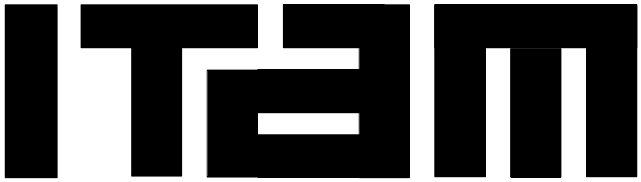
INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO



SISTEMA DE INTERACCIÓN HUMANO – ROBOT BASADO EN VOZ PARA EL ENTRENAMIENTO DE COMPORTAMIENTOS

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN TELEMÁTICA

P R E S E N T A

CARLOS FERNANDO RAMOS LÓPEZ

MÉXICO, D.F. **2009**

Con fundamento en los artículos 21 y 27 de la Ley Federal del Derecho de Autor y como titular de los derechos moral y patrimonial de la obra titulada “SISTEMA DE INTERACCIÓN HUMANO – ROBOT BASADO EN VOZ PARA EL ENTRENAMIENTO DE COMPORTAMIENTOS”, otorgo de manera gratuita y permanente al Instituto Tecnológico Autónomo de México y a la Biblioteca Raúl Bailléres Jr., autorización para que fijen la obra en cualquier medio, incluido el electrónico, y la divulguen entre sus usuarios, profesores, estudiantes o terceras personas, sin que pueda percibir por tal divulgación una contraprestación

Carlos Fernando Ramos López

_____

Fecha

_____

Firma

A mis padres, hermanos, familiares y amigos.
A todos aquellos que hicieron posible este trabajo.

RESUMEN

La interacción humano – robot involucra el diseño, evaluación e implementación de medios de comunicación que permitan el intercambio de información entres seres humanos y robots. El presente trabajo presenta la implementación de un sistema de interacción basado en voz que permite al usuario ordenar al robot Sony AIBO ERS-7 la ejecución de acciones, preguntar al robot información del medio que lo rodea y, finalmente, entrenar comportamientos. Los comportamientos se entrenaron utilizando el concepto de autómata situado: en cada estado del autómata, las percepciones del medio disparan la ejecución de distintas acciones pero el desarrollo del comportamiento es controlado por las transiciones entre los estados del autómata. Los comportamientos pueden estar basados en la ejecución de acciones condicionadas por percepciones o bien pueden construirse a partir en comportamientos previamente enseñados al robot. Las pruebas se realizaron en el contexto del fútbol robótico (*Standard Platform League – RoboCup*). Los resultados obtenidos demostraron la viabilidad del uso de la voz como medio de interacción con un robot.

Palabras clave: interacción humano – robot, entrenamiento de comportamientos, aplicaciones de reconocimiento del habla.

ABSTRACT

Human – robot interaction involves the design, evaluation and implementation of systems that enable the exchange of information between humans and robots. This document introduces the implementation of a voice-based interaction system that allows the user to request Sony AIBO ERS-7 robot the execution of actions, to consult information about the environment that surrounds it and, finally, teach behaviors. The behaviors were trained using the concept of situated automata: in each state of the automata, perceptions of the environment trigger the execution of various activities (actions), but the behavior is controlled by the transitions between states of the automaton. Behavior can be based on the execution of actions conditioned by perceptions or can be constructed based on other behaviors. Testing was conducted in the context of robot-soccer (*Standard Platform League – RoboCup*). The results indicate that the use of voice as a model of interaction between humans and robots is reliable.

Key words: human-robot interaction, teaching behaviors, application of speech recognition.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CUADROS

CAPÍTULO 1

INTRODUCCIÓN

La presente tesis es un trabajo enfocado al diseño e implementación de un sistema de interacción entre humanos y robots, utilizando el lenguaje oral, que le permite al robot Sony AIBO ERS-7 aprender comportamientos; para ello, se hizo uso de la herramienta *CSLU Speech Tools Rapid Application Development* (RAD) en los procesos de reconocimiento y síntesis de voz y del contexto del fútbol de robots de *RoboCup* como ambiente de pruebas.

En este primer capítulo, se describe el problema que esta tesis pretende resolver, se definen los objetivos de la misma así como el alcance de ella; además, se realiza una justificación y una mención de trabajos relacionados. Por último, se explica la estructura en que se organiza este documento.

1.1 ANTECEDENTES

Es común definir a un robot como una máquina programable cuyo objetivo es imitar las acciones o la apariencia de un organismo inteligente [TES 2005]. Para que una máquina califique bajo el denominativo de robot tiene que cumplir con dos características fundamentales: primero, la máquina debe ser capaz de obtener información del entorno que lo rodea; segundo, la máquina debe de poder realizar alguna interacción física con el medio, por ejemplo moverse o manipular objetos en su entorno.

Dada la constante evolución en las capacidades de procesamiento de información, los robots se han vuelto más inteligentes en los últimos años, existe en todo el mundo un importante esfuerzo de investigación enfocado al desarrollo de técnicas para hacer que los robots se muevan y “piensen” de forma más eficiente.

Los robots han evolucionado y se han diversificado para realizar diversas tareas; desde procesos repetitivos en una línea de producción hasta la exploración planetas o de zonas

profundas de los océanos son tareas que hoy realizan. Además, cada vez son más comunes los robots enfocados a realizar servicios personales o de entretenimiento.

Un robot móvil es aquel que tiene la capacidad de desplazarse en el entorno, es decir, no está limitado a una posición fija para realizar una tarea. Los robots pueden clasificarse, además, según su modo de operación en autónomos y no autónomos. A los primeros corresponden todos aquellos que no requieren de la asistencia de un ser humano para su operación. Los robots móviles no autónomos, o controlados, requieren de un operador humano que indique constantemente al dispositivo los movimientos que debe realizar, utilizando algún método de interacción.

Idealmente, el método de interacción entre robots y seres humanos debería ser tan natural como entre seres humanos. Sin embargo, el uso de comandos de movimientos muy específicos o de controladores remotos es ampliamente utilizado para el mando a distancia de robots no autónomos. El problema de este tipo de métodos radica en la complejidad que para el operador implica el aprender a utilizar la interfaz de control.

El uso de la voz como medio de interacción entre un humano y un robot parece ser el paso natural en la búsqueda de un modelo de interacción entre humanos y robots que facilite el intercambio de información o el envío de instrucciones al robot.

En el futuro, la investigación en el área de la interacción Humano-Robot permitirá una comunicación natural, ergonómica y óptima, permitiendo la cooperación entre los seres humanos y sistemas robóticos [WEI 2007].

1.2 PLANTEAMIENTO

El Laboratorio de Comportamiento Adaptivo Neuronal, Neurociencias y Simulaciones (CANNES) y el Laboratorio de Robótica son los centros de ivestigación del Instituto Tecnológico Autónomo de México (ITAM) en los que se desarrollan proyectos en los campos de la inteligencia artificial, visión por computadora, comunicaciones inalámbricas y colaboración de agentes robóticos [SOT 2006].

En año 2005, el Laboratorio de Robótica comenzó a participar en la categoría de cuadrúpedos (*4-Legged League*, posteriormente *Standard Platform League)* de *RoboCup*, utilizando la plataforma Sony AIBO, inicialmente el modelo ERS-210 y posteriormente el modelo ERS-7.

La incursión, hasta 2007, dentro de la liga utilizando el robot AIBO arrojó una amplia experiencia en uso de la plataforma, dicha experiencia es utilizada actualmente para el desarrollo de múltiples investigaciones tanto en el Laboratorio de Robótica como en el de CANNES.

Las recientes publicaciones ([WEI 2006] y [WEI 2008]) relacionadas con el desarrollo de una arquitectura de interacción basada en voz y el conocimientos acumulado sobre el funcionamiento del robot AIBO impulsaron la realización del presente trabajo.

1.3 OBJETIVO

Desarrollar un sistema de interacción humano-robot, basado en voz, que permita ordenar la ejecución de acciones, interrogar y entrenar al robot Sony AIBO para que aprenda comportamientos básicos en el contexto del fútbol de robot de *RoboCup*.

1.4 ALCANCE

En el presente trabajo se desarrolló un sistema de interacción entre humanos y robots utilizando como medio de comunicación el lenguaje hablado, dicho trabajo involucra los siguientes puntos:

- En el robot Sony AIBO, atomización de los comportamientos utilizados en la liga de plataforma estándar (*Standard Platform League*) de *RoboCup* por el equipo *Eagle Knights*, de tal manera que constituyen acciones básicas indivisibles. Por ejemplo, caminar, girar, etc.
- Desarrollo, también en el robot AIBO, de un módulo de comunicaciones que permite al robot la comunicación con otros robots y con agentes externos al contexto del fútbol robótico, por ejemplo una computadora.
- Creación un módulo de comunicaciones para computadora que permitir la interacción

con el robot mediante el envío y recepción de mensajes que en el robot son procesados por el módulos de comunicaciones mencionado en el punto anterior

- Diseño e implementación de un modelo de entrenamiento vía voz utilizando para ello la herramienta CSLU-RAD.
- El sistema debe ser capaz de proporcionar un entorno de prueba para los tres niveles de interacción mencionados: acción, interrogación y entrenamiento.
- Las pruebas se realizarán en un campo de juego de la liga de plataforma estándar y consistirán en el entrenamiento de secuencias de comportamientos básicos y complejos utilizando un robot Sony AIBO ERS-7.

Una *acción* se refiere al hecho de ordenar al robot la ejecución de algún comportamiento predefinido, por ejemplo, caminar, girar, detenerse, etc. Una *interrogación*, se refiere a la capacidad de obtener información del robot de forma remota, es decir, el hacer posible que el robot responda a ciertas preguntas utilizando para ello sus capacidades de percepción, por ejemplo. Finalmente *entrenamiento*, significa que el robot debe contar con la capacidad de aprender comportamientos relacionando percepciones del medio con acciones.

El término aprender se refiere a la capacidad de almacenar y reproducir comportamientos independientemente de cambios en el entorno del robot, por ejemplo un cambio en la posición inicial del robot.

El uso de la voz involucra automáticamente la necesidad de contar con un mecanismo de reconocimiento del habla; en este caso, se utilizara la herramienta RAD para realizar dicha función. El sistema será únicamente capaz de reconocer palabras o frases entre un conjunto de comandos de voz definido *a priori*.

1.5 JUSTIFICACIÓN

La interacción humano – computadora y la interacción humano – robot son disciplinas enfocadas al análisis, diseño e implementación de interfaces para sistemas computacionales interactivos y robots, respectivamente, que permitan una eficiente comunicación entre humanos y máquinas [HEW 1992].

En los últimos años el aumento acelerado en la capacidad de cómputo y la miniaturización de los circuitos integrados, entre otros factores, han permitido un desarrollo importante en la robótica y áreas afines a ésta; lo anterior ha permitido la migración de los robots desde la industria hacia el área de servicios profesionales, personales y de entretenimiento [THR 2003]. Ante esta presencia cada vez más común y con la finalidad de proporcionar una comunicación más eficiente con el robot, se hizo necesario el desarrollo de nuevas interfaces de interacción más allá de las interfaces de programación y de control remoto utilizadas en la industria.

El presente trabajo intenta demostrar la viabilidad de una solución basada en el lenguaje oral que permita la interacción entre el ser humano y el robot de una manera natural. Al mismo tiempo, el desarrollo aquí presentado ofrece una primera aproximación al uso de la voz dentro del contexto del fútbol entre robots de la liga de plataforma estándar (*Standard Platform League*) de *RoboCup*.

1.6 TRABAJOS RELACIONADOS

En 2006, Weitzenfeld y Dominey publican un trabajo titulado *Cognitive Robotics: Command, Interrogaction and Teaching in Robot Coaching* [WEI 2006] donde se describía un sistema que usó la voz como medio de interacción entre el ser humano y un robot para indicar instrucciones, realizar preguntas y lograr un aprendizaje a nivel básico; la presente tesis está basada en su totalidad en dicha publicación. En *Coaching Robots to Play Soccer via Spoken-Langua*ge [WEI 2008] se presentan los primeros resultados obtenidos a partir este trabajo, esos resultados así como los obtenidos en fechas posteriores a dicha publicación son presentados en esta tesis.

En 2005, Adrián Martínez presentó la tesis *Diseño del sistema de localización para robots autónomos* [MAR 2005] en la cual, además de tratar el problema de la localización de robots AIBO ERS-210 dentro del contexto de *RoboCup*, ofrece una introducción a los sistemas de visión y de comportamientos que se heredaron al robot AIBO ERS-7.

Posteriormente, en 2008, Alonso Martínez presentó la tesis *Desarrollo de un modelo de localización para múltiples robots móviles, autónomos y colaboradores dentro de la liga Four-Legged de RoboCup* [MAR 2008] en la cual elabora un sistema de localización, basado

en métodos probabilísticas, para el robot AIBO ERS-7, también dentro del contexto de *RoboCup*.

1.7 ORGANIZACIÓN DEL DOCUMENTO

Este primer capítulo contiene una breve introducción al problema de la interacción entre humanos y robots; además, se plantean objetivos y alcances del proyecto y se incluye información relacionada con publicaciones afines al tema tratado y que soportan el trabajo.

En el capítulo dos se presenta el marco teórico necesario para la comprensión de los conceptos utilizados en el desarrollo del trabajo. El capítulo tres presenta un análisis de los requerimientos funcionales del sistema y el diseño del mismo es mostrado en el capítulo cuarto.

El capítulo cinco presenta la implementación del sistema de interacción mientras que el sexto capítulo explica las pruebas y los resultados obtenidos en las mismas. Finalmente, en el capítulo siete se presentan las conclusiones obtenidas en el desarrollo de la tesis así como las líneas futuras.

CAPÍTULO 2

MARCO TEÓRICO

En este capítulo se presentan el conjunto de conocimientos necesarios para la comprensión del sistema de interacción mediante voz que se desarrolló. Este capítulo toca temas relacionados con la interacción entre humanos y robots, la robótica situada, el procesamiento de voz y el contexto en el que se realizaron las pruebas del capitulo seis.

2.1 PROCESAMIENTO DE VOZ

El procesamiento de voz se divide en tres grandes áreas: identificación del locutor, reconocimiento del habla y producción artificial o síntesis de voz.

2.1.1 Identificación del locutor

El reconocimiento del locutor es el proceso de reconocer automáticamente la identidad de la persona que habla utilizando para ello la información de las ondas de sonido que emite [DOD 1985]. Los sistemas de autenticación de locutor típicamente están compuestos por dos módulos: el primero de ellos se encarga de extraer de una muestra de voz las características relevantes para la identificación del hablante; el segundo módulo, se encarga de comparar dichas características con las que identifican a las personas conocidas por el sistema.

El procesamiento espectral de corto tiempo es el procedimiento comúnmente seguido para la identificación del locutor; en él, fragmentos de corta duración (menores a 100ms) son utilizados como muestras para las técnicas de caracterización. Los denominados *Mel-Frequency Cepstrum Coefficients* (MFCC) son una de los métodos de caracterización más utilizados. Los MFCC utilizan filtros diferenciados de forma lineal en bajas frecuencias y diferenciados de forma logarítmica en altas frecuencias para capturar las principales características de la voz; los MFCC buscan imitar el procesamiento que el oído humano realiza en la frecuencia y ancho de banda de la señales que percibe [CAM 1997].

2.1.2 Reconocimiento del habla

El reconocimiento del habla (palabras o frases) es el proceso de identificar automáticamente la palabra o serie de palabras que un locutor emite y transformarlas en entradas entendible para una máquina. El resultado de este proceso puede ser texto o una representación de palabras que a su vez sirva para, ejemplo, desencadenar un evento.

Existen básicamente dos tipos de reconocimiento automático del habla: *Direct voice input* (DVI) y *Large vocabulary continuous speech recognition* (LVCSR). Los sistemas DVI están típicamente configurados para una vocabularios relativamente pequeños (algunos cientos de palabras) y su objetivo es reconocer las palabras o frases y poner a disposición la información reconocida para otra aplicación. Los sistemas LVCSR, por otro lado, son utilizados para la creación de documentos a partir del reconocimiento de frases extensas [DES 1999].

Una de las técnicas empleadas para el reconocimiento del habla es la comparación de patrones; este método involucra el análisis espectral de muestras de voz para cada una de las palabras que se desea reconocer. Por supuesto, el inconveniente de la gran cantidad de muestras que son necesarias para acumular un extenso vocabulario salta a la vista inmediatamente; además, para garantizar el reconocimiento, la pronunciación de cada palabra o frase tendría que ser muy aproximada a la pronunciación de las muestras de entrenamiento.

Otra técnica consiste en identificar subunidades en las palabras, típicamente fonemas. Al reconocer los fonemas que componen cada palabra, la traducción a su equivalente escrito es sencilla; de este modo, en vez de almacenar miles de palabras, se almacena el reducido número de fonemas que componen un lenguaje [JAR 2007].

La calidad de los sistemas de reconocimiento del habla se mide en términos de precisión y velocidad. La precisión es evaluada a través de una tasa de error en las palabras que se identifican. La velocidad es un factor de evaluación cuyo peso depende de la aplicación del sistema de reconocimiento; existen aplicaciones que no necesitan un procesamiento en tiempo real y por lo tanto la importancia de la velocidad como factor de evaluación disminuye.

Existen en el mercado sistemas de reconocimiento del habla del tipo LVCSR que permiten,

por ejemplo, dictar a una computadora un documento [IBM 2009] u obtener transcripciones de una conversación telefónica [NUA 2009]. También existen aplicaciones comerciales del tipo DVI por ejemplo el software RAD utilizado en el desarrollo del presente trabajo.

2.1.3 Síntesis de voz

La síntesis de voz es un proceso que permite obtener voz a partir de un texto o una representación lingüística de un texto.

Los sistemas de síntesis de voz típicamente están construidos por dos partes: la primera de ellas es un módulo encargado de la traducción de texto a una representación lingüística fonética. Este proceso se realiza en tres pasos: primero, se cambian partes problemáticas como números o abreviaturas por palabras equivalentes, esto decir, el texto es normalizado; segundo, se toma el texto normalizado y se traduce cada palabra a su equivalente fonético; finalmente, se crean unidades prosódicas, por ejemplo oraciones. El segundo módulo de los sistemas de síntesis de voz toma la representación fonética del texto y la convierten en sonidos [SCH 2003].

La complejidad en la construcción de sistemas de síntesis de voz depende, entre otros factores, del lenguaje a sintetizar; factores tales como la relación entre la ortografía y la fonética de cada lenguaje varían la complejidad de la asignación fonética al texto y por tanto del proceso de síntesis de voz [BLA 2004].

La calidad de la síntesis de voz es evaluada a partir de dos aspectos: inteligibilidad e naturalidad. El criterio de inteligibilidad evalúa que tan fácil es entender la voz sintetizada. La naturalidad se refiere a la medida en que la voz sintética es parecida a una voz humana.

Existen distintos tipos de síntesis de voz, dos de ellos son los más habituales: síntesis por concatenación y síntesis de formantes. La síntesis por concatenación utiliza segmentos de voz grabadas para producir la voz deseada [CAR 1993]. Por otro lado, la síntesis de formantes produce el sonido utilizando un modelo acústico en las que variables como la frecuencia fundamental de la onda son modificados con el objetivo de producir el conjunto de sonidos de un lenguaje; en este caso, la voz resultante no es tan natural como la obtenida de un proceso de

concatenación pero el procesamiento necesario para la síntesis es menor.

En los sistemas de síntesis de voz a partir de texto la única diferencia entre la síntesis de una palabra o de un texto es el tiempo de procesamiento.

2.2 EL ROBOT Y LA INTERACCIÓN CON SU ENTORNO

La robótica situada (Situated Robotic) es el estudio de los robots en ambientes complejos y, a veces, cambiantes [MAJ 2002]. Automóviles robóticos transitando por avenidas o robots navegando en entornos llenos de personas son ejemplos de robots situados; por otro lado, un robot en una línea de producción, donde el entorno no cambia de manera significativa, es un ejemplo de un robot no situado.

El término en inglés *situatedness* se refiere a la forma en la que el entorno interviene en el comportamiento de un robot. *Embodiment* por otro lado tiene que ver con la propiedad de poseer un cuerpo físico que interactúa con el entorno.

La relación entre un robot y su entorno definen el modo en que el robot afecta el entorno (*embodiment*) y como este influye en su comportamiento en el ambiente (*situatedness*). Esta relación, junto con la serie de tareas para las cuales es diseñado un robot, define el tipo de control que debe tener el robot.

2.2.1 Control del robot

El control del robot es el proceso de adquirir información del entorno que rodea al robot por medio de sensores y determinar con ella la acción a realizar [MAJ 2002].

Básicamente existen cuatro tipos de control robótico:

- Control reactivo: este tipo de control es descrito como el equivalente biológico estimulo-respuesta. Permite al robot actuar rápidamente ante un ambiente cambiante pero es incapaz de almacenar mucha información acerca del entorno o de tener una representación interna del mundo que lo rodea [BRO 1991].
- Control deliberativo: con este esquema, el robot toma toda la información procedente

de sus sensores y la utiliza, junto con el conocimiento almacenado que posee, para decidir la acción que debe ejecutar. Lo anterior involucra que el robot posea una representación del mundo que lo rodea para poder utilizar la información de éste en la planeación de la acción a ejecutar.

- Control hibrido: el robot posee tanto el control reactivo como el deliberativo que trabajan en colaboración con un tercer módulo intermedio que controla el sistema. Este método permite una reacción rápida que encaja con una estrategia dirigida a logar una tarea.

Control basado en comportamientos: este tipo de control pretende imitar como algunos animales se desenvuelven en ambientes complejos. Comportamientos básicos, tales como evadir un obstáculo, son la base de este tipo de sistema. Comportamientos más complejos son agregados de manera incremental hasta lograr que el robot realice tareas complejas.

