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

■ Daniel Vera

Este proyecto consiste en realizar un trabajo de investigación acerca de cómo puede la tecnología J I N I 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*es utilizado por los clientes para obtener el servicio.

JINI está ubicado encima de cualquier protocolo de comunicación tal es el caso de RMI (Remote Method Invocation), *Sockets,* entre otros.

Luego de un análisis detallado de las herramientas necesarias, se realiza un experimento que incluye un robot capaz de explorar un terreno especifico en búsqueda de 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.

I. **Motivación**

Actualmente se han 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 o visualizado 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, ¿qué ventajas puede ofrecer ésta en la manipulación de los dispositivos a través de la red?

II. Objetivos del proyecto

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

III. Marco Referencia)

A continuación se presentan una serie de conceptos relacionados con todos los componentes utilizados en el ciclo de vida del proyecto. Entre los elementos que se engloban a la largo de este capítulo se encuentran:

- Una breve explicación sobre RMI.
- Filosofía y características de la tecnología JINI.
- Lego Mindstorm.

RMI (Invocación de Métodos Remotos)

En esta sección 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.

La idea básica de RMI es que objetos ejecutándose en una JVM (Máquina Virtual de Java) sean 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).

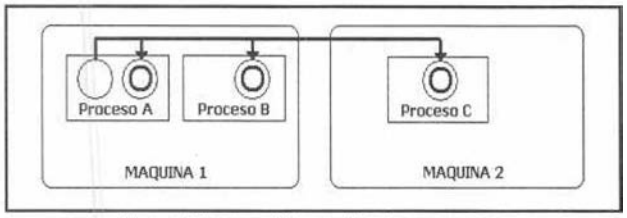

Figura 1: Invocación a métodos remotos en máquinas virtuales diferentes

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.

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, la 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 esta separación.

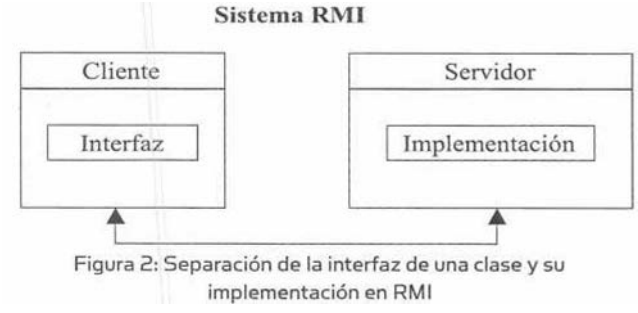

Figura 2: Separación de la interfaz de una clase y su implementación en RMI

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.

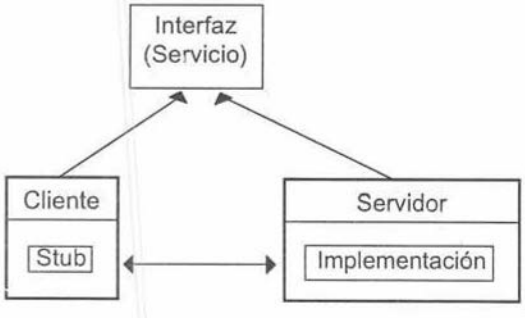

Figura 3: Servicios de una interfaz en RMI

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.

La arquitectura de RMI consta de tres capas: la capa *Stub,* la capa de referencia remota y la capa de transporte.

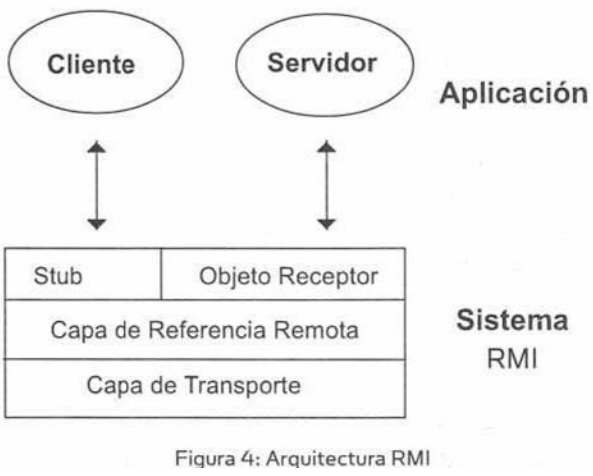

Figura 4: Arquitectura RMI

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.

