UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

CONSTRUCCIÓN DE UNA APLICACIÓN QUE INTERACTÚE CON UN ROBOT A TRAVÉS DE LA TECNOLOGÍA JINI.

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado:.....20 puntos. Mención publicación.....

JURADO EXAMINADOR

Firma:	Firma:	Firma:
Nombre: Fung, Jacinto	Nombre: Pereira, Wilmer	Nombre: Soto, Andrés

REALIZADO POR	Daniel E. Vera Gonzalez
PROFESOR GUÍA	Dr. Wilmer E. Pereira
FECHA	18 de Octubre de 2001

Resumen General

Este proyecto consiste en realizar un trabajo de investigación acerca de cómo puede la nueva tecnología JINI controlar remotamente un dispositivo de Hardware a través de una red local. Esta investigación se limita en trabajar únicamente con un robot que represente a estos dispositivos.

Dado esto, el objetivo principal es construir una aplicación que permita interactuar con un robot a partir de la tecnología JINI. Pero no con cualquier robot, específicamente se utiliza el sistema de invención de robot de Lego Mindstorm que posee como componente principal el RCX. Este dispositivo tiene una pequeña unidad de procesamiento con memoria, Sistema Operativo, así como sensores y motores que permiten construir robots que se desenvuelvan en un ambiente específico.

JINI es una tecnología de red creada por los mismos autores del lenguaje de programación Java que permite flexibilizar las labores de uso y administración de las redes. Esta tecnología sigue un modelo cliente/servidor. El servidor publica los servicios y luego los clientes obtienen estos servicios para su debida utilización. Esto se hace gracias al llamado servidor de búsqueda, que funciona como un ente centralizado que permite publicar estos servicios y hacerlos disponibles en la red según un modelo de arrendamiento(cuanto tiempo va a estar disponible este servicio para su utilización).

JINI se basa principalmente en tres protocolos llamados: Discovery, Join y Lookup. El primero se encarga de descubrir cualquier servidor de búsqueda que hay en la red, el segundo permite publicar un servicio en este servidor y Lookup se encarga de obtener el servicio.

JINI está montado sobre RMI (Remote Method Invocation) para correr objetos e máquinas virtuales diferentes.

Luego de un análisis detallado de las herramientas necesarias, se realiza u experimento que incluye un robot capaz de explorar un terreno específico en búsqueda d obstáculos. Este robot es teledirigido por una aplicación que a la vez es controlada por un operador ubicado en la red. Este operador aparte de manipular el robot puede visualizar la composición del terreno lo que le permite ver si hay o no obstáculos en la superficie y conocer dónde están ubicados estos. También existen cierto número de usuarios en espera del recurso que pueden visualizar la exploración hecha por el operador del robot, así como recibir el control del dispositivo una vez que este es liberado.

Índice General

Capítulo/ Sección	Título	Página
I	Introducción	1
I.1	Planteamiento del problema	1
I.2	Objetivo General	3
I.3	Objetivos Específicos	3
I.4	Alcances y limitaciones	4
II	Marco Teórico	5
II.1	RMI	5
II.1.1	Concepto	5
II.1.2	Características de RMI	6
II.1.3	Interfaces	7
II.1.4	Arquitectura	8
II.2	Tecnología JINI	12
II.2.1	Filosofía JINI	12
II.2.2	¿Qué es JINI?	13
II.2.3	El Ambiente JINI	14
II.2.4	Perspectiva general	14
II.2.5	Segmentación de JINI	17
II.3	JINI y RMI	21
II.4	Lego Mindstorm	23
II.4.1	¿Qué es Lego Mindstorm?	23
II.4.2	¿Qué es el RCX?	23
II.4.3	Tipos de Firmware	24
II.4.4	Hardware	25
II.4.5	Programación del RCX	27

III	Marco Metodológico	29
IV	Desarrollo	33
IV.1	Fase de Principio	35
IV.2	Fase de Elaboración	37
IV.3	Fase de Construcción	39
IV.4	Fase de Transición	44
V	Resultados	46
V.1	Fase de Principio	46
V.2	Fase de Elaboración	47
V.3	Fase de Construcción	54
V.4	Fase de Transición	63
VI	Conclusiones y Recomendaciones	66
VI.1	Conclusiones	66
VI.2	Recomendaciones	68
---	Bibliografía	69

Índice de figuras

Figura	Título	Página
1	Invocación a métodos Remotos en Máquinas Virtuales diferentes	6
2	Separación de la Interfaz de una clase y su implementación en RMI	7
3	Servicios de una interfaz en RMI	8
4	Arquitectura RMI	9
5	Proceso de invocación de métodos en RMI	10
6	Obtención del objeto stub a tiempo de ejecución	11
7	Discovery	19
8	Join	19
9	Lookup	20
10	El cliente usa el servicio	20
11	Arquitectura del RCX para el Firmware estándar	25
12	Un ciclo con sus fases e iteraciones	30
13	Cada iteración puede utilizar cada uno de los cinco principales flujos de trabajo	31
14	Las iteraciones se extienden a través de los flujos de trabajos solapándose entre sí	31
15	El énfasis de cada flujo de trabajo varía por cada iteración en cada fase	32
16	Actividades realizadas en cada fase del proyecto	35
17	Arquitectura para la manipulación local del robot	52

18	Arquitectura para la manipulación remota del robot	54
19	Prototipo del robot Signus Explorer	55
20	Prototipo del robot Omega Centaury	55
21	Prototipo de la aplicación	58
22	Proceso de obtención del servicio a través del componente JINI	60
23	Segmentación del cliente en el ambiente distribuido	62
24	Ejecución del experimento.	64
25	Representación real del terreno	65

Índice de tablas

Tabla	Título	Página
1	Iteraciones por cada fase del proyecto	34
2	Actividades realizadas en cada fase del proyecto	34
3	Especificaciones del 1er robot realizado	55
4	Especificaciones del 2do robot realizado	55
5	Especificaciones de los programas descargados al RCX	57

I.1 Planteamiento del problema

Actualmente se ha desarrollado un gran número de experimentos que incluyen inteligencia artificial o robótica con el fin de crear seres independientes que puedan tomar sus propias decisiones o realicen labores específicas que permitan al hombre satisfacer sus necesidades de una manera más fácil.

Aunque esto es de gran importancia, la independencia de estos robots es relativa. Esto puede deberse a limitaciones de la tecnología actual o un deficiente desarrollo en el área de inteligencia artificial. Por lo tanto hoy en día estos dispositivos necesitan de alguna manera la intervención humana para controlarlos.

Por esta razón, una política es manipular estos dispositivos con un computador que sirva como interfaz entre el operador y el robot, estableciendo un canal de comunicación entre ambos elementos.

Pero, ¿qué sucedería si este dispositivo debe ser controlado por varias personas en diferentes lugares?

Desde años atrás, la tecnología posee mecanismos que permiten la comunicación de varios computadores a través de las redes, pero ¿ podrían estos dispositivos adaptarse a la comunicación proveniente de estas redes? Si esta solución fuera posible ¿existe alguna manera eficiente de utilizar estas redes para que el mantenimiento de la comunicación sea más fácil de realizar independientemente de la plataforma en la cual trabaje cada computador?

Tomando como premisa principal de este trabajo de investigación la utilización de la reciente tecnología JINI lanzada por Sun Microsystems, ¿qué ventajas puede ofrecer ésta en la manipulación de los dispositivos a través de la red?

Apoyándonos en la premisa anterior, se realizará un experimento que permita controlar un robot teledirigido (no autónomo), que logre mostrar las diferentes funcionalidades de la tecnología JINI y la influencia de esta sobre el robot.

En este trabajo de investigación es utilizado lego Mindstorm como sistema de invención de robot por las siguientes razones:

- Permite crear diferentes tipos de robots capaces de realizar tareas específicas.
- Es económico.
- Existe abundante información acerca de su utilización así como una gran cantidad de recursos que permiten interactuar con el producto.
- Recurso disponible en el GIIA (Grupo de Investigación de Inteligencia Artificial) de la UCAB.

A lo largo del proyecto se tratará de dar respuestas a las preguntas mencionadas anteriormente teniendo como directriz los siguientes objetivos.

I.2 Objetivo General

Construir una aplicación que interactúe con un robot a partir de la tecnología JINI.

I.3 Objetivos Específicos

1. Investigar y estudiar la tecnología JINI.

2. Investigar y estudiar el lenguaje de programación Java, especialmente el uso de RMI (Invocación de Métodos Remotos).

3. Investigar y estudiar los dispositivos compatibles con Lego Mindstorm.

4. Diseñar un componente JINI, que permita interactuar con un robot.

5. Investigar distintas propuestas tecnológicas para su respectiva comparación con la tecnología JINI.

6. Realizar varios experimentos de navegación robótica que permitan establecer una comunicación entre la tecnología JINI y el dispositivo, en particular, una aplicación capaz de controlar y recibir información a partir de un robot que explore una superficie limitada y permita identificar la posición exacta de cuerpos u obstáculos dentro del área.

7. Diseñar y construir un robot que permita cumplir con los objetivos anteriores.

I.4 Alcances y limitaciones

El experimento se basa en la utilización de Lego Mindstorm como dispositivo de hardware capaz de ser controlado remotamente. El es el único recurso con el que el GIIA Grupo de Investigación de Inteligencia Artificial de la UCAB cuenta actualmente.

El desempeño del robot dependerá de las limitaciones específicas de Lego Mindstorm. Este producto posee fuertes limitaciones sobre todo en la parte de comunicación inalámbrica la cual presenta errores circunstanciales (que dependen del ambiente) asimismo la falta de precisión realizada en la rotación de los motores.

El objetivo de este trabajo no es la construcción de un robot óptimo que permita cumplir con el experimento. Sin embargo, este debe cumplir con los requerimientos mínimos para efectuar la experiencia.

En relación con el experimento, el objetivo de este trabajo de investigación es establecer la comunicación entre un usuario remoto y el robot a partir de la tecnología JINI. Por esta razón, la actuación del robot está estrechamente relacionada con el usuario que lo controla. De esta manera no se considera cualquier parte relacionada con Inteligencia Artificial que permita al robot aprender y tomar sus propias decisiones.

En conclusión se diseñará una arquitectura para un robot teledirigido procurando disminuir los posibles errores causados por algunos componentes del robot y el ambiente en el cual este se desenvuelve.

II.1 RMI (Invocación de Métodos Remotos)

A continuación se estudian varios puntos acerca de RMI por ser este el mecanismo de comunicación principal utilizado por la tecnología JINI. Por lo tanto también es empleado en el proyecto para invocar métodos que permitan manipular el robot remotamente.

II.1.1 Concepto

Cuando construimos aplicaciones distribuidas es posible aprovechar en toda su extensión el paradigma de la programación orientada a objetos. Carlos Beltrán (1998-1999) afirma que "los objetos no son sólo elementos independientes, sino que además se encuentran distribuidos en diversos ordenadores conectados a través de una red". Analizando la visión original del paradigma orientado a objetos se puede decir que nos acercamos un poco más a cómo funciona el mundo real, donde los objetos que manejamos diariamente no suelen estar en el mismo sitio.

Según Carlos Beltrán (1998-1999) uno de los objetivos primarios de los diseñadores de RMI era permitir a los programadores desarrollar programas Java distribuidos usando la misma sintaxis y semántica usadas para los programas no-distribuidos. Para hacer esto, ellos tuvieron que convertir cuidadosamente la forma en que las clases y objetos Java trabajan en una Máquina Virtual Java™ (JVM) individual, hacia un nuevo modelo de cómo las clases y objetos trabajarían en un ambiente de computación distribuida (con múltiples JVM).

En la mayoría de los casos, las aplicaciones distribuidas funcionan siguiendo un modelo cliente/servidor. Uno o más servidores crean unos objetos locales y luego atienden peticiones de acceso sobre esos objetos provenientes de clientes situados en lugares remotos

de la red. Por lo tanto, de forma general, podemos decir que las necesidades de una aplicaci distribuida son:

- Mecanismos para localizar los objetos en la red.
- Comunicación con los objetos remotos.
- Mecanismos de intercambio de información (paso de parámetros).

Diversas tecnologías nos proporcionan estos servicios. En el apéndice A se muestra una d estas alternativas. La idea básica de RMI es que objetos ejecutándose en una JVM sear capaces de invocar métodos de objetos presentes en JVMs diferentes, haciendo notar que las JVMs pueden estar en la misma máquina o en máquinas distintas conectadas por una red.(Ver figura 1)

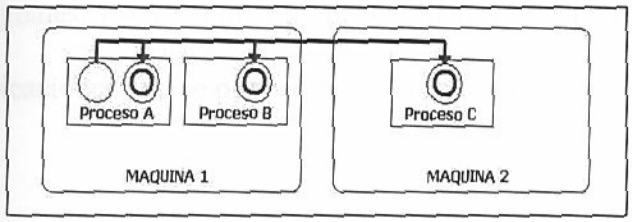


Figura 1: Invocación a métodos Remotos en Máquinas Virtuales diferentes.

En la figura 1 se muestra cómo el proceso "A" ubicado en la máquina 1 invoca un método implementado en el proceso C de la máquina 2.

II.1.2 Características de RMI

Carlos Beltrán (1998-1999) expresa que entre las principales características de RMI se pueden encontrar los siguientes aspectos: sencillez, transparencia, paso de objetos por valor (como parámetros de los métodos), implementación 100% JAVA así como independencia del protocolo de comunicación.

En RMI, cuando los objetos remotos ya han sido referenciados, el programador los puede utilizar igual que si estuviesen en la máquina local, accediendo a sus métodos y manejándolos

localmente. La capa RMI permanece oculta al programador proporcionando mayor clarida código y mayor comodidad a la hora de programar la aplicación.

"El paso de objetos por valor se revela como una de las características que diferencia RMI de las otras tecnologías de programación distribuida. Esta potente posibilidad se basa el uso del concepto de "serialización" para el paso de objetos a través de la red. Objetos todo tipo, tanto nativos JAVA como definidos por el usuario, pueden ser transferidos de for totalmente transparente al programador usando esta técnica" Carlos Beltrán (1998-1999).

Por otro lado, la principal ventaja de RMI, su implementación 100% JAVA, es también principal inconveniente. RMI no puede ser utilizado por otros lenguajes de programación, p lo que no es posible utilizarlo en aplicaciones distribuidas donde se utilizan distint lenguajes. Por lo tanto, en la actualidad, las soluciones basadas en RMI se reducen a aquella aplicaciones que se puedan implementar completamente en JAVA.

II.1.3 Interfaces: El corazón de RMI

Específicamente, en RMI la definición de un servicio remoto se codifica usando una interfaz Java. La implementación del servicio remoto se codifica en una clase. Por lo tanto, l clave para comprender RMI es según Carlos Beltrán (1998-1999) que "*las interfaces definen el comportamiento* y *las clases definen la implementación*". La Figura 2 ilustra est separación.

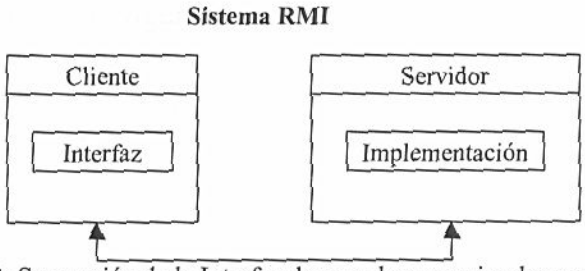


Figura 2: Separación de la Interfaz de una clase y su implementación en RMI

Sin embargo, hay que recordar que una interfaz Java no contiene código ejecutable. RMI soporta dos clases que implementan la misma interfaz. La primera clase es la implementación del comportamiento, ella es ejecutada en el servidor. La segunda clase actúa como un representante local del servicio remoto y es ejecutada en el cliente. La figura 3 muestra un diagrama que ilustra esta condición.

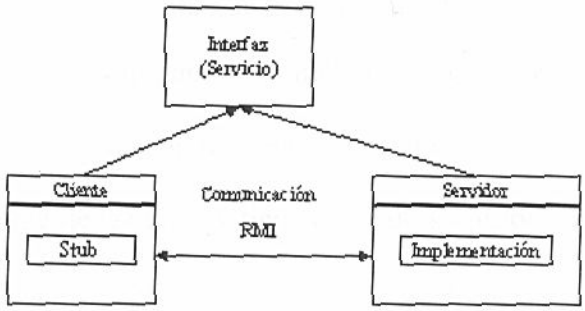


Figura 3: Servicios de una interfaz en RMI

Un programa cliente hace llamadas a métodos en el objeto representante, RMI envía la petición a la máquina virtual remota, y la remite a la implementación. Cualquier valor de retorno suministrado por la implementación es enviado de regreso al objeto representante y de ahí al programa del cliente.

II.1.4 Arquitectura RMI

La arquitectura de RMI descrita por Carlos Beltrán consta de tres capas:

- La capa stub.
- La capa de referencia remota.
- La capa de transporte, (Ver figura 4).

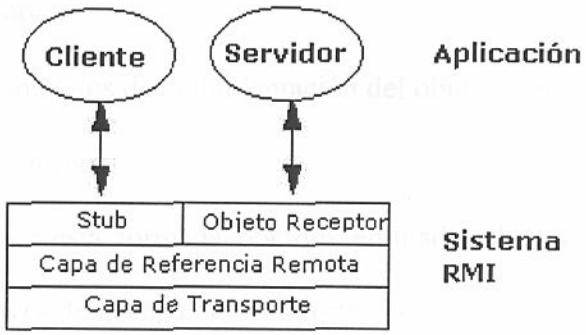


Figura 4: Arquitectura RMI

En la figura 4 se puede observar que el "Stub" se encuentra en el cliente y el objeto receptor en servidor, seguido de la capa de referencia remota y la capa de transporte.

Cada capa es independiente de las otras y tiene definido su propia interfaz y protocolo, de forma que una capa puede ser cambiada sin afectar a las otras. Por ejemplo, la capa de transporte puede utilizar distintos protocolos como TCP, UDP entre otros, sin que el resto de las capas se vean afectadas en su funcionamiento. "La comunicación entre las capas se realiza por medio de la abstracción de stream o flujos de datos" Carlos Beltrán (1998-1999).

El punto de contacto de la aplicación cliente con el objeto remoto se hace por medio de un "stub" local. Este "stub" implementa la interfaz del objeto remoto y gestiona toda la comunicación con el servidor a través de la capa de referencia remota. A todos los efectos, el "stub" es la representación local del objeto remoto. Entre sus responsabilidades destacan:

- Inicializar las llamadas a los objetos remotos.
- Serializar los argumentos para enviarlos por la red.
- Deserializar los argumentos devueltos en las llamadas.