2.2.2 Autómata situado

Todo sistema inteligente que opere en un moderadamente complejo o impredecible ambiente debe ser reactivo, esto es, debe responder dinámicamente a cambios en su ambiente [KAE 1986].

En [ROS 1995], Rosenschein y Kaelbling definen una arquitectura robótica basada en el concepto de autómata situado. Un autómata situado es esencialmente una máquina de estados finitos cuyas entradas son proveídas por los sensores y las salidas están conectadas con los actuadores [STU 2002].

Bajo la arquitectura propuesta por Rosenschein y Kaelbling, un robot ejecuta en el bajo nivel un comportamiento desencadenado por su percepción del ambiente; al mismo tiempo, el comportamiento a ejecutar esta determinado por un estado interno que forma parte de un autómata de estados finitos.

La arquitectura basada en autómatas situados permite la construcción de nuevos comportamientos complejos a partir de la integración otros comportamientos definidos con anterioridad.

2.2.3 Relación percepción-respuesta

Los objetivos del presente trabajo involucran el desarrollo de un mecanismo que le permita al robot aprender la relación existente entre una percepción y la respuesta que debe desencadenarse con ella.

Tres son los componentes que participan en dichas construcciones percepción- respuesta [WEI 2006]:

- la percepción: un comando verbal o un evento captado por el sistema de visión u otro sensor
- la respuesta a esta percepción: una respuesta verbal o una respuesta motriz del repertorio de comportamiento,
- la construcción de la pareja percepción-respuesta: el sistema debe relacionar y almacenar la pareja percepción-respuesta con el fin de que pueda ser utilizado en el futuro.

Así, uno de los objetivos que busca el trabajo aquí presentado es el desarrollo de un método capaz de proveer al robot de la capacidad para ligar percepciones y respuestas de modo que los comportamientos derivados de dichas relaciones sean reproducibles.

2.3 INTERACCIÓN HUMANO – ROBOT

La interacción entre humanos y computadoras (Human Computer Interaction – HCI) es una disciplina que involucra el diseño, evaluación e implementación de sistemas de cómputo interactivos para el uso humano así como los fenómenos alrededor de estos [HEW 1992].

En la misma lógica, la interacción entre humanos y robots (Human Robot Interaction – HRI) estudia la forma en que la gente se comunica con los robots y como hacer dicha comunicación eficiente.

Los robots pueden clasificarse según su aplicación en tres grandes categorías: robots industriales, robots de servicio profesional y robots de servicio personal. Cada uno de ellos

involucra distintos tipos de interacción con el hombre [THR 2003].

Los robots industriales son máquinas controladas por una computadora que son capaces de manipular físicamente el ambiente en el que operan. Las funciones clásicas de los robots en la industria incluyen el ensamblaje, transportación y manejo de materiales. La industria automotriz utiliza ampliamente los robots industriales en sus líneas de producción. La investigación de interfaces de interacción entre humanos y este tipo de robots se enfocó en el desarrollo de técnicas para la rápida configuración y programación de los robots [THR 2003].

Los robots de servicio profesional pueden manipular y navegar en ambientes físicos. Las funciones que estos robots desempeñan incluyen la exploración marítima, manipulación de químicos, aplicaciones militares y otras actividades que resultan riesgosas para el ser humano. Este tipo de robots interactúan con personas y comúnmente comparten el ambiente con ellas.

Finalmente, los robots de servicio personal están diseñados para auxiliar al hombre en tareas domesticas o entretenerlos. El espacio físico en que este tipo de robots se desenvuelve involucra una constante interacción con personas.

Los robots de servicio pueden tener dos tipos de interfaces de interacción: indirecta y directa [THR 2003]. Las interfaces indirectas de interacción permiten a un ser humano operar el robot, en ellas el flujo de información es unidireccional. Por otro lado, las interfaces directas de interacción permiten una comunicación bidireccional con el robot; un ejemplo de este tipo de interfaces son los robots que hablan y que son capaces de entender un vocabulario especifico que les permite una comunicación con las personas en su entorno [ASO 1997].

Otra clasificación de interfaces de usuario para la interacción con robots es proporcionada por Bartneck y Okada en [BAR 2001]. En ella, las interfaces se clasifican según cuatro criterios:

- Juguete – Herramienta (Toy – Tool Scale): la interfaz permite controlar al robot como un juguete cuya función es el entretenimiento o como una herramienta donde la interfaz permite realizar una tarea específica.
- Controlado Remotamente – Autónomo (Remote Control – Autonomous Scale): el robot es controlado remotamente mediante la interfaz o es capaz de actuar de manera autónoma.

- Reactivo – Dialogante (Reactive – Dialogue Scale): el robot responde de forma reactiva al uso de la interfaz bajo un esquema ajustado o es capaz de realizar una conversación y entender las instrucciones.
- Antropomorfismo (Anthropomorphism Scale): el robot posee características parecidas a las humanas y las utiliza para comunicarse con el usuario.

El diseño de las interfaces de interacción entre humanos y robots depende específicamente de la función que el robot desempeña en el ambiente y el grado de interacción con personas que se presentará.

En el presente trabajo se muestra parte de la investigación enfocada en arquitectura de interacción humano-robot que permitirá a entrenadores humanos intervenir en un juego de fútbol con robots a través de la lengua hablada. El presente proyecto forma parte de un conjunto de investigaciones relacionadas con el desarrollo de capacidades cognitivas en robótica. Dicha investigación consta de dos partes: la primera de ellas está enfocada al desarrollo de algoritmos que permitan aprender la relación existente entre una oración y su significado. La segunda parte está orientada a la adquisición de conocimientos basados en percepciones del medio que rodea al robot. [WEI 2008]

El trabajo aquí descrito está directamente enfocado al segundo punto, es decir, se orienta exclusivamente a analizar la factibilidad del uso del lenguaje hablado como método para proporcionar conocimiento basado en percepciones a un robot.

2.3.1 Niveles de interacción

Se definieron, para efectos de la presente tesis, tres niveles de interacción entre el humano y el robot [WEI 2006]:

- Acción: en este nivel el usuario se limita a ordenar la ejecución de acciones por parte del robot.
- Interrogación: en este segundo nivel, el usuario puede realizar preguntas al robot acerca de algún sensor o estado interno o pedir explicaciones acerca de algún comportamiento.

- Entrenamiento: se refiere a la posibilidad de enseñar al robot como construir nuevos comportamientos basados en la creación de relaciones entre percepciones y respuestas; los entrenamientos pueden ser:
    - Simples: construidos a partir de comandos de acciones básicas y percepciones acerca del medio que rodea al robot.
    - Complejos: basado en comportamientos aprendidos previamente.

2.3.2 Interacción mediante una interfaz basada en voz

Para que una orden de voz pueda desencadenar una acción por parte de un robot, un proceso complejo debe desarrollarse. Primero, se debe contar con un proceso de reconocimiento del habla, es decir, poseer la capacidad de identificar palabras y oraciones complejas; segundo, se debe poseer un método para descifrar el significado de la orden, es decir, un algoritmo capaz de "mapear" la orden a una serie de instrucciones para el robot. Tercero, una vez que se tiene conocimiento del significado de la orden, las acciones pueden ser ejecutadas por el robot. [COO 2002].

El primer paso del proceso descrito ha sido estudiado durante muchos años y existen una gran fuente de conocimiento relacionada con el reconocimiento del habla, algunas referencias a estas líneas de investigación han sido tocadas en la sección 2.1 de este capítulo.

Existen numerosos trabajos relacionados con la tarea de extraer significado a partir de oraciones u otros medios, videos por ejemplo. En [DO1 2005] y [DO2 2005], por ejemplo, Dominey y Boucher describen un sistema que puede adquirir una gramática limitada utilizando para ello un entrenamiento con vídeo narrado de eventos. Un algoritmo de procesamiento de imágenes extrae el significado de los acontecimientos narrados traduciéndolos en descriptores de acciones, detectando los contactos físicos entre los objetos. El sistema de procesamiento visual de la escena se basa en la caracterización física de eventos complejos (por ejemplo, colocar, tomar) en términos de la composición física de eventos básicos como el contacto.

Figura 2.1 Sistema de percepción de eventos desarrollado por Dominey.

Cada caso narrado genera un par oración-significado que se utiliza como entrada a un modelo que aprende la oración y la asignación de significado.

2.4 AMBIENTE DE PRUEBAS ROBOCUP

RoboCup es un proyecto internacional para promover el desarrollo de la Inteligencia Artificial (AI, por sus siglas en inglés), la robótica y los campos relacionados [RO1 2008].

RoboCup está dividida en cuatro áreas, en cada una de ellas se enfrentan problemas distintos:

- *RoboCup Junior*, encargada del fomento de la robótica entre estudiantes de bachillerato.
- *RoboCup@Home*, liga centrada en aplicaciones de robótica en entornos realistas, en este caso una casa habitación.
- *RoboCup Rescue*, destinada al fomento de la investigación que promueva el desarrollo de agentes que actúen ante situaciones de desastre.
- *RoboCup Soccer*, liga donde se eligió el fútbol como el tema central de la investigación y cuyo el objetivo final es desarrollar, para el año 2050, un equipo totalmente autónomo de robots humanoides que puedan competir contra el equipo campeón del mundo de ese año y que pueda ganarle en un encuentro.

RoboCup Soccer, se compone de cinco categorías, cada una de ellas provee a los equipos de una gama diferente de problemas a abordar. Las categorías son:

- **Simulation League:** En la liga de simulación se enfrentan dos equipos de once agentes virtuales. Un sistema provee una simulación realista de los agentes con sensores y acciones. Cada agente se comunica con un servidor central para proporcionarle instrucciones sobre los movimientos a realizar y para recibir información acerca del estado del juego.
- **Small Size League:** En la liga de robots pequeños se utilizan robots, comúnmente cilíndricos, de no más de 180mm de diámetro y 150mm de altura. Cada equipo está integrado por cinco robots. La característica principal de esta liga es que se cuenta con una serie de cámaras montadas a cuatro metros de altura sobre la cancha que proporcionan una visión global de lo que ocurre en ella. La información recolectada por las cámaras es procesada en una computadora donde se decide que movimientos se deben realizar. Las instrucciones, finalmente, son enviadas a los robots de modo inalámbrico.
- **Middle Size League:** En la liga de robots medianos está permitido jugar hasta con 6 jugadores por equipo. Cada uno de estos robots en un sistema totalmente autónomo con visión local. Los robots deben ser construidos por cada equipo. El tamaño de la pelota utilizada en esta liga es el mismo que la utilizada en competencias de fútbol entre humanos.
- **Humanoid League:** En la liga de humanoides, cada equipo se compone por dos elementos. Cada robot debe tener una forma similar a un humano, esto es, debe ser un bípedo. En esta categoría se permite construir el robot o comprar una plataforma base y modificar el hardware.
- **Standard Platform League:** La liga de plataforma estándar contaba, hasta 2008, con dos categorías: bípedos (*2-Legged*) y cuadrúpedos (*4-Legged*). En ambas categorías, robots fabricados por un proveedor ajeno a *RoboCup* son usados. En la categoría de bípedos, el robot Aldebaran Nao comenzó a usarse en la edición del año 2008 de *RoboCup*, mientras que en los cuadrúpedos, el robot Sony AIBO ha sido usado desde los inicios de la liga. El objetivo en esta liga se centra exclusivamente en la programación del robot; todos los equipos compiten con una plataforma robótica idéntica y la diferenciación se da en el ámbito del software [RO1 2008].

2.4.1 Standard Platform League (4-Legged)

En la liga de cuadrúpedos se juega sobre una cancha de 4.6 metros de ancho por 6.9 metros de largo en color verde sobre la que se trazan líneas en color blanco para delimitar las secciones del campo, en cada extremo del campo de juego se cuenta con una portería, tal como se muestra en la figura 2.2.

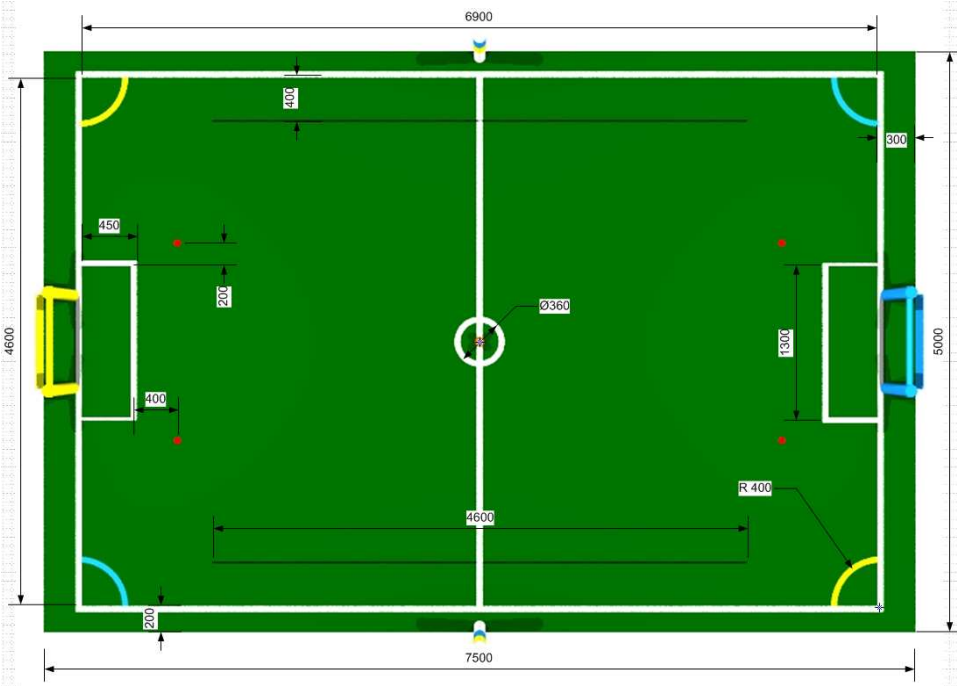


Figura 2.2 Campo de juego de la liga de cuadrúpedos de SPL.

En un encuentro de la liga de cuadrúpedos, cada equipo está compuesto por cuatro robots identificados por un uniforme en color rojo o azul. Cada uno de estos equipos defiende una portería también identificada por el color: el equipo rojo defiende la portería amarilla, mientras que el equipo en azul defiende la portería en color cian. Para ayudar la tarea de localización del robot dentro del campo de juego, se cuenta con un par de postes colocados a la mitad de la cancha, ambos postes están pintados con franjas en color amarillo y cian en distinto orden para distinguir una lado del otro.

Se juega con una pelota de color naranja y el objetivo de cada equipo es anotar la mayor cantidad de goles en la portería contraria. El juego se desarrolla en dos segmentos de 10 minutos cada uno separados por una pausa de igual duración en la que es permitida la modificación del software de competencia.

2.5 ROBOT SONY AIBO ERS-7

El AIBO es un robot cuadrúpedo inspirado en la anatomía de un perro que posee 20 grados de libertad de movimiento distribuidos de la siguiente forma:

- Tres grados de libertad en cada pata para un total de 12.
- Tres grados en la cabeza
- Dos grados en la cola
- Un grado en cada oreja para un total de 2 grados.
- Un grado en la boca.

El robot mide 317.38mm de largo y 180.15mm de ancho, tiene una altitud variable de acuerdo con la posición de las patas. En la configuración mostrada en la figura 2.4 su altura es de 278.23mm. El robot posee una tarjeta de red inalámbrica que cumple con la especificación IEEE 802.11b/g. Además, el robot está equipado procesador MIPS de 64 bits a 576MHz y con 64MB en memoria RAM.

Figura 2.3 Robot Sony AIBO ERS-7.

2.5.1 OPEN-R

Para poder programar el robot AIBO, Sony publico un SDK (Software Development Kit) conocida como OPEN-R. Dicho SDK contiene un conjunto de librerías que proporcionan al programador acceso total a cada uno de los sensores y actuadores con que cuenta el robot. De este modo, se proporciona a los usuarios la capacidad de controlar totalmente el

comportamiento del robot así como de utilizar la información percibida por el mismo en el desarrollo de aplicaciones diversas.

La forma de programación del robot es orientada objetos y totalmente modular, cada módulo es conocido como objeto OPEN-R. y es un hilo de ejecución o *thread*; así, cada uno de los objetos puede encargarse de realizar una operación específica sin interferir en la ejecución del resto de los objetos.

La comunicación entre objetos se da por medio de interfaces declaradas de manera externa a ellos, de este modo, cada objeto OPEN-R tiene la capacidad de funcionar de manera independiente o en interacción con otros objetos.

El ciclo de vida de un objeto OPEN-R tiene cinco etapas (figura 2.4):

- DO_INIT: el objeto es creado y se le asigna tiempo de ejecución en el procesador. Esta etapa se ejecuta cuando el robot es encendido.
- DO_START: el objeto inicia las comunicaciones con otros objetos y se inicializan variables de acceso a actuadotes.
- Ejecución: el objeto realiza la función específica para la que fue desarrollado, por ejemplo el procesamiento de imágenes.
- DO_STOP: el objeto cierra las comunicaciones con el resto de los objetos.
- DO_DESTROY: el objeto es retirado de la memoria del robot y posteriormente destruido. Comúnmente esta etapa se ejecuta cuando el robot es apagado.

OPEN-R requiere de un ambiente UNIX para su compilación. Si se utiliza Windows como plataforma es necesario el uso de una herramienta auxiliar para simular dicho ambiente, una de las más comunes es Cygwin.

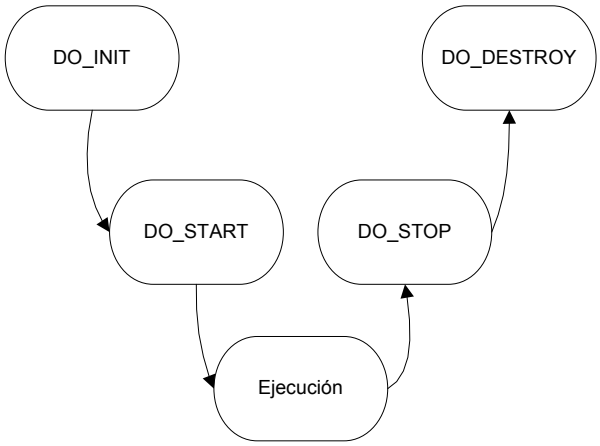


Figura 2.4 Ciclo de vida de un objeto OPEN-R.

2.6 EL EQUIPO DE ROBÓTICA DEL ITAM

El Laboratorio de Robótica del ITAM fue creado para proporcionar un espacio donde los alumnos pudieran poner en práctica sus conocimientos en áreas como la electrónica, la programación y otras relacionadas con la construcción y programación de robots autónomos móviles. Tras adquirir experiencia en las competencias nacionales, se decidió realizar proyectos de mayor complejidad, así como nació la idea de incorporarse a la *Small Size League* de *RoboCup* y posteriormente a la entonces llamada *4-Legged League*, hoy conocida como *Standard Platform League*.

El equipo del laboratorio de Robótica, denominado *Eagle Knights*, ha participado en ambas categorías en las últimas cuatro ediciones del campeonato mundial de *RoboCup* (Osaka 2005, Bremen 2006, Atlanta 2007 y Suzhou 2008), obteniendo destacadas participaciones y mejorando los resultados año con año.

2.6.1 Arquitectura del sistema para la Standard Platform League

La categoría *Standard Platform League* fue introducida en el Laboratorio de Robótica en el año 2005 con el objetivo de participar en el campeonato mundial de *RoboCup* que se realizó ese mismo año en la ciudad de Osaka, Japón. En aquella edición se participó utilizando como plataforma el robot Sony AIBO en su versión ERS-210. En las ediciones posteriores (Alemania 2006, Estados Unidos 2007 y China 2008) se utilizo la última versión del robot

AIBO, la ERS-7.

El robot proporciona una serie de sensores, una cámara incluida entre ellos, y una extensa serie de actuadotes, básicamente motores. La arquitectura del sistema desarrollado para esta liga se muestra en la figura 2.5.

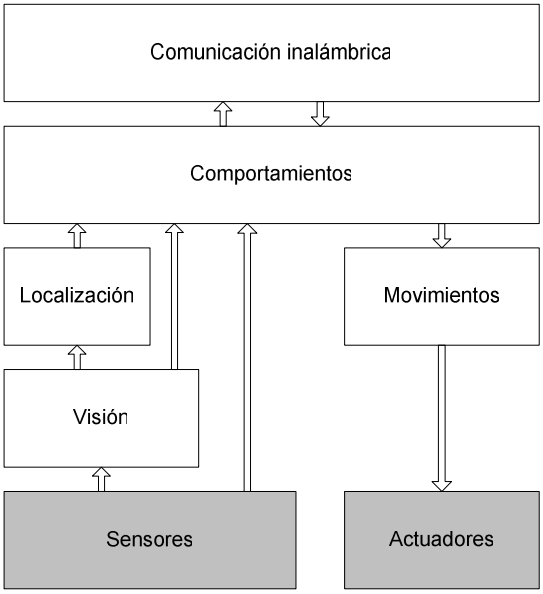


Figura 2.5 Arquitectura del sistema para la *Standard Platform League*.

La funcionalidad que proporciona cada bloque del diagrama es la siguiente:

- Sensores: recupera información de los sensores físicos. Son particularmente interesantes la cámara y la información relativa a la posición de cada motor.
- Visión: recibe una imagen directamente desde la cámara, nuestro sensor principal, y realiza un proceso de segmentación sobre ella. Este módulo es capaz de detectar diversos objetos relacionados con el campo de juego, por ejemplo, la pelota, las porterías, postes auxiliares u otros robots, ya sean del mismo equipo o del contrario.
- Movimiento: proporciona el control de movimientos del robots, por ejemplo caminar, girar, mover la cabeza, etc. Recibe comandos desde el módulo de comportamientos y los traduce en instrucciones individuales para cada uno de los motores involucrados con el movimiento.