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 y 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 y 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.

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.

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 porque puede 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 utilizados a lo largo de este trabajo de investigación.

"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 conectares extraños" *SunMicrosystems* (2000):

■ Operación inmediata: cuando se conecta un dispositivo con tecnología JINI a la red, sencillamente funciona. Los servicios y recursos están disponibles de inmediata.

■ Comunidades improvisadas: **el** *software* permite que todos los dispositivas 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 eléctricas 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 can tecnología JINI están disponibles en todo momento, cuando se requieran.

Como se puede observar en la figura 6, un sistema JINI es un sistema distribuido o federación basado en la idea de implementar grupos de usuarios y las 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 encontradas por clientes humanos y procesas. 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.

Entre los elementos que conforman la tecnología **JINI** se encuentran:

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

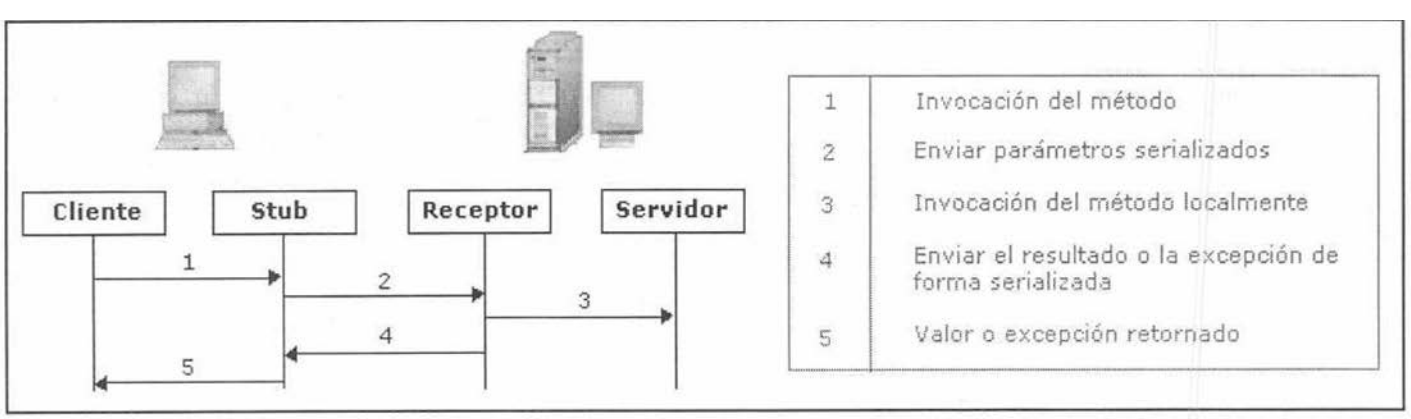

Figura 5: Proceso de invocación de métodos en RMI

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.

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.

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.

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.

Protocolos: según *Sun Microsystems* (2000), El corazón de un sistema JINI radica en este trío de protocolos llamados *Discovery, Joiny Lookup.* Un par

Figura 6: Filosofía de la Tecnología JINI

de esos protocolos *(Discovery y Join)* se activan cuando un dispositivo es instalado. El primero se activa cuando un servicio busca un servidor de búsqueda con el cual registrarse. El protocolo *Join* se activa cuando un servicio es localizado por un servidor de búsqueda y desea unirse al conjunto de servicios. *Lookup* ocurre cuando un cliente o usuario necesita localizar e invocar un servicio descrito por el tipo de interfaz (escrito en Java) y posiblemente otros atributos. La figura 7 ilustra el proceso *Discovery,* la figura 8 muestra el proceso de *Join* y la figura 9 muestra el proceso *Lookup.*

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).

