado el patrón proxy visual como un caso especial del patrón PAC para desarrollar interfaces de usuario utilizando el paradigma de orientación a objetos [Houlb,99]. Un *proxy* visual es una referencia inteligente de otro objeto (el objeto real) que materializa el objeto real cuando se referencia, por lo tanto implementa materialización bajo demanda. Una procuración visual se considera como una referencia inteligente porque se trata por un cliente como una referencia al objeto real. El *proxy* visual implementa la misma interfaz que el objeto real, por eso el cliente ve el *proxy* visual como si fuera el objeto real.

El patrón "*proxy* visual" es una especialización de la arquitectura PAC que define una estructura para sistemas de software interactivo como una jerarquía de agentes cooperantes como se ha mencionado anteriormente. Cada agente es responsable de un aspecto específico de la funcionalidad.

Al igual que la arquitectura PAC, la arquitectura de *proxy* visual consta, como se muestra en la figura 5.13, de los niveles de Presentación, Abstracción y Control.

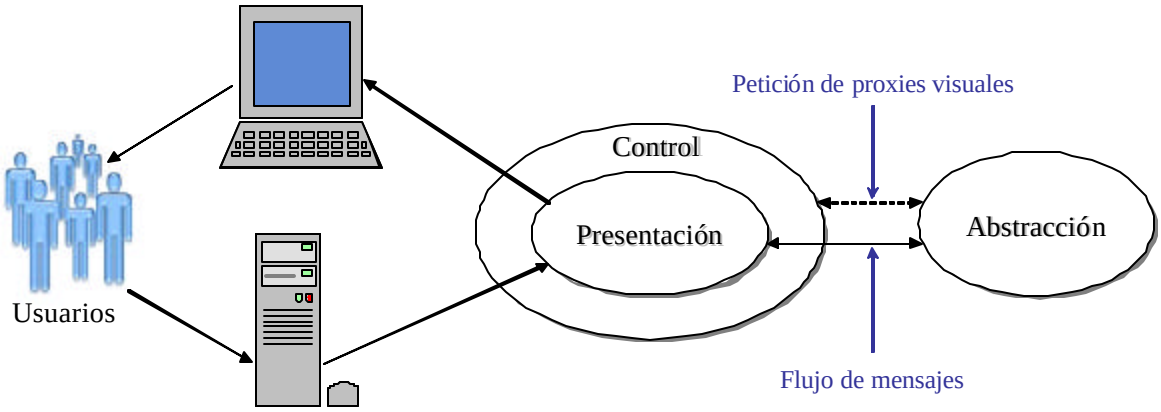


Fig. 5.13: El Patrón Proxy Visual

Este patrón permite la construcción de interfaces de usuario para sistemas orientados a objetos y garantizando la extensibilidad y la reutilización de los componentes del software implementados. Las tres capas que forman la arquitectura del proxy visual son:

- **Capa de Abstracción:** Es el grupo de objetos que directamente modela los conceptos abstractos en el dominio del problema. Esta capa es equivalente al modelo en la arquitectura MVC.
- **Capa de Presentación:** Son las clases que se encargan de dibujar la pantalla. Esas clases se pueden considerar como *proxies* visuales porque ellas realmente son *proxies* - representación – de los objetos de la capa de abstracción. En esta capa se gestionan las entradas y las salidas.
- **Capa de Control:** Pide los *proxies* visuales desde la capa de abstracción para construir la pantalla vista por el usuario. El objeto de control, entonces, está encargado de formar la interfaz de usuario.

Aunque las capas de presentación y abstracción son realmente análogas al modelo y la vista de MVC, el objeto de control realmente no tiene ninguna relación. Un controlador en MVC es un objeto que acepta eventos desde el lado de la vista y los traduce en mensajes al lado del modelo. El objeto de control de PAC, por otra parte, es pasivo con respecto al flujo de mensajes. Los mensajes van directamente desde el *proxy* visual en la capa de presentación al objeto de la capa de abstracción que creó el *proxy*. El objeto de control funciona como contenedor de los *proxies* visuales para formar la interfaz de usuario sin estar involucrada en el flujo de mensajes entre los *proxies* visuales y los objetos de abstracción.

Volviendo al ejemplo anterior del sistema sensorial, se supone que se pretende utilizar el patrón proxy visual para desarrollar la interfaz de usuario mostrada en la figura 5.14.

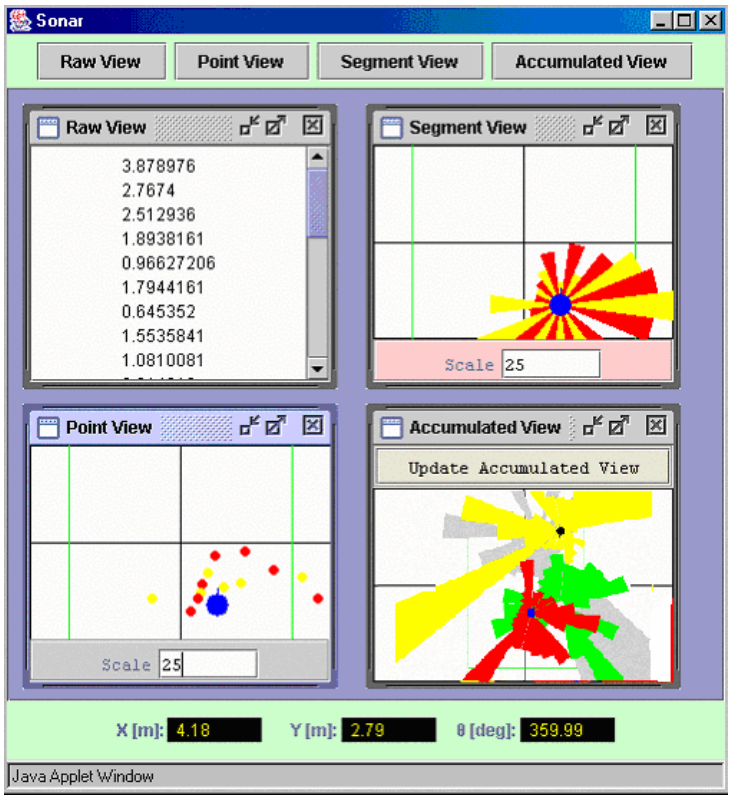


Fig. 5.14: Interfaz de Usuario del Sonar

Esta interfaz consiste en varios elementos como marcos de las vistas de los datos, un panel de la odometría y un panel de botones. La figura 5.15 muestra el diagrama de clases del patrón *proxy* visual que se puede usar para desarrollar esta interfaz.

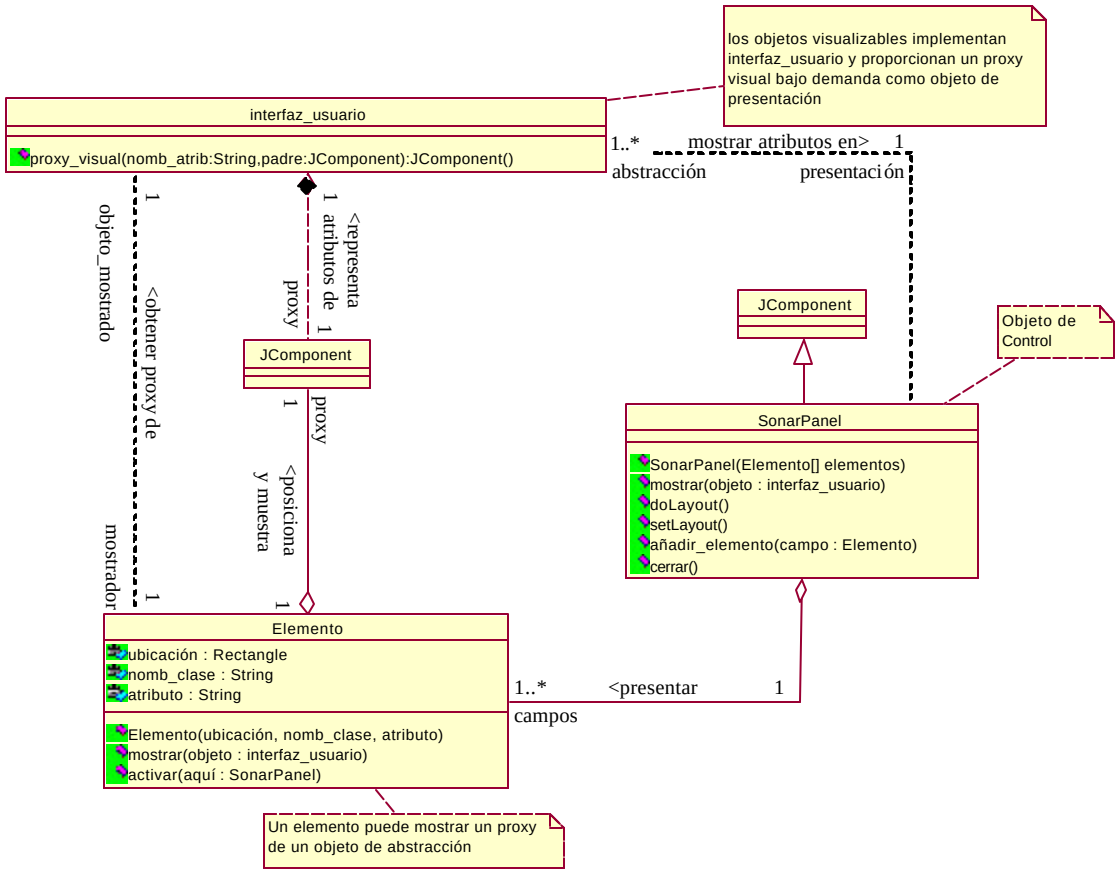


Fig. 5.15: Diagrama de Clases de Proxy Visual

En este diagrama, la clase *SonarPanel* es una forma genérica o un panel que se construye para mostrar proxies que representan las propiedades de la capa de abstracción. Esta clase se puede considerar como el objeto de control del PAC. La tarea principal de esta clase es pedir proxies a los objetos visualizables y organizar estos proxies en la ventana para formar la interfaz. *SonarPanel* extiende *JComponent*, por lo tanto un objeto puede usar todo el panel como un proxy formando así un patrón compuesto (paneles dentro de paneles). El *Elemento* es una clase privada que sostiene *ubicación* rectángulo y varias cadenas de caracteres que identifican la clase y los nombres de los atributos. Los elementos se colocan en el panel llamando al método *añadir_elemento (Campo:Elemento).*

En el diagrama de secuencia (véase la figura 5.16) se puede ver como las clases de la arquitectura PAC trabajan juntas para formar la interfaz de usuario.

A continuación, la tabla 5.4 muestra las ventajas y los inconvenientes en utilizar la arquitectura del proxy visual. Como se puede ver en el capítulo siguiente, se ha utilizado la arquitectura del proxy visual para desarrollar las interfaces de usuario del sistema propuesto en esta tesis con algunas modificaciones para evitar los problemas asociados con el nivel elevado de su complejidad.

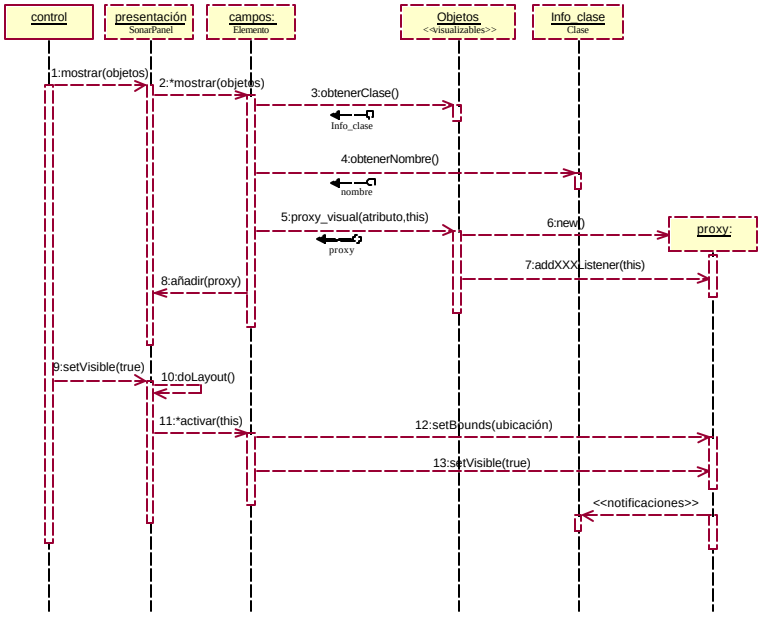


Fig. 5.16: Diagrama de Secuencia de Proxy Visual

Tabla 5.4: Ventajas e Inconvenientes del Patrón Proxy Visual

Ventajas	Inconvenientes
• Una separación real entre el nivel de presentación y la abstracción facilitando así cualquier modificación o extensión del sistema implementado. • Mayor eficiencia, por limitar el flujo de mensaje a estar entre el proxy visual y su objeto real en la capa de abstracción • El patrón de proxy visual proporciona interfaces de usuario flexibles con relaciones de acoplamiento mínimas entre los subsistemas. La generación de la interfaz de usuario está completamente separada de los objetos de la capa de abstracción proporcionando así facilidad de reutilización y extensibilidad. El único acoplamiento está entre el proxy visual en la capa de presentación y el objeto real en la capa de abstracción • Soporte para multitarea	• Los argumentos de esta arquitectura son persuasivos en términos de metodología de orientación a objetos, pero hace el desarrollo de interfaces de usuario más complejo y más lento. • Algunos consideran la arquitectura de proxy visual como un tipo de fundamentalismo religioso de orientación a objetos por su concentración en los conceptos de orientación a objetos. • La arquitectura recomienda no utilizar los métodos get/set por eso hace el desarrollo más difícil. • Esta arquitectura no se ha usado todavía en proyectos grandes para saber si puede usarse para desarrollar interfaces de usuario de un sistema a gran escala o no.

5.5 PATRONES PARA LAS INTERFACES MÁQUINA-MÁQUINA

La comunicación entre dos máquinas se puede desarrollar utilizando varios patrones de software como el patrón reenviador-receptor, proxy, cliente-manejador-servidor, etc. Todos estos patrones están basados en comunicar un cliente con un servidor para poder intercambiar peticiones y respuestas como se muestra en la figura 5.17.

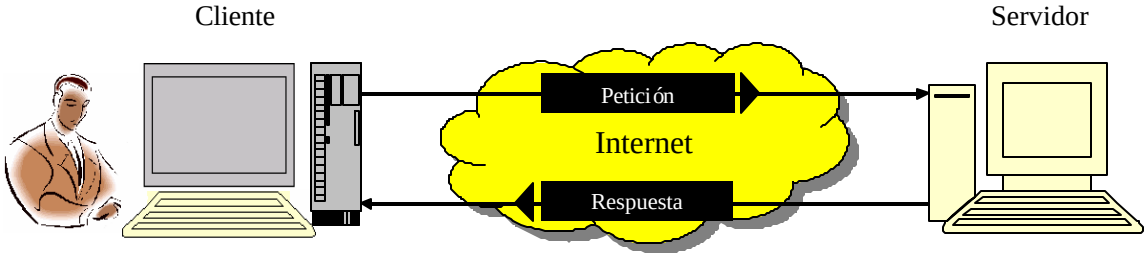


Fig. 5.17: Cliente-Servidor

En este modelo cliente-servidor, un cliente suele ser un programa que se ejecuta bajo demanda de un usuario humano y que precisa del acceso a algún recurso localizado en una máquina diferente. Para conseguir ese acceso, lo que hace es ponerse en contacto, gracias a la red, con el servidor que gestiona ese recurso en la máquina remota. Un servidor suele ser un programa que está constantemente en ejecución en una máquina. De no estar constantemente activo, existe algún mecanismo para que se "despierte" o ponga en marcha cuando sea necesario. El servidor está especializado en realizar una cierta tarea: gestionar una impresora, administrar un disco compartido a través de la red, hacer público un directorio con ficheros, etc. Es decir, el término servidor es aplicado a cualquier programa que ofrece servicios que pueden ser obtenidos en la red. El servidor acepta la solicitud recibida por la red, ejecuta los servicios solicitados, y retorna el resultado al programa que lo ha solicitado. Para los servicios simples o sencillos, cada solicitud llega en un paquete IP simple y el servidor retorna la respuesta en otro paquete IP.

La posibilidad de escribir aplicaciones distribuidas de forma rápida y sencilla es uno de los principales atractivos de Java. Java es una de las pocas herramientas que permiten a cualquier programador (sin necesidad de conocimientos avanzados de comunicaciones) escribir programas que se integren fácilmente en una red IP para acceder a bases de datos remotas, interactuar con otros programas (escritos o no en Java) y distribuir datos o aplicaciones a través de Internet (ver apéndice C).

Como se sabe, todas las máquinas conectadas a una red IP, bien sea la red pública Internet o una red privada, se distinguen por su dirección IP. Esta dirección IP es un número de 32 bits, que por comodidad suele expresarse en forma de 4 números decimales separados por puntos. Cada uno de estos números se corresponde con 8 bits de la dirección IP. Por ejemplo: 163.117.150.171 es una dirección IP. Esta dirección IP puede ser fija o puede ser distinta cada vez que la máquina se conecta a la red. Esto es lo que ocurre a casi todos los usuarios que se conectan a Internet a través de la línea telefónica. Para facilitar todavía más el trabajo con direcciones IP, existen los nombres de dominio. Estos nombres de dominio son cadenas alfanuméricas, más fáciles de recordar, y que suelen tener una única dirección IP asociada. Siguiendo con el ejemplo anterior, modelado1.uc3m.es es el nombre de dominio asociado con la dirección IP 163.117.150.171. Los encargados de traducir los nombres de dominios en dirección IP son los servidores DNS (Domain

Name Server). Estos servidores DNS mantienen unas tablas de correspondencias entre direcciones y dominios. Estas tablas se actualizan periódicamente a medida que los distintos servidores DNS intercambian sus datos entre sí. La forma general de establecer una comunicación a través de Internet es: 1) Indicar la dirección IP de la máquina con la que se quiere conectar y 2) especificar el número de puerto dentro de esa máquina a través del cual se quiere establecer la comunicación. Para que la comunicación se pueda establecer debe haber un proceso en esa máquina "escuchando" en el puerto especificado. Generalmente, cada máquina tiene una serie de servicios escuchando en ciertos puertos estándar: el servidor HTTP (servidor Web) en el puerto 80, el servidor FTP en el puerto 21, el puerto 25 para SMTP (correo electrónico), telnet en el puerto 23, etc. Estos puertos están asignados en el estándar RFC 1700.

La forma más común de transmitir información a través de Internet es mediante los protocolos de transporte TCP (Transport Control Protocol) y UDP (User Datagram Protocol). Estos protocolos se diferencian principalmente en que TCP (definido en RFC 793) está orientado a la conexión (es decir, necesita el establecimiento de una conexión entre ambos extremos y realiza un complejo control de errores), mientras que UDP (definido en RFC 768) se basa en el envío de paquetes individuales, sin establecimiento previo de una conexión y sin control de errores. Obviamente, cada tipo de comunicación tiene sus ventajas e inconvenientes, y será necesario decidirse por un protocolo u otro en función de las necesidades de cada aplicación.

El hecho de que el protocolo subyacente sea TCP oculta los detalles relacionados con la pérdida de datos, ya que es el propio protocolo el encargado de hacerlo. A todos los efectos, se puede tratar una conexión TCP como un canal carente de errores. Se ha utilizado este protocolo en el desarrollo de esta tesis. A continuación se dan dos ejemplos de la comunicación máquina-máquina basados en el patrón cliente-servidor utilizando el protocolo TCP.

El patrón cliente-servidor facilita el flujo de datos entre una máquina cliente y otra que funciona como servidor. Se puede imaginar un flujo como un tubo donde se puede leer o escribir bytes. No importa lo que pueda haber en el otro extremo del tubo: puede ser un teclado, un monitor, un archivo, un proceso, una conexión TCP/IP o un objeto Java. Todos los flujos que aparecen en Java (englobados generalmente en el paquete java.io) pertenecen a dos clases abstractas comunes: *java.io.InputStream* para los flujos de entrada (aquellos de los que se puede leer) y java.io.OutputStream para los flujos de salida (aquellos en los que se puede escribir). Estos flujos, como se ha mencionado antes, pueden tener orígenes diversos (un archivo, un socket TCP, etc.), pero una vez que se tiene una referencia a ellos, se puede trabajar siempre de la misma forma: leyendo datos mediante los métodos de la familia *read()* o escribiendo datos con los métodos *write()*. En las siguientes subsecciones, se plantean dos ejemplos de cómo se pueden trasmitir cadenas de caracteres y objetos en java debido a que la comunicación entre la máquina del cliente y la máquina del servidor intermedio en el sistema desarrollado en esta tesis está implementada en Java.

- **Transmisión de Cadenas de Caracteres**

Como se muestra en la figura 5.18, se supone que el cliente manda una petición en forma de una cadena de caracteres al apretar el botón *Enviar* en la interfaz grafica de usuario. El servidor (*PetResServlet*) recibe esta petición y devuelve la misma cadena de caracteres pero de forma inversa.

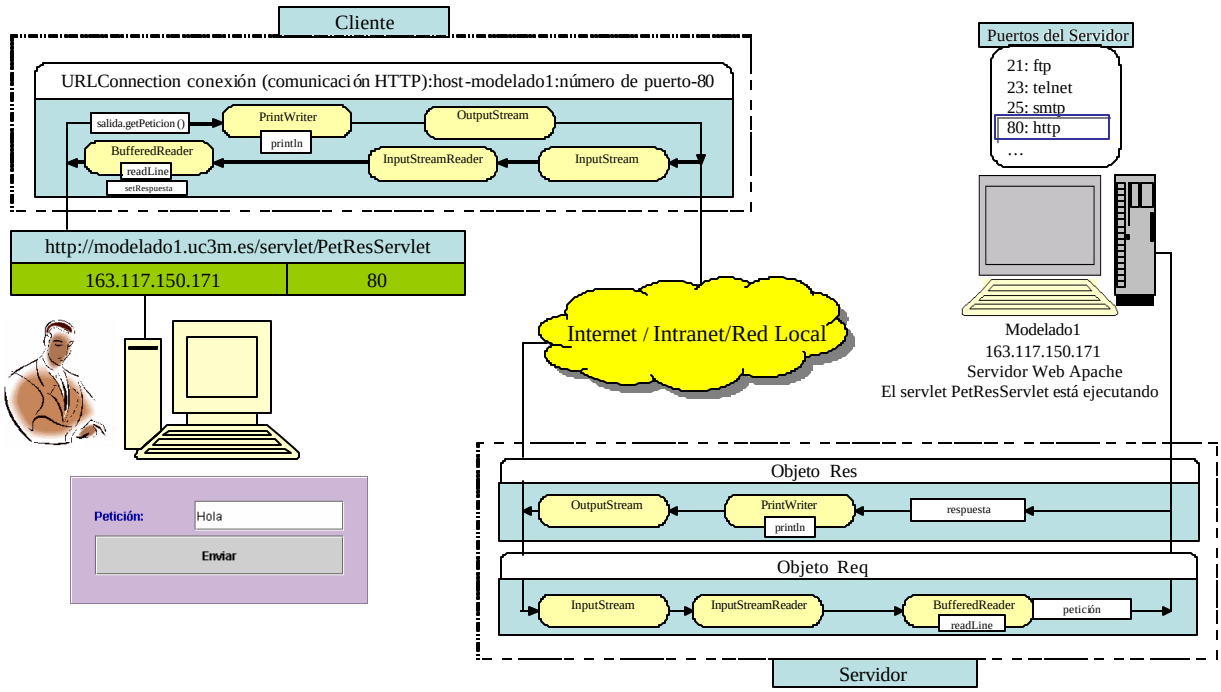


Fig. 5.18: Transmisión de Cadenas de Caracteres

Los listados 5.1 y 5.2 muestran los pasos necesarios para establecer la comunicación entre las dos maquinas y mandar y recibir una cadena de caracteres.

```
void Enviar_actionPerformed(ActionEvent e) {

          try {

               // Crear una conexión con el servlet
               URL url = new URL("http://modelado1.uc3m.es/servlet/PetResServlet");
               URLConnection conexion = url.openConnection();
               conexion.setDoOutput(true);    // Permite el envio de la informacion al metodo
                                                 doPost del servlet
               conexionn.setUseCaches(false);  // No usar el cache

               PrintWriter salida = new PrintWriter(conexion.getOutputStream(), true);
               salida.println(getPeticion()); //enviar la peticion

               BufferedReader entrada = new BufferedReader(new
               InputStreamReader(conexion.getInputStream()));
               setRespuesta(enterada.readline()); //recepcion de respuesta

               }
           catch (IOException error) {
                 error.printStackTrace();
           }
 }
```

Listado 5.1: Cliente para Trasmitir Cadenas de Caracteres

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;


public class PetResServlet extends HttpServlet {

          public void doPost(HttpServletRequest req, HttpServletResponse res) throws
          ServletException, IOException {

                    // Applet Connection Reference
          BufferedReader entrada = new BufferedReader(new
          InoputStreamReader(req.getInputStream()));
          String peticion=entrada.readLine();
          String respuesta="";

          for(int i=peticion.lenth()-1;i>=0; i--) repuesta+=peticion.charAt(i);
          PrintWriter salida=new PrintWriter(res.getOutputStream(), true);
          salida.println(respuesta);
          }
}
```

Listado 5.2: Servidor para Trasmitir Cadenas de Caracteres

- **Transmisión de Objetos**

Una vez está abierta la conexión con el servidor, se puede enviar objetos por serialización en vez de simplemente texto ASCII o datos binarios. En la figura 5.19, se muestra como se envía un objeto que puede ser un fichero o un vector, etc. desde el applet al servlet.

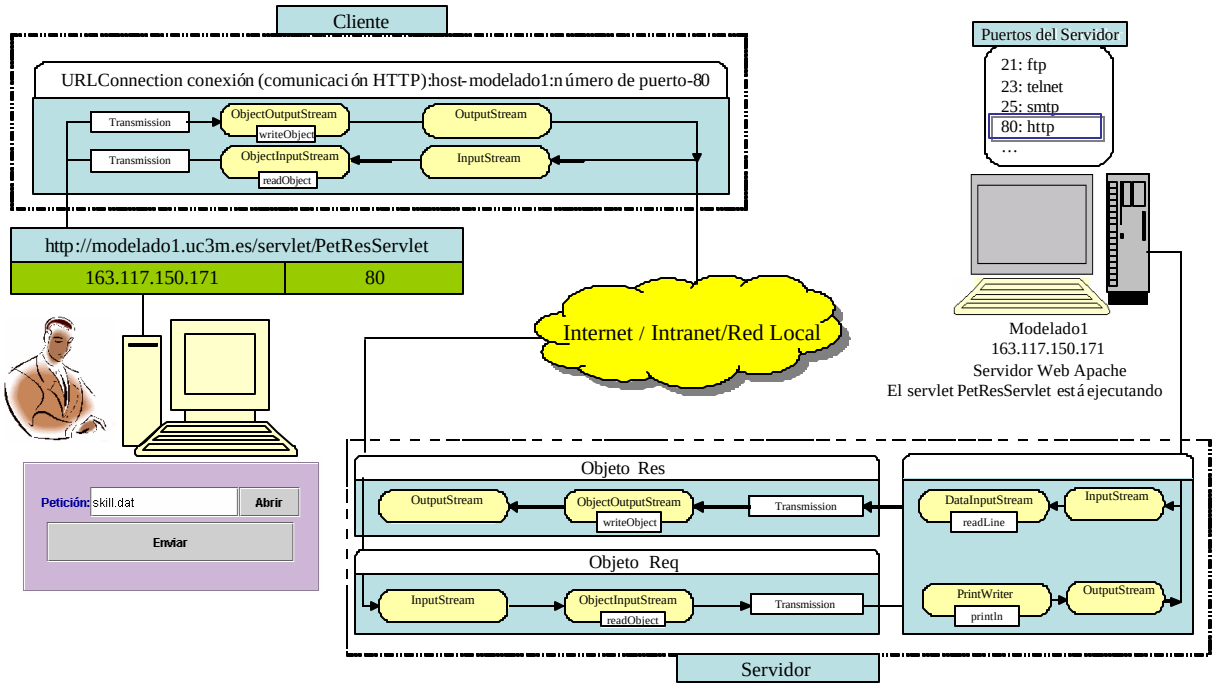


Fig. 5.19: Transmisión de Objetos

A continuación, se muestran en los listados 5.3, 5.4 los pasos necesarios para enviar una petición en forma de objeto al servlet.

```
void Enviar_actionPerformed(ActionEvent e) {

  try {

        // Crear una conexión con el servlet
        URL url = new URL("http://modelado1.uc3m.es/servlet/PetResServlet");
        URLConnection conexion = url.openConnection();

        conexion.setDoOutput(true);    // Permite el envio de la informacion al metodo doPost del
                                          servlet
        conexionn.setUseCaches(false);  // No usar el cache
        ObjectOutputStream salida = new ObjectOutputStream(conexion.getOutputStream());
        salida.writeObject(new Transmission(1234, getPeticion(), true));

        ObjectInputStream  entrada=new ObjectInputStream(conexion.getInputStream());
        Transmission tranmision=(Transmission)entrada.readObject();
        setRespuesta(transmission.getMessage());

       }
      catch (IOException error) {
          error.printStackTrace();
      }
 }
```

Listado 5.3: Cliente para Trasmitir Objetos

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;


public class PetResServlet extends HttpServlet {

          public void doPost(HttpServletRequest req, HttpServletResponse res) throws
          ServletException, IOException {

                    ObjectInputStream entrada=new ObjectInputStream(req.getInputStream());
                    Tranmission transmision=(Transmission)entrada.readObject();
                    int numeroPuerto=transmision.getPort();

                    try{

                    Socket conexion=new Socket("modelado1", numeroPuerto);
                    PrinterWriter enviar=new PrintWriter(conexion.getOutputStream(), true);
                    enviar.println(transmision.getMessage());
                    DataInputStream recebir=new DataInputStream(conexion.getInputStream());
                    transmision=new Transmission(numeroPuerto, recebir.readLine(),true);
                    }
                    catch(Exception error){
                        transmision=new Transmission(numeroPuerto, error.getMessage(), false);
                    }
                    ObjectOutputStream salida=new ObjectOutputStream(res.getOutputStream());
                    salida.writeObject(transmision);
                    }
                    catch(Exception error){
                       error.printStackTrace();
                    }
}
```

Listado 5.4: Servidor para Trasmitir Objetos

En el capítulo siguiente, se puede ver como se usa este patrón para comunicar los clientes de las interfaces con los servidores del middleware.

5.6 PATRONES PARA LAS INTERFACES MÁQUINA-ROBOT

El patrón Broker o Intermediario se usa para organizar sistemas distribuidos con componentes débilmente acoplados que interactúan entre sí invocando servicios remotos. El broker es responsable de coordinar la comunicación: cursa las peticiones de servicios remotos al servidor que corresponda en cada caso, y transmite a los usuarios los resultados de sus peticiones y las eventuales excepciones, si las hay.

Por ejemplo, el sistema de información turística pone a disposición del público diferentes contenidos. Los usuarios se conectan por Internet al servidor Web y desde ahí el sistema de información llama a una aplicación u otra. Los usuarios no ven que el sistema se compone de varios servidores. El sistema evoluciona rápidamente y por eso se necesita que los servicios sean independientes unos de otros. La figura 5.20 muestra el ejemplo del servidor de planos de la ciudad. Este servidor está en el ayuntamiento. Los demás servidores a los que accede el sistema están en otros sitios.

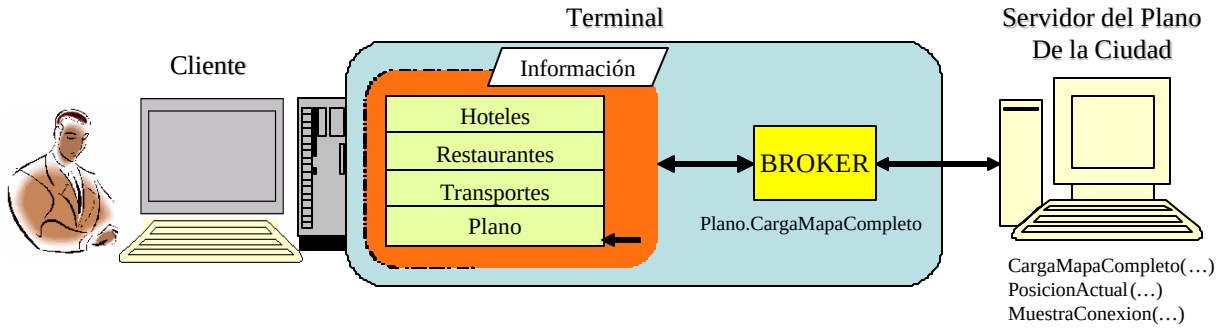


Fig. 5.20: Broker o Intermediario

La figura 5.21 muestra el diagrama de secuencia del patrón broker que ilustra el intercambio de informaciones en tiempo real entre los distintos componentes del patrón.

Como se muestra en el diagrama, en este patrón participan seis tipos de componentes: *clientes, servidores, broker o pasarela o intermediario, proxy en el cliente* y *proxy en el servidor*.

- El cliente es una aplicación que accede a los servicios de uno o más servidores a través del broker. El cliente utiliza para comunicarse con el broker un *proxy* o filtro, llamado *proxy* del cliente o stub.
- El servidor implementa unas ciertas funcionalidades (servicios). Se registra en el broker indicándole su nombre y ubicación. Manda sus respuestas y mensajes de error al cliente a través de su filtro (proxy en el servidor o esqueleto).
- La pasarela es un componente opcional que encapsula detalles de implementación si en el sistema hay dos o más brokers que usan diferentes protocolos de red.
- Los proxies del cliente y del servidor están para realizar operaciones del tipo:

  Traducción de los datos a formatos independientes de la máquina.

  Encapsular funciones específicas de la máquina del cliente o del servidor.

- El broker está entre los clientes y los servidores, que se comunican a través de él. Registra y da de baja a los servidores. Ofrece APIs a los clientes y a los servidores para que le puedan llamar, registrarse, etc. Se comunica con otros brokers a través de las pasarelas.

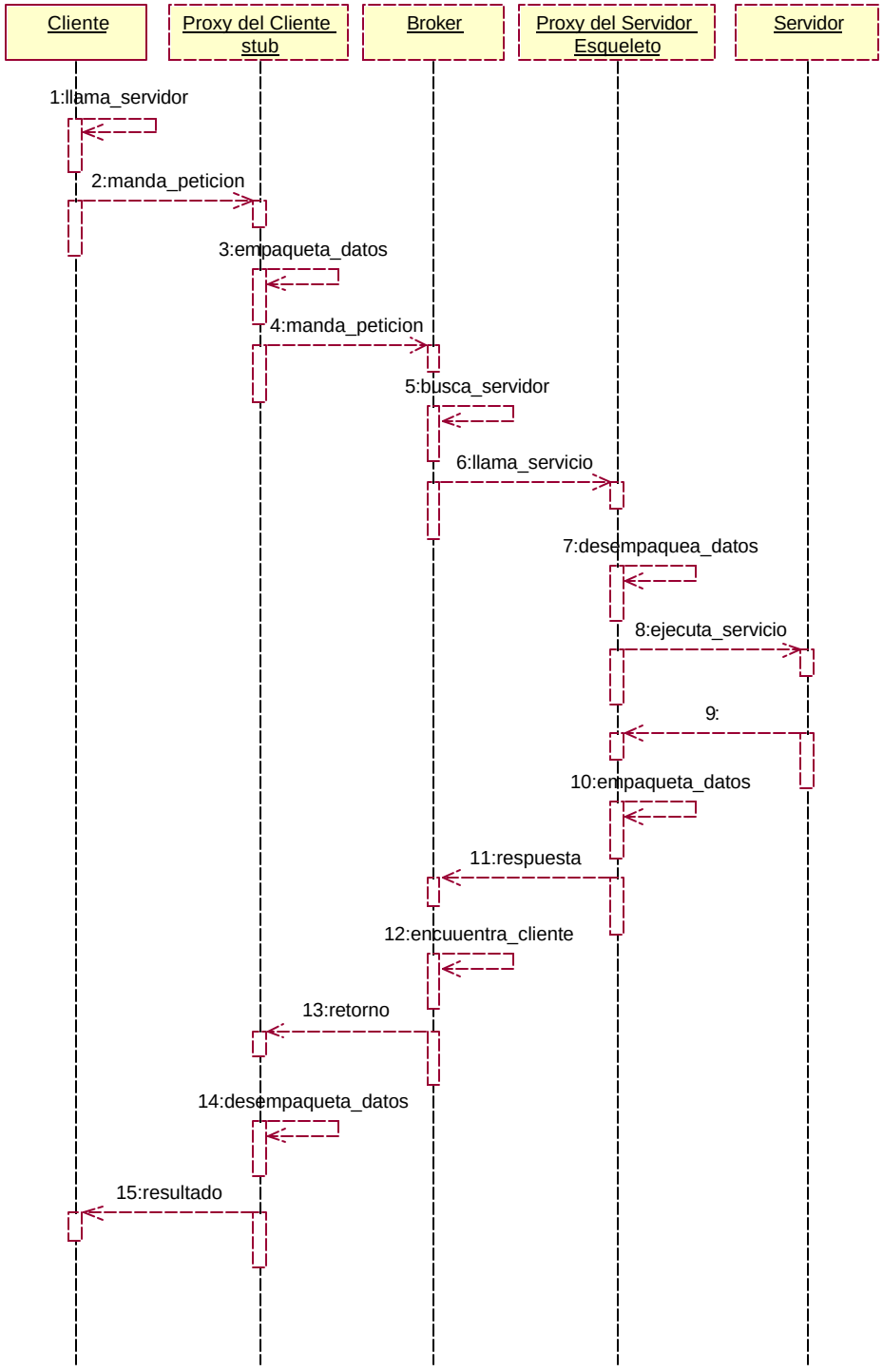


Fig. 5.21: Diagrama de Secuencia del Patrón Broker

En la tabla 5.5 se resumen las ventajas y los inconvenientes de este patrón.

Tabla 5.5: Ventajas e Inconvenientes del Patrón Broker

Ventajas	Inconvenientes
• Transparencia de ubicación • Facilidad para modificar un componente sin afectar al resto. • Portabilidad • Interoperabilidad entre distintos tipos de brokers. • Reusabilidad.	• Eficiencia restringida • Menor tolerancia frente a fallos: si falla el broker, todo el sistema se viene abajo. • Prueba y depuración costosas, pues existen muchos componentes en el sistema.

Como se puede ver en el apéndice C, existen varias arquitecturas que ofrecen soluciones para las aplicaciones distribuidas como CORBA, RMI, DCOM y MOM. CORBA y RMI implementan el patrón broker y se pueden utilizar para comunicar clientes o servidores intermedios con los servidores remotos del robot. Según una comparación cualitativa y cuantitativa entre CORBA y RMI, se ha concluido que CORBA es conveniente para aplicaciones Web de gran escala donde se necesita soporte para servidores escritos en diferentes lenguajes de programación y donde se espera alto rendimiento bajo carga pesada de clientes [Juric,00]. Además, los servidores pueden estar en cualquier sitio de Internet. RMI, por otro lado, es conveniente para aplicaciones Web de pequeña escala donde se comunican servidores escritos en Java y donde la facilidad de aprendizaje y de uso es más crítica que el rendimiento.

En el capítulo siguiente, se puede ver como se usa este patrón para comunicar los servidores del middleware con los servidores remotos del robot.

Capítulo

6

ARQUITECTURA SOFTWARE

Capítulo 6

ARQUITECTURA SOFTWARE

6.1 INTRODUCCIÓN

Una arquitectura software es una estructura de alto nivel de un sistema informático que comprende el conjunto de decisiones importantes sobre la organización del sistema como:

- La selección de los elementos estructurales y las interfaces que componen el sistema.
- El comportamiento, especificado como colaboraciones entre estos elementos.
- La composición de los elementos estructurales y de comportamientos básicos en un subsistema más amplio.
- El estilo arquitectónico que orienta esta organización [Bass,98].

Las propiedades importantes que una arquitectura de software incluye, son las siguientes:

- Tener un nivel de abstracción que permita ver el sistema en su totalidad.
- La estructura debe apoyar la funcionalidad requerida del sistema. Así el comportamiento dinámico del sistema debe tomarse en cuenta a la hora de diseñar la arquitectura.
- La arquitectura debe cumplir los requisitos no funcionales del sistema. Esto incluye el rendimiento, los requerimientos de fiabilidad y seguridad asociados con la funcionalidad actual, así como también los requerimientos de flexibilidad o extensibilidad asociados con una futura funcionalidad en un coste de cambio razonable.
- Al nivel arquitectónico, todos los detalles de implementación se ocultan [Bass,98].

El establecimiento de una base arquitectónica sólida es absolutamente esencial para el éxito de un sistema orientado a objetos. Ignorar este paso es desastroso y lleva a un software caótico [Booch,95].

En el presente capítulo, se describe la arquitectura software propuesta para desarrollar un sistema de interacción remota basado en Internet. En el siguiente apartado se presenta dicha arquitectura propuesta. Las tres capas que forman la arquitectura propuesta se discuten en detalle en los apartados 6.3, 6.4 y 6.5. Y finalmente, la interacción entre las capas de la arquitectura se presenta en el apartado 6.6.

6.2 ARQUITECTURA PROPUESTA

En el capítulo 4, se han discutido los laboratorios remotos como entornos innovadores que facilitan la interacción remota con los robots móviles. Estos laboratorios remotos están basados en arquitecturas similares y se desarrollan utilizando distintas tecnologías. Teniendo en cuenta los requisititos mencionados en el capítulo anterior se ha desarrollado una arquitectura basada en el modelo cliente servidor de tres capas (ver Apéndice C) como se muestra en la figura 6.1.

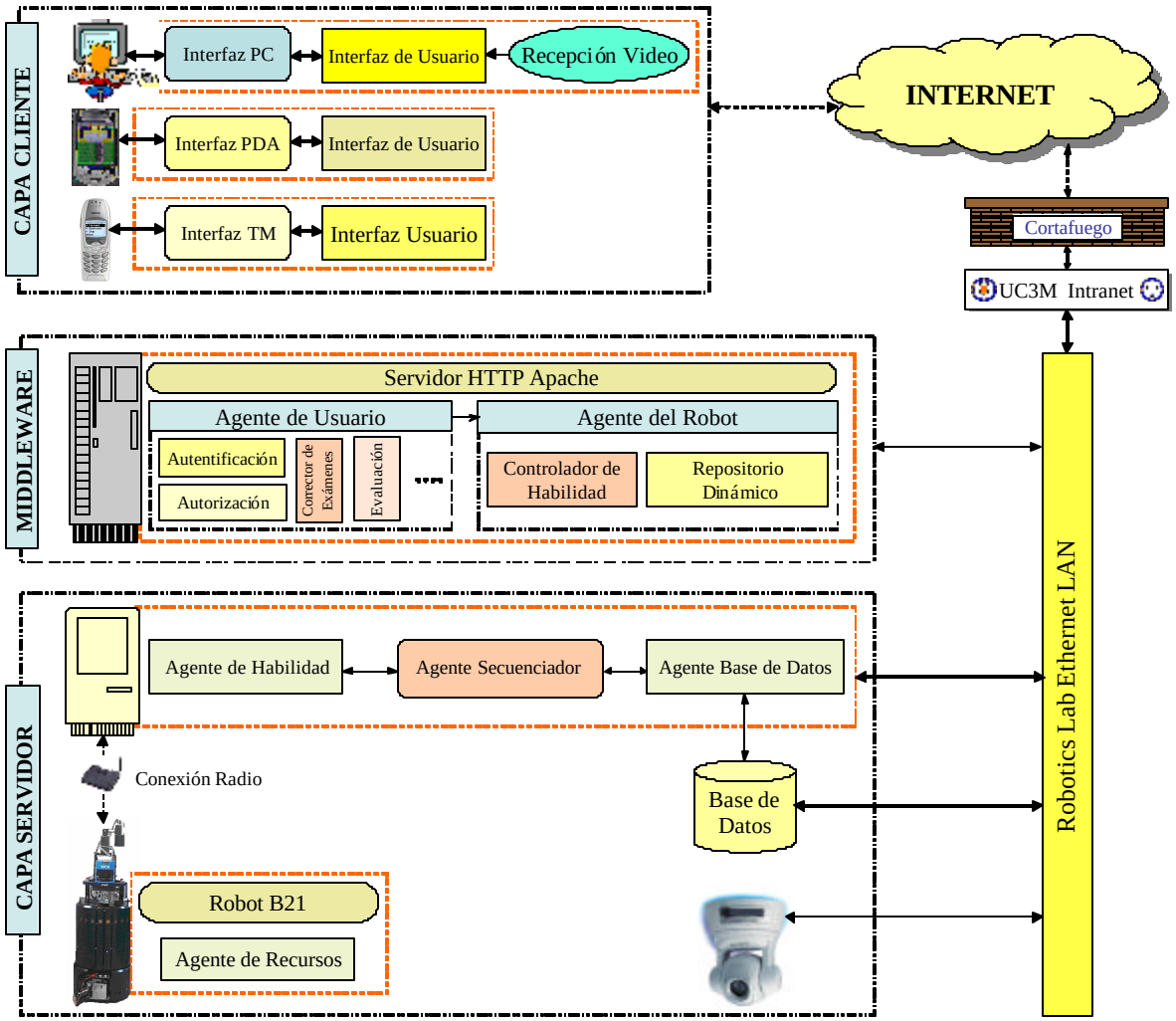


Fig. 6.1: Arquitectura del Sistema

En la arquitectura propuesta se usa el concepto de control supervisado por el cual el usuario tiene un papel de supervisión para mejorar el rendimiento del sistema y evitar los problemas de inestabilidad. Mediante el control supervisado, el usuario puede comunicarse con el sistema remoto a mayor nivel de abstracción usando comandos de alto nivel enviados al robot, el cual tiene un nivel mayor de autonomía para reducir la cantidad de datos que hay que transmitirse al sitio remoto. El nivel de autonomía del robot se incrementa encapsulando todos los comportamientos del robot en varios tipos de habilidades. Limitar la interacción remota a comandos de alto nivel ayuda a reducir el ancho de banda necesario e incrementar la autonomía del robot puede ayudar a disminuir la sensibilidad de la tarea teleoperada al retraso temporal. A continuación, se explican las tres capas que forman esta arquitectura.

6.3 CAPA DE CLIENTE

Como se muestra en la figura 6.2, la capa del cliente contiene la interfaz de usuario que juega un papel fundamental en cualquier sistema de interacción remota como ya se ha comentado en el capítulo anterior. La interfaz determina en gran medida la percepción e impresión que el usuario poseerá del sistema puesto que no está interesado en la estructura y funcionamiento interno del sistema. La tendencia actual es a desarrollar interfaces intuitivas, flexibles e independiente de la plataforma y del sistema operativo, está provocando que su diseño sea cada vez más complejo.

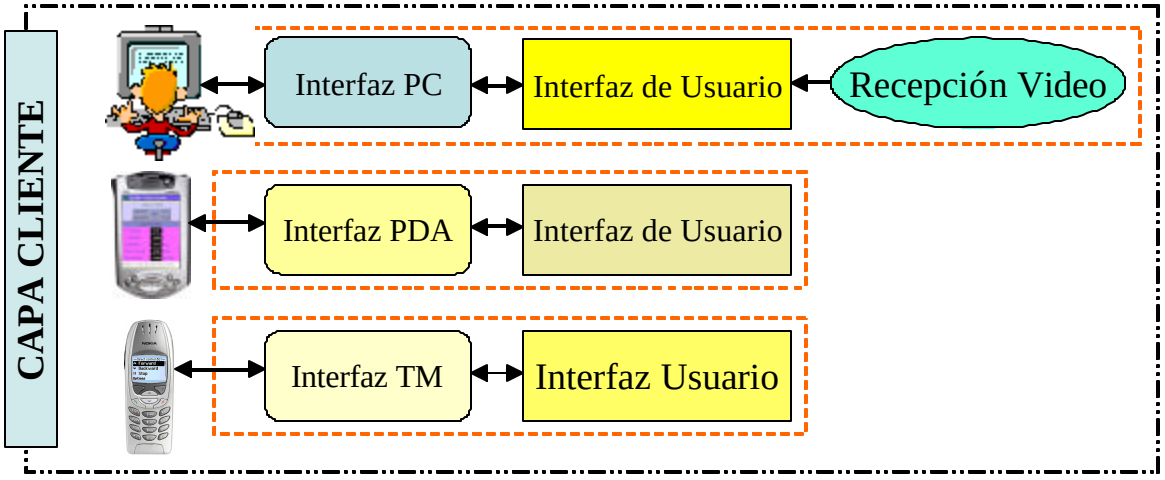


Fig. 6.2: Capa de Cliente

Como muestra la figura 6.2, en la capa del cliente se adapta el diseño de la interfaz según las características del dispositivo de interacción. Por ejemplo, en el caso de dispositivos móviles como las asistentes digitales personales (*PDA- Personal Digital Assistant*), el mismo diseño puede ser usado con algunas modificaciones necesarias para tratar con los problemas específicos relacionados con las limitaciones de los dispositivos de mano (procesadores lentos, memoria limitada y pequeñas pantallas) y de redes inalámbricas (ancho de banda limitado, alta latencia, baja estabilidad de la conexión y baja disponibilidad).

A continuación, se discute el diseño de la interfaz de usuario basada en ordenadores personales seguido por la adaptación del diseño para dispositivos móviles como las PDAs y los teléfonos móviles.

6.3.1 Ordenadores

Se han desarrollado las interfaces de usuario utilizando la arquitectura de *proxy* visual explicada en el capítulo anterior. El objetivo del patrón proxy visual es separar completamente la generación de la interfaz de usuario de los objetos de la capa de abstracción para proveer facilidades de reutilización y extensibilidad. En las subsecciones siguientes se describen los componentes de la interfaz de usuario, los módulos reutilizables desarrollados, la interacción entre los objetos de la interfaz y cómo se combinan habilidades simples en la capa de cliente para formar una habilidad compleja.

6.3.1.1 Componentes de la Interfaz

Como se ha mencionado en el capítulo anterior la arquitectura de *proxy* visual es un caso especial de la arquitectura multiagente PAC. La figura 6.3 muestra los componentes de la interfaz de usuario de un experimento.

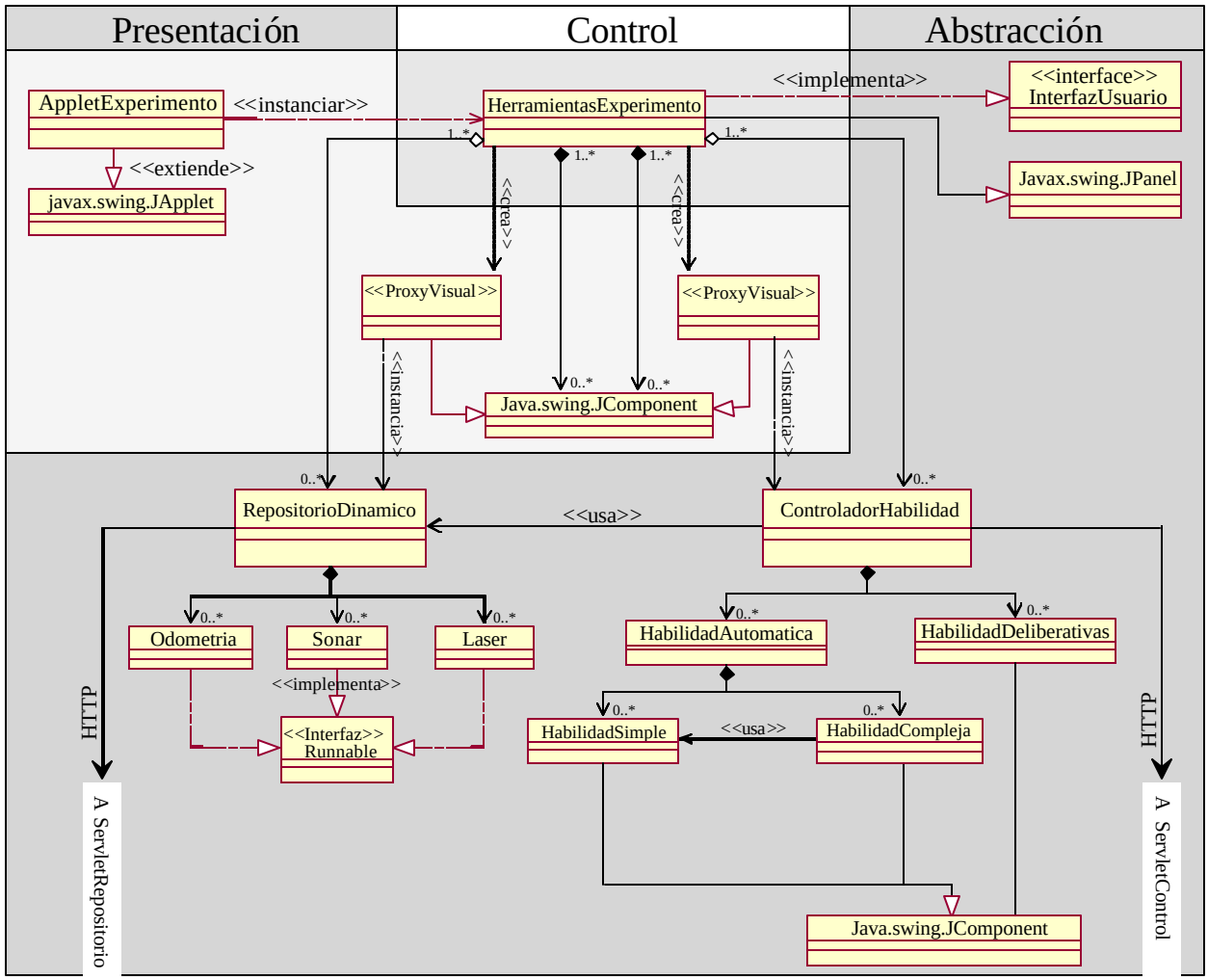


Fig. 6.3: Componentes de la Interfaz

En este diagrama, la clase *HerramientasExperimento* representa el objeto de control que implementa *InterfazUsuario* para producir proxies visuales cuando se le pida. Pide a las otras clases de la capa de abstracción (*ControladorHabilidad* y *RepositorioDinamico*) los proxies visuales como *JComponents*. Esta clase contiene un constructor e implementa todos los métodos requeridos por la interfaz pero no es una clase principal que controla todo por encima. Posiciona los proxies pedidos dentro del panel pero no hace nada más con ellos. Esta clase es simplemente un vehículo pasivo para contener proxies visuales. Es decir que el objeto de control es pasivo con respecto al flujo de mensajes. Los mensajes van directamente del proxy visual (capa de presentación) al objeto de la capa abstracta que produce el proxy. En la arquitectura de proxy visual, la encapsulación esta todavía intacta en el sentido de que la realización del objeto de la capa de abstracción puede cambiar sin tener que cambiar las otras capas. Cualquier cambio en la capa de abstracción se refleja automáticamente en la capa de presentación.

La clase *AppletExperimento* instancia *HerramientasExperimento* para formar la interfaz del experimento visto por el usuario por medio de algún navegador Web.

La clase *RepositorioDinamico* provee información a cerca del estado de los sensores y puede ser usado para adquirir remotamente datos de los sensores. Este componente esta implementado como un hilo de ejecución independiente al implementar la interfaz *Runnable* para proveer actualización de los datos sensoriales en tiempo real. Estos datos podrán ser de odometría, la cual indica la posición actual del robot y su velocidad linear y angular o datos de sensores ultrasónicos o láser.

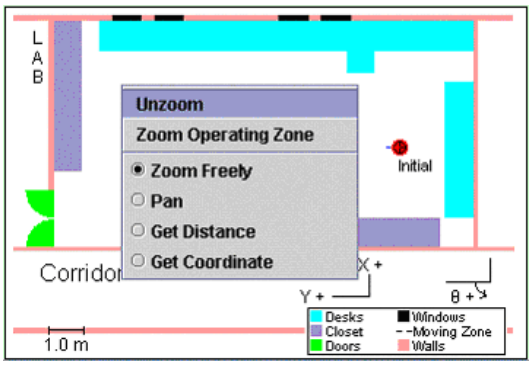
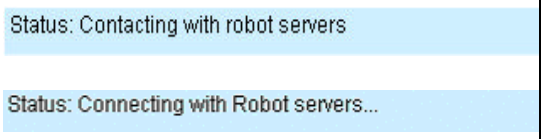
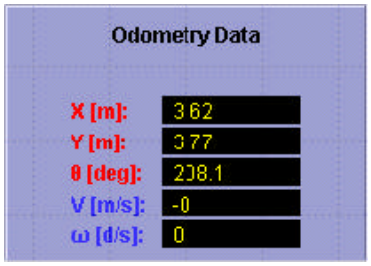
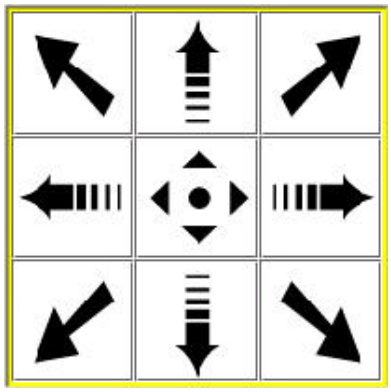
ControladorHabilidad es un componente que maneja el uso de una habilidad. Esta habilidad puede ser automática, deliberativa o combinación de varias habilidades automáticas en un panel compuesto. Este componente podrá usar la clase *RepositorioDinamico* para pedir los datos sensoriales, los cuales podrán ser necesarios para completar la tarea de control, como en el caso de la habilidad "Ir a Punto Detectando Obstáculos" usando información sensorial del sonar.

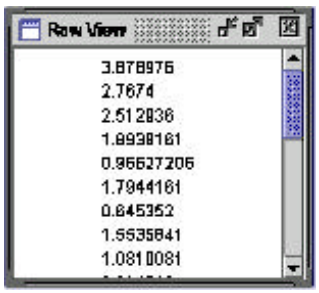
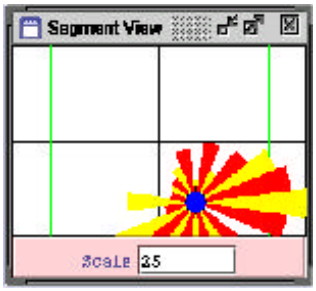
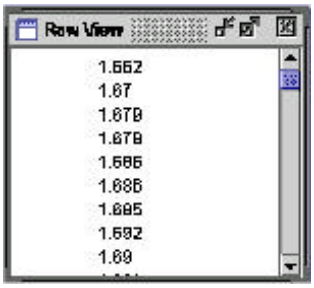
El *ControladorHabilidad* y el *RepositorioDinamico* interactúan con la capa de middleware usando el protocolo HTTP para enviar los comandos de control o las peticiones de invocar información sensorial a los servlets del controlador de la habilidad y al repositorio dinámico respectivamente y a su vez la información se transmite desde estos servlets a los servidores remotos del robot que se encuentran en la capa del servidor.

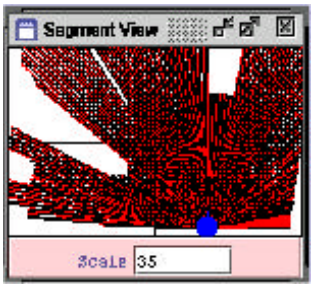
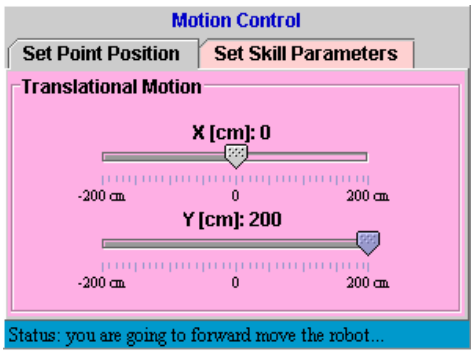
6.3.1.2 Módulos Reutilizables

El *ControladorHabilidad* se forma utilizando clases reutilizables que facilitan el desarrollo de las interfaces de usuario. Como se ha mencionado en el capítulo anterior, una de las ventajas de usar el patrón proxy visual es la separación real entre el nivel de presentación y la abstracción. La generación de la interfaz de usuario está completamente separada de los objetos de capa de abstracción proporcionando así facilidad para reutilizar o extender el código. Además cualquier cambio en el objeto de abstracción, se refleja automáticamente en el objeto de presentación (su proxy visual). Se han desarrollado varios módulos reutilizables en la capa de abstracción que proporcionan varias funciones como se muestra en la tabla 6.1.