- Actuadores: recibe las instrucciones de movimiento para cada motor o de acción para el resto de los actuadores, por ejemplo los LEDs.
- Comportamientos: toma decisiones de alto nivel relativas al comportamiento que se debe realizar; para ello, se toma la información directamente de los sensores y de los módulos de localización y visión para generar instrucciones enviadas al módulo de movimiento.
- Comunicación inalámbrica: se encarga de la comunicación entre robots y con el *Game Controller*, sistema externo que se encarga de cumplir las funciones de árbitro durante un partido. El módulo proporciona la capacidad para compartir información relevante como la posición del robot y de la pelota.
- Localización: realiza los cálculos necesarios para obtener una estimación de la posición del robot en el campo de juego. Para calcular la posición utiliza la información de los objetos detectados por el módulo de visión.

Cada módulo descrito es implementado como un objeto OPEN-R y se establecen los medios de comunicación entre ellos. La figura 2.6 muestra el diagrama del sistema implementado, a cada módulo descrito corresponde un objeto OPEN-R independiente del resto pero comunicado con otros objetos cuando el intercambio de información es necesario. Así, EKVision implementa el módulo Visión, EKMotion corresponde a Movimientos, EKLoc es la implementación de Localización, EKMain se encarga de Comportamientos y EKComm implementa la Comunicación Inalámbrica.

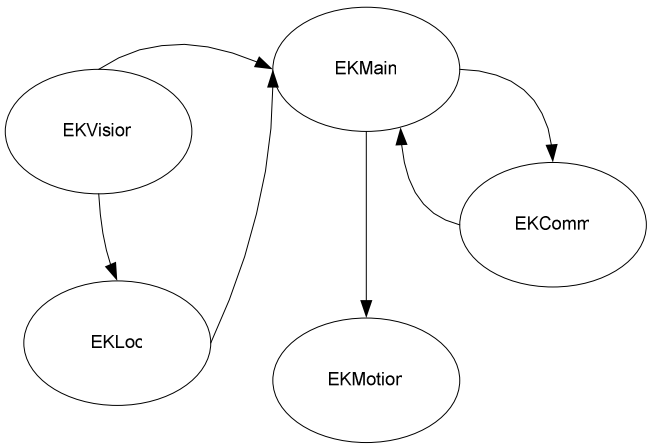


Figura 2.6 Sistema EK2007 para la *Standard Platform League*.

2.7 CSLU SPEECH TOOLKIT

El CSLU Speech Toolkit es un conjunto de herramientas desarrolladas por el Center for Spoken Language Understanding (CSLU) dependiente de la Oregon Health & Science University [CSL 2002].

Este paquete ofrece un conjunto de aplicaciones que combinan herramientas de audio y de video, reconocimiento del habla (figura 2.7), síntesis de voz y animación de caras para proveer a los usuarios la capacidad de desarrollar medios de interacción para propósitos diversos. CSLU Speech Toolkit es usado para objetivos educacionales, industriales y de investigación.

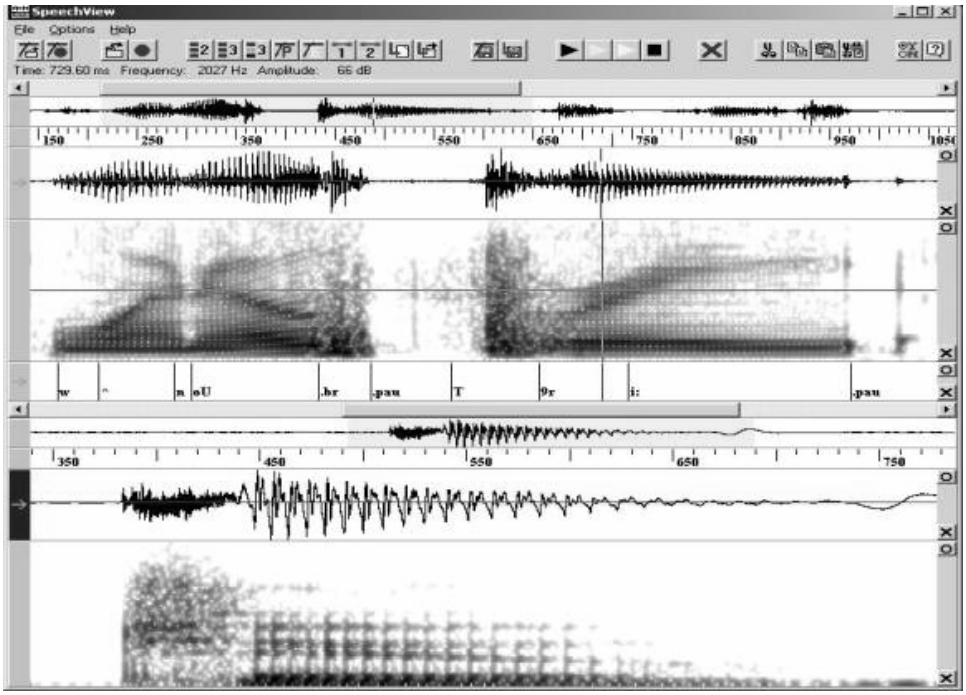


Figura 2.7 Herramienta de reconocimiento de voz.

La principal herramienta del *kit* es la aplicación *Rapid Application Developer* (*RAD*). Esta aplicación está enfocada al desarrollo de interfaces gráficas, llamadas diálogos, dotadas de capacidad de reconocimiento del habla, síntesis de voz y toma de decisiones. Además provee la capacidad de programación de scripts mediante el lenguaje TCL. La figura 2.8 muestra la pantalla principal de la aplicación, en ella se observar una barra de herramientas al dado izquierdo y una cara animada al costado derecho.

En la barra de tareas lateral se encuentra una serie de objetos que permiten incorporar diversas funcionalidades a cada diálogo. Nueve de estos objetos representan el núcleo de la aplicación. Sus iconos son mostrados en la figura 2.9 y sus funciones son descritas a continuación.

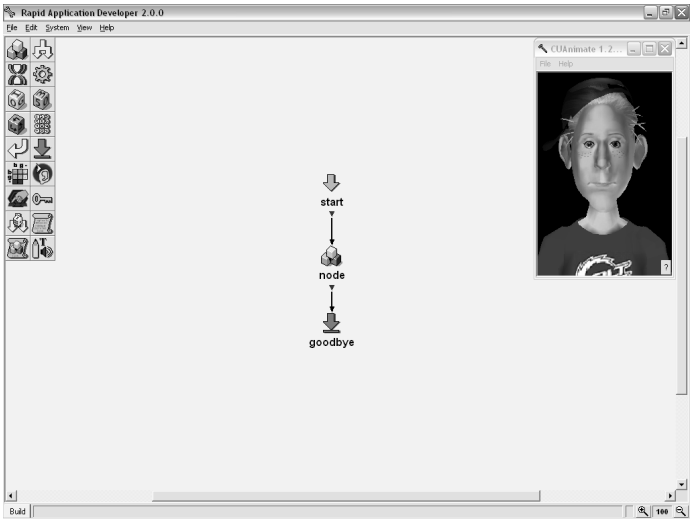


Figura 2.8 Rapid Application Developer.

Figura 2.9 Objetos base de RAD.

- *Action*: proporciona un objeto en el cual se evalúa código TCL /TK. Este espacio puede ser útil para la asignación de valores a variables utilizadas en el diálogo o para realizar cálculos.
- *Alpha-Digit*: este es un objeto que provee la capacidad de reconocimiento de letras y dígitos, útil para reconocer oraciones.
- *Conditional*: objeto que sirve para evaluar expresiones del lenguaje TCL y así poder crear derivaciones en el flujo basadas en los distintos resultados de las expresiones. Se trata de un nodo de control de flujo que ofrece varios puertos de salida.
- *Digit*: Objeto que permite el reconocimiento de números.

- *Enter*: Este objeto permite marca el punto en que comienza la ejecución de un subdiálogo.
- *Exit*: marca el punto de fin un sub-diálogo y, por tanto, el punto donde el diálogo principal toma el control del flujo.
- *Generic*: provee de capacidades de reconocimiento alfanumérico; además, es capaz de realizar síntesis de voz. Cuenta también con la capacidad de ejecutar código TCL /TK ya sea al comienzo o al final de su ejecución.
- *Goodbye*: este objeto indica el final de un diálogo.
- *Subdialogue*: Este objeto permite crear sub-diálogos con el objetivo de organizar de una mejor manera el flujo dentro del diálogo principal.

RAD permite con estos nodos crear complejos diálogos en los que el reconocimiento de palabras o frases constituyen bloques de control del flujo y a su vez desencadenan la ejecución de código TCL / TK. Un ejemplo de un diálogo en que el flujo es desviado por diversos factores es mostrado en la figura 2.10.

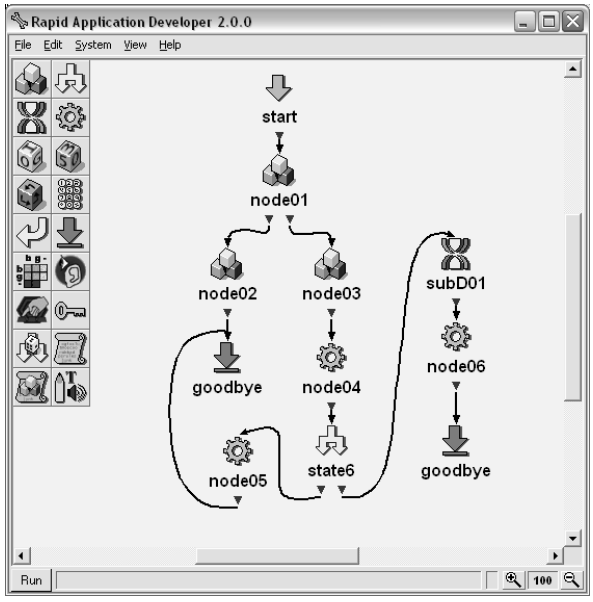


Figura 2.10 Diálogo ejemplo del RAD.

Para diálogos de poca complejidad como los desarrollados para la presente tesis, es suficiente el uso de los objetos genéricos (*Generic*), de acción (*Action*), condicionales (Condicional) y los relacionados con la implementación de subdiálogos.

CAPÍTULO 3

REQUERIMIENTOS DEL SISTEMA

El presente capítulo presenta los requerimientos de hardware y software del robot a utilizar así como los requerimientos generales del sistema y los particulares de cada una de las funciones que debe ofrecer al usuario. Además, se incluyen los casos de uso para dichas funciones.

3.1 SISTEMA DE INTERACCIÓN HUMANO-ROBOT

El sistema desarrollado en la presente tesis debe permitir a un usuario humano interactuar con un robot mediante comandos voz. El sistema es un módulo de enlace entre el usuario y el robot cuya función principal es reconocer instrucciones o interrogaciones del usuario y traducirlas a órdenes o consultas para el robot, respectivamente. Además, el sistema debe poder captar las respuestas del robot y presentarlas al usuario en forma oral. La figura 3.1 es un diagrama a bloques en el que se ilustra el lugar que ocupa el sistema de interacción.

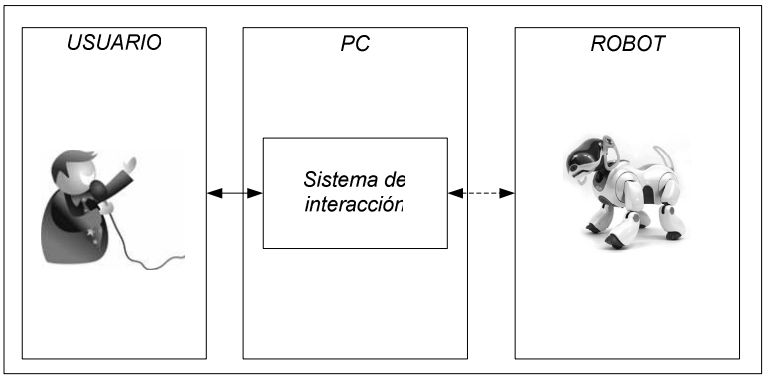


Figura 3.1. Sistema de interacción Humano-Robot.

Cabe destacar que si bien la mayor parte del sistema de interacción corre en una computadora, al menos un módulo de conexión debe ser desarrollado en el robot para proveer comunicaciones entre dicho agente y el sistema.

3.1.1 Robot

A continuación se enlistan las características de hardware y software que son necesarias para el desarrollo del presente trabajo. Un análisis de ellas nos llevó a la elección de robot Sony AIBO en su modelo ERS-7 como la mejor alternativa para esta tesis, entre otras cosas por el desarrollo previo de software que se ha hecho en el contexto de las competencias de *RoboCup*.

3.1.1.1 Hardware

Los requerimientos del robot se dividen en cuatro categorías: movilidad, sensores, capacidades de comunicación y procesamiento.

- Movilidad: se requiere de un robot que cuente con medios para poder desplazarse sobre el suelo en todas las direcciones posibles. Como se hace mención en el capítulo dos del presente trabajo, el robot AIBO cuenta con cuatro patas, cada una con tres grados de libertad.
- Sensores: se quiere además que el robot cuente con un conjunto de sensores que le permitan obtener información del entorno que lo rodea. Uno de los más importantes sensores con que debe contar es una cámara que permita obtener, cuadro a cuadro, imágenes del medio que rodea al robot. El robot AIBO cubre perfectamente este requerimiento ya que además de contar con una cámara, cuenta con sensores infrarrojos y de presión.
- Comunicación: se requiere que el robot cuente con capacidades de comunicación con otros dispositivos electrónicos. El AIBO cuenta con una tarjeta de red que cumple con el estándar IEEE 802.11b/g que permite comunicación inalámbrica con otros dispositivos compatibles con el mismo estándar.
- Procesamiento: se necesita que el robot cuente con un poder de procesamiento que le permita realizar procesamiento de imágenes al tiempo que controla su movimiento y mantiene un monitoreo de las comunicaciones. El robot AIBO cuenta con un procesador a 575MHz y con 64MB en memoria RAM, suficientes para cubrir este punto.

3.1.1.2 Software

En cuanto al software necesario para el robot se requiere:

- Visión: un mecanismo que permita el procesamiento de la imagen captada por la cámara con el objetivo de reconocer objetos por medio del color de los mismos. Dentro del proyecto *RoboCup*, el laboratorio de Robótica cuenta ya con un módulo de procesamiento de visión, denominado EKVision, que cumple con esta función.
- Movilidad: contar con un motor de movimientos que funcione como un intermediario entre los motores del robot y la programación en alto nivel. Al igual que en el punto anterior, existe ya un motor de movimientos llamado EKMotion que realiza esta función.
- Comunicación: desarrollar un mecanismo de monitoreo de comunicaciones que permita responder rápidamente al arribo de un mensaje desde otros dispositivos. El robot Sony AIBO ofrece la capacidad de comunicación inalámbrica; al inicio del presente trabajo existía ya una implementación en el robot que permite la comunicación entre robots con el objetivo de compartir información acerca del juego. Para no modificar este módulo, denominado EKComm, se decidió la implementación de un nuevo módulo, llamado MetaComm, con el objetivo de atender únicamente a las comunicaciones concernientes al proyecto aquí expuesto.
- Procesamiento: contar con un mecanismo que centralice la toma de decisiones basadas en la información proveniente de los sensores y las comunicaciones. Cada decisión debe ser convertida en un comportamiento de alto nivel que desencadenara en una acción de los actuadotes del robot. El módulo EKMain, desarrollado con anterioridad, ofrece el mecanismo ideal para el cumplimiento de este requisito.

3.1.2 Interacción Humano-Computadora

El sistema desarrollado en el presente trabajo debe cumplir con cuatro requerimientos fundamentales los cuales son descritos a continuación:

- El sistema debe ser capaz de traducir una instrucción verbal, de un conjunto definido *a priori*, en una acción motriz por parte del robot.
- El sistema debe poder traducir una pregunta, también de un limitado conjunto de ellas establecido previamente, y expresada de forma oral la respuesta que el robot emita tras

recibir la pregunta.

- El sistema debe permitir, de forma oral, enseñar al robot comportamientos básicos; estos comportamientos deben estar basados en una combinación de acciones y percepciones del entorno que rodea al robot.
- El sistema debe permitir enseñar al robot comportamientos complejos basados en la combinación de percepciones, acciones simples y comportamientos previamente aprendidos por el robot.

El término “enseñar” implica que el sistema tiene que ser capaz de reproducir en forma íntegra los comportamientos; para ello, cada comportamiento que se enseña al robot debe ser almacenado por el sistema y ponerse a disposición del usuario para su uso posterior.

3.1.3 Comunicación entre la computadora y el robot

El sistema de comunicaciones entre el sistema y el robot cubre los siguientes requerimientos:

- El sistema debe permitir la comunicación inalámbrica entre la computadora y el robot
- La comunicación debe ser bidireccional para permitir al sistema ordenar acciones y preguntar sobre percepciones y al mismo tiempo permitir al robot responder a estas últimas.
- El sistema de comunicaciones debe permitir el envío y recepción de múltiples paquetes de información por segundos.

3.2 ACCIÓN, INTERROGACIÓN Y ENTRENAMIENTO

El sistema cuenta solamente con dos actores, es decir, dos entidades que aportan o reciben información del sistema. Cabe mencionar que uno de estos actores no es humano.

- **Usuario:** Se trata de la persona que opera el sistema. Se encarga de iniciar el sistema, terminar la ejecución del mismo y, por supuesto, del uso de todas las capacidades con que cuenta el sistema mismo.
- **Robot:** El robot es también un actor ya que en los procesos de pregunta y entrenamiento, de él se obtiene información relativa a los sensores con los que cuenta.

3.2.1 Acción

Se definió un conjunto de acciones básicas que el robot tendría que realizar; éstas consisten en movimientos básicos relacionados con el fútbol y representan la base de conocimientos sobre la que se desarrollarán los entrenamientos de comportamientos. Los comportamientos básicos requeridos son:

- Detenerse: el robot detiene cualquier acción que este realizando.
- Caminar: el robot debe caminar hacia enfrente.
- Giro a la derecha: el robot debe rotar hacia la derecha sobre su eje.
- Giro a la izquierda: el robot debe rotar hacia la izquierda sobre su eje.
- Tomar la pelota: el robot debe sujetar la pelota con sus patas delanteras.
- Girar con pelota a la derecha: el robot, sosteniendo la pelota, debe rotar hacia la derecha.
- Girar con pelota a la izquierda: sosteniendo la pelota, el robot debe rotar hacia la izquierda.
- Ir a la pelota: el robot debe acercarse a la pelota.
- Bloquear: el robot debe extender sus patas delanteras lateralmente buscando bloquear el movimiento de la pelota.

Se busca que el usuario pueda tener una interacción como en el diálogo mostrado en la tabla 3.1, en ella es posible observar como el sistema pregunta al usuario que debe hacer, el usuario puede escoger entre el conjunto predefinido de acciones que puede ejecutar el robot. Cuando el usuario escoge una acción, el sistema debe transformar la instrucción verbal en una orden para enviar el robot y éste debe ejecutar el movimiento correspondiente.

Actor	Diálogo
Sistema	¿Qué quieres que haga?
Usuario	Gira a la derecha.
Sistema	¿Qué quieres que haga?
Usuario	Ve hacia la pelota.
Sistema	¿Qué quieres que haga?
Usuario	Toma la pelota.

Tabla 3.1 Conversación deseada para acciones simples.

3.2.1.1 Caso de uso

Este caso de uso se ejecuta cuando un usuario desea que el robot ejecute alguna acción. La acción forma parte de un conjunto definido *a priori*.

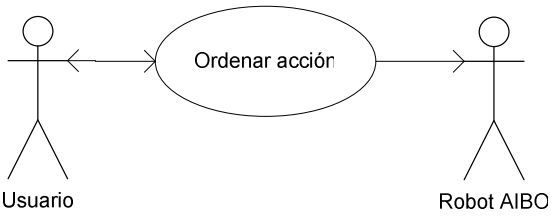


Figura 3.2 Caso de uso: Ordenar acción.

Caso de uso	Ordenar acción
Actores	Usuario, Robot AIBO
Precondiciones	Iniciar el *diálogo de interacción básica* en la aplicación CSLU – RAD
Flujo principal	[1]El caso de uso inicia cuando en usuario indica al diálogo que desea seleccionar una acción para que el robot la ejecute. [2]El flujo continúa con el diálogo solicitando indique la acción o comportamiento a realizar. [3]El usuario debe seleccionar una opción válida. Si no lo hace se ejecuta el subflujo S-1. [4]Una vez seleccionada una acción. El diálogo ejecuta el módulo HRI pasando como parámetro un identificador de la acción a realizar. [5]El módulo HRI crea un paquete UDP con la instrucción a ejecutar y la envía al robot. [6]El robot recibe el mensaje y ejecuta la acción. [7]Finalmente, el diálogo pregunta al usuario que tarea desea realizar ahora. Si decide indicar otra acción se regresa al punto 2; si no es así, el flujo termina.
Subflujos	S-1 El diálogo vuelva a preguntar al usuario que acción desea que el robot ejecute. Se continua con el diálogo principal en el punto [3]

3.2.2 Interrogación

Se requirió que el robot fuera capaz de responder a preguntas relacionadas con el entorno percibido; en particular, sobre las marcas de colores involucradas en un juego de fútbol de *RoboCup*. Se requiere que el robot sea capaz de:

- Reconocer la pelota.
- Identificar cuando la pelota está al alcance del robot.
- Reconocer la portería de color cian.
- Reconocer la portería de color amarillo.
- Identificar un robot con uniforme color azul.
- Identificar un robot con uniforme color rojo.