Figura 7: Discovery

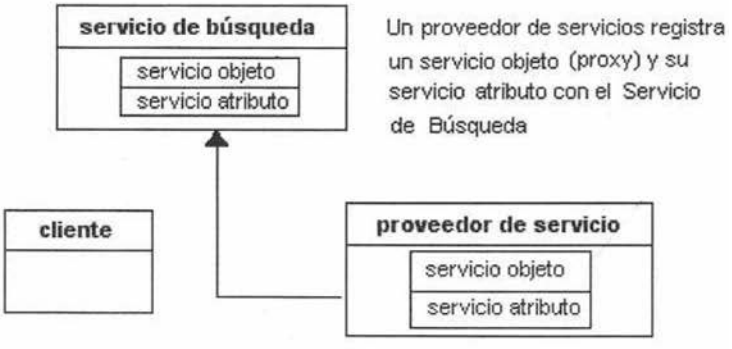

Figura 8: Join

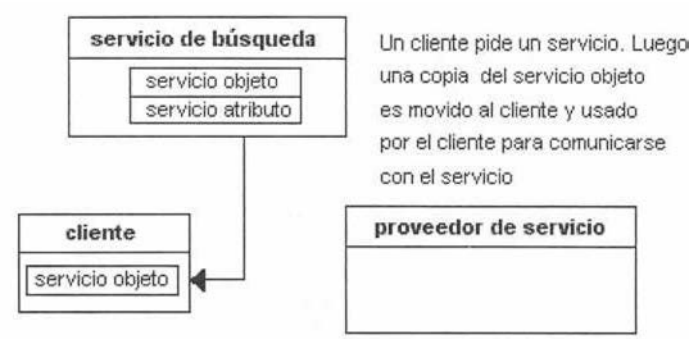

Figura 9: Lookup

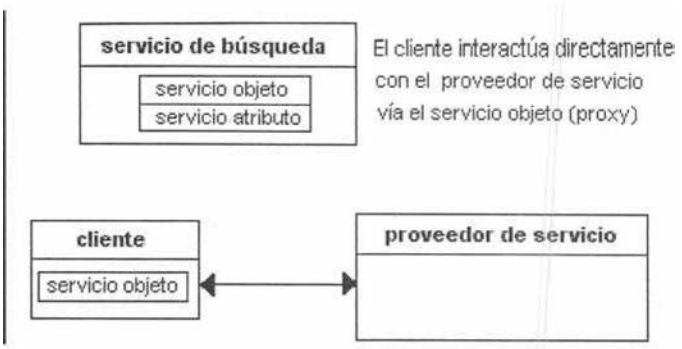

Figura 10: El cliente usa el servicio

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 como se muestra en la figura 10.

Lego Mindstorm

Una vez conocido todos los mecanismos de red que son utilizados en el experimento es necesario conocer brevemente qué es *Lego Mindstormy* cuales son sus componentes principales.

Lego Mindstorm, también denominado Sistema de Invención de Robots, es un producto desarrollado por la compañia Lego en 1998. Este permite diseñar y programar robots reales, los cuales actúan de una manera específica.

El componente principal es el RCX, el cual es el cerebro del sistema (mostrado en la figura 11). 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 ❑ave Baum (2000) el RCX se puede dividir en varias capas: la capa de *hard-ware,* la capa del sistema ROM y el *firmware.*

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.

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. 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. Actualmente existen varios firmware en el mercad❑. Entre ellos los mas importantes son: Firmware estándar, LegOS, PbForfth entre otros. En este trabajo de investigación fue escogido el firmware estándar por su razones explicadas en secciones posteriores.

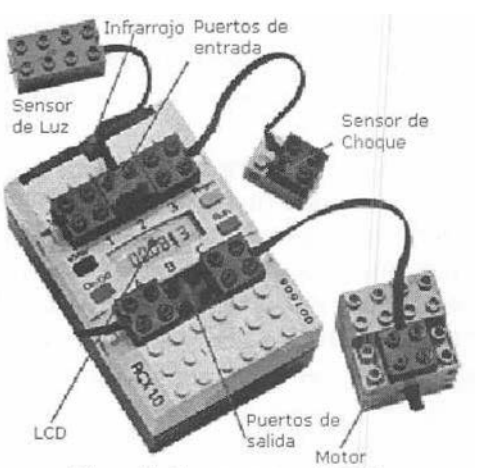

Figura 11: Componentes del RCX

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 12 se muestra como el código RCX y los programas NOC son ejecutados dentro del RCX.