En la parte del servidor, existe un objeto receptor equivalente al "stub" en el cliente. Este se encarga de traducir las invocaciones provenientes de la capa de referencia remota, así como de gestionar las respuestas. Entre sus actividades se destacan:

- Deserializar los argumentos.
- Hacer las llamadas a los métodos de la implantación del objeto remoto.
- Serializar los valores de retorno.

La capa de referencia remota está formada por dos entidades distintas, el cliente y el servidor, que se comunican a través de la capa de transporte. Es responsable de implementar la política de comunicación, que puede ser de distintos tipos:

- Invocación unicast punto-punto.
- Estrategias de reconexión.

La capa de transporte es responsable del establecimiento y mantenimiento de la conexión, proporcionando un canal de comunicación fiable entre las capas de referencia remota del cliente y del servidor. Sus principales responsabilidades son:

- Establecimiento y mantenimiento de la conexión.
- Atender a llamadas entrantes.
- Establecer la comunicación para las llamadas entrantes.

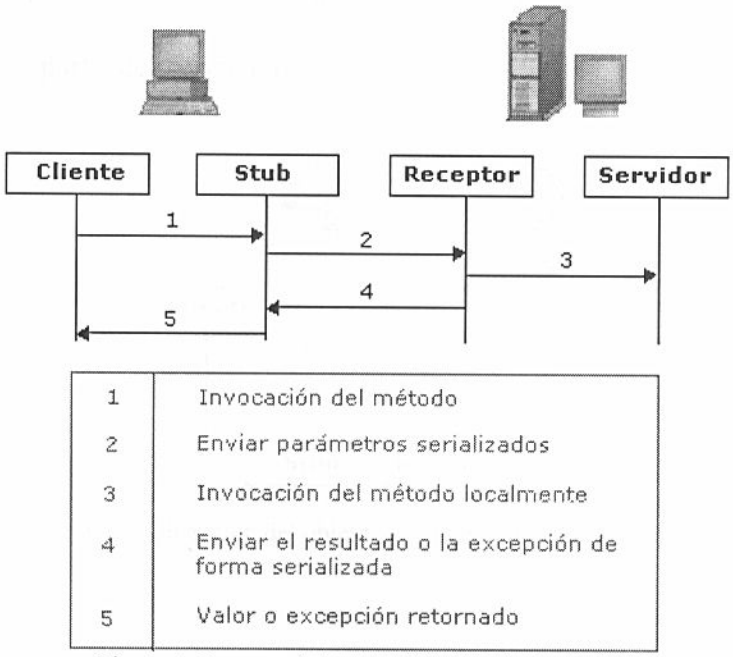


Figura 5: Proceso de invocación de métodos en RMI

Como se puede observar en la figura 5 el cliente invoca el método a partir del objeto "stub" ubicado en la JVM local, este objeto se encarga de serializar los parámetros para luego enviarlos al objeto receptor que se encarga de invocar el método localmente. Una vez obtenido el resultado o la excepción correspondiente, el objeto receptor envía este valor al "stub" que se encarga de deserializar el resultado para luego retornarlo.

De esta manera para acceder a un objeto remoto existente en el servidor, el cliente necesita un objeto "stub" local. RMI también posee mecanismos que permiten obtener dinámicamente este objeto mediante el "Bootstrap Registry Service" que es proporcionado por la librería de Sun RMI.

Un programa localizado en el servidor se registra con este servicio y el cliente obtiene el "stub" perteneciente al objeto remoto. Según Cay S. Horstmann y Gary Cornell (2000), los objetos del servidor son registrados en el "Bootstrap Registry Service" a partir de una referencia al objeto y un "String" correspondiente al nombre del objeto (debe ser único en toda la red). En RMI el proceso realizado por el cliente y que se encarga de buscar el objeto "stub" en el servicio de registro se denomina lookup. En la figura 6 se muestra el proceso de obtención del objeto a partir del servicio de registro.

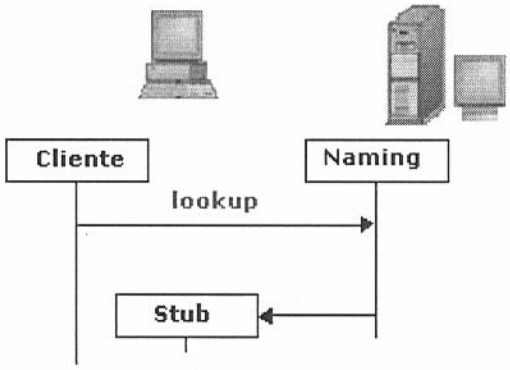


Figura 6: Obtención del objeto stub a tiempo de ejecución.

Tecnología JINI

Como se mencionó anteriormente el experimento se basa en un robot manipulado remotamente a partir de la tecnología JINI. También se describió brevemente a RMI por ser utilizada por JINI para movilizar objetos de una JVM a otra. A continuación se muestra una breve explicación acerca de los puntos más importantes sobre la tecnología JINI que luego serán utilizado a lo largo de este trabajo de investigación.

II.2.1 Filosofía de la tecnología JINI

"La tecnología de conectividad JINI™ está basada en un concepto sencillo: los dispositivos deben trabajar en conjunto. Deben interconectarse sin problema. Sin drivers, ni problemas con sistemas operativos, ni cables o conectores extraños. No obstante, la tecnología ofrece nuevos e interesantes conceptos y capacidades" Sun Microsystems (2000):

- *Operación inmediata:* cuando se conecta un dispositivo con tecnología JINI a la red, sencillamente funciona. Los servicios y recursos están disponibles de inmediato.
- *Comunidades improvisadas:* El software permite que todos los dispositivos trabajen en conjunto, de forma que se pueda crear una red propia o comunidad, a cualquier hora y en cualquier lugar. Se puede interconectar los aparatos eléctricos para controlarlos desde un punto centralizado por varios usuarios.
- *Flexible:* Las comunidades que utilizan la tecnología JINI se adaptan muy rápidamente a los cambios. Aunque los usuarios van y vienen, la comunidad continúa existiendo. Esta siempre está disponible, haciendo posible que los sistemas sean más tolerantes y redundantes.

• *Entrega especial:* Los servicios con tecnología JINI están disponibles en todo momento, cuando se requieran.

II.2.2 ¿Qué es JINI?

Según Sun Microsystems (2000), un sistema JINI es un sistema distribuido o federación basado en la idea de implementar grupos de usuarios y los recursos requeridos por esos usuarios. La meta global es la de convertir la red en una flexible y fácil herramienta de administración en la cual los recursos pueden ser encontrados por clientes humanos y procesos. Los recursos pueden ser dispositivos de hardware, programas de software, o una combinación de ambos. El foco del sistema es hacer de la red una entidad más dinámica que permita flexibilizar la capacidad de agregar y eliminar servicios. El sistema JINI consta de las siguientes partes:

- Un conjunto de componentes que provee una infraestructura para compartir servicios en un sistema distribuido.
- Un modelo de programación que soporta y fomenta la producción de servicios distribuidos confiables.
- Servicios que pueden ser parte de un sistema compartido JINI y que ofrece funcionalidad a cualquier otro miembro del grupo.

Aunque estos aspectos son diferentes, están interrelacionados entre sí. Sun Microsystems (2000) indica que los componentes que conforman la infraestructura de la tecnología JINI utilizan un modelo de programación especial; los servicios que residen dentro de la infraestructura también usan ese modelo, y a la vez ese modelo de programación es bien soportado por los componentes en la infraestructura. Las metas finales del sistema se pueden resumir en:

- Habilitar a los usuarios para compartir servicios y recursos a través de la red.
- Proveer a los usuarios un acceso fácil a los recursos en cualquier parte de la red, permitiendo cambios en la localización del usuario.
- Simplificar las tareas de construir, mantener y alterar los dispositivos, software y usuarios de una red.

II.2.3 El Ambiente JINI

Según Sun Microsystems (2000), el sistema JINI se encarga de agrupar computadoras y dispositivos dentro de lo que aparenta al usuario un sistema individual. Este cree en la existencia de una red de velocidad razonable conectándose con estos computadores y dispositivos. Algunos dispositivos requieren de mucho más ancho de banda y otros no (pantallas e impresoras son un ejemplo de puntos extremos). Se asume que la latencia de la red es razonable.

También se asume que cada dispositivo habilitado para la tecnología JINI posee memoria y poder de procesamiento. Dispositivos sin poder de procesamiento o memoria podrían ser conectados a un sistema JINI, pero deben ser controlados por otra pieza de hardware y/o software que represente al dispositivo dentro del sistema JINI.

II.2.4 Perspectiva general del Sistema

Según Sun Microsystems (2000), la tecnología JINI es una extensión más allá de la red física y espacio de dispositivo para abarcar el nivel arquitectónico de la red y el comportamiento de servicios de red. JINI puede ser usado para construir comunidades de servicio fuera de simples piezas de hardware y software.

Servicios

El concepto más importante dentro de la arquitectura JINI es el de servicio. Sun Microsystems (2000) define un servicio como una entidad que puede ser usada por una persona, un programa, u otro servicio. Este puede ser un canal de cómputo, almacenamiento o comunicación con un usuario, un filtro de Software, un dispositivo, u otro usuario. Dos ejemplos de servicios son la impresión de un documento y el traslado de un formato de procesador de palabras a otro.

Los miembros de un sistema JINI son agrupados para compartir accesos a servicios. El sistema consiste en un conjunto de servicios que pueden ser agrupados para desarrollar una tarea específica. Estos servicios pueden hacer uso de otros servicios y el cliente de un servicio puede ser un servicio.

Según Sun Microsystems (2000), los servicios en un sistema JINI se comunican unos a los otros usando un protocolo específico, el cual es un conjunto de interfaces escritas en el lenguaje de programación Java. La base del sistema JINI es definir un pequeño número de tales protocolos que definen servicios críticos de interacción.

Servicio de Búsqueda

Los servicios son encontrados a partir de un servicio de búsqueda. Sun Microsystems (2000) define que éste es el mecanismo central de rastreo del sistema y provee el mayor punto de contacto entre el usuario y el sistema. En pocas palabras un servicio de búsqueda expone ciertas interfaces indicando así la funcionalidad que este servicio provee a un conjunto de objetos que implementen el servicio. Esta información se amplía en el apéndice B.

Invocación de métodos remotos (RMI)

Según Sun Microsystems (2000), la comunicación entre servicios es llevada a cabo usando la invocación de métodos remotos de Java (RMI). El mecanismo que soporta la comunicación entre los servicios no es por si mismo un recurso que es encontrado y usado, sino que este es una parte de la infraestructura de la tecnología JINI. RMI provee mecanismos para encontrar, activar y eliminar grupos de objetos.

Seguridad

Sun Microsystems (2000) indica que el diseño del modelo de seguridad para la tecnología JINI es construido a partir de la doble noción de un director y una lista de control de acceso. Este es complementado en el apéndice B.

Modelo de Arrendamiento

Al acceder a muchos de los servicios en el ambiente del sistema JINI, éste se transforma en un modelo de arrendamiento. Según Sun Microsystems (2000), esto permite conceder una garantía de acceso a través de un período de tiempo. Cada arrendamiento es negociado entre el usuario y el proveedor del servicio como parte del servicio de protocolo: un servicio es pedido por algún período; el acceso es concedido por algún período. Si un arrendamiento no es renovado antes que sea liberado (tal vez porque el cliente o la red falla o también porque recurso no es muy solicitado) entonces el usuario y el proveedor del recurso pueden concluir que el recurso puede ser liberado. Esta información es complementada en el apéndice B.

Eventos

La arquitectura JINI soporta eventos distribuidos. Un objeto puede permitirle a otro registrar eventos interesantes de entrada y recibir una notificación de la ocurrencia de un evento. Esto permite escribir programas distribuidos basados en eventos que ocurran en JVMs diferentes.

II.2.5 Segmentación de la arquitectura JINI

Como se mencionó anteriormente la tecnología JINI se segmenta en:

Infraestructura: según Sun Microsystems (2000), la infraestructura incluye los siguientes elementos:

- Un sistema de seguridad distribuido, integrado por RMI, que extiende el modelo de seguridad propuesto por la plataforma de JAVA para el mundo de los sistemas distribuidos.
- Los protocolos Discovery y Join, protocolos de servicio que permite a otros servicios (tanto de hardware como de software) descubrirse mutuamente, convirtiéndose como parte de un proveedor de anuncio de servicio a otros miembros de la federación.
- El servidor de búsqueda, el cual sirve como un almacén de servicios. Las entradas en el servicio de búsqueda son objetos escritos en Java; estos objetos pueden ser obtenidos como parte de una operación de búsqueda y actúan como un servidor local del servicio que coloca el código dentro del servidor de búsqueda.

Los protocolos "Discovery" y "Join" definen la forma la cual un servicio de cualquier tipo se convierte en parte de un sistema JINI; RMI define la base del lenguaje dentro del cual la tecnología JINI permite a los servicios comunicarse entre sí; el modelo de seguridad distribuida y su implementación define cómo las entidades están identificadas y cómo

obtienen los derechos para realizar acciones en su propio nombre y el de los demás. El servidor de búsqueda refleja a todos los miembros de la federación y actúa como un mercado central para ofrecer y encontrar servicios dirigidos a esos miembros de la federación.

Modelo de Programación: según Sun Microsystems (2000), la infraestructura hace uso del modelo de programación. Las entradas en el servicio de búsqueda son arrendamientos, permitiendo al servicio reflejar con precisión el conjunto de recursos disponibles actualmente. Los eventos permiten notificar a ciertos objetos registrados si un servicio se incorpora o abandona el servidor de búsqueda. Este modelo de programación se basa en la habilidad de mover código, el cual es soportado por la base de la infraestructura. Entre las interfaces que constituyen el modelo de programación se encuentran:

- La interfaz de arrendamiento, la cual define una vía de asignación y liberación de recursos regidos por un tiempo determinado de disponibilidad.
- Las interfaces de Eventos y notificaciones, que son una extensión de los modelos de eventos utilizados por los componentes JavaBeans para un ambiente distribuido, permitiendo una comunicación basada en eventos entre la tecnología JINI y los servicios.
- Las interfaces de transacciones, las cuales permiten que todos los cambios realizados por las entidades sean atómicos, es decir, se realizan todos los cambio o ninguno. Algunos ejemplos de interfaces y servicios son mencionados en el apéndice B.

Servicios

Como se mencionó anteriormente la infraestructura y el modelo de programación se construyen para permitir que los servicios sean ofrecidos y encontrados en un sistema distribuido. Por esta razón existen tres protocolos que permiten publicar y obtener servicios en una red denominados "Discovery", "Join" y "Lookup".

Según Sun Microsystems (2000), El corazón de un sistema JINI radica en este trío protocolos llamados "Discovery", "Join", y "Lookup". Un par de esos protocolos (Discovery Join) se activan cuando un dispositivo es instalado. El primero se activa cuando un servic busca un servidor de búsqueda con el cual registrarse. El protocolo "Join" se activa cuando u servicio es localizado por un servidor de búsqueda y desea unirse al conjunto de servicio "Lookup" ocurre cuando un cliente o usuario necesita localizar e invocar un servicio descrit por el tipo de interfaz (escrito en Java) y posiblemente otros atributos. La figura 7 ilustra e proceso "Discovery", la figura 8 muestra el proceso de "Join" y la figura 9 muestra el proces "Lookup".

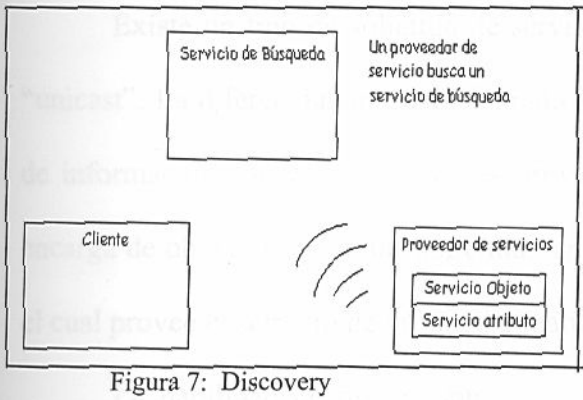


Figura 7: Discovery

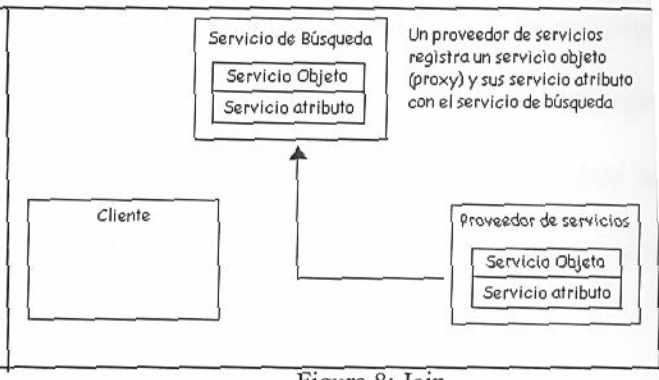


Figura 8: Join

Los protocolos "Discovery/Join" son el proceso de agregar un servicio a un sistema JINI. Un proveedor de servicio es el que lo origina (un dispositivo o software, por ejemplo). Principalmente, el proveedor de servicio localiza un servidor de búsqueda a partir de una solicitud "multicast" en la red local para cualquier servidor de búsqueda que lo identifique (Ver figura 7). Luego, un servicio objeto es cargado por el servidor de búsqueda (Ver figura 8). Este servicio objeto contiene una interfaz escrita en Java, incluyendo los métodos que el usuario y las aplicaciones invocarán para ejecutar el servicio.

Sun Microsystems (2000) indica que los servicios deben ser capaces de encontrar el servidor de búsqueda; sin embargo, un servicio puede delegar la tarea de encontrar un servicio

de búsqueda a otro servicio. El recurso está ahora listo para ser encontrado y usado, así com se muestra en la figura 9.