La tabla 3.2 muestra la conversación deseada para el sistema de interacción cuando se interroga al robot. La idea es que el usuario pueda preguntar acerca de ciertos objetos relacionados con el contexto de *RoboCup*, por ejemplo, la pelota o la portería.

Actor	Diálogo
Sistema	¿Qué quieres que haga?
Usuario	Responde a la pregunta. ¿Ves la pelota?
Sistema	Sí, veo la pelota.
Sistema	¿Qué quieres que haga?
Usuario	Responde a la pregunta ¿Ves la portería amarilla?
Sistema	No, no velo la portería amarilla.
Sistema	¿Qué quieres que haga?
Usuario	Responde a la pregunta. ¿Ves al robot rojo?
Sistema	Si, veo al robot rojo.

Tabla 3.2 Conversación deseada para interrogaciones.

3.2.2.1 Caso de uso

En este caso de uso, el usuario realiza una pregunta, también de un conjunto definido a priori. El sistema envía la pregunta al robot, él devuelve una respuesta correspondiente lo que sus sensores indican y, finalmente, el diálogo principal expresa mediante voz sintética la respuesta a la pregunta.

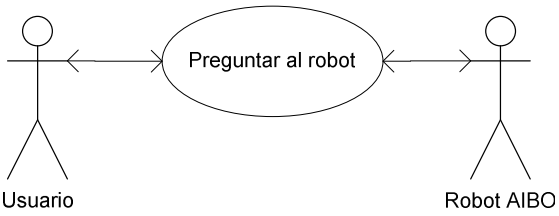


Figura 3.3 Caso de uso: Preguntar al robot.

Caso de uso	Preguntar al robot
Actores	Usuario, Robot AIBO.
Precondiciones	Iniciar el *diálogo de interacción básica* en la aplicación CSLU – RAD
Flujo principal	[1]El caso de uso inicia cuando en usuario indica al diálogo que desea seleccionar una pregunta para que el robot responda. [2] El diálogo indica al usuario que realice la pregunta [3]El usuario debe seleccionar una opción válida. Si no lo hace se ejecuta el subflujo S-1. [3]El diálogo invoca el módulo HRI pasando como parámetro un identificador de la pregunta que se realizó. [4]El módulo HRI crea un paquete UDP con la pregunta y la envía al robot. [5]El robot recibe el mensaje, verifica la información de sus sensores y envía de regreso un paquete con la respuesta. [6]El paquete respuesta es recibido por HRI y enviado al diálogo. [7]El diálogo recibe la respuesta y la hace saber al usuario utilizando sus capacidades de síntesis de voz. [6]Finalmente, el diálogo pregunta al usuario que tarea desea realizar ahora. Si decide indicar otra pregunta se regresa al punto 2; en otro caso, el flujo termina.
Subflujos	S-1 El diálogo solicita al usuario la pregunta a realizar. Se continúa con el diálogo principal en el punto 3.

3.2.3 Entrenamiento

Se requiere que el sistema sea capaz de interactuar con el robot de modo que se le puedan "enseñar" comportamientos a partir de acciones simples u otros comportamientos previamente entrenados, buscando siempre la interacción más natural posible entre el usuario y el sistema.

En general, se requiere de los entrenamientos las siguientes características:

- El entrenamiento debe estar basado en una combinación de acciones simples y la identificación de ciertos eventos relacionados con las percepciones del robot, tales como reconocer la pelota o a otro robot. Se pretende que los diálogos de entrenamiento conduzcan a la elaboración de frases tales como "gira a la derecha *hasta* que veas la pelota" donde se relacionan de forma explicita acciones (girar a la derecha) con eventos derivados de las percepciones del robot (reconocer la pelota)
- El sistema, además, debe ser capaz de ofrecer entrenamientos en un lenguaje lo más aproximado al natural. Para ello se utilizaron condicionantes tales como las palabras *hasta* y *mientras*.
- Los comportamientos derivados de los entrenamientos deben poder agregarse al conjunto de acciones que el robot puede reproducir en cualquier momento, para ello, cada comportamiento debe incorporarse al sistema como una acción a ejecutar. Lo anterior implica que cada comportamiento debe ser almacenado en algún medio que garantice la posibilidad de reproducirlo posteriormente.
- Debe ser posible crear entrenamientos basados en la ejecución de otros comportamientos previamente entrenados.
- Se debe contar con al menos una estructura de decisión que permita al usuario ordenar al robot ejecutar una de dos acciones diferentes según se presente o no una percepción.

La tabla 3.3 muestra algunos ejemplos de las conversaciones que se espera poder desarrollar entre el usuario y el sistema. Los dos primeros ejemplos muestran entrenamiento basados en acciones simples tal como girar o tirar a gol, el último ejemplo muestra como se desea utilizar la estructura de decisión mencionada en el último punto de la lista de requerimientos.

Actor	Diálogo
Sistema	¿Qué quieres que haga?
Usuario	Gira a la derecha hasta que veas la pelota.
Sistema	¿Qué quieres que haga?
Usuario	Ve hacia la pelota mientras la pelota este lejos, después toma la pelota.
Sistema	¿Qué quieres que haga?
Usuario	Si ves la portería tira a gol, sino pasa la pelota a un compañero.

Tabla 3.3 Conversación deseada para entrenamiento.

La capacidad de entrenamiento es la pieza en que se centra el presente trabajo; por ello, para garantizar que se cubran los requerimientos antes enlistados, se planeó realizar una serie de pruebas que demuestren las capacidades del sistema.

La primera de estas pruebas se denomina *Go* y debe cumplir los siguientes requerimientos:

- El robot debe ser capaz de identificar la pelota.
- El robot debe buscar la pelota cuando no la observe.
- El robot debe acercarse a la pelota mientras la vea.
- El robot debe tomar la pelota cuando este a su alcance.
- El robot debe poder realizar el comportamiento sin importar la posición inicial en la cancha o la distancia a la pelota.

La segunda prueba que se realizará comienza con el robot en posesión de la pelota, la prueba se llama *Shoot* y debe cumplir con los siguientes puntos:

- El robot debe ser capaz de identificar las porterías.
- El robot debe rotar, manteniendo la pelota, hasta identificar la portería.
- El robot debe tirar a gol cuando vea la pelota.
- El robot debe poder realizar el comportamiento sin importar la posición inicial en la cancha.

Una tercera prueba, llamada *Go & Shoot*, involucra a las dos anteriores; esta prueba es un entrenamiento complejo basado en los dos comportamientos entrenados anteriormente. El comportamiento involucra que el robot primero busque y tome la pelota, luego que busque la portería y que tire. Por lo anterior se requiere que:

- El sistema debe ser capaz de reproducir comportamientos almacenados
- El robot debe poder realizar el comportamiento sin importar la posición inicial en la

cancha o la distancia a la pelota.

Por último, una prueba denominada *Go & Decision* probará la efectividad de la estructura de decisión. Se requiere que:

- El sistema debe ser capaz de aceptar una estructura de decisión que permita al robot efectuar uno de dos comportamiento dependiendo de sus percepciones del entorno.

3.2.3.1 Caso de uso

En este caso de uso, el usuario entrena al robot para que este adquiera un nuevo comportamiento basado en acciones simples o en comportamientos anteriormente aprendidos. El usuario indica primero una acción a realizar seguida de una condición (mientras/hasta) relacionado con una percepción del robot. Cada vez que se realizan estas acciones, el sistema relaciona la percepción con la acción que se ordenó ejecutar. De este modo la percepción del robot se asocia a un comportamiento reproducible.

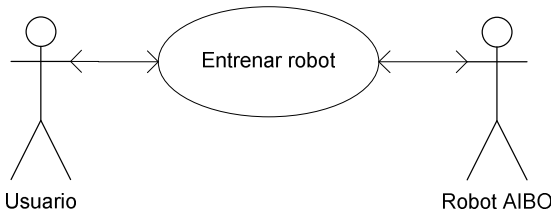


Figura 3.4 Caso de uso: Entrenar al robot.

Caso de uso	Entrenar al robot
Actores	Usuario, Robot AIBO.
Precondiciones	Iniciar el *diálogo de entrenamiento* en la aplicación CSLU – RAD
Flujo principal	[1]El caso de uso inicia cuando en usuario indica al diálogo que desea dar una instrucción al sistema. [2]El diálogo indica al usuario que proporcione la instrucción. [3]El usuario debe seleccionar una acción válida a realizar. Si no lo hace correctamente se ejecuta el subflujo S-1. [4]El usuario selecciona ahora una de las condiciones para la ejecución de la acción (hasta/mientras). Si no lo hace correctamente se ejecuta el

	subflujo S-1. [5]El usuario debe elegir la percepción a la que se asocia tanto la condición como la acción. Si no lo hace correctamente se ejecuta el subflujo S-1. [6]El módulo HRI es invocado para registrar la instrucción completa. [7]Finalmente, el diálogo pregunta al usuario si desea realizar otro ciclo de entrenamiento. Si decide que sí, el flujo regresa al punto 2; en caso contrario, termina.
Subflujos	S-1 El diálogo solicita al usuario proporcionar una opción válida. Se continúa con el diálogo principal en el punto 3.

CAPÍTULO 4

DISEÑO DEL SISTEMA

En el presente capítulo se presenta la arquitectura del sistema desarrollado en esta tesis. El capítulo contiene una explicación de la funcionalidad de cada uno de los módulos en que se divide el sistema. Además se da una explicación acerca de cómo se cubrieron los requerimientos del capítulo anterior.

4.1 SISTEMA DE INTERACCIÓN HUMANO-ROBOT

Como se mencionó en el capítulo anterior, para que el sistema funcione es necesario la intervención de dos actores: el robot y el usuario. La figura 4.1 muestra los tres módulos que constituyen el sistema de interacción:

- Diálogo de interacción: este módulo ofrece la interacción entre el usuario y el resto del sistema. Se trata, como su nombre lo indica, de un diálogo de interacción desarrollado con la herramienta *Rapid Application Developer* incluida en el *CSLU Toolkit*. El módulo se ejecuta en una computadora.
- HRI: este módulo será el enlace entre la computadora y el robot. Su función es tomar las preguntas e instrucciones de acciones y enviarlas al robot de manera inalámbrica. También es el encargado del proceso de relacionar percepciones con acciones, operación necesaria para el entrenamiento de comportamientos.
- EK2008: este módulo consiste en una actualización del sistema EK2007 utilizado para competencia de *RoboCup*. Esta actualización provee al robot de capacidades de comunicación con en módulo HRI, de ejecución de las acciones ordenadas y de responder a preguntas especificas relacionadas con objetos detectados por la cámara.

Los dos bloques que se ejecutan en la computadora constituyen el núcleo del sistema desarrollado en esta tesis.

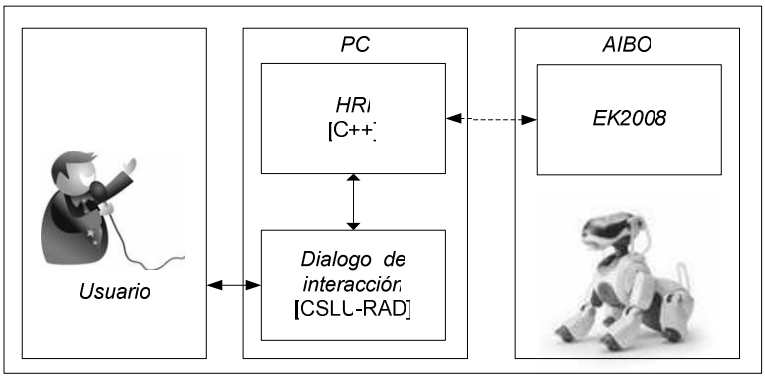


Figura 4.1 Arquitectura del sistema.

4.1.1 Robot

La figura 2.15 muestra los módulos que componen el sistema EK2007. La actualización denominada EK2008 incorpora un nuevo módulo llamado MetaComm encargado de establecer, en el robot, las comunicaciones con en módulo HRI que se ejecuta la computadora.

MetaComm es un subsistema de comunicación bidireccional que permite recibir y enviar mensajes UDP. Este subsistema se comunica además con el subsistema EKMain, encargado de ejecutar los movimientos o responder las preguntas que son recibidas por el robot. La figura 4.2 muestra la interacción de este nuevo módulo con el resto del sistema EK2008.

EKVision recibe de la cámara un cuadro cada 1/30 segundos, cada píxel en el cuadro es asociado con uno de nueve colores considerados relevantes (naranja, amarillo, cian, verde, blanco, rojo, azul, rosa o negro). La asignación considera los componentes Y, U y V de cada píxel y los clasifica de acuerdo con un cuadro de referencia proporcionado con anterioridad al robot. Posteriormente, se agrupan píxeles de acuerdo a su color y se definen regiones. Las regiones son evaluadas bajo ciertos criterios que determinan si se trata o no de un objeto relevante para el desarrollo del juego de fútbol. Como salida, EKVision entrega a EKMain y EKLoc una estructura de datos que contiene información sobre los objetos reconocidos. En [MAR 2005] se pueden consultar los algoritmos de procesamiento de imágenes utilizados por EKVision.

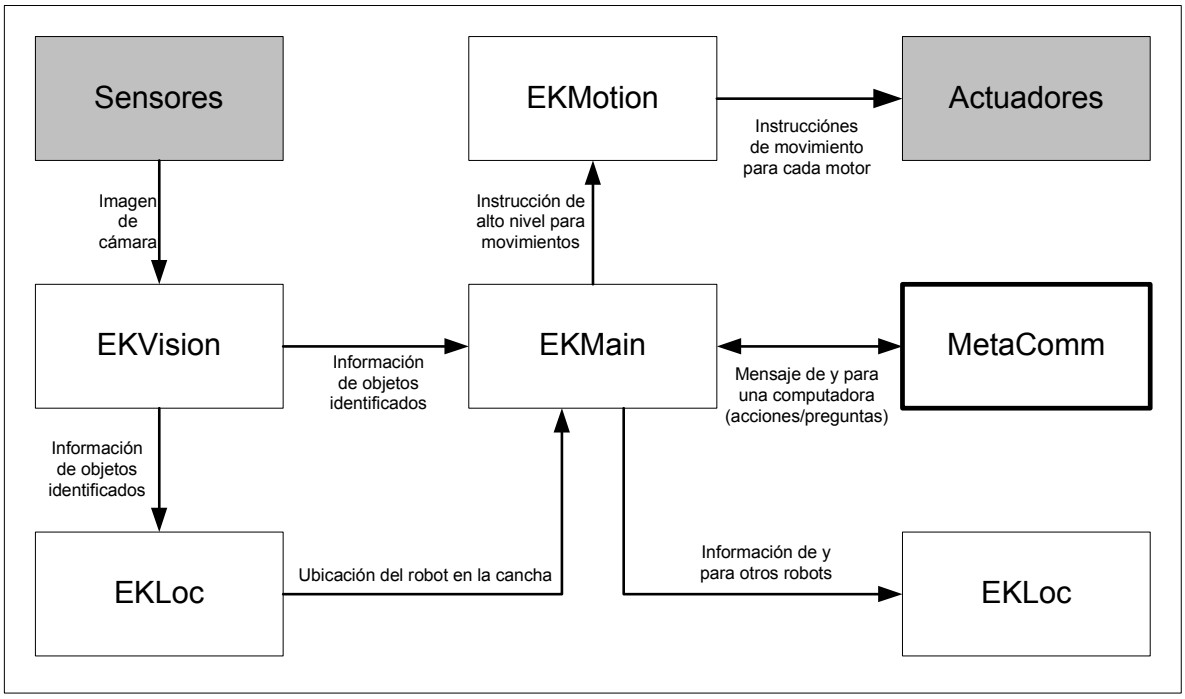


Figura 4.2 Sistema EK2008.

EKLoc toma la información proporcionada por EKVision para estimar, a través del cálculo de distancias a los objetos y utilizando métodos probabilísticas, una posición aproximada del robot en la cancha. EKLoc entrega a EKMain como resultado la estimación de la posición del robot así como un ángulo de orientación. En [MAR 2008] se documentó la información relacionada al sistema de localización del robot.

EKLoc es una interfaz de comunicaciones que permite al robot comunicarse con otros robots y compartir con ellos información relevante para el desarrollo del juego. El intercambio de información mediante EKLoc permite que cada jugador conozca la posición aproximada de todos los robots de su equipo así como la posición de la pelota cuando alguno de los robots la detecta. Esta información es útil para el desarrollo de estrategias de juego.

EKMain es el subsistema encargado de definir los comportamientos que debe ejecutar el robot. EKMain toma decisiones basado en la información que recibe de EKVision, EKLoc y EKComm. Una vez que toma la decisión, entrega a EKMotion una instrucción de movimiento.

EKMotion toma las instrucciones de movimientos de altos nivel enviadas por EKMain y las descompone en instrucciones para cada uno de los motores involucrados en la ejecución de los

movimientos. La implementación de nuevos movimientos fue necesario para la implementación de este trabajo por lo que el la siguiente sección se detalla el proceso del diseño de éstos.

4.1.1.1 Diseño de movimientos del robot

El diseño de los movimientos se realizó utilizando la herramienta MEdit distribuida por Sony para los usuarios del robot AIBO ERS-7. Esta aplicación despliega un modelo tridimensional del robot AIBO y permite, entre otras cosas, modificar los valores de los actuadores y ver reflejado en el modelo el resultado. La figura 4.3 muestra una captura de las ventanas principales del software MEdit.

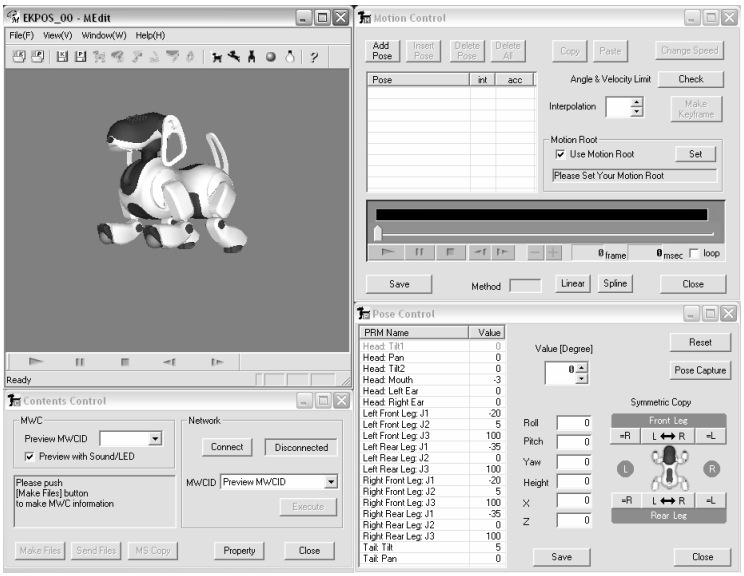


Figura 4.3 Software Sony MEdit.

MEdit además de permitir observar en el modelo tridimensional los valores en los actuadotes del robot, permite guardar cada configuración del robot en un archivo con extensión "pse". Se pueden diseñar las distintas posiciones que en un movimiento debe asumir el robot y mediante la herramienta Motion Control de MEdit obtener una animación donde la transición entre una posición y otra se interpola según el tiempo que se desee que tarde el movimiento completo. De este modo, MEdit puede desplegar animaciones sencillas que proporcionan una visión muy realista de lo que sería el movimiento ejecutado en el robot.

El software MEdit fue utilizado para el diseño de la mayoría de los movimientos utilizados en el presente trabajo. Por ejemplo, la figura 4.4 muestra la secuencia de movimientos necesarios para que el robot tome la pelota. El robot parte la posición estándar de caminata y utiliza las patas delanteras y la cabeza para sujetar la pelota.

Figura 4.4 Movimiento para tomar la pelota.

La secuencia de movimientos para la ejecución de un tiro a gol es similar a la anterior, el robot parte de la posición estándar de caminata y extiende sus patas delanteras y la cabeza para tomar la pelota. Una vez que tiene posesión de la pelota deja caer la parte trasera de su cuerpo al suelo y eleva su cabeza hasta el máximo que sus motores le permiten. Desde esa posición, el robot eleva un poco la parte frontal de su pecho y en un movimiento rápido se deja caer para impactar la pelota con la mayor fuerza posible. El impacto se realiza con la cabeza y la posición extendida de la patas hacia el frente permite guiar la pelota hacia enfrente. Tras ejecutar la secuencia de tiro, el robot contrae sus patas y se regresa a la posición estándar de caminata, listo para ejecutar el siguiente movimiento cuando sea requerido. La figura 4.5 muestra la ejecución del tiro desde la posición en que el robot sujeta la pelota.

Figura 4.5 Movimientos para ejecutar un tiro.

Otro movimiento que es requerido realizar es el denominado bloqueo, en este movimiento, el robot parte de la posición estándar de caminata y rápidamente extiende sus patas delanteras hacia los lados de manera tal que cae rápidamente al suelo para bloquear con toda la parte delantera el curso de la pelota. El robot necesita inmediatamente reincorporarse pues este movimiento está pensado para su ejecución por parte de un portero La figura 4.6 muestra dicho movimiento.