Según Baum (2000), existen herramientas especiales llamadas compiladores (en el caso de NOC) 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.

Una vez obtenidos estos códigos son transmitidos al RCX vía luz infrarrojo 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 diferentes tipos de programación para cada tipo de *firmware.* El *firmware* estándar ofrece una capacidad de memoria aproximadamente de 6KB para almacenar un programa de usuario así como 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, éste 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.

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 (htto://www.escape.com/ - dario/java/rcx).

IV. Metodología

La metodología utilizada en el desarrollo de este trabajo de investigación se basa principalmente en el proceso unificado con ciertas variantes debido a

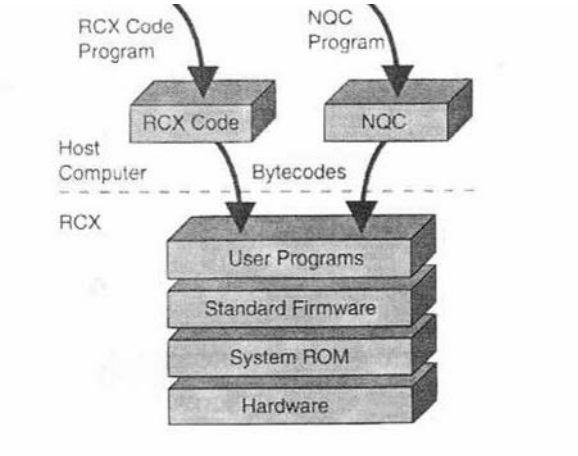

Figura 12: Arquitectura del RCX para el Firmware estándar

características propias del experimento realizado en este trabajo de grado.

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 13.

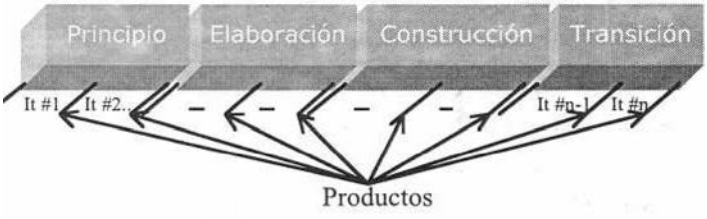

Figura 13: Un ciclo con sus fases e iteraciones

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

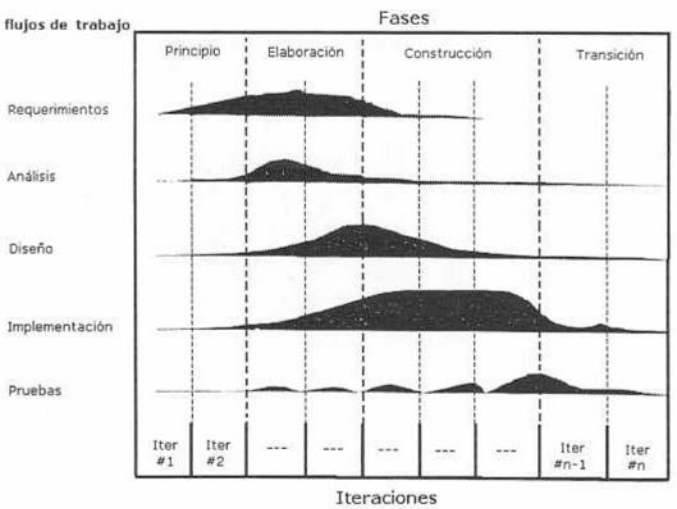

Figura 14: El énfasis de cada flujo de trabajo varía por cada iteración en cada fase

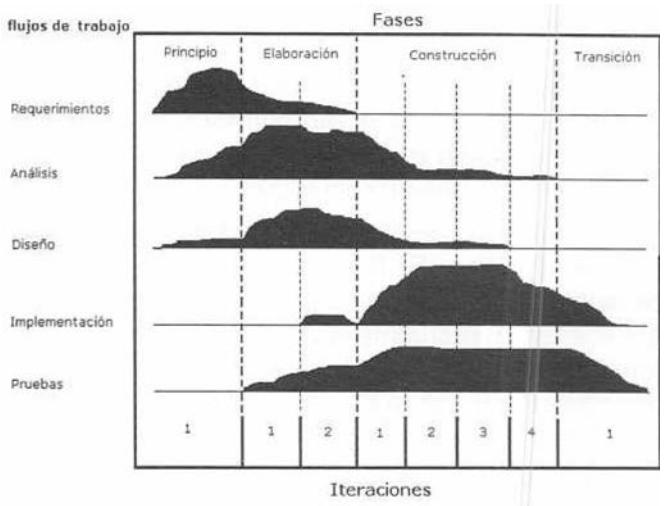

Figura 15: Actividades realizadas en cada una de las fases con sus respectivas iteraciones.