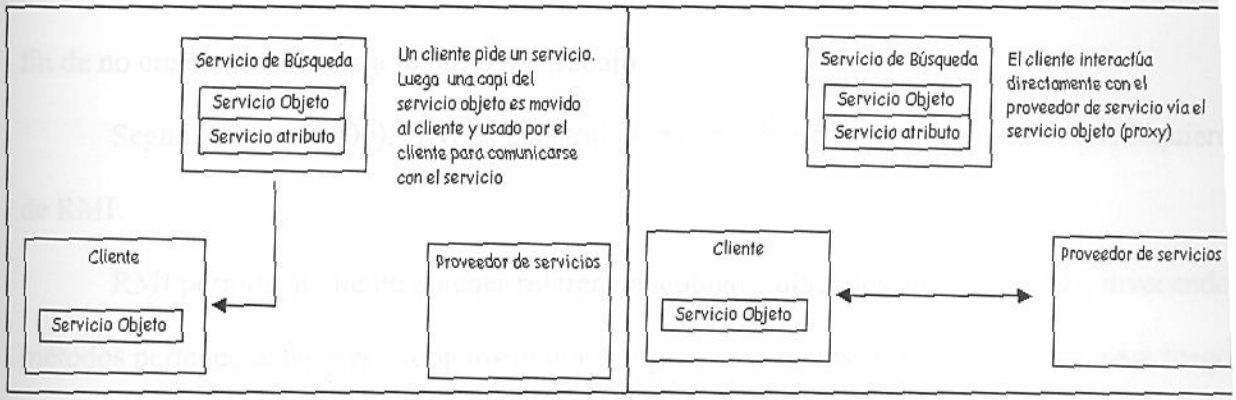


Figura 9: Lookup

Figura 10: El cliente usa el servicio

Existe un tipo de solicitud de servicios aparte de la "multicast" denominada solicitud "unicast". La diferencia entre ambas radica en que en la primera no es necesario ningún tipo de información sobre qué servidores proveen el servicio de búsqueda, simplemente ésta se encarga de obtenerlos. En una solicitud "unicast" es necesario conocer el nombre del servidor el cual provee el servicio de búsqueda, realizado comúnmente por RMI.

La habilidad de mover objetos y codificar del proveedor de servicio al servicio de búsqueda y de allí al cliente, se debe a que el proveedor de servicio da gran libertad en los modelos de comunicación entre el servicio y sus clientes (HTTP, FTP, entre otros).

Las interfaces pueden ser implementadas como una referencia RMI a un objeto remoto que implementa el servicio, así como un cómputo local que provee todos los servicios localmente, o combinación entre ambos elementos. Tales combinaciones, llamados proxies inteligentes, implementan algunas de las funciones de un servicio localmente y el resto a través de llamadas remotas para centralizar la implementación del servicio.

II.3 RMI y JINI

Una vez visto tanto JINI como RMI es necesario ver las diferencias entre ambos con el fin de no crear confusiones a lo largo del trabajo.

Según Artima (2000), JINI es construido encima de RMI. Por esta razón JINI requiere de RMI.

RMI permite al cliente obtener referencia a objetos ubicados en otras JVMs, invocando métodos pertenecientes a esos objetos remotos y pasando objetos como parámetros, para luego retornar un resultado o excepción. RMI también incluye un mecanismo que permite descargar cualquier código Java a través de la red entre JVM's diferentes. En cambio, JINI permite establecer una federación espontánea de clientes y servicios sobre una red.

Tanto JINI como RMI poseen un tipo de servicio de directorio. En el caso de JINI es llamado "Lookup Service" o servicio de búsqueda. En el caso de RMI es llamado "Registry" o registro. En RMI, cada objeto remoto es asociado a un registro según un nombre específico que debe ser único en toda la red. En cambio, los servicios de JINI son registrados a través de un identificador globalmente único y permanentemente mantenido por cada servicio, un objeto servicio, y cualquier número de atributos. En JINI, un cliente puede buscar un servicio por su identificador, por el tipo del servicio o por sus atributos. En RMI el cliente debe conocer el nombre con el cual el servicio es identificado para poder obtenerlo. JINI también ofrece otros mecanismos desconocidos por RMI, tal es el caso de un modelo arrendamiento que permite publicar el servicio por un tiempo determinado así como la ampliación de los eventos remotos propuestos por RMI.

En general, un programa cliente debe tener mayor conocimiento sobre la red en RMI, ya que este debe conocer donde se encuentra el servicio para poder usarlo (lo cual es menos

espontáneo). JINI posee un protocolo denominado "Discovery" que permite localizar servicios a través de la red sin necesidad de conocer donde están ubicados.

Otra gran discrepancia entre JINI y RMI es la diferencia entre el objeto servicio y el objeto stub. Aunque JINI utiliza RMI para enviar objetos de un lugar a otro, el objeto servicio en el cual el cliente invoca métodos necesita ser un RMI stub. El objeto servicio JINI podría ser un RMI stub, pero no es necesario. Este puede utilizar cualquier protocolo de red para comunicarse con cualquier servidor o dispositivo de hardware. Adicionalmente un objeto servicio JINI puede ser implementado completamente a nivel local lo cual no requeriría una comunicación en la red. Por el contrario, el objeto stub siempre usará el protocolo de comunicación RMI para comunicarse con el objeto remoto.

En conclusión, RMI se encarga de invocar métodos remotos en objetos ubicados en JVM's diferentes y permitir la transmisión de parámetros así como el retorno de valores y excepciones. JINI se encarga de crear una red espontánea de clientes y servicios que son capaces de encontrase por sí mismos en una red local.

II.4 Lego Mindstorm

Una vez conocido todos los mecanismos de red que son utilizados en el experimento es necesario conocer brevemente qué es Lego Mindstorm y cuales son sus componentes principales. Todos estos componentes son utilizados para construir el experimento.

II.4.1 ¿Qué es Lego Mindstorm?

Lego Mindstorm, también denominado Sistema de Invención de Robots, es un producto desarrollado por la compañía Lego en 1998. Este permite diseñar y programar robots reales, los cuales actúan de una manera específica. La historia de este producto es mencionada en el apéndice C.

II.4.2 ¿Qué es el RCX?

El componente principal es el RCX, el cual es el cerebro del sistema. Este puede programarse a partir de un computador, el cual se encarga de transmitir los programas al RCX a partir de una comunicación inalámbrica por medio de señales infrarrojo.

Según Dave Baum (2000) el RCX se puede dividir en varias capas:

La capa de Hardware

Es la capa inferior del dispositivo. Está compuesta por los siguientes mecanismos:

- Un microcontrol o CPU
- Una pantalla o LCD
- Memoria.
- Otros componentes electrónicos

La capa del sistema ROM

En adición al CPU y varios periféricos, el microcontrol también contiene un ROM (Memoria sólo para lectura), el cual viene programado con un lenguaje de bajo nivel. Este provee una interfaz con el hardware permitiendo mayor facilidad de manipulación con respecto al programador.

Según Dave Baum (2000) la característica más importante del sistema ROM es que permite manipular una pieza de software denominada Firmware que no es más que el sistema operativo del RCX. Este es transmitido al RCX vía infrarrojo y almacenado en la memoria.

Existen actualmente infinidades de Sistemas Operativos entre algunos de ellos están el firmware estándar de Lego, LegOS, pbForth, entre otros. El firmware en general es definido por Dave Baum (2000) como una expresión genérica para un software que es construido para el funcionamiento de un dispositivo y el cual permite funcionar como una interfaz entre un lenguaje de alto nivel y el Hardware.

II.4.3 Tipos de firmware

Actualmente existen varios firmware en el mercado. Entre ellos los mas importantes son: Firmware estándar, LegOS, PbForfth entre otros. A continuación se da una breve explicación sobre el Firmware estándar que es el utilizado en este trabajo de investigación y los demás son explicados brevemente en el apéndice C.

Firmware Estándar

Este es usado cuando se escriben programas usando cualquiera de los ambientes compatibles con Lego (código RCX, NQC, Robolab o Spirit.ocx) En la figura 11 se muestra como el código RCX y los programas NQC son ejecutados dentro del RCX.

Según Baum (2000), existen herramientas especiales llamadas compiladores (en el caso de NQC) los cuales están ubicados en el computador que está conectado al RCX. Estos compiladores se encargan de traducir el lenguaje usuario a unos códigos especiales entendidos por el RCX.

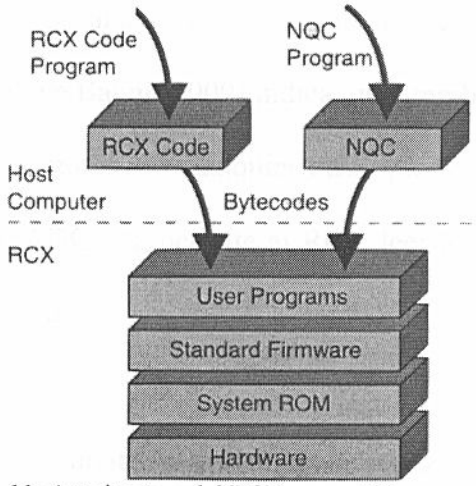


Figura 11: Arquitectura del RCX para el Firmware estándar

Una vez obtenidos estos códigos son transmitidos al RCX y almacenados como programas de usuario. "El CPU del RCX no puede ejecutar estos códigos directamente, por esto el firmware estándar de Lego se encarga de interpretar estos códigos" Dave Baum (2000).

Existen actualmente en el mercado varias versiones de este firmware. Recientemente Lego lanzó al mercado una nueva versión que es compatible con la anterior y ofrece un mayor número de funcionalidades. Este es conocido como RCX 2.0 firmware.

II.4.4 Hardware

Según Dave Baum (2000), el RCX está basado en un procesador Hitachi H8. Este posee un CPU de 8 bits y brinda mucho más que el control lógico del RCX. También incluye Entradas/Salidas por puerto serial, un adaptador o ADC (convertidor de corriente analógica a digital) y relojes integrados.

El RCX también contiene 16KB de espacio en memoria ROM y 32KB de memoria estática RAM. Esta memoria es utilizada para almacenar el firmware, los programas de usuarios y cualquier dato requerido para esos programas.

El RCX posee unos componentes especiales que funcionan como una interfaz con el usuario. Entre ello están el LCD o pantalla el cual muestra el estado del RCX, botones de control y una pequeña corneta. Dave Baum (2000) indica que también existe un chip especial que permite al RCX controlar los motores o cualquier otro tipo de dispositivo eléctrico unido al puerto de salida así como un ADC que permite al RCX leer los tres puertos de entrada y chequear la condición de las baterías.

Luz Infrarrojo

La comunicación entre el computador y el RCX se realiza vía señales infrarrojo. Para lograr esta comunicación la torre infrarrojo debe ser conectada al puerto serial del computador y el RCX debe estar accesible al espectro de señales emitidas por ambos dispositivos Este es complementado en el apéndice C.

Puertos de Salida

El RCX posee tres puertos de salida, el cual según Baum (2000) pueden ser operados en uno de tres modos: prendido, apagado o flotando. El modo prendido se utiliza para encender el motor. El modo apagado como la palabra lo indica sirve para detener cualquier motor conectado a ese puerto. En cambio el modo flotando, permite detener el suministro de energía al motor, pero este no es forzado a detenerse.

Otra característica de los puertos de salida es que poseen la capacidad de cambiar la dirección hacia adelante o hacia atrás así como graduar el poder o velocidad de cada uno de los motores conectados a estos puertos.

Puertos de Entrada

El RCX tiene tres puertos de entrada, cada uno de ellos puede manipular uno de los cuatro sensores de Lego (sensor de choque, de luz, de rotación y de temperatura). Estos sensores son conectados al RCX usando el mismo tipo de escritura que el utilizado para conectar los motores al puerto de salida.

Según Dave Baum (2000) existen dos clases de sensores: los pasivos y los activos. Los sensores pasivos operan como una simple resistencia y no requieren de energía eléctrica por parte del RCX para funcionar. Un ejemplo de esta clase son los sensores de choque y de temperatura. Mientras los sensores activos son aquellos que requieren energía y son mas sofisticados (sensor de luz y de rotación).

Dave Baum (2000) indica que el firmware estándar y pbForth adicionan dos características importantes a la hora de trabajar con sensores, entre ellas están el modo y el tipo de sensor. El tipo determina cómo el RCX interactúa con el sensor mientras que el modo indica cómo interpretar el valor obtenido por ese sensor. Estos son discutidos más ampliamente en el apéndice C.

II.4.5 Programación del RCX

Como se mencionó anteriormente la programación del RCX es realizada a través de un computador, para luego ser transmitida por medio de señales infrarrojo al dispositivo (ver figura 5 del apéndice C). Existen diferentes tipos de programación para cada tipo de firmware. Para este proyecto fue utilizado el firmware estándar por razones descritas a lo largo de este documento.

Según Dave Baum (2000), el firmware estándar ofrece una capacidad de memoria aproximadamente de 6KB para almacenar un programa de usuario.

Existen varias formas de programar en este sistema operativo. Las principales son: por NQC y por código RCX (utilizado por la mayoría de las aplicaciones de Lego).

NQC proviene de las siglas "Not Quite C", es un lenguaje de alto nivel desarrollado por Dave Baum, el cual posee un compilador que se encarga de convertir el lenguaje usuario a código RCX para luego ser transmitido al dispositivo. Existen otras herramientas que ofrecen mayores facilidades al momento de programar tal es el caso de la aplicación ofrecida por Lego Mindstorm el cual ofrece interfaces gráficas que transforma el programa en código RCX para luego ser transmitidas al dispositivo. Aunque ofrece una facilidad en la programación de código RCX, este posee grandes limitaciones al momento de realizar programas avanzados.

Aunque el uso de estas herramientas (NQC, Software de Lego) facilitan la programación del RCX, estos no permiten realizar tareas avanzadas como es el de controlar el RCX a tiempo real. Para esto es necesario transmitir código RCX puro a partir del computador permitiendo así manipular al dispositivo en un momento dado. De esta manera, Kekoa Proudfoot descifró los códigos necesarios para controlar el RCX desde el computador usando el firmware estándar. En el anexo 1 se muestran algunos de los comandos con su respectivos códigos que permiten ser interpretados por el RCX para su funcionamiento.

Para poder enviar estos comandos es necesario instalar algún programa que permita manipular los puertos seriales del computador y que tenga la capacidad de transmitir estos comandos a partir del protocolo de comunicación del RCX. Actualmente en Internet existen varios programas capaces de establecer esta comunicación, entre ellos "`send.c`" elaborado por Kekoa Proudfoot (http://graphics.stanford.edu/~kekoa/rcx/tools.html) y "`rcx.jar`" elaborado por Dario Laverde (http://www.escape.com/~dario/java/rcx).

III Marco Metodológico

La metodología utilizada en el desarrollo de este trabajo de Investigación se basa principalmente en el Proceso Unificado con ciertas variantes debido a características propias del experimento realizado en este trabajo de grado.

El proceso de desarrollo de un software es un conjunto de actividades necesarias para transformar los requerimientos de usuarios en un sistema. Según Booch, Jacobson y Rumbaugh (1999) el proceso unificado es más que eso, éste es un proceso genérico que puede ser utilizado para una amplia variedad de clases de software, diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes longitudes de proyectos utilizando como forma de modelaje UML (Unified Modeling Language).

Según Booch, Jacobson y Rumbaugh (1999) el Proceso Unificado se basa en tres premisas principales para el desarrollo de un software:

- ***Manejo de los casos de usos:*** para construir un sistema exitoso es necesario conocer las necesidades del usuario. Los casos de usos son una pieza de funcionalidad en el sistema que le brinda al usuario un resultado de valor. Este se encarga de capturar los requerimientos funcionales del sistema. Todos estos casos de uso son reunidos en el modelo de casos de usos el cual describe la completa funcionalidad del sistema.
- ***Orientado a una arquitectura:*** permite describir diferentes puntos de vista del sistema antes de construirlo. Esta arquitectura depende de muchos factores tal como la plataforma de Hardware y de Software, recursos, resistencia a cambios futuros por parte del usuario, tiempo disponible para el desarrollo, etc.
- ***Iterativo e incremental***: Para el desarrollo de software de gran tamaño, es necesario dividir el trabajo en pequeñas piezas o mini proyectos, esto con el fin de obtener un mayor control

y en consecuencia mayor éxito en las metas fijadas. Las iteraciones se refieren a las etapas del flujo de trabajo, mientras que incremental se refiere al crecimiento del producto.

Según Booch, Jacobson y Rumbaugh (1999) la base del proceso Unificado se centra en una serie de ciclos que le dan vida al producto final, y cada uno de estos ciclos culminan con un resultado de valor. Cada ciclo se divide en cuatro fases: principio, elaboración, construcción y transición tal como se describe en la figura 12.

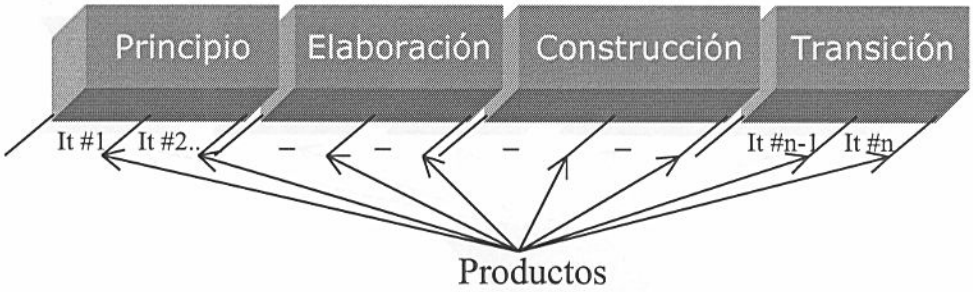


Figura 12: Un ciclo con sus fases e iteraciones

En la fase ***Principio*** se establecen los alcances de lo que debe ser el producto final, reducir los riesgos, y preparar una propuesta inicial del negocio. En resumen se establece el ciclo de vida de los objetivos para el proyecto.

En la fase de ***Elaboración*** se conforma la arquitectura, se capturan la mayoría de los requerimientos así como se reducen la mayor cantidad de riesgos posibles. En esta fase es posible estimar los costos y el plan de trabajo.

En la fase de ***construcción*** se desarrolla el sistema completo y se asegura que éste pueda pasar a la fase de transición.

Por último, la fase de ***transición*** se encarga de asegurar que el producto esté listo para ser utilizado por el o los usuarios. Esta fase incluye las pruebas del sistema y entrenamiento de los usuarios.

Booch, Jacobson y Rumbaugh (1999) indican que cada fase puede tener varias iteraciones. Una iteración es un mini proyecto, un flujo de trabajo conformado por las siguientes actividades: requerimientos, análisis, diseño, implementación y pruebas tal como se muestra en la figura 13. Lo que diferencia esto con el modelo de cascada es que existen varias iteraciones que pueden solaparse entre sí obteniendo un mayor aprovechamiento del tiempo y los recursos. En la figura 14 se muestra como interactúa cada una de las iteraciones.