Tabla 6.1: Módulos Reutilizables

Módulo	Interfaz	Descripción
Modelo 2D		Representa un modelo 2D del entorno de trabajo y del robot real para mostrar el movimiento del robot de forma grafica. Esta clase se adapta según la habilidad. Por ejemplo se ha añadido la posibilidad de dibujar la trayectoria en la habilidad Seguir Contorno e Ir a Punto. También se puede modificar este modelo fácilmente con el cambio de la habilitación donde se encuentra el robot.
Panel de Estado		Informa al usuario acerca del estado de la conexión con el servidor remoto.
Estado del Servidor		Informa al usuario acerca del estado del servidor remoto.
Panel de Odometría		Proporciona los datos de la odometría en tiempo real (la posición actual, la velocidad lineal y la velocidad angular)
Controlador de Movimiento		Es un controlador de movimiento que permite mover el robot en varias direcciones y que se puede integrar en varios experimentos.

Datos del Sonar en formato Vector		Proporciona los datos de los ultrasonidos en formato vector.
Datos del Sonar en formato Puntos		Proporciona los datos de los ultrasonidos en formato puntos.
Datos del Sonar en formato Segmentos		Proporciona los datos de los ultrasonidos en formato segmentos.
Datos del Láser en formato Vector		Proporciona los datos del láser en formato vector.
Datos del Láser en formato Puntos		Proporciona los datos del láser en formato puntos.

Datos del Láser en formato Segmentos		Proporciona los datos del láser en formato segmentos.
Panel de Comandos		Para activar o desactivar la habilidad. Se cambia la URL del servlet de habilidad y los parámetros que se mandan según la habilidad.
Panel de Control		Para asignar los parámetros de la habilidad. Esta clase se modifica según los parámetros que recibe cada habilidad. Por ejemplo la habilidad ir a punto recibe el punto del destino, la velocidad máxima y el error máximo. Otra habilidad como la de Seguir Contorno recibe la distancia mínima, la dirección y la velocidad máxima del robot.
Guión del Experimento		Contiene los pasos del experimento que se pueden cambiar fácilmente editando una de las clases que forman el guión.

Para desarrollar nuevas interfaces, el único que hay que hacer es generar proxies visuales de las clases reutilizables y colocarlos en el panel del experimento utilizando un método *añadir_elemento* del objeto de control *HerramientasExperimento.* Por ejemplo y como se muestra en la figura 6.4, las interfaces de las habilidades “ir a punto” y “seguir contorno”, están desarrolladas usando clases reutilizables comunes con pequeñas modificaciones para adaptar la interfaz con las necesidades de la habilidad.

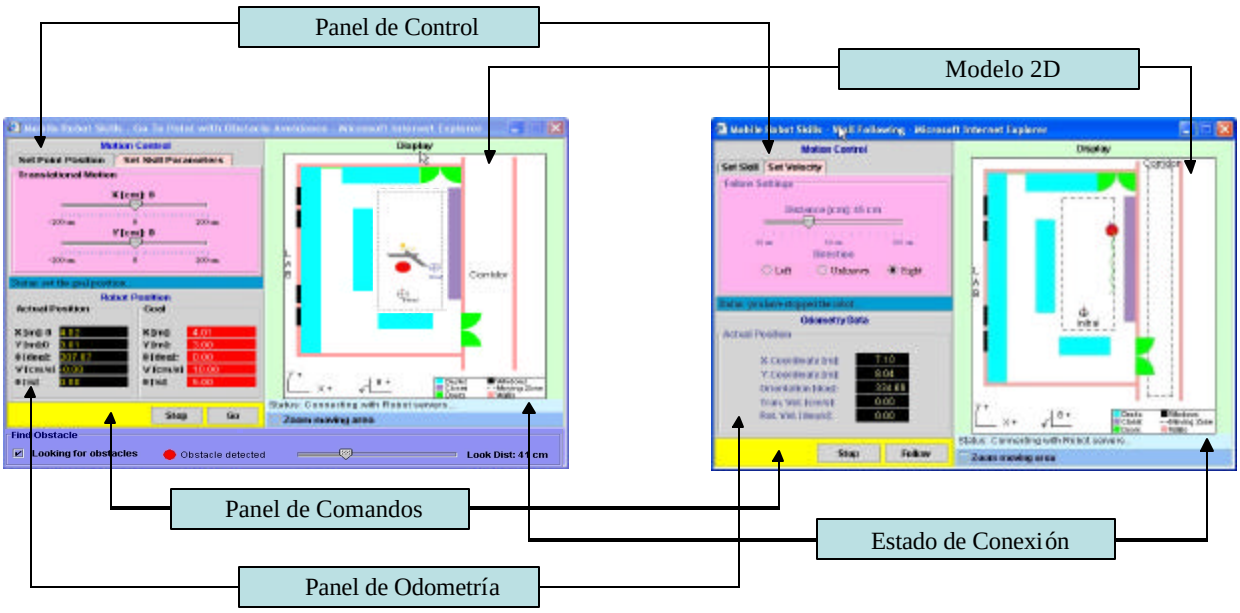


Fig. 6.4: Interfaces de las Habilidades "Ir a Punto" y "Seguir Contorno"

En la figura 6.5, se puede ver la tasa de reutilización del código para las interfaces desarrolladas en la presente tesis. Esta tasa es el porcentaje del código reutilizable con respecto al código total de la interfaz.

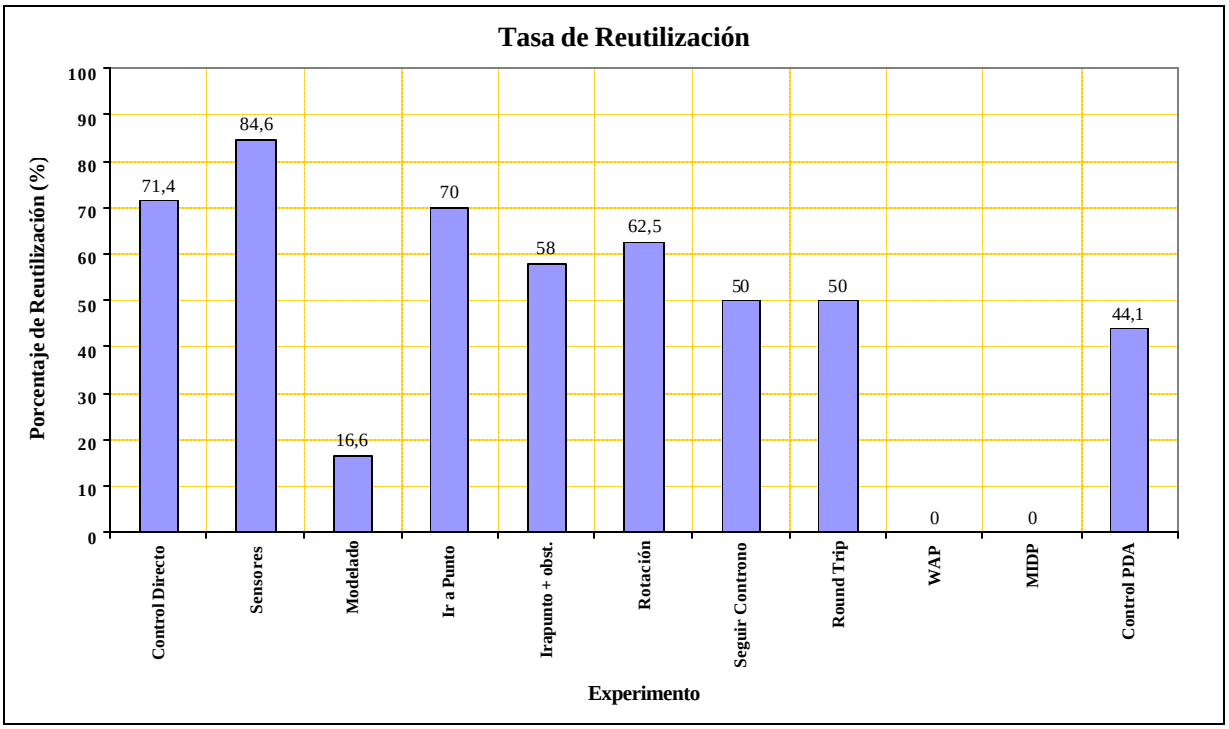


Fig. 6.5: Tasa de Reutilización

6.3.1.3 Colaboración entre los Objetos

El patrón *proxy* visual proporciona interfaces de usuario flexibles con relaciones de acoplamiento mínimas entre los subsistemas. El único acoplamiento está entre el proxy visual en la capa de presentación y el objeto real en la capa de abstracción desde el punto de vista del flujo de mensajes. El objeto de control es pasivo con respecto al flujo de mensajes. Los mensajes van directamente del proxy visual (capa de presentación) al objeto de la capa abstracta que produce el proxy. En la figura 6.6, se puede ver el diagrama de colaboración entre los objetos que forman el sistema.

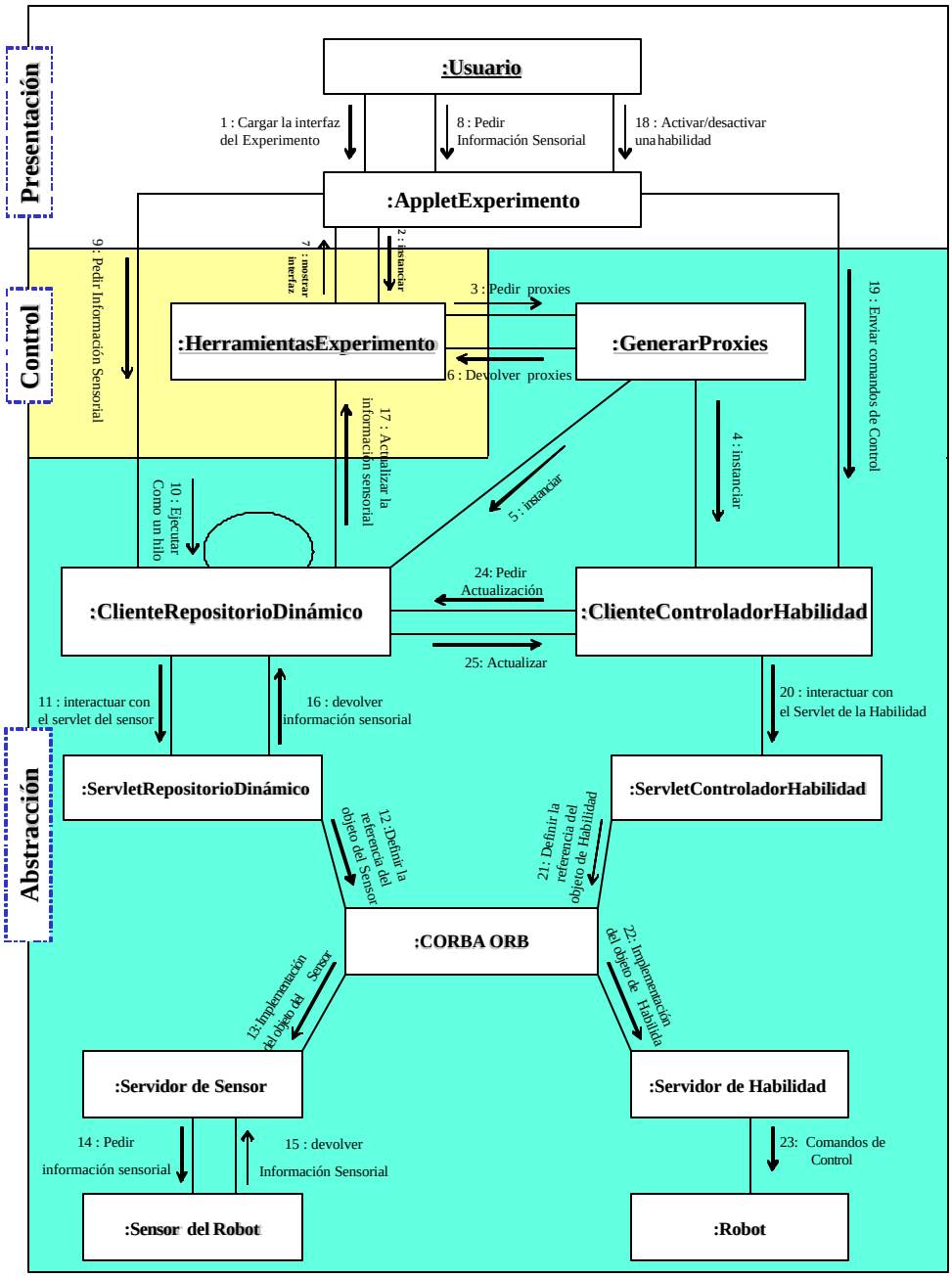


Fig. 6.6: Diagrama de Colaboración

Este diagrama presenta una alternativa al diagrama de secuencia para modelar interacciones entre objetos en el sistema. Mientras que el diagrama de secuencia se centra en la secuencia cronológica del escenario que se modela, el diagrama de colaboración muestra las interacciones entre objetos organizadas entorno a los objetos y los enlaces entre ellos. Los objetos se conectan por medio de enlaces, cada enlace representa una instancia de una asociación entre las clases implicadas. En el aparatado 6.6 se explica en detalle la interacción entre las tres capas.

6.3.1.4 Combinar Habilidades Simples

Utilizando los módulos reutilizables, se pueden combinar las habilidades simples para formar una habilidad compleja como en el caso de habilidad sensorimotora "Ir a Punto" con la habilidad perceptiva "Detectar Obstáculos". Esta combinación se suele llevar a cabo utilizando un secuenciador que se encarga de secuenciar el funcionamiento de las habilidades para evitar actuaciones incompatibles. El secuenciador es responsable de decidir que habilidades deben ser activadas en cada momento para evitar activar al mismo tiempo las habilidades que actúan sobre los mismos actuadores. En la sección 6.5.2 se discute en más detalle el secuenciador de las habilidades.

En la capa del cliente se puede también secuenciar dos o más habilidades para obtener el funcionamiento de una habilidad compleja aunque es más recomendable llevar a acabo la secuencia utilizando un secuenciador en la capa del servidor debido a que el retraso introducido por Internet limita la comunicación en tiempo real entre las distintas habilidades.

Se ha desarrollado una habilidad compleja llamada "Ir a Punto Detectando Obstáculos" (véase la figura 6.7), que representa una secuencia de las dos habilidades "Ir a Punto" y "Detectar Obstáculos" en la capa de cliente.

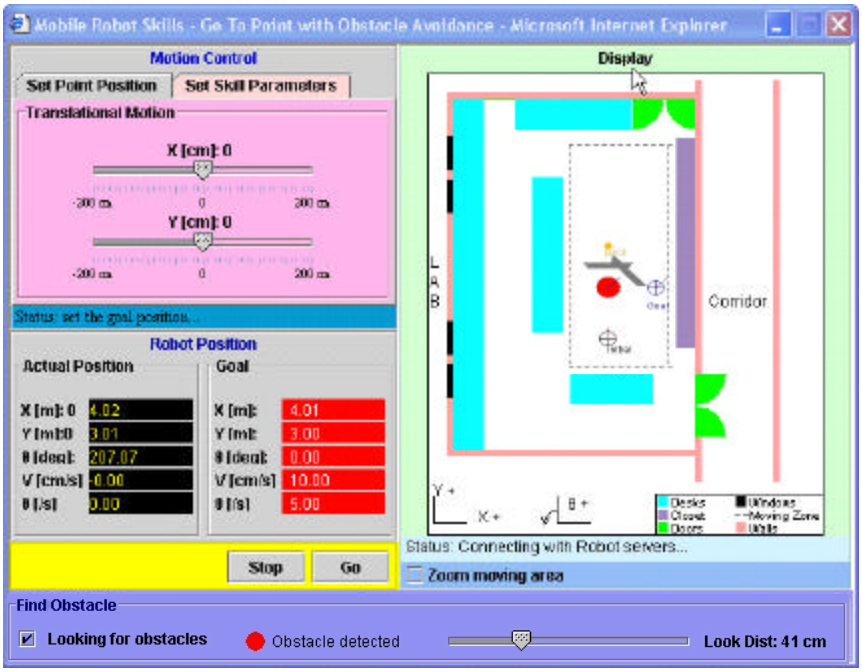


Fig. 6.7: Habilidad Ir a Punto Detectando Obstáculos

En el capítulo 7 se explica en detalle el desarrollo de la interfaz de esta habilidad.

6.3.2 PDA

Se ha estudiado el uso de las PDAs como iPAQ de Compaq 3870 PDA como un dispositivo de interacción con el robot B21 (ver Apéndice A).

Desde el punto de vista hardware y como se puede ver en la figura 6.8, se han incorporado los dos componentes siguientes en la capa de cliente para permitir el uso de la PDA como dispositivo de interacción.

- Una tarjeta Compaq WL 110 Wireless Cliente para la comunicación inalámbrica entre la PDA y el punto de acceso.
- Un punto de acceso LINKSYS 2.4GHz, 11bps con rango máximo hasta 350 metros en entornos exteriores.

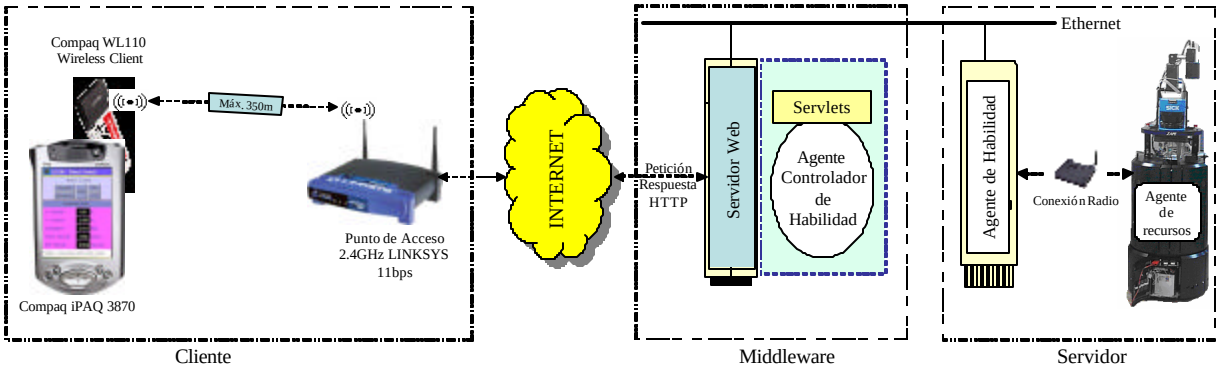


Fig. 6.8: Arquitectura de Interfaz PDA

Desde el punto de vista software, se ha tenido que modificar la capa del cliente para adaptar la interfaz con las limitaciones de la PDA (procesadores lentos, memoria limitada y pequeñas pantallas) y de redes inalámbricas (ancho de banda limitado, alta latencia, baja estabilidad de la conexión y baja disponibilidad predecible). La capa de middleware y la capa de servidor no han tenido ningún cambio.

Además para utilizar la interfaz, hay que instalar Jeode *Runtime que* es una implementación certificada de la especificación PersonalJava 1.2 de Sun [Insignia,03]. Se usa Jeode *Runtime* como conexión del Pocket Internet Explorer del iPAQ, para ejecutar subprogramas de Java desde una página Web, o como máquina virtual de Java independiente, para ejecutar aplicaciones de Java en iPAQ (ver Apéndice A).

Swing 1.1.1 no es compatible con PersonalJava 1.2; por lo tanto, Jeode *Runtime* no admitirá Swing 1.1.1. Por está razón, había que utilizar Java AWT en vez de Java Swing para el desarrollo de la interfaz.

6.3.3 Teléfonos Móviles

Las dos tecnologías principales para el desarrollo de aplicaciones inalámbricas son la tecnología WAP y el uso de la edición micro de java 2, J2ME. Para desarrollar interfaces accesibles vía teléfonos móviles, se ha cambiado totalmente la capa del cliente debido a los requisitos de las tecnologías de desarrollo (ver Apéndice A). No ha habido ningún cambio en la manera de programar los servlets de la capa de middleware. En cuanto a la capa del servidor, se han utilizando los servidores de las habilidades tal como en el caso de las interfaces PC y las de PDAs.

Se ha usado Java como herramienta principal para el desarrollo de interfaces de las habilidades de movimiento del robot B21. También, se ha desarrollado interfaz para la habilidad motora "control directo" utilizando la tecnología WAP. Existen muchos factores a favor de usar la tecnología Java para desarrollar aplicaciones inalámbricas (ver Apéndice A). Entre ellos cabe citar la compatibilidad de plataforma, la elección dinámica de aplicaciones y servicios, la seguridad y la disponibilidad de documentación y soporte técnico. La figura 6.9 resume los pasos de desarrollo de las interfaces.

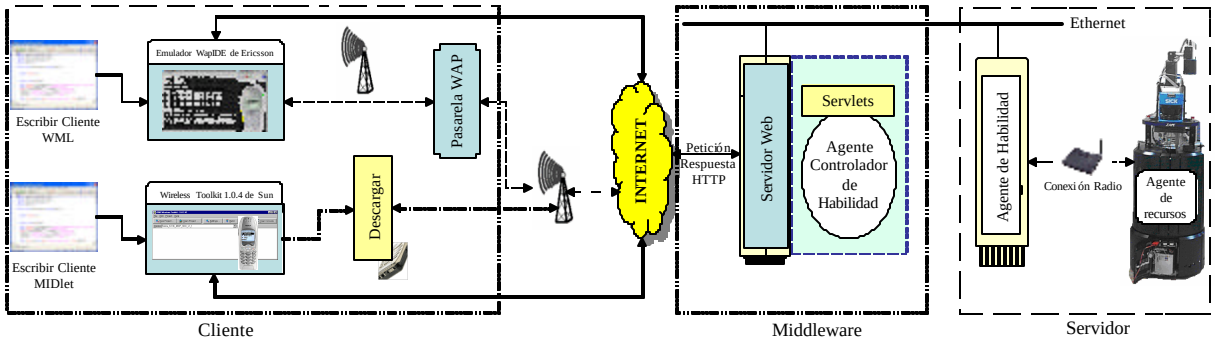


Fig. 6.9: Arquitectura de Interfaces de Teléfonos Móviles

Como se puede ver en dicha figura, se escribe el código de la interfaz en WML o como MIDlet utilizando un editor como por ejemplo EditPlus 2. Una vez escrito el código es necesario compilarlo. Aquí ya existen diferencias entre ambas interfaces. Mientras que para el desarrollo de la interfaz J2ME se utiliza *Wireless Toolkit 1.0.4* de la edición micro de Java 2 [Sun,02], para la interfaz WAP se ha utilizado la herramienta WapIDE de Ericsson [Ericsson,02] como entorno integrado de desarrollo (ver Apéndice A). Dichas herramientas son validas para depurar el código e incluso para emular el funcionamiento del móvil, de hecho, las pruebas realizadas en la presente tesis han sido desde estos emuladores.

En el capítulo siguiente se pueden ver en más detalle las interfaces implementadas para las habilidades de movimiento del robot B21 utilizando WAP y MIDP.

6.3.4 Recepción de video

Se ha integrado una camera comercial para proporcionar realimentación visual desde el sitio remoto del robot hasta el sitio local del usuario. Se trata de una cámara compacta con un servidor Web integrado y con funciones de ampliación, inclinación y giro (PTZ) por control remoto incorporadas (ver apéndice C). La cámara Sony SNC-RZ30P es la primera cámara IP capaz de grabar vídeo en movimiento (hasta 25 imágenes por segundo) con imágenes en color de alta calidad y resolución VGA (640x480) [Sony,03]. Se han configurado tres cuentas con distintos privilegios de control como se muestra en la tabla 6.2.

Tabla 6.2: Tipos de Acceso a la Cámara

Cuenta	Permiso de ver	Permiso de controlar	Permiso de administrar
Operador	✓		
Supervisor	✓	✓	
Administrador	✓	✓	✓

En la figura 6.10, se puede ver la realimentación visual proporcionada por la cámara para un operador. La velocidad de transmisión de la imagen se adapta automáticamente según el ancho de banda disponible.

Fig. 6.10: Realimentación Visual

6.4 CAPA DE MIDDLEWARE

La capa middleware se considera como una infraestructura lógica que proporciona los mecanismos básicos de direccionamiento y transporte entre la capa del cliente y la del servidor (ver Apéndice C). Las funciones principales llevadas a cabo por la capa middleware son las siguientes:

- Servir como módulo intermedio que comunica los objetos de la capa del cliente con los objetos remotos de la capa del servidor. Los servlets de esta capa se encargan de recibir las peticiones del cliente, localizar el servidor remoto y mandar las peticiones al servidor y luego las respuestas desde el servidor al cliente.
- En algunos casos como los servlets del agente de usuario, los servlets sirven como servidores que procesan directamente las peticiones del cliente sin redireccionarlas a otro servidor.
- Controlar las comunicaciones entre la capa del cliente y la capa del servidor según la configuración del servidor Web que aloja los servlets de esta capa. El Apache arranca la máquina virtual de Java donde se encuentra el motor de servlets que permitirá la comunicación de forma transparente de los objetos de la capa del cliente con los servlets gracias al modulo *mod_jserv* del Apache [Apache,03]. También permite restringir las direcciones IP que tienen acceso al sistema.

En esta capa, hay un PC conectado internamente con una LAN Ethernet y externamente con la Intranet de la universidad, ejecutando un servidor Web apache, en el que se hospedan dos agentes como se muestra en la figura 6.11.

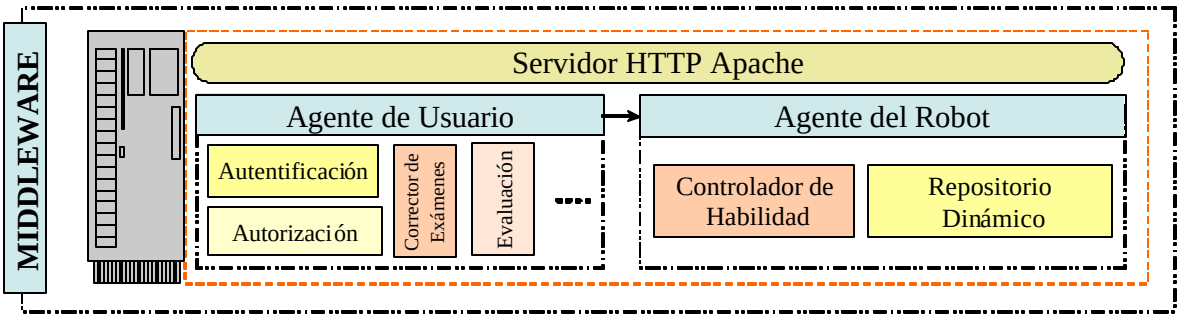


Fig. 6.11: Capa de Middleware

En las siguientes subsecciones, se explican cada agente con más detalles.

6.4.1 Agente de Usuario

El agente de usuario ofrece varios servicios como el manejo de acceso de los usuarios al sistema y los servicios necesarios para el entorno de teleeducacción. Este agente con la capa de cliente forman un modelo cliente-servidor de dos capas. A continuación, se explican los servicios ofrecidos por los servlets de este agente.

6.4.1.1 Autentificación

Se pueden distinguir entre tres tipos de usuarios según la autoridad que les da el sistema. Los usuarios normales no pueden acceder al laboratorio remoto pero si pueden utilizar libremente los módulos de teleeducacción del sistema como las clases en línea, los exámenes, los materiales descargables, etc. Los usuarios autentificados pueden acceder al laboratorio remoto después de haber introducido correctamente un nombre de usuario y una contraseña que pide el sistema. Este requerimiento se llama *autentificación*. Este protocolo de autentificación es conocido como ap 3.0 [Kurose,01]. La autentificación de usuario se lleva a cabo mediante el uso de un applet de login (véase la figura 6.12) y un servlet denominado LoginServlet.

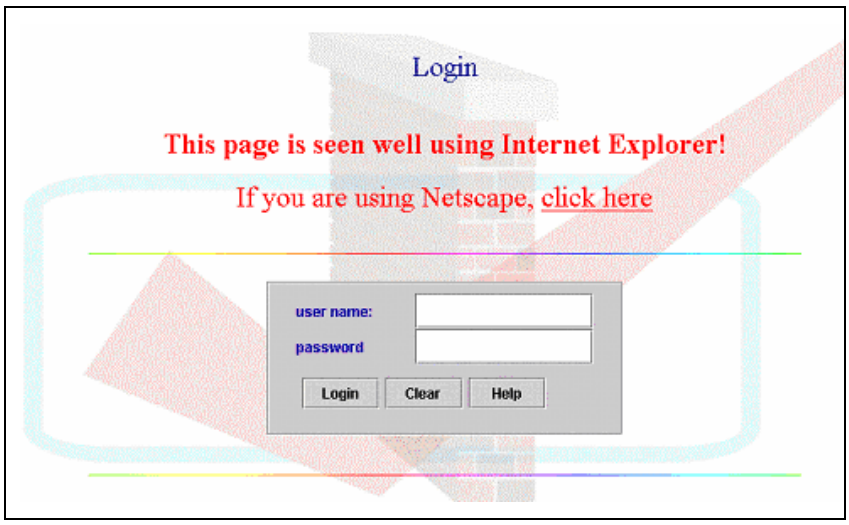


Fig. 6.12: Applet de *Login*

En caso de introducir los datos correctamente, el servlet le permite al usuario comenzar acceder al laboratorio. Si no, el servlet exige al usuario que se registre usando una página de registro. Los usuarios autentificados pueden observar el funcionamiento del sistema y pueden acceder al repositorio dinámico para adquirir remotamente la información sensorial en tiempo real. Estos usuarios no pueden controlar el robot directamente pero la petición tiene que pasarse por un agente de autorización que maneja el acceso al controlador de habilidades.

El tercer tipo de usuarios, son los usuarios autorizados que tiene derecho a controlar el robot remotamente. En el sistema actual, el servidor remoto de movimiento (actuador virtual) es el encargado de gestionar los comandos de control para impedir inconsistencias en los datos. Debido a que el uso del laboratorio remoto desarrollado en la presente tesis no es público, se supone que accede al sistema sólo un usuario que es el tutor que usa el sistema como una herramienta educativa. En el capítulo siguiente de la presente tesis, se puede ver, como se ha utilizado el laboratorio remoto en una asignatura de vehículos autónomos inteligentes. En la subsección siguiente se presenta una propuesta para manejar el acceso de múltiples usuarios mediante el servlet de autorización.

6.4.1.2 Autorización

La autorización es el proceso de dar permiso a alguien para hacer o tener algo. En los sistemas de múltiples usuarios, hay que definir al sistema quiénes pueden acceder a los servicios del sistema. En este tipo de sistemas, se puede manejar el acceso usando la metodología *Round Robin*. Como se muestra en la figura 6.13, el servlet de autenticación de usuario forma una lista de los usuarios autentificados utilizando sus direcciones IP.

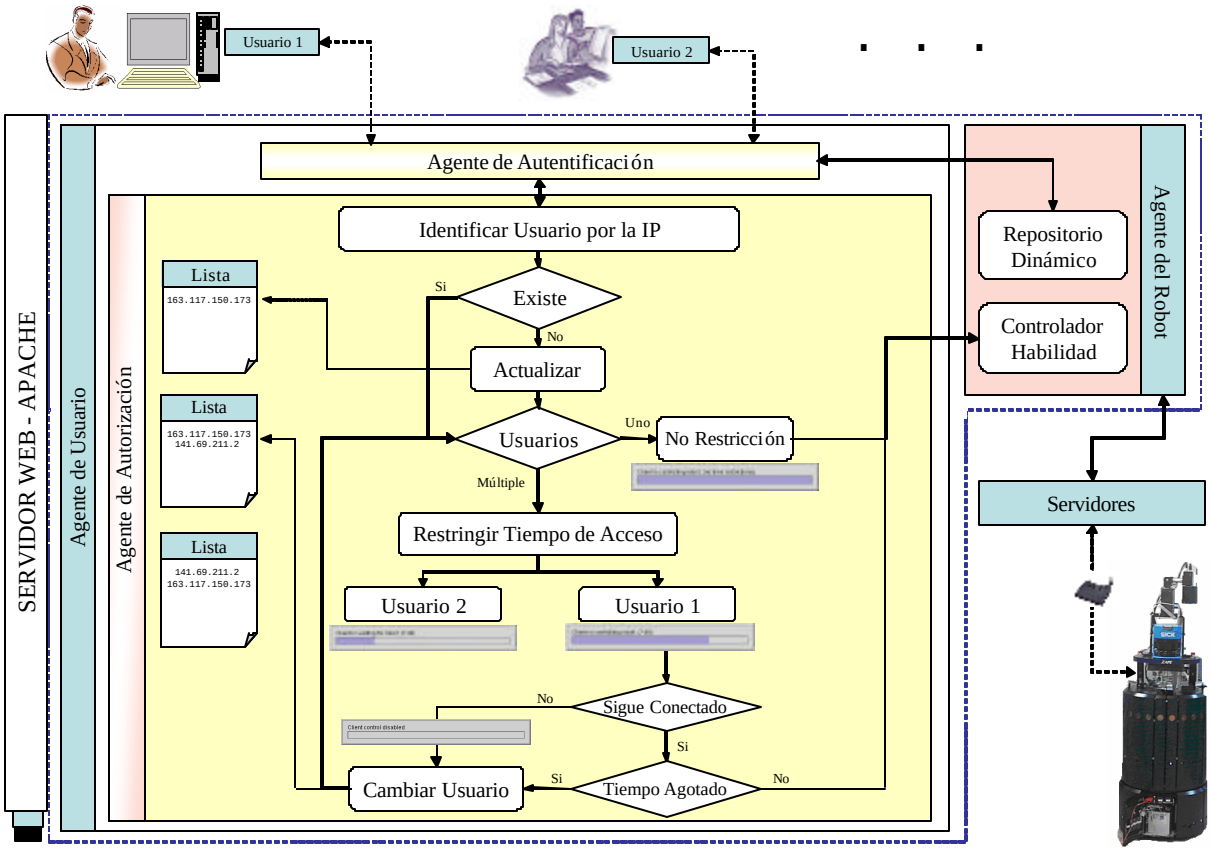


Fig. 6.13: Control de Acceso

Cuando sólo haya un usuario conectado, no se da ninguna restricción en el uso. Sin embargo, si se conectan varios usuarios simultáneamente, sólo uno tendrá el acceso durante un determinado periodo de tiempo. Una vez que el tiempo ha expirado, el próximo usuario en línea tendrá la oportunidad de controlar el sistema y el primer usuario se moverá al final de la lista de espera.

6.4.2.3 Corrector de Exámenes

Se han implementado varios servlets que corrigen y manejan automáticamente los exámenes y los tests para reducir el tiempo requerido para llevar a cabo estas tareas de evaluación. Como se muestra en la figura 6.14, el servlet recibe el examen contestado por el usuario desde un formulario html, lo corrige y luego genera una página html con las contestaciones correctas y la nota.

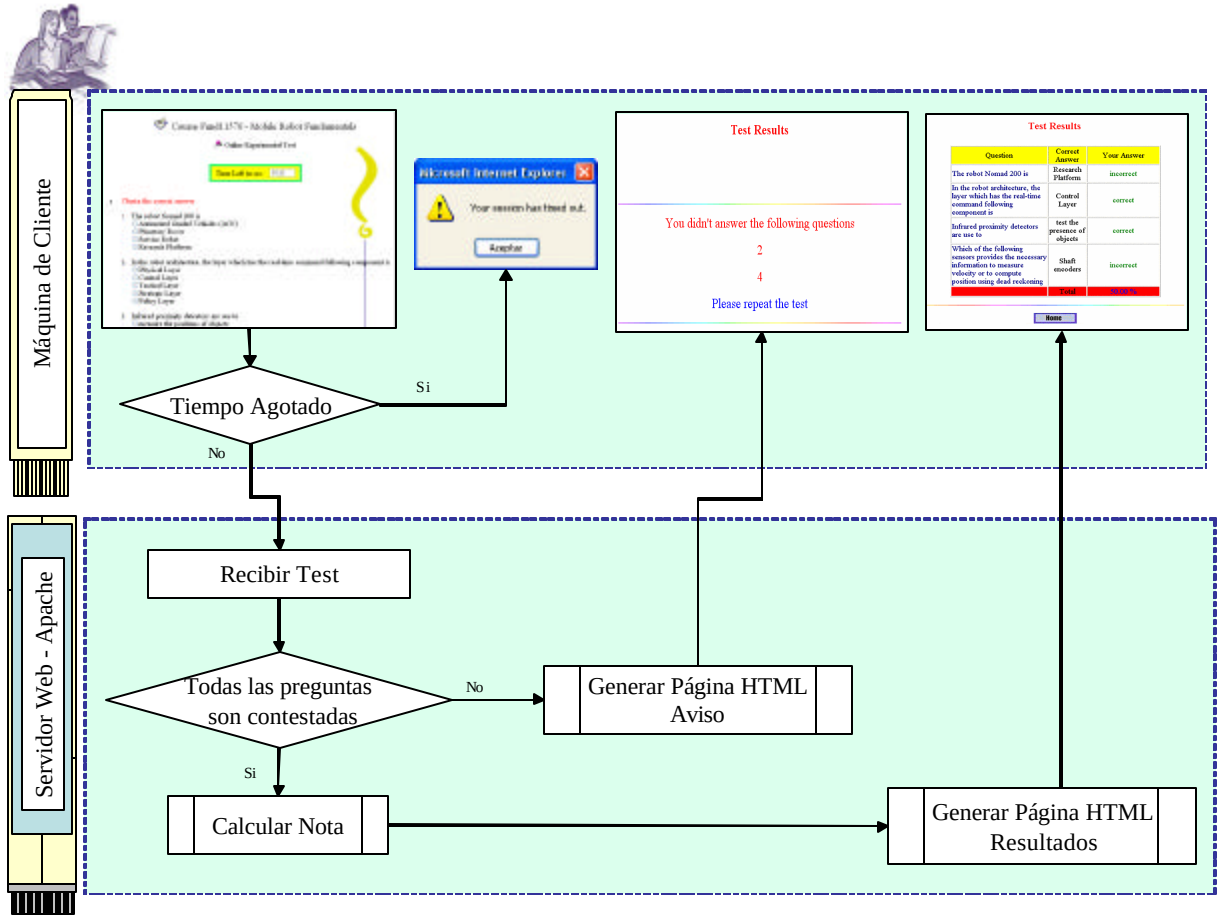


Fig. 6.14: Corrector de Exámenes

6.4.1.4 Otros Servlets

En este agente, se pueden encontrar también otros servlets como el servlet de evaluación del sistema que captura las opiniones de los usuario sobre el sistema con una encuesta en línea, el servlet de contador que cuenta el número de usuarios que acceden al sistema, el servlet de chat que proporciona el chat como un mecanismo de telecolaboración entre los usuarios, etc.

6.4.2 Agente del Robot

El agente del robot trata con el robot real, durante su funcionamiento. Este agente provee la información requerida para procesar cierta habilidad seleccionada por el usuario o requerida en un paso de un experimento. Contiene dos grupos de servlets Java, los servlets de los controladores de las habilidades y los servlets del repositorio dinámico. Un servlet de controlador de habilidad recibe peticiones del usuario (activar/desactivar la habilidad), localiza el servidor remoto de la habilidad correspondiente en el agente de habilidad de la capa de servidor utilizando CORBA y manda las peticiones a dicho servidor para activar o desactivar la habilidad. Los servlets del repositorio dinámico facilitan la adquisición remota de la información sensorial conectando el cliente con los servidores de los sensores en el agente de recursos de la capa de servidor.

CORBA se usa como arquitectura de comunicación entre los objetos distribuidos de la capa de middleware (los servlets en Java) y los objetos de la capa de servidor (los servidores en C++) (ver Apéndice C).

6.4.2.1 Repositorio Dinámico

En el repositorio dinámico se encuentran los servlets que comunican el cliente con el servidor remoto del sensor para adquirir remotamente la información sensorial. Se han desarrollado tres servlets para invocar información de la odometría, el sonar y el láser. Los servidores remotos de estos sensores están basados en utilizar el *framework* del robot "Mobility" (ver Apéndice D). En la subsección 6.5.4 se explican los servidores virtuales en detalle. En esta sección, se describen los tres servlets del repositorio dinámico.

- **Odometría**

El servlet de la odometría (*OdometryServlet*) tiene como misión hacer las peticiones necesarias al servidor de la base para la lectura de la posición del robot y de su velocidad lineal y angular. Como se muestra en la figura 6.15, para el correcto funcionamiento se sincronizan el servlet de la odometría con el hilo de la odometría. El hilo envía una petición al servlet para indicarle que quiere leer la odometría y se queda esperando respuestas hasta que éste le responde con los valores leídos.

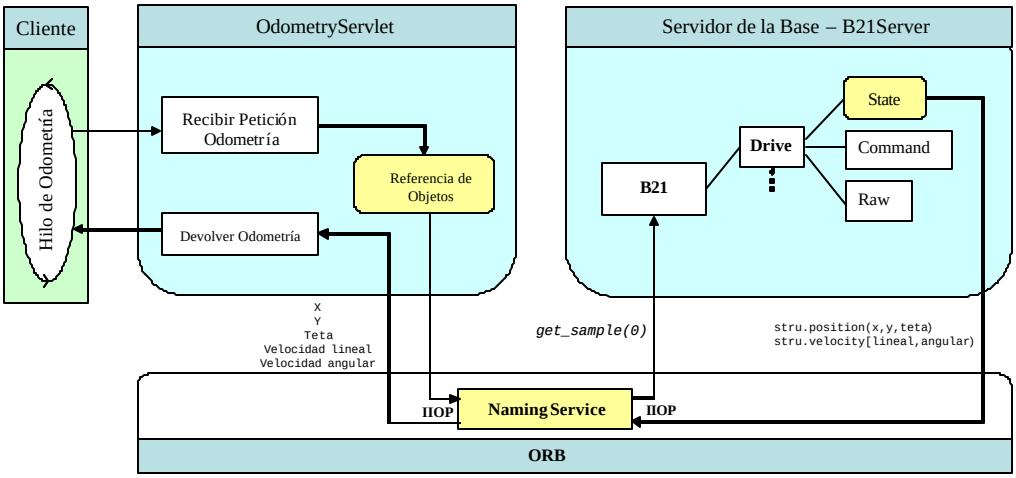


Fig. 6.15: Odometría

El objeto de Mobility utilizado para leer la posición y la velocidad es de tipo "MobilityActuator.ActuatorData" y su ruta es "*Drive/State*" dentro del objeto B21.

El método usado para leer la posición y la velocidad es get_sample(). Una vez obtenida la referencia de tipo MobilityActuator.ActuatorData se leen los distintos campos:

- *position[0]*, *position[1]* y el *position[2]*, para los datos estáticos, o lo que es lo mismo, las coordenadas x, y y theta respectivamente del robot B21.
- *velocity[0]*, *velocity[2]*, para los datos de dinámica como son velocidad de translación y velocidad de rotación.

- **Ultrasonidos**

El *SonarServlet* es el servlet encargado de hacer la petición al servidor de la base, por medio del ORB de CORBA, de la lectura del sensor de ultrasonidos. Para invocar la información del sonar, hay que acceder al objeto de Mobility que proporciona la lectura del Sonar, cuya ruta es *B21/Sonar/Raw* (este objeto es de tipo *MobilityData.FvectorData*), ver la figura 6.16.

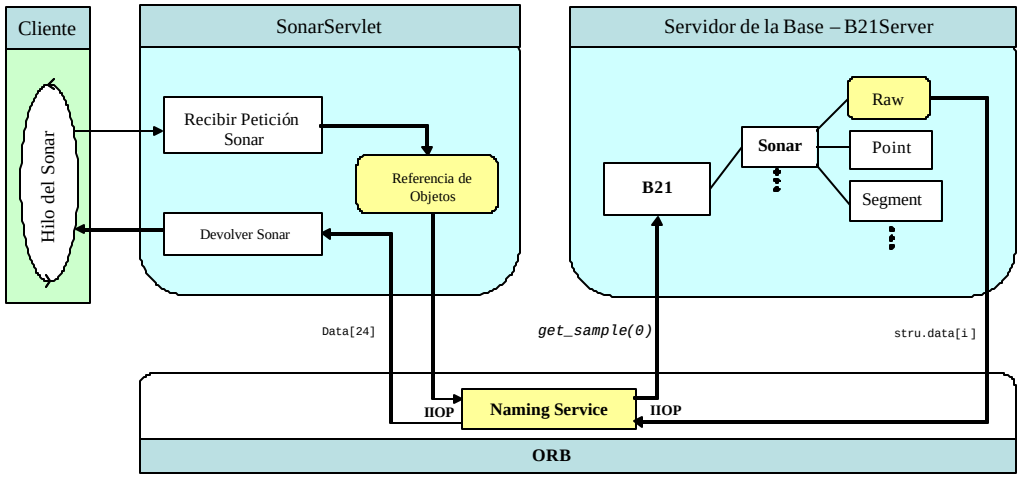


Fig. 6.16: Sonar

Una vez que el servlet conoce como puede acceder al servidor CORBA ya puede pedir una lectura y enviársela al applet. Para ello a la referencia del objeto se le aplica el método *get_sample()*. Los datos del sonar (24 lecturas de los 24 sensores) se almacenan en una estructura definida acorde con el tipo de datos que va a contener.

- **Láser**

El servlet del láser (*LaserServlet*) proporciona comunicación entre el cliente del sensor y su servidor para adquirir remotamente la información del láser. El diagrama de flujo de este servlet es análogo al de OdometryServlet y al de SonarServlet (ver la figura 6.17).

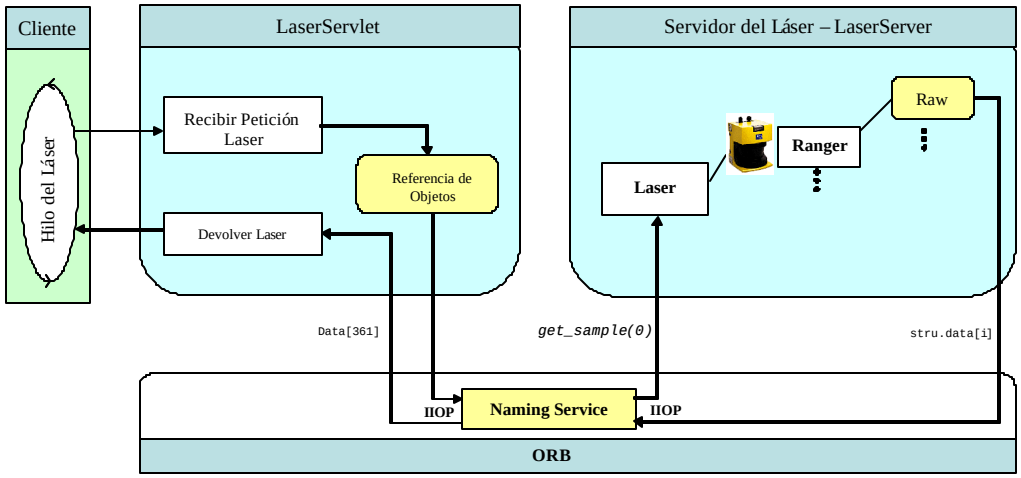


Fig. 6.17: Láser

La única diferencia está en que ahora, en vez de leer datos de odometría o del sonar, se leen lecturas del láser de SICK que el robot lleva a bordo. Los datos del láser (361 lecturas, una lectura cada medio grado de 180º que escanea el sensor) se almacenan en una estructura de datos y se devuelven al cliente.

6.4.2.2 Controlador de la Habilidad

Se ha desarrollado un servlet para cada habilidad para enviar los comandos de control al servidor correspondiente del agente de habilidad en la capa servidor. Los servicios que proporciona el servidor de una habilidad como objeto distribuido son dados por la interfaz de la habilidad implementada a través del IDL. Es decir, que las interfaces IDL funcionan como proveedores de servicio para comunicar el cliente de una habilidad (en este caso es el servlet) con su servidor remoto.

Para establecer comunicación entre el cliente de una habilidad con su servidor, hay que seguir los siguientes pasos mostrados en la figura 6.18.

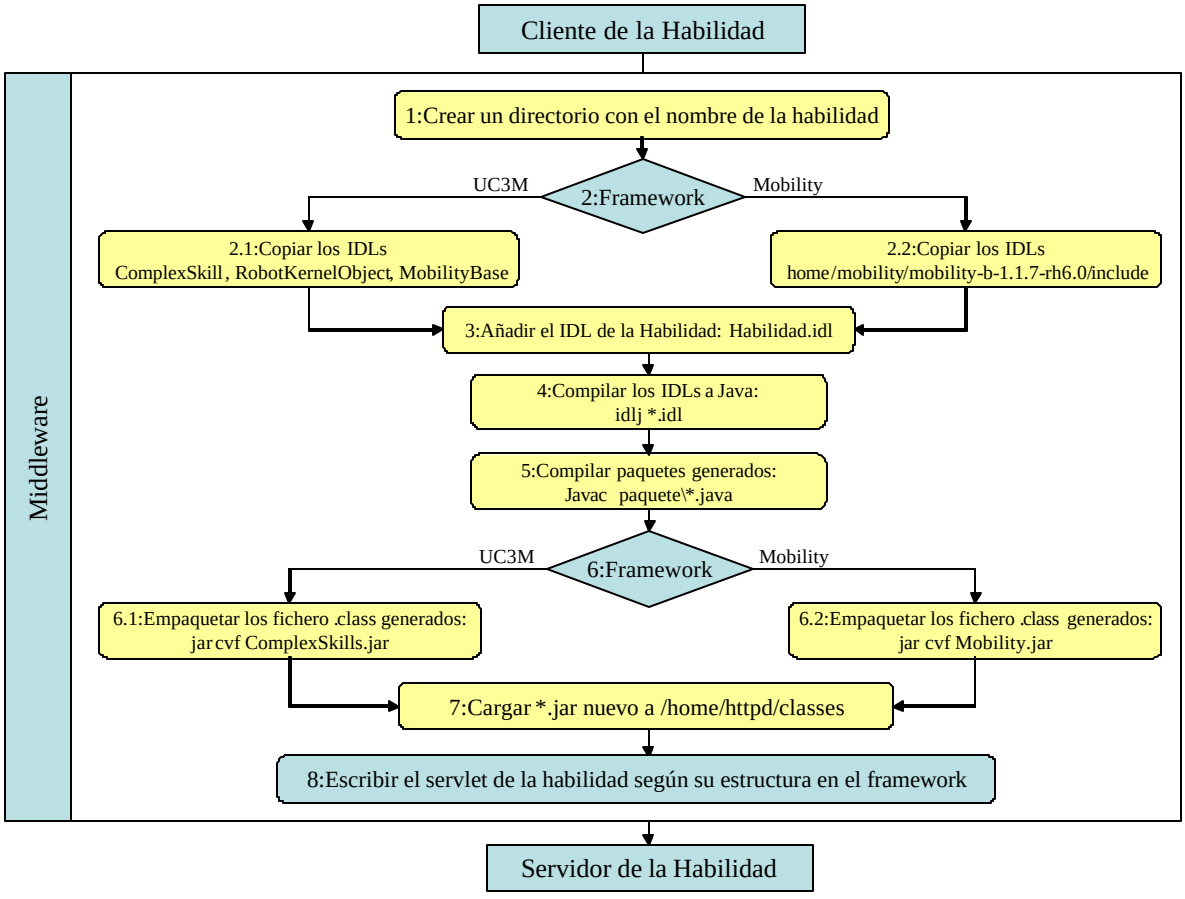


Fig. 6.18: Pasos para comunicar el cliente con el servidor

1: Crear un directorio con el nombre de la habilidad que se quiere desarrollar interfaz para ella.

2: Copiar todos ficheros *idl* del sistema en el directorio de la nueva habilidad.

2.1: En caso del framework desarrollado (sección 6.5.2) en el departamento para desarrollar habilidades complejas, se copian los ficheros siguientes que son interfaces *idl* de los paquetes del framework:

- ***MobilityBase***: Definiciones de secuencia para tipos y estructuras usadas en todas partes del entorno Mobility.
- ***ComplexSkill***: Este módulo está conformado por todas las interfaces de los componentes que permiten la creación y ejecución de habilidades complejas.
- ***RobotKernelObject***: Este módulo está conformado por todas las interfaces fundamentales para la generación de habilidades básicas y complejas.

2.2: En caso del *framework* del robot Mobility (ver Apéndice D) se copian las interfaces siguientes:

- ***MobilityCore***: Este módulo define todas las estructuras e interfaces usadas para la localización de objeto, el nombramiento, y la gestión de configuración y propiedad en el sistema de Mobility.
- ***MobilityBase***: Definiciones de secuencia para tipos y estructuras usadas en todas partes del entorno Mobility.
- ***MobilityComponents***: Este módulo define las interfaces de componentes y módulos de Mobility.
- ***MobilityExternalization***: Este módulo se base en varios estándares de CORBA y Java Externalization.
- ***MobilityActuator***: Definiciones para los objetos de fuentes de datos dinámicos.
- ***MobilityGeometry***: Definiciones para los objetos de fuentes de datos dinámicos.
- ***MobilityData***: Definiciones para los objetos de fuentes de datos dinámicos.
- ***MobilityImage***: Definiciones para los objetos de fuentes de datos dinámicos.
- ***OtraHabilidad***: interfaz de otra habilidad.

3: Añadir el fichero *idl* de la nueva habilidad *Habilidad.idl*

4: Compilar todos los ficheros *idl* con el compilador *idlj* empezando por las interfaces del framework y luego la interfaz de la nueva habilidad. Como se ha mencionado, el compilador *idlj* usa el mapeo IDL -> Java para convertir las definiciones de interfaces escritas en IDL a las interfaces, clases y métodos correspondientes en Java (ver Apéndice C). Este compilador genera un subdirectorio para cada interfaz con ficheros en java. La figura 6.19, muestra los ficheros generados al compilar la interfaz de la habilidad *Habilidad.idl*.

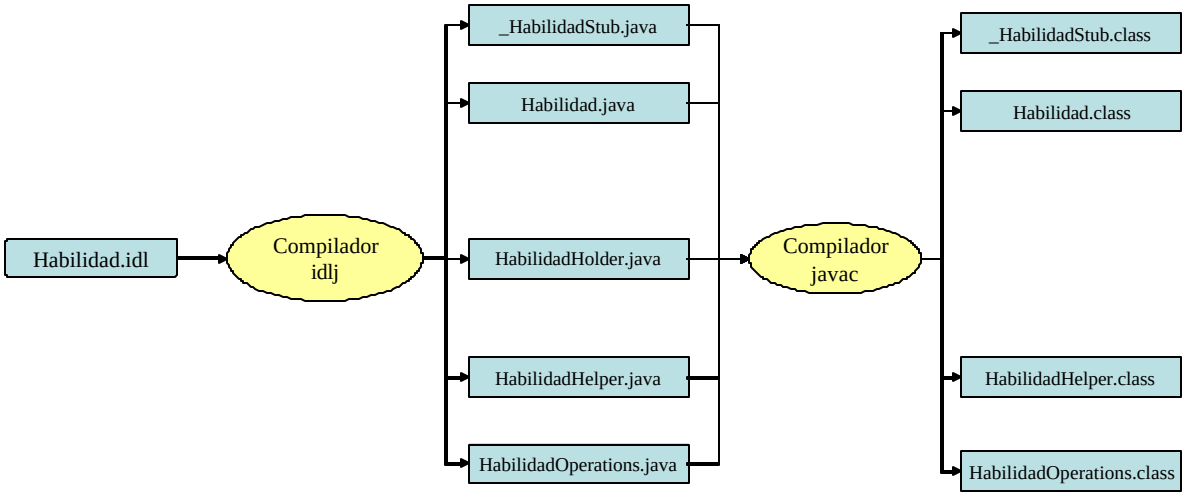


Fig. 6.19: Compilación de *Habilidad.idl*

- ***_HabilidadStub.java***: Esta clase es el *stub* del cliente que proporciona la funcionalidad de CORBA al cliente. Implementa la interfaz *Habilidad.java.* Un stub es un proxy entre el cliente de la habilidad y el ORB. Su propósito es lograr que la petición de un cliente llegue hasta el ORB. Se logra con él el acoplamiento entre el lenguaje de programación en que se escribe el cliente (java) y el ORB. El *stub* crea y expide las solicitudes del cliente.
- ***HabilidadOperations.java***: Esta interfaz contiene la versión Java del *idl* de la habilidad (*Habilidad.idl*). Simplemente, suele tener dos métodos *activate()* y *deactivte()* para activar o desactivar la habilidad.
- ***Habilidad.java***: Esta interfaz extiende *HabilidadOperations.java*
- ***HabilidadHelper.java***: Esta clase es de tipo *final* y proporciona funcionalidad auxiliar como el método *narrow()* que se necesita para adaptar las referencia de los objetos CORBA con sus tipos propios. Proporciona un mapeo detallado para traducir cada campo CORBA en su respectivo campo del lenguaje Java. Esto permite hacer el mapeo reverso más simple.
- ***HabilidadHolder.java***: Esta clase es de tipo *final* contiene una instancia pública de tipo *Habilidad*. Proporciona operaciones para los argumentos *out* y *inout* que CORBA tiene pero no se mapean fácilmente a las semánticas de Java.

5: Compilar todos los paquetes generados por el idlj para obtener *bytecode* de java.

6: Empaquetar todos los paquetes generador por el compilador *javac* en un fichero jar.

6.1: En el caso del framework desarrollado en el departamento, se nombra este fichero *ComplexSkill.jar*.

6.2: En caso de usar Mobility, se nombra *Mobility.jar*.

7: Cargar el nuevo jar a la carpeta adecuada del servidor Web según su estructura. En caso de Apache se carga este fichero a la carpeta /home/httpd/classes.

8: Escribir el servlet de la habilidad. Para localizar el objeto remoto de la habilidad, hay que escribir el servlet según la estructura de este objeto en el *framework*. El fabricante del robot B21 [iRobot,03] ofrece la posibilidad de usar su propio software Mobility (ver Apéndice D). Este software tiene unos objetos CORBA registrados, llamados contenedores, como son el objeto B21 o el Láser. Pero también ofrece la posibilidad de registrar otros objetos CORBA como es el caso de las habilidades, *GoToPoint*, *Rotate*, *LookObs*, etc. El otro *framework* desarrollado en el departamento permite acceder a habilidades complejas utilizando el mismo puerto como se puede ver en la figura 6.20. Para acceder a estos objetos se usa el ORB (se inicializa en el puerto 1358 de modelado1). Por debajo de estos objetos CORBA se tiene una jerarquía de clases y objetos que son accesibles por medio de métodos definidos en las interfaces IDL.

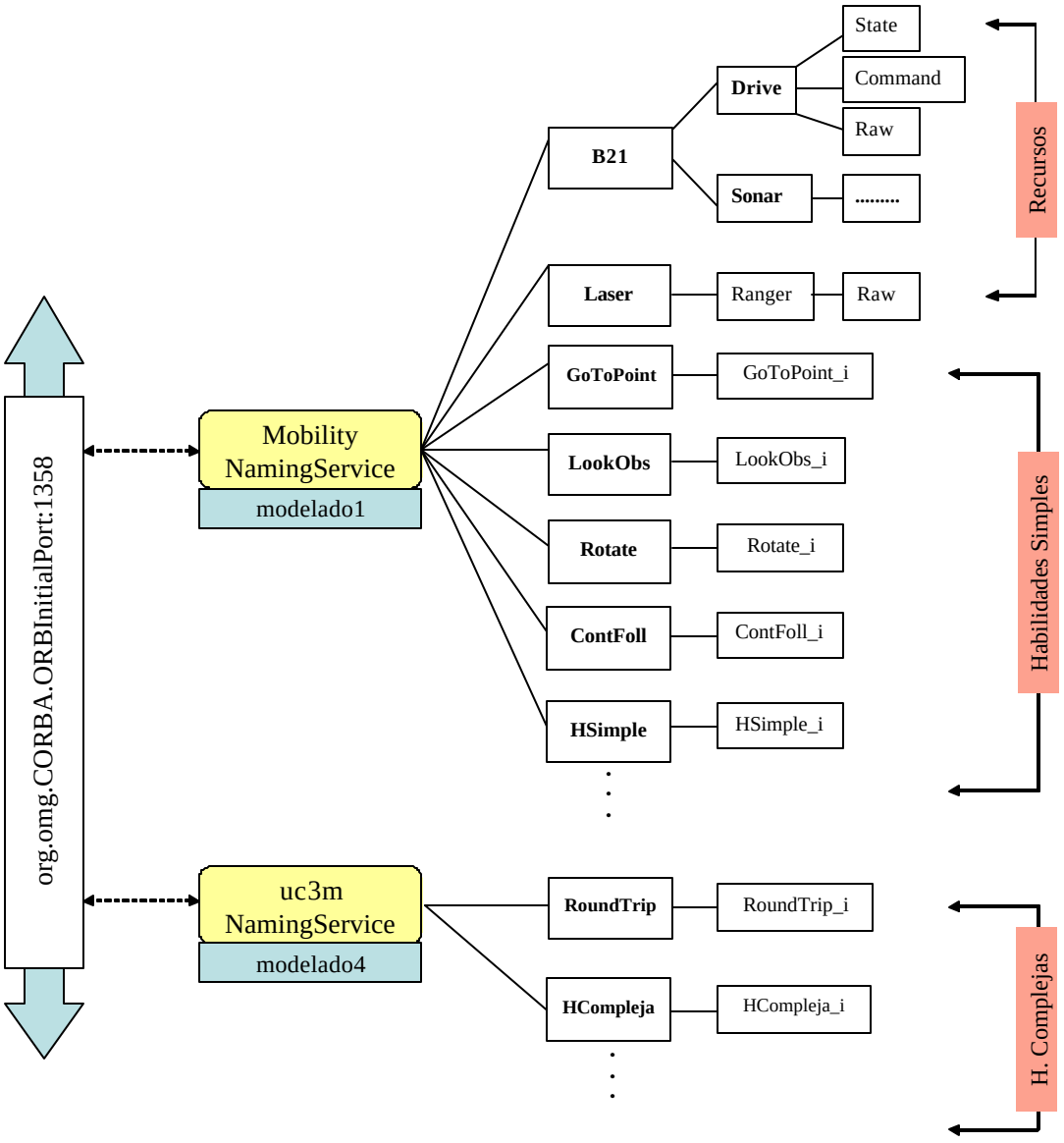


Fig. 6.20: Jerarquía Simplificada de los Objetos Distribuidos

En el apartado 6.6 se explican los pasos para configurar el servlet de la habilidad según su estructura en cada framework.