por las siguientes actividades: requerimientos, análisis, diseño, implementación y pruebas. 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.

Como se puede observar en la figura 14, se muestra un proyecto con sus respectivas 4 fases (principio, elaboración, construcción y transición). Cada fase posee un número limitado de iteraciones y en cada una de estas se cumple con una actividad especifica del proyecto.

V. Ejecución

Una vez aclarada la metodología utilizada y la información teórica necesaria para el entendimiento del proyecto, se procede a mostrar todas las actividades realizadas y resultados obtenidos en cada etapa del proyecto.

La figura 15 muestra la ejecución del proyecto según la metodología anteriormente mencionada. En general el proyecto constó de 8 iteraciones distribuidas de la siguiente manera: 1 en la fase de principio, 2 en la fase de elaboración, 4 en la fase de construcción y 1 en la fase de transición. También se puede observar que en cada iteración se hace un énfasis diferente de trabajo en las distintas actividades necesarias para culminar el proyecto (requerimientos, análisis, diseño implementación y pruebas).

En la tabla 1 se muestran las actividades realizadas en cada una de las iteraciones de dichas fases.

Tabla I: Actividades realizadas en cada fase del proyecto

Fase	Iteración	Actividad
Principio	1	Definición de los requerimientos necesarios para desarrollar el trabajo de investigación
Elaboración	I	Investigación y diseño de los componentes necesarios para la manipulación eI 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

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

Análisis de los requerimientos del proyecto (iteración 1)

Desarrollo

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 y Redes quienes aportaron ideas para el establecimiento de uno o varios experimentos que permitieran mostrar las funcionalidades de la tecnología la JI NI.

También se realizaron ciertas entrevistas a distancia de personas conocedoras de la tecnología JINI y Lego Mindstorm, como es el caso de la Ing. lain 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 una 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.

Resultados

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.

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. 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, ésta 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 puede controlar el robot.

Los recursos mínimos necesarios para la ejecución del proyecto son los siguientes: computador que funcione como servidor y cliente con interfaz de red, un sistema de invención de robots de *Lego Mindstorm,* un sensor de choque y dos sensores de rotación, y 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. Éste 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.

▪ 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, éste 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.

Fase de Elaboración

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.

Investigación y diseño de los componentes locales (Iteración 1)

Desarrollo

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 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 uña 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 éste 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 ésta debe ser capaz de interconectarse con otros componentes para su funcionamiento en Red.

Ésta 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.

Resultados

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 *firmwares* mencionados anteriormente, 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. Ésta aplicación actualmente disponible en Internet en el site http://legos.sourceforge.net/fileslinux/LNPD permite realizar esta función pero posee un inconveniente muy importante, está escrito en lenguaje C. Como J I NI sólo puede interactuar con objetos en Java, su implementación podría ser un 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 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 originaria que el robot no se desplace derecho sino que desviase a alguna otra dirección perdiendo la posición correcta del mismo.

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. Éste 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 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, ésta posee un enfoque local ya *que* en esta iteración se desconoce las características necesarias para controlar el robot remotamente.

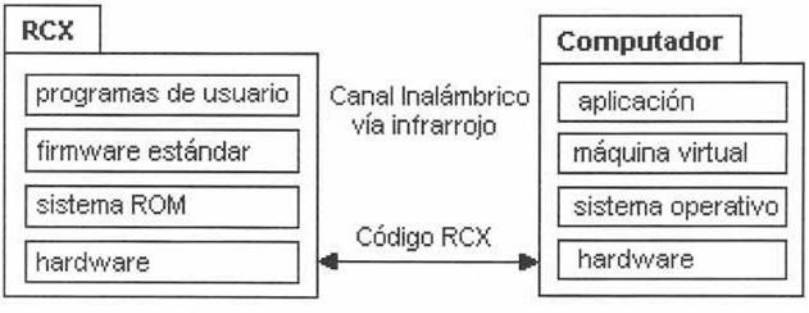

Figura 16: Arquitectura para la manipulación local del robot

Como se observa en la figura 16, 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.