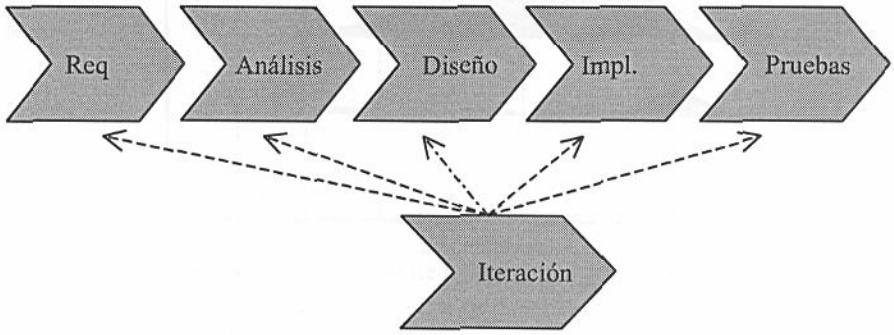


Figura 13: cada iteración puede utilizar cada uno de los cinco principales flujos de trabajo

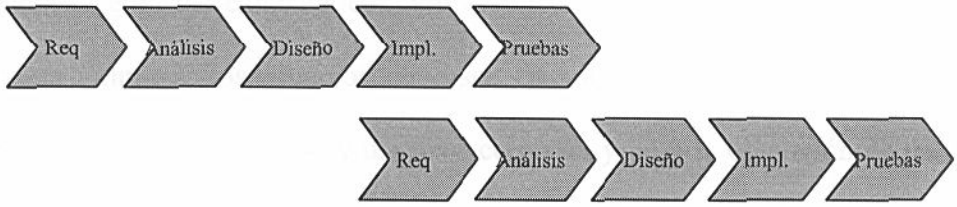


Figura 14: Las iteraciones se extienden a través de los flujos de trabajos solapándose entre sí

Cada iteración se extiende a través de cada flujo de trabajo desde los requerimientos hasta las pruebas, pero cada iteración tiene diferente énfasis en cada una de las fases. Como se puede observar en la figura 14, existe un solapamiento ya que se está realizando tanto la implementación como el análisis de requerimientos en iteraciones diferentes.

Como se muestra en la figura 15 durante la fase de principio y elaboración el mayor esfuerzo se dedica a capturar los requerimientos y análisis preliminar mientras que en las otras fases se realizan en mayor detalle el diseño, implementación y prueba del sistema.

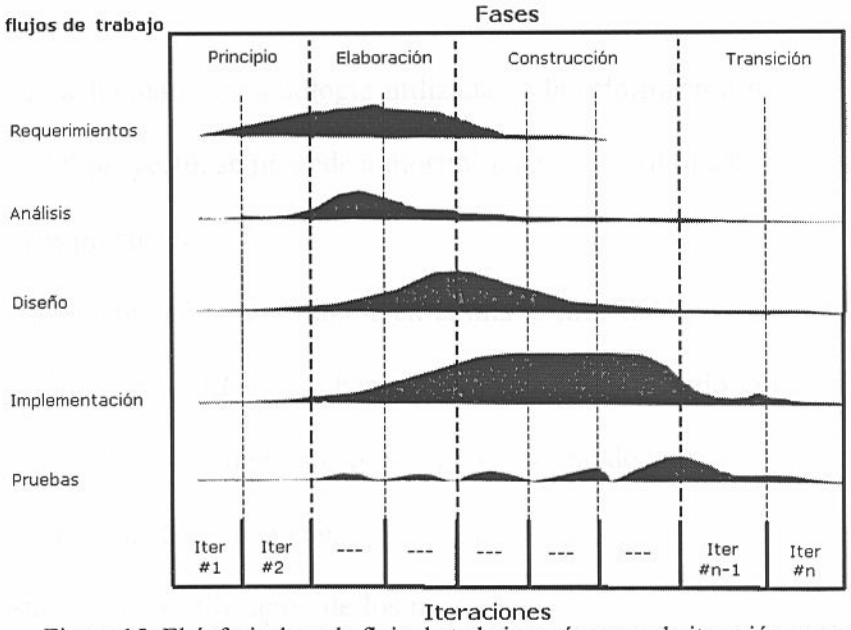


Figura 15: El énfasis de cada flujo de trabajo varía por cada iteración en cada fase

Como se mencionó anteriormente, la aplicación de esta metodología sufrió ciertas variantes. Dado el énfasis realizado en la investigación, mas no en la implantación del producto en una empresa, los modelos de casos de uso fueron descartados por ser estos principalmente enfocados al negocio y usuarios como tal y no a la obtención de un producto final.

IV Desarrollo

Una vez aclarada la metodología utilizada y la información teórica necesaria para el entendimiento del proyecto, se procede a mostrar todas las actividades realizadas para cumplir con los objetivos planteados.

Como se mencionó anteriormente en el marco metodológico, la metodología utilizada en la elaboración de este trabajo se basó en el Proceso Unificado con ciertas variantes. La variante más importante es que este proyecto va orientado a la investigación más no a la implantación del producto en una empresa u organización.

Por esta razón, la utilización de los modelos de los casos de uso fue descartada por no agregar información importante en la obtención de los resultados finales del trabajo.

Asimismo, las razones más importantes por las que se tomó esta metodología para la elaboración del trabajo de investigación fueron las siguientes:

- Iteraciones, que permitían separar un proyecto en mini-proyectos ofreciendo una mayor comprensión del problema.
- Una metodología más flexible, por permitir solapamiento entre las iteraciones ofreciendo así un mayor aprovechamiento de los recursos (como el tiempo).
- Por ser una metodología orientada al paradigma de objetos. Esto se debe a que la programación fue desarrollada en el lenguaje de programación Java.
- Por ofrecer mecanismos de modelaje conocidos, facilitando los procesos de elaboración y comprensión de los diagramas utilizados.

A grosso modo, el proyecto fue realizado a partir de un sólo ciclo que contiene las fases de: principio, elaboración, construcción y transición; cada una de estas fases están

complementadas con un número determinado de iteraciones. La tabla 1 muestra cada una de las fases del proyecto con su respectivo número de iteraciones.

Fase	# de iteraciones
Principio	1
Elaboración	2
Construcción	4
Transición	1

Tabla 1: Iteraciones por cada fase del proyecto

Como se puede observar en la tabla 1, hay un total de 8 iteraciones a lo largo del proyecto.

En la tabla 2 se muestran las actividades realizadas en cada una de las iteraciones de dichas fases.

Fase	Iteración	Actividad
Principio	1	Definición de los requerimientos necesarios para desarrollar el trabajo de investigación
Elaboración	1	Investigación y diseño de los componentes necesarios para la manipulación el robot localmente.
	2	Investigación y diseño de los componentes de la tecnología JINI necesarios para poder manipular remotamente la actuación del robot
Construcción	1	Desarrollo y configuración de los componentes de Java necesarios para la manipulación del robot localmente.
	2	Desarrollo y prueba de una aplicación que sirva como interfaz entre el operador y el robot.
	3	Elaboración de un componente JINI que permita obtener un objeto remoto.
	4	Adaptación de componentes desarrollados en las iteraciones 1,2 y 3
Transición	1	Elaboración del experimento y ajustes de la aplicación

Tabla 2: Actividades realizadas en cada fase del proyecto

Como se puede observar en la figura 16, se muestra una aproximación de cada una de las actividades básicas de un proyecto realizadas en cada una de las iteraciones de cada fase. En el caso de la fase de principio se hizo énfasis en el análisis y estudio de los requerimientos.

También se puede observar que en la fase de elaboración se realizan en mayor grado actividades de análisis y diseño de los componentes. Por último, las fases de construcción y transición se limitan al desarrollo y pruebas de estos componentes con pequeños rasgos de diseño en cada una de las iteraciones.

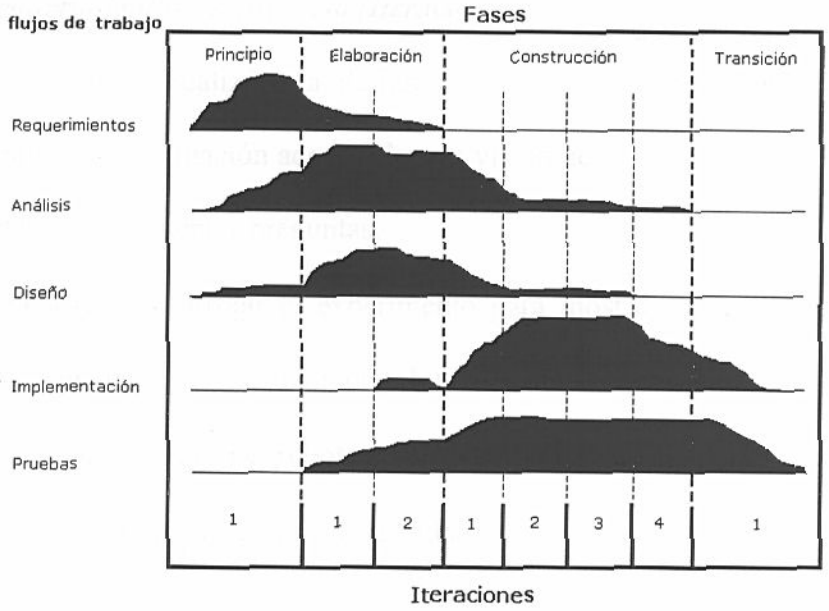


Figura 16: actividades realizadas en cada una de las fases con sus respectivas iteraciones

En los párrafos siguientes se explica detalladamente cada una de las fases con las respectivas actividades descritas en la tabla 2.

IV.1 Fase de Principio

Por ser un trabajo de investigación, esta fase se basa principalmente en un análisis tentativo de lo que se quiere obtener al final del proyecto, es decir, al producto final. Por esta razón se desecha todo tipo de análisis de requerimientos de un cliente en específico pero si se realizan reuniones con personal especializado en el área, que permite guiar el experimento a las necesidades reales en el área de robótica y redes.

Como se mencionó anteriormente esta fase posee una sola iteración, y en esta se realiza una investigación de los elementos necesarios para el desarrollo de una aplicación que interactúe con un robot a través de la tecnología JINI.

Análisis de los requerimientos del proyecto (Iteración 1)

En esta iteración se realizó una definición de los requerimientos necesarios para desarrollar el trabajo de investigación adoptando una visión general del problema. Para esto se buscaron respuestas a las siguientes preguntas:

- ¿Qué elementos debe arrojar el experimento para mostrar la funcionalidad de JINI como herramienta de programación distribuida sobre el Robot?
- ¿ Qué función debe cumplir el robot y de qué manera ?
- ¿ Qué debe mostrar la aplicación a desarrollar ?
- ¿ Quiénes son los usuarios?
- ¿ Qué recursos son necesarios para desarrollar el experimento?
- ¿ En cuanto tiempo se debe realizar ?

Como primer paso se establecieron varias reuniones con diferentes personas especializadas en el área de Robótica (Prof. Carolina Chang) y Redes (Ing. Elie Levy y prof. Wilmer Pereira) entre otros, quienes aportaron ideas para el establecimiento de uno o varios experimentos que permitieran mostrar las funcionalidades de la tecnología JINI.

También se realizaron ciertas entrevistas a distancia de personas conocedoras de la tecnología JINI y Lego Mindstorm, como es el caso de la Ing. Iain Shigeoka miembro de la lista de distribución de Sun Microsystems para usuarios de dicha tecnología y Kekoa Proudfoot gran conocedor de la familia de sistemas de invención de robot de lego Mindstorm .

Esto nos permitió conocer la factibilidad del experimento así como la opinión de cada uno de estas personas frente al proyecto.

A partir del aporte de estas personas, se realizó un análisis que permitiera conocer los recursos necesarios para la realización del experimento así como la debida planificación del proyecto.

Esta etapa también permitió desarrollar el plan de pruebas, el cual es implementado a lo largo del proyecto para obtener un mayor control de los errores. Este plan es mostrado en el apéndice F.

IV.2 Fase de Elaboración

Como se mencionó anteriormente, esta fase posee 2 iteraciones en las cuales se tratan dos elementos por separado. El primero, todo lo concerniente con el control local del robot y segundo todo lo que tiene que ver con la parte de red enfocado a los resultados obtenidos en la primera etapa.

Como se puede ver en la figura 16 en esta etapa predominan dos aspectos principales: un análisis más detallado del problema y el diseño de una arquitectura que permita resolver el problema.

También es importante mencionar que ambas iteraciones sufrieron un leve solapamiento el cual permitió un trabajo en paralelo. Este solapamiento comienza un poco antes de finalizar la primera iteración de esta fase. A continuación se describe cada una de estas etapas:

Investigación y diseño de los componentes locales (Iteración 1)

En esta iteración el trabajo se enfoca en investigar y diseñar los componentes necesarios para la manipulación el robot localmente.

En primer lugar se buscó información en Internet y bibliografía acerca de los diferentes modos de programación del RCX (ver marco teórico sección "programación del RCX") para luego escoger la opción más calificada según los requerimientos del experimento y la que más adaptase para cumplir los objetivos planteados.

Una vez obtenida una visión más clara del funcionamiento de Lego Mindstorm, se procedió a un análisis exhaustivo para determinar las características mínimas necesarias para el funcionamiento del robot, tal es el caso del firmware, número de motores, tipo de sensores, un diseño aproximado del robot y otros elementos necesarios para la construcción del robot. También se estudió como debe ser el ambiente donde este debe actuar como el tipo de superficie, el espacio y otros.

Una vez obtenida una posible estructura del robot, se realizó un diseño de la arquitectura necesaria para manipular el robot localmente, no olvidando que luego esta debe ser capaz de interconectarse con otros componentes para su funcionamiento en Red.

Al obtener este modelo de arquitectura comenzó la ejecución en paralelo de ésta con la segunda iteración.

Esta etapa también incluye la investigación y obtención de las herramientas necesarias para la manipulación del RCX desde el computador, dependiendo de los resultados obtenidos en la fase anterior.

Para finalizar, se procedió a establecer un diseño tentativo de las clases necesarias para establecer la comunicación del computador con el robot así como su respectivo diagrama de clases.

Investigación y diseño de los componentes remotos (Iteración 2)

En esta iteración se investigaron y diseñaron los componentes de la tecnología JINI necesarios para poder manipular remotamente la actuación del robot.

En primer lugar, se realizó un análisis más detallado y técnico de la tecnología. En el marco teórico de este informe se muestra parte de la información obtenida en esta etapa. También se observaron ejemplos prácticos obtenidos de Internet (http://www.jini.org) y posibles proyectos realizados por Jan Newmarch (2001) los cuales se había trabajado conjuntamente con Lego Míndstorm .

A partir de estos ejemplos se realizó una configuración general del ambiente distribuido basado en la ejecución del servidor de búsqueda así como corrida de ejemplos para así asegurar el dominio práctico de JINI. En el apéndice E se encuentra información de cómo correr estos ejemplos.

Esta etapa también incluye el diseño de la interfaz utilizada por cada uno de los usuarios para controlar el robot y observar la composición del terreno explorado.

Finalizando esta etapa, se realizó un diseño de todas las clases necesarias para desarrollar el cliente y el servidor que permitiese la comunicación a través de la red. Este diseño incluye un diagrama de clases tanto del cliente y el servidor como de la aplicación que sirve de interfaz entre el usuario y el componente JINI. También se diseñó una arquitectura posible para la manipulación del robot a partir de un cliente remoto.

IV.3 Fase de Construcción

Esta fase se conforma principalmente de tres procesos esenciales. El primero es el Desarrollo o programación del código fuente, basado en el diseño obtenido en la fase anterior así como la configuración de todos los elementos necesarios para el buen funcionamiento de los componentes. En segundo lugar, las pruebas unitarias de cada componente realizado de manera de poder encontrar errores en la programación y diseño de los mismos. En tercer y último lugar el rediseño de todos los componentes que así lo ameritase.

Como se muestra en la tabla 1 y 2, esta fase posee cuatro iteraciones que se describen a continuación:

Desarrollo y prueba de los componentes locales (Iteración 1)

Esta etapa se basó en el desarrollo y configuración de los componentes de Java necesarios para la manipulación del robot localmente.

Pero antes de construir los componentes locales era necesario construir y configurar un robot que permitiera ser controlado según las especificaciones obtenidas en la fase de elaboración y que son mencionadas en el capítulo de resultados. Las actividades realizadas en esta primera etapa fueron:

- Instalación del firmware.
- Construcción y prueba de diferentes robots tomando en cuenta los resultados obtenidos en la etapa de elaboración.
- Construcción del ambiente en la cual el robot se desenvuelve.
- Desarrollo de los programas de usuario del robot que permitieran su manipulación.

La primera actividad fue la instalación del firmware al RCX. Este permitiría programar las funciones del robot. Este firmware fue escogido en la fase de elaboración según las especificaciones del experimento.

Una vez instalado el sistema operativo del RCX, se procedió a construir varios prototipos del robot. Cada robot era probado de la siguiente manera: eran descargados tres programas, el primero desplazaba el robot hacia adelante hasta que este chocara con un obstáculo, el segundo y el tercero giraba el robot 90° hacia la derecha e izquierda respectivamente.

Una vez almacenado los programas en el RCX, eran corridos en cualquier terreno y se evaluaba el desempeño del robot. De esta manera eran reconstruidos para mejorar la precisión del dispositivo o simplemente optar por otro diseño que disminuya las limitaciones del robot.

También se realizaron pruebas en el que se evaluaba la precisión obtenida al usar sensores de rotación o simplemente ejecutar tareas en un tiempo determinado (tal es el caso de desplazarse por tres segundos). Esto permitiría comprobar cuán necesario es la utilización de los sensores de rotación para lograr un desplazamiento óptimo del robot.

Una vez obtenido el robot más capacitado para cumplir con los objetivos del experimento, se hicieron pruebas de navegación del robot en diferentes superficies. Entre ellas: granito, cemento, cartulina y fórmica. Esto con el fin de evaluar el desempeño del robot en cada una de estas y así escoger la alternativa más apta y más fácil de usar. En esta etapa también se planteó la forma cómo el robot debe desplazarse en el terreno en búsqueda de obstáculos. También se definió la delimitación y composición del terreno tal es el caso como la distancia entre cuadros, entre otros.

Una vez finalizadas estas actividades, se procedió a la programación del RCX. Este paso es muy importante ya que aquí se da la funcionalidad al robot, lo cual es acompañado paralelamente con reiteradas pruebas de cada uno de los programas por separado para lograr un alto nivel de precisión y evaluar el desempeño del robot.

Las pruebas permitieron medir la capacidad de giro y desplazamiento preciso del robot así como evaluar el funcionamiento de los sensores utilizados por el robot.

Una vez lograda la precisión deseada, se procedió a instalar los paquetes necesarios para poder controlar el robot desde cualquier aplicación Java. Esto incluye los paquetes javax.comm distribuidos por Sun Microsystems y rcx.jar distribuido por Dario Laverde. El primero permite controlar los puertos seriales del computador independientemente de la

plataforma y el segundo es un paquete realizado en Java que se encarga de enviarle comandos en tiempo real al RCX.