6.5 CAPA DE SERVIDOR

La capa del servidor contiene cuatro agentes principales como se muestra en la figura 6.21.

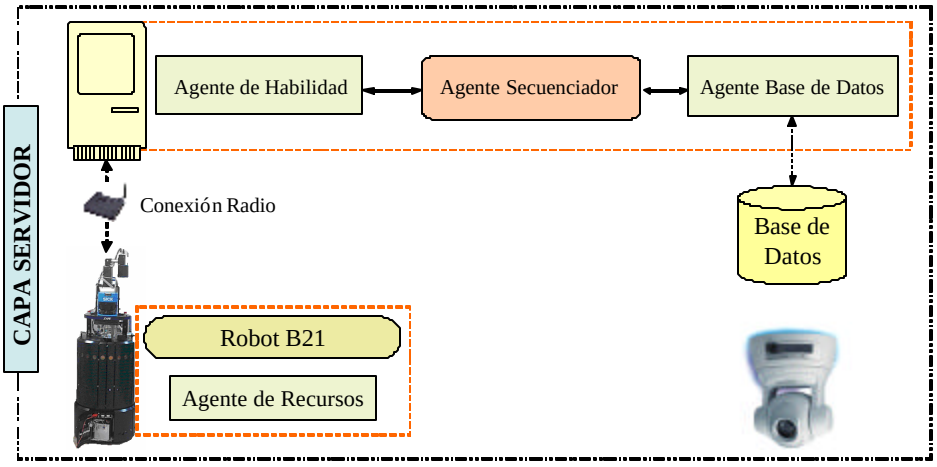


Fig. 6.21: Capa de Servidor

6.5.1 Agente de Habilidad

El agente de la habilidad representa el servidor de la habilidad, el cual según la arquitectura AD (ver Apéndice B), podrá ser un servidor de una habilidad automática simple o de una habilidad deliberativa. Las habilidades simples pueden ser combinadas para formar habilidades complejas usando el agente secuenciador.

Como se ha mencionado en el capítulo 3, las habilidades han sido realizadas en forma de módulos cliente-servidor, y por lo tanto actúan tanto como clientes o como servidores. En la estructura genérica de las habilidades, cada habilidad contiene un objeto activo, un objeto manejador de eventos y objetos de datos [Boada,02-a]. El objeto activo es el encargado de llevar a cabo la ejecución de la tarea de la habilidad. En la figura 6.22 se muestran los diferentes estados de un objeto activo en una habilidad y las relaciones entre ellos. Cuando el objeto se crea, su estado es no inicializado. El creador del objeto debe inicializarlo para que pueda ser utilizado y pasado como argumento a otras interfaces.

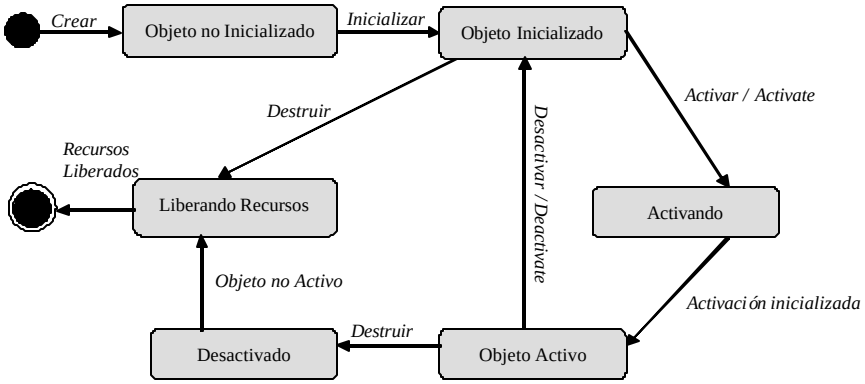


Fig. 6.22: Ciclo de Vida de un Objeto en una Habilidad Genérica

Una vez inicializado, a través del método *activate* puede llegar a ser un objeto activo, es decir, puede recibir su propio hilo de control. Los parámetros de ejecución se pasan en este método. En el estado activo es donde el objeto:

- ejecuta la habilidad,
- notifica a las habilidades, que se han registrado, el evento producido y,
- maneja los eventos que recibe de otras habilidades.

Cuando se invoca al método *deactivate*, el objeto pasa de nuevo al estado inicializado a la espera de ser de nuevo activado. Este método puede devolver un informe del estado en que ha quedado la habilidad. El final del estado se produce cuando se destruye el objeto y se liberan los recursos [Boada,02-a].

6.5.2 Agente Secuenciador

Una de las tareas principales que se realiza dentro de un robot móvil, es la ejecución de tareas a partir de planes predefinidos. Un plan se define como el conjunto de habilidades (motoras o sensorimotoras) que deben ejecutarse en un momento dado a fin de conseguir un objetivo. Se puede definir el secuenciador como el motor de inferencia encargado de la correcta ejecución de un plan [Rivero,03].

Secuenciar planes, es una actividad que se realiza en los dos niveles de la arquitectura híbrida AD (ver Apéndice B). En el nivel Automático, uno de los mecanismos existentes para el desarrollo o construcción de habilidades automáticas complejas, se basa en la secuenciación de habilidades automáticas simples, desarrollando de esta forma habilidades complejas.

En el nivel Deliberativo, la ejecución de plan de navegación consta de un conjunto de habilidades que deben ejecutarse a fin de cumplir una misión, su ejecución se realiza secuenciando habilidades automáticas simples o complejas. El proceso de puesta en marcha de un robot, se realiza ejecutando el plan de arranque del robot.

Al secuenciar un plan de navegación debe realizarse una supervisión del plan, a fin de poder tomar acciones alternativas en caso de que el plan falle. Una habilidad compleja puede realizar cálculos o procesos durante la ejecución de un plan. Sin embargo, su núcleo principal se encarga de la ejecución de un plan.

El secuenciador es modelado e implementado con un agente secuenciador. Cada proceso es ejecutado por un agente encargado de ejecutar los procesos y eventos asociados a cada acción de un plan y adicionalmente, calcular la siguiente acción a ejecutar una vez ejecutada y concluida la acción anterior. La figura 6.23 muestra la arquitectura software del secuenciador [Rivero,03].

En esta figura se definen tres componentes *State* que representa los tres estados que el secuenciador puede encontrarse, estos son:

- **IdleState:** Representa el estado *idle* del secuenciador, en este estado se construye la red que contiene el plan, y se ejecutan los métodos iniciales (opcional) a la ejecución de un plan.

- **ExecState:** Dentro de este estado se realiza la ejecución del plan, inicialmente se obtiene la etiqueta que indica el inicio del plan y se obtienen los procesos definidos en esta etiqueta. Posteriormente y de manera concurrente se lanza un agente secuenciador por cada proceso en un hilo de ejecución independiente, quedando el proceso principal, en espera de una señal que indique que alguno de los procesos ha finalizado.
- **ErrorState**: En el estado error se realizan las notificaciones necesarias a los procesos interesados en ser notificados de la presencia de errores y, si se pueden definir acciones que permitan resolver la causa de error (opcional). Si las causas de error se eliminan y la secuencia puede continuar, se modifica la etiqueta de inicio de la secuencia y se regresa nuevamente a ejecutar al estado de ejecución para reiniciar la ejecución del plan. De lo contrario, se pasa al estado *idle*.

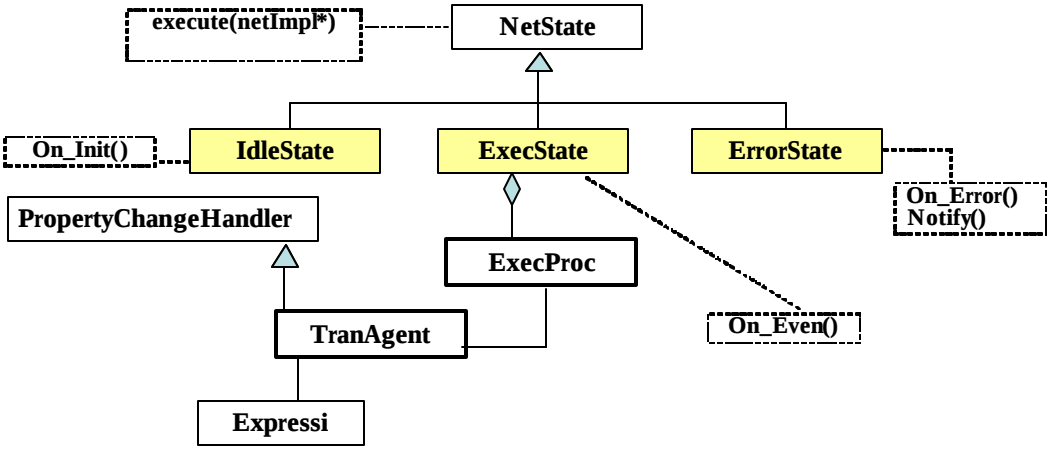


Fig. 6.23: Arquitectura del Secuenciador

En el capítulo siguiente, se puede ver el desarrollo de la interfaz de una habilidad generada por el secuenciador.

6.5.3 Agente de Base de Datos

La secuencia de una habilidad compleja puede ser generada de dos formas, estas son:

- Almacenando la secuencia en un archivo de texto siguiendo las reglas gramaticales de un lenguaje libre de contexto.
- Utilizando una herramienta gráfica para su construcción.

En la primera forma, el análisis de una secuencia se realiza en dos fases, la primera consiste de una análisis sintáctico que permita verificar si la secuencia cumple con las reglas gramaticales del lenguaje, ésta fase se realiza a través de un analizador léxico (YACC y LEX) [Rivero,03]. Para la segunda fase en la que se verifica si la secuencia es válida, un análisis semántico es necesario, para facilitar este análisis, se define una base de datos que permita almacenar la información referente a:

- Datos propios de cada habilidad.
- Datos que permitan verificar las restricciones que pueda tener asociada la habilidad.

En la segunda forma, utilizando una herramienta gráfica, el análisis sintáctico no es necesario, ya que la herramienta guía la construcción de la secuencia, garantizando el cumplimiento de las reglas gramaticales, sin embargo, el análisis semántico es necesario.

Se puede concluir que ambas formas necesitan de una base de datos para almacenar y gestionar los datos de una secuencia cuando se analiza. La herramienta gráfica permite la generación de secuencias básicas para usuarios inexpertos, restringiendo su acceso y controlando los valores asignados a los parámetros de cada habilidad utilizada en la secuencia. Adicionalmente, las secuencias generadas con la interfaz gráfica serán almacenadas en la base de datos. Esto permite utilizar una implementación adicional de la clase *NetImpl*, que use para la construcción de la secuencia esta base de datos.

Para lograr todo lo anteriormente expuesto, se propone utilizar un agente manipulador de la base de datos encargado de:

- Gestionar y almacenar los datos y restricciones de cada habilidad simple y compleja.
- Permitir almacenar y verificar la validez de las nuevas secuencias.
- Responder a las peticiones realizadas por la clase *NetImpl* para cada secuencia que se este ejecutando en el robot.

6.5.4 Agente de Recursos

El agente de recursos representa los servidores de los recursos del robot implementado en C++ como el servidor de la base (servidor del actuador, servidor de la odometría y servidor del sonar), el servidor del láser, etc. En el apéndice D, se describen el robot b21 usado durante el desarrollo de la presente tesis y el *framework* Mobility del robot que proporciona los módulos software básicos.

Los servidores de los sensores o los sensores virtuales proporcionan a las habilidades automáticas y acciones refleja los datos leídos de los sensores físicos del robot [Boada,02-a]. Estos sensores informan sobre el entorno en el que el robot se mueve (Ej. sensores de ultrasonidos, sensor del láser, sensores de infrarrojos, sistema de visión, etc.), o bien sobre su estado interno (Ej. sensores de odometría). La figura 6.24 muestra la estructura genérica de un sensor virtual.

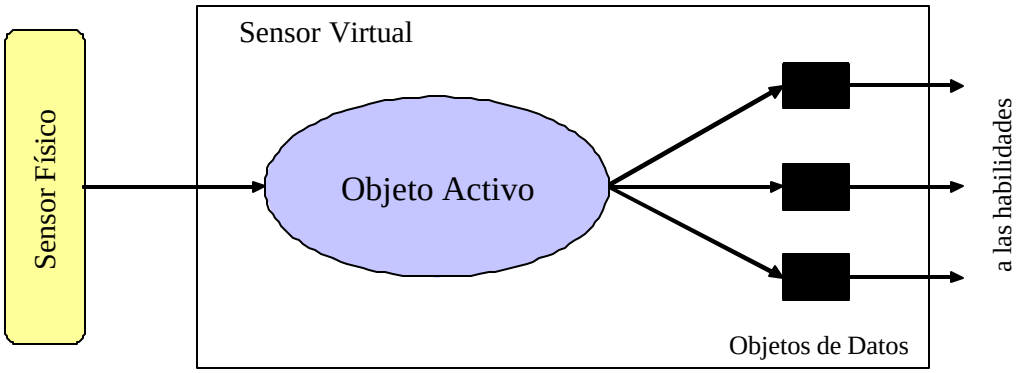


Fig. 6.24: Estructura Genérica de un Sensor Virtual

Se han implementado bs servidores de movimiento en forma de actuadores virtuales, que envían comandos de movimiento procedentes de las habilidades automáticas y acciones reflejas a los actuadores físicos del robot. El actuador virtual se encarga también de gestionar los comandos de control para impedir ninguna inconsistencia de los datos [Boada,02-a]. Al igual que ocurre con los sensores virtuales, éstos son servidores que contienen un objeto activo y objetos de datos, ver la figura 6.25.

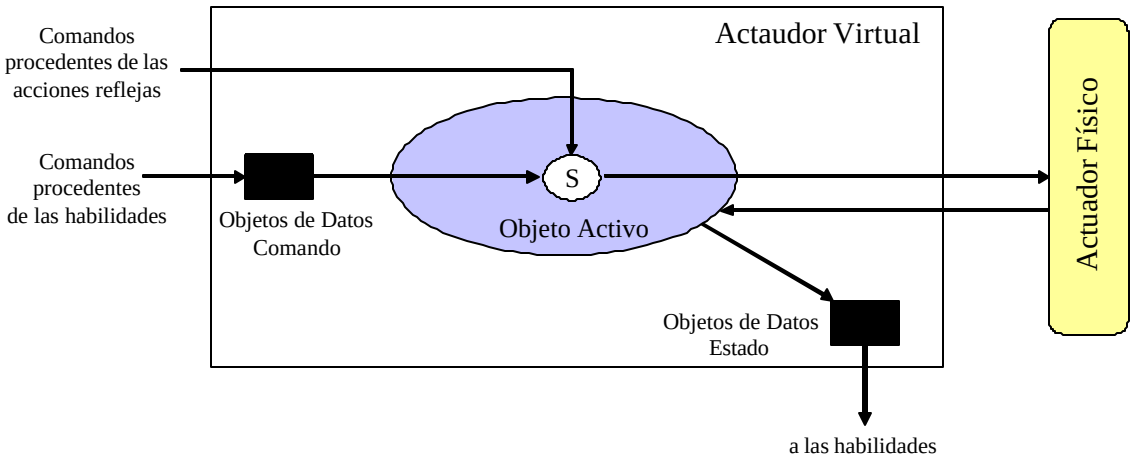


Fig. 6.25: Estructura Genérica de un Servidor Virtual

Un actuador no sólo recibe comandos de movimiento, también puede recibir comandos para informar acerca de su estado interno: velocidad a la que está girando, posición en la que se encuentra etc. El servidor de la base y el actuador del la torreta o plataforma *Pan-Tilt* sobre la que está montada la cámara del sistema de visión, son dos ejemplos de actuadores virtuales.

6.6 INTERACCIÓN ENTRE LAS CAPAS

Como se ha mencionado en el capítulo anterior, se usa el patrón cliente-servidor implementado en Java para comunicar el cliente de una habilidad con su servlet. La comunicación entre el servlet y el servidor remoto de la habilidad está basada en el patrón *broker* o intermediario. En la figura 6.26, se puede ver la interacción entre los tres objetos distribuidos que son el cliente de la habilidad *ClienteHabilidad*, su servlet *ServletHabilidad* y su servidor remoto, *ServidorHabilidad*.

Durante la interacción, el objeto *ServletHabilidad* no llama directamente al objeto *ServidorHabilidad* sino a través de una interfaz IDL (*IDLHabilidad*). Dicho objeto llama a los métodos de un objeto *proxy* que implementa la interfaz *IDLHabilidad*. Como esta clase llama métodos a través de una interfaz, no tiene que ser consciente del hecho de que está llamando los métodos de un objeto *stub* que es un proxy del objeto *ServidorHabilidad* en lugar de llamar el propio objeto *ServidorHabilidad*.

El objeto S*tubHabilidad* encapsula los detalles de cómo se realizan las llamadas al objeto *ServidorHabilidad* y dónde se encuentra el servidor. Estos detalles son transparentes al objeto *ServletHabilidad*. También el objeto S*tubHabilidad* forma un mensaje que contiene información

de identificación del objeto *ServidorHabilidad*, el método que está por llamar y los valores de los argumentos del método. En otro lado de la conexión, una parte del mensaje es interpretado por el objeto *DistribuidorDeLlamadas* (*CallDispatcher*) y el resto por el objeto *EsqueletoHabilidad*. Las clases de conexión son responsables del transporte del mensaje entre el entorno de *ServletHabilidad* (el Servidor de Web) y el de *ServidorHabilidad* (el Servidor de Robot).

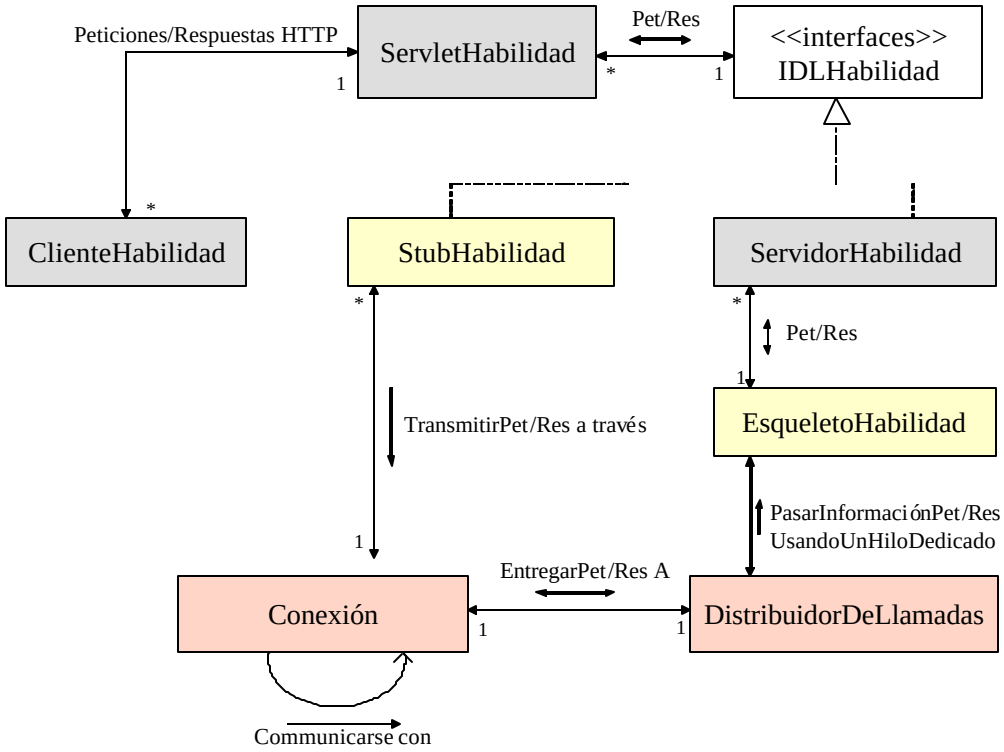


Fig. 6.26: Interacción Cliente/Servlet/Servidor

El *DistribuidorDeLlamadas* recibe el mensaje a través del objeto de conexión desde un objeto *stub* remoto y entonces se lo pasa a una instancia de clase esqueleto adecuado. El objeto *DistribuidorDeLlamadas* es responsable de identificar el *ServidorHabilidad* cuyo método va a ser llamado por el objeto esqueleto.

Las clases de esqueleto son responsables de llamar los métodos del objeto *ServidorHabilidad* de parte del objeto remoto *ServletHabilidad*. El objeto esqueleto extrae los valores de los argumentos del mensaje pasado por el *DistribuidorDeLlamadas* y entonces llama al método indicado pasándole los valores obtenidos. Si el mensaje llamado devuelve una respuesta, entonces el objeto esqueleto es responsable de crear un mensaje que contiene el valor de vuelta y lo devuelve al *stub* para que lo devuelva al servlet y luego al cliente. Si el mensaje llamado lanza una excepción, el objeto esqueleto es responsable de crear un mensaje apropiado.

La clase *ServidorHabilidad* (implementado en C ++) implementa la interfaz *IDLHabilidad*. Instancias de esta clase se pueden llamar localmente a través de la interfaz *IDLHabilidad*, o remotamente a través de un objeto *stub* que implementa la misma interfaz.

Se pueden utilizar los pasos siguientes para implementar la interacción entre el cliente, el servlet y el servidor de una habilidad como se muestra en la figura 6.27.

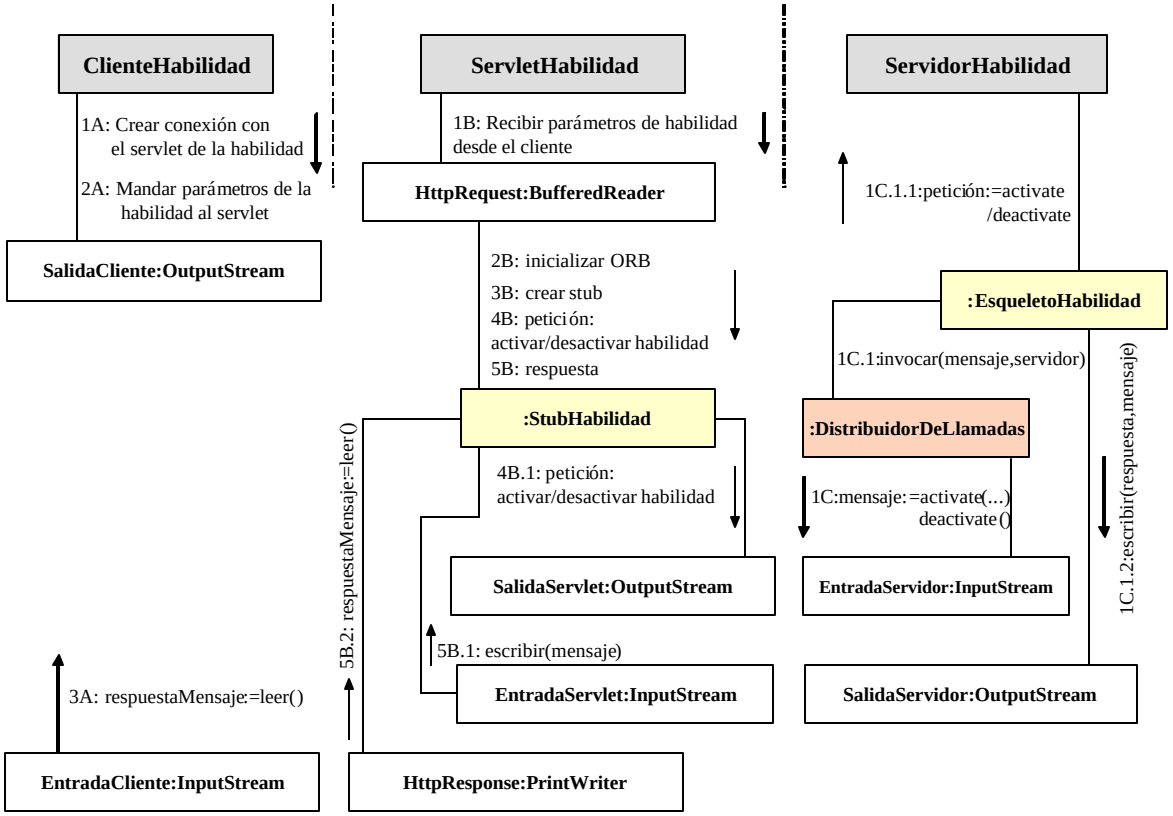


Fig. 6.27: Pasos de la Interacción

1A: El objeto *ClienteHabilidad* crea una conexión HTTP con el objeto *ServletHabilidad*.

```
URL ServletHabilidadURL = new URL(url);
URLConnection ServletHabilidadConexion = ServletHabilidadURL.openConnection();
ServletHabilidadConexion.setDoOutput(true);
ServletHabilidadConexion.setUseCaches(false);
```

2A: El cliente envía al servlet los parámetros necesarios para activar/desactivar la habilidad.

```
PrintStream salida = new PrintStream(ServletHabilidadConexion.getOutputStream());
out.println(param1);
out.println(param2);
out.println(param3);
.
.
.
out.close();
```

1B: *ServletHabilidad* recibe los parámetros enviados por el cliente.

```
// En el método doPost
BufferedReader desdeClienteHabilidad = req.getReader();
```

```
StringBuffer msgBuf = new StringBuffer();
String[] parameteros= new String[N]; //N número de parámetros
String linea;
int i=0;
while ((linea=desdeClienteHabilidad.readLine())!=null) {
      if (msgBuf.length()>0) msgBuf.append('\n');
            setParametros(linea,i);
            i++;
            msgBuf.append(linea);
      }
desdeClienteHabilidad.close();
```

2B: El servlet inicializa el ORB.

```
Properties props = new Properties();
props.put("org.omg.CORBA.ORBInitialPort",#);    //# es el número del puerto=1358
String[] args = new String[]{""};
ORB orb = ORB.init(args, props);
```

3B: El servlet obtiene el objeto de *stub* para llamar al objeto del servidor remoto. Los ORB proveen un mecanismo por lo cual se coge el nombre lógico de un objeto y se crea un objeto *stub* en consultación con un mecanismo que usa el patrón *Registry* para proporcionar el lugar del objeto servidor.

```
org.omg.CORBA.Object objRef = orb.resolve_initial_references("NameService");
NamingContext ncRef = NamingContextHelper.narrow(objRef);
NameComponent nc = new NameComponent("Habilidad", "Framework");
                                                //Framework = Mobility o uc3m
NameComponent path[] = {nc};
org.omg.CORBA.Object objRefHabilidad = ncRef.resolve(path);
FrameworkComponents.SystemModule HabilidadRef =
FrameworkComponents.SystemModuleHelper.narrow(objRefHabilidad);
//FrameworkComponents es MobilityComponents  en caso de Mobility  y
//RobotKernelObject  en caso de uc3m
FrameworkCore.ObjectContainer container = HabilidadRef;
//FrameworkCore es MobilityCore  en caso de Mobility  y
//RobotKernelObject  en caso de uc3m
String[] Habilidad_i = new String[]{"Habilidad_i"}; //Habilidad_i es el objeto de  la
                                                      implementación de la habilidad (ver
                                                      Figura 6.x)
org.omg.CORBA.Object obj = container.find_object(Habilidad_i);
HabilidadRef = RobotHabilidad.HabilidadHelper.narrow(obj); //en caso de Mobility
HabilidadRef = ComplexSkill.SkillComplexHelper.narrow(obj); //en caso de uc3m
```

4B: El servlet llama al objeto *StubHabilidad* para activar/desactivar la habilidad con intención de llamar el objeto de la habilidad que puede ser remoto o no.

```
HabilidadRef.activate (getParametros(0), getParametros(1), …);       //activar
HabilidadRef.deactivate();                                          //desactivar
```

4B.1: El objeto *StubHabilidad* pide al objeto *SalidaServlet* escribir un mensaje que incluye el nombre de la clase de la habilidad, el nombre del método y los argumentos que el objeto servlet pasa para activar o desactivar la habilidad. El objeto *SalidaServlet* pasa el mensaje a través de una conexión al objeto *EntradaServidor*.

1C: El objeto *DistribuidorDeLlamadas* extrae del mensaje el nombre del objeto que quiere llamar. Entonces, obtiene el objeto actual cuyo método está por llamar. Usando varios hilos, el objeto *DistribuidorDeLlamadas* llama asincrónicamente el método invocar del objeto *EsqueletoHabilidad*.

1.C.1.1: El objeto *EsqueletoHabilidad* extrae del mensaje el nombre del método a llamar y los argumentos. Entonces llama al método activate/deactivate del objeto del servidor.

1.C.1.2: El objeto *EsqueletoHabilidad* construye la respuesta producida por la llamada al método activate/deactivate del objeto de habilidad. Éste puede ser un valor devuelto o una excepción. Entonces pasa el mensaje al método escribir del objeto *SalidaServidor* a través de una conexión.

5B: El servlet devuelve el mensaje de la respuesta al objeto cliente.

```
resp.setContentType("text/plain");
PrintWriter aClienteHabilidad = resp.getWriter();
aClienteHabilidad.println(getRespuesta());
aClienteHabilidad.close();
```

5B.1: El método leer del objeto *EntradaServlet* devuelve el mensaje de la respuesta al objeto *StubHabilidad*.

5B.2: El objeto *StubHabilidad* extrae del mensaje la respuesta. Si la respuesta es un valor, lo devuelve en forma de respuesta HTTP. Si la respuesta es una excepción, la tira.

3A: El objeto *EntradaCliente* lee la respuesta a través de una conexión de red (Internet) y la devuelve al objeto *ClienteHabilidad*.

```
InputStream entrada = ServletHabilidadConexion.getInputStream();
StringBuffer respuesta = new StringBuffer();
int chr;
while ((chr=in.read())!=-1) {
     respuesta.append((char) chr);
}
entrada.close();
```

Capítulo

7

IMPLEMENTACIÓN

Capítulo 7 IMPLEMENTACIÓN

7.1 INTRODUCCIÓN

En el capítulo anterior se ha presentado una arquitectura cliente-servidor de tres capas para un sistema de interacción remota con robots móviles basado en Internet. En el presente capítulo se analiza detalladamente el diseño y la implementación de las interfaces de las diferentes habilidades automáticas tanto simples como complejas utilizando la arquitectura propuesta. Se pueden utilizar estas interfaces para ajustar los parámetros de las habilidades y activar o desactivar estas habilidades remotamente.

Como se ha mencionado en el capítulo 3, el control supervisado pretende hacer que la tarea de control sea compartida entre un lazo de control remoto y el operador. Este paradigma de control no es para que un robot realice todas las operaciones autónomamente, pero habilita al robot a realizar operaciones simples que el operador puede secuenciar. Por lo tanto, lanzando varias interfaces de distintas habilidades tanto simples como complejas, el usuario puede secuenciar estas habilidades manualmente para llevar a cabo una tarea concreta.

Se han realizado varios experimentos utilizando la arquitectura propuesta. En este capítulo, se describe también la integración de todos los experimentos implementados en un entorno educativo de robótica móvil.

A continuación, se describen los componentes de este entorno educativo tales como: las clases en línea, el laboratorio remoto, las herramientas de evaluación de los estudiantes, los mecanismos de comunicación y finalmente la evaluación del sistema por parte de los estudiantes.

7.2 DESCRIPCIÓN DE LOS EXPERIMENTOS IMPLEMENTADOS

Se han implementado varios experimentos utilizando la arquitectura propuesta. Se pueden usar estos experimentos en un laboratorio remoto de robótica móvil o en cualquier entorno de interacción remota con robots móviles basado en Internet.

7.2.1 Herramientas Genéricas

En las subsecciones siguientes se presentan algunas herramientas genéricas, que se pueden usar en varios experimentos. Dichas herramientas se diseñaron como componentes reutilizables, de manera que se puedan utilizar para construir varias interfaces para distintos experimentos como ya se ha comentado en el capítulo anterior.

7.2.1.1 Modelo 2D del Laboratorio

Como se ha mencionado en el capítulo 2, se pueden usar varias herramientas para proporcionar realimentación al usuario. La realimentación más frecuente es la realimentación visual mediante la transmisión de vídeo. La transmisión de vídeo exige la disponibilidad de un ancho de banda alto. Por eso se pueden utilizar los modelos gráficos y los modelos de realidad virtual como una alternativa para proveer la realimentación visual.

Utilizando modelos gráficos 2D o 3D del robot y su entorno de trabajo, el usuario puede tener la sensación visual que expresa la secuencia de sus comandos directamente en la interfaz de control.

El modelo 2D del laboratorio presenta una interfaz completamente análoga a la del entorno real. Para desarrollar este modelo, se ha tenido que estudiar el sistema de coordenadas. Los tres sistemas de coordenadas que se pueden ver en la figura 7.1 son:

- Sistema de Coordenadas del Robot (X_r, Y_r, θ_r): con origen en el punto en el que se enciende el robot (punto inicial) y unidad de medida en metros.
- Sistema de Coordenadas del Laboratorio (X_l, Y_l, θ_l): con origen cerca de la esquina superior derecha y unidad de medida en metros.
- Sistema del panel (X_p, Y_p): con origen en la esquina superior izquierda del panel gráfico, unidad de medida en píxeles.

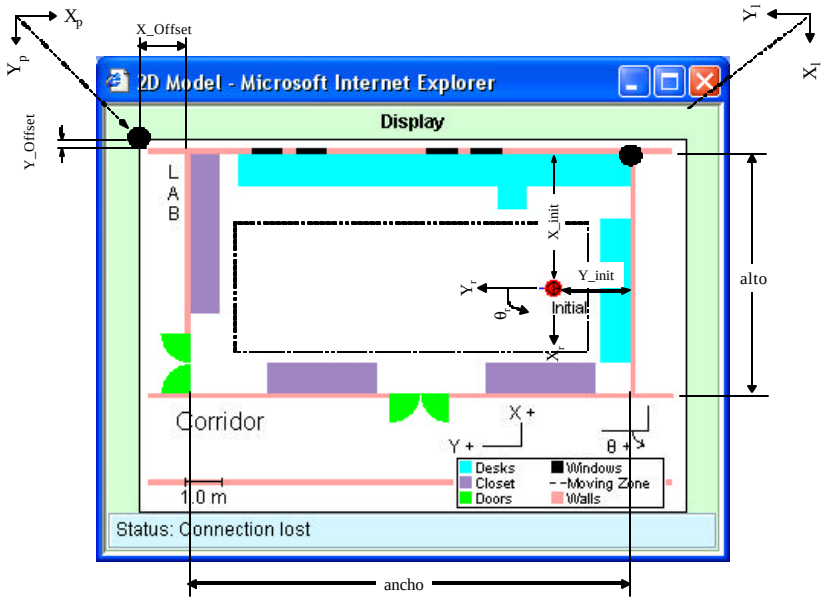


Fig. 7.1: Sistemas de Coordenadas

Los sistemas no están elegidos al azar, estos tienen una razón de ser. Se ha elegido el sistema del laboratorio para hacer más comprensibles los datos de posición que aparecen en la pantalla (el robot devuelve la posición respecto de su posición inicial). El sistema del robot viene predefinido, debido a que el servidor de la base del robot proporciona la posición respecto a su sistema. Por el lado de la pantalla, el del panel es el sistema de coordenadas que los métodos gráficos de Java toman por defecto, y es el utilizado para la representación gráfica.

Los cambios de sistemas de coordenadas necesarios serán:

Sistema de Coordenadas del Robot → Sistema de Coordenadas del Laboratorio

$$X_l = X_r + X\_init \qquad \text{Ecuación 7.1}$$

$$Y_l = Y_r + Y\_init \qquad \text{Ecuación 7.2}$$

$$\theta_l = \theta_r \qquad \text{Ecuación 7.3}$$

Sistema de Coordenadas del Laboratorio → Sistema de Coordenadas del Panel

$$X_p = (ancho - Y_l) * ESC + X\_Offset \qquad \text{Ecuación 7.4}$$

$$Y_p = X_l + Y\_Offset \qquad \text{Ecuación 7.5}$$

Donde, ESC es la escala de representación píxeles/metro, X_init y Y_init es la posición inicial del robot.

Con estos cambios se ha conseguido, por un lado, tener la posición del robot (metros) respecto al sistema del laboratorio y, por otro, tener la posición en píxeles que ocupa cualquier punto del laboratorio teniendo sus coordenadas con respecto al sistema de coordenadas del laboratorio.

Como se muestra en la figura 7.2, el modelo desarrollado muestra información visual de la posición actual del robot, la orientación actual, la zona de trabajo, los obstáculos fijos que rodean el robot, el estado de la conexión, etc. Este modelo es un modelo dinámico que se puede cambiar durante la ejecución de la tarea utilizando las funciones del enfoque.

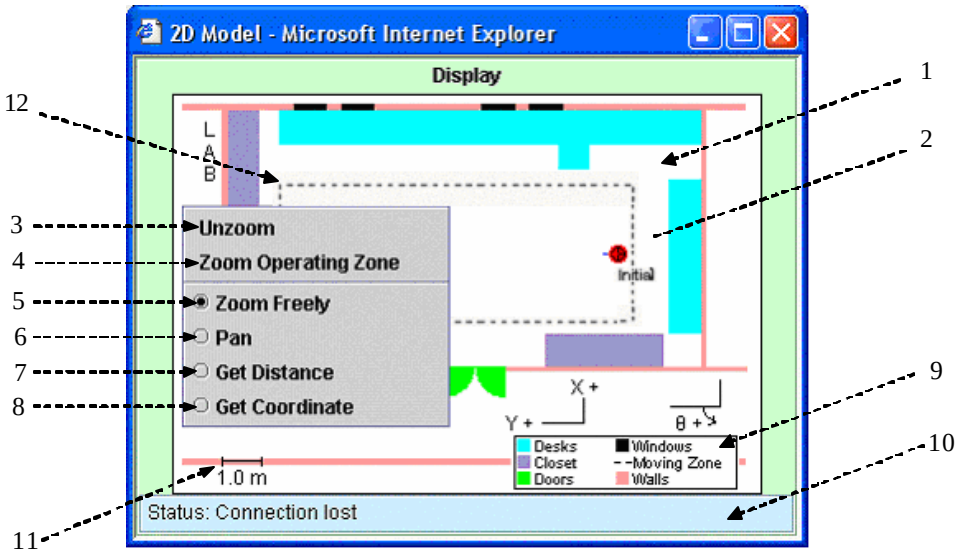


Fig. 7.2: Modelo 2D

En la tabla 7.1 se resumen las funciones implementadas en este modelo.

Tabla 7.1: Funciones del Modelo 2D

Función	Descripción
1	El panel *GraphicsPanel* que muestra el modelo del laboratorio
2	Estado actual del robot (posición-circulo rojo y orientación-línea azul).
3	Volver al enfoque por defecto.
4	Enfocar a la zona de trabajo.
5	Manteniendo el botón izquierdo del ratón apretado y moviendo el ratón a la derecha o a la izquierda para enfocar a una zona concreta.
6	Permite mover el modelo en cualquier dirección con el botón izquierda del ratón apretado.
7	Permite calcular la distancia entre dos puntos en el laboratorio.
8	Permite obtener las coordenadas de cualquier punto.
9	Leyenda del modelo
10	Estado de la conexión
11	Escala actual
12	Zona de trabajo

Este modelo está desarrollado teniendo en cuenta la necesidad de facilitar cualquier cambio en él en caso de cambiar el aula donde se encuentra el robot. Como se muestra en el diagrama de clases (véase la figura 7.3), se ha creado una clase abstracta denominada *EnvironmentModel*, que tiene los métodos necesarios para construir cualquier modelo 2D como ajustar las dimensiones del laboratorio, añadir los objetos del laboratorio tales como: paredes, ventanas, armarios, puertas, mesas, etc., asignar la zona del trabajo dentro de la cual se puede mover el robot, dibujar el robot, etc. Extendiendo esta clase, se puede construir un modelo específico para el aula donde se encuentra el robot actualmente. Por ejemplo, la clase *Lab1_3C12* extiende la clase genérica *EnvironmentModel* para formar el modelo del aula 1.3C12.

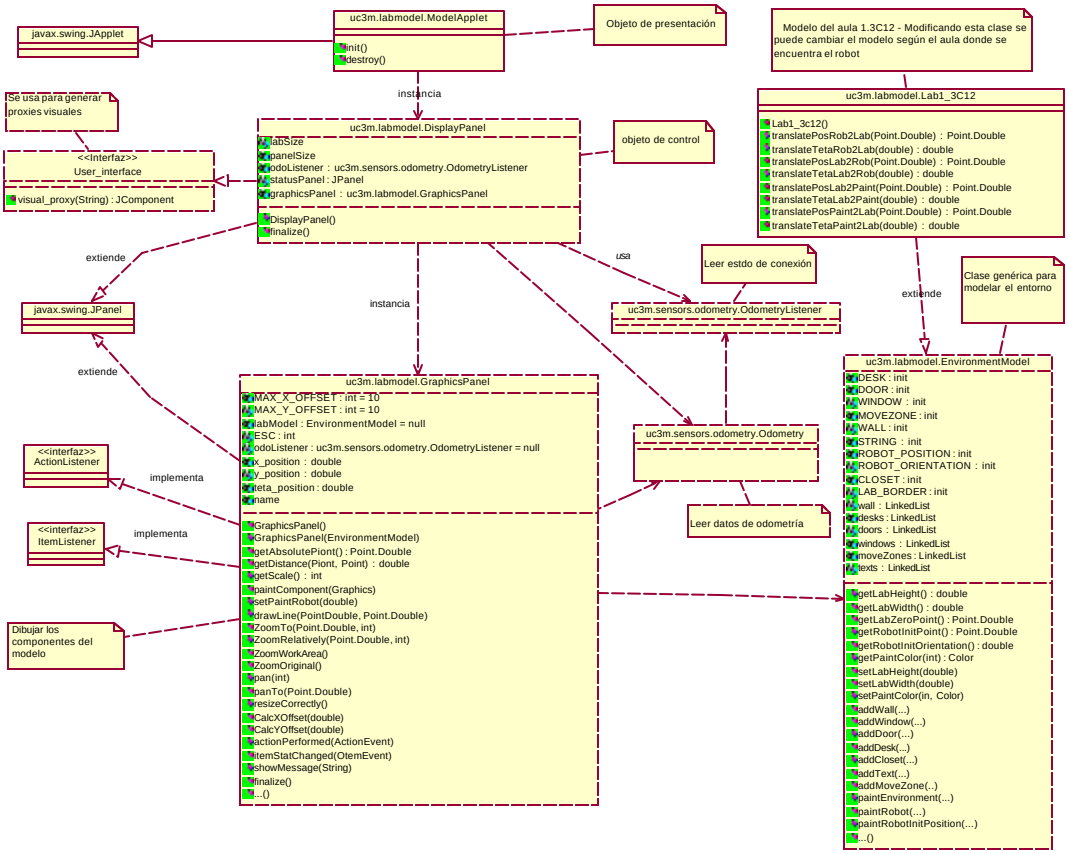


Fig. 7.3: Diagrama de Clases del Modelo 2D

A continuación, se explica la funcionalidad de cada una de las clases que forman el modelo 2d:

- ***DisplayPanel*:** Genera una instancia de la clase *GraphicsPanel* según el modelo del laboratorio actual (`graphicsPanel = new GraphicsPanel(new Lab1_3C12())`), luego la muestra en un panel y muestra el estado de la conexión con el servidor de la odometría del robot. Registra la interfaz *OdometryListener* a la clase *Odometry* en el paquete de la odometría para obtener información acerca del estado de la conexión.
- ***GraphicsPanel*:** Muestra el modelo del entorno de trabajo y ofrece funciones como ampliación, inclinación, posibilidad de medir distancias y obtener las coordenadas de cualquier punto en el laboratorio a través un menú desplegable. Registrando la interfaz *OdometryListener* en la clase *Odometry*, permite capturar los datos de odometría necesarios para dibujar el robot.
- ***EnvironmentModel***: Es una clase abstracta que sirve como modelo genérico para cualquier laboratorio. Define el sistema de coordenadas y los objetos del entorno. También define los colores que se usan para dibujar varios tipos de objetos y la configuración inicial del robot. Para usar esta clase, hay que extenderla, implementar los métodos de transformación de coordenadas, definir el punto cero del sistema de coordenadas del laboratorio, las dimensiones del laboratorio, la posición inicial del robot y finalmente añadir los diferentes objetos del laboratorio como paredes, ventanas, armarios, mesas, puertas, etc.

- ***Lab1_3C12***: Modelo del aula 1.3.C12 donde se encuentra el robot actualmente. Esta clase extiende la clase genérica *EnvironmentModel*.
- ***ModelApplet***: Esta clase contiene una instancia del *DisplayPanel* en un applet para que pueda mostrarse en un navegador.

Como trabajo futuro, se pretende desarrollar un modelo 3D para el laboratorio utilizando Java 3D.

7.2.1.2 Panel de la Odometría

Como se muestra en la figura 7.4, el panel de la odometría es el responsable de dar al usuario la información numérica de la posición y la orientación actual de robot, así como su velocidad lineal y angular.

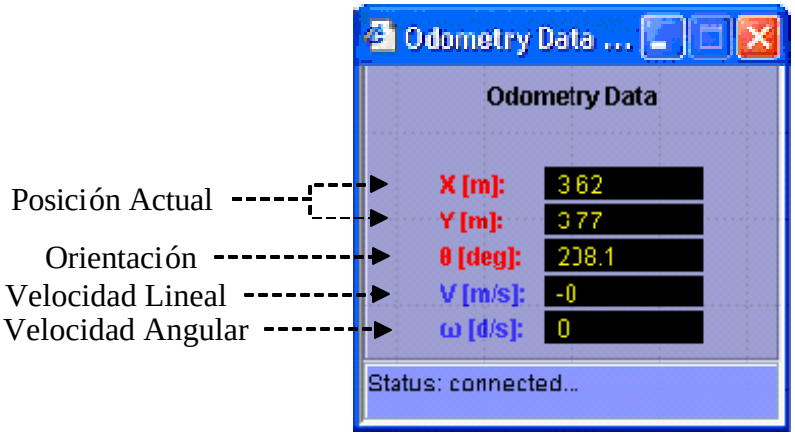


Fig. 7.4: Panel de la Odometría

Las coordenadas X, Y y ? que proporciona este panel están referidos al sistema de coordenadas del laboratorio. Toda la información es proporcionada por un hilo de ejecución independiente, que invoca la información sensorial y realiza el cambio de sistemas de coordenadas. Como se ha mencionado el la subsección anterior, es necesario realizar el cambio de coordenadas puesto que la información que proporciona la odometría del robot está referida al sistema de coordenadas del robot.

La figura 7.5 muestra las clases que forman el panel de control. A continuación se describe la función de cada clase.

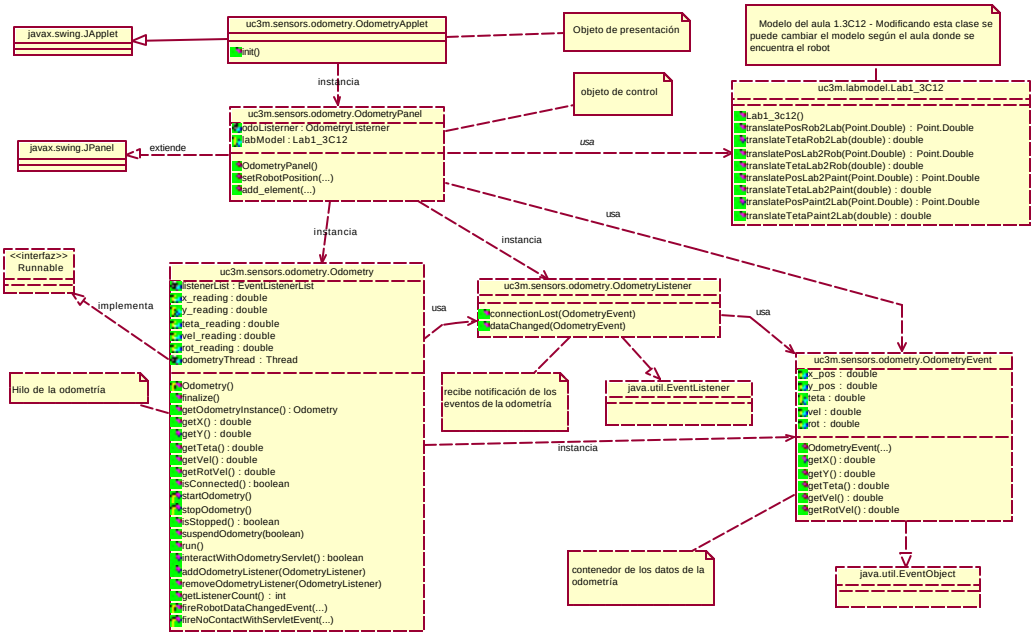


Fig. 7.5: Diagrama de Clases del Panel de la Odometría

- ***OdometryApplet***: Esta clase inserta un *OdometryPanel* en un applet para que pueda mostrarse en un navegador.
- ***OdometryPanel***: Muestra los datos de la odometría en un panel. Registra un *OdoemtryListerner* a una instancia de la clase *Odometry* para refrescar los datos. Usa la clase *Lab1_3C12* para obtener las transformaciones necesarias del sistema de coordenadas del robot al sistema de coordenadas del laboratorio.
- ***Odometry*:** Esta clase se comunica con el servlet de la odometría que se encuentra en el agente de repositorio dinámico para adquirir remotamente los datos de la odometría, que son la posición actual del robot y su velocidad lineal y angular. Esta clase manda eventos de notificación a todos los objetos registrados. Se comunica con el servlet con intervalos regulares si hay objetos registrados. En caso de que no haya ningún objeto registrado, deja de comunicarse con el servlet pero vuelve a comunicarse si recibe un registro nuevo. Si la conexión con el servlet falla, llama al método `OdometryListener.connectionLost(OdometryEvent)` para notificar a los objetos registrados acerca del fallo.
- ***OdometryListener***: Se puede usar esta interfaz para recibir las notificaciones de los eventos de la odometría y para saber el estado actual de la conexión.
- ***OdometryEvent*:** Se manda una instancia de esta clase a todos los *OdometryListener* registrados en la clase *Odometry*. Esta clase contiene los datos de la odometría.

7.2.1.3 Panel del Sonar

Como se puede ver en el apéndice D, el robot B21 dispone de un anillo de 24 sensores ultrasonidos Polaroid. El panel del sonar muestra la información sensorial de los 24 sensores en tres formatos, estos son: vector, puntos y segmentos tal como se puede ver en la figura 7.6.

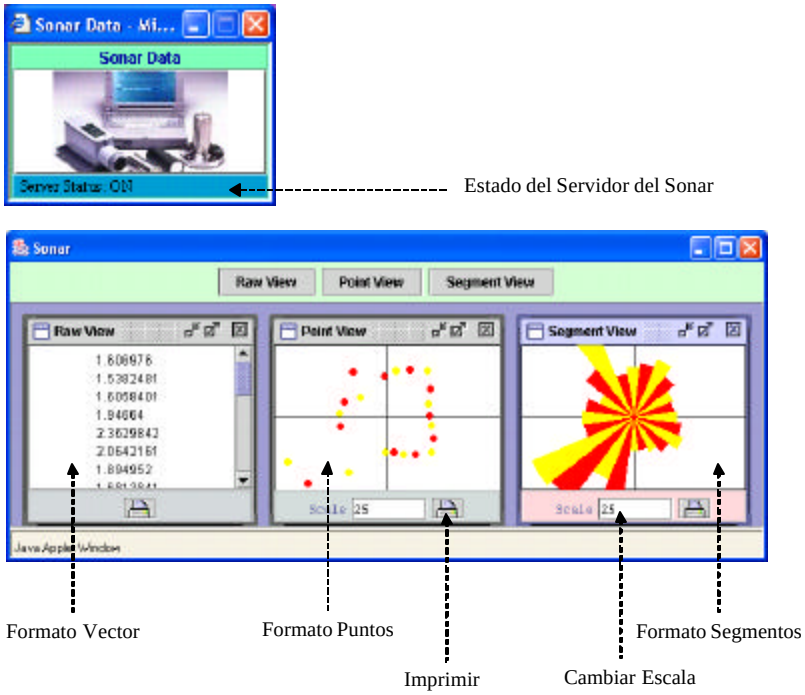


Fig. 7.6: Panel del Sonar

Como muestra en la figura 7.7, la unidad de medida ultrasónica Polaroid está formada por dos componentes, el transductor acústico y el circuito de medida [Polaroid,03]. Estos componentes juntos son capaces de detectar y calcular la distancia a la que se encuentran objetos dentro de un rango aproximado desde 27 cm. hasta 10.7 metros. En funcionamiento, un pulso se transmite hacia el objetivo y se detecta el eco resultante. El tiempo transcurrido entre la transmisión inicial y la detección del eco puede ser convertido en distancia a través de la velocidad del sonido. Para un pulso lanzado contra un objeto situado a 60 cm., el tiempo medio transcurrido es de 3.55 milisegundos.

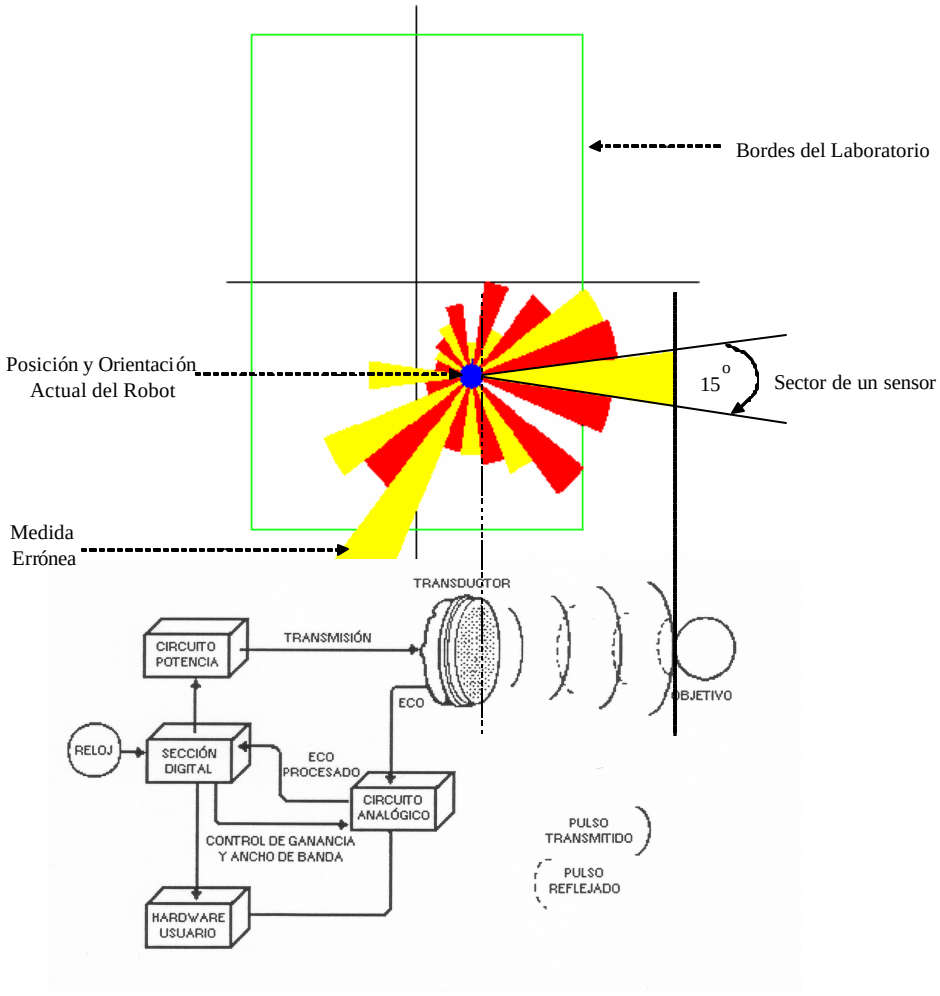


Fig. 7.7: Medidas del Sonar

La fuente de errores del sistema, se dividen en dos grupos; los debidos a la odometría y los causados por las características de los sensores de ultrasonidos.

Los errores debidos al cálculo de la posición del robot u odometría se deben principalmente a los deslizamientos, ya que el *encoder* no contempla esta posibilidad y la posición del robot va acumulando errores que pueden llegar a hacer totalmente inútil la lectura de la posición.

Entre los errores intrínsecos a los sensores de ultrasonidos se destacan los siguientes:

- Amplificador de ganancia variable con el tiempo. El amplificador analógico TVG Polaroid sólo hace una aproximación de 16 pasos a la curva exponencial ideal que cancelaría exactamente las pérdidas de atenuación y la dispersión del foco.
- Carga capacitiva en el circuito digital. El mecanismo por el cual se realiza la comparación se basa en la carga de un condensador, cuyo impacto puede ser un significativo foco de error cuando se usan pulsos de larga duración.
- Imposibilidad de detectar objetos pequeños y formas definidas a causa de la amplitud de 15 grados del sector angular y a la creación de sombras.
- Perdida de precisión para Radios mayores de 10 m. o menor de 30 cm.

- Problemas debidos al ángulo de incidencia y material (dureza y forma) de la superficie que refleja un eco.

La figura 7.8 muestra el diagrama de las clases que forman el panel del sonar.

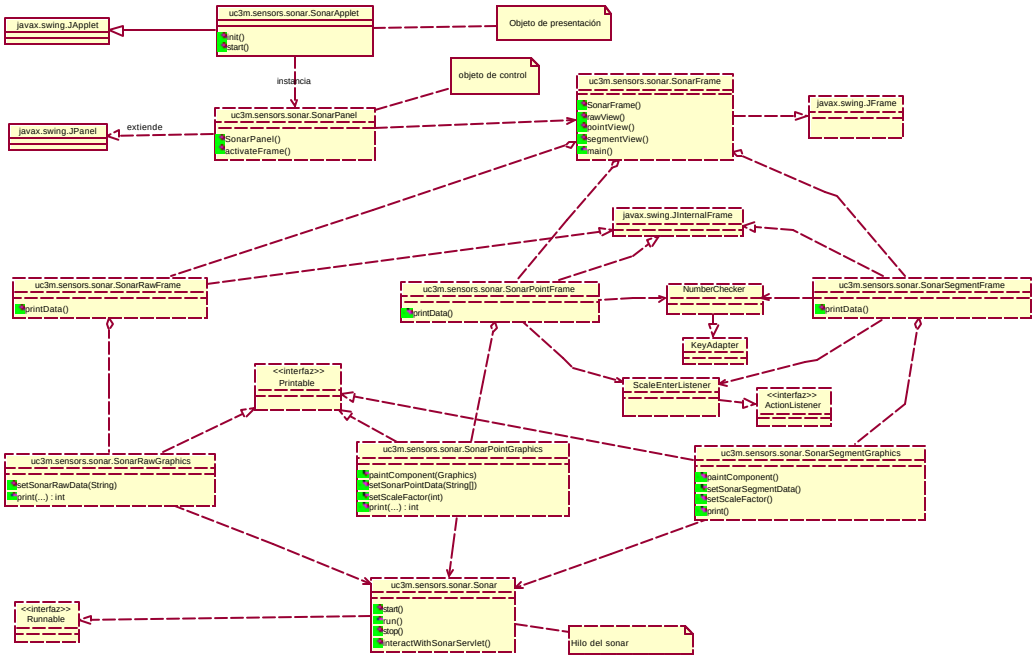


Fig. 7.8: Diagrama de Clases del Panel del Sonar

7.2.1.4 Panel del Láser

Al robot B21 se le ha incorporado un sensor láser que le proporciona una mayor precisión en el cálculo de distancias (ver apéndice D). Estos sistemas tienen mucha más exactitud que los ultrasonidos, siendo principalmente utilizados en exteriores.

Para la toma de medidas con el láser se ha utilizado un láser SICK de 50 m. de rango y 180 grados de cobertura, con una resolución de 0.5 ^{0} en ángulo y +/- 50 mm. en distancia [Sick,03]. El láser está situado mirando hacia el frente del robot, proporcionándonos por tanto gran precisión en las medidas tomadas en esta dirección. La figura 7.9 muestra el panel del láser que proporciona 361 medidas en tres formatos (vector, puntos y segmentos).