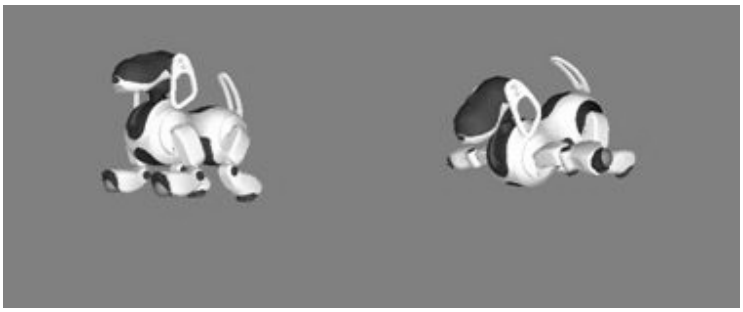


Figura 4.6 Movimiento de bloqueo.

Para los movimientos de rotación en cualquiera de las dos direcciones se tienen dos casos, cuando se tiene posesión de la pelota y cuando esto no ocurre. En el primero de los casos, el movimiento del robot comienza desde la posición de sujetar, es decir, el final del movimiento para tomar la pelota (figura 4.4); desde esta posición, el perro desplaza lateralmente su pata delantera del lado hacia el que quiere moverse. La figura 4.7 ilustra un ejemplo.

Si se desea rotar a la izquierda, la pata delantera izquierda del robot se desplazará lateralmente hacia fuera (izquierda en este caso); para compensar este movimiento, la extremidad opuesta (la pata trasera derecha, en este caso) se desplazará también hacia fuera (derecha). Ahora, el robot debe tomar nuevamente la posición de sujetar, al hacerlo la fricción contra la cancha obligará a rotar al robot en la dirección deseada

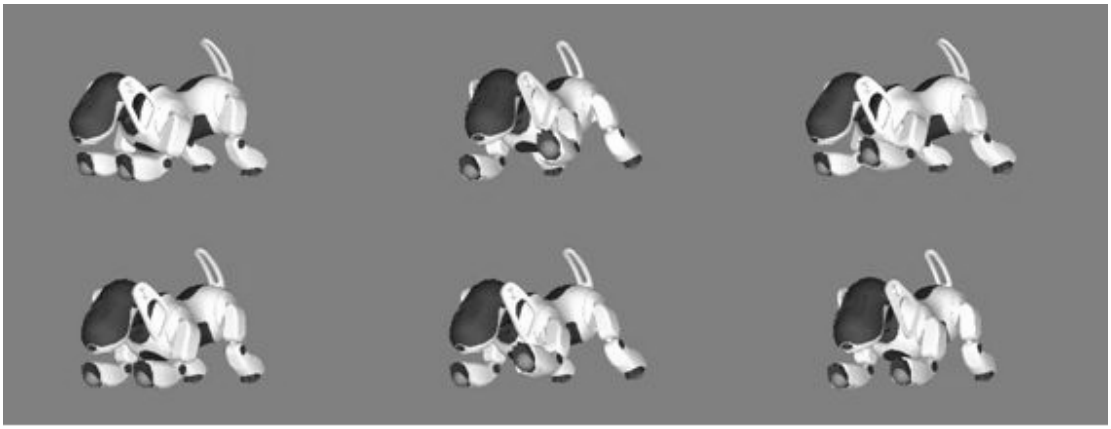


Figura 4.7 Rotar a la izquierda con pelota.

El movimiento para rotar cuando no se posee el balón es análogo al anterior. En este caso, el movimiento parte de la posición estándar de caminata y de igual forma la pata delantera del lado al que se quiere rotar efectúa el movimiento hacia fuera. La figura 4.8 muestra la secuencia de movimientos para este caso.

Una vez probados los movimientos mediante el software MEdit, los valores de cada uno de los actuadores en el modelo tridimensional son tomados y utilizados en la implementación de los mismos para el robot. El proceso de implementación de los movimientos se explicará en el siguiente capítulo.

Figura 4.8 Giro a la izquierda sin pelota.

4.1.2 Interacción Humano-Computadora

La interacción entre el humano y la computadora se realiza mediante los denominados diálogos de interacción.

Los diálogos de interacción son las pantallas que se presenta al usuario del sistema. Mediante estas pantallas se pretende dar al usuario la posibilidad de interactuar con el robot utilizando como medio la voz. Dado que el objetivo de esta tesis no es elaborar un sistema de reconocimiento de voz, se decidió el utilizar una de las múltiples herramientas con capacidades de reconocimiento de voz disponible en el mercado.

La herramienta seleccionada fue el *Rapid Application Developer* incluida dentro del *CSLU Toolkit.* Una breve introducción a las capacidades de esta herramienta fue incluida como punto final en el capítulo dos de este trabajo.

Cabe aclarar que las capacidades de reconocimiento de la herramienta RAD son limitados pues los nodos que realizan la tarea de reconocimiento de frases o palabras sólo son capaces de identificar palabra o frases de entre un conjunto definido *a priori*; lo anterior implica que el sistema no es flexible al uso de un vocabulario amplio y que el usuario debe adaptarse al limitado conjunto de frases y estructuras de comunicación definidas por el diálogo de interacción.

Con la herramienta RAD se diseño un primer flujo, denominado *diálogo de interacción básica*, que cuenta con dos subflujos principales. El primero de ellos corresponde a las acciones o comportamientos a ejecutar por el robot. El segundo de los subflujos está relacionado con las preguntas que se realizan al robot. Mediante el diálogo de interacción básica se pretende aproximar la interacción entre el usuario y el sistema a conversaciones como las presentadas en las tablas 3.1 y 3.2 del capítulo anterior.

Se diseño un segundo diálogo denominado *diálogo de entrenamiento* en el que cada instrucción de entrenamiento o de decisión contiene una secuencia de percepciones y acciones. Se pretendía que las conversaciones entre el usuario y el sistema fueran similares a la mostrada en la tabla 3.3.

La figura 4.9 muestra el proceso de ejecución de una instrucción de acción. El proceso empieza cuando el usuario emite el comando de acción, el diálogo de interacción realiza los procesos de reconocimientos del comando de voz, identificación de la acción a realizar y ejecución del módulo HRI; éste último toma de los parámetros de ejecución la información relacionada con la acción que se desea ejecutar. HRI empaqueta esta información y la envía en un mensaje UDP hacia el robot. Cuando es recibido en el robot, se extrae del mensaje la acción a ejecutar y se invoca a los procedimientos necesarios para su ejecución.

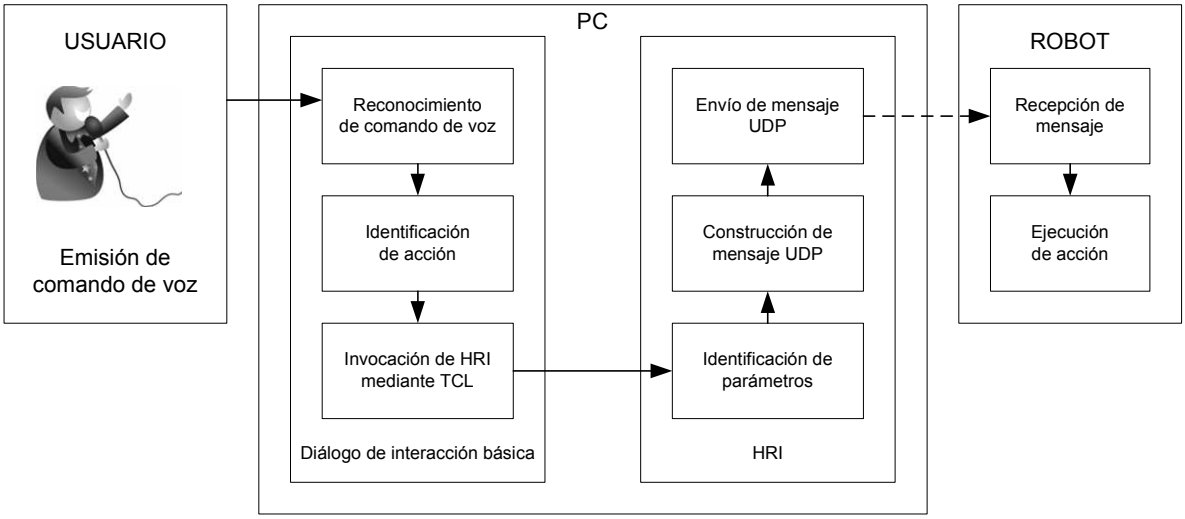


Figura 4.9 Proceso de ejecución de una acción.

Un proceso similar se ejecuta cuando se interroga al robot acerca de alguna percepción. En este caso, el proceso no termina cuando el robot recibe el mensaje; en lugar de ejecutar una acción, el robot verifica la información acerca de los objetos que percibe y genera una respuesta para la pregunta que recibió. Luego, el robot empaqueta la respuesta y la envía a la computadora mediante otro mensaje UDP. En la computadora, HRI recibe el mensaje y envía la respuesta al diálogo de interacción básica; este último despliega la respuesta utilizando el sintetizador de voz. La figura 4.10 muestra el proceso descrito anteriormente.

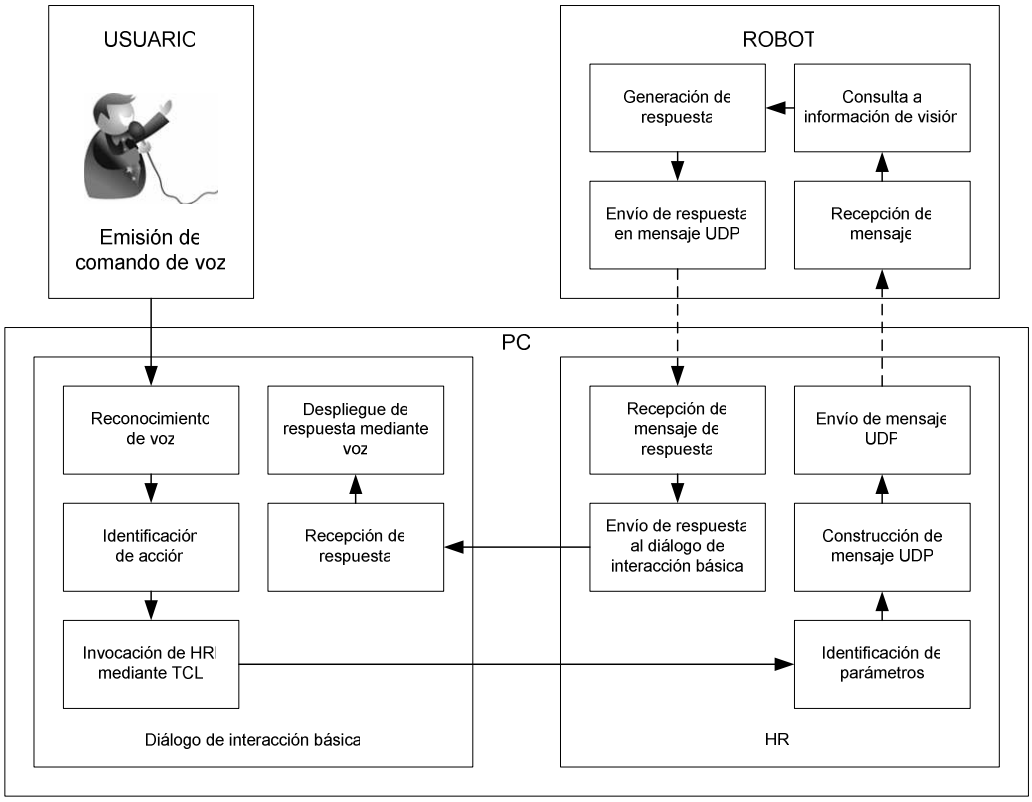


Figura 4.10 Proceso de ejecución de una pregunta.

Finalmente, cuando se trata de una instrucción de entrenamiento, el diálogo de entrenamiento identifica los componentes de la instrucción y los pasa como parámetros de ejecución a HRI. Este programa los toma y envía preguntas y acciones al robot hasta cumplir con la condición de la instrucción. El proceso es explicado a detalle en la sección 4.2.3.

4.1.3 Comunicación entre la computadora y el robot

La comunicación entre la computadora y el robot se realiza mediante la interacción de dos módulos, HRI y MetaComm. HRI es ejecutado en la computadora mientras que MetaComm, un subsistema de EK2008, es parte de la programación del robot.

HRI es un módulo desarrollado en C/C++ que implementa la funcionalidad de entrenamiento y que sirve como puente de comunicaciones con el robot mediante un socket. UDP.

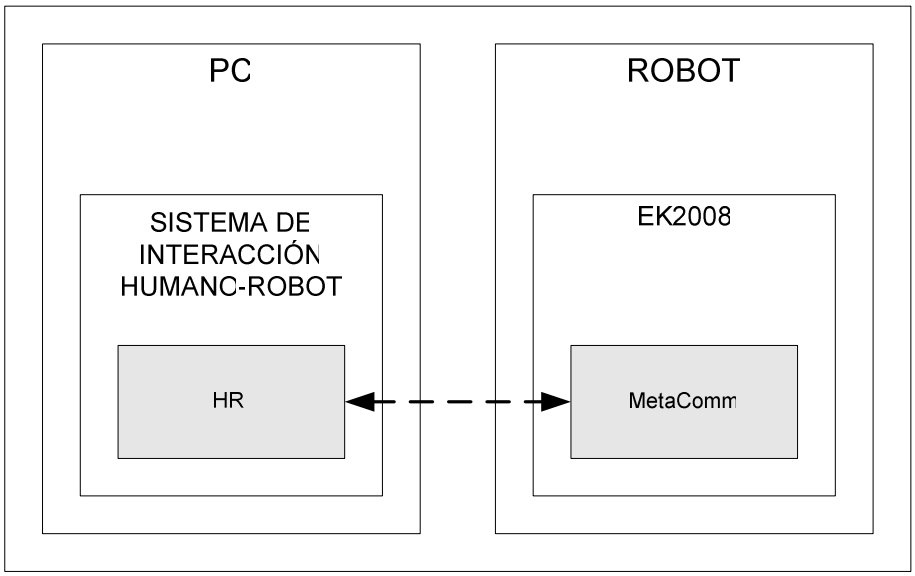


Figura 4.11 Comunicación entre la computadora y el robot.

Se seleccionó el protocolo UDP pues se deseaba enviar una gran cantidad de paquetes al robot en un tiempo corto de tiempo sin preocuparnos por la perdida de alguno de ellos, ya que en un periodo muy pequeño de tiempo se recibiría otro paquete. Así, un protocolo no orientado a conexión ofrece las características de velocidad y fiabilidad requeridas en la transmisión.

El módulo HRI realiza cinco comportamientos dependiendo de la información que recibe cuando es ejecutado:

- Cuando desde el diálogo de interacción se ordena la ejecución de una acción, HRI recibe como parámetro un identificador de la acción a realizar por el robot; toma este identificador y lo empaqueta en una estructura de datos para ser enviada vía un socket UDP al robot AIBO.
- Cuando se trata de una pregunta, HRI envía un identificador de la pregunta hacia el robot, pero justo antes de enviarlo arranca un hilo de ejecución que espera una respuesta del robot. Cuando el mensaje de respuesta es recibido, HRI devuelve al diálogo de interacción la información de la respuesta a la pregunta.
- Cuando se trata de una instrucción de entrenamiento, HRI recibe como parámetros los identificadores la percepción y la acción a realizar, así como la condición bajo la cual se relacionan. Para implementar el aprendizaje, se almacena en un archivo los identificadores, de este modo, el comportamiento puede ser invocado posteriormente.

- Cuando se enseña al robot a tomar una decisión, HRI recibe como parámetro la percepción que condiciona, una acción a realizar cuando se cumple la condición y otra a ejecutar cuando no ocurra la percepción.
- Por último, cuando se desee ejecutar un comportamiento aprendido anteriormente, HRI recibe como parámetro el nombre del archivo que contiene la información del comportamiento a ejecutar. HRI accede al archivo y de él extrae la información relacionada con el comportamiento; a partir de esa información se construye un autómata de estados finitos que controla el comportamiento a ejecutar.

En el robot, MetaComm se encarga de recibir todos los mensajes provenientes del módulo HRI en la computadora, verificar su integridad y pasarlos a EKMain. Para cada mensaje recibido, EKMain verifica si el mensaje recibido corresponde a una acción o a una pregunta. Si se trata de una acción se invocará a la función adecuada de EKMotion; por el contrario, si se trata de una pregunta, EKMain consulta la información de los objetos que detecta y genera una respuesta. La respuesta es enviada al módulo HRI en la computadora mediante el subsistema MetaComm. Un diagrama con el proceso aquí descrito puede observarse en la figura 4.12.

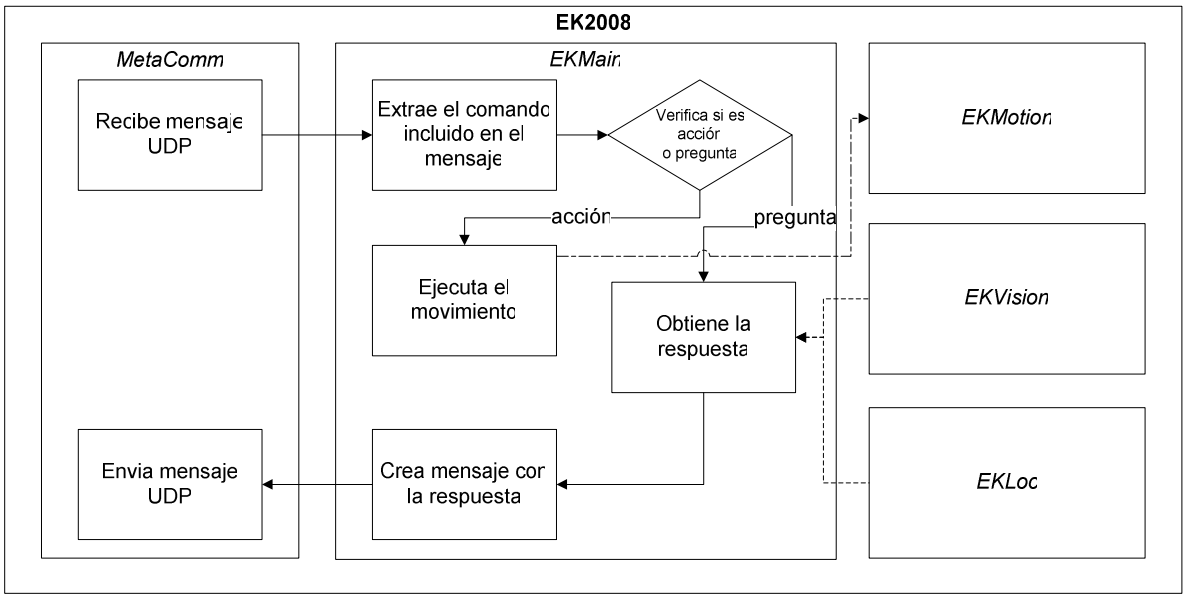


Figura 4.12 Proceso de recepción de mensaje en el robot AIBO.

4.2 ACCIÓN, INTERROGACIÓN Y ENTRENAMIENTO

Como se mencionó en las secciones anteriores, dos diálogos de interacción fueron desarrollados; el primero de ellos, llamado *diálogo de interacción básica* (figura 4.13), es utilizado para enviar comandos de acción y para interrogar al robot

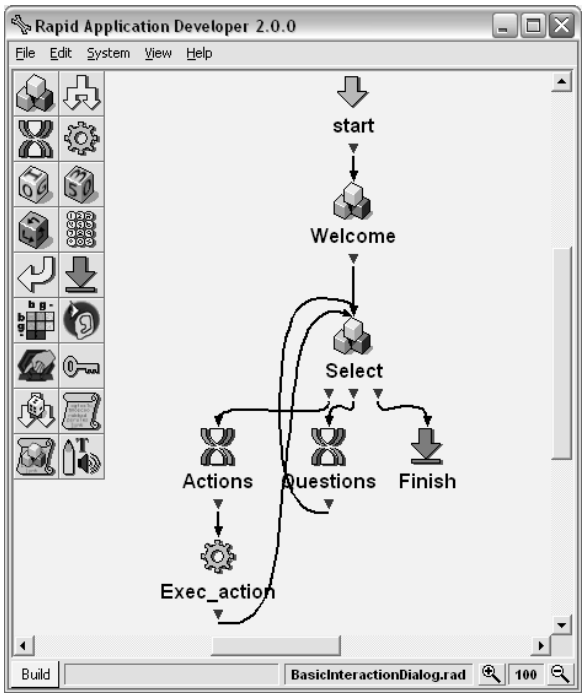


Figura 4.13 Diálogo de interacción básica.

4.2.1 Acción

El *diálogo de interacción básica* provee al usuario la capacidad de ordenar al robot diez acciones. Las acciones se enlistan en la tabla 4.1; en ésta, el comando de voz es la palabra o frase en idioma inglés con que se reconocer la acción y el nodo RAD se refiere al nodo del diálogo de interacción que reconocer el comando de voz.

Para la asignación del comando de voz se decidió utilizar palabras o frases de la menor longitud posible que describieran la acción a desencadenar. El uso de una sola palabra basto para acciones que involucran movimiento del robot sin interacción con objetos del entorno; por otro lado, cuando se interactúa con un objeto fue necesario describir la acción mediante un frase corta en la que se hace referencia al objeto.

Comando de voz	Nodo RAD	Acción
Stop	Stop	Detenerse.
Walk	Walk	Caminar.
Right	TurnR	Giro a la derecha.
Left	TurnL	Giro a la izquierda.
Kick	Kick	Patea la pelota.
Hold Ball	Hold	Toma la pelota.
Right with Ball	TRH	Giro a la derecha manteniendo la pelota.
Left with Ball	TLH	Giro a la izquierda manteniendo la pelota.
Go to Ball	Go -> Ball	Ir hacia la pelota y detenerse frente a ella.
Block	Block	Bloquear.

Tabla 4.1 Comandos de acción.