Por último, una vez obtenido el perfecto funcionamiento de los paquetes ya instalados en el computador, se programó un ejemplo en el cual se controla el robot ya construido a partir de un programa Java.

Desarrollo y prueba de la aplicación (Iteración 2)

Como se menciona en la tabla 2, en esta iteración se procedió a desarrollar una aplicación que sirviese como interfaz entre el operador y el robot. Para la elaboración de esta aplicación se toma en cuenta los resultados obtenidos tanto en la fase de principio como en la fase de elaboración. Esta aplicación es orientada a controlar el robot localmente, para así realizar pruebas que permitan mostrar tanto el desenvolvimiento del robot como la funcionalidad de la aplicación en sí.

Una vez construida esta aplicación, fue adaptada a los paquetes obtenidos en la iteración anterior con el fin de lograr una manipulación del robot a partir de la aplicación. Esta manipulación aunque es de forma local nos permitió evaluar el desempeño del robot luego de reiteradas exploraciones de un área del terreno.

Desarrollo y prueba del componente remoto (Iteración 3)

Esta etapa comienza poco antes de la finalización de la iteración dos, con lo cual se produce un solapamiento entre etapas. En esta se crea un componente JINI que permita obtener un objeto remoto. El comportamiento de este objeto remoto debe cumplir las características propuestas en la etapa de elaboración con respecto a la estructura del componente.

En este periodo se programaron los componentes necesarios para comunicar los clientes con el servidor en donde se ubica el servicio. Para esto son utilizados los conocimientos prácticos obtenidos en la iteración 2 de la fase de elaboración. En esta etapa se realizaron las siguientes actividades:

- Instalación de un servidor de búsqueda que permita publicar y ofrecer servicios en una red.
- Instalación de un servidor web que permita transmitir archivos de un lugar a otro. Para el experimento, este servidor es necesario ya que el proveedor de servicio transmite el objeto "stub" al cliente mediante el protocolo HTTP (Para mayor información ver capítulo de resultado)
- Un proveedor capaz de publicar un servicio en cualquier servidor de búsqueda ubicado en la red así como brindar el servicio apoyándose en el uso de notificaciones remotas.
- Un cliente capaz de buscar el servicio en cualquier parte de una red para luego obtenerlo y usarlo según especificaciones del servicio.

Una vez obtenidos estos programas, se incluyeron pruebas unitarias que permiten verificar la consistencia del programa realizado según especificaciones del plan de pruebas obtenido en la fase de principio (apéndice F).

Adaptación y prueba de componentes (Iteración 4)

Una vez culminado cada componente por separado (la interfaz, manipulación del robot localmente, y el componente JINI), se culmina la fase de construcción conectando cada uno de estos con el fin de lograr una manipulación remota entre la aplicación y el robot a partir del componente JINI.

Esta adaptación produjo pequeñas modificaciones de cada componente para lograr la unificación de las mismas en dos paquetes. Un paquete que contiene todas las funciones referentes al servidor quien debe controlar localmente el robot y un segundo paquete ubicado en el lado del cliente que permita, a partir de una aplicación, comunicarse con el servidor para manipular el robot.

En esta etapa también se hicieron pequeñas modificaciones a la aplicación para permitir que un sólo usuario (operador) controle el robot en un instante determinado. Cualquier otro usuario que intenta obtener el recurso es colocado en una cola en espera del recurso. Aunque estos usuarios no pueden controlar el robot, si pueden observar la exploración realizada por el operador. Una vez que el operador cierra la aplicación, el recurso es asignado al usuario que lleva más tiempo en espera.

En esta iteración también se realizaron pruebas pero estas son pruebas de integración en las cuales se evalúa el comportamiento general del sistema. Esto nos permite conocer ciertos problemas como qué sucede si se cae la red, si el servidor o cliente no encuentra el servicio de búsqueda, si el servicio no está disponible, si un operador o usuario pierde conexión entre otros.

IV.4 Fase de Transición

En esta fase se realizó el montaje del experimento con sus respectivas pruebas así como pequeñas mejoras en el sistema. Esto incluye reuniones con diferentes personas como ingenieros y especialistas con el fin de obtener una retroalimentación con respecto a los resultados obtenidos en las fases anteriores.

En esta etapa también se solucionaron conflictos obtenidos en la fase de construcción tal es el caso en que un operador o usuario pierda conexión con la red. Esta solución es explicada ampliamente en la sección de resultados.

En esta fase también se realizaron pruebas en diferentes plataformas con el fin de comprobar el concepto de multiplataforma de la tecnología JINI. Los siguientes puntos indican las plataformas en las cuales el experimento fue probado:

- Solaris Sparc 8.
- Windows 98.

También se construyeron los obstáculos necesarios para la conformación del experimento así como un soporte especial que facilitaría la comunicación entre el RCX y la torre infrarrojo. Este soporte permite fijar una distancia vertical constante entre ambos elementos disminuyendo los riesgos de falla de comunicación. Adicionalmente se realizaron pruebas acerca del grado de luminosidad que debe tener el ambiente con el fin de que interfiera lo menos posible con la actuación del robot.

V Resultados

En esta sección del informe se muestran los resultados obtenidos a partir de las actividades descritas en el capítulo anterior. Estos resultados son explicados siguiendo las fases e iteraciones del proyecto.

V.1 Fase de Principio

En esta fase representada por una sola iteración se obtuvieron los requerimientos del sistema en una visión generalizada. Esto se logró obteniendo respuesta a las preguntas mencionadas en el capítulo de desarrollo. A continuación se explican cada una de ellas:

El producto final del experimento debe ser el establecimiento de una comunicación entre la aplicación basada en un componente JINI y el robot, de esta manera se podría obtener una representación del manejo de dispositivos a partir de la tecnología JINI. Este tiene que brindar un control sobre el robot proporcionado por la aplicación desde cualquier parte de una red.

Contestando la segunda pregunta, la función del robot también debe determinarse de acuerdo a una constante comunicación entre el robot y la aplicación. No es necesario que el robot tome decisiones por si mismo o sea independiente (aunque esto puede ser tratado en trabajos posteriores), sino que exista como se mencionó anteriormente una comunicación entre el cliente y el robot. Por eso fue escogida como función principal, que este dispositivo deba explorar una zona específica y vaya mostrando los resultados al cliente.

Con respecto a la aplicación, esta debe ser capaz de controlar o visualizar las acciones del robot remotamente de una manera eficiente. Por esta simple razón, la aplicación debe soportar

un número considerado de clientes accediendo al mismo tiempo al sistema y adicionando aspectos de interfaz gráfica que permitan facilitar las funciones de aprendizaje al usuario.

Los usuarios pueden ser cualquier persona que tenga acceso a la red. Para lograr esto la aplicación basada en el componente JINI debe ser capaz de administrar los accesos de estos clientes y lograr que de alguna forma todos puedan controlar el robot. Asimismo, no deberían existir ningún tipo de restricciones al momento de utilizar la aplicación, todo usuario ubicado en la red local pude controlar el robot.

Los recursos mínimos necesarios para la ejecución del proyecto son los siguientes:

1. Computador que funcione como servidor y cliente con interfaz de red.
2. Un sistema de invención de robots de Lego Mindstorm.
3. Un sensor de choque y dos sensores de rotación.
4. Dos motores de Lego.

En esta fase también se obtuvo información acerca de las limitaciones del proyecto así como la necesidad de adquirir sensores adicionales específicamente de rotación ya que Lego Mindstorm no brinda este tipo de sensores al adquirir el producto. Estos son necesarios para lograr una mayor precisión en el desenvolvimiento del robot.

En esta etapa se adquirieron algunos rasgos de la funcionalidad del sistema. A continuación se muestran las conclusiones obtenidas con respecto a la funcionalidad :

- El robot debe desplazarse por el terreno en búsqueda de posibles obstáculos. Este será teledirigido desde un lugar remoto dentro de una red local. El robot se encargará de reconocer los obstáculos a partir del choque con los mismos utilizando un sensor de choque.
- El terreno debe ser uniforme, preferiblemente liso, sin la existencia de hoyos ni desniveles que puedan ofrecer resistencia al momento de realizarse la exploración

- La aplicación debe manipular el robot, así como mostrarle al cliente la representación del terreno obtenido por el robot. También debe mostrar ciertas estadísticas sobre la composición del terreno. Esta exploración puede inicializarse en cualquier momento.
- Para cumplir con los requerimientos del experimento, solo un usuario puede controlar el robot en un momento dado(exclusión mutua), de esta forma estamos asegurando la consistencia de la aplicación. Si todos los usuarios pudieran manipular el robot al mismo tiempo esto ofrecería inconformidad y rivalidad entre ellos. Otra solución (aunque no fue tomada en cuenta) era la utilización por intervalos de tiempo, permitiendo que cada usuario posea un quantum de tiempo para controlar el robot. En la realidad esto no es muy conveniente, ya que un usuario puede no culminar la utilización del recurso en el plazo de tiempo estipulado produciendo pérdida de información e inconformidad con el sistema.
- Una vez obtenido el número de usuarios que pueden controlar el dispositivos en un momento dado, se concluyó que aunque los demás usuarios no podrían utilizar el recurso en ese momento, si podrían observar la composición del terreno y la posición del robot. De esta manera los demás usuarios pueden observar la exploración realizada por el operador desde cualquier parte de la red, obteniendo datos actualizados.
- Al momento en que el operador libera el recurso, este debe ser reasignado a uno de los usuarios que así lo requieran. Este recurso es asignado preferiblemente al usuario que lleva mayor tiempo esperando en el sistema.

Antes de finalizar esta etapa se obtuvo un plan de pruebas que permite planificar el modo de ejecución de las pruebas a lo largo del proyecto. Este plan se encuentra especificado en el apéndice F.

V.2 Fase de Elaboración

Como se puede observar en la figura 16 (Pág. 35), esta etapa se divide en dos iteraciones, y como se menciona en el capítulo de desarrollo, se enfoca principalmente a un análisis más detallado del problema, acompañado de su respectivo diseño. A continuación se muestran los resultados obtenidos en cada iteración:

Investigación y diseño de los componentes locales (Iteración 1)

El análisis acerca de las diferentes formas de programación del RCX, arrojó distintas formas para el control del dispositivo desde cualquier computador. para esto se pueden utilizar distintos firmware's mencionados en el marco teórico, pero las opciones más importantes para los objetivos del experimento son las siguientes:

Utilización del firmware LegOS: aunque este firmware no posee ningún tipo de código que permita controlarlo en tiempo real, es uno de los sistemas operativos más completos. Además posee un protocolo de comunicación denominado LNP (Lego Network Protocol) que permite enviar y recibir mensajes al RCX desde el computador. Aunque esto permitiría controlar el robot, es necesario la instalación de una aplicación especial que permita comunicar el RCX con la torre infrarrojo según el protocolo anteriormente descrito. Esta aplicación actualmente disponible en Internet en el site http://legos.sourceforge.net/files/linux/LNPD permite realizar esta función pero posee un inconveniente muy importante, está escrito en lenguaje C.

Como se menciona en el Marco Teórico (Tecnología JINI), JINI sólo puede interactuar con objetos en Java lo cual sería un gran inconveniente al momento de adaptar el componente local y el remoto.

Utilización del firmware estándar de Lego: Como se menciona en el marco teórico, este sistema operativo ofrece un código RCX (el cual es bien conocido) que permite controlar las funciones del RCX en tiempo real desde un computador. Dario Laverde desarrolló un

programa en Java que permite enviar estos comandos desde cualquier plataforma compatible con Java. Este paquete fue el que se utilizó para adaptarlo a las especificaciones y requerimientos del proyecto.

Pero existen varias formas de controlar el RCX desde el computador utilizando este programa. El primero es enviar todos los comandos desde el computador permitiendo controlar el robot a tiempo real y facilitando así las labores de mantenimiento. Esto es posible pero es muy ineficiente. Al enviar cada comando este consume un cierto tiempo el cual puede ofrecer cierto grado de imprecisión (sin mencionar cuando la señal es débil produciendo una pérdida del paquete y su debida retransmisión) . Un ejemplo de la desventaja de este tipo de programación es: si prendemos el motor del puerto A y un segundo después prendemos el motor del puerto B, esto originaría que el robot no se desplace derecho sino que desviase a alguna otra dirección perdiendo la posición correcta del mismo.

Como mencionamos en el párrafo anterior, no es eficiente utilizar esta primera opción. Una segunda opción es que los programas estén almacenados en el RCX. De esta manera tenemos un mayor control de los motores, sensores y así lograr una mayor precisión y control del robot. Al tener almacenado estos programas en el RCX, el computador puede enviar comandos que permitan ejecutarlos evitando así la desventaja del tiempo entre comandos ya que este retardo entre los comandos no afecta el comportamiento del robot.

Una vez obtenido la forma en el que el RCX sería programado, se obtuvo un diseño aproximado del robot. Este debe poseer las siguientes características:

- 2 motores que funcionen independientemente para desplazar el robot. Deben ser independientes para girar el robot en cualquier dirección.
- 1 sensor de choque que indique si el robot encontró o no un obstáculo en el terreno.

- El robot debe moverse sobre el plano de forma horizontal y vertical según las coordenadas cartesianas. Para lograr esto es necesario que el robot pueda girar y desplazarse con gran precisión .

Para esta etapa del proyecto se obtuvo que era necesario evaluar ciertas características para escoger un diseño determinado. Entre las actividades planteadas se encuentran:

- Debe evaluarse la utilización de orugas que permitan doble tracción en las ruedas del robot contra el uso de una rueda loca. Esto permitiría observar cual de las dos alternativas soportan mayor libertad en el desplazamiento del robot.
- Debe evaluarse la utilización de 2 sensores de rotación contra el desplazamiento del robot en un tiempo determinado. Esto con respecto al grado de precisión obtenido en el desplazamiento y giro del robot.
- Debe evaluarse cual debe ser la inclinación del RCX en el robot. La manera más sencilla de construir el robot es incorporarlo de forma horizontal, ya que este dispositivo posee una pieza de Lego en la parte inferior que permite fijar otras piezas brindándoles mayor consistencia. En cambio, al colocar el RCX vertical colocaría la parte inferior del RCX perpendicular al suelo necesitando piezas especiales que permitan incorporar estructuras paralelas al plano donde se desenvuelven. En otro ámbito, al colocar el RCX horizontal es necesario que el puerto infrarrojo siempre apunte a la torre para establecer la comunicación inalámbrica. En caso que esté vertical se puede colocar la torre apuntando hacia el suelo y siempre estará en contacto con el puerto infrarrojo del RCX independientemente de la posición del robot.

También se obtuvieron características representativas de cómo debe estar constituido el terreno a explorar:

- El terreno podría ser una cartulina dibujada con un formato cuadricular. Cada cuadrícula debe poseer una dimensión mayor al robot y estas deben ser uniforme en todos los cuadros que conforman el terreno.
- Los obstáculos deben ser de forma cuadricular y poseer las mismas dimensiones que los cuadros que conforman el terreno, para permitir que el robot pueda captarlos desde cualquier ángulo y posición. Estos también deben ser de un material suficientemente pesado para no ser arrastrados al momento del choque con el robot.

En esta etapa también se obtuvo una representación de la arquitectura para controlar el robot, esta posee un enfoque local ya que en esta iteración se desconoce las características necesarias para controlar el robot remotamente.

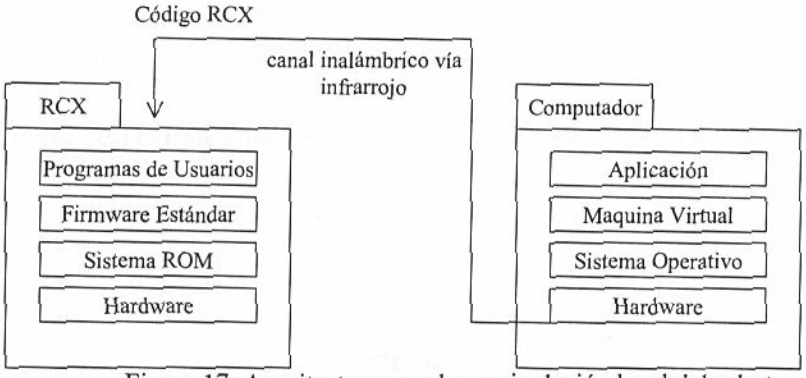


Figura 17: Arquitectura para la manipulación local del robot

Como se observa en la figura 17, la arquitectura del RCX se basa en el firmware estándar de Lego el cual posee 5 espacios de memoria para almacenar los programas de usuario. Este es controlado por un computador que se encarga de enviar código RCX al robot así como recibir una respuesta del estado de los sensores. Esto permitiría a la aplicación recibir si el robot detectó o no un obstáculo en la exploración. Todo esto se realiza a través de un programa realizado en Java.

En esta iteración también se obtuvo cada una de las clases a desarrollar, representadas en forma de un diagrama de clases que se muestra en el apéndice G. Estas establecen el punto de partida para la programación del componente local encargado de controlar el robot.

Investigación y diseño de los componentes remotos (Iteración 2)

En esta iteración se logró un alto dominio en la configuración de servidores de búsqueda así como en la creación de políticas de seguridad, aunque estas últimas no son utilizados en el experimento ya que cualquier persona en la red puede manipular el robot. También se realizó un ejemplo en el cual se obtenía un objeto remoto hospedado en un servidor a partir de un cliente, esto se logró a partir de la tecnología JINI.

Una vez obtenido un conocimiento más detallado de la tecnología, se obtuvo un diseño estimado de la interfaz de la aplicación que interactúa con cada uno de los usuarios. Esta se muestra en el apéndice H. Este diseño se enfoca en que el cliente pueda explorar y, al mismo tiempo, observar el estado del terreno de una manera fácil y atractiva para el usuario.

En esta etapa se obtiene un diagrama de clases de la aplicación diseñado para funcionar localmente (ver apéndice G). Este será utilizado en la iteración 2 de la etapa de construcción y permitirá probar el funcionamiento del robot a partir de la aplicación sin la incorporación del componente JINI.

Asimismo se obtiene un diagrama de clases del componente JINI complementando el diagrama de clases de la aplicación, esta vez se toma un enfoque cliente servidor. Estos diagramas se muestran en el apéndice G.