Como se puede observar, el formato vector representa la distancia (en metros) de los objetos que se encuentran en el entorno. El formato puntos representa un punto del objeto detectado en coordenadas del robot. Por último, el formato segmento representa el punto inicial y el punto final de la región del espacio libre desde el sensor láser hasta el objeto detectado.

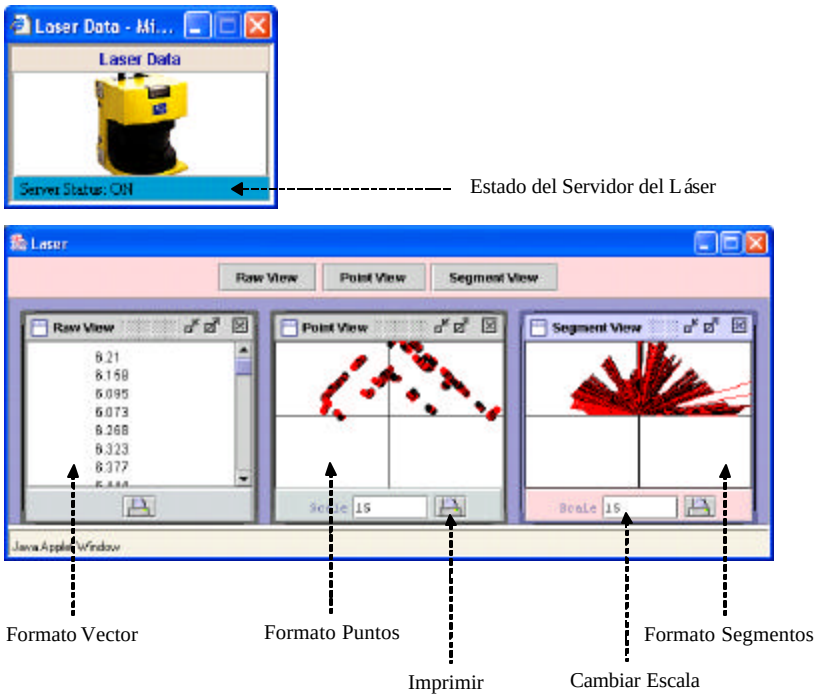


Fig. 7.9: Panel del Láser

La figura 7.10, muestra el diagrama de clases del panel del láser, como se puede observar, este diagrama es similar al de la figura 7.8. Sus implementaciones son muy similares. Su mayor diferencia está en la modificación de las fuentes de obtención de datos, en este caso el láser, y en la forma de dibujar los datos.

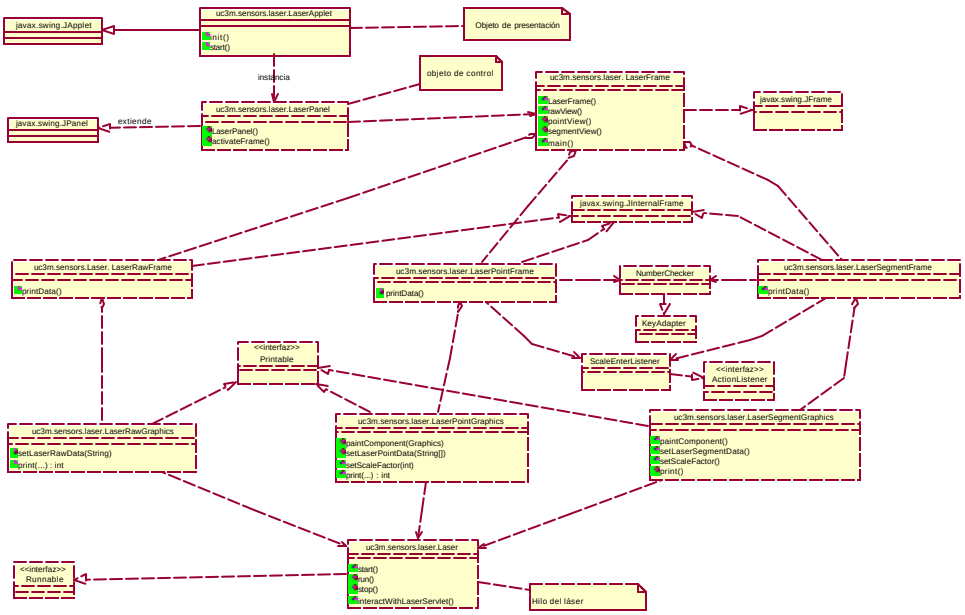


Fig. 7.10: Diagrama de Clases del Panel del Láser

7.2.1.5 Control de Movimiento

El panel de control de movimiento permite al usuario controlar de manera remota, el movimiento del robot utilizando cinco comandos, éstos son avanzar, retroceder, girar en sentido horario, girar en sentido antihorario y parar, tal como se muestra en la figura 7.11. Cada comando de traslación hace que el robot que se mueva 0,5 metros mientras que cada comando de rotación gira el robot 60 grados.

Fig. 7.11: Panel de Control

Como se puede ver más adelante en la subsección 7.2.2.2, el servlet *DriveServlet* es utilizado para interpretar los comandos de movimiento mandados desde el panel de control y para redireccionarlos al servidor de la base del robot. El servidor de la base viene incluido en el software de *Mobility* proporcionado por el fabricante y está escrito en C++ (ver apéndice D). El ORB OmniORB2 se usa para comunicar el servlet con el servidor de la base (ver apéndice C).

Se ha desarrollado otro panel de control de movimiento utilizando la habilidad "Ir a Punto". Este panel da más flexibilidad en el movimiento como se muestra en la figura 7.12. En la sección 7.2.3 se explicará detalladamente la habilidad ir a punto.

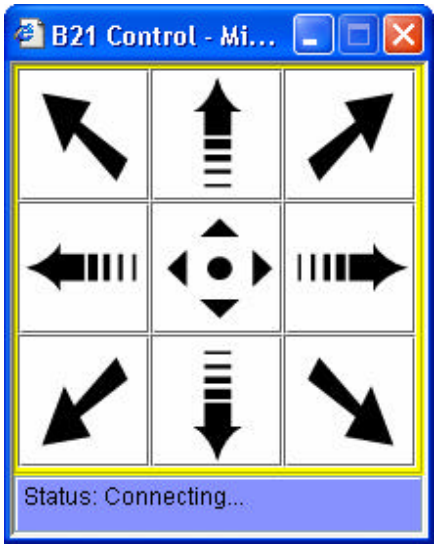


Fig. 7.12: Panel de Control utilizando la habilidad "Ir a Punto"

7.2.1.6 Panel de Ayuda

Se puede considerar este panel como el guión del experimento, que presenta al usuario información general acerca del laboratorio remoto, los objetivos del experimento, el esquema del experimento y los pasos como muestra en la figura 7.13.

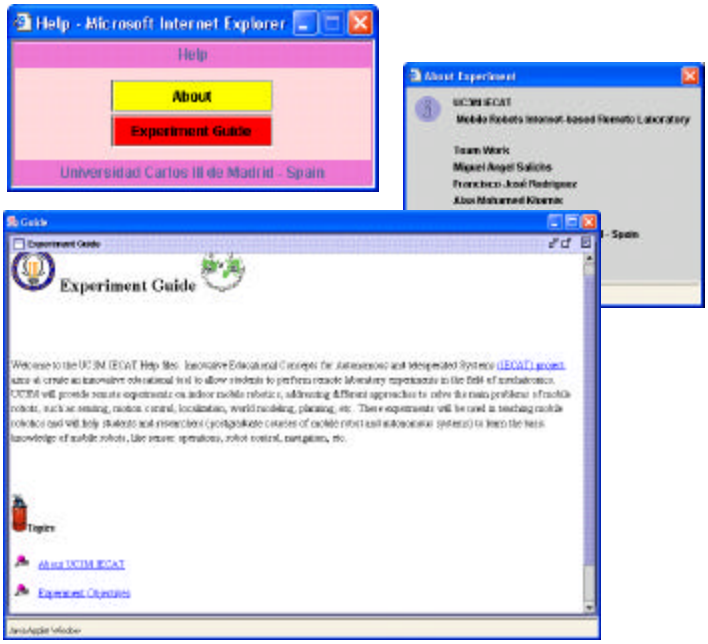


Fig. 7.13: Panel de la Ayuda

Se ha desarrollado este guión con Java y en forma de hipertexto para facilitar el uso. La figura 7.14 muestra las clases que forman el panel de la ayuda.

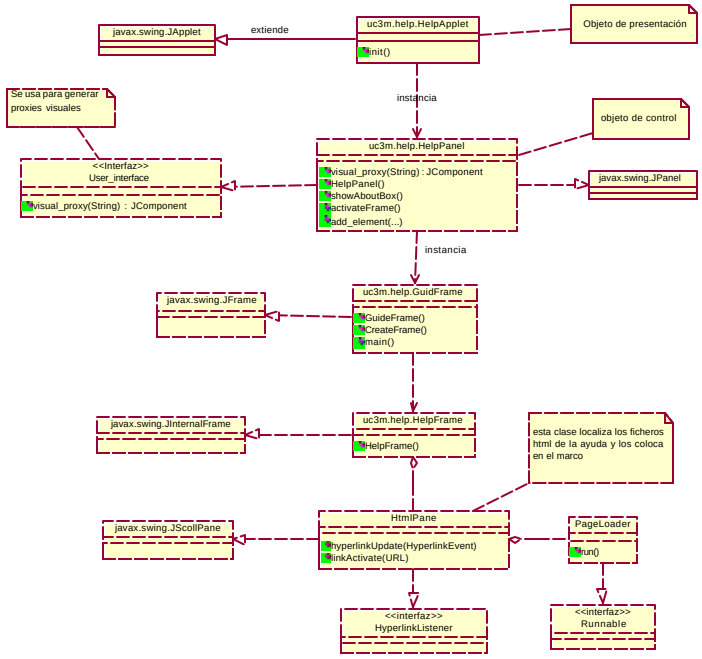


Fig. 7.14: Diagrama de Clases del Panel de la Ayuda

7.2.2 Control Directo

El control directo del robot B21 se considera como una habilidad motora simple que genera comandos de movimiento sin procesar información sensorial. A continuación se presenten los objetivos de este experimento, su desarrollo y las diferentes interfaces de usuario por medio de las cuales se puede utilizar.

7.2.2.1 Objetivos

El objetivo de este experimento es familiarizar al usuario con el control de movimiento de un robot móvil utilizando diferentes elementos de interacción como son los ordenadores personales, las PDAs o los teléfonos móviles. El usuario puede enviar órdenes de control directo para mover el robot B21 hacia adelante, hacia atrás o para girarlo en sentido horario o antihorario. Se provee realimentación visual según el dispositivo usado para controlar el robot para dar al usuario una sensación visual, que expresa la secuencia de sus comandos directamente en la interfaz de control.

7.2.2.2 Teoría

En este experimento, los comandos de movimiento se mandan desde el cliente a través del servlet *DriveServlet* al servidor remoto de la habilidad, que es el servidor de la base del robot. En la parte del cliente se usa el método *interactWithDriveServlet* para abrir la conexión con el servlet de movimiento *DriveServlet*, que recibe como parámetro una letra y en función de cual sea ésta, cambiará la velocidad lineal o angular del robot para controlar su movimiento como se muestra en la figura 7.15.

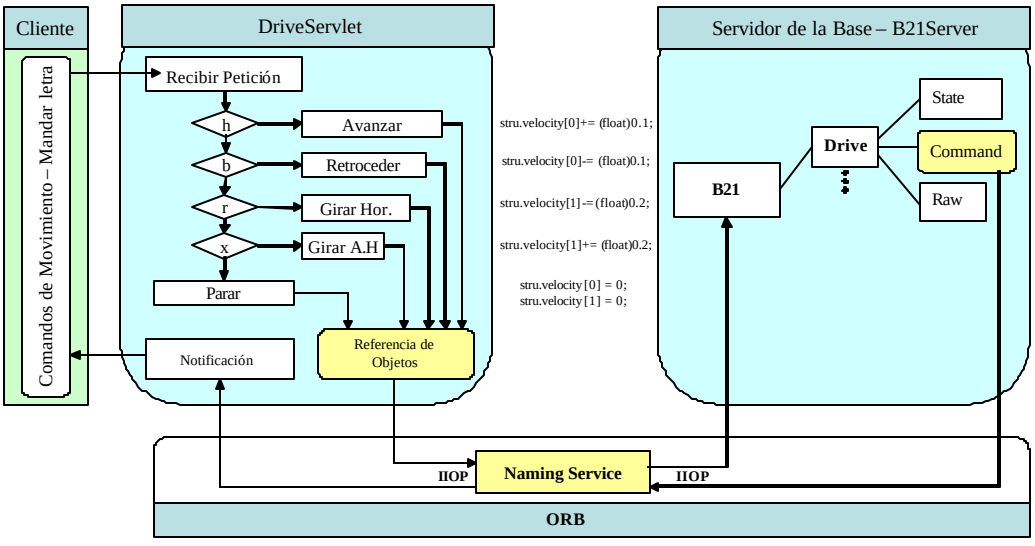


Fig. 7.15: Comunicación Cliente-Servlet-Servidor

El servidor de la base es el encargado del movimiento de la base, así como de la lectura de los sensores propios del robot como son el Sonar, los infrarrojos, y los sensores de contacto. Las órdenes de movimiento se dan mediante clientes CORBA, en este caso es el servlet *DriveServlet*.

Para que estos servidores puedan ser utilizados deben registrarse con el servicio de nombres (*Naming Service*).

El servidor de la base tiene un sistema de seguridad denominado *"perro guardián"* (*watchdog*). Este sistema, en el caso que se perdiera la comunicación con el robot, evita que el robot continúe moviéndose hasta producirse una colisión que dañará a los equipos o a las personas. Si el robot no recibe un comando de velocidad en unos segundos se para automáticamente a la espera de otra orden. En la figura 7.16 se puede observar el servidor funcionando, la indicación del estado de las baterías y la acción del "*Watchdog*".

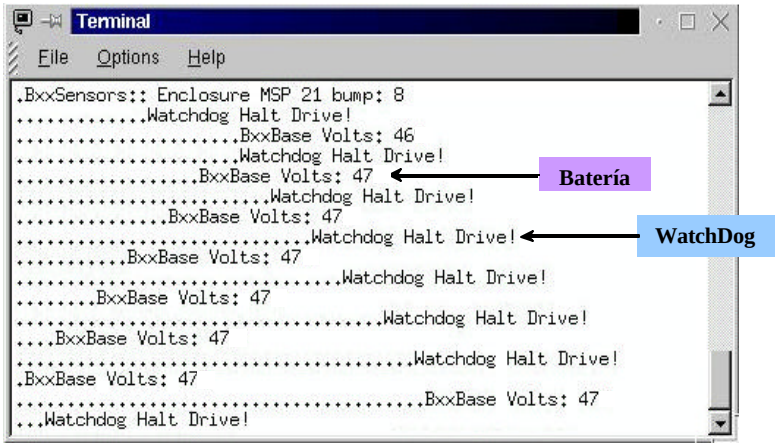


Fig. 7.16: Captura del servidor de la base funcionando

7.2.2.3 Cliente

- **PC**

Esta interfaz está formada utilizando cuatro módulos reutilizables, éstos son: un panel de control de movimiento, un panel de odometría, un panel de estado de conexión y un panel de modelo 2D del aula 1_3C13 donde se encuentra el robot. Se ha agregado un panel de tiempo de respuesta, que se encarga de mostrar el retraso temporal entre el envío del comando de control y el comienzo del movimiento y de dibujar las ultimas diez lecturas de este retraso como se muestra en la figura 7.17.

El área de trabajo aparece definida mediante una línea discontinua que delimita la zona en la que se permite el movimiento del robot. No se permitirá el movimiento del robot, si el usuario pretende enviar el robot a una zona fuera del área de trabajo. Esto se indicará mediante un mensaje de advertencia.

El usuario será capaz de ver el efecto de las órdenes enviadas usando la cámara de red instalada en el techo del laboratorio, el modelo 2D y los datos de la odometría (la posición y la orientación actual del robot y su velocidad lineal y angular). Además, el panel del estado de la conexión notifica al usuario de forma continua acerca el estado de la comunicación del sistema. En la figura 7.18 se puede ver el diagrama de secuencia, que resalta la ordenación temporales de los mensajes.

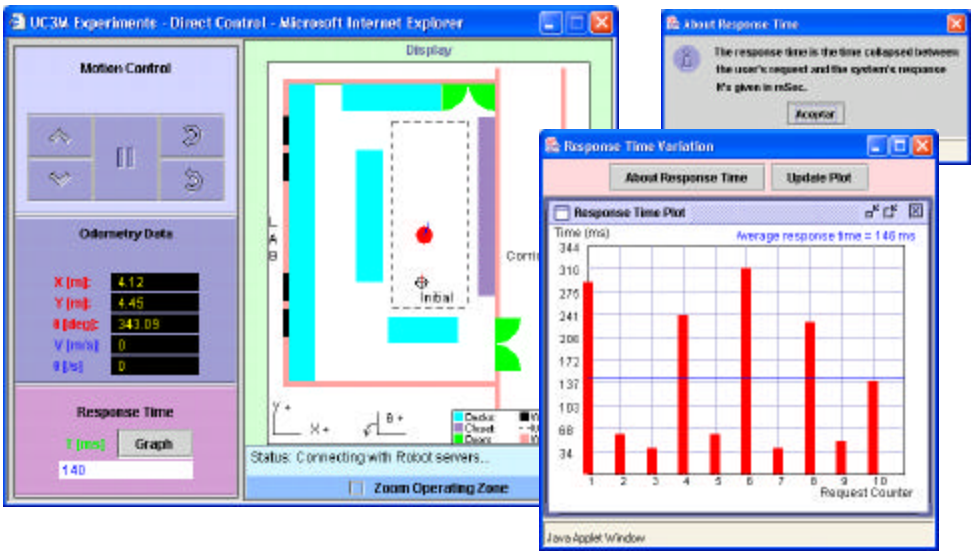


Fig. 7.17: Control Directo

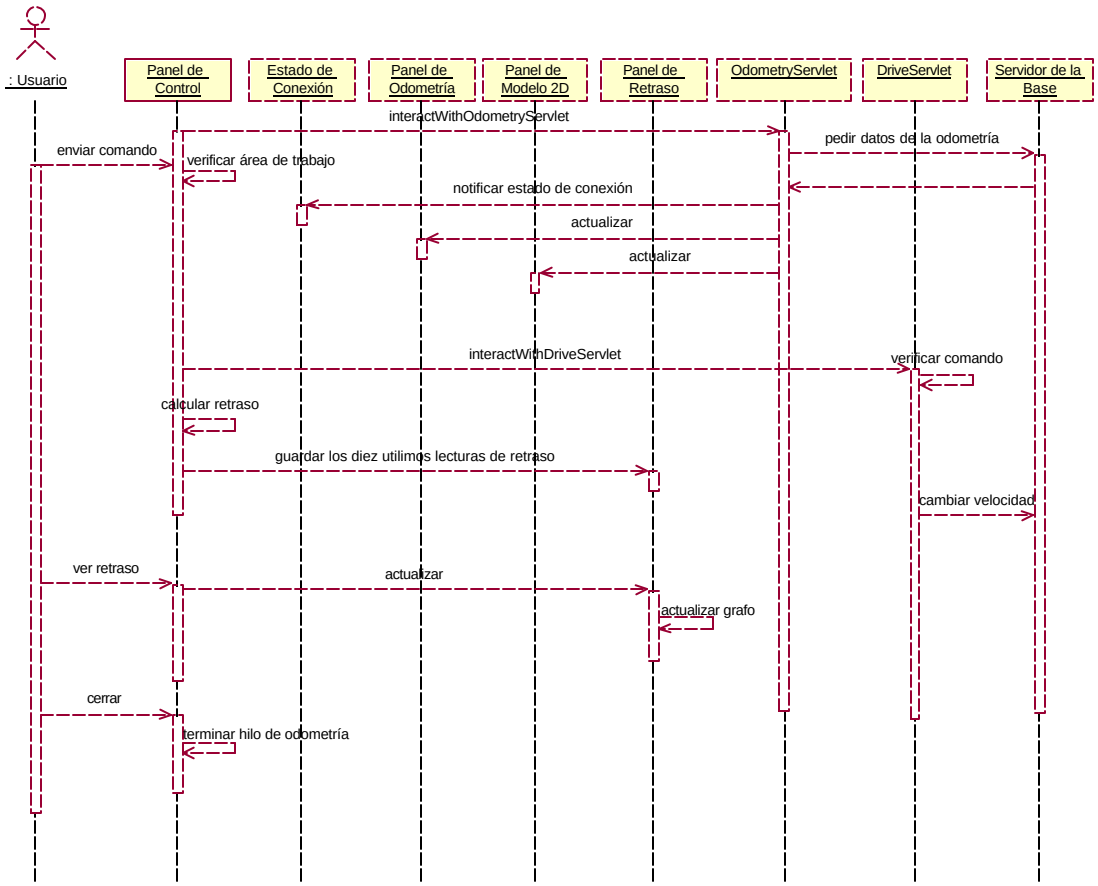


Fig. 7.18: Diagrama de Secuencia

- **Dispositivos Móviles**

Como ya se ha mencionado en el capítulo 2, los dispositivos móviles como las PDAs y los teléfonos móviles presentan la posibilidad de realizar control remoto de los robots móviles con mayor libertad. Estas tecnologías se pueden usar para mejorar el alcance y la funcionalidad de los entornos de teleoperación de robots.

Las posibles aplicaciones incluyen los robots personales y la automatización de la casa. Un robot personal se puede dirigir y controlar por el operador desde cualquier sitio mediante el uso de un dispositivo móvil.

En la figura 7.19 se muestra la interfaz desarrollada para controlar el robot utilizando la PDA Compaq iPAQ 3870. El diseño de esta interfaz es similar al diseño de la interfaz normal con sólo un cambio, que es usar AWT de Java en vez de Swing. Como se ha mencionado en el capítulo anterior, hace falta instalar JeodeRuntim en la PDA para poder utilizar la interfaz (ver apéndice A). La interfaz proporciona un panel de odometría como realimentación visual y un estado de conexión, que muestra el estado de la comunicación del sistema.

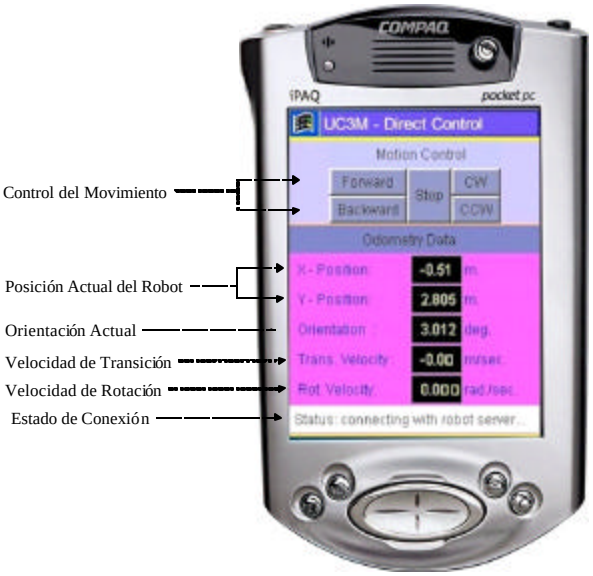


Fig. 7.19: Control Directo con PDA

Otra alternativa es utilizar las interfaces de los teléfonos móviles para interactuar remotamente con el robot. Se han desarrollado interfaces para varias habilidades de movimiento como la habilidad motora "control directo", la habilidad sensorimotora "girar" y la habilidad compleja *"Round Trip"* utilizando la versión micro de Java J2ME. Se ha estudiado también el uso de la tecnología WAP para el desarrollo de aplicaciones inalámbricas (ver apéndice A).

- Versión J2ME

La interfaz de la habilidad "control directo" permite utilizar un teléfono móvil Nokia (en este caso se ha utilizando el emulador Nokia 6310i) para controlar el movimiento del robot a través de 5 comandos, éstos son: 2 comandos de traslación, 2 de rotación y uno para parar tal como se muestra en la figura 7.20.

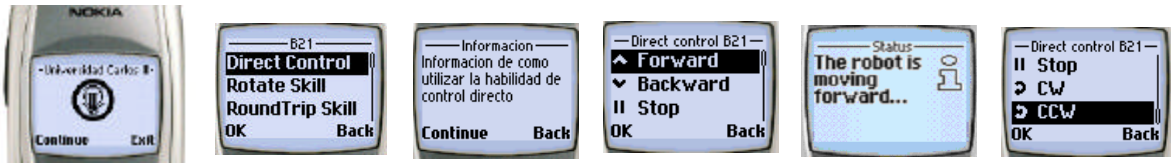


Fig. 7.20: Control Directo con Nokia 6310i

Cada vez que se activa un comando de traslación, el robot se desplaza una distancia fija de 0,5 m, mientras que si se activa un comando de rotación, el robot gira un ángulo fijo de 60 grados. El comando de paro se emplea para detener el desplazamiento o el giro del robot.

-Versión WML

La interfaz WML tiene el mismo funcionamiento que en J2ME, lo único que se diferencian, es en la apariencia de la interfaz como se puede ver en la figura 7.21.

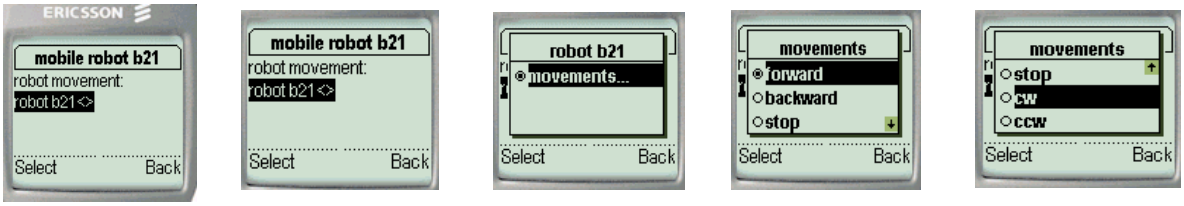


Fig. 7.21: Control Directo con Ericsson

Existen varios factores que favorecen el uso de la tecnología Java Micro para desarrollar aplicaciones para dispositivos móviles (ver Apéndice A). Por eso, se ha concentrado más en el uso de Java Micro para desarrollar interfaces de las habilidades como se verá en las secciones 7.2.5 y 7.2.7.

7.2.3 Habilidad "Ir a Punto"

La habilidad "ir a un punto" es una habilidad sensorimotora que se usa para enviar el robot a un punto concreto en su entorno. Estima la velocidad a la que el robot debe desplazarse para ir en línea recta hacia un punto determinado sin preocuparse de los obstáculos que haya por el camino. Como datos de entrada tiene los datos que le proporciona los sensores de odometría a través del servidor de la base. Los datos de salida son la velocidad angular y lineal a la que el robot debe de moverse [Boada,02-a]. La interfaz de esta habilidad permite al usuario ajustar los parámetros de la habilidad y activar o desactivar dicha habilidad remotamente.

7.2.3.1 Teoría

Durante la activación de dicha habilidad los parámetros que recibe son la distancia a la que se encuentra la meta (respecto a la posición actual de robot, y en coordenadas del robot), el error dentro del cual se considera que se ha alcanzado el destino, la velocidad máxima a la que puede desplazarse, y por último, si las velocidades obtenidas son enviadas directamente a los actuadores o son almacenadas en el objeto de datos para poder ser utilizadas por habilidades más complejas

Como se muestra en la figura 7.22, conociendo la distancia a la que se debe desplazar el robot, y la posición actual (en coordenadas del robot) el objeto activo calcula el rumbo que ha de seguir (? Rumbo).

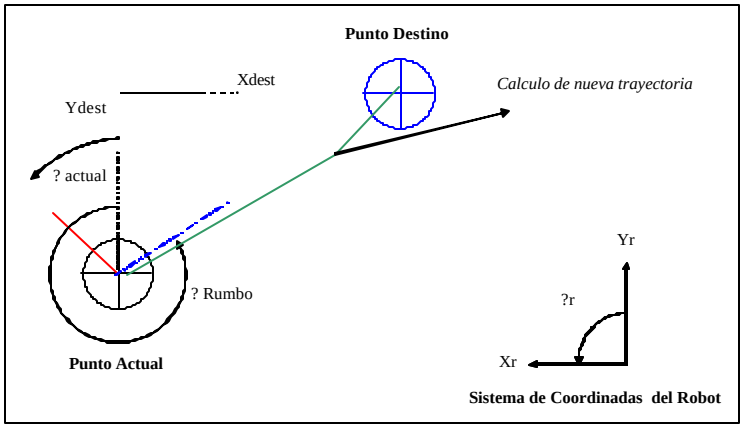


Fig. 7.22: Cálculo de la trayectoria y Proceso de acercamiento

Para alcanzar el destino, primero se orienta el robot actuando sobre la velocidad angular y luego se aproxima actuando sobre la velocidad de translación. Iterando este procedimiento se alcanza el destino [Blanco,01].

La habilidad no solo almacena en sus objetos de datos la velocidad a la que el robot se encuentra, sino también la posición en la que se encuentra con respecto a la meta. Cuando el robot ha alcanzado el punto objetivo, la habilidad genera el evento: PUNTO_ALCANZADO como se puede ver en la figura 7.23.

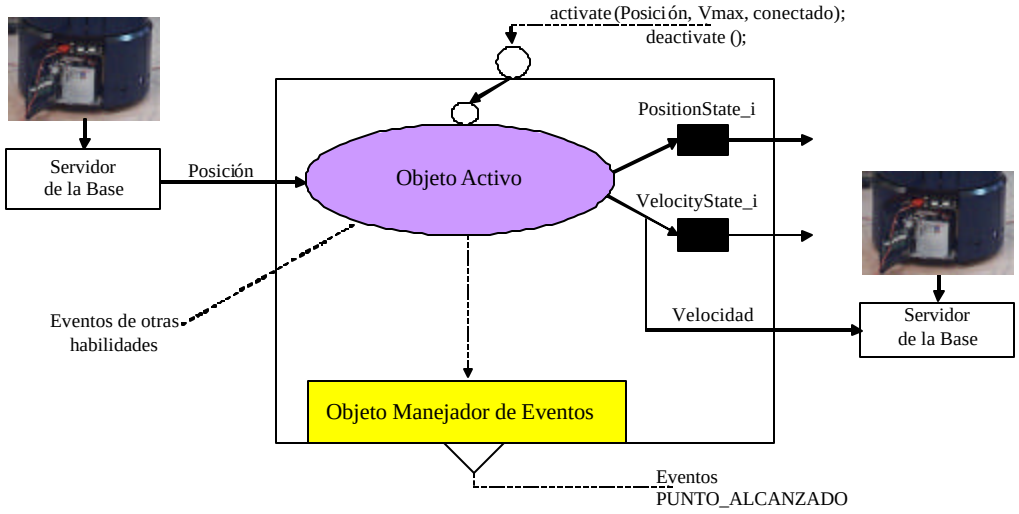


Fig. 7.23: Estructura de la habilidad "Ir a un Punto"

Para el funcionamiento de esta habilidad, hace falta arrancar los dos servidores siguientes:

- **Servidor de la base**

Como se ha mencionado anteriormente, el servidor de la base es el encargado del movimiento de la base, así como de la lectura de los sensores propios del robot como son el sonar, los infrarrojos, y los sensores de contacto. Las órdenes de movimiento se dan a través de clientes CORBA. Para que estos servidores puedan ser utilizados deben registrarse en el servicio de nombres.

- **Servidor de "Ir a un Punto"**

El servidor de ir a punto ha sido desarrollado en el Departamento de Ingeniería de Sistemas y Automática de la Universidad Carlos III de Madrid. Está desarrollado en C++, y de manera similar a los servidores implementados por el fabricante del Robot, utilizan también el mismo ORB (OmniORB2) [Boada,02-a]. La figura 7.24 muestra una captura de pantalla del servidor ir a punto funcionando.

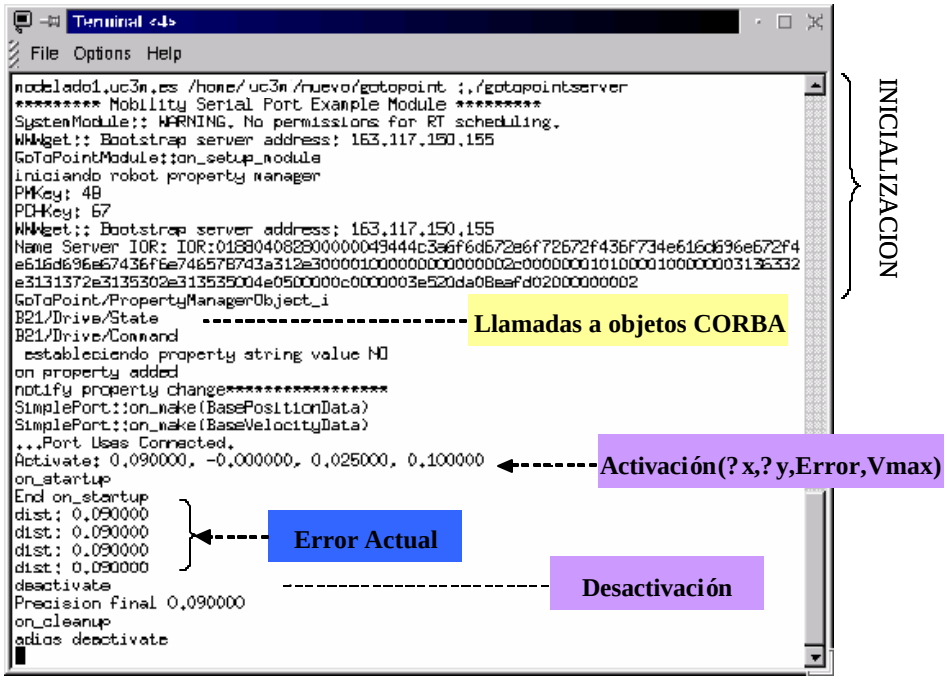


Fig. 7.24: Captura del servidor "Ir a un Punto" funcionando

Este servidor es el encargado de proporcionar al robot una habilidad automática simple como es la acción de dirigirse a un punto. Con los datos de la posición actual del robot que provee la odometría, con los datos del desplazamientos que se pasan en la activación de la habilidad y actuando sobre los motores se consigue controlar la habilidad. Para invocar la odometría y actuar sobre los motores se establecen conexiones con los siguientes objetos CORBA del servidor de la base:

- *B21/Drive/State*: objeto con el que se tiene acceso a los datos de la odometría del robot.
- *B21/Drive/Command*: objeto con el que se controla el movimiento del robot, estableciendo los parámetros que controlan los servomotores.

7.2.3.2 Cliente

Como se puede ver en la figura 7.25, la interfaz está desarrollada en forma de applet de Java que se ejecuta dentro de cualquier navegador.

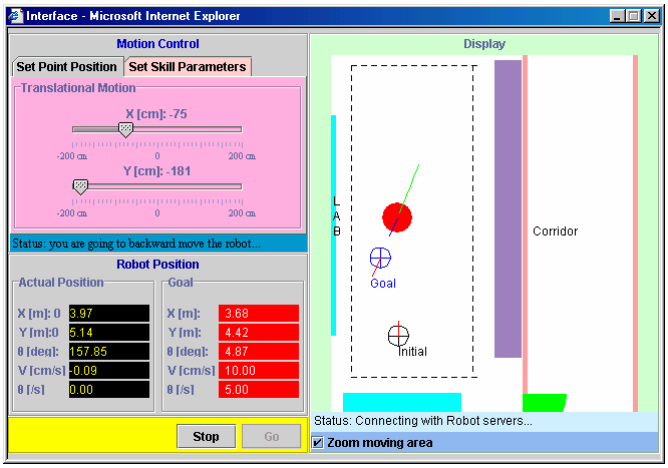


Fig. 7.25: La interfaz de habilidad "Ir a un Punto"

El diseño de esta interfaz está basado en la arquitectura de *proxy* visual. Se muestra en la figura 7.26, el diagrama de las clases que forman dicha interfaz.

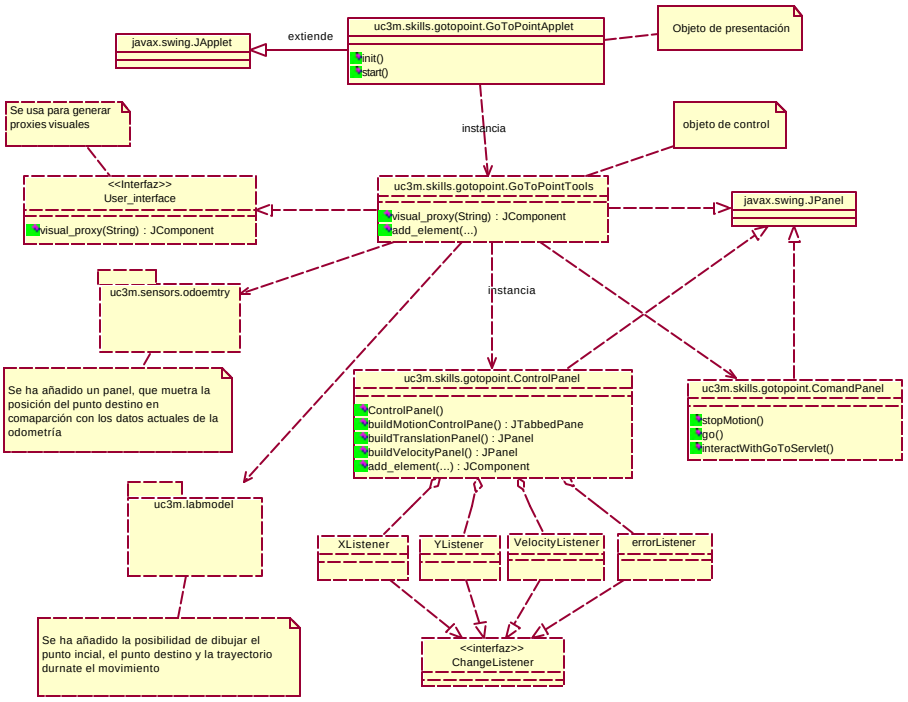


Fig. 7.26: Diagrama de Clases de La interfaz de habilidad "Ir a un Punto"

Se instancian las distintas herramientas invocando los *proxies* visuales de éstas, que son paneles derivadas de la clase JPanel. Estos paneles actúan a la vez como contenedores y

componentes. Contenedores al poder alojar otros componentes de java y componentes, puesto que los paneles pueden ser insertados dentro de los applets.

Para la distribución de los componentes dentro de sus contenedores no se han usado posiciones fijas, sino que están situados a través de una disposición controlada *(layouts)*. A continuación se describe brevemente la función de cada panel:

- *ControlPanel*, panel que permite seleccionar los parámetros de la habilidad.
- *OdometryPanel*, panel que muestra todos los datos de la odometría y posición del punto de destino.
- *DisplayPanel*, panel que muestra gráficamente la disposición del laboratorio y muestra el punto donde se encuentra el robot y su trayectoria durante el movimiento.
- *CommandPanel*, panel que activa y desactiva la habilidad.

7.2.3.3 Servlet

El servlet de "Ir a Punto" es el encargado de activar y desactivar la habilidad de ir a punto, para ello el panel de órdenes (*CommandPanel*) recoge todos los datos necesarios de las distintas herramientas del experimento y se las hace llegar al servlet de la habilidad. Los datos de interés que tiene que recibe el servlet son:

- Distancia a desplazarse, tanto en el eje X como en el eje Y del sistema de coordenadas del laboratorio.
- Error con el que se debe alcanzar el destino.
- Velocidad máxima durante el desplazamiento.
- Acción a realizar, moverse o parar.

Los métodos que permite el servidor de "Ir a Punto" son:

- long activate (in string x, in string y, in string error, in string vmax, in string conex), método para la activación de la habilidad.
- void deactivate(), método para la desactivación de la habilidad.

7.2.4 Habilidad "Ir a Punto Detectando Obstáculos"

Se ha desarrollado una habilidad compleja, que se puede usar para enviar el robot a un punto concreto de su entorno con la posibilidad de detectar obstáculos durante el movimiento. Dicha habilidad consta de dos habilidades simples, la habilidad sensorimotora "Ir a Punto" y la habilidad perceptiva "Detectar Obstáculos". Como se muestra en la figura 7.27, la habilidad "Detectar Obstáculos" se encarga de invocar la información sensorial bien del sonar o bien del láser, para determinar posteriormente si se encuentra presente algún obstáculo.

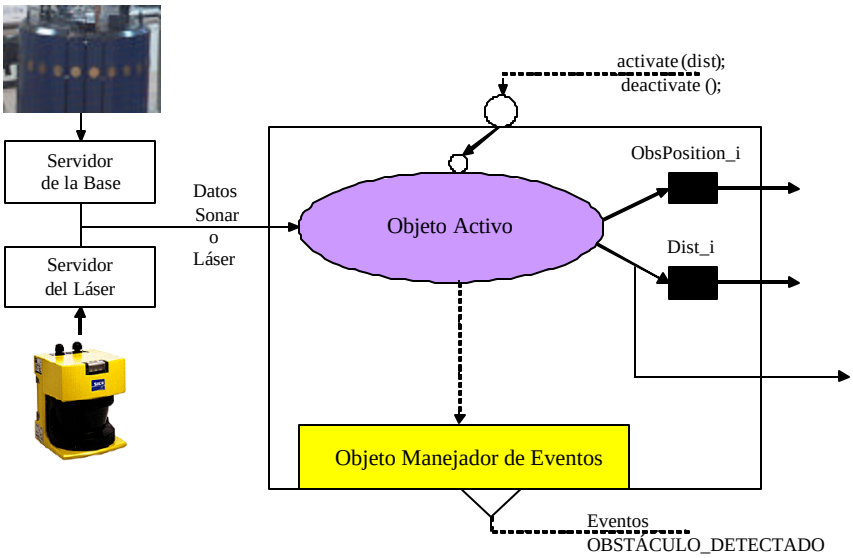


Fig. 7.27: Estructura de la habilidad "Detectar Obstáculos"

Como se puede ver en la figura 7.28, se interpreta como obstáculo cualquier objeto que se encuentre delante del robot, puesto que el robot no hace uso, en sus habilidades, de la marcha atrás.

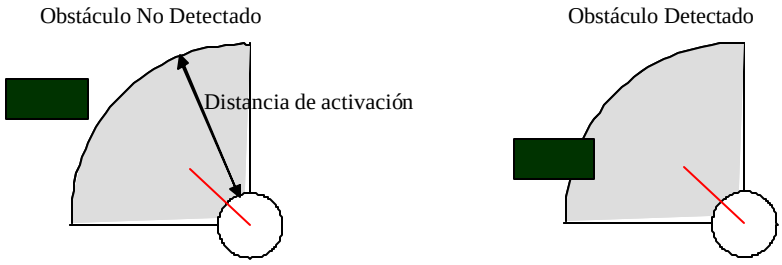


Fig. 7.28: Detectar los Obstáculos

Es por tanto una habilidad puramente sensitiva. La habilidad tiene como parámetro a definir la distancia a la cual el robot considera un objeto como obstáculo. Si los sensores utilizados para detectar un obstáculo detectan la presencia de un objeto, se generarán el evento OBSTACULO_DETECTADO.

El Servidor de "Buscar Obstáculo" forma parte del conjunto de habilidades simples automáticas, y al igual que el servidor de "Ir a Punto" ha sido desarrollado en el Departamento de Ingeniería de Sistemas y Automática [Blanco,01]. También está desarrollado en C++ al igual que los anteriores.

El servidor de encontrar obstáculo se encarga de muestrear continuamente la información sensorial en busca de obstáculos. El robot interpreta como obstáculo cualquier objeto que se encuentre al frente del robot dentro de la distancia seleccionada, cuando se activa la habilidad.

Esta habilidad está implementada de dos maneras. La primera hace uso de los datos que le proporciona el sonar y una segunda que hace uso de la información que le proporciona el láser. Los objetos que facilitan esto son en uno y otro caso:

- B21/Sonar/Raw, objeto con el que se tiene acceso al sonar.
- Laser/Ranger/Raw, para tener acceso al láser.

La figura 7.29 muestra el servidor de "Buscar Obstáculos-Sonar" funcionando.

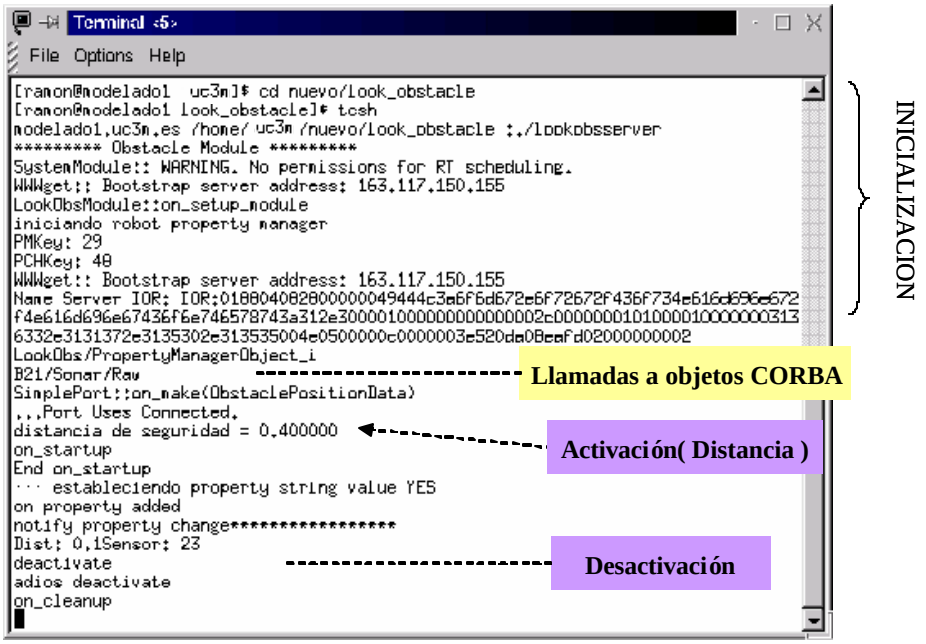


Fig. 7.29: Captura del servidor "Buscar Obstáculo" funcionando

7.2.4.1 Cliente

Combinando ambas habilidades (Ir a Punto y Detectar Obstáculos) en la interfaz gráfica se puede conseguir una habilidad automática compleja, en la que ahora la habilidad de ir a punto, ya es capaz de desactivarse si encuentra algún obstáculo en su trayectoria. Tan solo hay que invocar el método deactivate() de la habilidad "Ir a un Punto", cuando se genere el evento OBSTÁCULO_DETECTADO.

A la interfaz de la habilidad "Ir a Punto", se ha añadido un panel de detección de obstáculos y la posibilidad de dibujar los obstáculos detectados en el modelo 2D del laboratorio. Como se puede ver en la figura 7.30, el panel de buscar obstáculos permite controlar la activación de la habilidad de detección de obstáculos. Permite seleccionar la distancia a la que se buscan los obstáculos mediante la barra deslizante que incluye. Controlar la activación y desactivación de la habilidad cuando se ejecuta la habilida "Ir a Punto". Y por último informa si se ha detectado o no obstáculos.

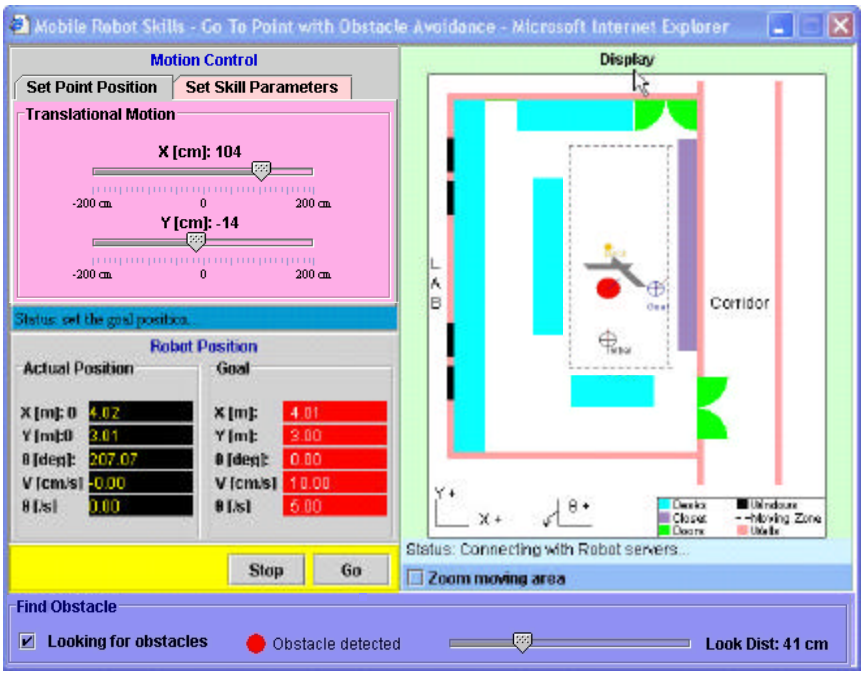


Fig. 7.30: Habilidad Ir a Punto Detectando Obstáculos

Los obstáculos detectados al contrario que los obstáculos fijos no se observan desde el principio. Estos obstáculos solo son representados en caso de que se active la habilidad de detección de obstáculos, y sean detectados por el robot. Una vez activada la habilidad de encontrar obstáculos se van guardando los obstáculos detectados para su posterior representación.

Cuando se inicia el applet se arrancan dos hilos mediante el método *"start"*. Estos dos hilos son el hilo de la odometría que está continuamente leyendo la información que envía el servlet de la odometría, que a su vez invoca la información sensorial que proporciona el servidor de la base. El otro hilo es el hilo de buscar obstáculo que está continuamente enviando y recibiendo información para que el servlet de encontrar obstáculos gestione el servidor de encontrar obstáculos.

La secuenciación de habilidades en el lado del cliente siempre conlleva un riesgo en la elaboración de laboratorios remotos para robótica móvil, debido al retraso temporal que introduce Internet. Por ello se recomienda la utilización de habilidades complejas generadas por un secuenciador como se ha mencionado en el capítulo anterior y como se puede ver en el caso de la habilidad *"Round Trip"*.

7.2.4.2 Servlet

Con el servlet de "Ir a Punto", se usa el servlet de "Buscar Obstáculos" para el funcionamiento de esta habilidad. Este servlet es el encargado de controlar el servidor de la habilidad de encontrar obstáculos, para ello también se sincroniza con el hilo de encontrar obstáculo. El servlet recibe del hilo de buscar obstáculos, si la habilidad se encuentra activada o no, y la distancia de activación del evento de encontrar obstáculo. El servlet activa la habilidad cuando detecta que la variable que guarda el estado de activación/desactivación de la habilidad pasa de desactivado a activado, y la desactiva cuando se produce el caso contrario. Cuando no

hay cambios en el estado y esta se encuentra activada busca obstáculos. Los métodos que permite el servidor de la habilidad de "Buscar Obstáculo" son:

- long activate(in string dist), que activa la habilidad con la distancia de búsqueda.
- void deactivate (), que desactiva la habilidad
- long get_obstacle (), que regresa un datos que indica si se ha encontrado un obstáculo (0 – No se encontró, 1 – Se encontró obstáculo).

7.2.5 Habilidad "Girar"

La habilidad de girar es una habilidad sensorimotora que se puede usar para controlar la orientación del robot. El funcionamiento de esta habilidad se basa en la minimización del error o diferencia entre la orientación que se debe alcanzar según se indica, es decir, el ángulo destino y la orientación actual del robot.

7.2.5.1 Teoría

En esta habilidad sólo se utilizan los sensores internos (odometría) del robot para conocer el rumbo referido al sistema inicial [Blanco,01]. Este parámetro representa el ángulo que forma el frontal del robot con el eje X del sistema fijo (ver la figura 7.31).

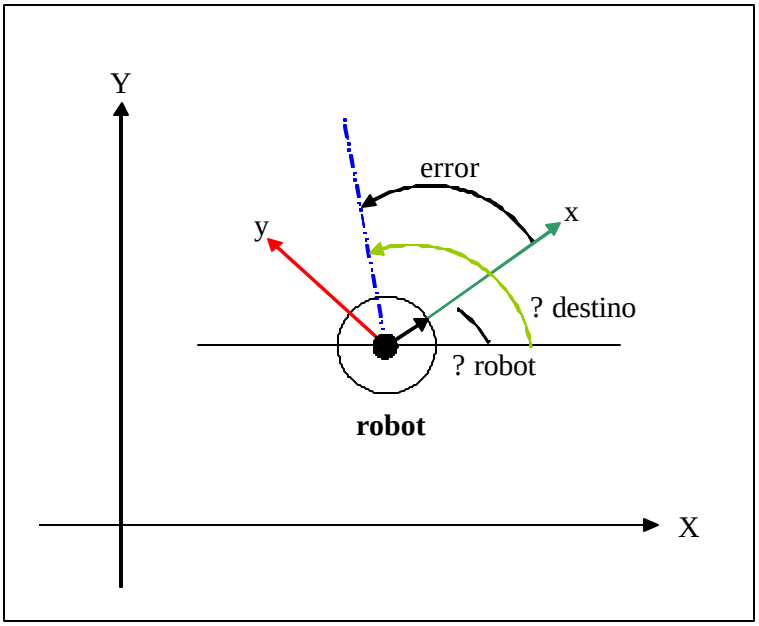


Fig. 7.31: Variables de la Habilidad "Girar"

Las variables a controlar son las velocidades de rotación y de translación, sin embargo, en esta conducta la velocidad de translación se mantiene constante con valor nulo y se ejerce control sólo sobre la de rotación, ya que es esta variable la que influye en el error a minimizar.

$V_{translación} = 0$

La función del regulador que proporciona la rotación en cada ciclo de control tiene la forma siguiente:

$$V_{rotación} = K_p.error + K_d.d_{error}$$ Ecuación 7.6

Siendo:

$$error = q_{destino} - q_{robot}$$ Ecuación 7.7

$$d_{error} = error_i - error_{i-1}$$ Ecuación 7.8

Donde: $error_i$ y $error_{i-1}$ son el error en el instante i y en el instante i-1 respectivamente, las constantes Kp y Kd tienen un valor determinado experimentalmente de forma que la regulación se realice lo más eficiente posible, es decir, el giro se realice con la mayor precisión y en el menor tiempo (mayor velocidad).

7.2.5.2 Cliente

- **PC**

Siguiendo la misma arquitectura de interfaces de usuario, se ha desarrollado una interfaz para la habilidad sensorimora "Girar" para permitir controlar la orientación del robot remotamente (ver figura 7.32).

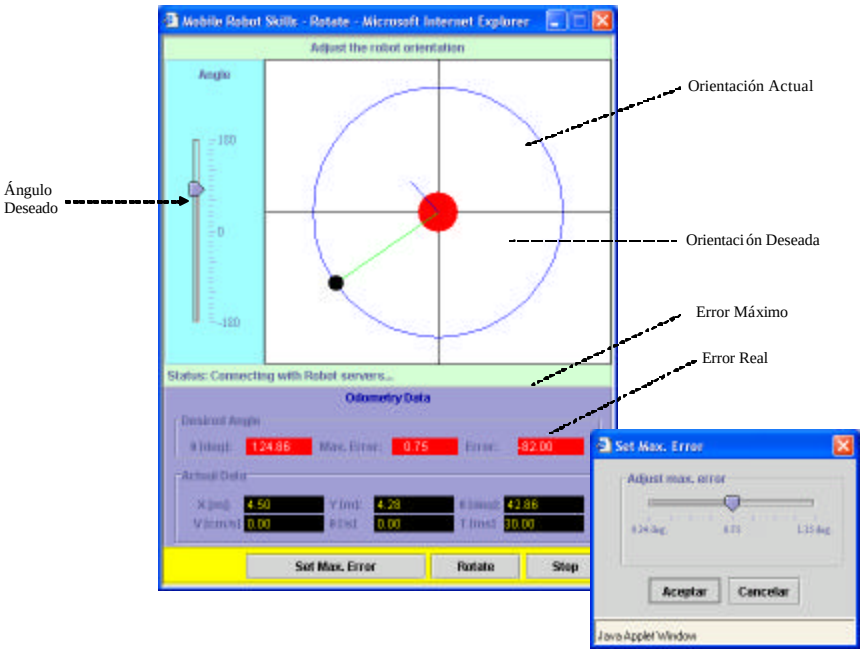


Fig. 7.32: Interfaz de la Habilidad "Girar"

Después de ajustar la orientación del destino y el error máximo de la habilidad, se puede activar la habilidad para cambiar la orientación actual del robot a la orientación deseada. Se compara continuamente en el cliente el valor absoluto de la diferencia entre la orientación actual y la deseada (el error actual) con el error máximo asignado por el usuario. Si el error actual llega a ser igual o menor que el error máximo, se desactiva la habilidad automáticamente.

- **Teléfonos Móviles**

Se ha desarrollado una interfaz con *J2ME* para esta habilidad como se puede observar en la figura 7.33. Dicha interfaz permite rotar el robot un número entero de grados, comprendido entre 0 y 360 grados, si bien la interfaz admite cantidades hasta 999, el robot, si se superan los 360 grados, lo que hace es descontar dicha cantidad, por ejemplo, si se introduce 400, solo se girará 40.

Fig. 7.33: Interfaz J2ME de Habilidad "Girar"

Además se puede seleccionar el sentido de giro, con los comandos CW(rotación horaria) y CCW (rotación horaria). La selección del número de grados y el sentido de giro se hace previamente a la activación de la habilidad con el comando *Activate*. El comando *Deactivate*, se puede utilizar para detener la rotación del robot o bien si está detenido para permitir una nueva activación con el comando *Activate*. El comando *Exit* se usa para salir de la interfaz de esta habilidad y volver a la lista de las habilidades.

7.2.5.3 Servlet

Se han implementado dos servlets para esta habilidad. Uno recibe los comandos (activar o desactivar) y los parámetros de la habilidad (ángulo deseado y error máximo) desde un applet y los redirecciona al servidor remoto de la habilidad. El otro servlet se usa para establecer comunicación entre un MIDlet en un teléfono móvil o un emulador con el servidor remoto de la habilidad.

7.2.6 Habilidad "Seguimiento de Contorno"

La habilidad "Seguimiento de Contorno" consiste en hacer que el robot siga el contorno de un obstáculo situado a una distancia determinada. Las entradas que recibe son las lecturas de los sensores de ultrasonidos y las salidas que genera son las velocidades a las que el robot se tiene que desplazar [Boada,02-a]. En concreto, con la velocidad de rotación se hace que el robot se mantenga paralelo al entorno a seguir mientras que la velocidad de traslación hace desplazarse al robot. En la figura 7.34, se puede observar la estructura de esta habilidad.

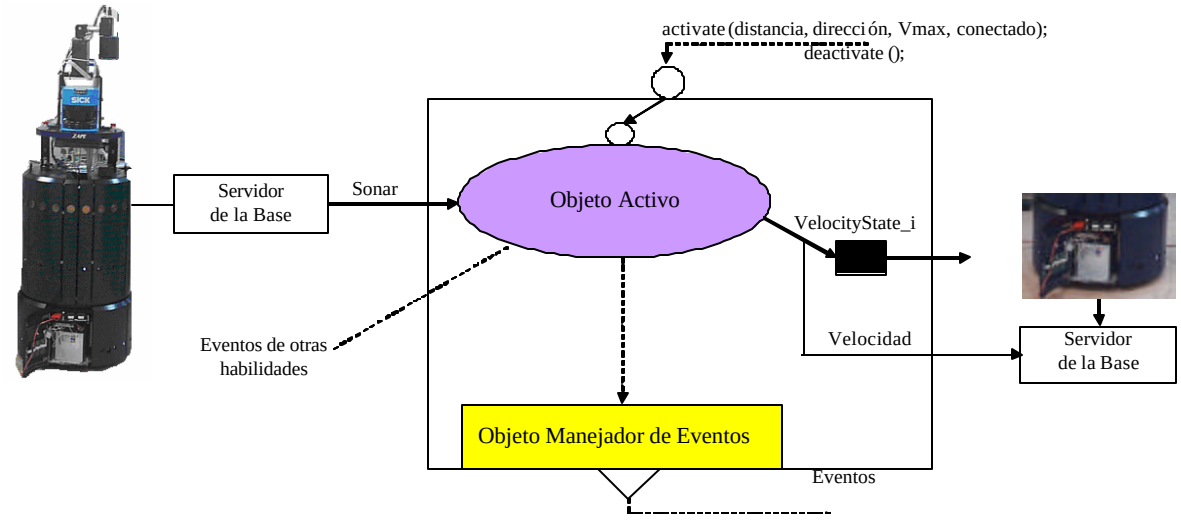


Fig. 7.34: Estructura de la Habilidad "Seguimiento de Contorno"

El servidor de la habilidad de seguimiento de contorno se encarga de dar las órdenes pertinentes de alejamiento o acercamiento al robot (servidor de la base) para adecuar la mínima distancia que le proporciona el servidor de mínima distancia del sonar. Los movimientos de acercamiento o alejamiento se encargarán de hacer girar la trayectoria del robot en el sentido adecuado para que, junto al movimiento constante de translación, de cómo resultado, acercarse o alejarse de la superficie y así conseguir la distancia pedida. La figura 7.35 muestra el servidor de la habilidad funcionando.