El software RAD proporciona la capacidad para reutilizar parte de los diálogos mediante la creación de subdiálogos; así, se puede tener un diálogo principal con pocos nodos cuya funcionalidad total deriva de la complejidad de cada uno de los subdiálogos que posee.

Como las instrucciones de acción son utilizadas en todos los diálogo, se decidió crea un subdiálogo denominado *Actions* que contiene los nodos necesarios para la identificación de las palabras indicando acciones. El subdiálogo *Actions* puede observarse en la figura 4.14.

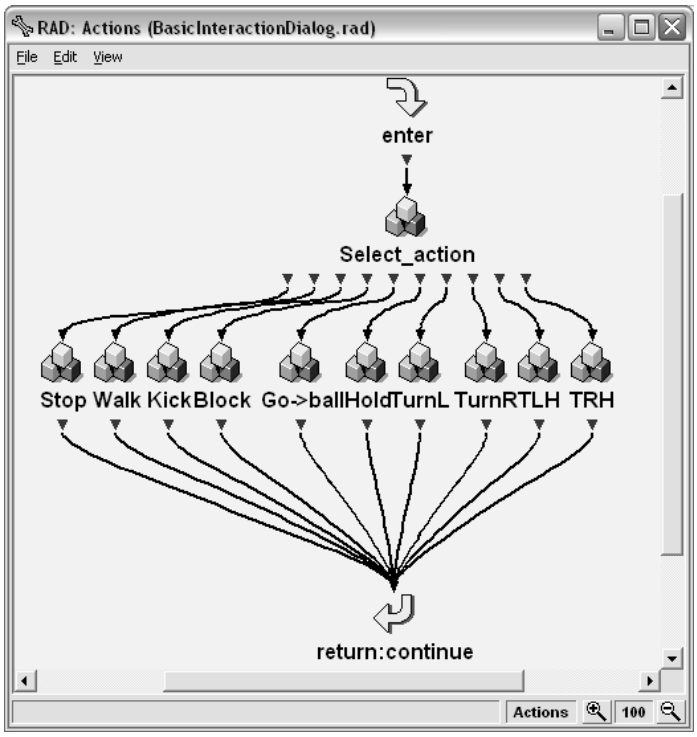


Figura 4.14 Subdiálogo *Actions*.

4.2.2 Interrogación

Al igual que con las acciones, la identificación de las preguntas y la síntesis de la respuesta constituyen un bloque reutilizable de nodos y por tanto se decidió incluirlos en un subdiálogo llamado *Questions*. El subdiálogo puede observarse en la figura 4.15.

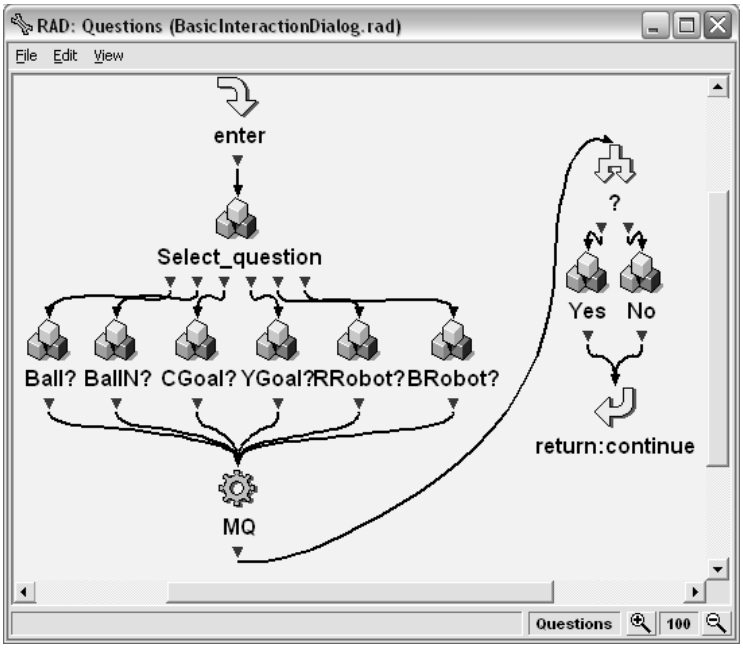


Figura 4.15 Subdiálogo *Questions*.

La tabla 4.2 muestra el conjunto de preguntas que se pueden realizar al robot. La primera columna corresponde al comando de voz que identifica dentro del diálogo de interacción a cada pregunta. La segunda columna, al igual que en la tabla 4.1, identifica el nodo en el diálogo de interacción en que se implementa parcialmente la interrogación. El identificador corresponde al valor entero que distingue dentro del sistema a la pregunta que se realizar. La última columna muestra la respuesta que se recibe del sistema según la percepción que tenga el robot de su entorno. Para la selección de los comandos de voz se buscó una pregunta lo más corta posible en la que se hiciera referencia explicita a los objetos sobre los que se pregunta.

Comando de voz	*Nodo CSLU*	*Id. Respuesta*
Do you see the ball?	Ball?	Ve la pelota – 1, No vela pelota - 0
Do you see the yellow goal?	YGoal?	Ve la portería amarilla – 1, No la ve - 0
Do you see the cyan goal?	CGoal?	Ve la portería cian – 1, No la ve – 0
Do you see the ball near?	BallN?	Ve la pelota cerca – 1, No la ve cerca - 0
Do you see a red robot?	RedR?	Ve robot rojo – 1, No lo ve – 0
Do you see a blue robot?	BlueR?	Ve robot azul – 1, No lo ve – 0

Tabla 4.2 Comandos de interrogación.

El subdiálogo *Questions* es ligeramente más complejo que el subdiálogo *Actions*; al igual que en este último, *Question* provee la capacidad de reconocer entre un número predeterminado de comandos de voz, en este caso comandos de interrogación. Cuando se ha seleccionado alguna de las preguntas disponibles en la tabla 4.2, el nodo *MQ* ejecuta HRI; éste a su vez envía la pregunta al robot y espera por la respuesta, cuando la respuesta es recibida, HRI devuelve el resultado al diálogo de interacción. El nodo *?* del diálogo identifica si la repuesta es positiva o negativa y según sea el caso dirige el flujo hacia los nodos *Yes* o *No*, los cuales expresan de forma oral la respuesta.

El diálogo completo ofrece al usuario la posibilidad de ordenar al AIBO que realice alguna acción o bien que conteste a alguna pregunta del conjunto que le es posible responder.

4.2.3 Entrenamiento

El *diálogo de entrenamiento* ofrece al usuario la posibilidad de enseñar al robot comportamientos basados en una serie de acciones simples. Para lograr esto, se debe relacionar alguna percepción del robot con una acción.

El subdiálogo *Perceptions* (figura 4.16) es una modificación del subdiálogo *Question*, en el que en lugar de realizar la pregunta, se hace referencia directa al objeto que se percibe. Por ejemplo, en lugar de la pregunta "*Do you see the ball?*", se hace referencia al hecho de reconocer la pelota mediante el comando de voz "*you see the ball*"; lo anterior se hace con el fin de hacer posible una interacción más natural entre el usuario y el sistema, de modo que se imiten conversaciones como la presentada en la tabla 3.3 del capítulo anterior.

El subdiálogo no emite una respuesta, sólo permite seleccionar alguna de las percepciones

utilizando los comandos mostrados en la tabla 4.3. Al igual que en las dos tabla anteriores, la primera columna de la tabla contiene el comando de voz que identifica la percepción, la segunda columna, se refiere al nodo del subdiálogo de interacción que identifica al comando y, por ultimo, el tercer elemento es el identificador numérico para la percepción dentro del sistema. Cabe resaltar el hecho de que los identificadores para los subdiálogos *Questions* y *Perceptions* son los mismos ya que en ambos casos se hace referencia a una percepción del robot.

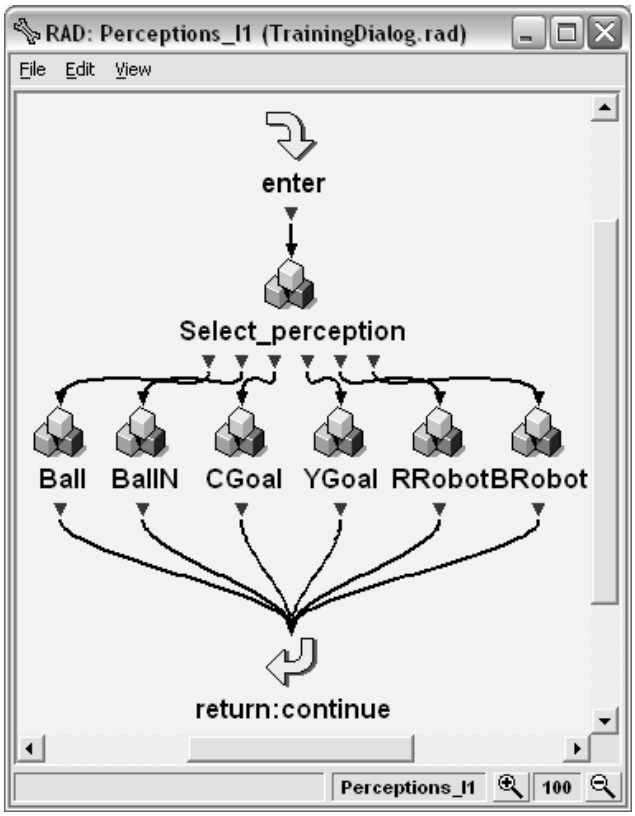


Figura 4.16 Subdiálogo *Perceptions*.

Comando de voz	***Nodo CSLU***
You see the ball	Ball
You see the yellow goal	YGoal
You see the cyan goal	CGoal
You see a red robot	RedR
You see a blue robot	BlueR
You see the ball near	BallN

Tabla 4.3 Comandos de percepción.

.

La figura 4.17 muestra el *diálogo de entrenamiento*; este diálogo está integrado por dos partes o flujos principales: el primero se utiliza para realizar instrucciones de entrenamiento y el segundo para realizar una instrucción de decisión.

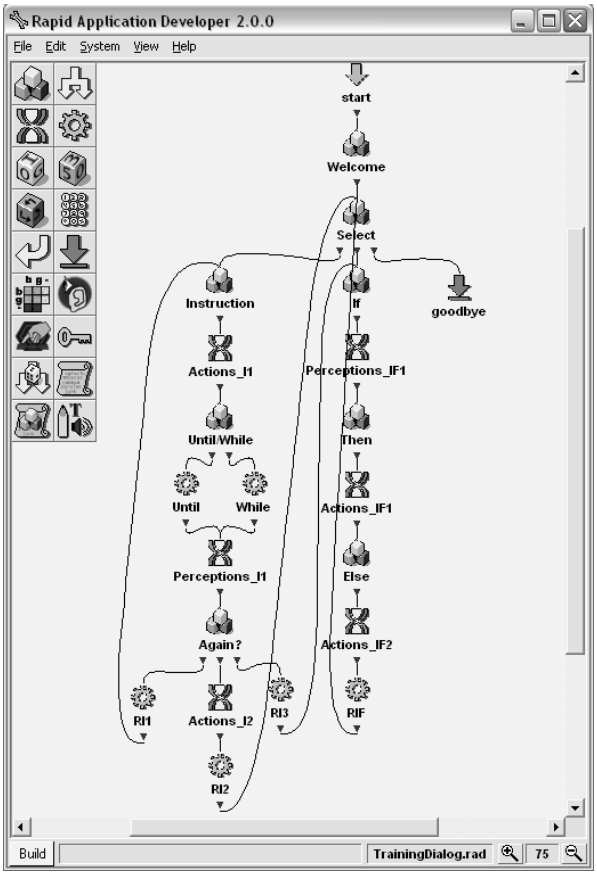


Figura 4.17 Diálogo de entrenamiento.

Una instrucción de entrenamiento es un comando de voz del tipo: “turn left until you see the ball” en el que se pueden identificar tres componentes; el primero de ellos corresponde a una acción a realizar, en el ejemplo dado se trata de girar a la izquierda (“turn left”); el segundo elemento es un condicionante que indica cuando se realizará la acción, la acción puede realizarse mientras (“while”) o hasta (“until”) que se cumpla una percepción; el tercer y último elemento de la instrucción de entrenamiento es precisamente la percepción, en el ejemplo “you see the ball”. Dado que el software RAD no nos permite identificar estos elementos a partir de un solo comando de voz, la instrucción tiene que proporcionarse al sistema en tres nodos. El primero de estos elementos se identifica utilizando el subdiálogo *Actions*, el segundo utilizando el nodo *Until/While* y el tercero mediante el subdiálogo *Perceptions*. De este modo,

es posible aproximar el tipo de interacción presentado en la tabla 3.3. La figura 4.18 muestra la estructura a la que se debe ajustar el usuario cuando emite una instrucción de entrenamiento.

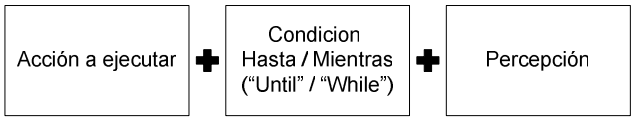


Figura 4.18 Estructura de una instrucción de entrenamiento

Cada vez que se completa una instrucción de entrenamiento, se almacena el identificador de la acción, la condición y la percepción a que se asocia; de este modo, cuando el robot tenga que ejecutar el comportamiento aprendido, utilizará una pregunta referente a la percepción y su respuesta como disparador de la acción. Una vez aprendido un comportamiento nuevo, este puede ser añadido de forma manual al subdiálogo *Actions* de modo que este disponible para su utilización en entrenamientos futuros.

Una instrucción de decisión, por otro lado, permite indicar al robot que existe la posibilidad de realizar una de dos tareas dependiendo de una percepción. Las instrucciones que se pueden implementar son del tipo “if you see the cyan goal then kick else turn left with the ball”; en esta instrucción pueden identificarse también tres componentes principales el primero de ellos es la percepción de condiciona la acción a realizar, en este caso se trata de “you see the ball”; el segundo elemento es la acción a realizar cuando la condición se cumple (“kick”); por último se tienen la acción que se realizará cuando la condición no se cumpla (“turn left with ball”). En las instrucciones de decisión, las palabras “if” “then” “else” dan sentido a la oración pero no forman parte del conjunto identificable por alguno de los subdiálogos desarrollados, por tanto se incluyeron nodos especiales cuya única función es identificarlas. Así, una instrucción de decisión requiere de un primer nodo para identificar la palabra “if” seguido del subdiálogo *Perceptions*; después, se debe colocar un nodo que identifique la palabra “then” seguido de un primer subdiálogo *Actions*; finalmente, un nodo para la identificación de la palabra “else” debe ser colocado junto antes de otro subdiálogo *Actions*. La figura 4.19 muestra la estructura que guarda una instrucción de decisión.

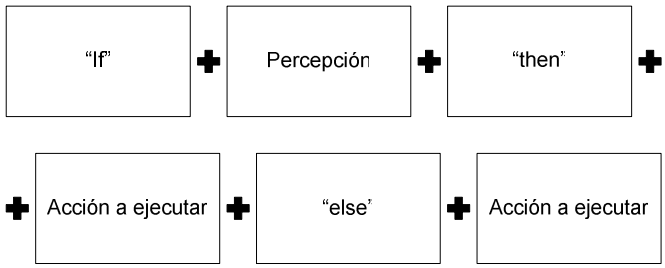


Figura 4.19 Estructura de una instrucción de decisión

En el capítulo anterior se definieron los requerimientos para las cuatro pruebas que se realizaron y que se explican a detalle en el capítulo seis.

Los requerimientos para las pruebas *Go* y *Shoot* se cubren con el diseño de las instrucciones de entrenamientos. El hecho de incluir los condicionantes “until” y “while” permiten que las tareas de búsqueda y aproximación a la pelota para el comportamiento *Go* y de búsqueda de la portería para *Shoot* se realicen en repetidas ocasiones hasta que se cumpla la percepción o mientras que la percepción se cumple. Por lo anterior, la configuración inicial de la prueba no es factor para el éxito de la misma y sólo afectara el tiempo de ejecución. Las pruebas se realizarán desde cinco configuraciones distintas para *Go* y desde tres para *Shoot*. Los detalles pueden consultarse en la sección 6.2.3 de este trabajo.

Para la prueba *Go & Shoot*, los requerimientos involucran añadir al subdiálogo *Actions* nodos adicionales a los enlistados en la tabla 4.1. Esta operación se debe hacer manualmente. El identificador para estos comportamientos es el nombre del archivo que contiene la información del comportamiento. Por ejemplo, si se tienen ya los comportamientos *Go*, *Shoot* y *Pass* almacenados en los archivos *SGo.txt*, *SShoot.txt* y *SPass.txt*, respectivamente, las acciones disponibles en el subdiálogo *Actions* son las que se presentan en la tabla 4.4 y el subdiálogo será el mostrado en la figura 6.15.

Finalmente, los requerimientos de la prueba *Go & Decision*, son cubiertas mediante la implementación de la instrucción de decisión.

Comando de voz	Nodo	Identificador	Acción
Stop	Stop	0	Detenerse.
Walk	Walk	1	Caminar.
Right	TurnR	9	Giro a la derecha.
Left	TurnL	10	Giro a la izquierda.
Kick	Kick	11	Patea la pelota.
Hold Ball	Hold	12	Toma la pelota.
Right with Ball	TRH	13	Giro a la derecha manteniendo la pelota.
Left with Ball	TLH	14	Giro a la izquierda manteniendo la pelota.
Go to Ball	Go -> Ball	20	Ir hacia la pelota y detenerse frente a ella.
Block	Block	22	Bloquear.
Go	GO	SGo.txt	Buscar, acercarse y tomar la pelota
Shoot	SHOOT	SShoot.txt	Buscar portería y tirar a gol
Pass	PASS	SPass.txt	Buscar robot rojo y pasar la pelota.

Tabla 4.4 Comandos de acción ampliados.

CAPÍTULO 5

IMPLEMENTACIÓN

En este capítulo se expone la implementación del sistema. Además, se presentan las herramientas utilizadas en la implementación del mismo.

5.1 SISTEMA DE INTERACCIÓN HUMANO-ROBOT

En este apartado se exponen las características de los equipos electrónicos utilizados en la implementación de este trabajo; además se presenta una explicación de la implementación de los módulos que constituyen el sistema de interacción.

5.1.1 Robot

5.1.1.1 Hardware y software del robot

El hardware del robot es el incluido desde su fabricación y corresponde a las siguientes características:

- Procesador MIPS de 64 bits RISC @ 576 MHz.
- 64MB de memoria RAM.
- Tarjeta de red inalámbrica IEEE 802.11b/g.

Además, se utilizó el sistema EK2008 para proveer ente otras cosas:

- Control de los motores.
- Procesamiento de imágenes.
- Comunicaciones con otros dispositivos, por ejemplo una computadora.
- Control de comportamientos.

5.1.1.2 EK2008

EK2008, a través de sus subsistemas, implementa los servicios de comunicación entre el robot y el sistema de interacción, así como la ejecución de acciones y el envío de respuestas. En los siguientes apartados se explica la implementación del proceso de recepción de los mensajes y el procesamiento que se realiza con cada uno; además, por ser relevante en el presente trabajo, se explica también como el sistema controla los movimientos en el robot.

5.1.1.2.1 Acciones y preguntas en el robot

EK2008 cuenta con un subsistema llamado MetaComm que habilita en el robot la recepción y envió de mensajes. El código de para la recepción y el envió de paquetes puede observarse en el cuadros 5.1.

```
void MetaComm::ReceiveCont(ANTENVMSG msg) {

    UDPEndpointReceiveMsg* receiveMsg =
      (UDPEndpointReceiveMsg*)antEnvMsg::Receive(msg);

    int index = (int)(receiveMsg->continuation);
    if (connection[index].state == CONNECTION_CLOSED) return;
    if (receiveMsg->error != UDP_SUCCESS) {
        OSYSLOG1((osyslogERROR, "%s : %s %d",
                  "MetaComm::ReceiveCont()",
                  "FAILED. receiveMsg->error", receiveMsg->error));
        Close(index);
        return;
    }
    MetaData* tempData = (MetaData*)(receiveMsg->buffer);
    if (validatePacket(tempData)) {

       memcpy(&metaCommData, tempData, sizeof(MetaData));

       notifyDataObservers();

    }
    connection[index].state    = CONNECTION_CONNECTED;
    connection[index].recvSize = METADATA_BUFFER_SIZE;
    Receive(index);
}


void MetaComm::SendCont(ANTENVMSG msg)
{
    OSYSDEBUG(("MetaComm::SendCont()\n"));

    UDPEndpointSendMsg* sendMsg =
      (UDPEndpointSendMsg*)antEnvMsg::Receive(msg);
```

```
    int index = (int)(sendMsg->continuation);
    if (connection[index].state == CONNECTION_CLOSED)
        return;
    if (sendMsg->error != UDP_SUCCESS) {
        OSYSLOG1((osyslogERROR, "%s : %s %d",
                  "MetaComm::SendCont()",
                  "FAILED. sendMsg->error", sendMsg->error));
        Close(index);
        return;
    }
    connection[index].state = CONNECTION_CONNECTED;
    connection[index].recvSize = METADATA_BUFFER_SIZE;
}
```

Cuadro 5.1 Fragmento de MetaComm

Una vez que el paquete ha sido recibido y verificado como válido, su contenido es enviado al subsistema EKMain (ver figura 4.12) en el que se verifica si el mensaje recibido corresponde a una acción o a una pregunta. Si es el primer caso, el método metaAction es ejecutado; en el otro caso, el método metaQuestion es ejecutado. El siguiente cuadro (5.2) muestra fragmentos del contenido de estos métodos.

```
void
EKMain::metaAction()
{
      if (metaMov != -1){
            switch (metaMov){
                  case 0:
                        //Alto
                        hold = false;
                        metaMov = -1;
                        motData.command = 0;
                        motData.x_walk  = 0;
                        motData.y_walk  = 0;
                        motData.ang_walk = 0;
                        motData.x_head  = 0;
                        motData.y_head  = 2700;
                        motData.z_head  = 0;
                        metaMov = -1;
                        SendToMotion();
                        break;
                  case 1:
                        //Adelante
                        hold = false;
                        motData.command = 0;
                        motData.x_walk  = 0;
                        motData.y_walk  = 5;
                        motData.ang_walk = 0;
                        SendToMotion();
                        break;
```

```
                  […]
                  case 9:
                        cntMeta++;
                        if (cntMeta < 6)
                        {
                              //GiroDerecha
                              hold = false;
                              motData.command = 0;
                              motData.x_walk  = 0;
                              motData.y_walk  = 0;
                              motData.ang_walk = -0.5;
                              SendToMotion();
                        }
                        else
                              metaMov = 0;
                        break;
                  case 10:
                        cntMeta++;
                        if (cntMeta < 6)
                        {
                              //GiroIzquierda
                              hold = false;
                              motData.command = 0;
                              motData.x_walk  = 0;
                              motData.y_walk  = 0;
                              motData.ang_walk = 0.5;
                              SendToMotion();
                        }
                        else
                              metaMov = 0;
                        break;
                  case 11:
                        //Tiro, movimiento completo
                        hold = false;
                        metaMov = -1;
                        OSYSPRINT (("Movimiento 102\n"));
                        movimiento(102);
                        break;
                  case 12:
                        //sujetar bola
                        hold = false;
                        metaMov = -1;
                        //OSYSPRINT (("Movimiento 101\n"));
                        bolaCerca = false;
                        movimiento(101);
                        break;
                  //[…]
            }
      }
}
void
EKMain::metaQuestion(int metaCons, char *resp)
{
      int r;
      strcpy(resp,"");
      switch(metaCons)
      {
      //   0   Pelota
      case 50:
```

```
            if (visData.objetos[0].visto)
                  aTexto(visData.objetos[0].area,resp);
            else
                  strcpy(resp,"-1");
            break;
      //    1   Portería Amarilla
      case 51:
            if (visData.objetos[1].visto && visData.objetos[1].cen_x
                  > 50 && visData.objetos[1].cen_x < 158)
                  aTexto(visData.objetos[1].area,resp);
            else
                  strcpy(resp,"-1");
            OSYSPRINT (("PAm: %s\n",resp));

            break;
      //[…]
      default:
            OSYSPRINT (("Comando no válido\n"));
            break;
      }

}
```

Cuadro 5.2 Fragmento de EKMain.