Por último se obtiene una representación aproximada de la arquitectura necesaria para manipular el robot remotamente. Este modelo es mostrado en la figura 18 y está conformada por un servidor conectado al robot vía infrarrojo y que se encarga de publicar en el servicio de búsqueda un servicio para manipular o visualizar el estado del robot. Un servidor web que se

encarga de transmitir los objetos "stub" a los clientes que va a permitir la invocación de métodos remotos ubicados en el servidor. Por último, un servidor de búsqueda el cual se encarga de publicar y asignar el servicio.

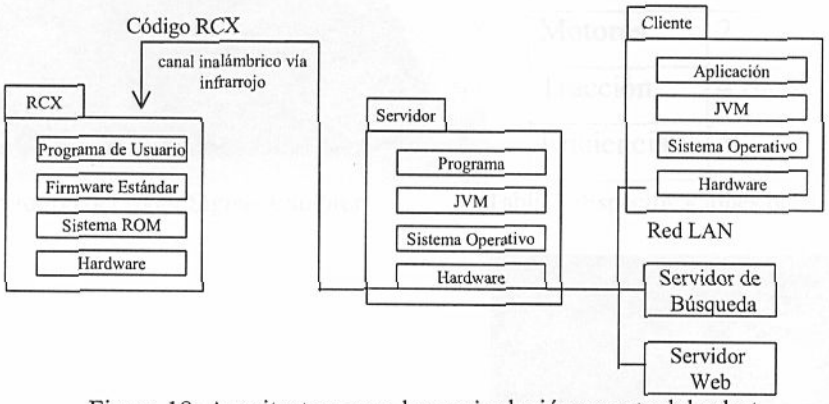


Figura 18: Arquitectura para la manipulación remota del robot

V.3 Fase de Construcción

Como se mencionó anteriormente esta fase se encarga principalmente de la programación y pruebas de todos los componentes separados para su posterior vinculación y adaptación al ambiente distribuido. A continuación se muestran cada uno de los resultados obtenidos en cada iteración:

Desarrollo y prueba de los componentes locales (Iteración 1)

En esta iteración se obtuvo toda la programación y construcción necesaria para controlar el robot localmente sin necesidad de poseer una aplicación que funcione como interfaz (esta aplicación es realizada en las iteraciones posteriores).

A partir del análisis obtenido en las fases anteriores se realizaron dos prototipos de robots de exploración mostrados en la figura 19 y 20:

Figura 19: Prototipo del robot Signus Explorer

Nombre	Signus Explorer
Sensores	1 de choque
Motores	2
Tracción	4 ruedas, oruga
Eficiencia	*

Tabla 3: Especificaciones del 1er robot realizado

Nombre	Omega Centaury
Sensores	1 de choque, 2 de rotación
Motores	2
Tracción	2 ruedas, independientes
Eficiencia	****

Tabla 4: Especificaciones del 2do robot realizado

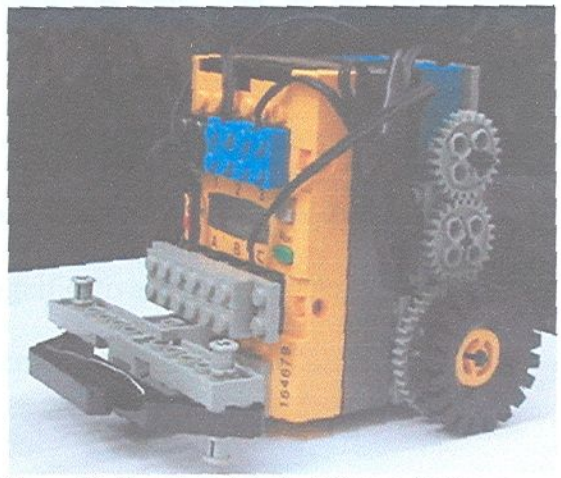


Figura 20: Prototipo del robot Omega Centaury

Como se puede observar en la figura 19, el primer robot construido fue el Signus Explorer. Este robot no fue utilizado por varias razones. La primera fue la falta de precisión del robot al momento de realizar las pruebas, su desplazamiento era medido en unidad de tiempo generando fuertes problemas de precisión a la hora de explorar. También se dificultaba el giro ya que las orugas ofrecían una gran resistencia a los motores. La segunda razón es que el robot no era capaz de recibir las señales de la torre infrarrojo desde cualquier posición por lo cual era necesario construir programas especiales que permitieran ubicarlo en una posición adecuada para recibir y transmitir las señales agregando mayor complejidad a la exploración.

Cierto tiempo después para cubrir las ineficiencias de Signus Explorer, fue contruido Omega Centaury. Para esto fue necesario el préstamo de dos sensores de rotación que servirían como indicadores rotacionales ofreciendo una mayor precisión en el desplazamiento

del robot. Este robot posee dos motores conectados a un sistema de engranajes que permiten movilizar las ruedas y los sensores de rotación al mismo tiempo. También está constituido por un sensor de rotación unido a dos pequeñas piezas que funcionan como parachoques permitiendo registrar un tropiezo en varios puntos de la parte delantera del robot. Omega Centaury posee una pieza especial en la parte inferior que permite que el robot gire y se desplace libremente solucionando los conflictos producidos por la resistencia de las ruedas delanteras. En el apéndice I se muestran detalladamente cada uno de los componentes mencionados anteriormente.

A partir de este robot, el desplazamiento del mismo viene dado por el número de vueltas que realizan los motores y no por el tiempo el cual ha transcurrido desde que el robot comenzó a moverse. Aunque todo esto aumenta significativamente la eficiencia del robot al momento de la exploración, este puede tener ciertas desventajas que serán mencionadas posteriormente.

Con respecto al terreno, este es construido sobre una cartulina blanca el cual posee cuadros de 30 cm de lado. Otros materiales como fórmica y granito permitían que el robot se desenvolviera naturalmente pero son difíciles de transportar de un lugar a otro. La longitud del lado fue obtenido a partir de los análisis realizados en las etapas anteriores y es aproximadamente tres veces mayor a las dimensiones del robot Omega Centaury. Esta cartulina es colocada encima de un pedazo de formica o cualquier otra superficie totalmente lisa para ofrecer la menor resistencia posible en el momento que el robot realice la exploración. En la figura 25 se muestra la representación del terreno una vez construido.

En esta fase también se obtuvo los programas que permiten al robot desenvolverse. Estos fueron programados y creados con la aplicación distribuida por Lego para luego ser descargados al RCX. El uso de NQC (ver sección de Lego Mindstorm) no fue necesario dado la simplicidad de los programas. En la tabla 5 se muestran cada uno de estos programas:

Programa	Función
IrAdelante()	Mover el robot hacia delante tantas vueltas, si consigue un obstáculo detiene los motores e invoca la rutina atrás.
Atrás()	Se encarga de devolver al robot a su posición inicial al momento de chocar con un obstáculo
Medio(mensaje msg)	Se encarga de mover el robot pocas vueltas hacia atrás o hacia delante dependiendo del mensaje enviado por la aplicación ejecutada en el computador
Girar90Izq()	Gira el robot 90 grados hacia la izquierda
Mover90Der()	Gira el robot 90 grados hacia la derecha

Tabla 5: Especificaciones de los programas descargados al RCX

Cada uno de estos programas fueron descargados en un espacio de memoria del RCX. En el apéndice J se muestran cada uno de los programas con sus respectivos espacios de memoria en el cual fueron almacenados.

Una vez realizadas las pruebas unitarias de cada programa, aparecieron ciertas limitaciones en el desenvolvimiento del robot:

- Es necesario el control de la velocidad. A mayor velocidad el robot obtiene mucha menor precisión (deslizamientos, mala lectura de los sensores de rotación, etc). Si la velocidad es muy baja, puede permitir la inmovilidad del robot por la resistencia ofrecida en los motores.
- La variación de la velocidad del robot con respecto a la energía. La energía es directamente proporcional a la velocidad del robot. Es decir, si la energía es alta la velocidad es alta y viceversa. Esto puede provocar problemas descritos en el primer punto.

En esta iteración se logró también la configuración de los paquetes javax.COMM y rcx.jar los cuales permiten controlar el robot desde cualquier aplicación Java. Una vez obtenida esta configuración se obtuvo un programa que se encarga de enviar los comandos que permiten ejecutar los programas ubicados en el RCX, logrando así un control del experimento localmente.

Una vez logrado estos resultados se obtuvieron nuevas limitaciones para controlar el robot desde un computador:

- La interferencia causada por la mayoría de las luces. Esta es agravada por la posición vertical del RCX (el cual coloca el puerto infrarrojo mirando hacia arriba) siendo interferido por cualquier luz de techo ubicado en el ambiente.
- Es necesario que la torre infrarrojo esté constantemente dentro del umbral de comunicación para evitar pérdida de paquetes. Para lograr esto es necesario que el terreno sea lo suficientemente pequeño o en su defecto lograr que la torre infrarrojo se movilice conjuntamente con el robot.

Desarrollo y prueba de la aplicación (Iteración 2)

En esta etapa se obtuvieron distintos archivos que permiten ejecutar una aplicación gráfica para controlar el robot y ver el resultado de la exploración realizada por el robot. Este control es realizado localmente lo cual no posee ningún componente especial que permite controlarlo desde cualquier lugar de una red.

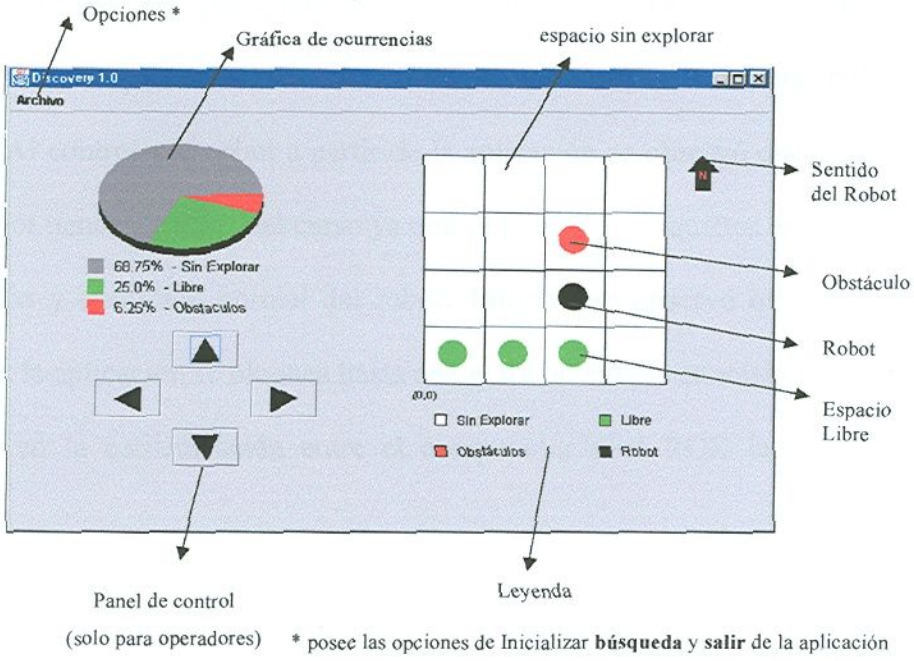


Figura 21: Prototipo de la aplicación

En la figura 21 se puede observar el prototipo de la aplicación. Esta está constituida por tres secciones principales. La primera conforma la representación del terreno, esta puede poseer cualquier dimensión tomando en cuenta que debe ser cuadrado. En este mapa existen varias representaciones: el color blanco quiere decir que el área no ha sido explorada, el color negro indica la posición exacta del robot, el color verde indica que el área esta libre (no existen obstáculos) y por último el color rojo indica que el robot colisionó con un obstáculo en esa posición. También se adiciona una figura en la parte superior derecha que indica la dirección y sentido actual del robot.

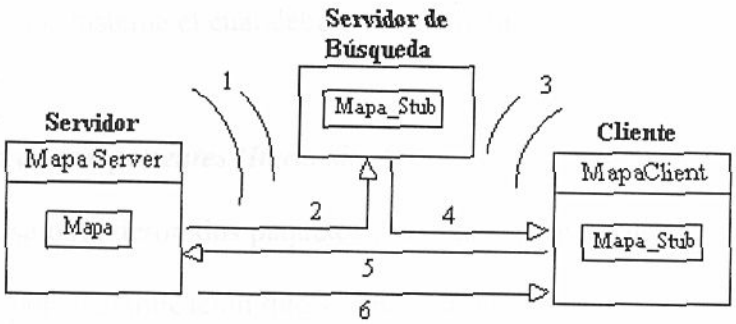
En la parte superior izquierda se encuentra una figura que indica el porcentaje de areas libres, no exploradas y obstáculos obtenidos durante la exploración.

En la parte inferior izquierda de la aplicación se encuentra el panel de control el cual se encarga de manipular el robot hacia el norte, sur, este, oeste. Este panel estará sólo disponible al operador.

Una vez obtenida esta aplicación, se adapta a los resultados obtenidos en la iteración anterior, logrando obtener la manipulación local del robot a partir de una aplicación con interfaz gráfica. Al controlar el robot a partir de la aplicación se observó que luego de varias iteraciones el robot tiende a salirse del curso ya que se producen pequeños errores que al final alteran la posición y ubicación normal del robot. También se observó que al momento de controlar el robot la aplicación se bloquea hasta que el robot cumple su misión, de esta manera si hay un error en la comunicación entre el computador y el RCX la aplicación queda inhabilitada.

Desarrollo y prueba del componente remoto (Iteración 3)

En esta etapa, se programa y prueba el componente JINI, dado por un servidor capaz de publicar un objeto remoto en los servidores de búsqueda ubicados en una red, así como un cliente capaz de obtener el servicio en cualquier servidor de búsqueda invocando así los método en forma remota vía RMI. En la figura 22 se muestra el proceso de obtención del servicio mediante el componente JINI construido.



1 Exploración de todos los servidores de búsqueda ubicados en la red

2 Publicación del objeto "Mapa_Stub" en el servidor de búsqueda

3 Exploración de todos los servidores de búsqueda por parte del cliente

4 Obtención del objeto "Mapa_Stub" por medio del protocolo "Lookup"

5 Invocación de métodos para la manipulación del robot o para la visualización del terreno

6 Notificación

Figura 22: Proceso de obtención del servicio a través del componente JINI

En la figura 22 se puede observar como el servidor realiza una exploración de todos los servidores de búsqueda disponibles en la red a través del protocolo "Discovery" de JINI. Luego publica un servicio "Mapa_Stub" el cual contiene los métodos para manipular el robot y visualizar la composición del terreno de manera remota. Una vez publicado el servicio el servidor obtiene un ID que será utilizado posteriormente para poder reanudar de nuevo la publicación del servicio en el servidor de búsqueda.

Para la obtención del servicio, el cliente realiza una exploración de todos los servidores de búsqueda disponibles en la red (Discovery), para luego obtener el objeto por medio del protocolo "Lookup" de JINI. De esta manera el cliente obtiene un proxy RMI que es definida

por una interfaz MapaInterface que permite invocar los métodos del objeto Mapa (servidor) de forma remota. Asimismo, los métodos son ejecutados completamente en el servidor y no en el cliente. Esto permitió obtener un recurso centralizado dado por la manipulación del robot y una representación del terreno común a todos los usuarios de la aplicación.

Esta etapa también permitió desarrollar una comunicación entre el cliente y el servidor a través de eventos remotos, de esta manera podríamos notificar a cada uno de los usuarios que hubo un cambio en el sistema el cual debe ser registrado.

Adaptación y prueba de componentes (Iteración 4)

En esta etapa se obtuvieron dos paquetes dividídos en las siguientes partes: Un cliente el cual está formado por una aplicación que se encarga de invocar a un objeto MapaClient. Este último se encarga de obtener un objeto remoto (A partir de JINI) el cual viene dado por un mapa representando el terreno donde el robot se desenvuelve. También se obtiene un servidor el cual se basa en notificaciones remotas para comunicar un cambio en el sistema y posee todos los métodos necesarios para controlar el robot según especificaciones de un operador. En el apéndice J se presenta cada clase perteneciente a cada paquete.

Este objeto remoto (Mapa) implementa una interfaz MapaInterface. Esta define todos los métodos necesarios para controlar el robot y a la vez todos los métodos necesarios para visualizar la composición del terreno y recibir notificaciones del servidor. De esta manera se segmentan los usuarios en el sistema distribuido. El primero controla el robot y el segundo visualiza el estado de la exploración hasta que el recurso sea liberado.

El proceso de obtención del servicio Mapa es el siguiente. El cliente se conecta con el o los servidores de búsqueda realizando una solicitud de un servicio que invoque los métodos pertenecientes a la interfaz MapaInterface. Una vez obtenido, el cliente verifica si algún usuario se encuentra manipulando el robot en ese instante, si no existe simplemente carga la

aplicación como modo escritura. El modo lectura, permite ejecutar todos los método definidos por la interfaz MapaInterface y principalmente los métodos de control del Robot (Mover el robot hacia el Norte, Sur, Este u Oeste) .

En el caso en que no este disponible el recurso, este se incorpora al sistema como sólo lectura. Este no es capaz de manipular el robot pero puede recibir notificaciones remotas por parte del servidor acerca del estado de la exploración.. En la figura 23 se muestra este proceso.

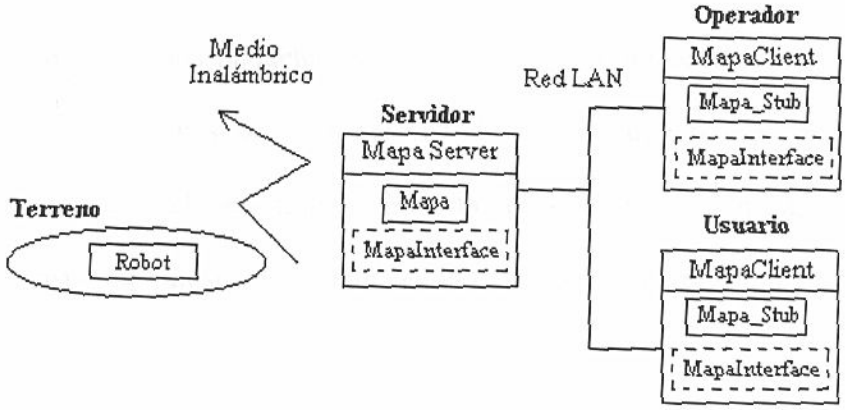


Figura 23: Segmentación del cliente en el sistema distribuido.

En la figura 23, se muestra un operador que obtiene un objeto "Mapa_Stub" que se encarga de realizar llamadas a métodos ubicados en el servidor. Este operador está capacitado para controlar el robot.