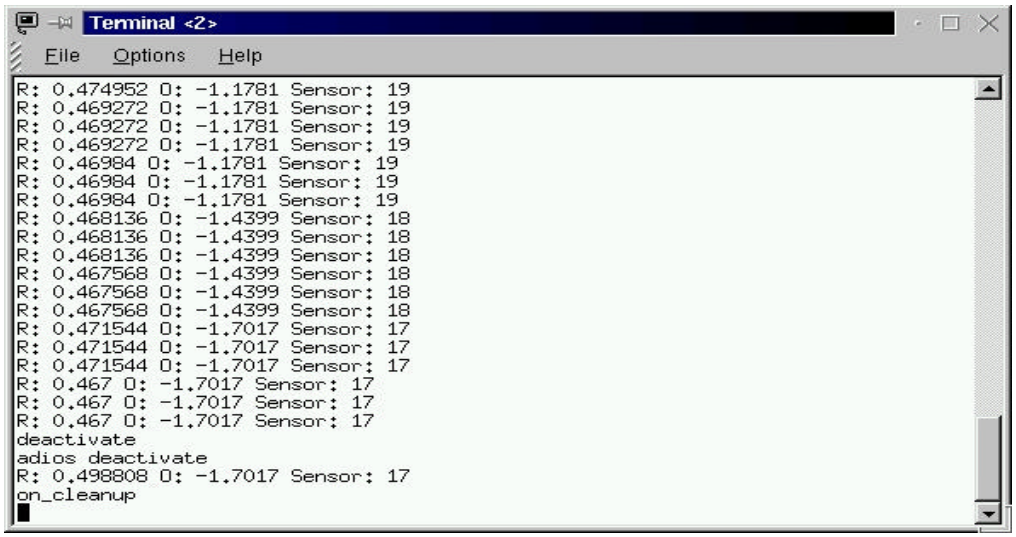


Fig. 7.35: Consola del servidor de la habilidad

El servidor de la habilidad de "Seguimiento de Contorno" necesita del apoyo del servidor de base del robot y del servidor de mínima distancia utilizando sonar. El servidor de mínima distancia se encarga de determinar las distancias mínimas de los objetos ubicados en los alrededores del robot para esto utiliza los datos del sonar que son suministrados por el servidor de

la base. Este servidor se encarga de indicar cuál de los sensores ha detectado la mínima distancia (derecha = sensores 0 a 11 o izquierda = 12 a 23).

El diseño de la capa del cliente de esta habilidad sigue el patrón con el que han sido diseñadas todas las interfaces gráficas de las habilidades del laboratorio remoto.

Como se ha mencionado anteriormente la habilidad de seguimiento de contorno permite al robot seguir una superficie frontera a la distancia que se le indique desde la interfaz. Desde la interfaz (véase la figura 7.36) se marca la distancia a mantener con el contorno, en que lado debe buscar esa superficie y con que velocidad debe realizar el experimento.

Fig. 7.36: Interfaz de la Habilidad "Seguimiento de Contorno"

Izquierda: en el laboratorio Derecha: en el pasillo

Se manda iniciar o detener la habilidad al servidor con los parámetros establecidos. Estos son: la distancia, la dirección a seguir y la velocidad máxima. Las tres opciones de la dirección son izquierda (*Left*), derecha (*Right*) y desconocida (*Unknown*). Dos de ellas son obvias y la tercera es para movimiento libre por el cual el robot seguirá el primer contorno que se encuentre a la distancia definida.

También en la interfaz se podrá comprobar los progresos del robot tanto visualmente, por medio de una representación gráfica del robot, de su entorno y de la trayectoria del movimiento o a través de las imágenes de la cámara. Así mismo, se mantendrá en tiempo real los datos de odometría del robot y el estado de la conexión.

Se puede utilizar esta habilidad en el sistema de navegación topológica EDN (*Event Driven Navigation*) [Barber,00]. Este sistema intenta que el robot vaya de un lugar a otro mediante indicaciones parecidas a las que se harían a un ser humano que desea realizar el recorrido entre esos mismos lugares. Este sistema se base también en utilizar cartas de navegación, que son mapas topológicos formados por unas sucesiones de tareas a realizar durante la navegación.

7.2.7 Habilidad Compleja "*Round Trip*"

Como se ha mencionado en el capítulo anterior, se puede utilizar el agente del secuenciador para combinar habilidades simples y generar habilidades automáticas complejas. El secuenciador ejecuta un plan definido previamente. Este plan se define utilizando un lenguaje libre de contexto, su sintaxis se define utilizando BNF (Backus Naur Form) [Rivero,03]. En la figura

7.37 se puede ver el plan ejecutado por el secuenciar para generar la habilidad automática compleja denominada "Ir y Volver" o *GoBack*.

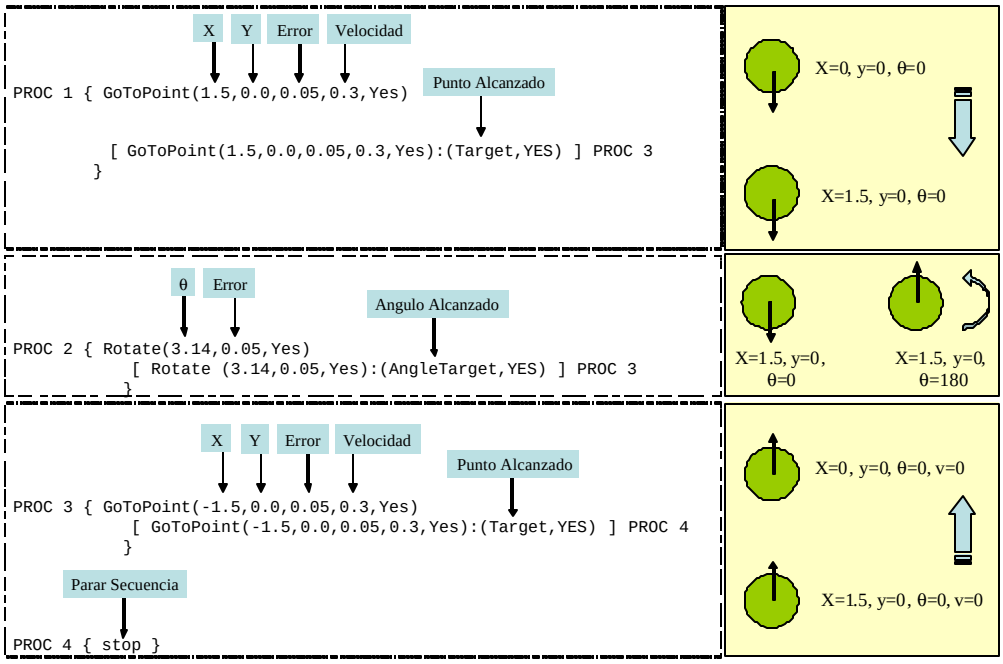


Fig. 7.37: Habilidad Compleja *"GoBack"*

Esta habilidad es una combinación de la habilidad sensorimotora simple "Ir a Punto" y la habilidad "Girar". Se usa para hacer que el robot se desplace una distancia fija (1,5 metros) y luego vuelva al punto inicial. Esta habilidad no recibe ningún parámetro, sólo se activa o desactiva con los parámetros grabados en el plan del secuenciador.

Se ha realizado un cambio en el plan del secuenciador para hacer al robot dar una vuelta en forma de cuadrado (2,5X2,5 metros) como se puede ver en la figura 7.38 y el listado 7.1. Esta habilidad se ha sido denominada *"Round Trip"*.

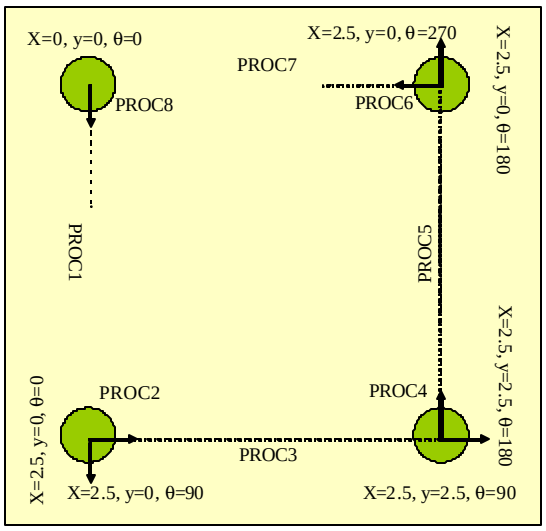


Fig. 7.38: Habilidad Compleja *"Round Trip"*

Como se puede ver en el listado 7.1, finalizando un proceso el secuenciador indicará cual es el siguiente proceso o acción a realizar. El proceso principal procede a buscar el siguiente proceso a ejecutar y a validar si este proceso puede ser activado o debe esperar a la finalización de otro proceso. Si el proceso puede ser ejecutado, se lanzará el nuevo agente secuenciador encargado de ejecutar el proceso. De lo contrario, se quedará en estado de espera. Si el proceso se ejecuta con éxito, y se llega al fin del plan, el estado de ejecución finaliza, retornando al estado inactivo. De lo contrario, si surge algún error durante la ejecución del plan, el estado ejecución es abandonado y se pasa al estado de error.

```
PROC 1 { GoToPoint(2.5,0.0,0.01,0.4,Yes)
            [ GoToPoint(2.5,0.0,0.01,0.4,Yes):(%Target = YES) ] PROC 2
       }
PROC 2 { Rotate(1.57,0.05,Yes)
           [ Rotate (1.57,0.05,Yes):(%AngleTarget=YES) ] PROC 3
       }
PROC 3 { GoToPoint(0.0,2.5,0.01,0.4,Yes)
            [ GoToPoint(0.0,2.5,0.01,0.4,Yes):(%Target=YES ) ] PROC 4
       }
PROC 4 { Rotate(1.57,0.05,Yes)
           [ Rotate (1.57,0.05,Yes):(%AngleTarget=YES) ] PROC 5
       }
PROC 5 { GoToPoint(-2.5,0.0,0.01,0.4,Yes)
            [ GoToPoint(-2.5,0.0,0.01,0.4,Yes):(%Target=YES ) ] PROC 6
       }
PROC 6 { Rotate(1.57,0.05,Yes)
           [ Rotate (1.57,0.05,Yes):(%AngleTarget=YES) ] PROC 7
       }
PROC 7 { GoToPoint(0.0,-2.5,0.01,0.4,Yes)
            [ GoToPoint(0.0,-2.5,0.01,0.4,Yes):(%Target=YES ) ] PROC 8
       }
PROC 8 { stop }
```

Listado 7.1: Secuencia de la Habilidad Compleja *"Round Trip"*

Fig. 7.39: Interfaz de la Habilidad *"RoundTrip"*

La interfaz desarrollada para esta habilidad (véase la figura 7.39) es muy simple y permite activar o desactivar la habilidad debido a que la habilidad no recibe parámetros de activación. Se ha añadido un panel de estado que muestra las acciones realizadas por el usuario. Se pueden utilizar el modelo 2D, el panel de odometría y las imágenes de la cámara como herramientas de realimentación visual.

Se ha desarrollado otra interfaz para activar/desactivar esta habilidad utilizando un emulador de un teléfono móvil. Como se puede ver en la figura 7.40, el comando *Activate* permite la activación de la habilidad y el comando *Deactivate* se puede utilizar para detener el movimiento del robot y reiniciar las variables de la habilidad.

Fig. 7.40: Interfaz J2ME de la Habilidad *"Round Trip"*

Una de las posibles aplicaciones educativas de esta habilidad es usarla para calcular los errores sistemáticos de la odometría del robot utilizando un método como UMBmark (*University of Michigan Benchmark test)* [Borenstein,96].

7.2.8 Habilidad Compleja *"GoToLab"*

Esta habilidad es una combinación de diferentes habilidades sensorimotoras y perceptivas que se puede usar para hacer que el robot se desplace del pasillo al laboratorio 1.3C13 pasando por el laboratorio 1.3C12. Las habilidades usadas para formar esta habilidad compleja son: "Ir a un Punto", "Seguir Pasillo", "Detectar Puerto con el Láser", "Acercar Puerta", "Cruzar Puerta" y "Seguimiento de contorno" (véase la figura 7.41). Esta habilidad también no recibe ningún parámetro, sólo se activa o desactiva con los parámetros grabados en el plan del secuenciador.

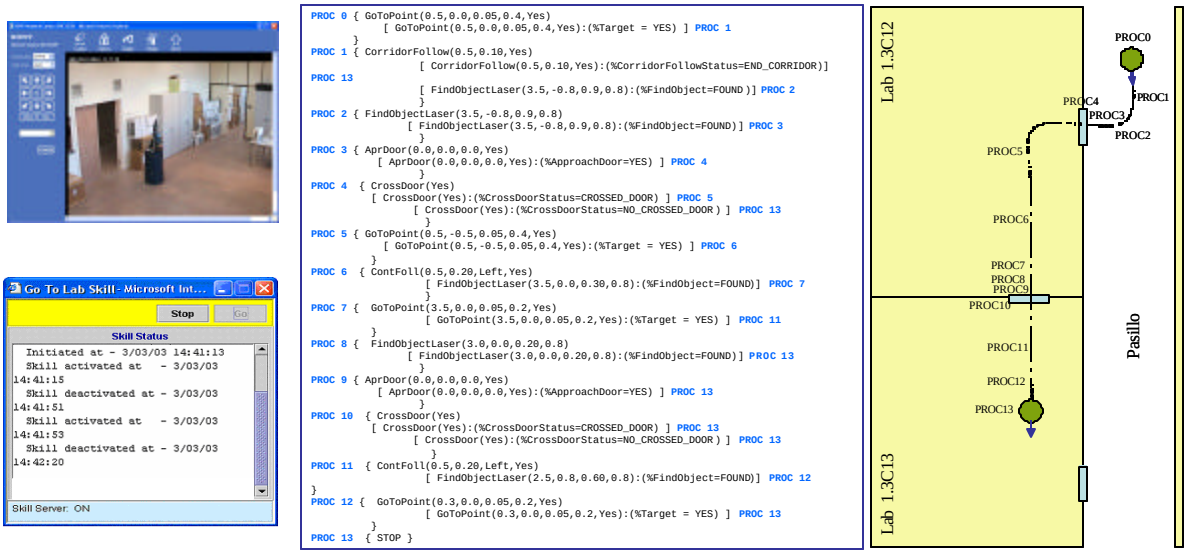


Fig. 7.41: Habilidad Compleja *"GoToLab"*

7.2.9 Percepción del Entorno utilizando Información Sensorial

Para que un robot móvil sea capaz de desplazarse con cierta seguridad dentro de un entorno desconocido, son imprescindibles las lecturas de los sensores que lleva a bordo. Estos sensores suelen ser de varios tipos: ultrasonidos, láser, infrarrojos, visión artificial, etc.

Son necesarios distintos tipos de sensores por el hecho de que cada uno de estos sensores tiene puntos débiles y por tanto si se utiliza un sólo tipo, cuando su punto débil apareciese, el robot móvil recibiría unas lecturas del entorno que no serían las reales y pondría en peligro al entorno que le rodea y a él mismo. A continuación se presentan algunos puntos fuertes y débiles de los diferentes sensores:

- Láser: suele ser caro pero tiene mejor precisión que el sonar.
- Ultrasonidos: son relativamente baratos pero tienen problemas con esquinas y ángulos pronunciados (reflejos en superficies metálicas) y necesitan ser calibrados a menudo.
- Infrarrojos: proporcionan buena precisión en distancias cortas pero las lecturas son sensibles al color del obstáculo y necesitan ser calibrados a menudo (los errores pueden llegar a ser del orden de magnitud de la medida que se está tomando).

7.2.9.1 Objetivos

El objetivo de este experimento es el reconocimiento del entorno utilizando la información sensorial. Se usan el sonar y el láser como fuentes de información y se comparan los resultados y las precisiones. Se pretende que el usuario, tras la realización del experimento, sea capaz de tener una idea del entorno en el que se mueve el robot mediante la lectura de los sensores y el movimiento del robot a lo largo del laboratorio y sea capaz de entender la problemática que lleva consigo el reconocimiento del entorno en robótica móvil.

7.2.9.2 Sonar

Los ultrasonidos se utilizan fundamentalmente como sensores de proximidad o de medida de distancias, por la técnica de eco-detección, permitiendo detectar la presencia de objetos u obstáculos en el entorno del robot. Se emplean 24 sensores, cada uno de ellos abarca un sector angular alrededor del robot móvil de 15 grados de amplitud. De este modo se cubre completamente la zona alrededor del mismo.

La práctica se divide en dos partes, una con el robot parado y otra moviendo el robot. Con la primera parte se pretende que el usuario entienda el significado de la lectura de los sensores en general y en particular del sonar (véase la figura 7.42). Se realizarán varias medidas sin mover el robot y se podrá comprobar que las medidas van cambiando. Se puede relacionar también las lecturas del sensor con los objetos reales del entorno. La segunda parte se refiere al reconocimiento del entorno utilizando varias medidas cambiando el robot de posición.

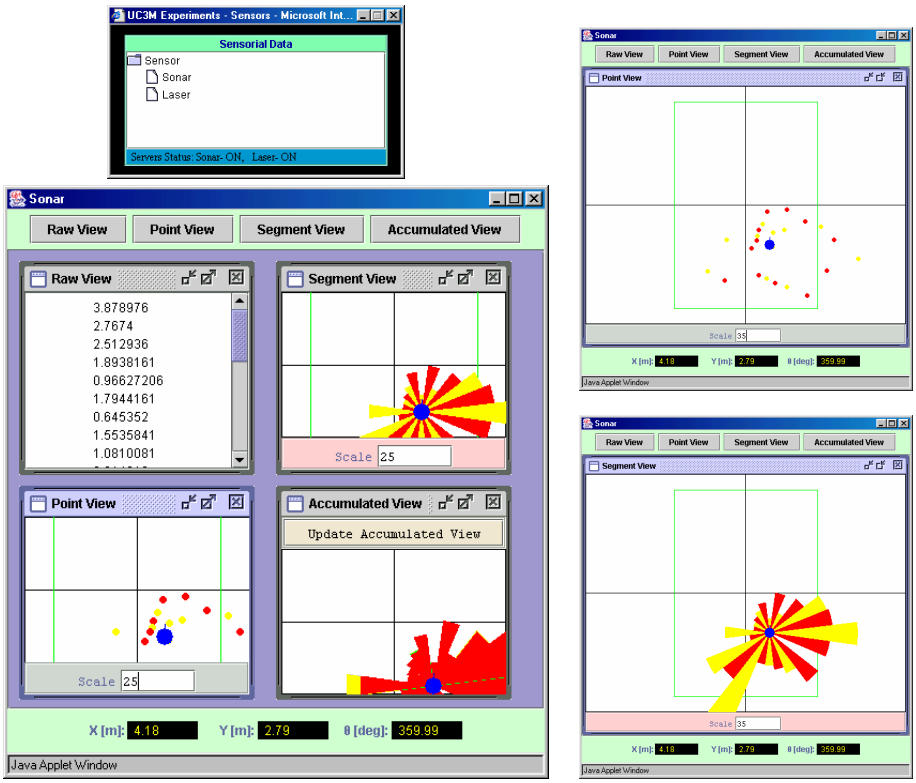


Fig. 7.42: Lectura del Sonar

Como se puede ver en la figura 7.42, algunas medidas se salen de rango esto se debe a los ángulos de incidencia en esquinas y huecos entre armarios, meses, cajas, etc... En este caso el eco nunca llega al receptor y por eso a veces se producen unas medidas correspondientes al máximo valor.

7.2.9.3 Láser

Como se sabe, el láser se basa en la medida del tiempo de vuelo de un haz luminoso devuelto por un objeto (véase la figura 7.43).

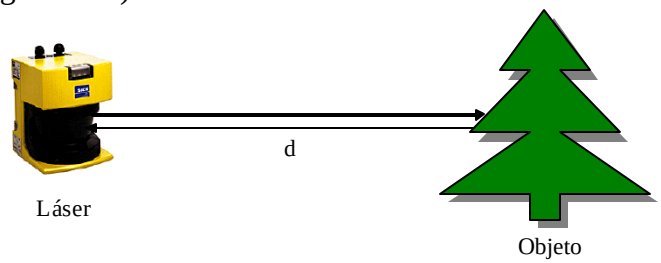


Fig. 7.43: Principio de Funcionamiento de un Láser

No necesita una segunda unidad como receptor, ya que él mismo se encarga de emitir y recibir el haz. Al tratarse de un sistema de detección sin contacto, no existen desgastes [Galeote00]. Se puede calcular la distancia con la siguiente ecuación:

$$d = v.\Delta t/2 \qquad \text{Ecuación 7.9}$$

Donde:

v = velocidad de propagación en el medio

Δt = tiempo de propagación (ida + vuelta)

d = distancia láser-superficie del objeto

En la figura 7.43 se observan los elementos que intervienen en la medida (láser, medio de propagación y objeto). Estos pueden influir sobre las lecturas que se tomen de la siguiente manera. Cualquier error en la superficie del espejo que refleja el haz interiormente influirá, así como las características del medio (como esta práctica se realiza en interiores, puede considerarse que el medio no va a influir) y el tipo de superficie del objeto (si el objeto absorbe parte del haz, la señal que llegue al láser será más débil y provocará errores de medida).

En esta práctica podrá verse como influye la superficie del objeto, con el ejemplo de las ventanas del laboratorio donde se encuentra el robot móvil.

Como en la práctica del sonar, esta práctica se divide en dos partes, toma de datos sin desplazamiento del robot para observar la variabilidad de los datos del sensor láser (véase la figura 7.44) y toma de datos con desplazamiento del robot, que pretende inspeccionar el entorno del robot mediante la información del láser.

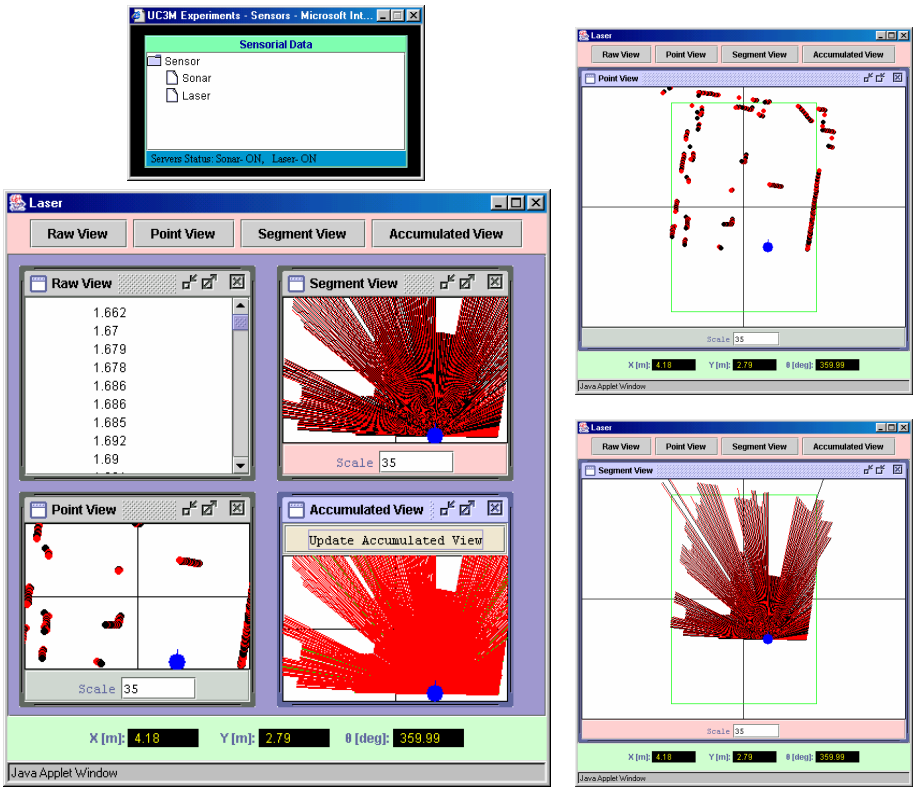


Fig. 7.44: Lectura del Láser

Se puede observar en las figuras de toma de datos (Figura 7.44) como algunos rayos del láser se salen fuera de los bordes del laboratorio. Esto se debe a que el láser tiene problemas con los

objetos transparentes o reflectantes ya que estos no reflejan el haz que emite el láser, o lo reflejan con un ángulo en el que el receptor no recibe el rayo reflejado, y por tanto para el láser esos objetos no existen. En este caso, las ventanas del laboratorio y el extintor son los que provocan estos errores en las lecturas.

Existe otro problema, la acumulación de errores en la odometría debido a errores en los codificadores de posición y deslizamientos en las ruedas que obligan a una localización del robot móvil a menudo.

7.2.9.4 Percepción

Como se puede ver en la figura 7.45, si se superponen cuatro medidas tomadas en distintos sitios a lo largo del laboratorio, se puede apreciar como aparece el contorno del obstáculo que se pretendía identificar. El hueco que queda sin pintar en el centro de las cuatro medidas se corresponde con el espacio ocupado por las cajas. El resto de las medidas, excepto las que se salen de rango, marcan el entorno real formado por las mesas, ordenadores, otros robots en la sala, etc...

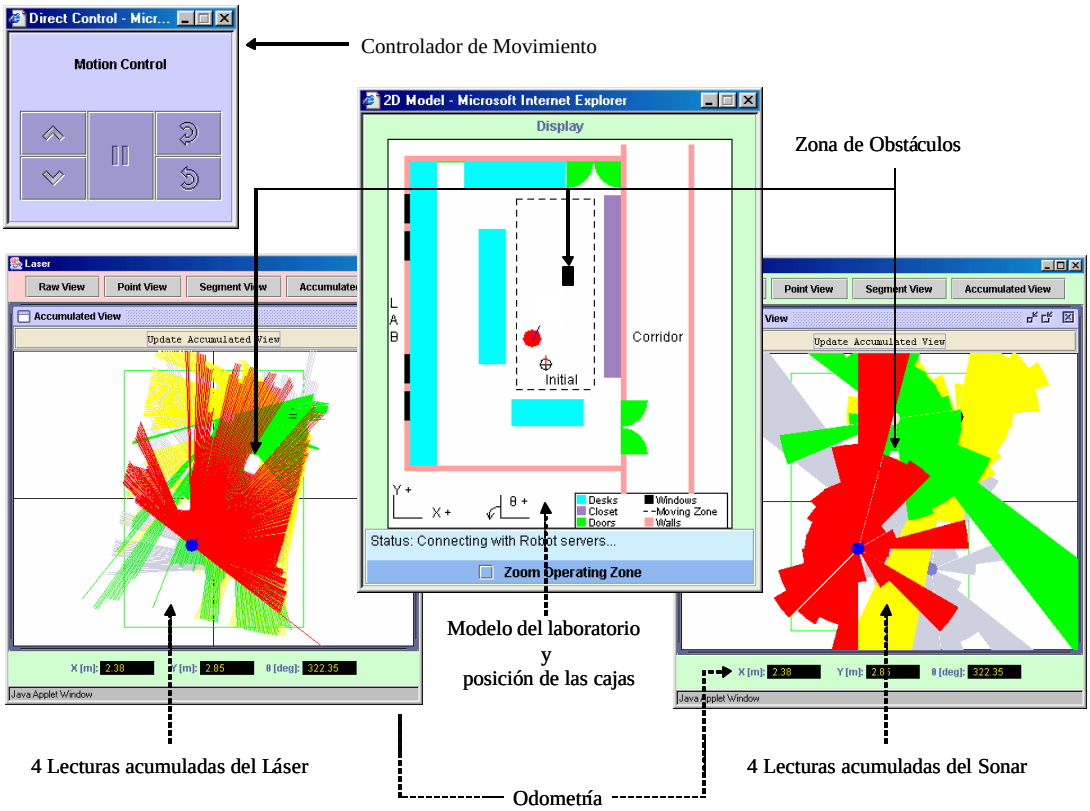


Fig. 7.45: Percepción del Entorno

Se puede observar también que las lecturas del láser proporcionan mejor precisión que las del sonar. Esto demuestra porqué un robot móvil no puede depender exclusivamente de un tipo de sensor. Es necesario que utilicen sensores de distintos tipos para que se complementen entre sí. Además se puede fusionar los datos del sonar con los del láser para obtener datos más fiables y precisos.

7.2.10 Modelado del Entorno

Un robot móvil que posea la capacidad de crear un modelo de su entorno, puede utilizar eficientemente ese modelo en diversas tareas de control complejas. Esta capacidad permite al robot actuar coherentemente con su entorno, ya sea interpretando adecuadamente los datos de los sensores y tomando decisiones a corto plazo, o adquiriendo y manipulando un modelo completo del entorno para realizar control de trayectorias y toma de decisiones a largo plazo.

7.2.10.1 Objetivos

El objetivo de la práctica es adquirir conocimientos de modelado del entorno en robótica móvil, realizando para ello un mapa del laboratorio en el que se halla el robot utilizando dos métodos distintos y basándose en la utilización de la información sensorial del sonar. Se pretende que el estudiante aprenda como se realiza el modelado del entorno mediante estos procedimientos. Así mismo, después de la práctica se podrá comparar los mapas obtenidos mediante ambos métodos, y observar cómo su precisión depende considerablemente del número de iteraciones realizadas.

7.2.10.2 Generación de Modelos

La tarea de generar modelos es compleja y está lejos de estar totalmente resuelta. Los siguientes factores imponen limitaciones prácticas en la habilidad de aprender y usar modelos precisos:

- **Sensores:** Los sensores a menudo no son capaces de ofrecer directamente medidas de interés. Por ejemplo, las cámaras miden color, brillo y saturación, mientras que, en navegación, resultaría mucho más interesante aserciones como “hay una puerta delante del robot”.
- **Limitaciones perceptúales:** El rango perceptual de la mayoría de los sensores (tales como cámaras, láser, ultrasonidos, etc.) está limitado a una pequeña zona alrededor del robot. Para adquirir información de forma global, necesita explorar activamente su entorno.
- **Ruido sensorial:** Las medidas que devuelven los sensores están típicamente mezcladas con ruido. Normalmente, la distribución de este ruido es desconocida (y rara vez es Gausiana).
- **Deslizamientos:** El movimiento del robot es poco preciso. Desafortunadamente, los errores de odometría se acumulan con el tiempo.
- **Entornos complejos y dinámicos:** El entorno del robot es normalmente complejo y cambia constantemente, haciendo imposible el mantener modelos adecuados del mismo.
- **Requerimientos en tiempo real:** Los requerimientos en tiempo real a menudo requieren que los modelos generados sean simples y accesibles. Por ejemplo, un modelo preciso de un entorno interior complejo puede ser una desventaja si el robot ha de actuar rápidamente.

Las investigaciones en tareas de mapeado han producido tres paradigmas fundamentales: el paradigma geométrico, el paradigma topológico y el paradigma topo-geométrico. Cada uno tiene sus ventajas e inconvenientes. Si es necesario que un robot conozca exactamente su posición en términos de coordenadas métricas, en este caso, los mapas métricos son mejor elección. En otros ambientes, como edificios de oficinas con pasillos y habitaciones, los mapas simplemente especifican la topología de localizaciones importantes y sus conexiones. Estos mapas son menos

complejos y soportan de una manera más eficiente la planificación que los mapas métricos o geométricos.

Los mapas topo-geométricos combinan los dos paradigmas anteriores. Es decir, generar mapas topológicos que facilitan la planificación y que poseen información geométrica que permita una localización más precisa.

Los mapas geométricos describen el entorno como un conjunto de objetos o de posiciones ocupadas en el espacio, y las relaciones geométricas entre ellos. Cuando se genera un mapa geométrico, existen dos métodos básicos: los mapas basados en la extracción de características geométricas (marcas artificiales o marcas naturales) y los mapas basados en celdillas (mapas de ocupación).

El método utilizado en el desarrollo de esta práctica está basado en la generación de mapas de ocupación.

7.2.10.3 Mapas de Ocupación

En robótica móvil, un mapa de ocupación es una representación del entorno en forma de mosaico o rejilla compuesta de celdas, que almacenas información del entorno [Elfes,87]. La información contenida habitualmente es de carácter probabilístico, y se recoge con sensores montados en una plataforma móvil, como puede ser el sonar. Esta rejilla es generada por un robot en movimiento, que va actualizando de forma recursiva cada celdilla con el cambio de posición. A menudo, la información contenida en la celda no es más que un simple número que representa la creencia de que esa parte del entorno está ocupada por un objeto. Usando esa información de las celdas, se pueden programar tareas de localización o programación de trayectorias.

Una variación de este tipo de mapas utiliza en vez de celdillas los llamados *quadtrees*. Se considera una única celdilla inicialmente, y se va comprobando de forma recursiva si esta celdilla está totalmente ocupada o libre. Si sólo parte de la superficie que abarca la celdilla está ocupada, se divide esta celdilla en otras cuatro celdillas y se realiza de nuevo la comprobación. Así se obtiene mapas de resolución variable, con mayor resolución en los bordes de los objetos detectados. Un inconveniente de este método es la necesidad de conocer el entorno a priori, lo cual ha acabado relegando su uso a aplicaciones con robots industriales.

Los mapas basadas en celdillas son muy utilizados para almacenar y mantener información de ocupación debido a que son muy fáciles de mantener y construir. Sin embargo, es difícil seleccionar la resolución de la celdilla que es adecuada para la representación, y que sirve como base para la representación del entorno. Por ejemplo, una localización muy precisa requiere una resolución alta del mapa sobre el espacio entero. Esto implica unos requerimientos altos en cuanto a almacenamiento de datos. Existen distintas estrategias a la hora de actualizar la información almacenada en las celdillas. A continuación se describen en breve los dos métodos utilizados en la práctica:

- **Método de Bayes:** es el método más ampliamente utilizado en mapas de ocupación, y el primero desarrollado con éxito. En este método, un modelo del sensor utilizado convierte las medidas tomadas en probabilidades condicionales, y confía posteriormente en la regla de Bayes para actualizar estas probabilidades [Elfes,87].
- **Teoría de Broenstein y Koren:** presenta un nuevo método llamado HIMM (*Histogramic In-Motion mapping*) para determinar si un determinado elemento de una rejilla de ocupación está

vacía u ocupado [Borenstein,91]. Este método surge de la necesidad de mejorar la capacidad de evitar obstáculos en navegación, así como de la necesidad de algoritmos más rápidos. Este método utiliza un modelo de sonar más simple que el método de Bayes, con dos diferencias notables. Primero, sólo son actualizados los elementos a lo largo del eje acústico. Esto permite una mayor velocidad de cálculo, pues eliminaría hasta el 90% de las celdillas actualizadas con el otro método [Borenstein,91]. Así pues, si bien las lecturas del sonar son interpretadas de una forma distinta y sobre un menor número de elementos. Segundo, la incertidumbre se expresa con un entero entre 0 y 15, lo que significa que se puede representar con un único byte. La regla de actualización resulta también más simple y computacionalmente eficiente. Se realiza mediante adiciones y substracciones, en vez de multiplicaciones. Si un elemento se detecta como ocupado, aumenta su valor en 3 puntos, y si se detecta como vacío, disminuye en un punto.

7.2.10.4 Elementos Necesarios para el Mapeado

Para realizar un mapa de ocupación, se necesitan los siguientes elementos:

- **Una rejilla de ocupación:** Esta rejilla se corresponde con una matriz de elementos con los valores de ocupación de cada región espacial.
- **Inicialización del mapa de ocupación:** Para la inicialización del mapa bayesiano, se establecen las probabilidades de las celdas con un valor 0,5, esto es, a priori desconocido (ni ocupado ni vacío). En el caso del método de Borenstein, el valor de ocupación inicial que se toma es 0.
- **Un modelo de interpretación espacial:** Desarrollado para cada tipo de sensor, que traduzca los datos del sensor en una declaración acerca del área o volumen (en caso de mapas 3D) que está ocupado o vacío. En el caso del algoritmo bayesiano, incorporando la incertidumbre del sensor en la forma de función de densidad de probabilidad, se obtiene un modelo probablistico del sensor. El modelo del sensor que se utiliza para el método de Broenstein resulta mucho más sencillo, ya que se actualizan únicamente los valores sobre el eje acústico.
- **Un modelo de actualización del mapa:** Este modelo permite componer los distintos mapas parciales que se van realizando. El coste computacional de actualizar el mapa de Broenstein es mucho menor que el de Bayes.

7.2.10.5 Pasos del Mapeado

Para realizar una iteración en el mapa de ocupación, se han de realizar los siguientes pasos:

- Obtener la posición y orientación del robot y las medidas de los 24 sensores del robot.
- Calcular la posición relativa de la celdilla con respecto a la posición del robot. Así, se obtiene la distancia absoluta entre ambas celdillas (el módulo del vector distancia) y su posición angular (el argumento).
- Hallar a qué sector (de los 24) correspondería esa medida.
- Una vez averiguado el sector, con la medida de ese sensor se aplica el modelo del sensor, y se halla el valor de probabilidad de ocupación de la rejilla.
- Se aplica el modelo de actualización.

7.2.10.6 Interfaz del Experimento

En la figura 7.46, se pueden ver los mapas generados utilizando el método de Bayes y el método de Borestein-Koren. En la figura 7.46 (izquierda), se puede ver el mapa representado tras 76 iteraciones. Los colores representan la probabilidad de ocupación, siendo el blanco el estado vacío y el negro ocupado. Los valores representados en naranja que hay detrás de las zonas más oscuras significan estado desconocido y tienen un valor de 0,5. A medida que se va realizando el mapa, se puede ver como las zonas más oscuras aumentan su valor y las más claras la disminuyen, aumentado la precisión del mapa con el tiempo.

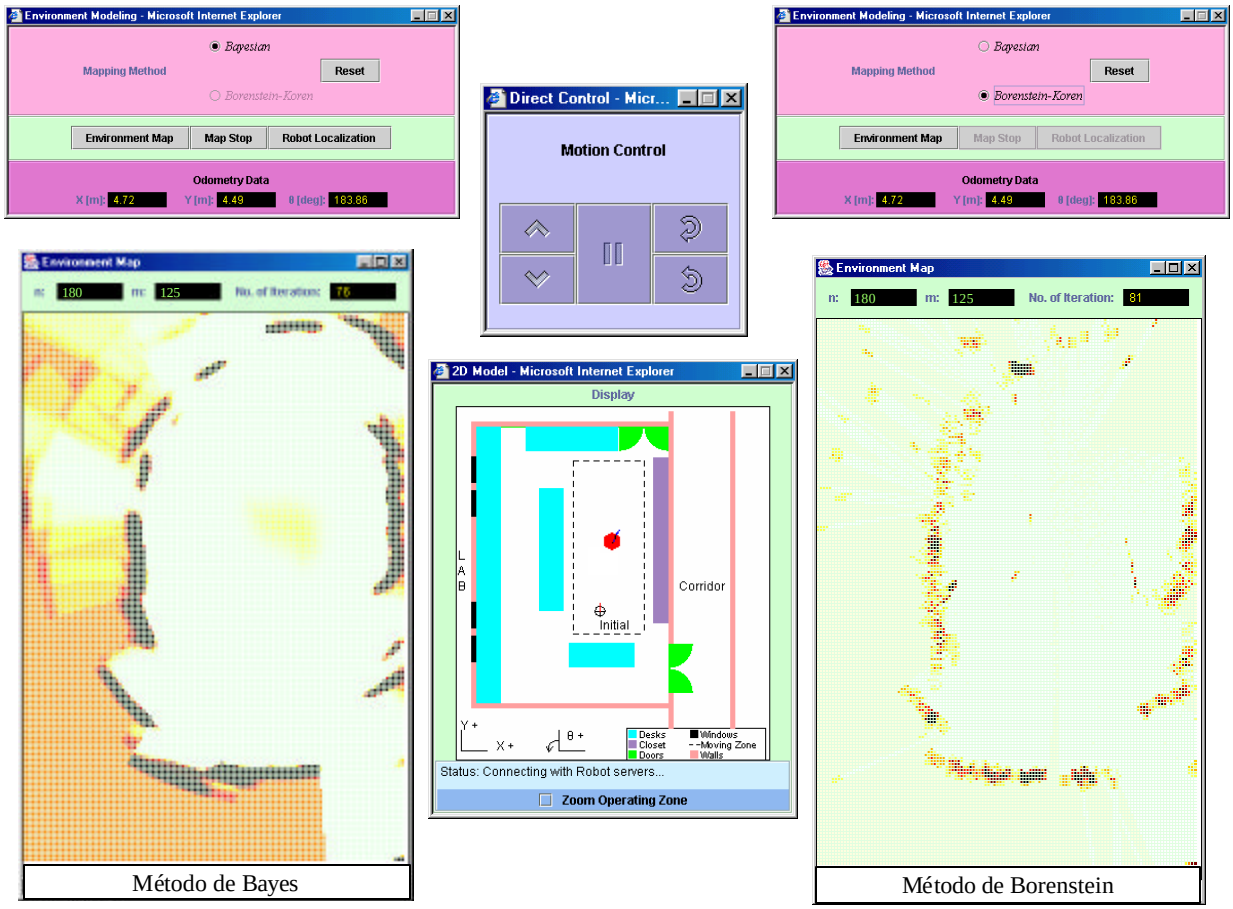


Fig. 7.46: Modelado del Entorno

Igualmente, con el método Borenstein-Koren, se va realizando el mapa. Hay que tener en cuenta que se realiza una iteración cada 500 ms. Tras realizar un recorrido por el laboratorio, se ve el mapa (la figura 7.46 - derecha) tras 81 iteraciones. En este método, la escala de colores es equivalente a la del método bayesiano, solo que inicialmente el mapa se considera vacío y se va oscureciendo a medida que se van detectando los objetos del laboratorio. Aquí un valor de color intermedio significa un valor de ocupación de probabilidad media, no desconocido como en el método bayesiano.

Una ampliación interesante sería la fusión de datos sensoriales del sonar y del láser, mejorando la precisión de los mapas realizados.

7.2.11 Localización

Localización es el proceso a través del cual se determina la posición del robot dentro del entorno. Se ha desarrollado una práctica para estudiar el problema de la localización utilizando un algoritmo descrito en [Brown,00]. El concepto más destacable de este algoritmo es el de posición factible (*feasible pose*). Una posición factible es aquella que es coherente con los datos de alcance (lectura del sensor) y la información procedente del mapa. El algoritmo divide el espacio donde se mueve el robot en posiciones que están en conflicto con los datos del sensor y posiciones que no lo están. Éstas últimas serían factibles.

Como se muestra en la figura 7.47, las posiciones son factibles dependiendo de varios parámetros. Una posición es factible con respecto a un mapa y un vector de alcance si esa posición sitúa al robot de forma que el vector de alcance termine en el borde de un obstáculo. Y no sería factible si el extremo del vector terminara en una posición vacía. Una posición es factible con respecto a un mapa y un conjunto de vectores de alcance si es factible con respecto al mapa y a cada uno de los vectores, de la forma como se ha explicado antes.

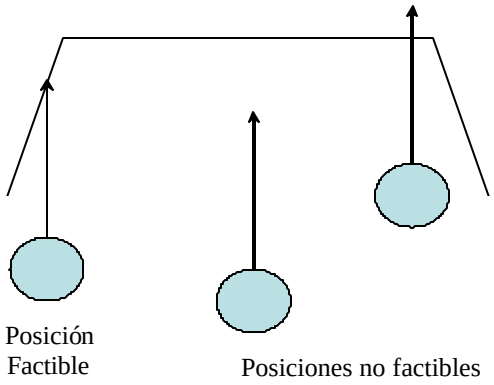


Fig. 7.47: Posiciones factible y no factibles

Se puede determinar el grado con el que una posición es factible con un margen de error: para ello se puede permitir que la longitud del vector de alcance varíe dentro de un limite de incertidumbre, y determinar que posiciones son coherentes con un error dentro del limite de incertidumbre.

¿Cómo se pueden evaluar las posiciones factibles para un mapa M y un vector de alcance denominado z? De forma intuitiva, el procedimiento es encontrar las posiciones para el robot de forma que su sensor (sonar) diera como lectura el vector z en el mundo representado por M. El vector z es tal que su base se sitúa en el robot y su extremo se apoye en el borde de algún obstáculo. Si z puede estar situado con su base en algún punto P, de forma que el extremo esté en contacto con algún obstáculo, y que no exista ninguno que lo intersecte –excepto en su extremo-, entonces P es factible para el mapa M y el vector de alcance z.

La figura 7.48 muestra como trabaja el método de localización utilizado en el desarrollo de esta práctica. En la nomenclatura utilizada, las posiciones factibles con respecto a un mapa M y una sola lectura de alcance –denominada z-, frente a las posiciones factibles con respecto a un mapa M y un conjunto Z de lecturas de alcance se denota FP(M,z) frente a FP(M,Z).

La parte a) de la figura 7.48 muestra el mapa M y las partes b), d) y g) muestran los tres vectores: x, y y z, que representan tres lecturas del sonar, o lo que es lo mismo, tres vectores de alcance. La parte c) muestra FP(M,x): las líneas continuas en negro son FP(M,x) porque todas son posiciones dentro de M en las que se podría situar la base de x de forma que su extremo se apoye en un obstáculo pero su cuerpo no.

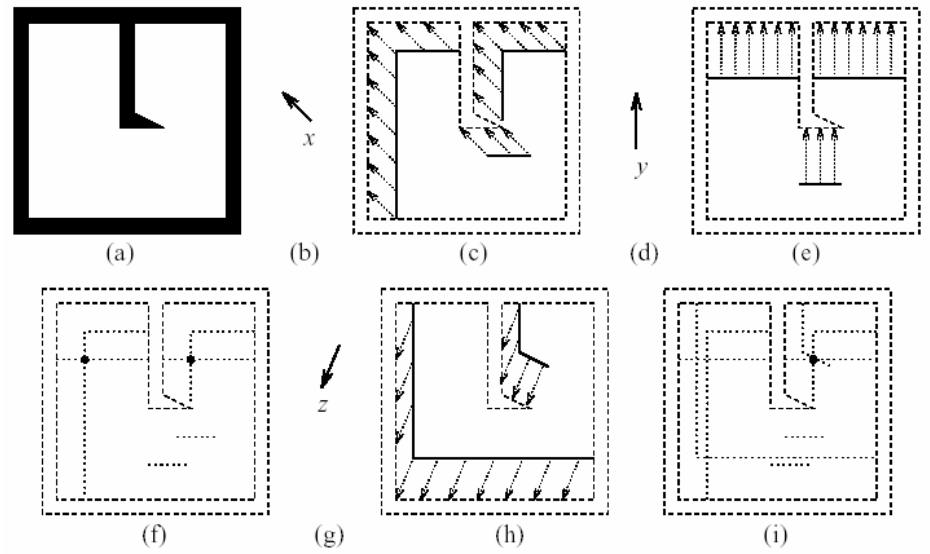


Fig. 7.47: Método de Localización

Las partes e) y h) muestran FP(M,y) y FP(M,z). La parte f) muestra FP(M,{x,y}): los dos puntos resaltados en negro son las posiciones dentro de M donde intersectan FP(M,x) y FP(M,y). El hecho de que haya dos puntos indica que dos vectores de alcance, x e y, son insuficientes para localizar el robot de forma inequívoca dentro de M. Tomando una tercera lectura, esto es, otro vector de alcance –en este caso z., se observa que el robot puede ser localizado correctamente.

La parte i) muestra la intersección de FP(M,x), FP(M,y) y FP(M,z). Esta intersección se sitúa en un único punto. De esta forma, se puede afirmar que este punto, que se corresponde con FP(M,{x,y,z}), es el punto en el que se localiza el robot.

En la figura 7.49, se pueden ver las entradas del algoritmo de localización implementado. Como se ha mencionado anteriormente, el mapa de ocupación, se trata de una matriz rectangular, que representa un modelo geométrico del entorno, con valores de celdilla entre probabilidad cero (celda vacío con total seguridad) y uno (celda ocupada con total seguridad). El algoritmo de localización utiliza como entradas mapas binario. Por lo tanto, es necesario segmentar los valores del mapa en dos grupos. Se considerara celdilla ocupada toda aquella que tenga un valor de 0,5 o mayor, y se considerara celdilla vacía la que tenga un valor inferior a 0,5.

La lectura de alcance es la salida del sensor, representada por un vector en metros hasta un obstáculo. Un vector de alcance se construye considerando el modulo igual a este valor, y el argumento se obtendrá del ángulo de orientación del robot. Esta lectura es un valor continuo, y necesita ser discretizado para que se pueda utilizar como entrada del algoritmo de localización. Esto supone una pérdida de precisión igual a la unidad mínima de discretización, que será el tamaño de celdilla. Existen 24 sensores (numerados de 0 a 23), que suministran lecturas de forma ordenada y simultánea; esto es, dan lecturas en el mismo instante de tiempo (las lecturas no se

producen en el mismo instante de tiempo; existe un pequeño retardo entre la obtención de cada lectura, para que no se produzcan interferencias, pero a efectos de trabajo, se transmiten cuando se tienen las 24 lecturas). Las lecturas se agrupan de forma sistemática en una estructura de datos tipo vector de 24 elementos. La orientación del robot es otro dato, que debe ser considerado como entrada al algoritmo de localización.

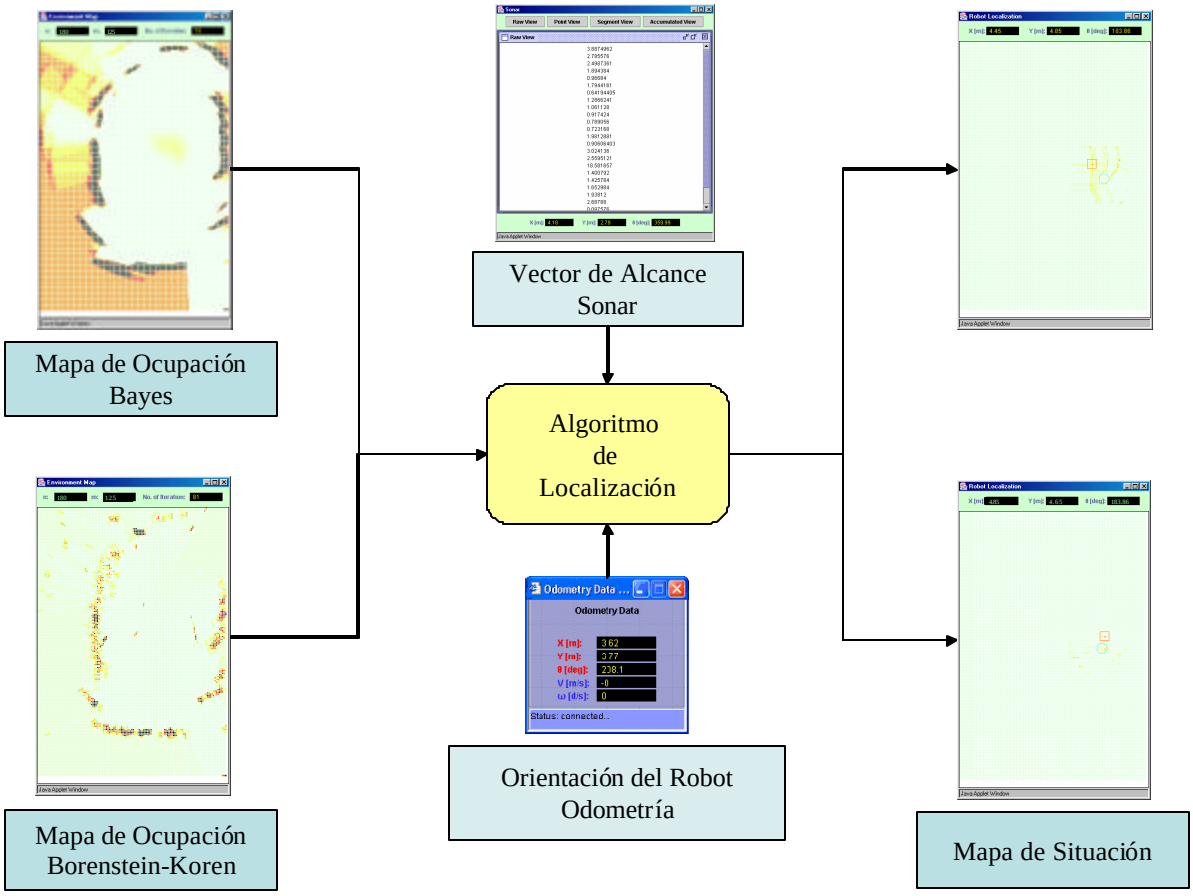


Fig. 7.48: Entradas del Algoritmo de Localización

La localización utiliza la última actualización del mapa de ocupación. Por ello es necesario que la tarea de mapeado haya sido arrancada con anterioridad. Cuanto más preciso sea el mapa que se utilice, de forma más adecuada se desarrollará la localización. En este sentido, se han observado diferencias según el método de mapeado utilizado. Para obtener un mapa con un grado de detalle importante, el método de Borenstein-Koren necesita bastantes más iteraciones que el método de Bayes.

En la figura 7.50 aparece la localización cuando se utiliza como entrada un mapa desarrollado por el método de Bayes y cuando se usa el mapa de Borestein-Koren.

El conjunto de celdillas que constituyen posiciones factibles, se representan gráficamente en un mapa de las mismas dimensiones que el laboratorio que constituye el entorno del robot. A esta representación se le ha denominado mapa de situación. Las posiciones factibles que más se repiten para cada una de las lecturas de alcance, se representan con celdillas coloreadas con mayor intensidad, y concretamente la posición factible más repetida, viene recuadrada en rojo. Así mismo, aparece una circunferencia en color azul, que se corresponde con la posición dada por

la odometría. Esto permite comparar visualmente la localización a través de ultrasonidos aplicando el algoritmo de localización, con la localización que proporciona la odometría.

Se observa que la estimación de la posición del robot derivada del algoritmo de localización varía muy poco con respecto al valor obtenido a través de la odometría. La precisión en la localización, es directamente proporcional a la precisión del mapa de ocupación. Las características, número y disposición de los sensores son los que en primer lugar determinan la calidad del mapa de ocupación.

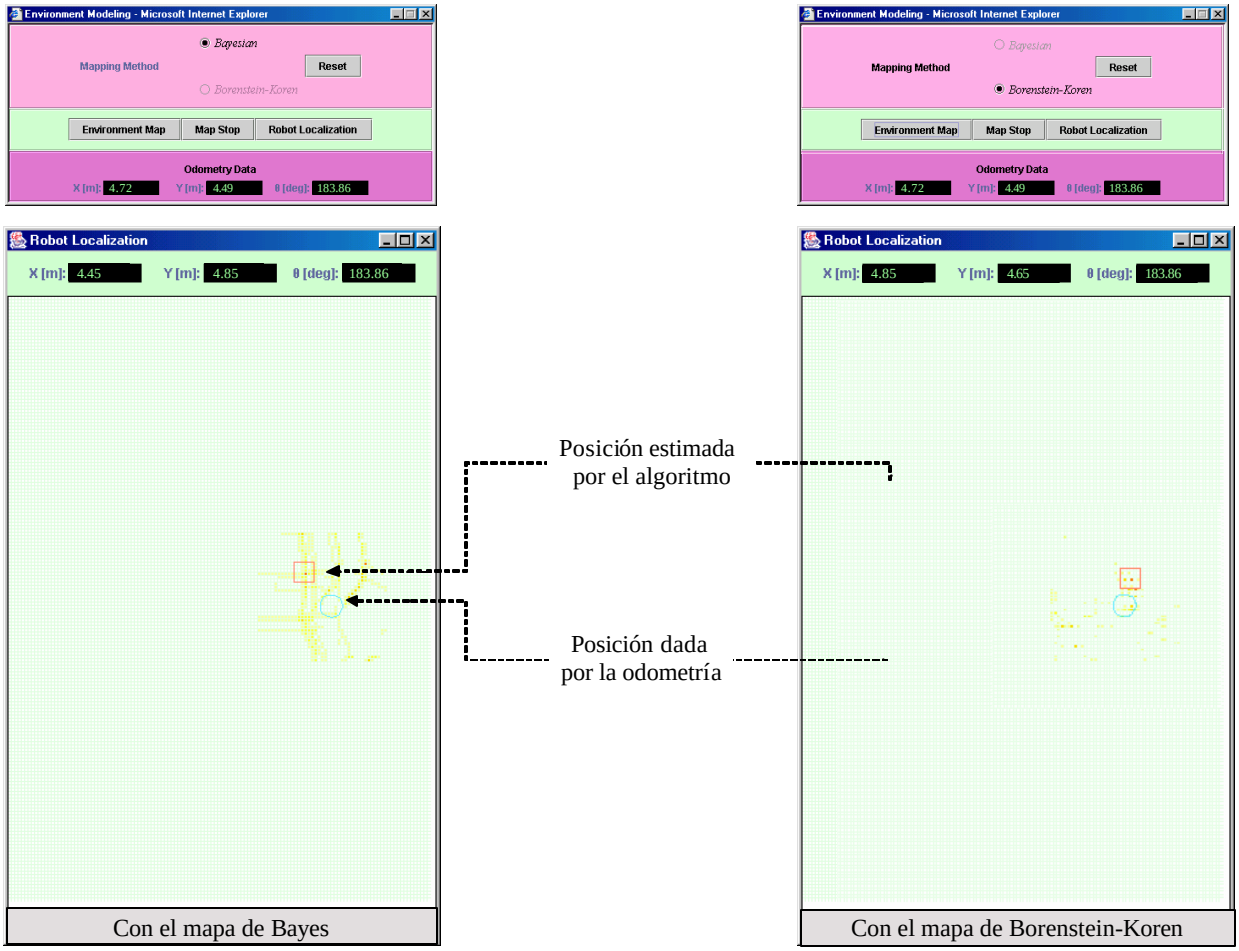


Fig. 7.50: Mapas de Situación

Como trabajo futuro, se pretende desarrollar una interfaz para la habilidad SLAM desarrollada en el departamento. La habilidad SLAM (*Simultaneous Localization and Mapping*) es una simulación e implementación del problema de realizar la localización y el mapeado simultáneamente para robots móviles en entornos interiores.

7.3 UN ENTORNO EDUCATIVO PARA LA ROBÓTICA MÓVIL

En el capítulo 4 se ha propuesto una metodología, que combina diferentes actividades educativas para construir entornos educativos innovadores para la robótica móvil. En este apartado se explica cómo se puede integrar el laboratorio remoto desarrollado en esta tesis en un entorno educativo basándose en esta metodología. La figura 7.51 muestra las actividades implementadas en un curso fundamental de robótica móvil [UC3M,03].

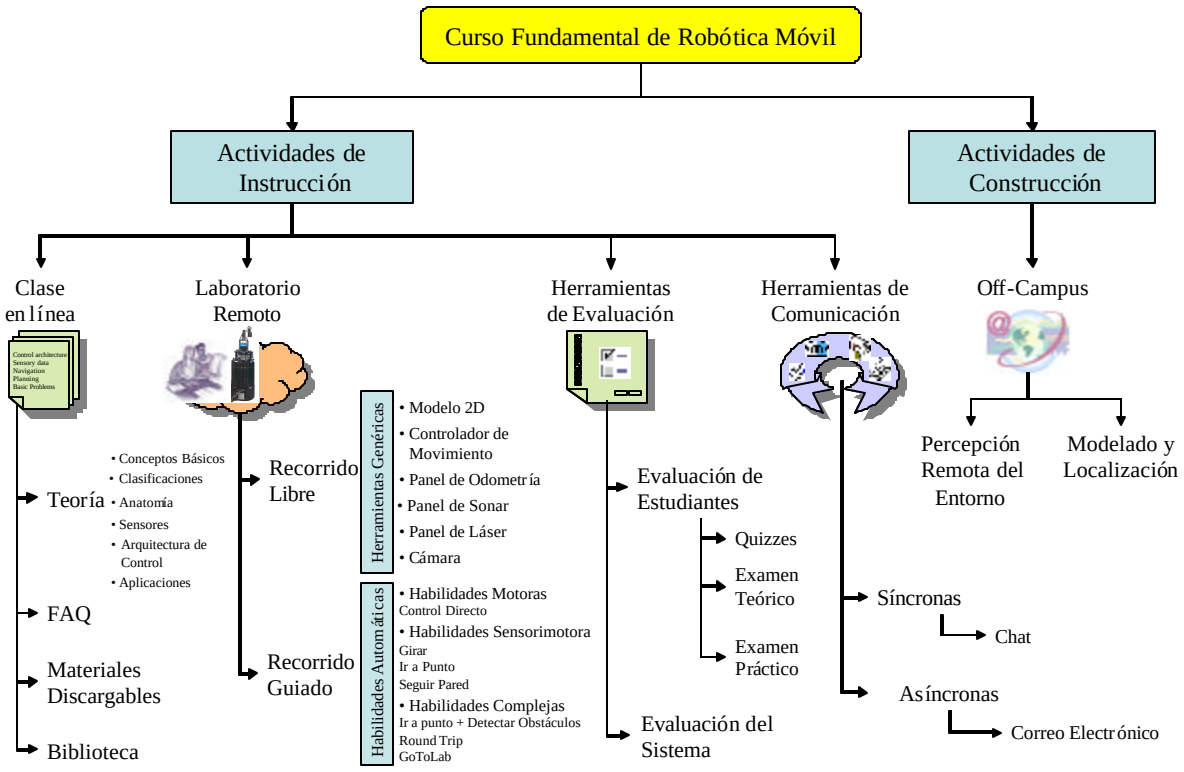


Fig. 7.51: Actividades Implementadas de un Curso Fundamental de Robótica Móvil

Como se ha mencionado en el capítulo 4, las actividades de instrucción basadas en Web se usan para mapear la enseñanza tradicional y para resolver los problemas de este paradigma tradicional como el número elevado de estudiantes y los recursos limitados. Estas actividades deben complementarse con otras actividades o herramientas que permiten a los estudiantes tener un papel activo en el sistema y que pretenden mejorar la creatividad de los estudiantes. Esto se puede conseguir mediante el desarrollo de actividades de construcción, que dan a los estudiantes la oportunidad de construir físicamente y poner en práctica las ideas derivadas del curso. Estas actividades también pretenden proporcionar la comunicación cara a cara entre el estudiante y el tutor y la comunicación hombre-máquina que es muy importante en un campo como la robótica móvil. A continuación se explican las actividades con más detalles.