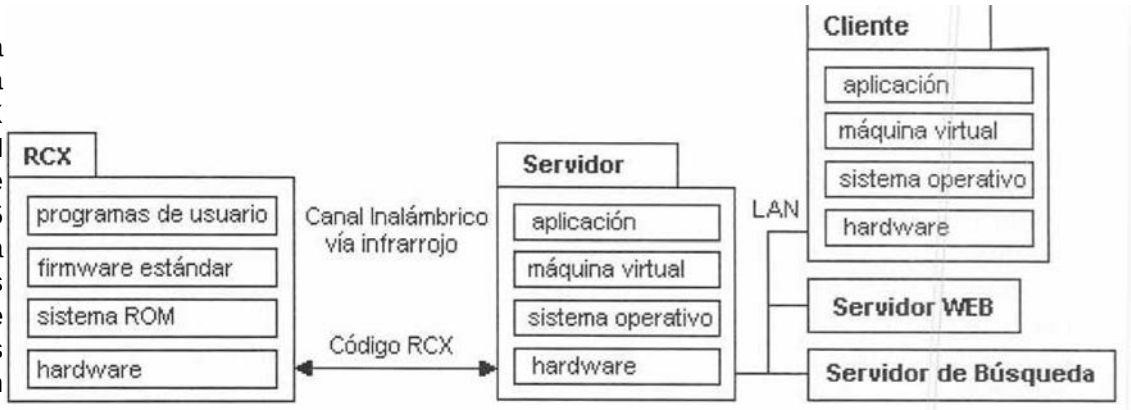

Figura 17: Arquitectura para la manipulación remota del robot

Investigación y diseño de los componentes remotos (Iteración 2)

Desarrollo

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 JINI. 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 Mindstorm.

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.

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* diseñó una arquitectura posible para la manipulación del robot a partir de un cliente remoto.

Resultados

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.

Por último se obtiene una representación aproximada de la arquitectura necesaria para manipular el robot remotamente. Este modelo es mostrado en la figura 17 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.

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 ameritasen.

Desarrollo y prueba de los componentes locales (Iteración 1)

Desarrollo

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. Las actividades realizadas en esta primera etapa fueron:

- Instalación del *firmware.*
- Construcción y prueba de diferentes robots tomando en cuenta los resultados obtenidos en ta 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 (un ejemplo es 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. 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.

Resultados

A partir del análisis obtenido en las fases anteriores se realizaron dos prototipos de robots de exploración.

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 (ver figura 18). 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 choque 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.

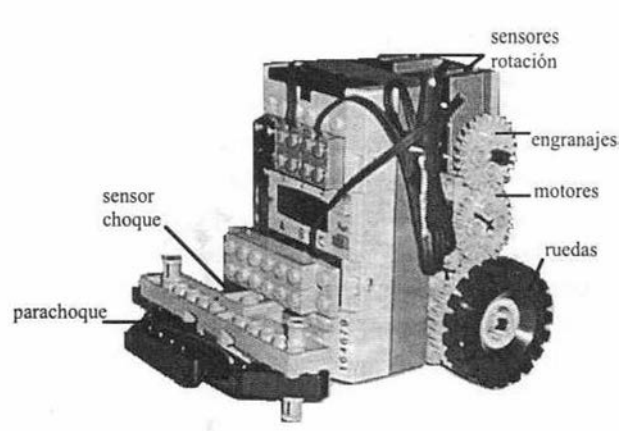

Figura 18: Prototipo del robot Omega Centaury

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 que 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, éste es construido sobre una cartulina blanca el cual posee cuadros de 30 cm de lado. 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 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 2 se muestran cada uno de estos programas:

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)

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
Girar90lzq()	Gira eI robot 90 grados hacia la izquierda
Mover90Der()	Gira el robot 90 grados hacia la derecha

Tabla 2: Especificaciones de los programas descargados al RCX

Desarrollo

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 si.