En caso de que otro usuario acceda al sistema este es colocado en una cola de espera (ubicada en el servidor) y obtiene un objeto "Mapa_Stub" que no puede manipular el robot pero es capaz de obtener información acerca del estado de la exploración. Una vez que el operador cierra la aplicación el primer usuario en la cola recibe una notificación especial en el cual se le asigna el permiso de controlar el robot. En caso de que este usuario haya perdido la conexión con la red, el recurso es asignado al próximo en la lista.

Una vez realizadas las pruebas mencionadas en la sección de desarrollo, se obtuvo ciertos conflictos con el sistema.

- Si un usuario en espera del recurso pierde la conexión el sistema no debe tratar de asignar el recurso ya que esto provocaría que ningún otro usuario pueda obtener el recurso.
- En caso de que el operador pierda conexión de red o simplemente se caiga el sistema este recurso no es liberado y por lo tanto ningún usuario puede obtenerlo.

V.4 Fase de Transición

Una vez obtenidos los paquetes para manipular el robot remotamente se procedió a realizar ciertas modificaciones de la aplicación de manera que pueda funcionar para las características específicas del experimento. En esta etapa se resolvieron algunas limitaciones de la navegación así como ajustes en la aplicación para poder soportar un número limitado de usuarios.

En cuanto a las limitaciones obtenidas en la etapa de construcción, se obtuvieron soluciones a cada una de ellas, entre las cuales se pueden mencionar:

- En caso de que el usuario en espera del recurso pierda la conexión, el recurso es asignado al próximo de la lista y así sucesivamente.
- En caso de que el operador se caiga, Es necesario correr un proceso paralelo o hilo en el servidor que notifique al objeto Mapa si un operador ha estado cierto tiempo sin controlar el robot. Este contador es reiniciado cada vez que el robot realiza un movimiento. En caso de que el operador no haya manipulado el robot en el tiempo reglamentario, es forzado a obtener el recurso como sólo lectura (mediante un mensaje) ubicándolo en la posición final de la lista de espera. Una vez culminado esto, el recurso es asignado al primer usuario en la lista.

Una vez solucionado las limitaciones anteriores se obtuvieron los obstáculos necesarios para el experimento. También se obtuvo un soporte especial hecho de madera balsa que permite sostener la torre infrarrojo apuntando hacia el suelo y así evitar algunos problemas de comunicación entre la torre y el RCX al momento de realizarse el experimento.

Por último se comprobó la multiplataforma de la tecnología JINI pudiendo conectar un cliente en plataforma Windows con un servidor ubicado en plataforma Solaris Sparc. En la figura 24 y 25 se muestra el ambiente en el cual el robot se desenvuelve así como la ejecución del experimento.

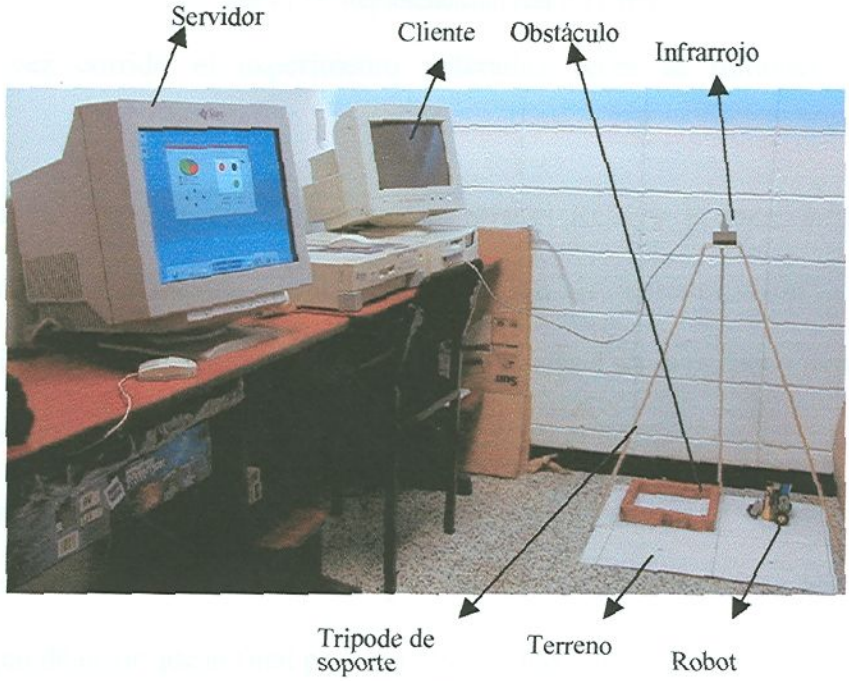


Figura 24: Ejecución del experimento.

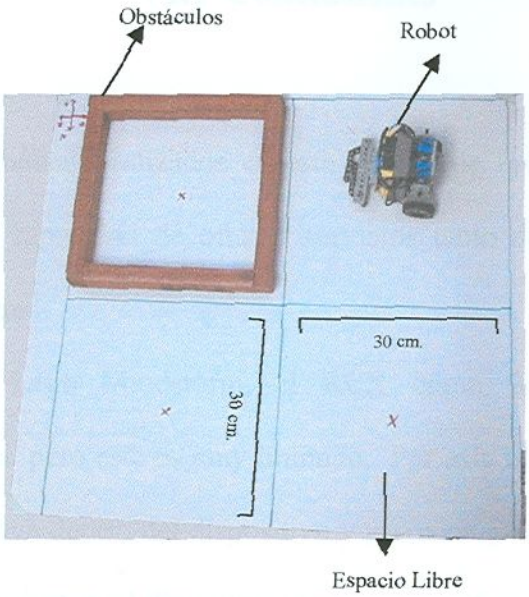


Figura 25: Representación real del terreno.

Una vez corrido el experimento reiteradas veces se obtuvieron los siguientes resultados:

- Frecuente pérdida de comunicación entre el RCX y la torre infrarrojo. Esto por limitaciones mencionadas anteriormente en la comunicación inalámbrica. Las condiciones del experimentos deben ser reducidas a un lugar poco iluminado, así como una posición y distancia correcta entre la torre y el RCX.
- Luego de que el robot realiza varios desplazamientos en el terreno, puede perder la posición correcta. Esto se debe a que cada movimiento realizado produce un pequeño margen de error que al final provoca que el robot pueda desviarse de su objetivo.
- Algunos conflictos en la obtención de los servicios. Estos que se consideran propios de JINI vienen dados por errores que en ocasiones se producen al momento de establecer la comunicación entre el servidor de búsqueda y el cliente. Esto se ve mejorado al ejecutar varias veces la aplicación.

VI.1 Conclusiones

A partir de los análisis realizados en este trabajo de investigación, sabemos que la tecnología JINI tiene la capacidad de ofrecer servicios tanto de dispositivos de Hardware como servicios de software.

En el caso de Lego Mindstorm, el RCX como dispositivo posee poder de procesamiento y memoria, pero este es muy limitado. Por esta razón el RCX no puede correr una máquina virtual de Java dentro del dispositivo y por consiguiente no puede almacenar los paquetes de JINI que permitieran al dispositivo comunicarse directamente con la red y ofrecer sus servicios sin necesidad de un computador centralizado que lo controle. Además el RCX no posee mecanismos que permitan comunicarse con una red en específico.

De esta manera es necesario comunicar el RCX con un computador que actúe como proxy del dispositivo dado las limitaciones del mismo.

Con respecto al experimento, la comunicación entre la torre infrarrojo y el puerto infrarrojo del RCX está muy limitada a la posición y espacio de separación entre ambos elementos, así como fuertes interferencias en relación al ambiente. Por esta razón no se recomienda la utilización de este dispositivo para usos avanzados en el cual se necesite una constante comunicación entre elementos móviles.

Las ventajas de la tecnología JINI que fueron observadas en la ejecución del experimento se muestran a continuación.

- La flexibilidad mostrada en la utilización de distintos protocolos de comunicación de redes. En el experimento se utilizaron los protocolos de RMI (para invocar métodos remotos ubicados en el servidor proveedor del servicio) y http (para transmitir el

objeto stub al cliente y poder establecer la comunicación directa entre este y el servidor que provee el servicio).

- Permite una red más dinámica, las aplicaciones clientes no tienen que conocer donde está ubicado el servicio, de esta manera cualquier cambio ocurrido en la red con respecto a la ubicación del servicio es transparente para el cliente.
- Extiende las ventajas de un modelo cliente/servidor. Al facilitar la administración de la red se pueden crear servicios distribuidos vs. centralizados (dependiendo del caso), que permitan interactuar en una federación JINI y así cumplir con las necesidades de una organización.
- Gracias a las listas de seguridad, no es necesaria la instalación y configuración de firewalls internos que permitan restringir accesos a recursos no autorizados.

Como se observó a lo largo del proyecto hubo una gran desventaja en el uso de la tecnología JINI. Esta restringió el uso de otras alternativas de firmware (específicamente LegOS), por no existir aplicaciones realizadas en el lenguaje de programación Java que establecieran la comunicación entre el RCX y el computador. En la vida real, la mayoría de las aplicaciones están escritas en diferentes lenguajes de programación siendo esta razón principal por la cual esta tecnología pueda ser descartada en el uso dentro de una organización.

En un futuro, JINI tendrá que evolucionar para unificarse con otras tecnologías y así poder sobrevivir en un mundo cambiante, en el que predomina una fuerte competencia.

V.2 Recomendaciones

Para trabajos complementarios futuros, sería interesante la implementación de varios servicios en la red, que interactúen unos con otros para resolver un problema en específico. También se podría implementar la utilización de varios dispositivos de Lego Mindstorm interactuando unos con otros así como lograr que estos puedan aprender y tomar decisiones por sí mismos utilizando Inteligencia Artificial.

Otra recomendación es adaptar proyectos de Inteligencia Artificial a un ambiente multiusuario con la tecnología JINI. También sería importante evaluar tanto la tecnología JINI comparándolo con Corba a partir de ciertos experimentos que permitan mostrar las ventajas y desventajas de cada una de estas alternativas.

Otro trabajo de gran importancia en la extensión de este trabajo de investigación, sería la utilización de otros dispositivos de hardware como es el caso de impresoras, otros productos de invención de robots, cafeteras y otros. que permita comprobar la compatibilidad de JINI con otros dispositivos.

Bibliografía

- Dave Baum, Michael Gasperi, Ralph Hempel & Luis Villa (2000). Extreme Mindstorms: an advanced guide to Lego Mindstorms.
- Sun Microsystems (1995-2001). Fundamentals of RMI. Jguru. Extraído de la página Web http://developer.java.sun.com/developer/onlineTraining/rmi/RMI.html.
- Artima Software, Inc. (1996-2001). FAQ for JINI-user mailing list. Extraído de la página Web http://www.artima.com/jini/faq.html.
- Jan Newmarch (12 de Junio de 2001). Jan Newmarch's Guide to JINI Technologies versión 2.8. Extraído de la página Web: http://pandonia.canberra.edu.au/java/jini/tutorial/Jini.xml.
- Carlos Beltrán Gonzalez (1998-1999). RMI mano a mano con SSL: Construyendo aplicaciones distribuidas seguras. Extraído de la página Web: http://java.programacion.net/taller/joa_rmissl.php
- Sun Microsystem (2000). Jini™ Architecture Specification versión 1.1. Documento Extraído de la página Web: http://www.sun.com/jini/.
- Noel Enete (2001). Noel's Nuggets Examples. Extraído de la página web: http://www.enete.com/download/#_nuggets_
- Ivar Jacobson, Grady Booch & James Rumbaugh (1999). The Unified Software Development Process. Adisson Wesley Longman, Inc.
- Sun Microsystems Inc. (2001), commApi package Version 2.0.2 for Solaris Sparc & Version 2.0 for Microsoft Windows. obtenido de la página Web: http://java.sun.com/products/javacomm/index.html.

- Dario Laverde (1999). rcx.jar. Extraído de la página web: http://www.escape.com/~dario/java/rcx/
- Kekoa Proudfoot (1998-1999). RCX Internals. Extraído de la página Web: http://graphics.stanford.edu/~kekoa/rcx/#Protocol
- Lego Group (2001). Extraído de la página web: http://www.lego.com/home.asp.
- Patricio Salinas Caro & Nancy Histchfeld K. Tutorial de UML. Extraído de la página Web: http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html.
- Cay S. Horstmann & Gary Cornell (2000). Core Java 2 Volume II – Advanced Features. Prentice may.

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

CONSTRUCCIÓN DE UNA APLICACIÓN QUE INTERACTÚE CON UN ROBOT A TRAVÉS DE LA TECNOLOGÍA JINI.

(APÉNDICES Y ANEXOS)

TRABAJO ESPECIAL DE GRADO

presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

como parte de los requisitos para optar por el título de

INGENIERO EN INFORMÁTICA

REALIZADO POR Daniel E. Vera Gonzalez

PROFESOR GUÍA Dr. Wilmer E. Pereira

FECHA 18 de Octubre de 2001

Índice de Apéndices

Apéndice A : RMI y otras tecnologías 1

Apéndice B : Continuación de la Tecnología JINI 3

Apéndice C : Lego Mindstorm 7

Apéndice D : JINI y Corba 13

Apéndice E : Corrida de un componente JINI 15

Apéndice F : Plan de Pruebas 19

Apéndice G : Diagrama de Clases 21

Apéndice H : Diseño de la Interfaz 33

Apéndice I : Prototipo de los Robots 34

Apéndice J : Paquetes resultantes 35

Índice de Anexos

Anexo 1 : Comandos del RCX 39

Apéndice A : RMI y otras tecnologías

Carlos Beltrán (1998-1999) expresa que existen diversas tecnologías que proporcionan estos servicios. Entre estas la más popular (hasta el momento) ha sido CORBA (Common Object Request Broker Architecture). CORBA es una tecnología creada y mantenida por el OMG (Object Management Group), un consorcio formado por un grupo de empresas implicadas en el uso de programación distribuida. La principal característica de CORBA es su independencia del lenguaje de programación mediante el uso de una interfaz de programación común denominada IDL (CORBA Interface Definition Language).

"Desde Java es posible usar CORBA mediante Java IDL y aprovechar todas sus características, sin embargo, los ingenieros de SUN debieron pensar que esta solución era poco elegante y decidieron implementar una tecnología propia de programación distribuida orientada a objetos totalmente basada en JAVA" Carlos Beltrán (1998-1999). Esta tecnología es conocida como RMI (Remote Method Invocation).

"Pero, ¿por qué utilizar RMI y no utilizar CORBA?. Sin duda, es casi obligatorio realizar una comparativa entre ambas tecnologías debido a que, efectivamente, podemos elegir entre las dos para implementar nuestras aplicaciones distribuidas. Numerosos artículos cubren esta discusión y prácticamente llegan a una misma conclusión: si nuestra aplicación distribuida se desarrolla totalmente en JAVA, utiliza RMI" Carlos Beltrán (1998-1999).

Carlos Beltrán (1998-1999) indica que, por otro lado, CORBA es una tecnología asentada con el paso de los años y con un sólido soporte de muchas empresas. Ciertamente

RMI no esta en la misma situación y si es utilizada se debe confiar en la "promesa" de SUN Microsystems de seguir soportando la tecnología RMI, mejorándola e incrementándola, de forma que las aplicaciones podrán seguir funcionado en el futuro. En cualquier caso, el objetivo público de SUN es hacer compatibles ambas tecnologías, quizás de la mano de IIOP(Internet Inter-Orb Protocol). La figura 1 muestra una representación entre Corba y RMI.

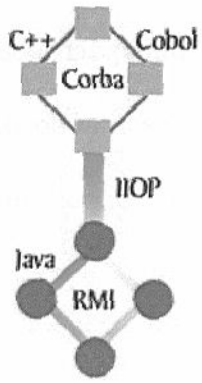


Figura 1: Relación entre RMI y Corba

Apéndice B : Continuación de la tecnología JINI

Servicio de búsqueda

Sun Microsystems (2000) indica que los objetos en un servicio de búsqueda pueden incluir otros servicios; esto provee búsquedas jerárquicas. Además, un servicio de búsqueda puede contener objetos que encapsulen otros nombres o servicios de directorios, dando un camino para construir puentes entre un servicio de búsqueda JINI y otros servicios de búsqueda.

Seguridad

Según Sun Microsystems (2000), los servicios JINI son accedidos en nombre de una entidad (el director) el cual generalmente se remonta a un usuario en particular del sistema. Estos servicios pueden pedir acceso a otros basándose en la identificación de los objetos que lo implementa. El acceso a un servicio es permitido dependiendo del contenido de la lista de control de acceso asociado a cada objeto.

Arrendamiento

Sun Microsystems (2000) afirma que el arrendamiento puede ser de dos formas: exclusiva o no exclusiva. El arrendamiento exclusivo asegura que nadie más puede arrendar el recurso durante el período de préstamo. Por otra parte el arrendamiento no exclusivo permite que múltiples usuario comparta el recurso.

Componentes

Un sistema JINI puede ser visto como una extensión de la infraestructura de una red, así como el modelo de programación y los servicios que hacen de la tecnología JAVA un éxito en una máquina individual. Estas categorías con sus componentes correspondientes son mostrados en la siguiente tabla:

	Infraestructura	**Modelo de programación**	**Servicios**
Java	Máquina Virtual de Java RMI Seguridad Java	API de Java Java Beans	JNDI Enterprise Beans JTS
Java + Jini	Discovery/Join Distributed Security Lookup	Leasing Transactions Events	Printing Transaction Manager JavaSpaces Service

Tabla 1: Segmentación de la arquitectura JINI

Interfaces

Según Sun Microsystems (2000), las interfaces en la tecnología JINI se pueden dividir en:

La interfaz de *arrendamiento* es una extensión del modelo de lenguaje de programación Java agregando la noción de tiempo a la pertenencia de un recurso, habilitando las referencias a resguardarse ante las fallas de la red.

Las interfaces de *eventos* y *notificación* extienden el modelo estándar usado por los componentes de JavaBeans y el ambiente de aplicación Java para un caso distribuido, esto permite habilitar a los eventos para ser manejados por otros objetos mientras hacen varias

entregas y garantiza la puntualidad. El modelo también reconoce que la entrega de una notificación distribuida puede ser retrasada.

Las interfaces de *transacción* introducen un ligero protocolo orientado a objeto que permite a las aplicaciones usar tecnología JINI para coordinar cambios de estado. El protocolo de transacciones provee dos etapas para coordinar las acciones de un grupo distribuido de objetos. La primera etapa es llamada la Fase de votos, en el cual cada objeto vota si tienen completa su porción de tarea y esta listo para comprometer cualquier cambio hecho. La segunda etapa, el cual un coordinador emite un "commit"pedido a cada objeto.