5.1.1.3.2 Implementación de las acciones en el robot

Una vez que se tiene completo el diseño de un movimiento, del software MEdit se puede sustraer los valores de los actuadores para cada una de las posiciones que asume el robot cuando lo ejecuta.

La implementación de estos movimientos o acciones se realiza en el módulo EKMotion del sistema EK2008. Este módulo actúa como una interfaz entre los actuadores y el resto del sistema. El módulo cuenta con un archivo llamado EKMoves.h que cumple la función de librería en la que se registran los valores de los actuadores de cada una de las posiciones necesarias para un movimiento.

Se utilizará como ejemplo el movimiento de tiro (*Kick*) para explicar como se realiza la implementación. El cuadro 5.3 muestra un fragmento del contenido del archivo EKMoves.h; el archivo contiene declaraciones de arreglos de números enteros en los cuales se almacenan los valores en grados de los actuadores. Los primero cuatro números definen la posición de la cabeza; los siguientes, en grupos de tres números, representan los valores para la pata delantera izquierda, trasera izquierda, delantera derecha y trasera derecha respectivamente.

```
const double EK_POS[] = {
      //Tilt2,Pan,Tilt2,Mouth,
      //LFLEG_J1, LFLEG_J2, LFLEG_J3
      //LRLEG_J1, LRLEG_J2, LRLEG_J3
      //RFLEG_J1, RFLEG_J2, RFLEG_J3
      //RRLEG_J1, RRLEG_J2, RRLEG_J3
      0,0,0,0,
      -20,5,100,
      -35,0,100,
      -20,5,100,
      -35,0,100
};


const double KICK_101_01[] = {
      45,0,-65,-3,
      35,-5,65,
      -25,0,70,
      35,-5,65,
      -25,0,70
};

const double KICK_101_02[] = {
      45,0,-65,-3,
      60,-5,30,
      60,0,35,
      60,-5,30,
      60,0,35
};
//[...]
```

Cuadro 5.3 Fragmento del archivo EKMoves.h

En el archivo EKMotion.cc está contenido todo el código que implementa la interfaz entre los actuadores y el resto del módulo. EKMotion proporciona a EKMain la posibilidad de invocar una acción mediante las funciones *SendToMotion* y *movimiento* de EKMain, tal como puede observarse en el cuadro 5.2. Cuando alguna de estas funciones es ejecutada, EKMotion recibe la información del movimiento solicitado a través de la función *Push*.

Push toma la información enviada desde EKMain y extrae de ésta el identificador del movimiento a realizar e invoca a la función *MoveToPos* enviando como parámetro dicho identificador.

MoveToPos (cuadro 5.4) verifica el identificador de la acción a realizar y dependiendo del valor de este se ejecuta otra función, por ejemplo *Kickb.*

```
MovingResult
EKMotion::MoveToPos(int cmd) {
      MovingResult result;
      subject[sbjMove]->ClearBuffer();

      switch (cmd) {
            case 110: result = Kickc(); break;
            case 101: result = Kicka(); break;
            case 102: result = Kickb(); break;
            case 103: result = Suelta(); break;
            case 105: result = Kick(); break;
            case 201: result = KickIzq(); break;
            case 301: result = KickDer(); break;
            case 401: result = Bloqueo(); break;
            case 402: result = BloqueoIzq(); break;
            case 403: result = BloqueoDer(); break;
            case 501: result = Festejo(); break;
            case 601: result = Paro();break;
            case 701: result = Receptor();break;
            case 702: result = PelotaReceptor();break;
            case 901: result = TiroR();break;
            case 801: result = Boca1();break;
            case 802: result = Boca2();break;
            case 210: result = HeadCenter();break;
            case 211: result = HeadLeft();break;
            case 212: result = HeadRight();break;
      }
      SendJointCommandVector();
    subject[sbjMove]->NotifyObservers();
      return result;
}
```

Cuadro 5.4 *MoveToPos* en EKMotion.

Las funciones invocadas desde *MoveToPos* se encargan de controlar la ejecución de cada una de las posiciones que integran el movimiento.

En *Kickb*, por ejemplo, un contador determina el momento de ejecución de cada una de las posiciones que integran el movimiento de tiro, este contador se incrementa en cada ciclo de movimientos. Entre mayor sea la separación entre los valores de dos posiciones en la estructura case, mayor será el tiempo en que se ejecutará el movimiento entre dichas posiciones. El cuadro 5.11 muestra la función *Kickb* que ejecuta la acción de tiro a portería, en esta función la variable *pos* determina que posición de la secuencia de movimientos debe ejecutarse.

```
MovingResult
EKMotion::Kickb() {
      MovingResult result;
      cont += 1;
      result = MOVING_CONT;
      switch (cont) {
            case 1:     pos = 3;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 50:    pos = 4;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 100:   pos = 5;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 150:   pos = 6;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 200:   pos = 7;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 250:   pos = 8;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 300:   pos = 1;
                  MoveHeadKick(pos);
                  MoveLegsKick(pos);
                  break;
            case 310:   cont = 0;
                  result = MOVING_FINISH;break;
      }
      return result;
}
```

Cuadro 5.5 Función Kickb en EKMotion.

En MoveHeadKick y MoveLegsKick, los valores de cada posición son copiados a un nuevo arreglo denominado *jointValue*. Tras esto, se invocan las funciones SetHeadJoints y SetLegJoints.

```
void
EKMotion::MoveHeadKick(int n) {
    OSYSPRINT(("\nMOTION: %i",n));
    double jointValue[NUMBER_HEAD_JOINTS];
```

```
    switch (n) {
      case 1: for (int i = 0; i<4; i++)
            jointValue[i] = (EK_POS[i]*PI)/180;
            break;
      case 3: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_01[i]*PI)/180;
            break;
      case 4: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_02[i]*PI)/180;
            break;
      case 5: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_03[i]*PI)/180;
            break;
      case 6: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_04[i]*PI)/180;
            break;
      case 7: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_05[i]*PI)/180;
            break;
      case 8: for (int i = 0; i<4; i++)
            jointValue[i] = (KICK_101_06[i]*PI)/180;
      //[...]
      }
      SetHeadJoints(jointValue);
      return;
}
void
EKMotion::MoveLegsKick(int n) {
    static double leg_joints[NUM_LEG_JOINT];
    switch (n) {
      case 1: for (int i = 0; i<12; i++)
            leg_joints[i] = (EK_POS[i+4]*PI)/180;
            break;
      case 2: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_01a[i+4]*PI)/180;
            break;
      case 3: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_01[i+4]*PI)/180;
            break;
      case 4: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_02[i+4]*PI)/180;
            break;
      case 5: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_03[i+4]*PI)/180;
            break;
      case 6: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_04[i+4]*PI)/180;
            break;
      case 7: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_05[i+4]*PI)/180;
            break;
      case 8: for (int i = 0; i<12; i++)
            leg_joints[i] = (KICK_101_06[i+4]*PI)/180;
            break;
      //[...]
      }
      SetLegJoints(leg_joints);
}
```

Cuadro 5.6 Funciones MoveHeadKick y MoveLegsKick.

Finalmente, las funciones SetHeadJoints y SetLegsJoints invocan a la función SetJoint que copia los valores de los actuadores a otro arreglo denominado joint_value; este último arreglo es el que encargado de almacenar los valores de los actuadotes que se desean fijar en el robot. En el siguiente ciclo de movimientos, EKMotion enviará dichos valores a los actuadotes y el robot tomara la configuración deseada.

```
int EKMotion::SetHeadJoints(double *x){
      for (int i = 0; i < NUMBER_HEAD_JOINTS; i++){
            SetJoint(HEAD_JOINT_INDEX[i], x[i]);
      }
      return 1;
}

int EKMotion::SetLegJoints(double *x){
      for (int i = 0; i < NUMBER_LEG_JOINTS; i++){
            SetJoint(LEG_JOINT_INDEX[i], x[i]);
      }
      return 1;
}

void EKMotion::SetJoint(int index, double val){
      if ((index >= 0) && (index < NUMBER_JOINTS)){
            joint_values[index] = int(1e6*val);
            send_joint_values[index] = true;
      }
      send_joints = true;
}
```

Cuadro 5.7 Funciones SetHeadJoints, SetLegJoints y SetJoint

5.1.2 Interacción Humano-Computadora

A continuación se presentan las características del hardware y software de la computadora utilizada para implementar el sistema de interacción.

5.1.2.1 Hardware

Los módulos ejecutados en el equipo de cómputo fueron desarrollados y probado sobre un hardware no especializado, una computadora portátil con las siguientes características:

- Procesador Intel Pentium M @ 1.6 GHz.
- 1.5GB de memoria RAM.
- Tarjeta de red inalámbrica IEEE 802.11b/g.

5.1.2.2 Software

Se utilizó como sistema operativo Windows XP SP3. El *software CSLU – Rapid Application Developer* (RAD) fue utilizado para la implementación de los diálogos de interacción. El paquete Microsoft Visual Studio 6.0 se usó como entorno de desarrollo del módulo HRI; éste último fue programado en el lenguaje de programación C/C++. Para la compilación del código EK2008 se utilizó el paquete Cygwin, un emulador de entorno Unix para la plataforma Windows.

5.1.2.3 Invocación del módulo HRI desde los diálogos de interacción

Las invocaciones al módulo HRI se realizan en los nodos de tipo *Action* (véase sección 2.7 del documento), estos nodos permiten la ejecución de código TCL. Mediante la instrucción *exec*, TCL puede iniciar la ejecución de programas.

HRI es siempre invocado sin importar la tarea a realizar: enviar acción, realizar pregunta o entrenar; para que el programa pueda distinguir entre estas diferentes operaciones, el primer parámetro de ejecución de HRI indica cual función debe ejecutar. La tabla 5.1 muestra los valores posible de este parámetro.

Valor de *mode*	Modo de ejecución
0	Acción.
1	Pregunta.
2	Instrucción de entrenamiento.
3	Instrucción de decisión.

Tabla 5.1 Modos de ejecución de HRI.

5.1.2.3.1 Acción

Cuando lo que se desea es ejecutar una acción, HRI recibe dos parejas de parámetros, cada pareja en su primer elemento contiene texto para identificar de qué parámetro se trata y como segundo elemento contiene el valor de ese parámetro.

La primera pareja de parámetros que se recibe define el modo de operación (*mode*) en que tiene que funcionar HRI.

En este caso, el valor de *mode* es el número cero. El segundo par de parámetros corresponde a la acción a realizar, el identificador de este parámetro es *acc* y el valor es un identificador de la

acción a realizar (véase tabla 5.2). La figura 5.1 muestra una captura del nodo *Exec_action*, en el diálogo de interacción básica, donde se puede observar la ejecución de HRI

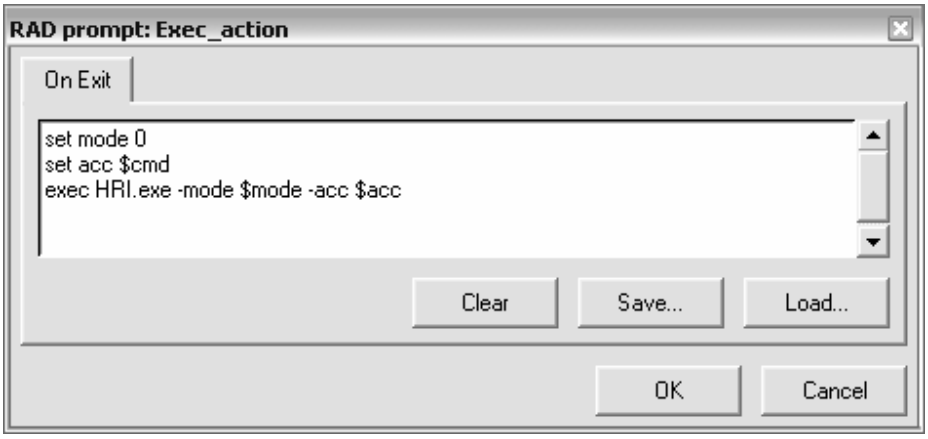


Figura 5.1 Configuración del nodo *Exec_action*.

5.1.2.3.2 Interrogación

La invocación de HRI para una pregunta es similar; en este caso, el valor del parámetro *mode* debe ser el número uno y el parámetro *ques* tendrá como valor algunos de los identificadores de la tabla 5.4. La instrucción *exec* se ejecuta dentro de una instrucción *catch*, esta instrucción captura la última impresión en pantalla realizada por un programa y lo almacena en una variable, en este caso la variable *resp*. La figura 5.2 muestra la invocación a HRI en el nodo *MQ* del subdiálogo *Questions.*

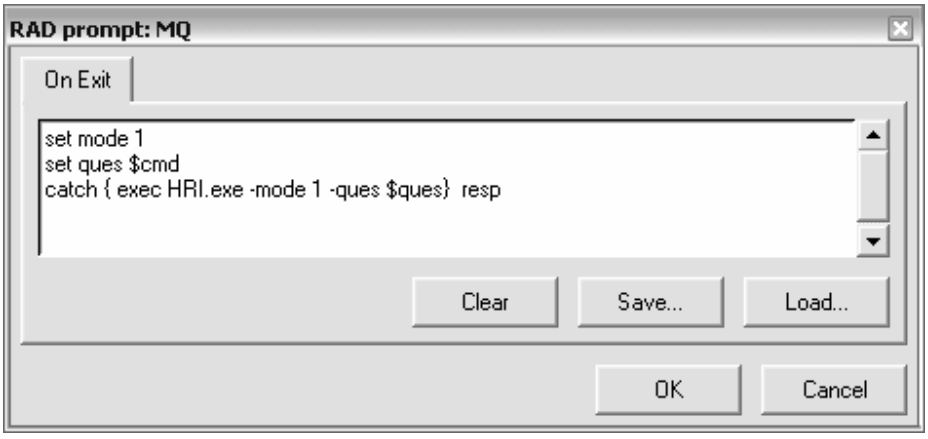


Figura 5.2 Configuración del nodo *Question.*

5.1.2.3.3 Entrenamiento

Cuando se desea ejecutar una instrucción de entrenamiento se necesitan pasar como parámetros el identificador de la acción a realizar (*acc)*, un condicionante (*dec*) que determina si la acción se realizará hasta o mientras se de la percepción y la finalmente el identificador de la percepción que condiciona toda la instrucción (*per*). Los valores para el parámetro de percepción son los mostrados como identificadores en la tabla 5.5. Cuando la instrucción que se está ejecutando es seguida por otra, estos parámetros son suficientes y la ejecución se realiza como en la figura 5.3.

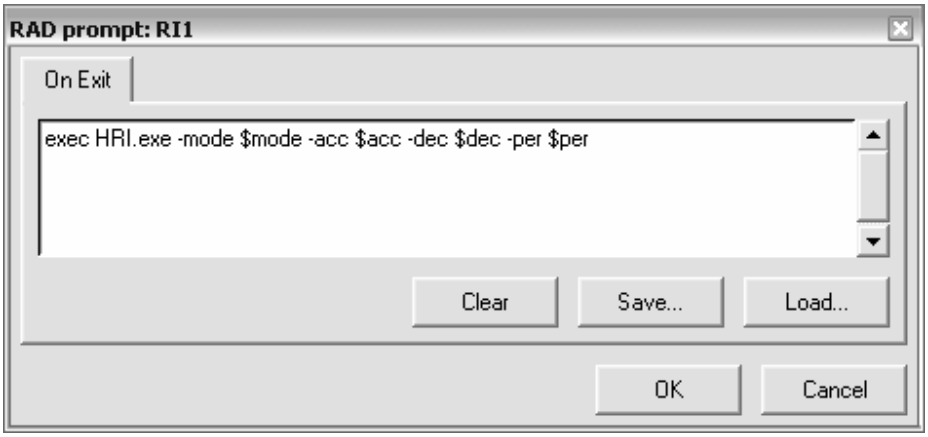


Figura 5.3 Configuración del nodo *RI1*.

Cuando se desea que el robot haga algo cuando la instrucción termine y antes de pasar a la siguiente, se debe agregar el parámetro *then* que corresponde a una acción. Un ejemplo es la configuración del nodo *RI2* en el diálogo de entrenamiento mostrado en la figura 5.4.

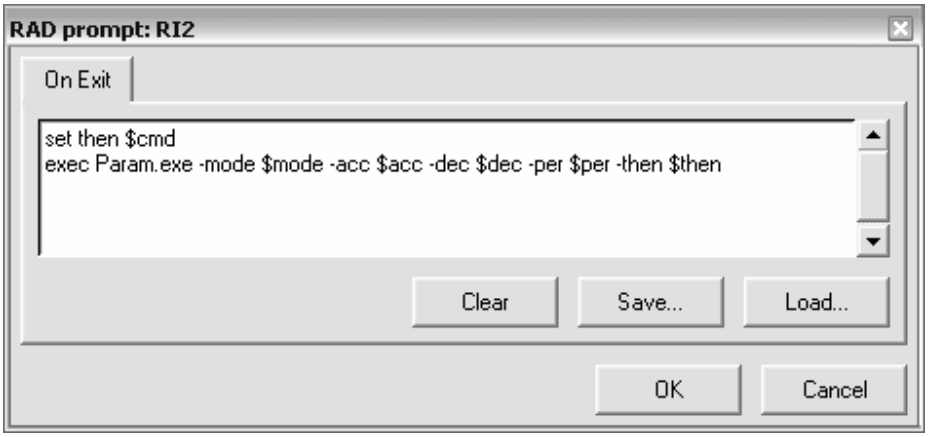


Figura 5.4 Configuración del nodo *RI2*.

Finalmente, cuando se ejecuta una instrucción de decisión, se pasa como parámetro los identificadores de la percepción que condiciona (per) y dos identificadores de acción (*then* y *else*). El primer identificador corresponde a la acción a ejecutar cuando la condición se cumple y el segundo es el de la acción que se ejecutará cuando no se cumple. La figura 5.5 muestra la configuración del nodo *RIF* en el diálogo de entrenamiento, este nodo se encarga de ejecutar la instrucción de decisión.

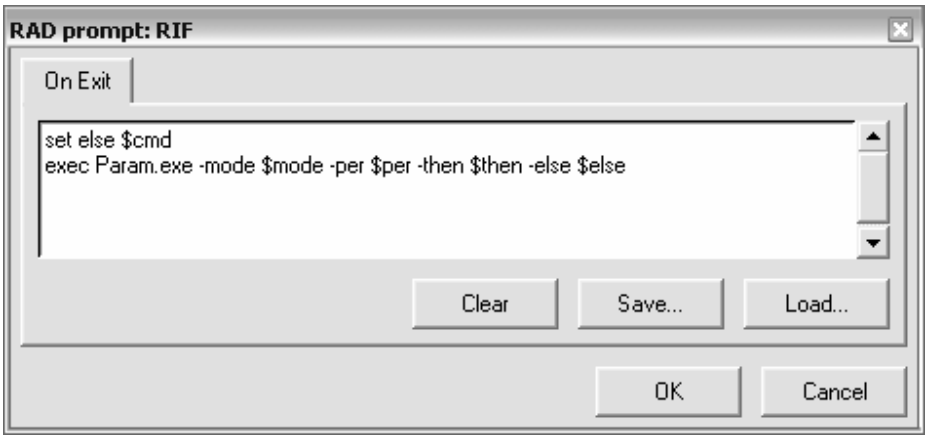


Figura 5.5 Configuración del nodo *RIF*.