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.

Resultados

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.

Una vez obtenida esta aplicación, se adapta a los

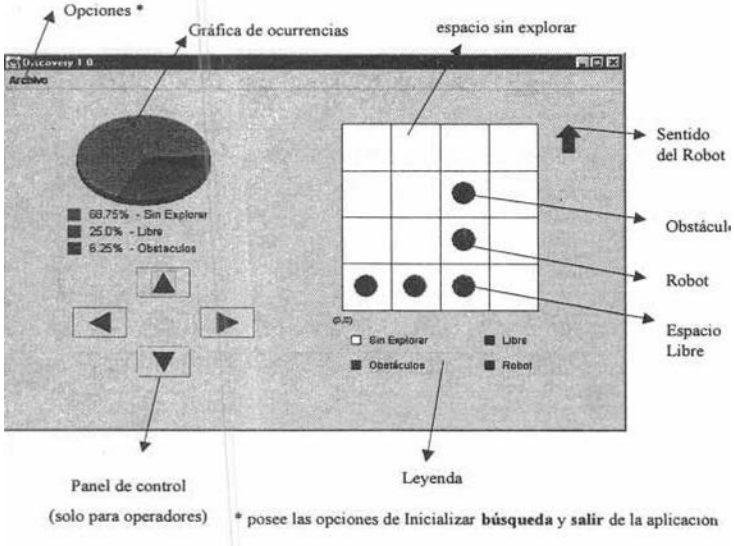

Figura 19: prototipo de la aplicación

resultados obtenidos en la iteración anterior, logrando obtener la manipulación local del robot a partir de una aplicación con interfaz gráfica.

Desarrollo y prueba del componente remoto (Iteración 3)

Desarrollo

En ésta se crea un componente JINI que permite obtener un objeto remoto. De esta manera 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 capitulo 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.

Resultados

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.

En la figura 20 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

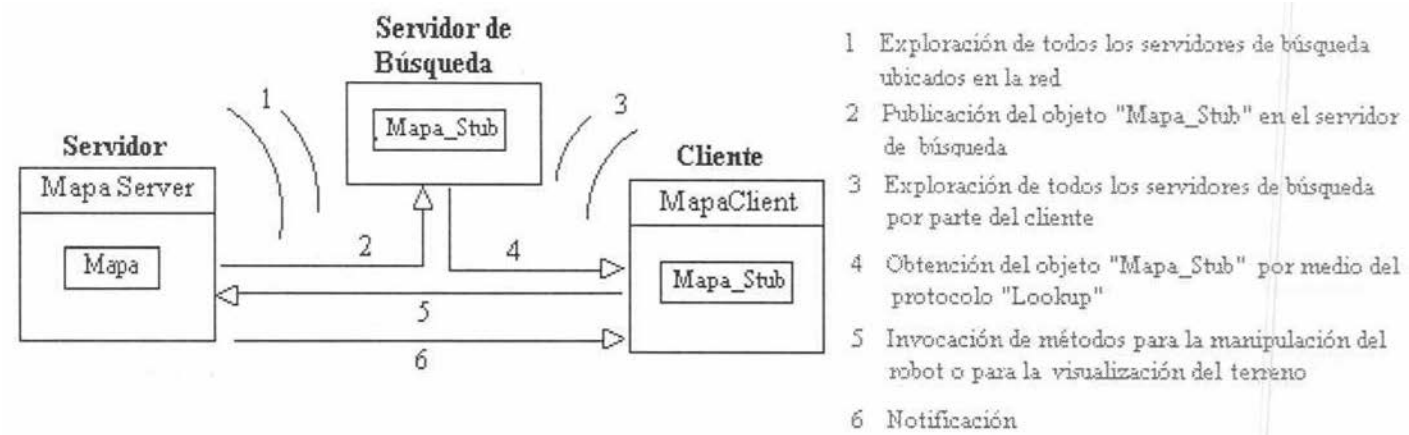

Figura 20: Proceso de obtención del servicio a través del componente JINI

definida por una interfaz *Mapa Interface* que permite invocar los métodos del objeto Mapa (servidor) de forma remota. Asimismo, los métodos son ejecutados completamente en e! 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 e! cual debe ser registrado.