Según Sun Microsystems (2000), el protocolo de transacción JINI difiere de muchas interfaces de transacción en el cual no se asume que las transacciones ocurren en un sistema de procesamiento de transacciones. Tales sistemas definen mecanismos y requerimientos de programación que garantizan la implementación correcta de una semántica de transacciones en particular. El protocolo de transacciones JINI toma una vista más tradicional de una semántica de transacción orientada a objetos.

La ventaja de un protocolo de transacción es definir las interacciones entre los objetos para coordinar tales grupos de operaciones. Las interfaces que definen el modelo de programación JINI son usadas por los componentes de la infraestructura. Por ejemplo, el servicio de búsqueda hace uso del arrendamiento e interfaces de eventos. El arrendamiento asegura que los servicios registrados estén disponibles. Los eventos ayudan al administrador a descubrir problemas así como los dispositivos que necesiten configuración. El servicio de JavaSpaces, es un ejemplo de un servicio utilizado en la tecnología JINI. Este utiliza arrendamiento y eventos, y también soporta el protocolo de transacciones JINI. El gerente de

transacciones puede ser usado para coordinar la fase de votos de una transacción para esos objetos que soportan protocolos de transacciones.

Según Sun Microsystems (2000) afirma que la implementación de un servicio no es requerida en el modelo de programación JINI, pero tales servicios necesitan usar ese modelo para sus interacciones con la infraestructura de la tecnología. Por ejemplo, cualquier servicio interactúa con el servicio de búsqueda usando el modelo de programación, aunque este ofrezca recursos sobre una base de arrendamiento o no, el servicio de registro con el servicio de búsqueda será por arrendamiento y necesitará ser periódicamente renovado.

Según Sun Microsystems (2000) indica que la mezcla de un modelo de programación a los servicios y la infraestructura hace que la Agrupación de un sistema JINI no sea una colección de servicios y protocolos. La combinación de infraestructura, servicio y modelo de programación están diseñadas para trabajar unidas, simplificando el sistema y unificándolo de tal forma que lo hace fácil de entender.

Algunos ejemplos de servicios en la tecnología JINI son:

- Un servicio de impresión, el cual puede imprimir desde aplicaciones escritas en un lenguaje de programación Java.
- Un servicio JavaSpace, el cual puede ser usado para establecer una comunicación simple y para almacenar grupos de objetos escritos en Java y relacionados entre sí.
- Un gerente de transacciones, el cual permita a un grupo de objetos participar en un protocolo de transacciones JINI definido por el modelo de programación.

Apéndice C : Lego Mindstorm

Historia

Según Dave Baum (2000) en 1949, Godtfred Kirk Christiansen produjo el primer prototipo de Lego denominado Vinculación de Piezas Automáticas del inglés "Automatic Binding Bricks", pocos años después denominado Ladrillos de Lego del inglés "Lego Bricks". A través de los años 1960 y 1970 estos ladrillos o piezas aparecieron en nuevos tamaños y formas.

En 1977, Lego lanzó un producto denominado Technic el cual poseía otro tipo de componentes como ejes, engranajes, vigas, anillos así como articulaciones universales. Estos y otros elementos más sofisticados son usados actualmente en el desarrollo de estructuras robóticas.

Según Baum (1999), no fue hasta 1998 cuando Lego lanzó al mercado la primera colección de Lego Mindstorm (Sistema de Invención de Robots –RIS 1.0–). Este aparte de contener los componentes mencionados anteriormente adiciona otra cantidad de ellos. También incluyó el RCX que es un computador programable de Lego que no solo permite realizar movimientos sino que también tiene la capacidad de sentir y responder a su ambiente.

En la figura 2 se muestra los componentes que conforman el RCX.

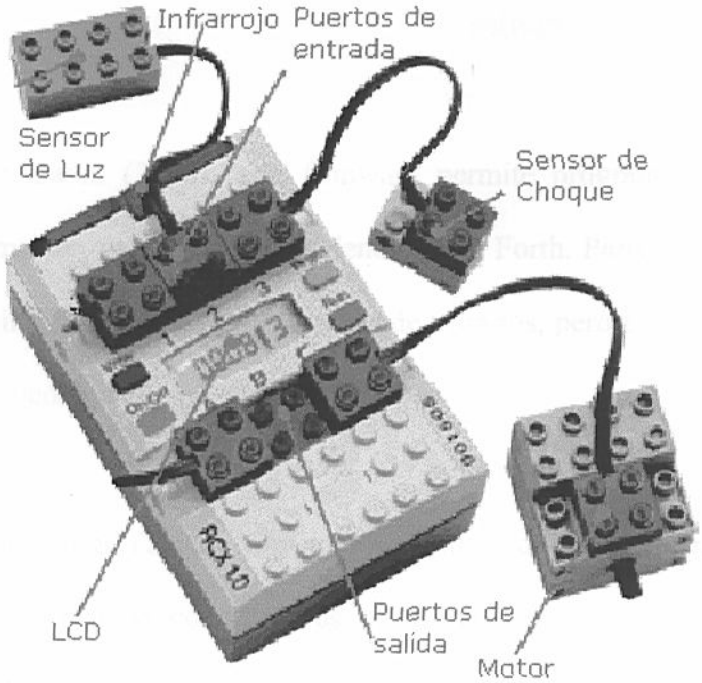


Figura 2: Componentes del RCX

Botones de control

On-Off: se encarga de encender apagar el RCX.

View: se encarga de seleccionar qué sensor o motor aparecerá en la pantalla, permitiendo ver el estado actual.

Prgm: se encarga de seleccionar uno de los 5 programas establecidos para el firmware estándar de lego.

Run: ejecutar o detener el programa seleccionado.

Firmware

Aparte del firmware estándar de Lego, existen otros que permiten programar al RCX para su debido funcionamiento. Entre los más comunes están:

PbForth

Según Dave Baum (2000), este firmware permite programar al RCX usando un lenguaje de programación de computadoras denominado Forth. Parecido al fimware estándar, pbForth provee un interprete para los programas de usuarios, pero a diferencia de los códigos de Lego, este ejecuta código compilado Forth.

Según Dave Baum (2000), otra particularidad de pbForth radica en que otras herramientas de programación compilan los programas de usuario en el computador para luego transmitir al RCX el código traducido. En cambio, pbForth permite que el código sea traducido en el RCX. La figura 3 ilustra esta operación.

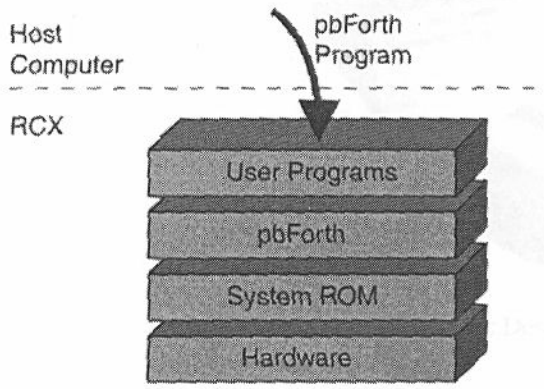


Figura 3: Arquitectura del RCX con el firmware pbForth

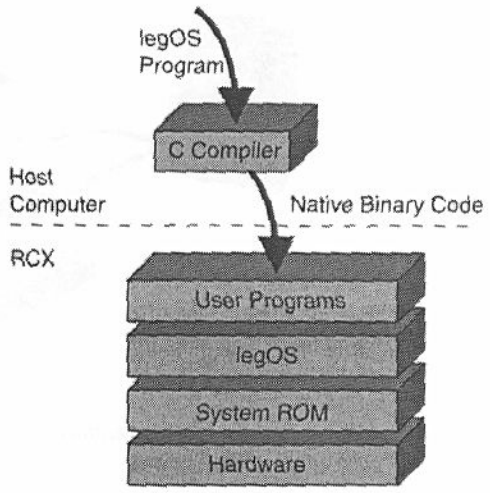


Figura 4: Arquitectura del RCX para LegOS

LegOS

Este firmware es otra sustitución del sistema operativo estándar de Lego. Este se programa con un lenguaje semejante a C. Como se muestra en la figura 4 del Apéndice C, el computador compila un programa LegOS y genera código binario que puede ser ejecutado nativamente por el RCX. Este código binario es transmitido al robot donde LegOS se encarga

de ejecutar el programa y provee una interfaz con el Hardware del dispositivo. Es uno de los más usados por compartir una afinidad con el lenguaje de programación C.

Luz Infrarrojo

El RCX utiliza una luz infrarrojo para comunicarse con el computador o con otro RCX. Esta comunicación es realizada por una torre de infrarrojo que se muestra en la figura 5 y que es conectado al puerto serial (de 9 pines) del computador.

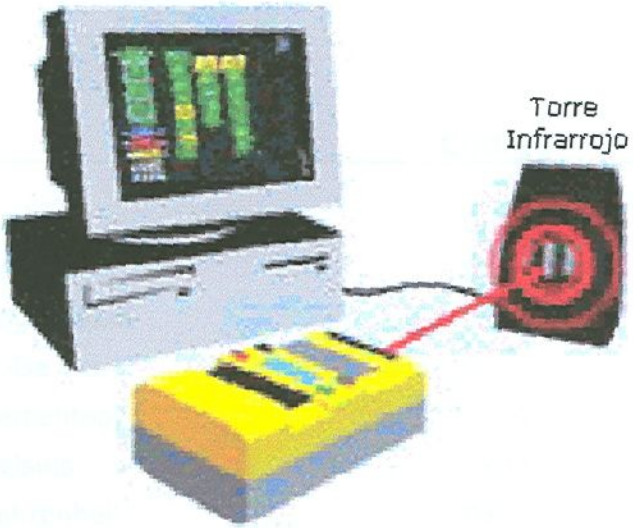


Figura 5: Descarga de programas al RCX

Sensores

En la tabla 2 se muestran los diferentes tipos de sensores y sus respectivos modos predeterminados:

Tipo	Código	Clase
Ninguno	0	Pasiva
Choque	1	Pasiva
Temperatura	2	Pasiva
Luz	3	Activa
Rotación	4	Activa

Tabla 2: Tipos de sensores.

En la tabla 3 se muestran los diferentes modos que un sensor puede obtener:

Modo	Código
Raw	0x00
Bolean	0x20
Edge Count	0x40
Pulse Count	0x60
Percentage	0x80
Celsius	0xa0
Fahrenheit	0xc0
Rotation	0xe0

Tabla 3: Modos de los sensores para el firmware estándar.

Como se puede observar en la tabla 3, Dave Baum (2000) expresa que existen ocho diferentes modos de sensores los cuales sólo funcionan para un cierto tipo de sensor. El más general es el modo "Raw" el cual se encarga de medir valores entre 0 y 1023 unidades. Para saber si un sensor de choque se encuentra presionado se obtiene un registro entre 0 y 460 unidades, en caso contrario se obtiene un registro entre 562 y 1023. Para los registros obtenidos entre 460 y 562 unidades el valor no es definido. El caso del modo "Bolean", este

posee solo dos valores 1 si es positivo y 0 si es negativo. Los otros modos son variaciones que permiten obtener valores precisos para cada tipo de sensor.

Apéndice D : Jini y Corba

Unas de las tecnologías de red más cercana a la filosofía de JINI es Corba, ya que posee ciertas características similares que las relacionan estrechamente en la utilización de recursos remotos en un sistema distribuido.

Según Artima (2001), Corba se asemeja a RMI con la diferencia que posee un elemento denominado "trader service" traducido como comerciante de servicios. Este mecanismo es muy parecido al servicio de búsqueda de JINI con ligeras diferencias que más adelante serán nombradas. Según Artima (2001) la tabla 1 muestra las diferencias entre RMI y Corba:

Corba	RMI
Permite invocar métodos en objetos remotos escritos en cualquier lenguaje de programación.	Permite invocar métodos provenientes de máquinas virtuales de java diferentes
Con respecto a la programación, Se requiere que el código del objeto Stub sea conocido por el programador que desarrolla el cliente.	Este puede enviar los archivos .class del objeto stub por la red permitiendo que el cliente pueda cargar dinamicamente el código del objeto stub.
Ofrece "invocación dinámica de interfaces" que permite al cliente usar objetos remotos sin la definición del objeto stub.	Es necesaria la definición del objeto stub en el cliente
Hasta CorbaIIOP 2.2 (1998) sólo es permitido pasar objetos por referencia.	Permite pasar objetos por valor o por referencia cuando se invoca un método remoto.
Corba no permite obtener objetos que extienden de otros objetos, lo cual si un método remoto espera una clase Animal y el cliente pasa a un "Perro", el objeto remoto recibirá un Animal y no un Perro	Este permite obtener objetos que extienden otros objetos ya que aunque la máquina virtual de java del servidor no conozca la clase "Perro"el cliente enviará esta clase a través de la red para poder obtener la definición de esta clase.

Tabla 4: diferencias entre Corba y RMI

Como se mencionó anteriormente, Corba incluye un servicio que es semejante al servicio de búsqueda de la tecnología JINI, y es el punto principal de comparación entre ambos.

Según Artima (2001), estos servicios más que proveer un nombre con el cual un objeto remoto es asociado (como es el servidor de nombres en Corba y el registro RMI), se realiza una búsqueda por el tipo de objeto remoto que se desea obtener.

En cuanto a esto, el servicio de búsqueda de JINI ofrece un poco más de flexibilidad en la búsqueda, ya que se puede realizar de otras maneras, sea por el identificador del servicio o por atributos del servicio.

Pero, según Artima (2000), la mayor diferencia entre ambos servicios radica en el resultado de las consultas. El "Corba trader service" retorna una referencia remota a un objeto remoto. Mientras que el servicio de búsqueda retorna un objeto proxy por valor.

Con respecto a Corba, cuando se obtiene una referencia remota al objeto remoto, el stub local se comunicará con el objeto remoto vía Corba.

En el caso del servicio de búsqueda de JINI el servicio objeto podría no comunicarse a través de la red completamente. Este puede implementar el servicio localmente en la máquina virtual del cliente. También puede comunicarse con un servidor(es) o algún dispositivo de hardware vía Sockets y Streams. Otra ventaja es que el objeto servicio puede comunicarse por cualquier protocolo de red incluyendo Corba, agregando una mayor flexibilidad en las comunicaciones entre los objetos.

Apéndice E : Corrida de un componente JINI

Es necesario cumplir con tres requisitos principales para lograr la corrida de un componente JINI. Estos requisitos son los siguientes:

1. ***La existencia de un servidor de búsqueda en la red***: en caso de que no exista uno, este debe ser configurado e instalado para poder lograr la comunicación entre el servidor y el cliente. Los pasos necesarios para lograr la instalación de un servidor de búsqueda en el sistema operativo Solaris Sparc 8 son:

   - Instalar los paquetes de JINI: la versión utilizada en este trabajo de investigación fue la V. 1.1.
   - Configuración del classpath: consiste en configurar el Classpath y variables de ambiente necesarios para el buen funcionamiento de los archivos de JINI. Un ejemplo se muestra en los siguientes comandos:

     JINI=/dvera/jini/JINI1_1

     CLASSPATH=$JINI/lib/reggie.jar;$JINI/lib/jini-core.jar;$JINI/lib/jini-ext.jar;.

   - Correr el servidor WEB: se puede saltar este paso en el caso que exista uno en la red. Aunque puede ser utilizado cualquier tipo de servidor web, JINI posee un simple servidor web capaz de transmitir archivos .class por la red y en el caso de JINI es utilizado para transmitir los objetos remotos de un lugar a otro. Este servidor es ejecutado de la siguiente manera:

     Java –jar $JINI/lib/tools.jar -dir $JINI/lib/ -port 8080

- Correr RMID: este es un demonio que permite establecer la comunicación entre el cliente y el servidor. También brinda la seguridad en la red permitiendo que procesos y personas no autorizados traten de acceder a los objetos que son publicados en el servidor de búsqueda. Este se corre de la siguiente manera:

  rmid -J-Dsun.rmi.activation.execPolicy=none

- Correr Reggie: reggie es el servidor de búsqueda proporcionado por JINI. Para su funcionamiento es necesario que los pasos anteriores hayan sido cumplidos exitosamente. El comando para correr este servidor es el siguiente:

  java -jar $JINI/lib/reggie.jar http://sun/reggie-dl.jar $JINI/example/lookup/policy.all /dvera/reggie_log public

Este comando contiene un total de 4 parámetros que son explicados en la tabla 5:

Parámetro	Significado
http://sun/reggie-dl.jar	permite obtener desde cualquier parte de una red las clases que contienen el objeto registrar. Esto se logra publicándolo en un servidor web
$JINI/example/lookup/policy.all	Consiste en la ubicación del archivo donde se encuentran las políticas de seguridad del servicio de búsqueda
/dvera/reggie_log	La ubicación del log en donde se almacenan todas las solicitudes realizadas al servicio de búsqueda
public	El grupo que puede utilizar el servicio

Tabla 5: significado de cada uno de los parámetros utilizados para correr el servidor de búsqueda

2. ***Correr el servidor:*** Este paso es necesario para publicar el servicio en el servidor de búsqueda. se debe compilar los archivos.java del servidor y luego generar los Stub publicando estas clases en el servidor web para que otros clientes puedan obtener estas clases. Los pasos a seguir se muestra en la tabla 6

Comando	Significado
Javac *.java	De esta forma se compilan todos los archivos java generando los archivos .class
Rmic –v1.2 –d $JINI/lib/clases/ Mapa	De esta forma se generan los archivos Stub, el parámetro –v indica que sólo se generaran stubs para versiones de java superiores al 1.2. El parámetro –d indica en que path serán colocados los stub generados. Por último Mapa es el archivo el cual extiende de la clase UnicastRemoteObject permitiendo al servidor ser manipulado remotamente por cualquier cliente.
java -Djava.security.policy=%JINI%\policy\policy.all -Djava.rmi.server.codebase=http://veragon/clases/ MapaServer	Este comando se encarga de correr el servidor y posee varios parámetros. El primer parámetro indica que todo usuario ubicado dentro de la red puede obtener el objeto y el segundo que todos los archivos necesarios para obtener el objeto están ubicados en ese servidor web. MapaServer es la clase que publica el objeto en el servidor de búsqueda

Tabla 6: comandos necesarios para correr el servidor proveedor del servicio

3. ***Correr el cliente:*** para corre el cliente es necesario estar conectado a la red y poseer los paquetes de JINI en el Classpath. El cliente debe conocer las interfaces que son implementadas por la clase que va a ser publica, de otra forma no es posible obtener el objeto. También es necesario compilar todos los archivos java para obtener los archivos.class. El comando para correr el cliente es el siguiente:

```
java -Djava.security.policy=$JINI/policy/policy.all Discovery
```