7.3.1 Actividades de Instrucción

Se han implementado varias actividades de instrucción utilizando el modelo de educación basado en Web para mapear las actividades de la enseñanza tradicional. En las subsecciones siguientes se plantean las actividades implementadas en el curso fundamental de robótica móvil.

7.3.1.1 Clase en línea

Esta clase pretende introducir los conceptos básicos de los robots móviles como sus componentes, los sensores, la arquitectura de control, etc. También se han añadido a este curso, páginas de preguntas más frecuentes (FAQ), una máquina de búsqueda, una biblioteca digital y varios materiales descargables como se muestra en la figura 7.52.

Fig. 7.52: Clase en Línea

7.3.1.2 Laboratorio Remoto

En el laboratorio remoto, se presentan dos tipos de recorridos de instrucción al estudiante como herramientas educativas innovadoras para profundizar y aplicar el conocimiento sistemático de robótica móvil. Como se ha mencionado en el capítulo 4, se clasifican según el nivel de apoyo, en un recorrido libre y otro guiado. El recorrido libre no determina ninguna orden de situaciones o tareas. Este recorrido proporciona a los usuarios herramientas genéricas por

medio de las cuales el usuario puede personalizar el experimento según sus necesidades. Como se muestra en la figura 7.53, estas herramientas genéricas incluyen un modelo 2D para el robot y el laboratorio, un panel para los datos de la odometría, los datos del sonar, los datos del láser, un controlador de movimiento y una cámara Web. El recorrido guiado presenta a los usuarios las interfaces de las habilidades automáticas implementadas.

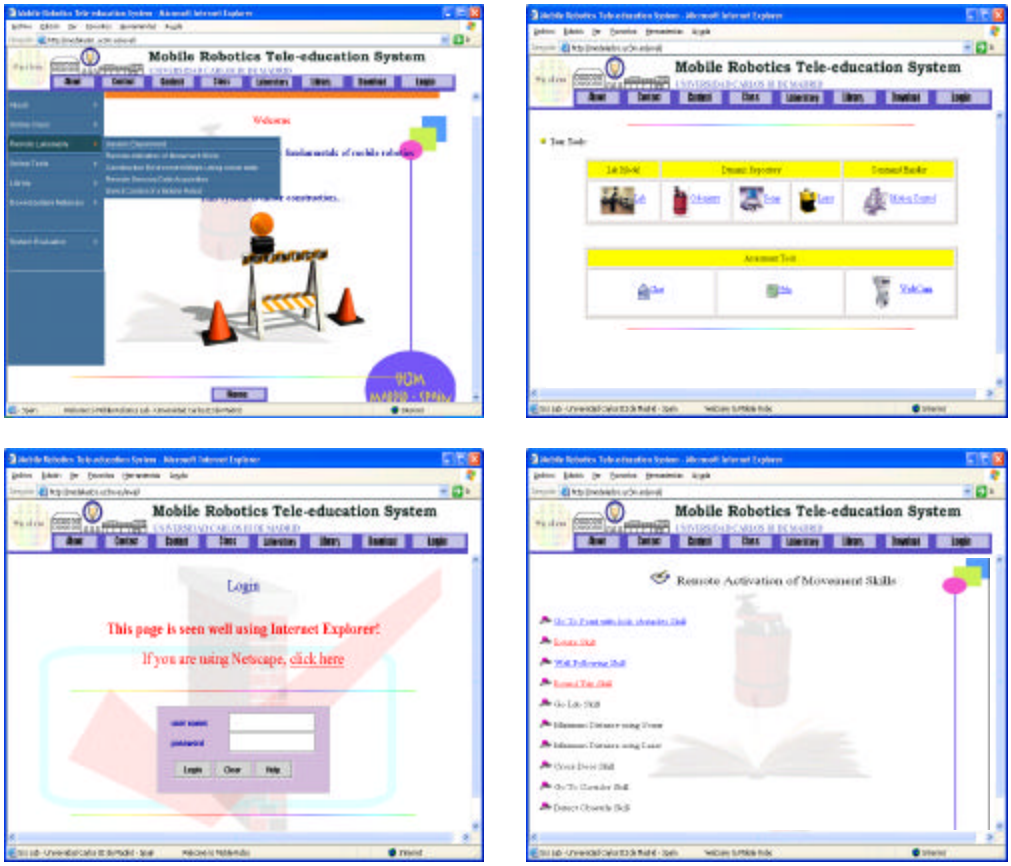


Fig. 7.53: Laboratorio Remoto

7.3.1.3 Herramientas de Evaluación

Se han diseñado tres tipos de exámenes como herramientas de evaluación. Exámenes cortos (*Quizzes*) que se usan para evaluar los conocimientos de los estudiantes, exámenes teóricos con tiempo limitado que se usan para evaluar los conocimientos teóricos adquiridos en el curso y finalmente exámenes prácticos con el objetivo de ayudar a los estudiantes a entender bien los problemas prácticos.

7.3.2 Actividades de Construcción

Como se ha mencionado en el capítulo 4, las actividades de construcción ayudan a los estudiantes a tener un papel activo y creativo en el proceso de educación. Se pueden utilizar las prácticas de percepción remota del entorno, modelado y localización como actividades de construcción remotas, que se pueden realizar si necesidad de que el estudiante esté físicamente en el laboratorio. Como trabajo futuro se pretende facilitar la programación remota del robot

mediante un editor de comandos por medio del cual los usuarios pueden enviar comandos de bajo nivel al robot para llevar a cabo varias tareas como estudiar el efecto de variar los parámetros del controlador, cambiar la velocidad del robot, informar sobre el estado de la batería del robot, etc.

También, como trabajo futuro, se pretenden desarrollar actividades de construcción locales como diseñar y construir nuevos prototipos o aplicar ingeniería inversa a sistemas existentes. Estas actividades como se ha mencionado el capítulo 4, se pueden usar para proporcionar la interacción cara-a-cara entre los estudiantes y los tutores y entre el estudiante y la máquina, y por consiguiente se aumenta la motivación de los estudiantes en la participación en el proceso educativo y se disminuye la sensación de aislamiento que algunos sienten al usar los sistemas de educación a distancia totalmente basados en Web. Ayudan también a aumentar la creatividad de los estudiantes y el trabajo en equipo. Para desarrollar estas actividades, se puede pedir a los estudiantes que construyan prototipos robóticos utilizando componentes hardware como Lego o Fischertechnik como componentes del hardware y los microcontroladores como Motorola 88HC11 o 16-bit Siemens 80C167 y utilizar varios sensores disponibles para los robots móviles.

7.3.3 Evaluación del Sistema

El entorno educativo propuesto se usa, a partir del año académico 2001-2002, para modernizar una asignatura de doctorado de robots autónomos inteligentes. Los participantes en la asignatura (12 estudiantes) han evaluado el sistema mediante una encuesta en línea. Generalmente la impresión de los estudiantes ha sido positiva especialmente respecto al uso del laboratorio remoto. La mayoría de los estudiantes sentían que los experimentos en línea les ayudaron a lograr un entendiendo más profundo. La figura 7.54 muestra la evaluación del sistema por parte de los estudiantes. La apreciación de los participantes, que queda almacenada en el servidor Web, indica que los principales objetivos tanto del curso, como del laboratorio remoto han cumplido las expectativas.

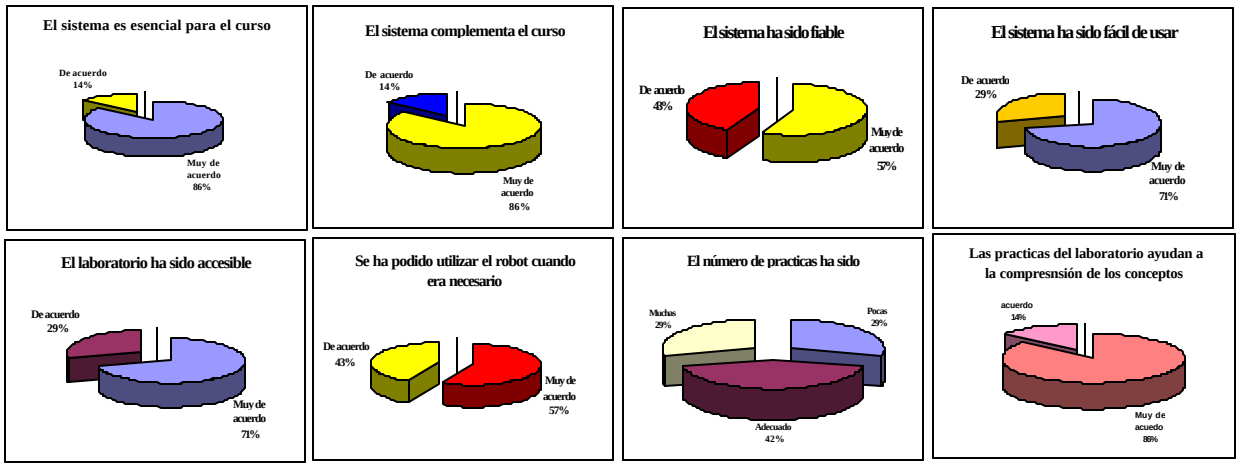


Fig. 7.54: Evaluación del Sistema

La figura 7.55 muestra las opiniones de los participantes respecto a cómo mejorar el sistema.

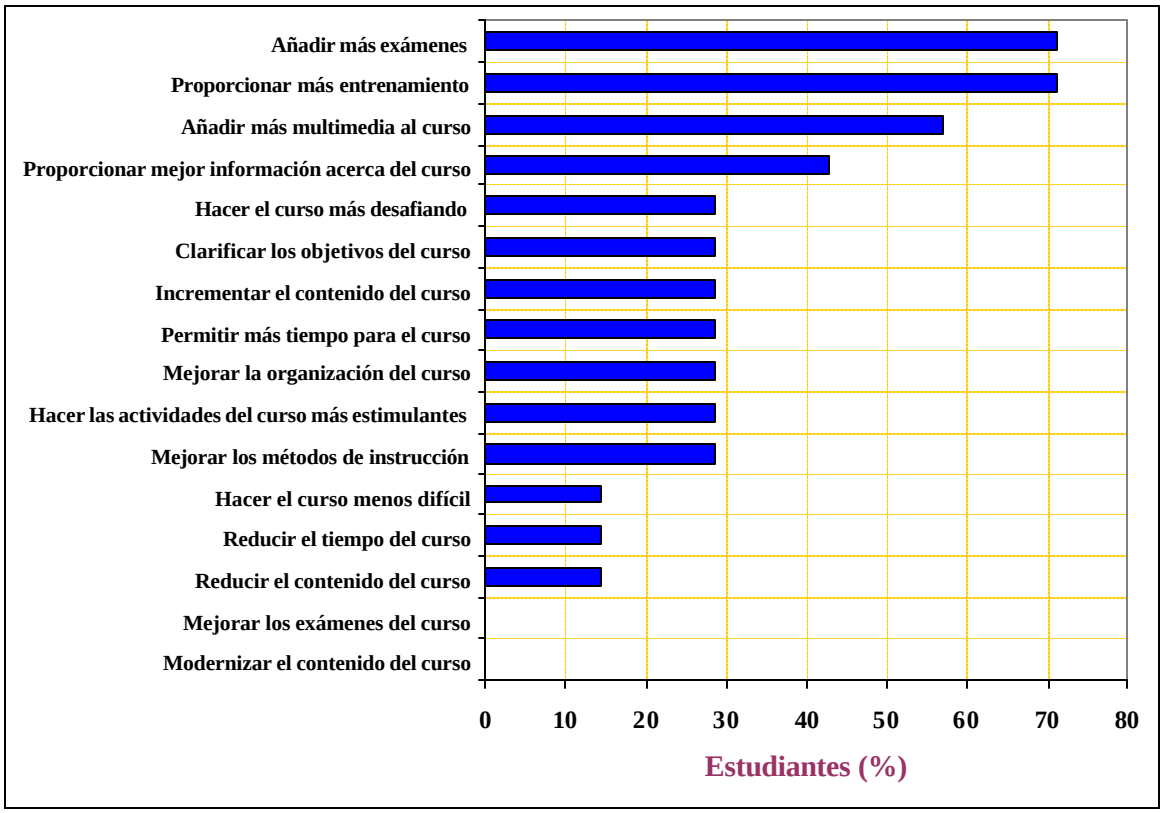


Fig. 7.55: Mejorar el Sistema

Se ha realizado esta tesis en el marco del proyecto IECAT [IECAT,03] que pretende desarrollar una red de laboratorios remotos entre seis universidades en Europa y EE.UU. en el campo de los sistemas autónomos y teleoperados. Este laboratorio distribuido internacional va a permitir a los estudiantes llevar a cabo experimentos con equipamiento real y simulado perteneciente a las universidades participantes. Debido a la larga distancia que separa unas universidades de otras, los estudiantes tendrán a su disposición todo lo necesario para la teleoperación de robots y otros experimentos mediante laboratorios remotos, accesibles vía Internet.

El sistema propuesto va a ser la aportación principal de la Universidad Carlos III de Madrid en el proyecto IECAT, que termina al final de este año 2003. En la fase final del proyecto se pretenden hacer pruebas de operación entre las universidad participantes.

Capítulo

8

CONCLUSIONES, APORTACIONES Y DESARROLLOS FUTUROS

Capítulo 8 CONCLUSIONES, APORTACIONES Y FUTUROS DESARROLLOS

8.1 CONCLUSIONES

Se ha centrado esta tesis en el desarrollo de una arquitectura software para un laboratorio remoto en el campo de la robótica móvil. El laboratorio desarrollado proporciona un entorno innovador, que facilita la interacción remota con los robots móviles utilizando varios elementos de interacción. Dicho entorno garantiza también un envolvimiento continuo y activo del usuario en el sistema, proporcionándole realimentación suficiente sobre todas las tareas que el robot está llevando a cabo y notificándole continuamente acerca del estado de la conexión.

Algunas conclusiones derivadas del trabajo desarrollado se recogen en los siguientes puntos:

- Se puede usar la interacción remota basada en Internet con los robots móviles en varias aplicaciones útiles como experimentación remota, teleoperación, telepercepción, teleprogramación, etc. La experimentación remota tiene la ventaja de proporcionar la posibilidad de compartir un experimento o varios experimentos entre varios operadores localizados en lugares distintos. De esta manera, podrían compartirse fácilmente experimentos reales entre varios laboratorios y, por lo tanto, se podrían reducir los costes.

- Los laboratorios remotos y distribuidos son entornos innovadores que pueden usarse para facilitar la experimentación remota vía Internet. Dichos laboratorios pueden ser una posible solución a los problemas que existen en los laboratorios experimentales universitarios. Estos laboratorios ayudan a intercambiar recursos de hardware y materiales educativos entre los

participantes. También tienen la ventaja de no estar restringidos a la asistencia sincronizada por instructores y estudiantes, teniendo la capacidad de proporcionar acceso constante siempre y cuando sea necesario. Se pueden usar estos laboratorios para proporcionar a los usuarios un conjunto coordinado de experimentos con recursos físicamente distribuidos en diferentes lugares pero accesibles vía Internet.

- La falta de un contacto físico y la necesidad de proporcionar herramientas genéricas y flexibles a los usuarios, que les permitan diseñar sus propios experimentos son algunos de los inconvenientes de los laboratorios remotos y distribuidos. Se pueden superar dichos inconvenientes utilizando varias actividades educativas, que combinan actividades de instrucción basadas en Web y actividades de construcción. Las actividades de instrucción, tales como clases en línea, laboratorios remotos, herramientas de evaluación y mecanismos de comunicación síncronos y asíncronos, pretenden mapear las actividades de la enseñanza tradicional y solucionar los problemas asociados con dichos sistemas como el número elevado de estudiantes y los recursos limitados. Por otro lado, las actividades de construcción pretenden dar a los estudiantes la oportunidad de construir físicamente y poner en práctica las ideas derivadas del curso. En lugar de recibir información de una manera unidireccional, los estudiantes desarrollan su propio conocimiento y comprensión de un asunto a través de la construcción física y la implementación de sus ideas. Estas actividades proporcionan la interacción cara-a-cara entre los estudiantes y los tutores y entre el estudiante y la máquina, y por consiguiente se aumenta la motivación de los estudiantes en la participación en el proceso educativo y se disminuye la sensación de aislamiento que algunos sienten al usar sistemas de educación a distancia totalmente basados en Web. Además ayudan a aumentar la creatividad de los estudiantes y el trabajo en equipo.

- En esta tesis se ha estudiado las ventajas y los inconvenientes de utilizar la red Internet como medio de comunicación en los sistemas de interacción remota con los robots móviles. Se ha realizado este estudio para evaluar el rendimiento de la comunicación entre la Universidad Carlos III de Madrid, y la Universidad de Ciencias Aplicadas FH-Weingarten en Alemania, que son miembros del proyecto IECAT. Aunque Internet proporciona un medio de comunicación barato y permanentemente disponible para la interacción remota con los robots, existen todavía muchos problemas que están por resolver. Entre los más importantes cabe citar el ancho de banda restringido y el retraso de la transmisión arbitrariamente grande que influencia el rendimiento y la seguridad y la fiabilidad del sistema basado en Internet. Las conclusiones más destacadas de este estudio han sido las siguientes:

  - **El retraso temporal:** Es muy imprevisible e inevitable a diferencia de los sistemas tradicionales de teleoperación. El retraso de encolamiento representa la mayor parte del retraso total de la comunicación. Las distancias físicas no tienen efecto dominante en el retraso. Este retraso es especialmente crítico debido a que su valor afecta directamente a la realimentación sensorial y a las capacidades de control de bucle cerrado. El retraso de transmisión tiene que ser aceptable o se tiene que utilizar una conexión dedicada. Debido a la variación de arquitecturas de teleoperación, no se puede dar ninguna formulación general de los requisitos mínimos. Normalmente, un retraso de comunicación de un segundo como máximo, se tiene como referencia para garantizar la operabilidad del sistema.

- **El Ancho de Banda:** La idea de superar las restricciones de la comunicación utilizando interacción de nivel más abstracto y aumentando la autonomía del robot es una idea fundamental para el control remoto vía Internet. Los modelos gráficos y las imágenes de realidad virtual tienen necesidades mucho más bajas en cuanto al ancho de banda, por lo tanto se pueden considerar como una mejor alternativa a las imágenes reales de la cámara en este tipo de situaciones. Para proporcionar en tiempo real imágenes de sitios remotos, el sistema debería reaccionar dinámicamente a los cambios del ancho de banda y a los recursos computacionales.
- **Pérdida de Paquetes:** Probablemente la preocupación más grande de los sistemas de interacción remota basados en Internet es el comportamiento no determinista del sistema que resultaría durante la pérdida de paquetes o la caída total de la comunicación entre los sitios del sistema. Una posibilidad para prevenir la pérdida de paquetes está implementada en TCP. En este protocolo, cuando se descubre una pérdida de paquetes, se pide un reenvío por el receptor. Esto produce una latencia más alta con el protocolo TCP comparándolo con el UDP, existiendo una compensación entre la porción de pérdida de paquetes y el retraso temporal.
- ***Jitter*:** Las aplicaciones de voz son muy sensibles a la variabilidad instantánea o el *jitter*. Aunque utilizar la comunicación hablada como herramienta de interacción remota requiere transmitir poca cantidad de datos, el *jitter* afecta a la calidad de la voz llegada al sitio remoto donde se encuentra el robot.

- En esta tesis, se ha estudiado también la estrategia de control adecuada para los sistemas de interacción remota basados en Internet. Se recomienda utilizar el paradigma del control supervisado para disminuir el ancho de banda necesario y la sensibilidad del sistema al retraso temporal. El control supervisado pretende hacer que la tarea de control sea compartida entre un lazo de control remoto y el operador. Este paradigma de control no es para que un robot realice todas las operaciones autónomamente, pero habilita al robot a realizar operaciones simples que el operador puede secuenciar.

- Los patrones de software ayudan a construir software basado en la reutilización. Los propios patrones se reutilizan cada vez que se vuelven a aplicar. Se pueden usar los patrones para desarrollar interfaces hombre-robot intuitivas, extensibles y reutilizables.

- Se ha estudiado el uso de las tecnologías disponibles para desarrollar aplicaciones inalámbricas como la tecnología Java y el WML. Existen muchos factores a favor de usar la tecnología Java para desarrollar aplicaciones para dispositivos móviles. Entre ellos cabe citar la compatibilidad de plataforma, la elección dinámica de aplicaciones y servicios, la seguridad y la disponibilidad de documentación y soporte técnico.

8.2 APORTACIONES DE LA TESIS

Las principales aportaciones de la tesis son:

- El desarrollo una arquitectura software cliente-servidor de tres capas para construir un laboratorio remoto en el campo de robótica móvil. Se ha diseñado dicha arquitectura teniendo en cuenta varios requisitos tales como la manejabilidad de la arquitectura, la portabilidad de las interfaces y la reutilización y la extensibilidad del código. En la arquitectura propuesta se usan los patrones de software para desarrollar interfaces hombre-robot intuitivas, extensibles y reutilizables. Se han desarrollado las interfaces de usuario utilizando el patrón *proxy* visual. Este patrón proporciona interfaces de usuario flexibles con relaciones de acoplamiento mínimas entre los subsistemas. La generación de la interfaz de usuario está completamente separada de los objetos de la capa de abstracción proporcionando así facilidad de reutilización y extensibilidad.

- Se ha utilizado el paradigma del control supervisado en el desarrollo del sistema propuesto para disminuir el ancho de banda necesario y la sensibilidad del sistema al retraso temporal. Mediante el control supervisado el usuario se comunica con el sistema remoto a un nivel más abstracto enviando comandos de alto nivel al robot, el cual, al incrementar su nivel de autonomía permite disminuir la cantidad de datos a enviar. Limitar la interacción remota a comandos de alto nivel ayuda a disminuir el ancho de banda necesario, e incrementar la autonomía del robot ayuda a reducir la sensibilidad al retraso. El nivel de autonomía del robot remoto ha sido aumentado encapsulando todos los comportamientos del robot en diferentes tipos de habilidades basadas en la arquitectura de control AD. Utilizando las interfaces desarrolladas, se han obtenido valores aceptables de retraso temporal (138 mseg. desde UC3M y 324,3 mseg. desde Weingarten) teniendo en cuenta que normalmente se puede considerar un segundo como una referencia de operabilidad para el retraso máximo de comunicación en los sistemas de teleoperación.

- Se han implementado varias interfaces para habilidades automáticas tanto simples como complejas basándose en la arquitectura propuesta. Se han integradas también varias herramientas de realimentación visual tales como un modelo 2D, una cámara Web, paneles de estado y datos sensoriales. Estas herramientas dan al usuario una sensación visual, que expresa la secuencia de sus comandos directamente en la interfaz de control.

- Se han desarrollado también varias interfaces para activar o desactivar habilidades de movimiento del robot B21 utilizando dispositivos móviles tales como las PDAs y los teléfonos móviles. Utilizar dichos dispositivos móviles como elementos de interacción con los robots móviles mejora el alcance y la funcionalidad de los entornos de interacción remota.

- Otra aportación de la tesis es la metodología propuesta para construir entornos educativos innovadores para la robótica móvil. Dicha metodología combina varias actividades para resolver los problemas del paradigma tradicional de instrucción y para permitir a los estudiantes tener un papel activo y creativo en el sistema. Se ha integrado el laboratorio remoto desarrollado en esta tesis en un entorno educativo de robótica móvil basándose en esta metodología. Dicho entorno se usa, a partir del año académico 2001-2002, para modernizar

una asignatura de doctorado de robots autónomos inteligentes. Generalmente la impresión de los estudiantes ha sido positiva especialmente respecto al uso del laboratorio remoto. La mayoría de los estudiantes sentían que los experimentos en línea les ayudaron a lograr un entendiendo más profundo de la materia.

- Finalmente, se han cumplido todos los objetivos del proyecto IECAT en marco del cual se ha realizada esta tesis. El sistema propuesto es la aportación principal de la Universidad Carlos III de Madrid en el proyecto IECAT. Este laboratorio distribuido internacional permite a los estudiantes llevar a cabo experimentos con equipamiento real y simulado perteneciente a las universidades participantes en Europa y EE.UU. en el campo de los sistemas autónomos y teleoperados.

8.3 FUTUROS DESARROLLOS

Existen diferentes líneas de trabajo de gran interés para la continuación de la presente tesis, entre las que cabe mencionar las siguientes:

- Se pretende usar el dispositivo móvil como un elemento principal de interacción por medio de lo cual el usuario será capaz de interactuar con un robot personal, que se está construyendo en el departamento. Se proponen desarrollar varios agentes de interacción como:
  - **Agente de Movimiento:** Consta de diferentes habilidades de movimiento que permiten la movilidad segura del robot en el entorno.
  - **Agente de Percepción:** Permite al robot localizar un objeto y moverse hacia ello. El objeto asignado puede ser una persona, una estación de recarga, un aparato doméstico (como la TV para encenderla), etc.
  - **Agente de Personalidad:** Permite al usuario elegir de una librería de dibujos un carácter para su robot con emociones que se pueden expresar durante la interacción. Este agente proporciona la funcionalidad para crear una interacción hombre-robot eficaz o crear robots de entretenimiento para actividades como recreación y juegos.
  - **Agente de Comunicación Hablada:** Mediante este agente el usuario puede dar al robot comandos de voz y generar voz sintetizada o sonidos.
  - **Agente de Realimentación Visual:** Provee realimentación visual al usuario en forma de modelos gráficos del entorno y también se pretende estudiar el uso de la tecnologías "*push*" (tal como *streaming video*) para proveer imágenes en tiempo real.

- Se propone estudiar el uso de otros dispositivos tales como los joysticks y los dispositivos de interfaz háptica como elementos de interacción remota basada en Internet con los robots móviles.

- Se pretende implementar la metodología de control de acceso propuesta para manejar el acceso de los usuarios al sistema. También, se pretende mejorar la realimentación visual desarrollando un modelo 3D para el laboratorio utilizando Java 3D o un modelo de realidad

virtual utilizando X-VRML. Se propone también usar las ayudas kinestéticas, como la realimentación háptica para proporcionar una realimentación física adicional. Se pretende también agregar una demo para mostrar a los usuarios cómo se puede usar el laboratorio remoto.

- Se propone el desarrollo de nuevas interfaces para otras habilidades como la habilidad SLAM (*Simultaneous Localization And Mapping*) desarrollada en el departamento. Esta habilidad es una simulación e implementación del problema de realizar la localización y el mapeado simultáneamente para robots móviles en entornos interiores.

- Facilitar la programación remota del robot mediante un editor de comandos por medio del cual los usuarios pueden enviar comandos de bajo nivel al robot para llevar a cabo varias tareas como estudiar el efecto de variar los parámetros del controlador, controlar la velocidad del robot, probar varios algoritmos de navegación, etc.

- También, se pretenden implementar las actividades de construcción locales propuestas como diseñar y construir nuevos prototipos o aplicar ingeniería inversa a sistemas existentes. Para desarrollar estas actividades, se puede pedir a los estudiantes que construyan prototipos robóticos utilizando componentes hardware como Lego o Fischertechnik como componentes del hardware y los microcontroladores como Motorola 88HC11 o 16-bit Siemens 80C167 y utilizar varios sensores disponibles para los robots móviles.

- Por último, se pretende desarrollar una herramienta gráfica, que guía la construcción de habilidades complejas utilizando el secuenciador. Dicha herramienta gráfica permite la generación de secuencias básicas para usuarios inexpertos, restringiendo su acceso y controlando los valores asignados a los parámetros de cada habilidad utilizada en la secuencia. Se propone también implementar el agente de base de datos para almacenar y gestionar los datos de una secuencia cuando se analiza en el agente del secuenciador.

Apéndice

A

DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Apéndice A DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

A.1 DISPOSITIVOS MÓVILES

A pesar del crecimiento en las tecnologías inalámbricas y los servicios en los años recientes ha habido una falta relativa de investigación en los posibles usos de estos sistemas en el control remoto de los robots móviles. Esto principalmente es una consecuencia del hecho de que el uso de estos sistemas tiene que tratar con problemas específicos relacionados con las limitaciones de los dispositivos móviles (procesadores lentos, memoria limitada, y pantallas pequeñas) y las redes inalámbricas (ancho de banda limitado, retardos altos, estabilidad de conexión baja, y disponibilidad predecible baja). Estas tecnologías se pueden usar para mejorar el alcance y la funcionalidad de los entornos de teleoperación de los robots. En este apéndice, se describen las características básicas de los dispositivos móviles y las tecnologías de desarrollo disponibles para las aplicaciones inalámbricas.

A.1.1 Teléfonos Móviles

El mundo de Internet se ha llegado más allá de los ordenadores y ha ido a parar a unos aparatos que son mucho más comunes: los teléfonos móviles. En España, la telefonía móvil superaba en septiembre de 2002 los 32,4 millones de clientes, alcanzando una tasa de penetración del 78,5%, lo que representa un incremento de más de 10 puntos porcentuales respecto a septiembre de 2001. La tasa de penetración de los móviles en España se aproxima con rapidez a los niveles de saturación [Crisando,02]. El número total de teléfonos móviles registrados en el mundo en el año 2003 se estima que supere los 1000 millones. Esta cifra contrasta con el número de ordenadores personales instalados en todo el mundo en el año 2000 que era de 311 millones [Torres,03].

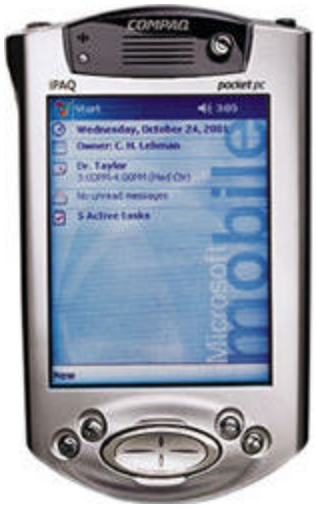
Como consecuencia de este rápido crecimiento, se han desarrollado varias tecnologías que permiten desarrollar aplicaciones útiles para mejorar la calidad de vida. Utilizar los teléfonos móviles como elementos de interacción con los robots puede ampliar el uso de los robots en la vida diaria. En el apartado A.2 se discuten las tecnologías, que se pueden utilizar para llevar a cabo aplicaciones inalámbricas.

A.1.2 Asistentes Digitales Personales (PDA)

Las asistentes digitales personales (*PDA - Personal Digital Assistant*) pueden ser útiles para las aplicaciones que requieren control remoto como las aplicaciones de los robots u otros equipos de automatización. Estos dispositivos son totalmente configurables según las necesidades del usuario. Actualmente existe una gran cantidad de software en Internet para instalar en las PDAs (juegos, bases de datos, calculadoras,...), y tienen muchos accesorios, como fundas, lápices, módems, memoria Compact Flash, etc. Las PDAs, además, tienen muchas otras propiedades interesantes, como la de poder conectarse a un PC para intercambiar información.

Las PDAs disponibles varían en apariencia, funcionalidad y programas disponibles. En una encuesta entre 344 consumidores realizada el año 2003, Compaq iPaq 3870 ha tenido la clasificación más alta entre las PDAs más vendida en el mercado [Ciao,03]. Los criterios de la encuesta han sido la confianza de la marca, la facilidad de uso, la memoria/capacidad y la robustez /durabilidad.

Se ha estudiado el uso de las PDAs como iPAQ de Compaq 3870 PDA, que tiene las características mostradas en la figura A.1, como un dispositivo de interacción con el robot B21.



- Procesador 206-MHz Intel StrongARM SA-1110 32-bit RISC.
- Memoria 64 Mb RAM / 32 Mb ROM Posibilidad de ampliación de memoria. Admite tarjetas de memoria MMC (Multimedia Card) y SD (Secure Digital) Mediante módulos de expansión Compact Flash o PCMCIA.
- Pantalla táctil TFT con 65.536 colores de 16 bit. Resolución de 240 x 320 píxeles. Ancho de pixel: 0.24 mm Ajuste automático de iluminación.
- Sistema Operativo: Microsoft Pocket PC 2002.
- Medidas: 133 x 84 x 16 mm.
- Alimentación: Batería recargable 1400 mAh Lithium Polymer.

Fig. A.1: Características de Compaq iPAQ 3870

Para poder utilizar interfaces Java con este dispositivo, hay que instalar Jeode*Runtime* que es una implementación certificada de la especificación PersonalJava 1.2 de Sun [Insignia,03]. Se usa Jeode*Runtime* como conexión del Pocket Internet Explorer del iPAQ, para ejecutar subprogramas

de Java desde una página Web. O como máquina virtual de Java independiente, para ejecutar aplicaciones de Java en iPAQ.

Se pueden resumir las características básicas de JeodeRuntime en los siguientes puntos:

- **Ejecución de subprogramas de Java:** La conexión de Jeode para Pocket Internet Explorer se incluye al instalar Jeode*Runtime.* La próxima vez que el explorador abra una página Web que contenga un subprograma de Java lo ejecutará. Todo esto sucede automáticamente, sin necesidad de que el usuario haga nada.
- **Subprogramas en archivos CAB**: Actualmente, Jeode*Runtime* no admite la carga de subprogramas de archivos CAB de Microsoft, ya que los archivos CAB no forman parte de la tecnología estándar de Java. Si el sitio Web asume (según el tipo de explorador) el comportamiento de la máquina virtual que se está usando, puede provocar problemas en la carga del subprograma.
- **Subprogramas en archivos JAR o ZIP:** Jeode*Runtime* admite el uso de subprogramas contenidos en archivos JAR o ZIP. Sin embargo, debido a que Pocket Internet Explorer es compatible con la Especificación HTML 3.2 , éste no admite el uso del atributo ARCHIVE de la etiqueta APPLET en la página HTML que llama al subprograma; por lo tanto, en esas circunstancias Pocket Internet Explorer no admitirá la carga de subprogramas de archivos JAR o ZIP. Si se puede rescribir el código HTML de la página Web que llama al subprograma, se pueden usar los archivos JAR y ZIP especificando el contenedor como nombre de parámetro, como atributo ARCHIVE (mediante <param name=archive>). Por ejemplo: `<param name=archive value=myclasses.jar>`
- **Fallo al ejecutar el applet:** Puede ver un mensaje del tipo:

  ```
  Failed to run applet. Please Reload
  ```

  Esto significa que la conexión ha detectado una condición que impide la ejecución del subprograma. Hay varias posibilidades: un problema habitual es la falta de memoria en el iPAQ, ya que algunos subprogramas están escritos asumiendo que habrá una cantidad ilimitada de memoria. La conexión Jeode no tiene control sobre los requisitos de memoria del subprograma. Si no queda suficiente memoria para que la conexión siga ejecutándose, emitirá un mensaje y se cerrará.
- **Jeode*Runtime* con JavaScript:** Mientras que Java sólo lo usan programadores para crear objetos y subprogramas nuevos, JavaScript está diseñado para que lo usen autores de páginas HTML para secuenciar dinámicamente el comportamiento de objetos que se ejecutan en el cliente o en el servidor. JavaScript y Java son lenguajes de programación completamente distintos; por lo tanto, Jeode no se utiliza para ejecutar JavaScript. Además, la conexión Jeode actualmente no admite las comunicaciones entre JavaScript y subprogramas de Java. Eso significa que los subprogramas que requieren esta interacción no funcionarán con Jeode*Runtime.*
- **Jeode*Runtime* con Swing:** Lamentablemente, Swing 1.1.1 no es compatible con PersonalJava 1.2; por lo tanto, Jeode*Runtime* no admitirá Swing 1.1.1. Por está razón, sólo se puede utilizar Java AWT en lugar de Java Swing para el desarrollo de la interfaz.

A.2 TECNOLOGÍAS DE DESARROLLO

Las dos tecnologías principales para el desarrollo de aplicaciones inalámbricas son la tecnología WAP y el uso de la edición micro de java 2 (J2ME). A continuación se describe el uso de las dos tecnologías para desarrollar aplicaciones inalámbricas.

A.2.1 Tecnología Java

En Junio de 1998, comenzó el proyecto *Spotless* en los laboratorios de *Sun Microsystems* para investigar el uso de Java en dispositivos pequeños [Bush,99]. El logro de este proyecto fue construir un entorno Java capaz de ejecutarse en un espacio igual a la décima parte de un tamaño estándar. Otro logro fue construir una máquina virtual Java con las siguientes características:

- Tamaño pequeño.
- Portabilidad.
- Facilidad de uso y lectura del código fuente.

El tamaño reducido es importante, ya que la mayoría de los dispositivos móviles tiene capacidades reducidas y a menudo sólo unas pocas decenas o centenas de Kbytes de memoria disponibles para las aplicaciones. La mayoría de los fabricantes de estos dispositivos tienen decenas o quizá centenas de diferentes configuraciones, y sería muy costoso el desarrollo para todas las configuraciones [Riggs,01].

Cuando el proyecto estuvo iniciado, el grupo de proyecto estableció contactos con clientes. Los clientes externos, especialmente Motorola jugó un papel significativo para convencer a Sun para que este proyecto se convirtiera en un producto comercial. La versión del producto de la máquina virtual *Spotless* es hoy en día conocida como máquina virtual K o máquina virtual Kjava.

A.2.1.1 Edición Micro de Java (J2ME)

La edición micro de Java 2 (J2ME - Java 2 Micro Edition) es una versión de Java estándar (J2SE) para dispositivos con capacidades hardware y software mucho más limitadas que los PC como, las PDAs, teléfonos móviles, electrodomésticos inteligentes, etc.

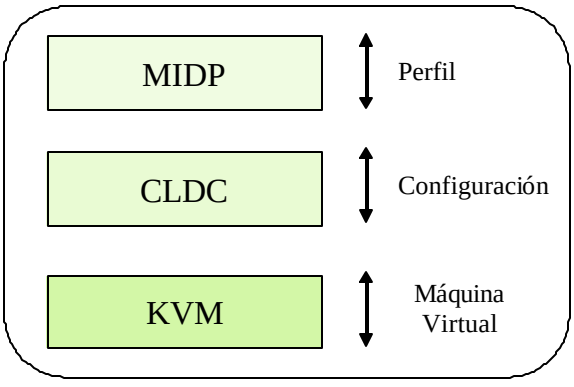


Fig. A.2: Componentes de J2ME

J2ME se compone de una selección de los componentes siguientes como se muestra en la figura A.2:

- **Maquina Virtual Reducida (KVM):** KVM está específicamente definida para dispositivos pequeños. Se puede ejecutar KVM con microprocesadores de 16/32 bits, además la memoria mínima para implementar KVM está sobre 128 KB incluyendo la máquina virtual, librerías mínimas y algo de espacio para ejecutar aplicaciones Java. Las aplicaciones más típicas requieren 256 Kbytes, de los cuales al menos 32 Kbytes son utilizados para aplicaciones, 60 u 80 Kbytes para la máquina virtual y el resto está reservado para librerías [Riggs,01].
- **Configuración CLDC:** La especificación CLDC (*Connected Limited Device Configuration*) tiene como objetivo definir el "denominador común más bajo" de la plataforma Java para una gran variedad de dispositivos pequeños. Esta configuración define los componentes y las librerías Java mínimas para estos pequeños dispositivos. El lenguaje Java y las características de máquina virtual, librerías, entrada/salida, red y seguridad son los términos primarios tratados por la especificación CLDC. El estándar CLDC define un bloque de propósito general para el perfil de la categoría de dispositivos definidos.
- **Perfiles Específicos para los Diferentes Dispositivos:** El perfil MIDP (*Mobile Information Device Profile*) está basado en la plataforma definida por la estandarización CLDC, añadiendo características y API's que están especialmente enfocados a la comunicación de dos direcciones de dispositivos inalámbricos. Módulo de aplicación, interfaz de usuario, red y almacenamiento de API's son las áreas primarias de la especialización MIDP.

A.2.1.2 Modelo de Programación

Como se muestra en la figura A.3, este modelo es una mezcla del modelo de programación Java y el modelo de programación Web [Mahmoud,02].

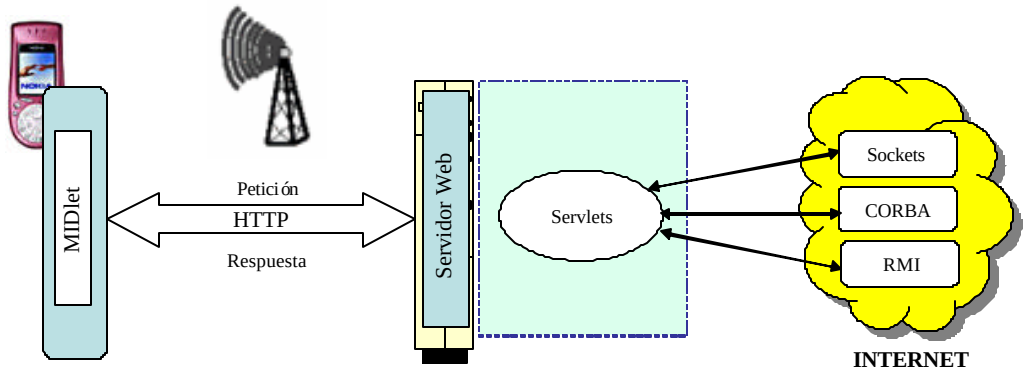


Fig. A.3: Modelo de Programación MIDP

Un MIDlet es un pequeño programa J2ME que implementa la configuración CLDC y el perfil MIDP, al que deben su nombre. Está formado por un fichero Descriptor Java (.JAD) y un fichero ejecutable (.JAR). Un MIDlet se descarga en un terminal y puede ejecutarse localmente sin

conexión, o bien puede conectarse con otros elementos de la red, mediante GSM, GPRS o UMTS, facilitando el intercambio de información a través de la red y el mantenimiento de una relación cliente-servidor con algún servidor remoto. La tecnología J2ME permite que los MIDlets guarden de forma persistente datos en el terminal, por lo que sólo se navega cuando es realmente necesario, utilizando la red de la forma más eficiente posible.

De forma similar a los Applets, donde un Applet está descrito en un fichero HTML, un MIDlet o grupos de MIDlets (conocidos como MIDlet suite) está descrito en un fichero JAD. Mientras que los Applets se ejecutan en un navegador Web, los MIDlets se ejecutan en un software de manejo de MIDlet, que está preinstalado en los dispositivos MIDP y que proporciona un entorno operativo para KVM y MIDlet. A diferencia de los Applets, los MIDlet no se destruyen cuando finaliza su ejecución, se mantienen instalados en el dispositivo hasta que sean expresamente borrados. Así se mantienen disponibles para usarse *off-line* ya que MIDP soporta operaciones sin conexión. Esto es una ventaja para aplicaciones de entretenimiento como los juegos.

En la presente tesis, se ha utilizado el emulador *J2ME Wireless Toolkit 1.0.4* [Sun,02] para desarrollar y probar las interfaces J2ME. Este emulador soporta el desarrollo de aplicaciones Java que funcionan en dispositivos MIDP, tales como teléfonos celulares o PDAs. Como se muestra en la figura A.4, la herramienta *KToolBar* incluida en *J2ME Wireless Toolkit* es el entorno mínimo de desarrollo para compilar, empaquetar y ejecutar aplicaciones MIDP.

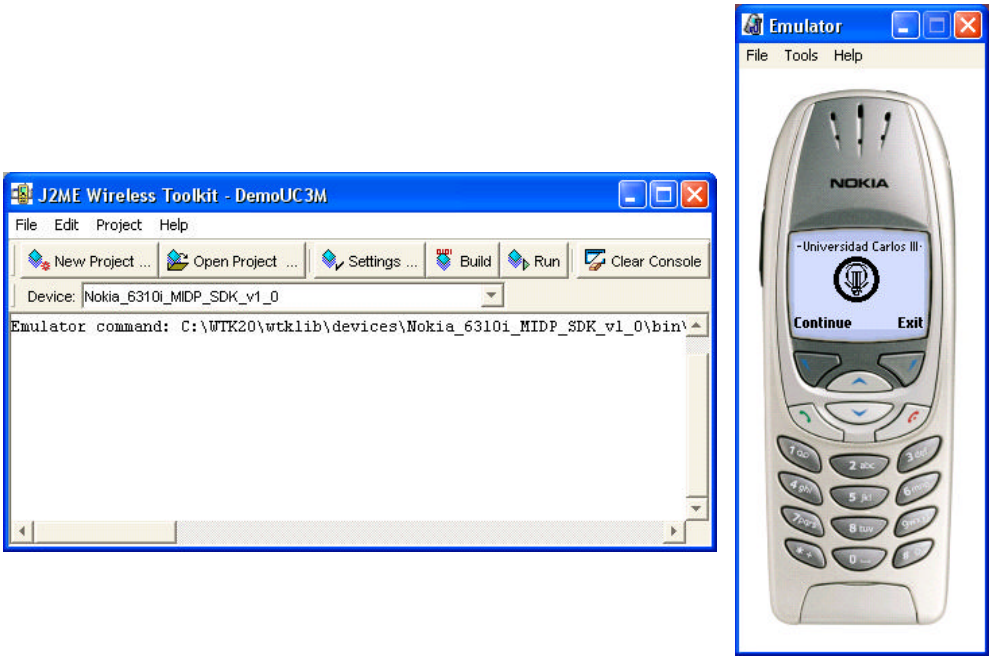


Fig. A.4: J2ME Wíreless Toolkit

Los dispositivos permiten descargar aplicaciones escritas en J2ME vía cable o utilizando la técnica *Over The Air (OTA)* [Giguere,03]. Actualmente los MIDlets no se pueden descargar con la técnica OTA directamente. Para facilitar esta descarga se necesita alguna clase de entornos en los dispositivos que permita a los usuarios introducir una URL para un MIDlet, por ejemplo. Este entorno puede muy bien ser un navegador WAP. Similar a los Java Applets que están integrados en HTML, los MIDlets pueden integrarse en paginas WML. La página WML puede entonces ser llamada desde un navegador WAP y los MIDlets habilitados consiguen instalarse en el

dispositivo. Para facilitar esto se necesita un navegador WAP que soporte descarga OTA en el dispositivo.

A.2.2 Tecnología WAP

La tecnología WAP permite acceder a ciertos contenidos Web mediante dispositivos móviles que estén preparados para ello. En los apartados siguientes, se presentan el protocolo WAP y el modelo de programación de esta tecnología.

A.2.2.1 Protocolo WAP

El Protocolo de Aplicaciones Inalámbricas o *Wireless Application Protocol* (WAP) es una especificación abierta que maneja las características de las redes inalámbricas para adaptarlas a los diferentes dispositivos inalámbricos usando protocolos Web, o introduciendo algunos nuevos. La reutilización de las tecnologías Web existentes reduce el tiempo de desarrollo de aplicaciones WAP y este tiempo es similar al tiempo de desarrollo de aplicaciones Web basadas en HTML [WAP,98].

Se ha utilizado WapIDE 3.2.1 como herramienta para desarrollar una interfaz WAP para el control directo del robot. Esta herramienta es un entorno de programación de libre distribución, que permite desarrollar y probar aplicaciones WAP reales [Ericsson,02]. La figura A.5 muestra el entorno WapIDE 3.2.1.

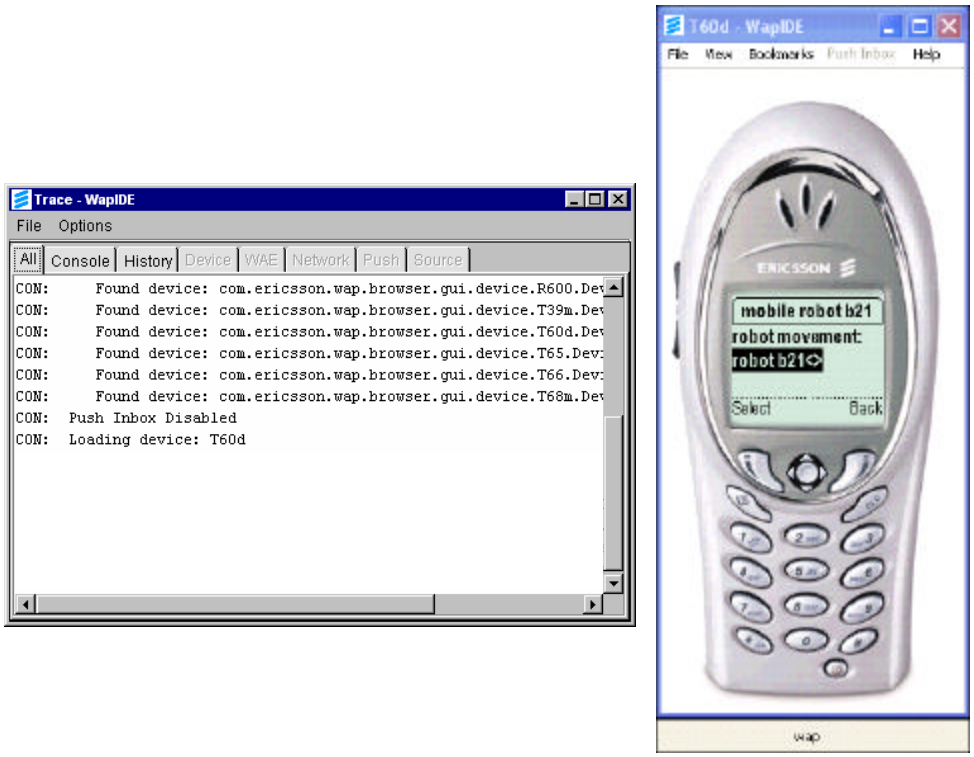


Fig. A.5: WapIDE 3.2.1

A.2.2.2 Modelo de Programación WAP

Este modelo es similar al modelo de programación Web con modificaciones para adaptarse a las características del entorno inalámbrico [Mahmoud,02]. La figura A.6 muestra el modelo de programación WAP.

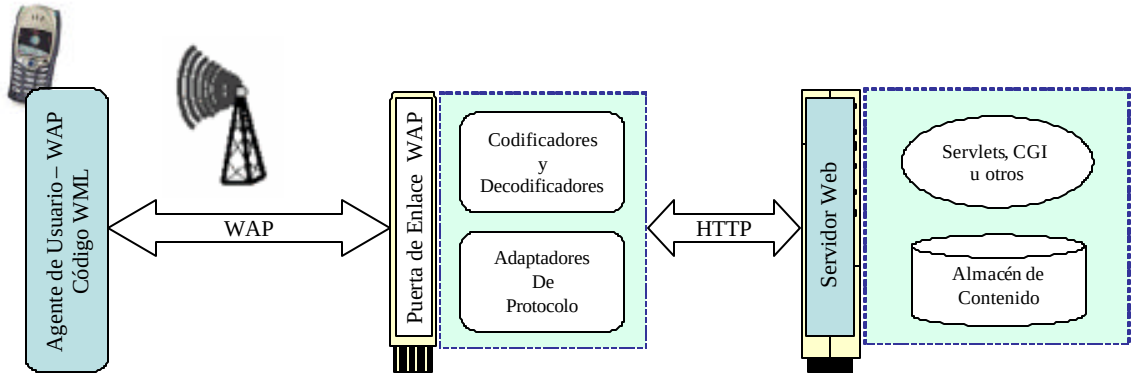


Fig. A.6: Modelo de Programación WAP

Como puede verse el modelo de programación WAP está basado en el modelo de programación Web. Si bien WAP no se ha diseñado para usar HTML, en algunos casos, los servicios de datos localizados en el servidor Web están basados en HTML. Algunas puertas de enlace WAP (*WAP gateways*) son capaces de convertir páginas HTML a un formato tal que se puedan visualizar en dispositivos inalámbricos. A su vez HTML no fue diseñado para pantallas pequeñas, por lo que el protocolo WAP define su propio lenguaje, *Wireless Markup Language* (WML), el cual se une al estándar XML y está diseñado para habilitar aplicaciones potentes sin las restricciones de los dispositivos móviles.

A.2.3 Comparativa

Existen muchos factores a favor de usar la tecnología Java para desarrollar aplicaciones inalámbricas. Entre ellos cabe citar la compatibilidad de plataforma, la elección dinámica de aplicaciones y servicios, la seguridad y la disponibilidad de documentación y soporte técnico. Los siguientes apartados discuten con más detalles las diferencias entre las dos tecnologías.

A.2.3.1 Navegación

La navegación en aplicaciones WAP se basa en el uso de la etiqueta <A>, la cual aparece como una opción del menú y puede seleccionar el movimiento a la siguiente carta. De otra manera la navegación entre pantallas de los MIDlets se puede realizar también usando comandos. Los comandos en MIDP son similares a la etiqueta <A> ya que se implementan en las teclas programables. A diferencia de MIDP, WAP proporciona una mala navegación y un mal modelo de interacción con algunos de sus rutas de navegación ocultos.

Para dar una apariencia profesional, se pueden asociar iconos con menajes de error y confirmación. Se puede hacer en MIDP y WAP. WAP soporta formato Wireless bitmap (WBMP) y MIDP soporta el formato no patentado Portable Network Graphics (PNG). Las figuras A.7 y A.8 muestran la navegación a través de una interfaz.

Fig. A.7: Navegación por la interfaz WML

Fig. A.8: Navegación por la interfaz J2ME

WAP y MIDP resuelven problemas similares pero cada uno toma un par de soluciones distintas. Hay características especiales que están disponibles en WAP pero no en MIDP y viceversa.

- WAP tiene soporte de funcionalidades adicionales para teléfonos tales como configuración e integración con agendas. A pesar de que no todos los teléfonos WAP soportan esta característica, no hay APIs equivalentes disponibles para MIDP. Es posible que estén disponibles en una futura versión de MIDP.
- MIDP tiene APIs gráficas de alto nivel (tales como Form, List, Choice Group y otros). Proporciona también APIs de bajo nivel que habilitan a los programadores a tener control sobre cada píxel de la pantalla del dispositivo.
- En aplicaciones de entretenimiento, los MIDlets existen en un dispositivo hasta que son explícitamente eliminados, permitiendo así a los usuarios, ejecutarlos aunque no haya conexión con el servidor.
- WML proporciona etiquetas y atributos de presentación pero no define un modelo de interacción. Por ejemplo WML define un elemento SELECT para proporcionar una lista. Algunos dispositivos WAP interpretaran la etiqueta como una lista de menú, mientras que otros lo interpretaran como un menú que puede ser usado para navegación. No hay un modelo estándar de interacción definido para este elemento. Si un desarrollador lo usa, la aplicación se ejecutará bien en unos dispositivos y mal en otros. Los MIDlets proporcionan un estándar claramente definido para la interacción usando comandos.

A.2.3.2 Factores a Favor de Java

Los proveedores y fabricantes de productos inalámbricos citan cinco factores que conducen a usar la tecnología Java en dispositivos inalámbricos [Hardee,00].

- **Elección dinámica de aplicaciones y servicios:** A diferencia de la mayoría de los dispositivos de hoy, los dispositivos de próximas generaciones serán capaces de descargar aplicaciones – seguramente- en tiempo real.
- **Compatibilidad de plataforma:** Ya que las aplicaciones escritas para la tecnología Java se ejecuta en múltiples dispositivos. Se puede ejecutar idénticos servicios en PDAs y teléfonos móviles. Se puede descargar la misma aplicación, escrita con especificaciones CLDC y MIDP, en un teléfono Motorola o en un Nokia y funcionarán exactamente igual. Todos estos dispositivos ejecutan diferentes sistemas operativos, tienen diferentes microprocesadores y en algunos casos diferentes protocolos de red. Esta compatibilidad es extremadamente importante para fabricantes, proveedores de contenidos, etc.
- **Experiencia de usuario:** Los programadores pueden escribir aplicaciones más ricas y más útiles utilizando tecnología Java que con entornos basados en navegadores. Las aplicaciones Java tienen gráficos más ricos, con interacción más rápida. Existen ejemplos que incluyen mapas de ciudades descargables, juegos y venta de entradas de conciertos. Todos ejecutan protocolos de diferentes vendedores. Se proporcionan APIs para habilitar a los programadores para crear rápidamente componentes de trabajo.
- **Acceso sin conexión:** Las aplicaciones de teléfonos de tecnología Java, se pueden ejecutar cuando el teléfono está desconectado o fuera de cobertura. Cuando se usa una aplicación en un dispositivo WAP se necesita estar conectado todo el tiempo, si se está fuera de cobertura ya no funciona.
- **Seguridad:** La próxima generación de teléfonos trabajará sobre TCP/IP y es más fácil escribir aplicaciones Java compatibles para trabajar sobre TCP/IP. Con la nueva generación de teléfonos se introducirá un nuevo nivel de seguridad para el mundo inalámbrico. El comercio móvil actualmente se basa en *i-mode* y WAP pero se hará realidad cuando esté en TCP/IP y exista más seguridad. Hoy los teléfonos WAP proporcionan y visualizan datos vía micronavegador, pero necesitan una puerta de enlace para hacer conversiones de protocolo entre protocolos orientados a Internet (TCP/IP, SSL. etc.) y la red inalámbrica. La puerta de enlace es necesaria porque convierte el protocolo entre el teléfono y el servidor y aquí hay una brecha en la seguridad. La seguridad de la plataforma Java es muy importante, ya que dispone de verificación bytecode y cada vez que se carga la aplicación, asegura la integridad del código. Así se asegura que no tenga virus y que la aplicación funcione bien, lo que es muy importante para operaciones de red.

Apéndice

B

ARQUITECTURA AD

Apéndice B ARQUITECTURA AD

B.1 ARQUITECTURA AD

La arquitectura AD (Automática-Deliberativa) es una arquitectura híbrida propuesta por R. Barber para el control de los robots móviles autónomos [Barber,00][Barber,01]. Está basada en la capacidad de razonar y de actuar de los seres humanos. Como se muestra en la figura B.1, los dos niveles que forman la arquitectura son el Nivel Automático y el Nivel Deliberativo.

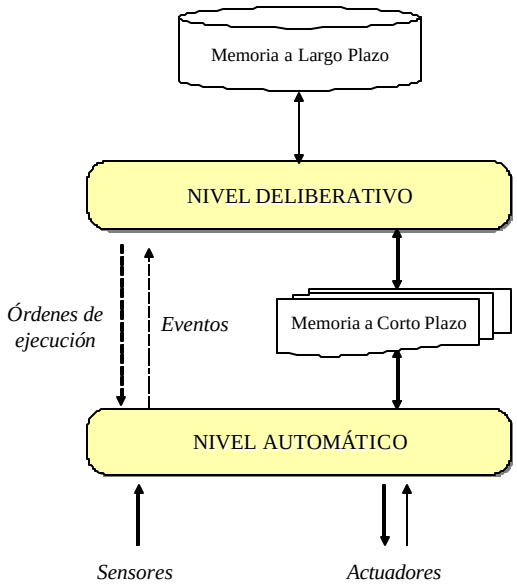


Fig. B.1: Niveles de la Arquitectura AD

Ambos nivel de la arquitectura presentan las mismas características: están formados por habilidades. Las habilidades son las diferentes capacidades para razonar o llevar a cabo una acción. La comunicación entre el Nivel Deliberativo y el Automático es bidireccional. El nivel Deliberativo envía órdenes de ejecución al Nivel Automático y este le devuelve eventos.

La arquitectura considera también como forma de intercambio de información diversos tipos de memoria. La memoria a corto plazo o memoria de trabajo aparece como un elemento de intercambio de información entre los dos niveles de la arquitectura. Esta se diferencia de la memoria a largo plazo porque la información almacenada es la información del estado del robot, mientras que en la memoria a largo plazo los datos almacenados pueden ser considerados estables en el tiempo.

B.1.1 Nivel Automático

El Nivel Automático se encarga del control de los dispositivos con los que el robot interactúa con el entorno: sensores y actuadores. Este nivel permite al robot tener la reactividad necesaria para responder rápidamente a cambios producidos en su entorno. La figura B.2 muestra los elementos que constituyen este nivel.