Adaptación y prueba de componentes (Iteración 4)

Desarrollo

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.

Resultados

En esta etapa se obtuvieron dos paquetes divididos en las siguientes partes: un cliente el cual está formado por una aplicación que se encarga de obtener un objeto remoto (a partir de JIM) 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.

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. 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 Mapa Interface. 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 escritura, permite ejecutar todos los métodos 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, éste se incorpora al sistema como sólo lectura. Este no es capaz de manipular el robot pero puede recibir

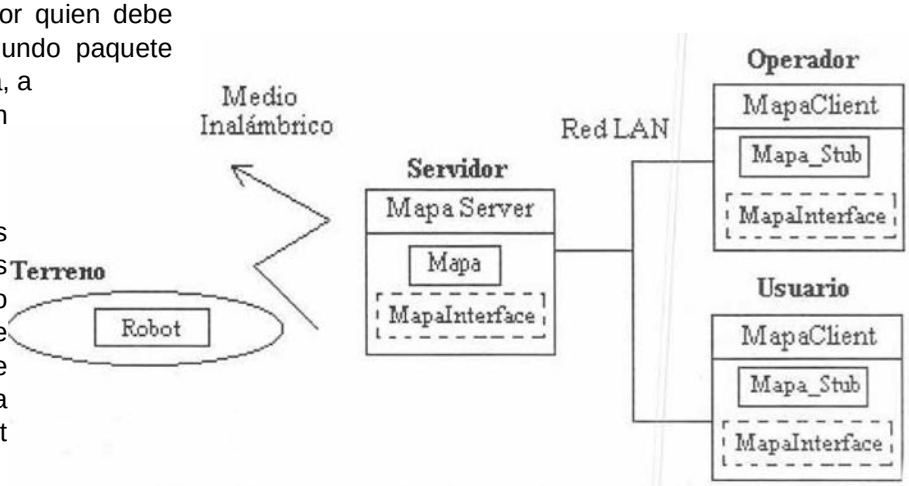

Figura 21: Segmentación del cliente en el sistema distribuido.

notificaciones remotas por parte del servidor acerca del estado de la exploración. En la figura 21 se muestra este proceso.

En la figura 21, 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.

Fase de Transición

Desarrollo

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 se muestran a continuación:

- 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.

Resultados

En fa figura 22 se muestra la corrida del experimento. Se puede observar obstáculos construidos en madera con dimensiones de 30 Cm de lado. Tambien se puede observar como el robot intenta explorar el recuadro en búsqueda de obstáculos y transmitiendo la información hacia la torre infrarrojo del computador.

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

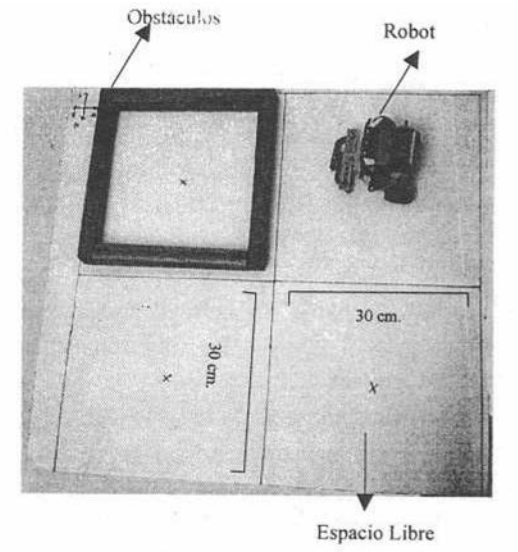

Figura 22: Corrida del experimento

- 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.

V. Conclusiones

A partir de los análisis realizados en este trabajo de investigación, sabemos que la tecnología J I N I 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 corno 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.

VI. 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://graohics.stanford.edu/ - k e koa/rcx/# P roto co l

Lego Group (2001). Extraído de la página web: http://www.lego.com/home.asp.

Cay S. Horstmann & Gary Cornell (2000). Core Java 2 Volume II - Advanced Features. Prentice may.