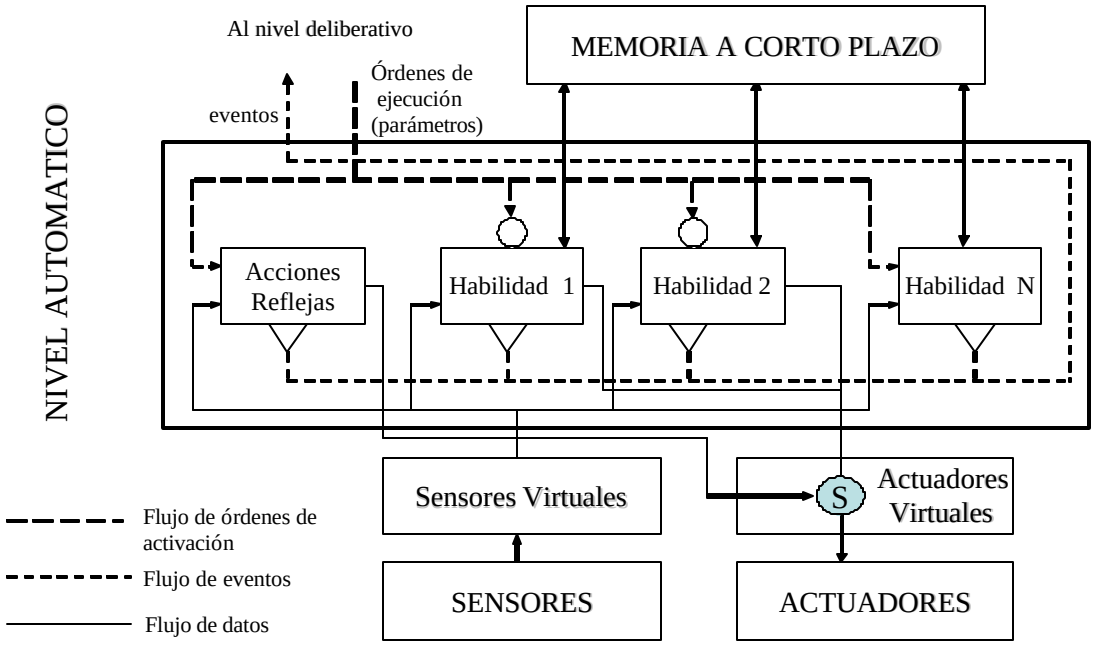


Fig. B.2: Nivel Automático de la Arquitectura AD

Dichos elementos son:

- **Sensores y Actuadores Virtuales:** El Nivel Automático se comunica con el hardware del robot a través de los sensores y actuadores virtuales.
- **Acciones Reflejas:** Son las respuestas involuntarias y prioritarias a los estímulos.
- **Habilidades Automáticas:** Son las capacidades sensoriales y motoras del sistema y son la base del concepto de comportamiento.

A continuación, se explican con más detalles las acciones reflejas y las habilidades automáticas.

B.1.1.1 Acciones Reflejas

Son respuestas prioritarias e involuntarias a un determinado estímulo. En estas acciones reflejas existe una *conexión directa* entre percepción y acción sin necesidad de un modelo del mundo. Se procesa la información sensorial en tiempo real y se envían con carácter prioritario órdenes de ejecución a otras habilidades, o comandos de movimiento a los actuadores.

B.1.1.2 Habilidades Automáticas

Las habilidades automáticas son definidas como la capacidad de procesar información procedente de los sensores virtuales y/o generar órdenes de ejecución sobre los actuadores virtuales.

Las habilidades automáticas se pueden clasificar en función de los tipos de datos que obtienen como resultados y de cómo interactúan con los dispositivos hardware del robot, es decir, sensores y actuadores. Según este tipo de clasificación las habilidades pueden ser:

- **Habilidades Perceptivas:** Son aquellas que interpretan la información procedente de los sensores o de otras habilidades, y que no generan comandos de movimiento como la habilidad "Detectar Obstáculos" con el sonar o con el láser y la habilidad "Detectar Puerta".
- **Habilidades Sensorimotoras:** Además de interpretar la información sensorial o de otras habilidades, se encargan de enviar órdenes de movimiento al robot en determinadas circunstancias. Para ello se deben conectar al objeto de datos del actuador virtual correspondiente y actualiza sus valores. Las habilidades "Ir a Punto", "Girar" y "Evitar Obstáculos" representan habilidades sensorimotoras.

Todas las habilidades automáticas tienen las siguientes características:

- Pueden ser activadas por habilidades situadas en el mismo nivel o en el nivel deliberativo. Una habilidad sólo puede desactivar habilidades que ella mima ha activado previamente.
- Deben almacenar los resultados de forma tal que puedan ser utilizados por otras habilidades.
- Pueden generar eventos diversos y los notificarán únicamente a aquellas habilidades que los hayan pedido previamente.

B.1.2 Nivel Deliberativo

En el Nivel Deliberativo de la arquitectura AD se encuentran los módulos que requieren capacidad de razonamiento. Estos módulos no proporcionan respuestas inmediatas debido a que necesitan procesar la información para planear la acción o acciones a ejecutar. En la figura B.3 se puede ver un esquema de este nivel.

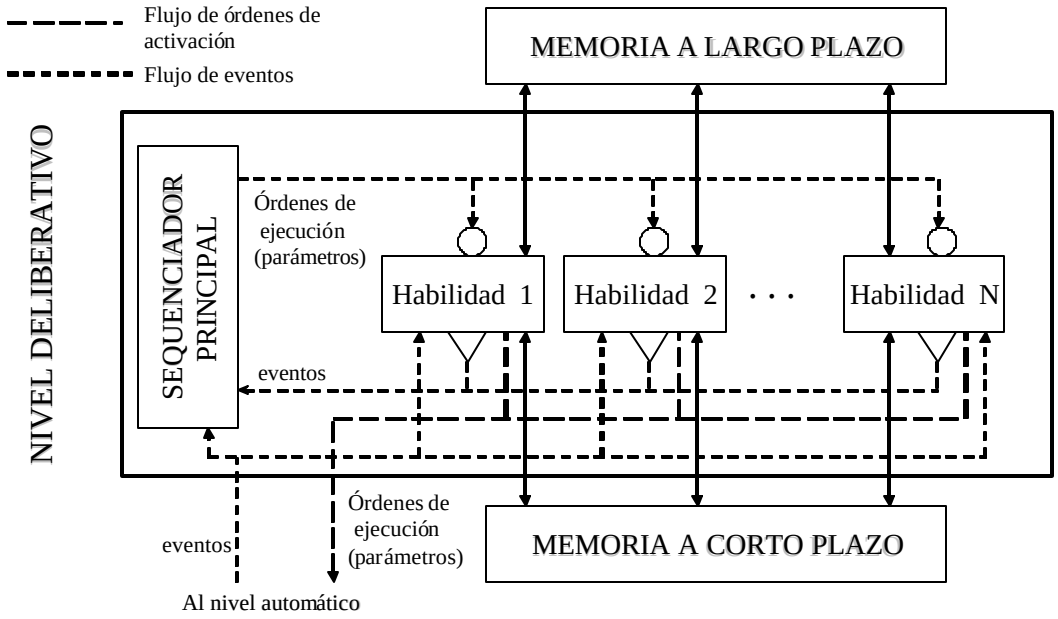


Fig. B.3: Nivel Deliberativo de la Arquitectura AD

A continuación se explican los elementos que forman parte de este nivel.

B.1.2.1 Habilidades Deliberativas

Las habilidades deliberativas son cada una de las capacidades de las capacidades de razonamiento y aprendizaje de las que dispone el sistema autónomo. Estas habilidades se encargan tanto de gestionar y modificar la memoria a largo plazo como de gestionar y secuenciar las habilidades del Nivel Automático. Algunos ejemplos de estas habilidades pueden ser los planificadores, los supervisores, los módulos que gestionan el mapa, los módulos que realizan exploración, los módulos de modelado del entorno, los módulos de relocalización y los módulos que gestionan las comunicaciones con el usuario.

Estas habilidades deliberativas se caracterizan por:

- **Carácter secuencial**. Las actividades deliberativas se llevan a cabo secuencialmente. Es decir, no es posible estar reflexionando sobre más de una cosa al mismo tiempo.
- **Carácter temporal**. Para razonar es necesario tener en cuenta las acciones pasadas a fin de adquirir experiencia, e imaginarse posibles acciones futuras a fin de prever las consecuencias de cada una de ellas.
- **Adaptabilidad**. El análisis de forma deliberativa permite determinar cual es la respuesta más adecuada ante situaciones nuevas que se presentan y, si no se elige la repuesta más conveniente, se tendrá en cuenta para la siguiente vez que se produzca la misma situación, adaptando la forma de actuar, es decir, aprendiendo.
- **Velocidad de respuesta lenta**. Las habilidades deliberativas requieren gran cantidad de tiempo para el análisis y razonamiento.

- **Las habilidades deliberativas pueden pasar a ser automáticas**. Cuando se ejecuta por primera vez una tarea, el nivel Deliberativo es consciente de los movimientos que realiza, prestando atención de todo lo que le rodea y de los resultados obtenidos. Una vez que el sistema aprende dicha tarea, su ejecución se realiza de manera automática. Se puede asemejar a lo que ocurre al aprender a conducir. Al principio los movimientos son lentos y requieren de razonamiento, para luego convertirse en movimientos más rápidos y menos dependientes de los procesos mentales, pasando incluso a ser reflejos.

B.1.2.2 Memoria a largo plazo

La memoria a largo plazo contiene información que puede ser considerada más estable a lo largo del tiempo, es decir, no dependiente del estado del robot. A dicha memoria sólo tiene acceso las habilidades deliberativas, las cuales pueden realizar razonamientos sobre dicha información, modificando la información cuando sea necesario. En ella se incluirán la información a priori, como pueden ser los mapas, e información procedente del razonamiento o aprendizaje de las distintas habilidades deliberativas.

B.1.2.3 Secuenciador Principal

El secuenciador se encarga de gestionar las habilidades deliberativas, dando la orden de ejecución de cada una de ellas en el momento oportuno. Este secuenciador viene dado a priori y es el que define el comportamiento del sistema. El secuenciador va decidiendo que habilidades debe activar en función de su secuencia y de los eventos que le vayan llegando tanto de cada una de las habilidades de este nivel como del nivel automático.

A parte de este secuenciador, la arquitectura contempla la posibilidad de otros secuenciadores tanto en habilidades deliberativas que permiten activar habilidades del Nivel Automático, así como secuenciadores dentro de las habilidades automáticas que permiten formar otras habilidades más complejas.

Apéndice

C

DESARROLLO DE LABORATORIOS REMOTOS

Apéndice C

DESARROLLO DE LABORATORIOS REMOTOS

C.1 INTRODUCCIÓN

Los laboratorios remotos se desarrollan basándose en arquitecturas similares y se implementan utilizando distintas tecnologías. En este apéndice, se discuten los sistemas distribuidos y la arquitectura cliente-servidor utilizada para desarrollar este tipo de sistemas. Se discuten también las tecnologías disponibles para implementar sistemas basadas en Web.

C.2 SISTEMAS DISTRIBUIDOS

Un sistema distribuido consiste en un conjunto de ordenadores autónomos enlazados por medio de una red y que disponen de un software común que les permite coordinar las actividades y compartir recursos del sistema, con lo que desde el punto de vista del usuario se ve como si fuera un sistema único y centralizado [Renaud,96]. Las características por las que se distinguen estos sistemas son:

- **Recursos compartidos**. Los datos y periféricos son compartidos en el sistema reduciéndose costes y evitando problemas de duplicidad en los datos.
- **Modularidad**. El sistema se compone de módulos autónomos relacionados entre si por medio de la red y del software del sistema.
- **Concurrencia**. La capacidad de multiproceso es necesaria tanto en el uso de una aplicación por varios usuarios como en la ejecución en paralelo de varios procesos.
- **Escalabilidad**. El sistema puede aumentar la magnitud de su entorno sin que la funcionabilidad se pierda.

- **Tolerancia a fallos**. Los fallos en los módulos no deben repercutir en el correcto funcionamiento del sistema distribuido. Deben existir mecanismos que minimicen la influencia como redundancia de hardware y programas capaces de recuperarse de fallos.
- **Transparencia**. El sistema distribuido se entiende como un todo en vez de un conjunto de componentes independientes. Se trata de un grado de abstracción donde no se ve cómo se hace sino qué hace de forma global.

Las ventajas que ofrece un sistema distribuido frente a uno centralizado son:

- **Economía**: Los microprocesadores ofrecen mejor relación precio/rendimiento que los *mainframes*.
- **Velocidad**: Mayor capacidad de procesamiento.
- **Distribución inherente**: Aplicaciones en maquinas separadas geográficamente.
- **Fiabilidad**: El sistema no se cae al fallar una maquina.
- **Desarrollo Incremental**: Se puede añadir nuevas unidades sin coste adicional al de la nueva unidad.

Frente a un ordenador aislado destacan por:

- **Datos compartidos**: El usuario accede a una base de datos común.
- **Periféricos compartidos**: Los usuarios comparten los periféricos caros reduciendo costes.
- **Comunicación**: Facilita la comunicación persona a persona.
- **Flexibilidad**: La carga se distribuye entre las maquinas disponibles de forma eficaz.

Las principales desventajas de los sistemas distribuidos son:

- **Software**: Existe poca variedad de software para estos sistemas. Suele ser a medida.
- **Redes**: Se pueden saturar y causar la caída del sistema.
- **Seguridad**: Aunque cada vez se mejora más en este aspecto, el riesgo de sufrir intrusiones externas existe debido a que la gran mayoría de los sistemas tienen acceso a Internet, donde la seguridad es reducida.

Existen distintas limitaciones que crean problemas tecnológicos en los sistemas distribuidos, como la no existencia de una memoria global sino memorias locales. En este caso, es difícil ver el estado global del sistema y no se puede asegurar un tiempo global de respuesta.

C.3 ARQUITECTURA CLIENTE-SERVIDOR

La arquitectura Cliente-Servidor es un modelo para el desarrollo de sistemas de información, en el que las transacciones se dividen en elementos independientes que cooperan entre sí para intercambiar información, servicios o recursos [Orfali,98].

En este tipo de arquitecturas el ordenador de cada uno de los usuarios llamado cliente, inicia un proceso de diálogo: produce una demanda de información o solicita recursos. El ordenador que responde a la demanda del cliente, se conoce como servidor. Bajo este modelo cada usuario tiene la libertad de obtener la información que requiera en un momento dado proveniente de una o varias fuentes locales o distantes y de procesarla según le convenga. Los distintos servidores también pueden intercambiar información dentro de esta arquitectura.

Los clientes y los servidores pueden estar conectados a una red local o una *Intranet* o Internet. Cliente-Servidor es el modelo de interacción más común entre aplicaciones en una red. No forma parte de los conceptos de Internet como los protocolos IP, TCP o UDP, sin embargo todos los servicios estándares de alto nivel propuestos en Internet funcionan según este modelo.

Se puede decir que la arquitectura cliente-servidor es la integración distribuida de un sistema en red, con los recursos, medios y aplicaciones definidos modularmente en los servidores, que administran, ejecutan y atienden las solicitudes de los clientes; todos interrelacionados física y lógicamente, compartiendo datos, procesos e información; estableciendo así un enlace de comunicación transparente entre los elementos que conforman la estructura.

C.3.1 Características

Entre las principales características de la arquitectura cliente-servidor, pueden destacarse las siguientes:

- El servidor presenta a todos sus clientes una interfaz única y bien definida.
- El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- Los cambios en el servidor implican pocos o ningún cambio en el cliente.

Todos los sistemas desarrollados en arquitectura cliente-servidor poseen las siguientes características distintivas de otras formas de software distribuido:

- **Servicio:** El servidor es un proveedor de servicios; el cliente es un consumidor de servicios.
- **Recursos compartidos:** Un servidor puede atender a muchos clientes al mismo tiempo y regular su acceso a recursos compartidos.
- **Protocolos Asimétricos:** La relación entre cliente y servidor es de muchos a uno; los clientes solicitan servicios, mientras los servidores esperan las solicitudes pasivamente.
- **Transparencia de ubicación**: El software cliente-servidor siempre oculta a los clientes la ubicación del servidor.
- **Mezcla e igualdad:** El software es independiente del hardware o de las plataformas de software del sistema operativo; se puede tener las mismas o diferentes plataformas de cliente y servidor.
- **Intercambio basados en mensajes:** Los sistemas interactúan a través de un mecanismo de transmisión de mensajes: la entrega de solicitudes y respuestas del servicio.

- **Encapsulamiento de servicios:** Los servidores pueden ser sustituidos sin afectar a los clientes, siempre y cuando la interfaz para recibir peticiones y ofrecer servicios no cambie.
- **Facilidad de escalabilidad:** Los sistemas cliente-servidor pueden escalarse horizontal o verticalmente. Es decir, se pueden adicionar o eliminar clientes (con apenas un ligero impacto en al rendimiento del sistema); o bien, se puede cambiar a un servidor más grande o a servidores múltiples.
- **Integridad:** El código y los datos del servidor se conservan centralmente; esto implica menor coste de mantenimiento y protección de la integridad de los datos compartidos. Además, los clientes mantienen su individualidad e independencia.

La arquitectura cliente-servidor es una infraestructura versátil modular y basada en mensajes que pretende mejorar la portabilidad, la interoperabilidad y la escalabilidad de la computación; además invita a participar a una variedad de plataformas, hardware y software del sistema.

C.3.2 Componentes

Conceptualmente, los componentes de la arquitectura cliente-servidor son el cliente, el servidor y la infraestructura de comunicaciones (*middleware*), como se muestra en la figura C.1.

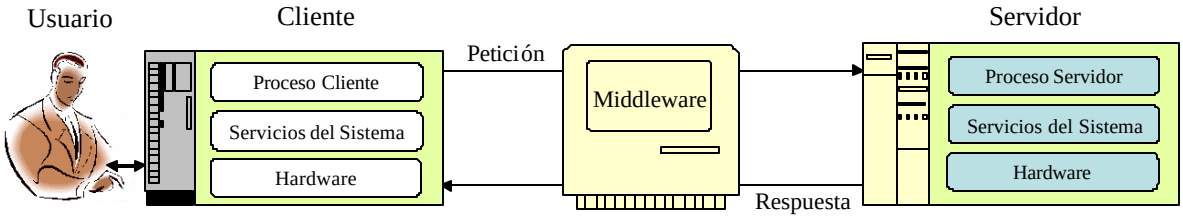


Fig. C.1: Arquitectura Cliente-Servidor

- **Cliente**

El cliente es la entidad por medio de la cual un usuario solicita un servicio, realiza una petición o demanda el uso de recursos. Este elemento se encarga, básicamente, de la presentación de los datos y/o información al usuario en un entorno gráfico. Como ejemplos de clientes pueden citarse interfaces de usuario para enviar comandos a un servidor, APIs para el desarrollo de aplicaciones distribuidas, herramientas en el cliente para tener acceso a servidores remotos (por ejemplo, servidores de un robot) o aplicaciones que solicitan acceso a servidores para algunos servicios.

- **Servidor**

El servidor es la entidad física que provee un servicio y devuelve resultados; ejecuta el procesamiento de datos, las aplicaciones y el manejo de información o recursos. En algunos casos existen procesos auxiliares que se encargan de recibir las solicitudes del cliente, verificar la protección, activar un proceso servidor para satisfacer la petición, recibir su respuesta y enviarla al cliente. Además, deben manejar los interbloqueos, la recuperación ante fallos, y otros aspectos afines. Por las razones anteriores, la plataforma computacional asociada con los servidores es más poderosa que la de los clientes. Por esta razón se utilizan PC's potentes, estaciones de trabajo,

minicomputadores o sistemas grandes. Además deben manejar servicios como administración de la red, mensajes, control y administración de la entrada al sistema ("login"), etc..

- **Middleware**

Para que los clientes y los servidores puedan comunicarse se requiere una infraestructura lógica que proporcione los mecanismos básicos de direccionamiento y transporte. A dicha infraestructura se le denomina Middleware, el cual es un término que abarca a todo el software distribuido necesario para el soporte de interacciones entre clientes y servidores.

El *middleware* es un módulo intermedio que no pertenece a los dominios del servidor, ni a la interfaz de usuario, ni a la lógica de la aplicación en los dominios del cliente; tampoco debe confundirse con la red física en sí (cableado, señales de radio o infrarrojas). Es una interfaz lógica estándar de los servicios de red. Sus funciones son:

- Independizar las dos entidades: El cliente y el servidor no necesitan saber comunicarse entre ellos, sino cómo comunicarse con el módulo de middleware.
- Traducir la información de una aplicación y pasarla a la otra: acepta consultas y datos recuperándolos de la aplicación cliente, los transmite y envía la respuesta de regreso. También genera los códigos de error.
- Controlar las comunicaciones: da a la red las características adecuadas de desempeño, confiabilidad, transparencia y administración.

Existen dos tipos de middleware:

1) El middleware general es el sustrato de la mayoría de las interacciones de cliente-servidor. En el capítulo 4 se discuten las distintas arquitecturas utilizadas como middleware en la computación distribuida.
2) El middleware de servicios específicos es necesario para cumplir un tipo particular de servicio de cliente-servidor. Así, existe un middleware específico para los servidores dedicados: Middleware para bases de datos, middleware para trabajo en groupo (*groupware*); etc..

El middleware es una herramienta adecuada, que no sólo es flexible y segura, sino que también protege la inversión en tecnología y permite manejar diferentes ambientes de computación.

C.3.3 Modelos de la Arquitectura Cliente-Servidor

Los componentes de una arquitectura cliente-servidor se pueden representar utilizando uno de los dos modelos siguientes:

- **Modelos de Dos Capas**

Se considera como modelo cliente-servidor de dos niveles o capas a la estructura más simple, cuyos componentes son:

Clientes: Por medio de la interfaz con el usuario vía una petición se solicita un servicio, el uso de un recurso, o bien el acceso a un conjunto de datos.

Servidores: Satisfacen la solicitud del usuario recibiendo la petición, direccionándola y enviando la respuesta al cliente, ya sea la consulta respectiva de datos, ejecutando el proceso requerido o permitiendo el acceso y/o uso del recurso.

En este modelo se acostumbra a instalar las bases de datos dentro del servidor, por las ventajas de almacenamiento y velocidad que ofrece en comparación con las del cliente.

- **Modelo de Tres Capas**

  El objetivo de este modelo es dividir las funciones de una aplicación en tres componentes:

  Presentación: Este componente se encarga de la interacción hombre máquina a través del monitor, teclado, ratón, o bien mediante algún otro medio como un dispositivo móvil o un reconocedor de voz.

  Control: Compuesto por varios servidores o componentes de software localizados en una o más plataformas que se encargará de conectar los sistemas existentes.

  Abstracción: En este componente se incluye la información en sí, los sistemas y aplicaciones existentes.

Cualquier aplicación de software tiene tres funciones fundamentales: administración de los datos (control), lógica de la aplicación (abstracción) y lógica de la presentación (presentación). En la mayoría de los casos, con el modelo de dos capas, el único servicio proporcionado por el servidor es el acceso a una base de datos. En esas situaciones, el cliente debe acceder a los datos, implementar la lógica de negocio, convertir los resultados en un formato apropiado, mostrar la interfaz destinada al usuario, y aceptar datos de entrada. Aunque este modelo es generalmente fácil de implementar al principio, es difícil mejorarlo, actualizarlo o hacerlo escalable. El modelo de tres capas se puede adaptar mejor a las aplicaciones distribuidas en Internet como los sistemas de interacción remota debido a la separación entre las funciones básicas que forman el sistema.

C.4 FORMULARIOS HTML Y CGI

Una de las tecnologías usadas para desarrollar entornos basados en Web es utilizar la interfaz CGI, que se basa en el paso de parámetros y formularios HTML, combinado con programas en C o Perl. Los formularios HTML permiten de alguna manera invertir el sentido del flujo de la información. Cuando en un formulario HTML se pulsa en el botón Enviar, los datos tecleados por el cliente se envían al servidor para su procesamiento. ¿Cómo recibe el servidor los datos de un formulario y qué hace con ellos? Éste es el problema que tradicionalmente han resuelto los programas CGI. Cada formulario lleva incluido un campo llamado *Action* con el que se asocia el nombre de programa en el servidor. El servidor arranca dicho programa y le pasa los datos que han llegado con el formulario. Existen dos formas principales de pasar los datos del formulario al programa CGI:

- Por medio de una variable de entorno del sistema operativo del servidor, de tipo String (método GET).
- Por medio de un flujo de caracteres que llega a través de la entrada estándar (stdin o System.in), que de ordinario está asociada al teclado (método POST).

En ambos casos, la información introducida por el usuario en el formulario llega en la forma de una única cadena de caracteres en la que el nombre de cada campo del formulario se asocia con el valor asignado por el usuario. Lo primero que tiene que hacer el programa CGI es decodificar esta información y separar los valores de los distintos campos. Después ya puede realizar su tarea específica. Normalmente, el programa CGI termina enviando al cliente (el navegador desde el que se envió el formulario) una página HTML en la que le informa de las tareas realizadas. La forma de enviar esta página HTML al cliente es a través de la salida estándar (stduot o System.out).

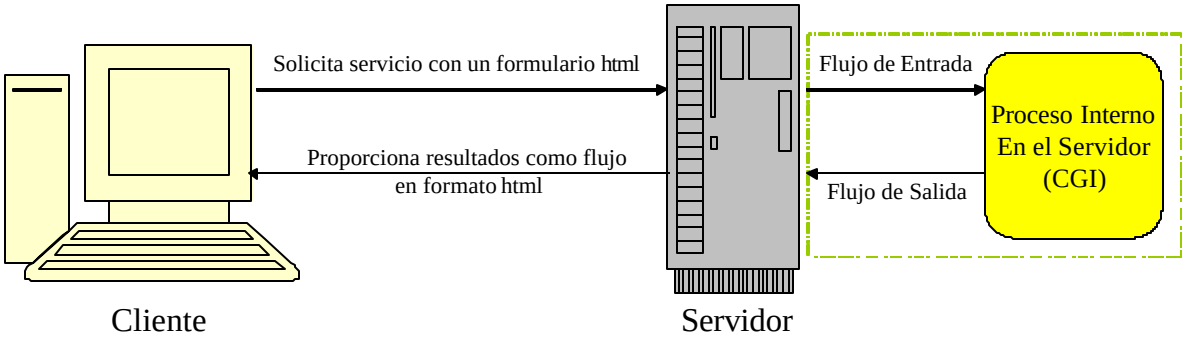


Fig. C.2: Esquema de Formulario HTML/CGI

En principio, los programas CGI pueden estar escritos en cualquier lenguaje de programación, aunque se utilizan principalmente los lenguajes Perl y C/ C++. Un claro ejemplo de un programa CGI sería el de un formulario en el que el usuario introdujera sus datos personales para registrarse en un sitio Web. El programa CGI recibiría los datos del usuario, introduciéndolos en la base de datos correspondiente y devolviendo al usuario una página HTML donde se le informaría de que sus datos habían sido registrados. La Figura C.2 muestra el esquema básico de los programas CGI.

Como se puede observar en los ejemplos planteados anteriormente en el capítulo 4, los interfaces basados en HTML no son suficientes para sistemas altamente interactivos porque demandan mucho procesamiento en el lado del servidor. Para solucionar el problema hay que incrementar el procesamiento en el cliente, y en este caso la tecnología Java proporciona un marco de trabajo múltiplataforma para conseguir interfaces de alto rendimiento.

C.5 JAVA

Creado por un equipo de Sun Microsystems, Java consiguió en los primeros meses de 1996 una atención bastante considerable, al ser presentado como un medio para ayudar a dominar Internet. Java es el primer lenguaje que tiene la virtud de ser compilado e interpretado de forma simultanea. El compilador de Java únicamente genera un código intermedio entre el lenguaje máquina que reconoce el procesador y Java, denominado *ByteCode*. Por lo tanto para ejecutar una aplicación Java es necesario disponer de un mecanismo que permita ejecutar el *ByteCode*, al que se le denomina Máquina Virtual de Java (Java Virtual Machine - JVM). Los *ByteCodes* se ejecutan en una implementación de JVM, que se puede entender como un ordenador dentro de un ordenador. Existe una JVM específica para cada plataforma [Java,03]. Es por tanto la máquina virtual Java la que proporciona la portabilidad de Java, convirtiéndolo en un sistema

multiplataforma (Figura C.3). Un mismo *ByteCode* se ejecutará de forma correcta en diferentes plataformas, que tendrán la JVM adecuada. Las características de Java respecto a cualquier otro lenguaje de programación, se han resumido en [Froufe,00].

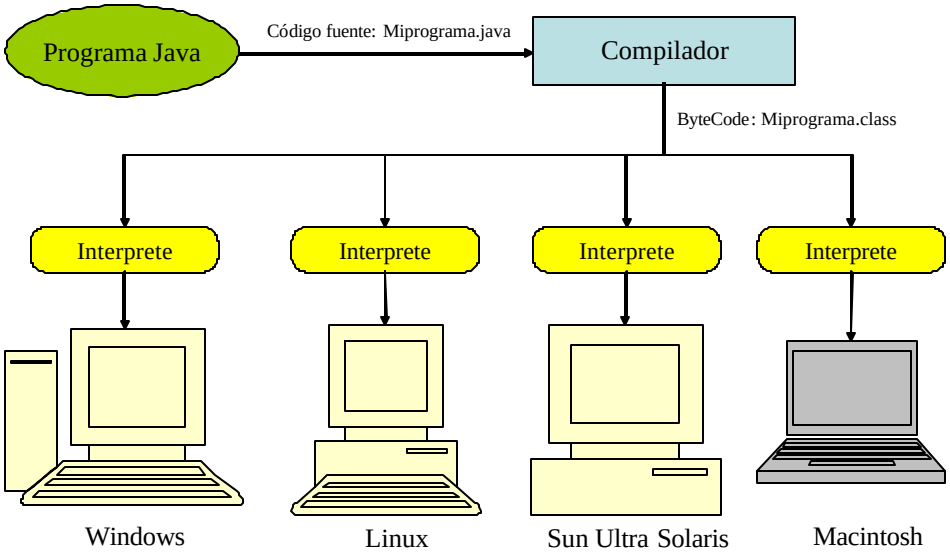


Fig. C.3: Multiplataforma de Java

La tecnología Java ha invadido aproximadamente cualquier aspecto de desarrollo Web. Los applets de Java se usan para hacer clientes dinámicos. Java Server Pages (JSP) y servlets se usan en el servidor. JavaBeans son pequeños componentes reutilizables que pueden agregarse para construir programas Java y applets más grandes, y han traído Java al mundo de componentes. La revolución de Java se ha extendido también en el mundo de los servidores de aplicaciones, con el Enterprise Java Beans EJB (objetos transaccionales) y Java Database Connectivity (JDBC).

En la arquitectura propuesta en la presente tesis se han usado applets para la interfaz gráfica de usuario y servlets para manejar las peticiones de los usuarios y establecer comunicación con los servidores del robot a través de CORBA. Los servlets presentan varias ventajas respecto a los CGI [Froufe,00]:

- Un servlet no se ejecuta en un proceso separado, lo que evita tener que lanzar una nueva instancia cada vez que se solicita su intervención.
- Una vez que se ha invocado, se queda en memoria, sin consumir más recursos del servidor, aunque se llame insistentemente. Un programa CGI necesita ser cargado y descargado en cada invocación.
- Solamente hay una instancia del servlet que responde a todas las peticiones concurrentemente, lo cual, además de suponer un considerado ahorro de memoria, puede manejar de forma muy efectiva datos persistentes.
- Un servlet puede ser ejecutado por un motor Servlet en una caja restringida (SandBox), de la misma forma que un applet es ejecutado en un navegador en una caja restringida; lo cual permite a los proveedores de servicios de Internet admitir que sus usuarios coloquen servlets en sus páginas, protegiendo al servidor ante el uso de servlets no fiables.

- Los servlets están escritos en Java, es decir, tienen un futuro asegurado. Por lo tanto tienen acceso a todos los paquetes Java.
- Los servlets son portables entre plataformas. Los Servlets están soportados directamente o mediante plug-in en la mayoría de los servidores Web.
- Los servlets, disfrutan de las características de seguridad inherentes a Java, como el manejo de memoria.
- Los servlets son mucho más eficientes que los CGI, en términos de memoria, cachés, tiempo de ejecución, …
- Barato. Hay un número de servidores Web gratuitos o muy baratos que son buenos para uso "personal" o en sitios Web sencillos. Sin embargo, con la excepción de Apache, que es gratuito, la mayoría de los servidores Web comerciales son relativamente caros. Una vez que tengamos un servidor Web, no importa el coste del servidor, añadirle soporte para Servlets (si no viene preconfigurado para soportarlos) es gratuito o muy barato.
- Potencia. Los Servlets Java nos permiten fácilmente hacer muchas cosas que son difíciles o imposibles con CGI normal. Por ejemplo, los servlets pueden hablar directamente con el servidor Web. Esto simplifica las operaciones que se necesitan para buscar imágenes y otros datos almacenados en situaciones estándar.

C.6 CORBA/JAVA RMI/DCOM/MOM

La Arquitectura CORBA (*Common Object Request Broker Architecture*) no es un producto, sino que es una especificación del grupo de Gestión de Objetos (Object Management Group – OMG) [OMG,03], un grupo de vendedores de la industria que publica especificaciones que los vendedores individuales pueden usar para hacer software compatible e interoperable. El middleware que CORBA define para la comunicación de objetos en máquinas diferentes se llama object request broker (ORB), los ORBs manejan el envío y la recepción de invocaciones de métodos a través de la red. Un ORB se instala sobre el cliente y el servidor. Las aplicaciones de cliente crean instancia de objetos remotos, usando los servicios de CORBA, e invocan método en ellos. El valor de retorno se envía a los clientes, como se muestra en la figura C.4.

CORBA define dos tipos de interfaces que pueden ser usados por los clientes que quieran invocar un método de un objeto remoto: stubs e invocación dinámica. Los stubs, como en el método de invocación remota RMI (*Remote Method Invocation*), representan un enlace fijo entre la interfaz y especifican la implementación del comportamiento del objeto. Las interfaces para stubs se definen en el Lenguaje de Definición de Interfaz (*Interface Definition Language-IDL*). La invocación dinámica, por otra parte, es una interfaz genérica de un objeto. En vez de estar definidas en un fichero IDL, las interfaces del objeto se agregan a una interfaz del repositorio de servicio, que representa los componentes de una interfaz como objetos, permitiendo el acceso a ellos en tiempo de ejecución [Conallen,99].

CORBA 2.0 añadió la interoperabilidad como un objetivo en la especificación. En particular, CORBA 2.0 define un protocolo de red llamado IIOP (*Internet Inter-ORB Protocol*), que permite a los clientes usar productos CORBA de cualquier desarrollador que se comuniquen con objetos CORBA de cualquier otro desarrollador. IIOP trabaja sobre Internet, o más exactamente, sobre cualquier implementación de TCP/IP.

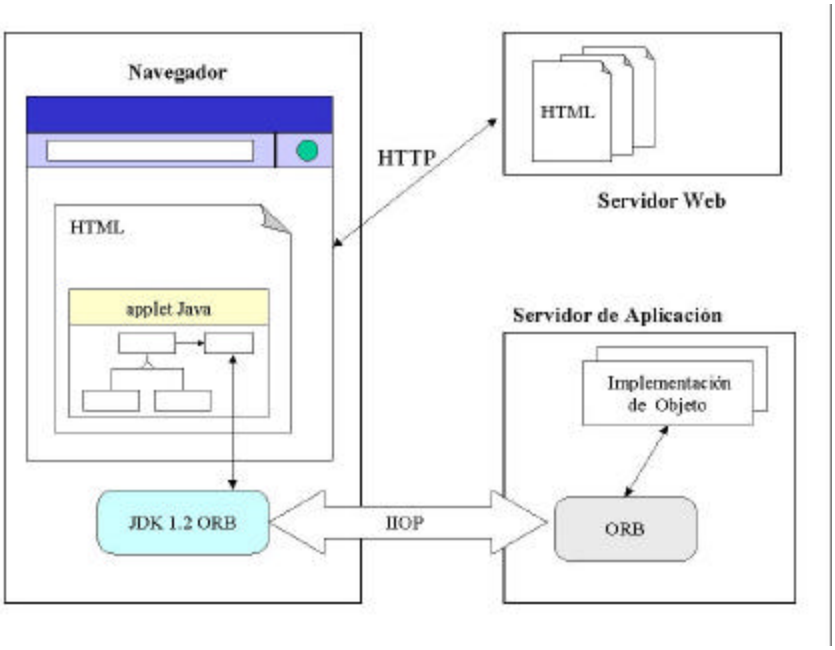


Fig. C.4: CORBA en una aplicación Web

La Invocación Remota de Métodos RMI, es la norma de Java para objetos distribuidos. RMI permite clases Java que se comuniquen con otras clases Java ubicadas en máquinas diferentes. Toda la conversión (*marshling*) de protocolos se maneja en las clases del stub, en el esqueleto (*skeleton*) y en la infraestructura de la RMI. La figura C.5 muestra como trabajan juntos los applets y los objetos remotos.

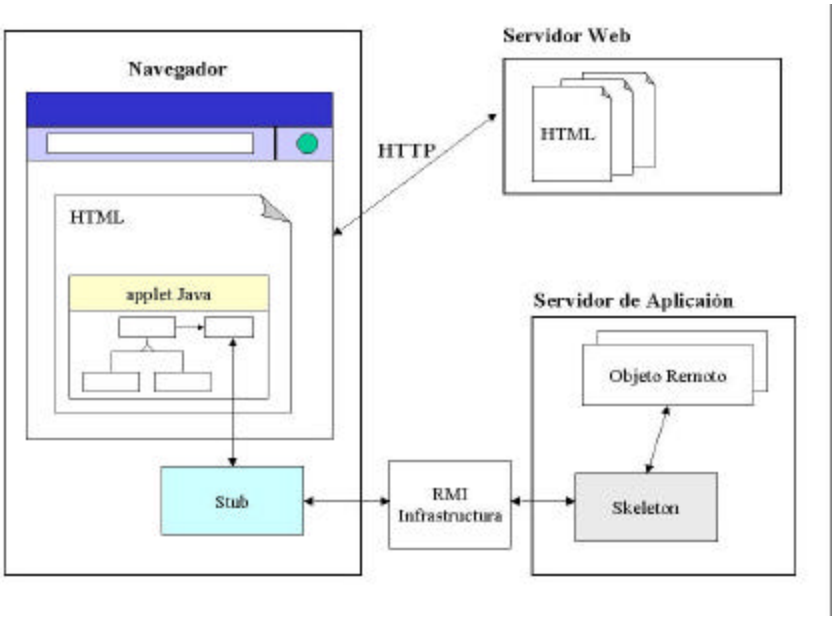


Fig. C.5: Applets usan RMI

En particular, IIOP se usa como protocolo de transporte en una versión de Java RMI (llamada "RMI over IIOP"). Dado que EJB está definido en términos de RMI, también puede usar IIOP. Varios servidores de aplicaciones disponibles en el mercado usan IIOP, pero no implementan el

API completo de CORBA. Dado que todos ellos usan IIOP, los programas pueden interoperar con cualquier otro que esté escrito para el API de CORBA.

La solución de Microsoft para la gestión de objetos distribuidos está en extender el modelo popular de Objeto (*Component Object Model-COM*) con COM Distribuido (DCOM). Microsoft describe DCOM como simplemente COM con un cable más largo. La figura C.6 muestra el uso de DCOM en una aplicación Web.

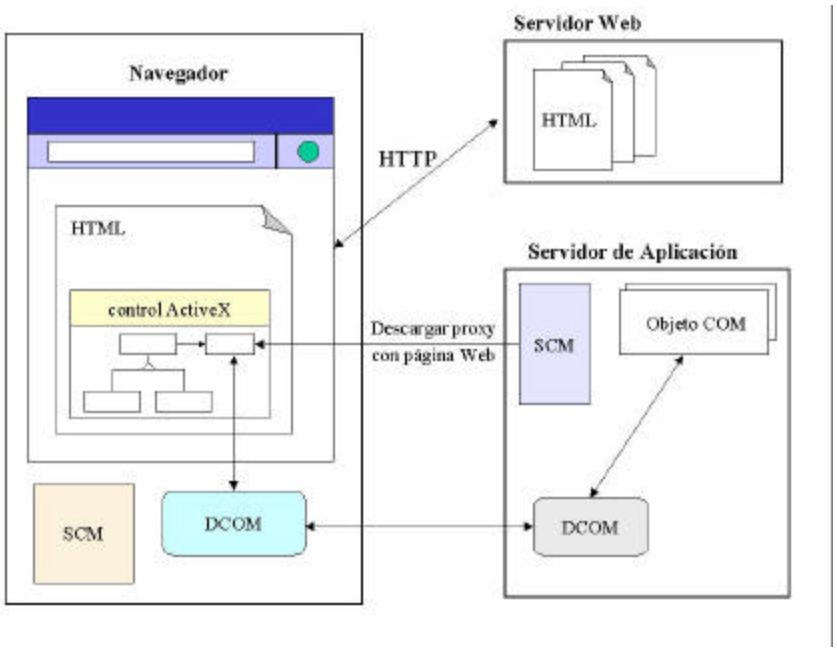


Fig. C.6: Uso de DCOM en una aplicación Web

Así como RMI hace, DCOM aísla al desarrollador de objetos de los detalles de su distribución. RMI es diferente, en que los objetos del servidor deben implementar interfaces remotas concretas. DCOM da al desarrollador del objeto completa independencia de la infraestructura de objeto distribuido. La mayor desventaja de usar DCOM en vez de RMI o CORBA es el requerimiento del cliente de funcionar bajo Windows. Aunque COM y DCOM son especificaciones públicas, la realidad es que los sistemas operativos basados en Windows los apoyan. Por otro lado, Microsoft's Internet Explorer es el único navegador que tiene apoyo nativo para COM [Conallen,99].

MOM (*Message-Oriented Middleware*) proporciona otra solución para la computación distribuida [MOM,03]. Al contrario que CORBA y RMI, MOM no está basado en un estándar de la industria. Los datos se intercambian mediante paso de mensajes o colas de mensajes, soportando ambos mecanismos interacciones síncronas o asíncronas entre los procesos distribuidos. MOM asegura el correcto reparto de mensajes mediante colas proporcionando todo el soporte necesario para ello (directorio, seguridad y servicios administrativos). El sistema de colas es útil para procesos donde cada paso es independiente del anterior. MOM, al estar basado en colas de mensajes, está orientado para comunicaciones asíncronas, mientras que RMI y CORBA están diseñados para soportar comunicaciones síncronas. MOM también permite la función de broadcasting (envío de información a múltiples receptores). Tanto CORBA como RMI son estándares lo que les proporciona un mayor potencial de interoperabilidad. Sin embargo,

hoy en día todavía existen problemas para soportar CORBA y RMI en los navegadores debido a intereses de terceras partes. MOM no tiene este problema ya que permite incluir la API necesaria en el applet. MOM está orientado para aplicaciones menos complejas y que no necesiten comunicaciones síncronas. En aplicaciones de mayor envergadura son preferibles RMI y CORBA.

En el sistema propuesto en la presente tesis, se ha utilizando CORBA como arquitectura de objetos distribuidos. La decisión de usar CORBA como arquitectura de objetos distribuidos de un laboratorio remoto esta basado en dos comparativas. La primera comparativa pretende dar una información global de los aspectos más interesantes a tener en cuenta para elegir la arquitectura (4 es la máxima puntuación y 0 es la mínima). La tabla C.1 muestra la comparación entre CORBA, DCOM y RMI [Gopalan,00].

Tabla C.1: Comparativa CORBA, DCOM, RMI.

Características	CORBA	DCOM	RMI
Nivel de abstracción	4	4	4
Integración con Java	4	4	4
Soporte de SO	4	2	4
Todas las implementaciones de Java	4	1	4
Facilidad de configuración	3	3	3
Invocación distribuida de métodos	4	3	3
Invocaciones guardan el estado	4	3	3
Inspección dinámica	4	3	2
Invocación dinámica	4	4	1
Eficiencia	3 (3.5mseg.)	3 (3.8mseg.)	3 (3.3-5.5mseg.)
Seguridad a nivel de cable	4	4	3
Transacciones a nivel de cable	4	3	0
Referencias persistentes a objetos	4	1	0
Servicios de nombres basados en URLs	4	2	2
Invocaciones multilenguaje	4	4	0
Escalabilidad /interoperabilidad	4	2	1
Estándar abierto	4	2	2

Las tres soluciones son aconsejables pero se aprecia que CORBA sólo tiene un posible competidor: DCOM. Este modelo de objetos de Microsoft, es relativamente robusto, su defecto es que está disponible casi exclusivamente en el entorno Windows. RMI, cuya principal ventaja

es que soporta pasar objetos por valor, es una tecnología muy sencilla y de fácil programación pero su dependencia de Java y otras carencias la ponen en desventaja frente a las otras soluciones.

La otra comparativa es una comparación cualitativa y cuantitativa entre CORBA y RMI [Juric,00]. Este estudio concluyo que CORBA es adecuado para aplicaciones de gran escala o parcialmente abiertas a la Web donde el soporte de software de otros fabricantes (*legacy software*) es necesario y por su buen rendimiento ante la esperada gran carga de clientes. Además, los servidores CORBA pueden ser situados en cualquier sitio de Internet. RMI, por otro lado, es adecuado para aplicaciones totalmente abiertas a Internet de pequeña escala donde dicho soporte puede ser gestionado por puentes de propia construcción o preconstruidos, donde la facilidad de aprendizaje y la facilidad de uso es más crítica que el rendimiento.

Para el sistema desarrollado en esta tesis, la característica decisiva ha sido poder comunicar dos objetos implementados en distintos lenguaje de programación (Java y C++). Con RMI se puede sólo comunicar con objetos distribuidos de Java aunque con la integración de RMI con IIOP utilizando el protocolo RMI-over-ORB se puede proporcionar facilidad de uso con la ventaja de interoperabilidad de CORBA [Schaaf,01].

C.7 JAVA IDL

El lenguaje de definición de interfaces (IDL) es un lenguaje de programación pensado exclusivamente para especificar las interfaces de las clases cuyas instancias se quieren hacer públicas a objetos remotos que las usaran como clientes [OMG,03]. La necesidad de un IDL viene dada por la independencia de CORBA respecto a la arquitectura y al lenguaje de programación. Distintos lenguajes soportan diferentes tipos de datos y tienen distintas formas de especificar clases. IDL pone de acuerdo a distintos lenguajes en el formato y tamaño de sus especificaciones.

Para el desarrollo de objetos distribuidos con Java IDL, los pasos siguientes son necesarios:

- **Definir la Interfaz Remota**

Los objetos distribuidos están identificados por referencias a objetos, las cuales se describen mediante las interfaces IDL. Se define la interfaz para el objeto remoto usando el IDL en lugar de Java porque el compilador `idlj` (u otros como idltojava o idl2java) automáticamente genera los ficheros de *`stub`* y *`skeleton`* en Java a partir de la definición IDL, así como toda la infraestructura para conectar con el ORB. También, usando IDL, se hace posible a los desarrolladores implementar clientes y servidores en cualquier otro lenguaje compatible con CORBA.

- **Compilar la Interfaz Remoto**

Cuando se ejecuta el compilador IDL a Java (`idlj`) con la definición de la interfaz, se genera una versión en Java de la interfaz, así como el código de las clases de los ficheros de `stub` y `skeleton` que permiten enganchar la aplicación al ORB. En la tabla C.2 quedan resumidos los mapeos que se hacen entre constructores de IDL y Java.

Tabla C.2: Mapeo IDL -> Java

IDL	Java
Module	Package
Interface	Interface
struct	Helper class
typedef	Helper class
const	static final
enum	final class

Y en la tabla C.3, el mapeo para los tipos de datos entre IDL y Java.

Tabla C.3: Tipos de Datos IDL y Java

IDL	Java
Boolean	Boolean
char / wchar	Char
Octet	Byte
short / unsigned short	Short
long / unsigned long	Int
long long / unsigned long long	Long
Float	Float
Double	Double
string / wstring	String

- **Implementar el Servidor**

Una vez ejecutado el compilador `idlj`, se puede usar el esqueleto generado para implementar la aplicación servidora. Además de implementar los métodos de la interfaz remota, el código del servidor incluye un mecanismo para iniciar el ORB y esperar una petición de un cliente remoto. En el sistema propuesto en esta tesis, se implementa el servidor en C++ en vez de java y se usa el esqueleto generado por un compilador IDL a C++ como proxy del servidor.

- **Implementar el Cliente**

De forma similar, se usarán los `stubs` generados por el compilador `idlj` como base para la aplicación cliente. El código del cliente inicia su ORB, busca el servidor usando el servicio de

nombrado proporcionado con Java IDL, obtiene una referencia al objeto remoto y llama al método.

- **Iniciar las aplicaciones**

Una vez que estén implementados el cliente y el servidor, se puede iniciar el servicio de nombres, iniciar el servidor y por último arrancar el cliente.

C.8 SERVIDORES WEB

Los servidores son programas que se encuentran permanentemente esperando a que algún otro ordenador realice una solicitud de conexión. En un mismo ordenador es posible tener simultáneamente servidores de distintos servicios (HTTP, FTP, TELNET, etc.). Cuando a dicho ordenador llega un requerimiento de servicio enviado por otro ordenador de la red, se interpreta el tipo de llamada, y se pasa el control de la conexión al servidor correspondiente a dicho requerimiento. En caso de no tener el servidor adecuado para responder a la comunicación, está será rechazada. Un ejemplo de rechazo ocurre cuando se quiere conectar a través de TELNET (típico de los sistemas UNIX) con un ordenador que utilice Windows 95/98.

La comunicación a través del protocolo HTTP es diferente, ya que es necesario establecer una comunicación o conexión distinta para cada elemento que se desea leer. Esto significa que en un documento HTML con 10 imágenes son necesarias 11 conexiones distintas con el servidor HTTP, esto es, una para el texto del documento HTML con las etiquetas y las otras 10 para traer las imágenes referenciadas en el documento HTML.

La tabla C.4 lista algunos de los servidores Web disponible. El servidor Apache es un servidor Web potente y estable. Este servidor ha mostrado ser substancialmente más rápido que muchos otros servidores. Aunque ciertos servidores han dicho que sobrepasan la velocidad de Apache, se puede pensar que es mejor tener un servidor rápido y gratis que uno extremadamente rápido que tiene alto coste económico.

Tabla C.4: Servidores Web

Servidor	Descripción	Plataforma	Tipo
Apache	Servidor Web de propósito general con muchas prestaciones.	Win/Linux/Mac	Gratis
NES	Netscape Enterprise Server. Tiene alto rendimiento y es muy fiable.	Win NT UNIX	No es gratis
MIIS	Microsoft Internet Information Services. Tiene alto rendimiento y opciones robustas.	Windows	No es gratis
Fnord	Un servidor básico de Web.	Win 95/NT	Gratis
HTTPS	Un servidor muy básico	NT	Gratis
NetPresenz	Un servidor Web, ftp y Gopher. Sporta la identificación de usuarios.	Macintosh	SW

Se ha decidido utilizar Apache según una comparativa entre los tres servidores Web más populares (*Netscape Communications Enterprise Server* N.E.S 4.0, *Microsoft Internet Information Server* I.I.S 4.0, y *Apache Server* 1.3.9) [Abualsamid,99][Yerxa,99]. Esta comparativa se ha realizado desde el punto de vista de las prestaciones del servidor, soporte para diversas plataformas, posibilidades de gestión y fiabilidad. En la figura C.7, se resume esta comparativa.

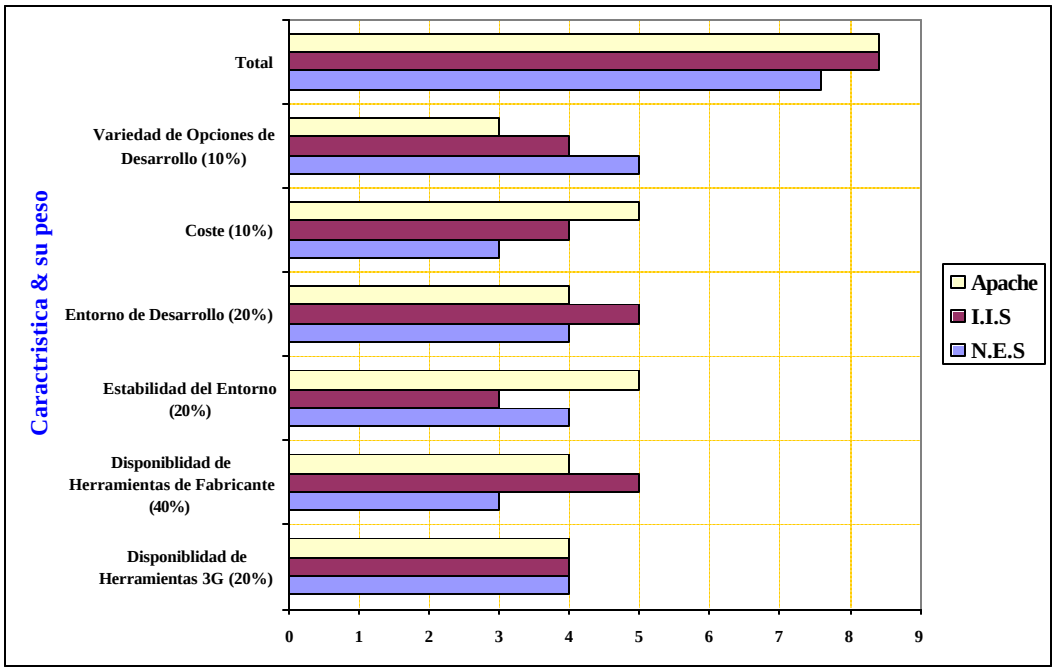


Fig. C.7: Comparativa entre los Servidores Web más usados

Como primer criterio de comparación se tiene la gestión y ajuste de cada servidor Web. En este aspecto, casi siempre han sido Windows NT e IIS los que causan mayores problemas frente a los productos de Apache y Netscape, basados en plataformas Solaris o Linux. C.E.S de Netscape también necesita muchas menos reinicializaciones trabajando sobre la plataforma de Intel.

La capacidad del servidor Web para añadir funcionalidad y control a los contenidos de la sede Web se toma como segundo criterio. En este aspecto sobresale la consola de gestión de IIS, que ha demostrado ser la opción más sencilla para configurar y controlar el servidor Web y para aplicar ajustes de este tipo.

También se tiene en cuenta el entorno nativo de desarrollo de cada servidor Web. El claro favorito para muchos es Apache y su utilidad de creación de scripts PHP, por su fiabilidad y sus prestaciones básicas.

Aunque cada uno de los competidores tiene sus ventajas e inconvenientes, no se pueden ignorar las amplias capacidades y soporte de plataformas de Netscape Enterprise Server, ni las robustas opciones de desarrollo Web y elevado rendimiento de IIS.

El servidor Apache, aunque no presenta los mejores resultados en cuanto a rendimiento ni dispone de las mejores opciones de desarrollo, soporta casi cualquier plataforma, tiene una extensa documentación y un amplio seguimiento en la comunidad de Internet lo que lo convierte

en una gran apuesta. Además, no se olvida, que Apache Server es distribuido gratuitamente a través de Internet y se ha ganado la reputación de ser el servidor Web más fiable de los disponibles.

C.9 STREAMING VIDEO

El *streaming* es una tecnología que nos permite la emisión (recepción) de contenidos audiovisuales utilizando las redes de datos. Mediante esta tecnología es posible recibir los vídeos, sin necesidad de descargar el fichero completamente. Esto es posible gracias a un sistema de almacenamiento intermedio entre el servidor y el reproductor de vídeo. Estos reproductores suelen incluir conectores que permiten su visualización desde navegadores como Netscape o Internet Explorer. Un sistema de streaming viene definido por una codificación y un sistema de transporte. La codificación puede ser MPEG-1, MPEG-2, Real, MPEG-4, QT,... y se utiliza UDP (unicast o multicast) como protocolo de transporte. Los contenidos, pueden estar almacenados en un servidor (vídeo bajo demanda) o bien crearse en el mismo momento de su difusión (emisiones en directo). A continuación la arquitectura básica de un sistema de streaming se muestra en la figura C.8.

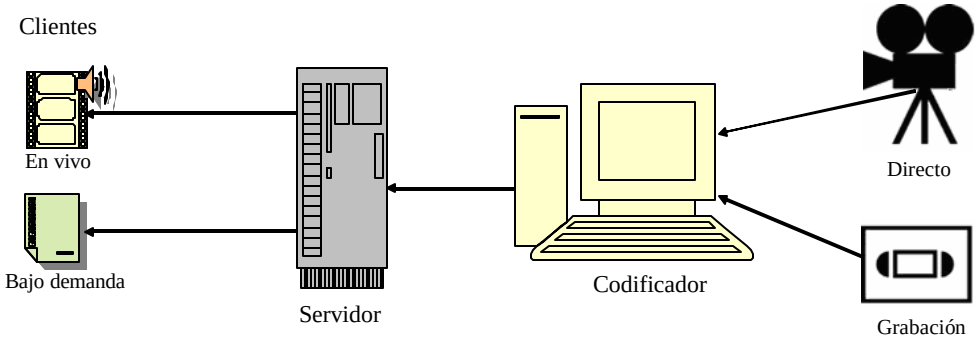


Fig. C.8: Arquitectura Básica del Streaming de Vídeo

Hoy en día existen muchos productos comerciales de streaming de video que representan una nueva tecnología de altas prestaciones que revolucionan los sistemas CCTV y de video vigilancia convencional permitiendo la transmisión de video a través de cualquier red de datos, incluyendo redes locales de amplia cobertura e incluso a través de Internet. Entre los cuales cabe citar los siguientes:

- **Los Servidores de AXIS**

Los servidores de video AXIS (2400 y 2401) ofrecen una solución completa en una sola caja para la emisión de vídeo. Conectándose directamente por un lado a una cámara y por otro a una red Ethernet de 10/100 Mbps (véase la figura C.9), este tipo de servidores permiten visualizar vídeo en tiempo real en un computador remoto sin las complicaciones que en algunas ocasiones conlleva la instalación de servidores de vídeo software. Estos servidores envían imágenes animadas JPEG con un rango de hasta 30 imágenes por segundo NTSC y 25 PAL. Los servidores de video AXIS utilizan los protocolos de TCP/IP y se usan con la mayoría de los sistemas operativos: Windows 95,98 y NT así como Linux, UNIX, Mac y muchos otros. El único software necesario es Internet Explorer 4.x con los controles ActiveX de AXIS o Netscape Navigador 4.x [AXIS,03].

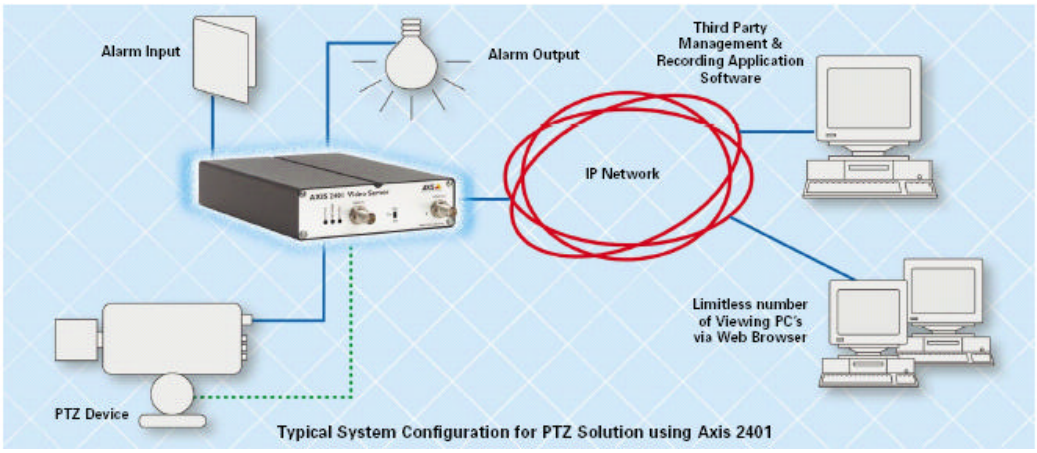


Fig. C.9: Servidor de video AXIS 2401

- **Sony SNC-RZ30**

En el sistema propuesto del laboratorio remoto se ha elegido una solución presentada por Sony para proporcionar supervisión remota a los usuarios. La camera Sony SNC-RZ30P es la primera cámara IP capaz de grabar vídeo en movimiento (hasta 25 imágenes por segundo) con imágenes en color de alta calidad y resolución VGA (640x480) [Sony,03]. Se trata de una cámara compacta de sólo 1,2 Kg que incorpora funciones de ampliación, inclinación y giro (PTZ) por control remoto así como un servidor IP con el que puede conectarse directamente a cualquier red de área local (LAN). De esta manera, la cámara es capaz de proporcionar imágenes de gran calidad a cualquier PC conectado a la red a través de un navegador Web estándar o mediante el software de aplicaciones adecuado. Además, la SNC-RZ30 se puede utilizar tanto para funciones de supervisión remota como de vigilancia. La figura C.10 muestra como se integra la cámara en un sistema.

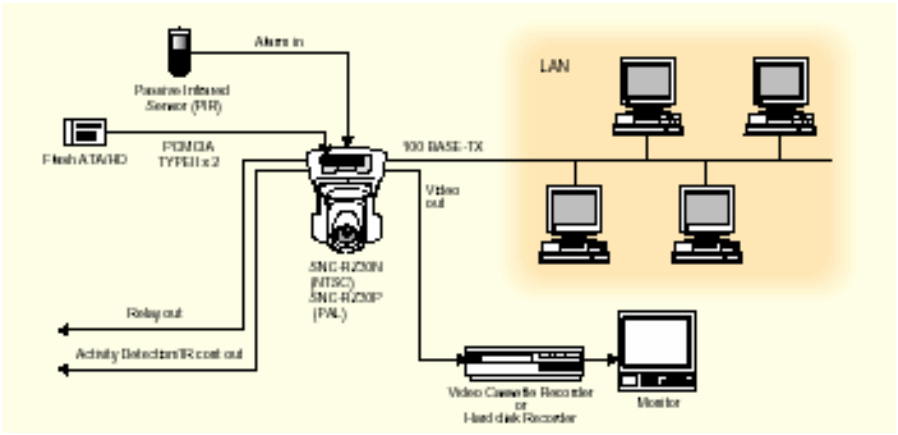


Fig. C.10: Sony SNC-RZ30 y el diagrama de la conexión

El mecanismo de control directo de inclinación y giro es extremadamente silencioso cuando está en funcionamiento, lo cual es fundamental para las aplicaciones donde es crucial no alterar el entorno con molestos ruidos. El equipo cuenta con un objetivo de enfoque automático y zoom óptico de 25x y, además, capta hasta el más mínimo detalle durante el día y la noche gracias a la función día/noche.

- **Otras Soluciones**

En la tabla C.5 se resumen otras soluciones comerciales que se pueden usar para proporcionar streaming video.

Tabla C.5: Servidores de Video Comerciales

Fabricante	Modelo	Descripción
Panasonic	KX-HCM10 Network Camera	Este modelo tiene un servidor Web, software de control (TCP/UDP) y software de correo electrónico (SMTP) todos integrados en esta cámara pequeña de peso ligero. Solamente se enchufa la cámara en una red Ethernet, se puede controlar la cámara desde un ordenador personal usando un navegador Web. Esta cámara ofrece resolución de vídeo hasta 640x480 así como el control remoto del Pan/Tilt.  También puede enviar correo electrónico automáticamente cuando se lanza por el sensor de tiempo o un sensor de seguridad opcional. Con el KX-HCM10, se puede ver la cámara por hasta 10 usuarios al mismo tiempo. Además, se pueden tener hasta 4 cámaras simultáneamente.
IQInvision	IQEye3	Este modelo es una cámara de red inteligente capaz de transmitir más de 1.8 Mpixels/second JPEG. Las bajas exigencias de potencia y la flexibilidad de conectar a la red hacen el IQEYE3 perfecto para el despliegue en una amplia variedad de entornos. IQPOET opcional (Power over Ethernet) se puede añadir para hacer el IQEYE3 fácil de instalar y mantener. 
iPIX	180 NETCAM	Con la cámara de red iPIX 180 º se puede capturar una escena completa del entorno de trabajo entregando vistas digitalmente navegables de 180 º con la adición simple de una lente y el visor de Java en el servidor. Las cámaras realzadas con el iPIX 180 º ofrecen la capacidad para visores ilimitados simultáneos para controlar su propia vista y eliminar la necesidad de un motor de Pan/Tilt mecánico 
JVC	VN-C30U	La cámara VN-C30U trae un estándar nuevo de calidad y versatilidad a la transmisión de imágenes basadas en red. Además de la calidad dramáticamente mejorada de fotos gracias a su poderosa cámara CCD, el VN-C30U destaca por un servidor Web integrado que permite conectarla directamente a Internet sin un ordenador adicional.  Esto hace mucho más fácil establecer sistemas de vigilancia remotos. La compresión MPEG1 asegura la entrega de imagen lisa, en tiempo real, mientras una interfaz integrada 10Base-T/100Base-TX da conexión de red de alta velocidad para transmitir datos confiables y para la operación.

C.10 HERRAMIENTAS COMERCIALES PARA MÓDULOS EDUCATIVOS

Como se muestra en la tabla C.6, existen varias herramientas comerciales que se pueden utilizar para desarrollar los módulos educativos de los sistemas de educación basados en la Web.

Tabla C.6: Herramientas Comerciales

Paquete		Descripción	Plataforma	Tipo
Correo-a-Página Web	MhonArc	MhonArc es una utilidad, escrita en Perl, que traduce un fichero Unix de correo a una página Web. Cualquier vinculo en un mensaje de correo se convierten en hipervínculos y cualquier fichero adjunto al mensaje de correo se convierte a formato Web o se provee como un hipervínculo.	Unix	Gratis
Creadores de Examines	Web Worksheet	Una construcción completa de preguntas y un entorno de gestión de exámenes.	Perl	Gratis
	TestCreator	Un programa de construcción de exámenes. Se usa con otro programa 'answer' para ofrecer exámenes basados en servidor. También capaz de producir exámenes en JavaScript.	Unix	Gratis
	Test2000	Crea exámenes tipo test JavaScript, que pueden almacenarse y editarse.	Windows	Gratis
	Jquiz	Crea páginas de examen JavaScript. Los exámenes se contestan escribiendo palabras o frases cortas. Se permiten tres respuestas posibles.	Win 95/ NT	Gratis
Ayudantes de Corrección	Markin	Un programa que permite insertar un texto de respuesta en una ventana, con enlace a comentarios predefinidos.	Windows	SW
Constructores de Clases	TopClass	Una construcción de clases en línea vía Web y un paquete de gestión.	Win 95, Unix , Macintosh	Demo
	WebCT	Construcción de una clase basada en Web y un paquete de gestión.	Perl	Beta
	WebFuse	Sistema de publicación Web con características de gestión.	Perl	Beta
Clasificación y Búsqueda	WWWWais	Interfaz Web para buscar índices creados por Swish	Unix	Gratis
	Swish	Un programa, que indexa las páginas Web	Unix	Gratis
	ICE	Dos programas, uno que indexa las páginas Web, y otro que actúa como una interfaz Web para la búsqueda con este índice.	Perl	Gratis
Logfile	Follow	Una utilidad, que permite seguir la sucesión de página Web.	Perl	Gratis
	Analog	Utilidad de propósito general para analizar la utilización de un servidor Web.	Unix, Windows, Macintosh	Gratis

Apéndice

D

ROBOT B21

Apéndice D ROBOT B21

D.1 ROBOT B21

Se ha empleado en el desarrollo de esta tesis el vehículo B21 de *Real World Interface* [iRobot,03], diseñado para trabajos de investigación y desarrollo en medios interiores. El robot (véase la figura D.1) consta de tres secciones principales que son la base, el cuerpo y la consola. El cuerpo y la consola están físicamente unidos, mientras que la base es fácilmente separable del resto.

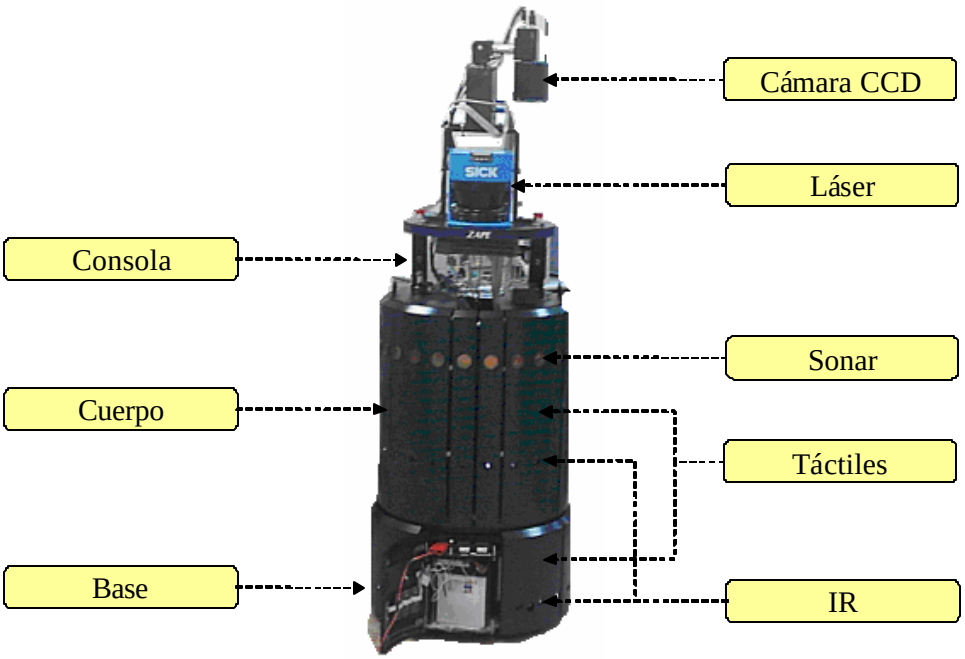


Fig. D.1: Robot B21

El robot B21 se mueve mediante 4 ruedas que son guiadas de forma sincronizada. La base no rota con las ruedas, al ser éstas dirigidas, sino que permanece siempre con la misma orientación. Es el cuerpo el que se orienta según el ángulo de rotación de las ruedas. La base del robot posee dos motores y aloja 32 sensores de infrarrojos y 32 sensores táctiles situados en las ocho puertas que forman la cubierta y protegen los componentes internos.

En el cuerpo del robot se encuentran 24 sensores de ultrasonidos, 24 infrarrojos y 24 táctiles. En el interior del robot se encuentra la CPU principal que está equipada con Linux y se comunica con los sensores del cuerpo mediante un bus de alta velocidad y con la base vía protocolo serie RS-232 a 9600 baudios. Además de esta CPU principal se dispone de otra CPU para el procesamiento y adquisición de imágenes. La comunicación entre las dos CPU se realiza a través de Ethernet. El robot B21 también tiene un sensor láser PLS de la casa SICK. Este telémetro láser realiza un escaneado entre 0^o a 180^o del entrono proporcionado la magnitud de las distancias a los objetos que se encuentran en el plano de medida. La conexión con el PC del robot se realiza vía protocolos RS-232 a 9600 baudios. El sistema de vision del robot B21 está formando por una plataforma con movimientos *pan* y *tilt* de TRC modelo Zebra, que permite el movimiento de una cámara CDD.

D.2 MOBILITY

Mobility es un paquete software, integrado y orientado a objetos, para control de robots desarrollado por Real World Interface [iRobot,03]. Este framwork permite diseñar software de control de sistemas mono y/o multirobot [Mobility,99]. En la figura D.2, se puede la arquitectura de Mobility.

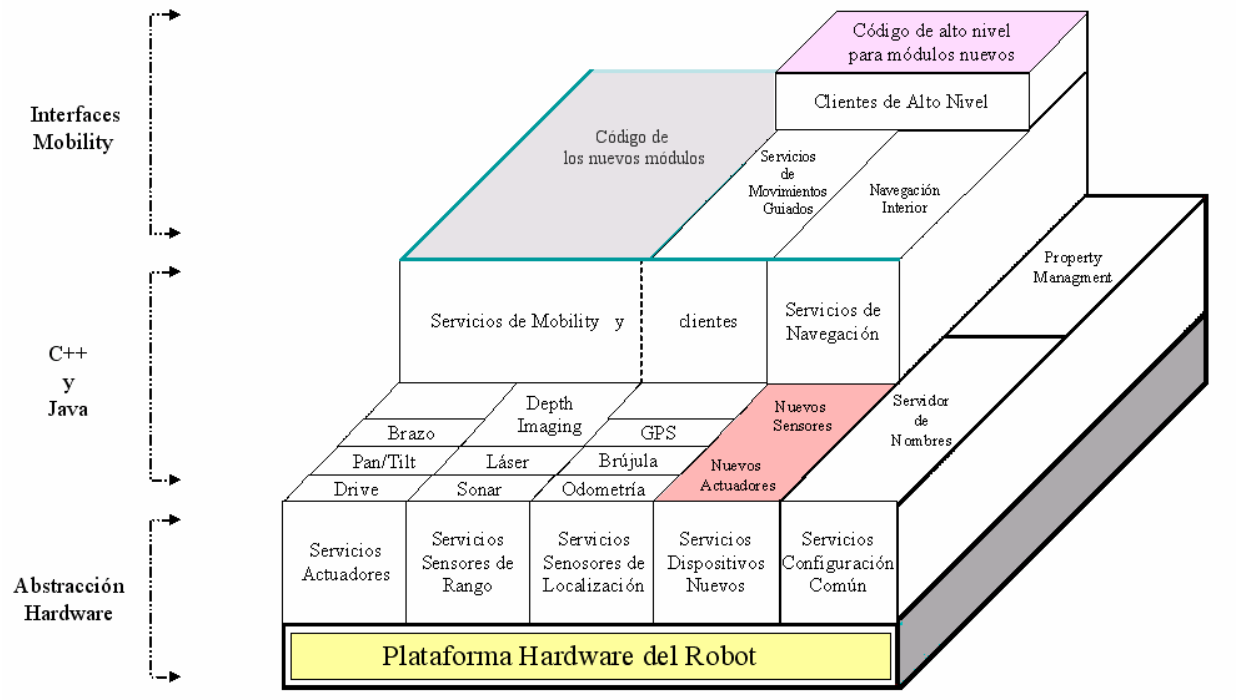


Fig. D.2: Arquitectura de Mobility

Este *framework* está compuesto de las siguientes partes:

- Un conjunto de herramientas software entre las que destaca la interfaz gráfica MOM (*Mobility Object Manager*),
- Un modelo basado en objetos del robot,
- Un conjunto de módulos de control sencillos,
- Una librería de clases en C++ orientada a objetos para simplificar el desarrollo del código.

Como se puede ver en la figura D.2, Mobility define un tipo de arquitectura flexible y distribuida basada en objetos organizados de forma jerárquica, interconectados unos con otros mediante el estándar CORBA 2.0 /IIOP, establecido por OMG [OMG,03], el cual utiliza IDL para describir los interfaces entre objetos. Cada uno de estos objetos tiene su propia identidad, interfaz y estado. Estos objetos son representaciones abstractas de sensores, actuadores, comportamientos, datos, etc. Esta relación de unos objetos con otros mediante IDL permite que se puedan añadir futuros módulos a la arquitectura inicial de Mobility.

La gran ventaja de esta arquitectura es que cada módulo sea una parte independiente (autónoma) y que variaciones en algún módulo, la creación de nuevos módulos o la eliminación de módulos ya existentes no afecte al resto de la arquitectura, teniendo la arquitectura un comportamiento tipo “mecano” con todos los módulos que la componen.

La librería de clases en C++ que incorpora Mobility facilita mucho el proceso de desarrollo (derivando clases de la librería) de nuevos módulos permitiendo una de las grandes ventajas de los lenguajes orientados a objetos, la reutilización de código ya existente, lo que conlleva una disminución en el tiempo de desarrollo muy importante.

Apéndice

E

LISTA DE ACRÓNIMOS

Apéndice E LISTA DE ACRÓNIMOS

A

ap 3.0	Protocolo de autentificación.
API	*Application Programming Interfaces.* Interfaces de programación de aplicaciones.
APPLET	Aplicación muy específica, que puede ser importada de Internet, escrito en lenguaje Java.
ARITI	*Augmented Reality Interface for Telerobotic Applications via Internet*.
ASCII	*American Standard Code for Information Interchange.* Código estándar americano para el intercambio de información. Asigna a una serie de caracteres alfanuméricos un código equivalente en hexadecimal.
AWT	*Abstract Windowing Toolkit Package*. Paquete de herramientas abstractas para trabajar con ventanas de Java. Contiene todas las clases e interfaces necesarias para crear y manipular interfaces gráficas de usuario.

B

C

CGI	*Common Gateway Interface*. Interfaz de Compuerta Común. Tecnología usada para crear documentos dinámicos del WWW.
CLDC	*Connected Limited Device Configuration*. Estándar para dispositivos J2ME.
CORBA	*Common Object Request Broker Architecture.* Arquitectura de intercambio de solicitudes de objetos comunes. CORBA define la infraestructura para la arquitectura OMA (*Object Management Arquitecture*) de OMG, especificando los estándares necesarios para la invocación de métodos sobre objetos en entornos heterogéneos.
CPU	*Central Processing Unit*. Unidad Central de Proceso.

D

DCOM	*Distributed Component Object Model*. Es el modelo de objetos distribuidos de Microsoft.
DNS	*Domain Naming Service*. Servicio de Dominio de Nombres. Servicio automático que resuelve nombres de identificación de ordenadores a direcciones IP.

E

EJB	*Enterprise Java Beans*. Los Enterprise Java Beans(EJB) son las componentes del modelo de desarrollo de aplicaciones enterprise de Sun.
ESA	*European Space Agency*. Agencia Espacial Europea.

F

FAQ	*Frequently Asked Questions*. Preguntas más frecuentes.
Framework	Los *frameworks* son los generadores de aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.
FTP	*File Transfer Protocol*. Protocolo de Transferencia de Archivos. Permite obtener ficheros de un ordenador remoto (usualmente a través de Internet).

G

GDL	Grados de Libertad.
GIF	*Graphical Interchange Format*. Formato de intercambio de gráficos. Método de gran popularidad por CompuServe para compresión de gráficos.
GPRS	General Packet Radio System. Servicio de Radio de Paquetes Generales.
GPS	*Global Positioning System*. Sistema de posicionamiento global por satélites.
GSM	Global System for Mobile Communications. Sistema global para comunicaciones móviles. Sistema de telefonía de segunda generación.

H

HIMM	*Histogramic In-Motion Mapping*. Un método de localización que se usa para determinar si un determinado elemento de una rejilla de ocupación está vacía u ocupado.
Host	Servidor principal.
HTML	*Hypertext Markup Language*. Lenguaje enriquecido para hipertextos. Lenguaje que permite crear hipertexto, en el que están escritas las páginas Web.
HTTP	*Hypertext Transfer Protocol*. Protocolo de transferencia hipertexto.

	Protocolo utilizado para transmitir la información escrita en lenguaje HTML.

I

IA	Inteligencia Artificial.
IDL	*Interface Definition Language*. IDL es un lenguaje que permite definir una serie de interfaces para la comunicación entre objetos distribuidos.
idlj	Compilador IDL -> Java.
IECAT	*Innovative Educational Concepts for Autonomous and Teleoperated Systems.* Conceptos Educativos Innovadores para Sistemas Teleoperados y Autónomos.
IEEE	*Institute of Electrical and Electronics Engineers*. Instituto de Ingenieros Eléctricos y Electrónicos.
IHR	Interacción Hombre-Robot.
IIOP	*Internet Inter-ORB Protocol*. Protocolo interORB de Internet. IIOP es el protocolo abierto de Internet para la comunicación entre objetos y aplicaciones.
i-mode	Es el nombre comercial del servicio de acceso a contenidos a través del teléfono móvil desarrollado por NTT DoCoMo e introducido en Japón en 1999. En la actualidad, *i-mode* es utilizado por más de 36 millones de usuarios en el país asiático, o lo que es lo mismo, por más del 80% de los clientes de dicha operadora.
IP	*Internet Protocol*. Protocolo de Internet. IP es un protocolo de redes basadas en Internet que provee servicios, sin conexión, a través de múltiples redes de conmutación de paquetes.
IPO	Interacción Persona Ordenador.
IR	*Infra Red*. Infrarrojo. Tecnología de ondas de radio frecuencia.

J

J2ME	*Java 2 Micro Edition*. Versión Micro de Java 2
JAD	*Java Application Descriptor*. Fichero descriptor de aplicación Java.
JAR	*Java Archive.* Fichero ejecutable Java.
JAVA	Lenguaje de programación orientado a objetos diseñado específicamente para su uso en Internet.
JDBC	*Java Database Connectivity.* JDBC es un conjunto de objetos y funciones para que los programadores puedan integrar servicios de bases de datos dentro de sus aplicaciones y applets creados en Java.
Jitter	El cambio del retraso temporal de un paquete a otro.

JMF	*Java Media Framework*. JMF es el API de Java para controlar de la forma más sencilla posible dentro de una aplicación o applet de Java todos los componentes basados en una secuencia de tiempo, como son el sonido y el video.
JPEG	*Joint Photographic Experts Group*. Grupo Unido de Expertos en Fotografía. Estándar ISO para comprimir imágenes estáticas. El estándar equivalente para imágenes en movimiento en MPEG (*Moving Pictures Experts Group*).
JSP	*Java Server Pages*. Páginas de Servidor Java es una tecnología orientada a crear páginas Web con programación en Java.
JVM	*Java Virtual Machina*. Máquina Virtual de Java.

K

KVM	*K Virtual Machine*. Máquina virtual K Java reducida.

L

LAN	*Local Area Network*. Red de Área Local. Red que interconecta, a alta velocidad, una serie de terminales informáticos, permitiendo de esta manera se comparten recursos. Una LAN funciona sobre distancias relativamente cortas uniendo sistemas dentro de un mismo edificio o en edificios próximos.
LEX	El programa `lex` genera analizadores léxicos a partir de una especificación de los componentes léxicos en términos de expresiones regulares.

M

Mbps	Mega bits por segundo, unidad de medida de transferencia de datos.
Middleware	El middleware es la infraestructura lógica que proporcione los mecanismos básicos de direccionamiento y transporte entre un cliente y un servidor.
MIDP	*Mobile Information Device Profile*.
MIT	*Massachusetts Institute of Technology*.
MOM	*Message-Oriented Middleware*. MOM es una solución para crear aplicaciones distribuidas complejas.
MVC	Modelo-Vista-Controlador.

N

NASA	*National Aeronautics and Space Administration*. Agencia Nacional del Espacio y la Aeronáutica.

O

OMG	*Object Management Group*.

OmniORB2	Un ORB para CORBA 2.
ORB	*Object Request Broker*. ORB proporciona mecanismos transparentes de comunicación entre objetos. Se encarga de los detalles de la invocación remota.
OTA	*Over The Air*. Tipo de descarga de MIDlets desde un servidor hasta un dispositivo móvil.

P

PAC	Arquitectura Multiagentes, Presentación /Abstracción /Control.
PC	*Personal Computer*. Ordenador Personal.
PDA	*Personal Digital Assistant*. Asistente Digital Personal. Agenda electrónica portátil con capacidad de procesamiento computacional.
PHP	*Hipertext Preprocesor*. Preprocesador de Hipertexto. Es un lenguaje de programación del lado del sevidor.
PID	*Proportional Integral Derivative.* Controlador Proporcional, Integral, Derivativo.
PNG	*Portable Network Graphics*. Formato de imágenes usado en entornos J2ME.
POO	Programación Orientada a Objetos.
QoS	*Quality of Service*. Calidad de servicio.
Quizzes	Exámenes cortos.

R

RFC	*Request For Call.* Pedido de Comentarios. Documentos de trabajo utilizados por la comunidad de científicos y técnicos encargados de expedir nuevos estándares, regulaciones, y recomendaciones tecnológicas. En el caso de Internet, el IEFT es responsable de publicar los RFCs.
RMI	*Remote Method Invocation*. *Invocación Remota de Métodos.*
Round Robin	Metodología para manejar el acceso de los usuarios en los sistemas de múltiples usuarios.
RTP	*Real Time Protocol*.
RTT	*Round Trip Time.* Tiempo de ciclo.
RV	Realidad Virtual.

S

Servlet	Los servlets son componentes del servidor. Estos componentes pueden ser ejecutados en cualquier plataforma o en cualquier servidor debido a la tecnología Java que se usa para implementarlos. Los servlets incrementan la funcionalidad de una aplicación Web. Se cargan de forma dinámica por el

	entorno de ejecución Java del servidor cuando se necesitan.
SLAM	*Simultaneous Localization and Mapping*. Localización y Mapeado Simultaneo.
SMS	*Short Messages Service*. Servicio de mensajes cortos.
SMTP	Servicio que ofrecen los servidores de Internet por el cual se puede remitir correo electrónico al usuario en el servidor.
Sockets	Conectores. Mecanismo de comunicación entre ordenadores conectados en red.
Swing	Un paquete Java para crear interfaces gráficas.

T

TCP	*Transmission Control Protocol*. Protocolo de control de transmisión.
TCP/IP	*Transmission Control Protocol/Internet Protocol*. Protocolo de control de transmisión/Protocolo Internet. Protocolo de comunicaciones en Internet.
TEAM	*Teleeducation in Aerospace and Mechatronics using a Virtual International Laboratory*.
Telnet	Servicio que prestan los servidores de Internet por el cual se permite al cliente interactuar con el servidor actuando con el sistema remotamente.
Throughput	La cantidad de datos trasmitidos con éxito de un lugar a otro en un periodo de tiempo dado.
TOM	*TeleOperated Machina*. Máquina Teleoperada.

U

UDP	*User Datagram Protocol*. Protocolo de Datagramas de Usuario. Protocolo de capa de transporte que se utiliza como parte de TCP/IP. Provee un servicio sin conexión para procedimientos en el nivel de aplicación. No garantiza entrega, preservación de secuencia ni protección contra duplicación.
UMBmark	*University of Michigan Benchmark Test*. Un método para calcular los errores sistemáticos de la odometría.
UMTS	*Universal Mobile Telecommunication System*. Sistema de telefonía móvil universal. Implementación europea del sistema de telefonía de tercera generación, sucesora de GSM. Está pensado para dar tránsito internacional de video y características personalizadas. Se espera que pueda enviar 2 Mbits/s.
URL	*Uniform Resource Locator*. Localizador Uniforme de Recursos. Forma sintáctica usada para identificar una página de información en Internet. Contiene también el nombre de dominio que identifica al ordenador en Internet, y una descripción jerárquica de la localización del recurso en el

	ordenador.

V

VRML	*Virtual Reality Modeling Language*. Lenguaje para Modelado de Realidad Virtual.

W

WAP	*Wireless Access Protocol*. Protocolo para aplicaciones inalámbricas.
WAP Gateway	Puerta de enlace WAP. Dispositivo traductor entre un servidor HTTP y un cliente WAP que permite la comunicación entre ambos adaptando los protocolos.
WapIDE 3.2.1	Herramienta que proporciona Ericsson para el desarrollo de interfaces WML.
WBMP	*Wireless Bitmap*. Formato de imágenes basado en el BMP (bitmap) adaptado para las comunicaciones inalámbricas, sobre todo usado en entornos WAP.
WITS	*Web Interface for Telescience.*
WML	*Wireless Markup Language*.
WTK104	*Wireless Tool Kit 1.0.4*. Herramienta que Sun Microsystems suministra para el desarrollo de interfaces móviles J2ME.
WWW	*World Wide Web*.

X

XML	*Extensible Markup Language*. Lenguaje enriquecido extensible. Lenguaje utilizado en Web que complementa y mejora al HTML.
X-VRML	VRML basada en XML.

Y

YACC	A partir de un fichero de especificación, `yacc` genera un fichero en C, que contiene un analizador sintáctico ascendente por desplazamiento y reducción, y una función que ejecuta las acciones semánticas que aparecen en el fichero de especificación.

Z

BIBLIOGRAFÍA

Bibliografía

[Abualsamid,99] A. Abualsamid, “Evaluación de las Herramientas de Desarrollo”. Global Communication, 31: 52-55, Diciembre 1999.

[Agah,01] A. Agah, “Human Interactions with Intelligent Systems: Research Taxonomy”, Computers and Electrical Engineering, 27 (2001) 71-107.

[Akamatsu,94-a] M. Akamatsu, “Touch with a Mouse - a Mouse Type Interface Device with Tactile and Force Display”, Proceedings of the 3rd IEEE International Workshop on Robot and human Communication, Nagoya, Japan, July. 1994.

[Akamatsu,94-b] M. Akamatsu, S. Sato, I. Mackenzie, “Multimodal Mouse: a Mouse-type Device with Tactile and Force Display”, Presence 1994; 3:73-80.

[Alami,98] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingrand. "An Architecture for Autonomy". The International Journal of Robotics Research, 17 (4): 315-337, 1998.

[Alavi,95] M. Alavi, B. Wheeler, y J. Valacich, “Using IT to Reengineer Business Education: An Exploratory Investigation of Collaborative Telelearning”, MIS Quartely, 19 (3): 293-312, 1995.

[Alexander,79] C. Alexander. *The Timeless Way of Building*. Oxford University Press, 1979.

[Ando,99] N.Ando, J. Lee, H. Hashimoto, “A Study on Influence of Time Delay in Teleoperation”, Proceeding of the 1999 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Atlanta, USA, pp. 317-322, 1999.

[Apache,03] Página Web de Apache. http://www.apache.org/httpd

[Aracil,02] R. Aracil, “Teleoperación”, III Jornadas de Trabajo Enseñanza vía Internet/Web de la Ingeniería de Sistemas y Automática, Alicante, 18-19 de Abril de 2002, pp.13-18.

[ARITI,03] ARITI, http://lsc.cemif.univ-evry.fr:8080/Projets/ARITI/index.html

[Ashutosh,98] Ashutosh, A. Singh, and S. Banerjee, "A Gesture based Interface for Remote Robot Control", Proc. IEEE TENCON, New Delhi, 1998.

[Atienza,02] R. Atienza y A. Zelinsky, "Active Gaze Tracking for Human-Robot Interaction", The 4th International Conference on Multimodal Interfaces (ICMI'2002), Pittsburgh, 14-16 October 2002.

[AXIS,03] Servidor de Video AXIS, http://www.wepa.com.gt/axis/productos/video.htm

[Backes,00-a] P. Backes, K. Tso, J. Norris, G. Tharp, J. Slostad y R. Bonitz, "Mars Polar Lander Mission Distributed Operations", IEEE Aerospace Conf., Big Sky, USA.

[Backes,00-b] P. Backes, K. Tso, J. Norris, G. Tharp, J. Slostad y R. Bonitz, K. Ali, "Internet-Based Operations for the Mars Polar Lander Mission", IEEE Intl. Conf. on Robotics and Automation, San Francisco, USA, 2000.

[Bapna,98] D. Bapna, E. Rollins, A Foessel, R. Whittaker, "Antenna Pointing for High Bandwidth Communications from Mobile Robots", IEEE Intl. Conf. on Robotics and Automation, ICRA98.

[Barber,00] R. Barber. *Desarrollo de una arquitectura para robots móviles autónomos. Aplicación a un sistema de navegación topológica*. Tesis Doctoral, Departamento de Ingeniería Eléctrica, Electrónica y Automática (UCIIIM), 2000.

[Barber,01] R. Barber, M.A. Salichs, "A new human based architecture for intelligent autonomous robots". The Fourth IFAC Symposium on Intelligent Autonomous Vehicles, pp85-90. Sapporo, Japan. September 2001.

[Barberá,01] H. Barberá. Una Arquitectura Distribuida para el Control de Robots Móviles Autónomos. Un enfoque aplicado a la Construcción de la Plataforma Quaky-Ant. Tesis Doctoral, Universidad de Murcia, 2001.

[Bass,98] L. Bass, P. Clements, y R. Kazman. *Software Architecture in Practice*. 1st Edition, Addison Wesley Professional, 1998.

[Bergamasco,95] M. Bergamasco, "Haptic interfaces: the study of force and tactile feedback systems", Proceedings of the 4th IEEE International Workshop on Robot and human Communication, Tokyo, Japan, July. 1995.

[Biena,03] Z. Biena, and W. Songb, "Blend of soft computing techniques for effective human–machine interaction in service robotic systems", Fuzzy Sets and Systems 134 (2003) 5–25.

[Billings,97] C. Billings, "Issues Concerning Human-Centered Intelligent Systems: What's 'human-centered' and what's the problem?", plenary talk at National Science Foundation Workshop on Human-Centered Systems: Information, Interactivity, and Intelligence, Arlington VA, February 1997.

[Blanco,01] J.A. Blanco. *Desarrollo de Habilidades Automáticas para el Sistema de Navegación EDN*. Proyecto fin de carrera, Departamento de Ingeniería de sistemas y Automática, Universidad Carlos III de Madrid, 2001.

[Boada,02-a] Mª Jesús López. *Sistema de Control para Robots Móviles Autónomos basado en Habilidades Reactivas*. Tesis Doctoral, Departamento de Ingeniería Eléctrica, Electrónica y Automática , UC3M, 2002.

[Boada,02-b] M.J.L. Boada, R. Barber y M.A. Salichs, "Visual Approach Skill for a Mobile Robot Using Learning and Fusion of Simple Skills", Robotics and Autonomous Systems, 38:157-170, 2002.

[Booch,95] G. Booch. *Object Solutions*. Redwood City, Calif.: Addison-Wesley, 1995.

[Borella,97] M. Borella, A. Sears and J. Jacko, "The Effect of Internet Latency on User Perception of Information Content", IEEE Global Telecommunication Conference, GLOBECOM'97, vol.3, 3-8, pp. 1932-1936, 1997.

[Borenstein ,91] J. Borenstein y Y. Koren, "Histrogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance", IEEE Journal of Robotics and Automation, Vol. 7, No. 4, pp. 535-539, 1991.

[Borenstein,96] J. Borenstein y L. FENA, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots", IEEE Transactions on Robotics and Automation, Vol 12, No 5, October 1996.

[Bourne,96] J. Bourne, A. Brodersen, J. Campbell, M. Dawant y R. Shiavi, "A Model for On-line Learning Networks in Engineering Education", Journal of Engineering Education, 85 (3): 253-262, Jul.96.

[Brady,98] K. Brady y T. Tarn, "Internet-based Remote Teleoperation". Proceeding of the 1998 IEEE International Conference on Robotics & Automation pp.65-70, Leuven, Belgium., May 1998.

[Breazeal,01] C. Breazeal, "Emotive qualities in robot speech", Proceedings of the International Conference on Intelligent Robotics and Systems, 2001.

[Breazeal,02-a] C. Breazeal. *Designing Sociable Robots*. MIT Press, 2002.

[Breazeal,02-b] C. Breazeal and B. Scassellati, "Robots that imitate humans", TRENDS in Cognitive Sciences Vol.6 No.11 November 2002.

[Breazeal,98] C. Breazeal, "A Motivational System for Regulating Human-Robot Interaction", AAAI-98. Disponible en: http://citeseer.nj.nec.com/breazeal98motivational.html

[Breazeal,99] C. Breazeal and B. Scassellati, "How to Build Robots That Make Friends and Influence People", In Proceedings of IROS-99, Kyonju, Korea.

[Breemen,03] A. Breemen et al, "A User-Interface Robot for Ambient Intelligent Environments." Proce. 1st International Workshop on Advances in Service Robotics, ASER03, Bardolino, Italy, 2003.

[Brown,00] R.Brown y B. Donald, "Mobile Robot Self-Localization without Explicit Landmarks", Algorithmica 26, pp.515-559, 2000.

[Bruce,02] A. Bruce, I. Nourbakhsh, y R. Simmons, "The Role of Experssiveness and Attention in Human-Robot Interaction", IEEE International Conference on Robotics and Automation, (ICRA'02), Washington D.C, May 11-15, 2002.

[Brugali,02] D. Brugali y M. Fayad, "Distributed Computing in Robotics and Automation", IEEE Transactions on Robotics and Automation, vol. 18, No. 4, August 2002, pp. 409-420.

[Burdea,94] G. Burdea, y P. Coiffet. *Virtual Reality Technology*. New York: Wiley, 1994.

[Buschman,96] F. Buschman, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*. John Wiley & Sons, Chichester, 1996.

[Bush,99] B. Bush, D. Simon, A. Taivalsaari. *The Spotless System: Implementing a Java System for the Palm Connected Organizer*. Sun Labs Technical Report SMLI TR-99-73, Feb., 1999.

[Caldwell,94-a] G. Caldwell y A. Wardle, "Tele-presence: feedback and control of a twin armed mobile robot", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Munich, Germany, vol. 1, September. 1994.

[Caldwell,94-b] G. Caldwell, A. Wardle, M. Goodwin, "Tele-presence: visual, audio and tactile feedback and control of a twin armed mobile robot", Proceedings of IEEE International Conference on Robotics and Automation, San Diego, California, vol. 1, May. ICRA94.

[Camurri,97] A. Camurri, A. Coglio, P. Coletta, and C. Massucco, "An architecture for Multimodal Environment Agents", in *Proc. of Intl. Workshop on Kansei - Technology of emotion*, pp.48-53, 1997.

[Ciao,03] Ciao, http://www.ciao.es/

[Colombo,97] C. Colombo y A. Bimbo, "Interacting Through Eyes", Robotics & Autonomous Systems, 19, pp. 359-368, 1997.

[Conallen,99] J. Conallen. *Building Web Applications with UML*. Addison-Wesley Longman, Inc., 1999.

[Cottrell,01] L. Cottrell, W. Matthews and C. Logg. *Tutorial on Internet Monitoring & PingER at SLAC*. Disponible en: http://www.slac.stanford.edu/comp/net/wan-mon/tutorial.html#jitter

[Crisando,02] P. Crisando, "Cara y Cruz del Móvil en España", Articulo extraído del periódico Cinco Días, del Sábado 23 de noviembre de 2002.

[Crowley,85] J. Crowley, "Navigation for an Intelligent Mobile Robot", IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, pp. 31-41, March 1985.

[Douglas,97] W. Douglas, "Network Protocols for Mobile Robots System". Proceedings of SPIE Mobile Robots XII, Pittsburgh PA, 3210: 14-17, October 1997.

[Draper,94] V. Draper. *Environmental Restoration and Waste Management Program Teleoperator Hand Controllers: Contextual Human Factors Assessment*. A Report Prepared by the OAK Ridge National Laboratory for the U.S. Department of Energy, ORNL/TM-12762, May, 1994.

[Elfes,87] A. Elfes, "Sonar-based Real World Mapping and Navigation". IEEE Journal of Robotics and Automation, 3(3), pp.249-265, 1987.

[Ellis,91] S. Ellis, "Nature and Origins of Virtual Environments: A Bibliographical Essay", Computer Systems in Engineering, vol.2, no. 4, 1991, pp.321-347.

[Ericsson,02] Ericsson. *WapIDE 3.2.1 User Guide*. Abril 2002.

[Essa,97] I. Essa y A. Pentland, "Coding analysis, interpretation, and recognition of facial expressions", IEEE Trans. Pattern Anal. Mach. Intell. 19, 757–763, 1997.

[Fasbender,95] A. Fasbender y P. Davids, "Measurements, Modelling and Emulation of Internet Round-Trip Delays", in Joint Conference Performance Tools'95 and MMB'95, Springer Verlag, LNCS 977, 1995.

[Fitzpatrick,99] T. Fitzpatrick, "Feature Service Robots that think on their feet now possible", http://news-info.wustl.edu/feature/1999/Jan99-robots.html

[Fong,00] T. Fong, F. Conti, S. Grange, y C. Baur, "Novel Interfaces for Remote Driving: Gesture, Háptic and PDA", SPIE 4195-33, SPIE Telemanipulator

and Telepresence Technolgies VII, Boston, MA, 2000.

[Fong,02] T. Fong, I. Nourbakhsh, y K. Dautenhahn, "A Survey of Socially Interactive Robots: Concepts, Design, and Applications", the Robotics Institute Carnegie Mellon University, Report CMU-RI-TR-02-29, 2002.

[Froufe,00] A. Froufe. *Java 2, Manual de Usuario y Tutorial*. 2º edición. Editorial RA-MA. Junio 2000. http://www.members.es.tripod.de/froufe/

[Galeote,00] J. Galeote. *Modelado 3D del Entorno a partir de la Telemetría Láser*. Proyecto Fin de Carrera. Departamento de Ingeniería de Sistemas y Automática, Universidad Carlos III de Madrid, 2000.

[Gardeazabal,02] L. Gardeazabal, N. Garay y J. Abacal, "Emulación de Teclado y Ráton por medio de Pantallas Táctiles", Actas del III Congreso Internacional de Interacción Persona-Ordenador, Universidad Carlos III de Madrid, pp.172-179, 2002.

[Ghidary,01] S. Ghidary, Y. Nakata, H. Saito, M. Hattori, T. Takamori, "Multi-Modal Human Robot Interaction for Map Generation", IEEE International Conference on Intelligent Robots and Systems, IROS2001, Hawaii, USA, 2001.

[Giguere,03] E. Giguere, "Over-the-Air Provisioning with the J2ME Wireless Toolkit", Sun Microsystems, 2003, Disponible en: http://wireless.java.sun.com/midp/ttips/wtkota/

[Gilliland,94] K. Gilliland, R. Schlegel, "Tactile stimulation of the human head for information display", Human Factors; 36:700-717, 1994.

[Gopalan,00] R. Gopalan, "RMI-CORBA-DCOM", 2000. Disponible en: http://www.crackinguniversity2000.it/cracking/Novita/Comparazione/comparazione.html

[Green,00] A. Green, "Human Interaction with Intelligent Service Robots", American Association for Artificial Intelligence, http://www.aaai.org , disponible en: http://www.nada.kth.se/~green/publications/ files/aaaiss00green.pdf

[Habib,92] M. Habib, H. Asama, Y. Ishida, A. Matsumoto, I. Endo, "Simulation environment for an autonomous and decentralized multi-agent robotic system", Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots Systems, pp.1550-1557, 1992.

[Halme,99] Halme, J. Suomela y M. Savela, "Applying Telepresence and Augmented Reality to Teleoperate Field Robots", Robotics and Autonmous Systems, vol.26, 1999, pp. 117-125.

[Han,01] K. Han, S. Kim, Y. Kin y J. Kim, "Internet Control Architecture for

Internet-based Personal Robot", Autonomous Robot, 10, 135-147, 2001.

[Hannaford,89] B. Hannaford, and W. Kim, "Force reflection, shared control., and time delay in telemanipulations", in Proceedings of the 1989 IEEE Conference on Systems, Man., and Cybernetics, MIT Press, Cambridge, MA, 1989.

[Hara,95] F. Hara, H. Kobayashi, "Use of face robot for human-computer communication", Proc. IEEE Conf. SMC '95, Vancouver, B.C., Canada, 1995, pp. 1515–1520.

[Hardee,00] M. Hardee. "Why Wireless needs Java Technology", 2000, Disponible en: http://java.sun.com/features/2000/07/wireless.html

[Hauser,96] M. Hauser. *The Evolution of Communication*. MIT Press, 1996.

[Hiltz,97] S. Hiltz, "Synchronous Learning Networks as a Virtual Classroom", Communication of the ACM, 40(9): 44-49, 1997.

[Hirche,02] S. Hireche. *Control of Teleoperation Systems in QoS Communication Networks.* Thesis, Technische Universitat Berlin, 2002.

[Houlb,99] A. Houlb, "Building User Interfaces for Object-oriented Systems", Parts 1:6, JavaWorld, July 1999, Disponible en: http://www.javaworld.com/javaworld/jw-07-1999/jw-07-toolbox.html

[Huang,95] Y. Huang, H. Dohi, M. Ishizuka, "A real-time visual tracking system with two cameras for feature recognition of moving human face", In: Proceedings of the 4th IEEE International Workshop on Robot and human Communication, Tokyo, Japan, July. 1995.

[IECAT,03] Página principal del Proyecto IECAT. http://ars-sun4.ars.fh-weingarten.de/iecat/iecat.html

[Insignia,03] Insignia Soultions, http://www.insignia.com/

[IPMA,03] C. Labovitz, F. Jahanian, R. Malan, y A. Ahuja. The Internet Performance and Measurement Analysis Project. Disponible en: http://nic.merit.edu/ipma/

[IPPM,03] Internet Protocol Performance Metrics. The Internet Engineering Task Force Disponible en: http://www.advanced.org/ippm.html

[iRobot,03] Real World Interface, http://www.irobot.com

[ITR,03] Internet Traffic Report. Opnix, Tempe, AZ. Disponible en: http://www.internettrafficreport.com/

[IWR,03] The Internet Weather Report. Matrix.net, Austin, TX. Disponible en:

http://www.mids.org/weather

[Java,03] Java, http://java.sun.com

[Johnsen,71] E. Johnsen y W. Corliss, Human Factors Applications in Teleoperator Design and Applications. New York: Wiley, 1971, pp. 90–92.

[Juric,00] M. Juric, I. Rozman y M. Hericko, "Performance Comparison of CORBA and RMI", Information and Software Technology, 42:915-933, 2000.

[Juric,00] M. Juric, I. Rozman y M. Hericko, "Performance Comparison of CORBA and RMI", Information and Software Technology, 42:915-933, 2000.

[Kanda,01] T. Kanda, H. Ishiguro, and T. Ishida, "Psychological Analysis on Human-Robot Interaction", IEEE International Conference on Robotics and Automation (ICRA 2001), pp.4166-4173, 2001.

[Kaplan,00] F. Kaplan, "Talking AIBO: First Experimentation of Verbal Interactions with an Autonomous Four-legged Robot", In Nijholt, A., Heylen, D. & Jokinen, K. (eds.), *Learning to Behave: Interacting agents.* cele-twente Workshop on Language Technology, pp. 57-63. Disponible en: http://citeseer.nj.nec.com/kaplan00talking.html

[Khamis,02] A. Khamis, M. Pérez Vernet, K. Schilling, "A Remote Experiment on Motor Control of Mobile Robots", The 10th Mediterranean Conference on Control and Automation, MED 2002, Lisbon, Portugal, July 9-12, 2002.

[Knudsen,01] J. Knudsen, "FLASHBOT A Lego Robot Controlled by a J2ME Device", Junio 2001. http://home.sprynet.com/~jknudsen/flashbot/

[Kobayashi,94] H. Kobayashi, T. Une, S. Takahashi, K. Mikuriya, A. Ohya, M. Yanagawa, "An autonomous eye robot for tele-watching", In: Proceedings of 3rd IEEE Workshop on Robot and Human Communication, Nagoya, Japan, July. 1994.

[Kobayashi,95-a] H. Kobayashi, J. Tatsuno, A. Kuroda, S. Matsuyama, K. Yana, H. Mizuta, "A biodynamical measure of master-slave maneuverability", Proceedings of the 4th IEEE International Workshop on Robot and Human Communication, Tokyo, Japan, July. 1995.

[Kobayashi,95-b] H. Kobayashi, A. Tange, F. Hara, "Real-time recognition of six basic facial expressions", In: Proceedings of the 4th IEEE International Workshop on Robot and Human Communication, Tokyo, Japan, July. 1995. p. 179-86.

[Kotoku,91] T. Kotoku, K. Tanie, A. Fujikawa, "Environment Modeling for the Interactive Display (EMID) used in Telerobotic", in Proc. of IEEE Int. workshop on Intelligent Robots and Systems, IROS'91, pp. 999-1004, 1991.

[Kress,01] L. Kress, R. Hamel, P. Murray, y K. Bills, "Control Strategies for Teleoperated Internet Assembly", IEEE/ASME Transactions on Mechatronics, 6(4), 2001, pp.410-416.

[Krueger,91] W. Krueger. *Artificial Reality II.* Addison-Wesley, 1991.

[Kubik,01] T. Kubik y M. Sugisaka, "Use of a Cellular Phone in Mobile Robot Voice Control", Julio 2001. Nagoya.

[Kurose,01] J. Kurose and K. Ross. *Computer Networking*. Addison-Wesley, 2001.

[LabVir,03] Laboratorios Virtuales, Instituto tecnológico y de estudios superiores de Monterrey. Temixco, Morelos, México. http://www.mor.itesm.mx/~lsi/laboratorio_virtual.html

[Liu,00] Liu, Y., Chen, C. and Meng, M. "A Study on the Teleoperation of Robot System via WWW". The 2000 Canadian Conference on Electrical and Computer Engineering, 2: 836 –840, 2000.

[Mahmoud,02] Q. Mahmoud, "J2ME MIDP and WAP Complementary Technologies", February 2002, Disponible en: http://wireless.java.sun.com/midp/articles/midpwap/

[MARON,03] http://www.expo21xx.com/automation77/news/2087_home_robot_fujitzu/news_default.htm

[Mason,94] J. Mason, "Economic FAQs about the Internet", Journal of Economic Perspectives, 1994. http//www.ipps.lsa.umich.edu

[Massimino,93] M. Massimino, T. Sheridan, "Sensory substitution for force feedback in teleoperation", Presence, 1993;2:344-52.

[McCormack,97] C .McCormack, y D. Jones, "Building a Web-Based Education System", John Wiley & Sons, Inc. 1997.

[McGovern,90] D. McGovern, "Experiences and Results in Teleoperation of Land Vehicles", Report SAND87-0646, Sandia National Lab, Albuquerque, PA, 1990.

[Mercury,o3] Mercury Project, University of Southern California, http://www.usc.edu/dept/raiders

[Mobility,99] Mobility, Robot Integration Software. *User's Guide*. Real World Interface. A Division of IS Robotics, Inc., 1999.

[MOM,03] Introduction to Middle-Oriented Middleware, http://dsonline.computer.org/middleware/intro_MOM.html

[Moore,73] M. Moore, "Towards a Theory of Independent Learning and Teaching", Journal of Higher Education, 44, 666-678, 1973.

[Moreno,01] J. Moreno. Un Nuevo Enfoque Metodológico para la Enseñanza a Distancia de Asignaturas Experimentales: Análisis, Diseño y Desarrollo de un Laboratorio Virtual y Remoto para el Estudio de la Automática a Través de Internet. Tesis Doctoral, Universidad Nacional de Educación a Distancia (UNED), 2001.

[Mukherjee,94] A. Mukherjee, "On the Dynamic and Significance of Low Frequency Components of Internet Load", Internetworking: Research and Enterprise, vol. 5, pp. 163-205, 1994.

[Netperf,03] Netperf http://www.netperf.org/netperf/NetperfPage.html

[Nikitas,02] M. Nikitas y S. Gerogiannakis, "Integrating WAP-based Wireless Devices in Robot Teleoperation Environments", Proceedings of the 2002 IEEE International Conference on Robotics & Automation. Washington, DC. Mayo 2002, pp.1191-1196.

[Norman,93] D. Norman. *Things That Make Us Smart: Defending Human Attributes in the Age of the Machine*. Addison-Wesley, New York, 1993.

[Ogata,99] T. Ogata, et al. "Emotional Communication Between Humans and the Autonomous Robot which has the Emotion Model", Proceedings IEEE International Conference on Robotic and Automation ICRA99, pp. 3177-3182, 1999.

[OMG,03] Object Management Group, OMG, http://www.omg.org

[Orfali,98] R. Orfali, D. Harkey, J. Edwards. *Cliente/Servidor: Guía de supervivencia*. 2ª. Edición McGraw-Hill, 1998.

[Papert,91] S. Papert y I. Harel. *Situating Construction. Constructionism*. Norwood, NJ: Ablex Publishing, 1991.

[Pérez,97] Manuel Ferre Pérez. "Diseño de Interfaces Avanzadas para Robots Teleoperados. Desarrollo de un Entorno de Teleoperación con Característica Multimedia". Tesis Doctoral, Universidad Politécnica de Madrid, 1997.

[PingER,03] PingER, http://www-iepm.slac.stanford.edu/pinger/

[Polaroid,03] Página Web de Polaroid, http://www.Polaroid.com

[Ralescu,97] A. Ralescu, R. Hartani, "Fuzzy modeling based approach to facial expressions understanding", J. Adv. Comput. Intell. 1 (1997) 45–61.

[RedRover] RedRover, http://redrover.ars.fh-weingarten.de/

[Renaud,96] E. Renaud. Introduction to client/server systems. A practical guide for systems professionals. 2nd Edition John Wiley &Sons, New York, USA. 1996.

[RFC1889] RTP: A Transport Protocol for Real-Time Applications. Disponable en http://www.faqs.org/rfcs/rfc1889.html

[RFC879,83] RFC 879, Disponible en: http://www.armware.dk/RFC/rfc/rfc879.html

[Ricel,01] J. Ricel, J. Gratch, R. Hill, S. Marsella, and W. Swartout, "Steve Goes to Bosnia: Towards a New Generation of Virtual Humans for Interactive Experiences", the 2001 AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, Technical Report FS-00-04, Stanford University, CA, 2001.

[Riggs,01] R. Riggs, A. Taivalsaari, M. VandenBrink. Programming Wireless Devices with the Java 2 Platform, Micro Edition. Addison-Wesley. 2001.

[Rivero,03] D.M. Rivero, A. Khamis, F.J. Rodriguez, M.A. Salichs, "A Patterns-Oriented Framework for the Development of Automatic and Deliberative skills for mobile robots", The 11th International Conference on Advanced Robotics, ICAR03, University of Coimbra, Portugal, 2003.

[Röfer,98] T. Röfer, and A. Lankenau, "Architecture and Applications of the Bremen Autonomous Wheelchair", In P. Wang (Ed.), *Information Sciences 126*:1-4. Elsevier Science BV (pp. 1-20), 1998. http://ww.informatik.uni-bremen.de/kogrob/talks/jcis98.pdf

[Roger,93] C. Roger, "Asimov's Laws of Robotics: Implications for Information Technology", IEEE Computer. (Published in two parts, in IEEE Computer 26,12 (December 1993) pp. 53-61 and 27,1 (January 1994), pp. 57-66). Disponible en: http://www.anu.edu.au/people/Roger.Clarke/SOS/Asimov.html.)

[Rogers,01] E. Rogers and R. Murphy, "Human-Robot Interaction", *Final Report for DARPA/NSF Study on Human-Robot Interaction*, 2001. http://www.csc.calpoly.edu/~erogers/HRI/HRI-report-final.html.

[Rooy,02] D. Rooy et al, "Using a simulated user to explore human-robot interfaces", Manuscript for Special Issue on Human-Robot Interaction - IEEE Transactions on Systems, Man and Cybernetics,2002. Disponible en: http://www.csc.ncsu.edu/faculty/stamant/ papers/HRI-v4_06-25.pdf

[Saucy,00] P. Saucy y F. Mondada, "KhepOnTheWeb: Open Access to a Mobile Robot on the Internet". IEEE Robotics & automation Magazine, 7 (1): 41-47,

March 2000.

[Sayers,99] C. Sayers. *Remote Control Robotics*. 1st Edition, Springer Verlag, 1999.

[Schaaf,01] M. Schaaf y F. Maurer, "Integrating Java and CORBA: A Programmer's Perspective", IEEE Internet Computing, 5:72-78, 2001.

[Schilling,99] Schilling, K. and H. Roth, "Mobile Robots for Education in Telematics, Control and Mechatronics", Proceedings of 14th IFAC World Congress, Volume M, p. 211 - 215. Beijing, 1999.

[Schulz,00] D. Schulz, W. Burgard, D. Fox, S. Thrun y A. Cremers, "Web Interfaces for Mobile Robots in Public Places". IEEE Robotics & automation Magazine, 7 (1): 48-56, March 2000.

[Sears,97] A. Sears, J. Kacko, y M. Borella, "Internet Delay Effects: How Users Perceive Quality, Organization, and Ease of Use of Information", in Proceedings, ACM SIGGHI Conference Companion, (Atlanta, GA), pp.353-354, Mar. 1997.

[Sheridan,78] T. Sheridan, y L. Verplank. *Human and Computer Control of Undersea Teleoperators*. Technical Report for the Office of Naval Research, Massachussetts Institute of Technology, Cambridge, MA, 1978.

[Sheridan,92] T. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press, Cambridge, MA, p. 6, 1992.

[Sheridan,92] T. Sheridan. Telerobotics, Automation, and Human Supervisory Control. MIT Press, Cambridge, USA, 1992.

[Sheridan,97] T. Sheridan, "Eight Ultimate Challenges of Human-Robot Communication", In Proceedings RO-MAN 1997, IEEE, Piscataway.

[Sick,03] Página Web de Sick, http://www.sick.com

[Simmons,00] R. Simmons y J. Fernandez, R. Goodwin, S. Koeing y J. O'Sullivan, "Lessons Learned from Xavier". IEEE Robotics & automation Magazine, 7 (2): 33-39, June 2000.

[Sony,03] http://www.fabrimex.ch/pdfs/SNC-RZ30PGB.pdf

[Stein,00] M. Stein, "Interactive Internet Artistry: Painting on the World Wide Web with the PumaPaint Project", IEEE Robotics and Automation MAgazine, 7(2), June 2000.

[Stiefelhagen,97] R. Stiefelhagen, y J. Yang, "Gaze Tracking for Multimodal Human-Computer Interaction", Proc. of ICASSP, Munich, Germany, 1997.

[Sun,02] Sun. Wireless Toolkit Version 1.0.4, Java(TM) 2Platform, Micro edition, User Guide. Sun Microsystems, Inc. Palo Alto, USA, Junio 2002.

[Suzuki,98] K. Suzuki, A. Camurri, P. Ferrentino, S. Hashimoto, "Intelligent agent system for human-robot interaction through artificial emotion", in Proc. IEEE Intl Conf. On System Man and Cybernetics (SMC'98), San Diego, CA, 1998.

[Tach,88] S. Tach, H. Arai, T. Maeda, "Tele-existence Simulator with Artificial Reality", in Proc. of IEEE Int. workshop on Intelligent Robots and Systems, IROS'88, pp. 719-724, 1988.

[Taylor,00-a] K. Taylor y B. Dalton, "Distributed Robotics over the Internet", IEEE Robotics and Automation Magazine, 7(2):22-27, 2000.

[Taylor,00-b] K. Taylor y B. Dalton, "Internet Robots: a New Robotics Niche", IEEE Robotics and Automation Magazine, 7(1):27-34, 2000.

[TEAM,03] TEAM, http://www.ars.fh-weingarten.de/team/

[Telegarden,03] Telegarden, http://www.usc.edu/dept/garden/

[Thurn,00] S. Thurn, M. Beetz, M. Bennewitz, W. Burgard, A. Creners, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Probablistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva", International Journal of Robotics Research 19(11), 972-999, 2000.

[TIPHON,98] General aspects of Quality of Service (QoS). DTR/TIPHON-05001 V1.2.5 Technical Report, 1998.

[TNSofres,03] Taylor Nelson Sofres Interactive, http://www.intersearch.tnsofres.com

[Torres,02] F. Torres, F. A. Candelas, J. Pomares, S. T. Puente, F. G. Ortiz, P. Gil, "Laboratorio Virtual Remoto para la Enseñanza de Robótica", III Jornadas de Trabajo Enseñanza vía Internet / Web de la Ingeniería de Sistemas y Automática, Alicante, 18-19 de Abril de 2002.
RoboLab: http://www.disclab.ua.es/gava/inf.html

[Torres,03] G. Torres, "Telefónica Móviles, análisis fundamental", Semanario Económico Inversión y Capital, Nº 450, Febrero, 2003.

[Trinidad] M. Trinidad, "Telecontrol para el robot Nomad 2002", Disponible en: http://www.mor.itesm.mx/~lsi/redii/telecontrol/control_distancia.html

[Tso,98] K. Tso, P. Backes, G. Tharp, "Mars Pathfinder Mission Internet-Based Operations using Wits", IEEE Intl. Conf. on Robotics and Automation,

Leuven, Belgium, 1998.

[ttcp,03] ttcp http://ftp.arl.mil/~mike/ttcp.html

[UC3M] http://modelado1.uc3m.es/eval

[UNESCO,87] UNESCO, "Distance Learning System and Structures: Training Manual, Report of a Sub-Regional Training Workshop", Vol. II, Bangkok, 1987.

[Vernet,01] M. Pérez Vernet, K. Schilling. "Virtual Reality for Tele-Education Experiment with Remote Mobile Hardware", Proceedings of the IFAC Workshop on Internet Based Control Education, pp. 97-102, Madrid, (2001).

[Vertut,85] J. Vertut, R. Foutnier, B. Espiau, y G. Andre, "Sensor-aided and/or Computer-aided Bilateral Teleoperator System (SCATS), in Theory and Practice of Robots and Manipulators", Proceedings of the RoManSy '84: The 5th CISM-IFToMM Symposium, MIT press, pp.281-292, 1985.

[WAP,98] Wireless Application Protocol Architecture Specification, 1998, Disponible en: http://www.wmlclub.com/docs/especwap1.2/SPEC-WAPArch-19980430.pdf

[Webster,97] J. Webster y P. Hackley, "Teaching Effectiveness in Technology-Mediated Distance Learning", Academy of Management Journal, 40(6): 1289-1309, 1997.

[Whalen,98] T. Whalen y D. Wright, "Distance Training in the Virtual Workspace", In M. Igbaria, M. Tan. (Eds.). The Virtual Workspace. Herahey, PA: Idea Group Publishing, 1998.

[Whatis,03] http://www.whatis.com

[WITS,03] NASA Web Interface for Telescience , http://wits.jpl.nasa.gov:8080/WITS

[Yacoob,96] Y. Yacoob, L. Davis, "Recognizing human facial expressions from long image sequences using optical Now", IEEE Trans. PAMI 18 (1996) 636–642.

[Yerxa,99] M.Yerxa, "Las Mejores Apuestas para el Desarrollo Web", Revista Global Communications, diciembre 1999.

[Yoerger,90] D. Yoerger, "The Supervisory Control of Underwater Telerobots", En Robotics, Control, and Society, by N. Moray, W. R. Ferrell, and W. B. Rouse (eds.), (Taylor & Francis Ltd., London), pp. 53-54, 1990.