5.1.3 Comunicación entre la computadora y el robot

En las siguientes secciones se expone la implementación de las comunicaciones inalámbricas desde la computadora hacia el robot.

5.1.3.1 Hardware y software

Para la comunicación inalámbrica entre la computadora y el robot se utilizaron las tarjetas de red IEEE 802.11b/g con que cuentan tanto la computadora portátil como el robot AIBO. En la implementación del socket UDP se utilizaron las librerías *Practical C++ Socket*.

5.1.3.2 Envío de mensajes al robot desde HRI

Para el envío de comandos, ya sean acciones o preguntas, hacia el robot se definió una estructura que contiene datos relevantes relacionados con el origen y destino de cada mensaje. La definición de la estructura de datos utilizada, llamada MetaData, se presenta en el cuadro 5.8.

```
struct MetaData {
    char     header[4];        // header to identify the structure
    int      version;          // version of the data structure
    int      playerNumber;     // who am I sending to?
    int      senderNumber;     // who am I?
    char     msg[255];
};
```

Cuadro 5.8 Estructura de datos *MetaData*.

MetaData cuenta con cinco elementos, el primero de ellos (*header*) es un arreglo de cuatro caracteres que cumple la función de identificación de la estructura, el valor de este arreglo debe ser “EKMe” para ser identificado correctamente tanto en el módulo HRI como en el sistema EK2008 del robot AIBO. El segundo elemento (*version*) es un entero que actúa como identificador de la versión, el valor debe ser 1. El campo *playerNumber* es un identificador que debe contener el número de robot destino. El elemento *senderNumber* identifica al remitente del mensaje. Para estos último dos campos los valores por omisión son 0 si se trata de la computadora o 1 si se trata del robot. Finalmente con el objeto de poder mandar una cantidad considerable de información hacia y desde el robot se definió el campo *msg*, un arreglo de caracteres con una longitud de 255 elementos.

Para implementar el socket UDP utilizado en las comunicaciones con el AIBO se utilizaron las librerías *Practical C++ Socket*, estas librerías son públicas y pueden descargarse de [PCS 2002]. El cuadro 5.9 muestra el código del *thread* que implementa la transmisión del mensaje.

```
class transmissionThread : public CThread {
      virtual DWORD Run( LPVOID /* arg */ ) {
            byte m[255];
            try {
                  memcpy(m,&meta,sizeof(meta));
                  sockT.sendTo(m,sizeof(MetaData),address,portT);
            } catch (SocketException &e) {
                  exit(1);
            }
            return NULL;
      }
};
```

Cuadro 5.9 Clase *transmissionThread*.

5.1.3.3 Recepción de mensajes provenientes del robot en HRI

Cuando el comando que se desea enviar corresponde a una pregunta, justo antes de iniciar el hilo encargado del envío del mensaje, se inicia un hilo de ejecución encargado de esperar la respuesta del robot. El código de la clase que implementa el *thread* de recepción es mostrado en el cuadro 5.10.

```
class receptionThread : public CThread {
    virtual DWORD Run( LPVOID /* arg */ ) {
      //respBuffer: respuesta del AIBO
      byte respBuffer[1024];
      MetaData* metaR;
      UDP_Recibido = false;
      Try {
          while(!UDP_Recibido){
            //Recibiendo el mensaje
            sockT.recvFrom(respBuffer, 1024, sAddress, sPort);

            //Verificando que sea la respuesta que se espera
            if (metaR->senderNumber == meta.playerNumber &&
                  metaR->playerNumber == meta.senderNumber &&
                  metaR->version == meta.version){
                        UDP_Respuesta = atoi(metaR->msg) > MIN_TRUE;
                        UDP_Recibido = true;
                  }
            }

      } catch (SocketException &e) {
            exit(1);
      }
      return NULL;
    }
};
```

Cuadro 5.10 Clase *receptionThread*.

Con el objetivo de hacer cómodo el envío de instrucciones y de preguntas, así como la recepción de las respuestas, se implemento la clase *AIBO*; ésta cuenta con un método llamado *action* y una función llamada *question*. El método envía al robot una instrucción. La función envía una pregunta y espera la respuesta del robot, la función entrega el resultado de la consulta. La clase AIBO hace transparente el uso de las clases *transmissionThread* y *receptionThread* al resto de la aplicación; así, en el resto del código del módulo HRI basta con invocar los métodos de la clase para establecer comunicación con el robot. El cuadro 5.5 muestra la implementación de la clase *AIBO*.

```
class AIBO{
public:
	transmissionThread TxUDP;
	receptionThread RxUDP;
	AIBO();
	void action(char *cmd);
	bool question(char *cmd);

};
AIBO::AIBO(){
	strcpy(meta.header,"EKMe");
	meta.version = 1;
	meta.playerNumber = 4;
	meta.senderNumber = 0;
	try
	{
		sockT.setLocalPort(portT);
	} catch (SocketException &e) {
		cerr << e.what() << endl;
	}

}
void AIBO::action(char *cmd){
	strcpy(meta.msg,cmd);
	TxUDP.Start(NULL);
	Sleep(400);
}
bool AIBO::question(char *cmd){

	int version = 0;
	UDP_Recibido = false;
	strcpy(meta.msg,cmd);
	cout<<"Pregunta: "<<cmd<<endl;
	while(!UDP_Recibido){ //1
		//Mientras no se recibe respuesta
		if (version > 100)
			version = 0;

		meta.version = version;
		version ++;

		RxUDP.Start(NULL);
		TxUDP.Start(NULL);
		Sleep(400);
		if (!UDP_Recibido){
			RxUDP.Stop();
			UDP_Recibido = false;
		}
	}
	if (UDP_Respuesta)
		cout<<"Respuesta: 1";
	else
		cout<<"Respuesta: 0";
	return UDP_Respuesta;
}
```

Cuadro 5.11 Clase *AIBO*.

5.2 ACCIÓN, INTERROGACIÓN Y ENTRENAMIENTO

HRI es una aplicación desarrollada en C/C++ que actúa como conexión entre la interfaz de voz y el robot. El software ofrece los servicios de comunicación con el robot AIBO así como el almacenamiento de comportamientos y la reproducción posterior de los mismos. Algunas de sus funciones son:

- Envío de mensajes de acción al robot.
- Envío de mensajes de interrogación al robot.
- Recepción de respuestas del robot.
- Almacenamiento de nuevos comportamientos.
- Reproducción de movimientos almacenados.

La primera tarea que realiza la aplicación HRI es determinar el modo de operación (véase tabla 5.1) para así ejecutar el método adecuado. El cuadro 5.1 muestra el método principal (*main*) de la aplicación. Puede notarse que los métodos y funciones *actions*, *question*, *instruction* y *instIF* son ejecutadas según el modo de operación.

```
void main(int argc,char *argv[]){
      strcpy(Then,"NULL");
      strcpy(Else,"NULL");
      get_parameters(argc,argv);
      switch(Mode){
            case 0:
                  //cout<<"Action"<<endl;
                  actions(Acc);
                  break;
            case 1:
                  //cout<<"Question"<<endl;
                  question();
                  break;
            case 2:
                  //cout<<"Instruction"<<endl;
                  instruction();
                  break;
            case 3:
                  instIF();
                  //cout<<"If"<<endl;
                  break;
      }
}
```

Cuadro 5.12 Método *main* de HRI.

5.2.1 Acción

Cuando se desea ejecutar una acción se pueden presentar dos casos, en el primero de ellos la acción corresponde a algún comportamiento básico que el robot puede ejecutar, estos movimientos fueron expuestos en la tabla 4.1; si este es el caso, el parámetro que HRI recibe es un número entero que identifica la acción a ejecutar, este identificador es enviado al robot utilizando la clase AIBO. El cuadro 5.13 muestra la implementación del método *actions* mientras que la tabla 5.2 muestra los identificadores que HRI recibe para cada una de las acciones que desde los diálogos de interacción se pueden ordenar enviar.

```
void actions(char *acc){
      if (acc[0] != 'S')
            aibo.action(acc);
      else
            executeSequence(acc);
}
```

Cuadro 5.13 Método *actions*.

Comando de voz	Identificador	Acción
Stop	0	Detenerse.
Walk	1	Caminar.
Right	9	Giro a la derecha.
Left	10	Giro a la izquierda.
Kick	11	Patea la pelota.
Hold Ball	12	Toma la pelota.
Right with Ball	13	Giro a la derecha manteniendo la pelota.
Left with Ball	14	Giro a la izquierda manteniendo la pelota.
Go to Ball	20	Ir hacia la pelota y detenerse frente a ella.
Block	22	Bloquear.

Tabla 5.2 Identificadores para los comandos de acción.

Si la acción a ejecutar es un comportamiento entrenado previamente y almacenado en un archivo, HRI recibe como parámetro el nombre de dicho archivo. Para discriminar fácilmente ente acciones simples y comportamientos, se impuso como norma comenzar los nombres de los archivos con una letra ‘S’. El método *actions* ejecuta el método *executeSequence* (cuadro 5.14); éste invoca al método *addNodesFromFile*, encargado de leer el archivo que contiene el comportamiento almacenado.

Normalmente, el archivo contiene información en cuarteros, cada grupo de cuatro líneas está compuesto por un identificador de la acción a realizar, un identificador de la condición (“until” o “while”), un identificador de una percepción y, finalmente, un identificador para la acción a realizar al final de la instrucción. Cuando la acción que se ejecutará corresponde a un comportamiento ya almacenado, en lugar del identificador de acción se proporciona el nombre del archivo que contiene la secuencia a ejecutar.

Finalmente, cuando se encuentra una letra ‘D’ en el archivo quiere decir que le siguen tres identificadores que definen una estructura de decisión: el primero elemento es el identificador de una percepción, el segundo la acción asociada a una percepción positiva del robot y, finalmente, el tercer elemento es una acción asociada a una percepción negativa del robot. El cuadro 5.14 muestra un ejemplo de un archivo que almacena un comportamiento.

SGo.txt	SGoShoot.txt	SGoDecision.txt
9 0 50 Next 20 0 59 12	SGo.txt 0 59 Next SShoot.txt 0 52 Next	SGo.txt 0 59 Next D 61 SPass.txt SShoot.txt

Tabla 5.3 Ejemplos de archivos de comportamientos almacenados.

Cuando cada uno de estos conjuntos de elementos es leído se forma un nodo que es insertado en una lista ligada.

La figura 5.6 muestra el diagrama de flujo del método *addNodeFromFile*, este método es el encargado de leer los archivos que contienen la información del comportamiento a ejecutar. Tras la ejecución de *addNodeFromFile* se obtiene una lista ligada.

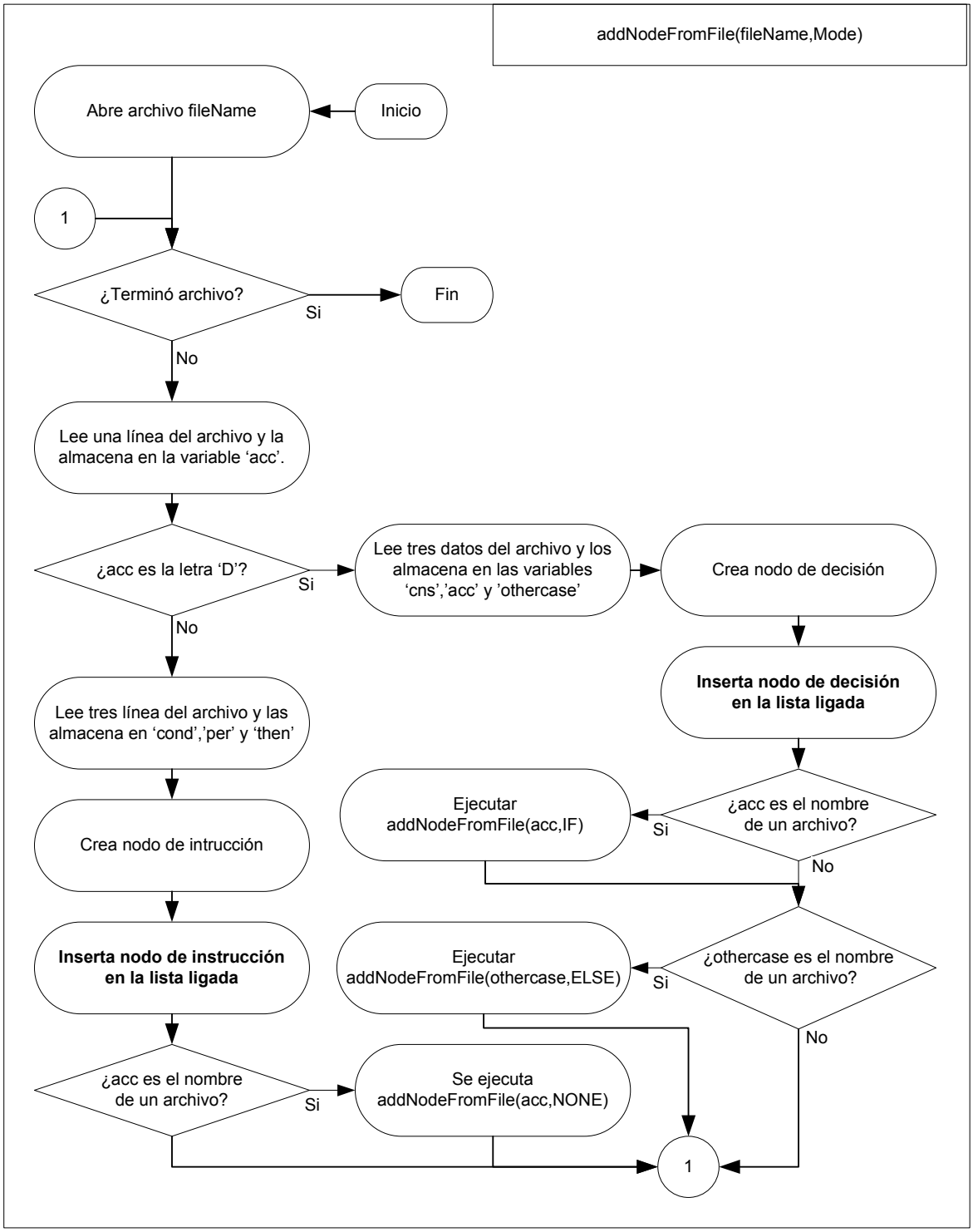


Figura 5.6 Diagrama de flujo del método *addNodeFromFile*.

La lista ligada obtenida representa un autómata de estados finitos cuyos estados y transiciones determinan el comportamiento del robot. La figura 5.7 ilustra el autómata de estados finitos que se construye a partir del archivo SGo.txt.

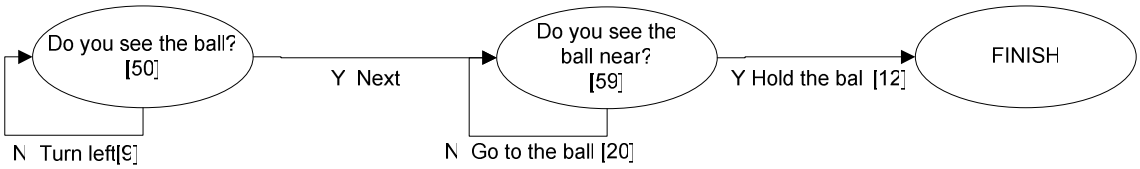


Figura 5.7 Lista ligada para el comportamiento *Go*.

En este comportamiento *Go*, las transiciones en el primer estado están condicionados por la identificación de la pelota; cuando la pelota no está a la vista del robot, éste debe girar a la izquierda pero si la pelota es visible el comportamiento avanza al siguiente estado. En el segundo estado la condición de transición es que la pelota esté cerca del robot; cuando no lo esta, el robot debe acercarse a la pelota y cuando se encuentre a su alcance, el robot debe intentar tomarla y así finalizar el comportamiento.

La figura 5.8 el autómata de estados finitos que se crea cuando se usa el archivo SGoShoot.txt. En esta figura el primer y cuarto nodo derivan directamente de las instrucciones que se encuentran en el archivo SGoShoot.txt. El segundo y tercer nodo corresponden a la máquina de estados mostrada en la figura 5.7, es decir, a la derivada del archivo SGo.txt. Finalmente, el quinto nodo corresponde a la representación del contenido del archivo SShoot.txt. En esta figura es posible apreciar que el primer y cuarto nodo sólo sirven para desencadenar los comportamientos almacenados en otros archivos pues, sin importar cual sea el resultado de la pregunta del nodo, siempre se avanza al siguiente nodo.

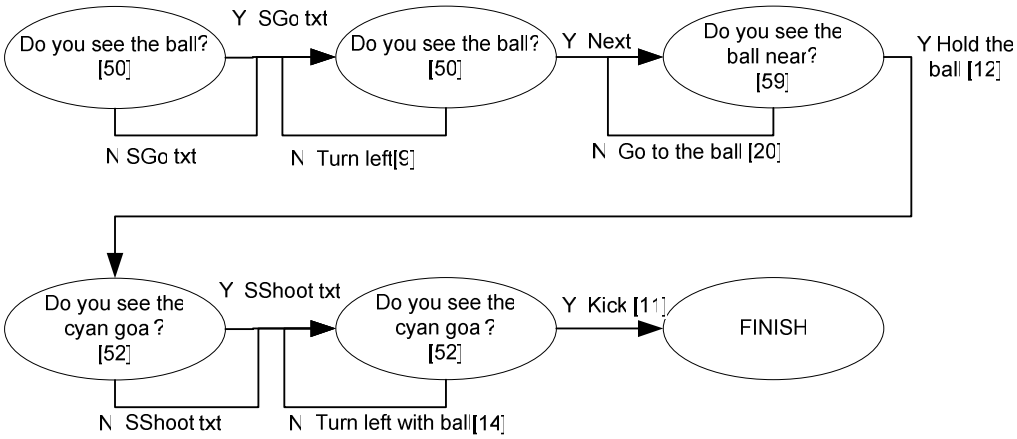


Figura 5.8 Lista ligada del comportamiento *Go & Shoot*.

Finalmente, en la figura 5.9 se observa la lista ligada que se forma con el archivo SGoDecision.txt. Los primeros tres estados corresponden a la ejecución del comportamiento *Go*. El cuarto nodo corresponde a la estructura de decisión contenida en el archivo; cuando la respuesta a la pregunta es afirmativa, se avanza a los nodos que definen la acción a realizar cuando la respuesta es positiva, dichos nodos serán llamados nodos afirmativos. Cuando la respuesta en el cuarto nodo no es positiva, la lista es recorrida ignorando todos los nodos afirmativos hasta encontrar los nodos denominados negativos (los que corresponden a la acción a ejecutar cuando la respuesta es negativa). La acción de saltar los nodos positivos está representada por la flecha punteada que inicia en el cuarto nodo; sin embargo, dicho apuntador no existe en el autómata de estados finitos ya que éste está implementado en una lista ligada.

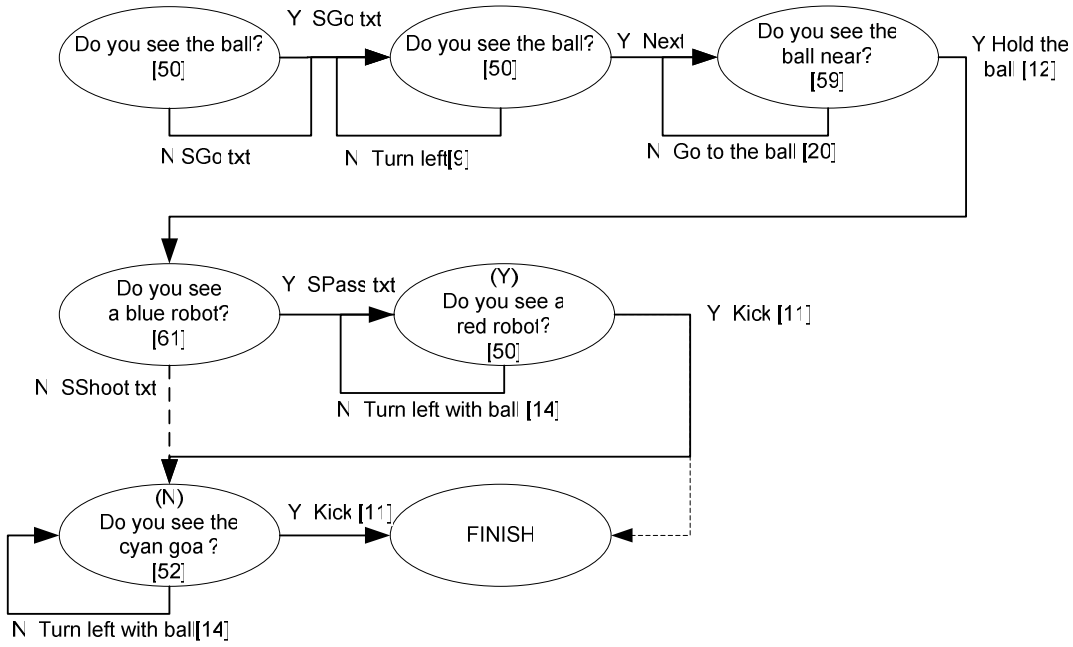


Figura 5.9 Lista ligada para el comportamiento *Go & Decision*.

El módulo *executeSequence* es el encargado de recorrer el autómata y ejecutar cada una de las instrucciones contenidas en ella de acuerdo con las respuestas recibidas por el robot. Para cada nodo, *executeSequence* verifica si se trata de un nodo de instrucción o uno de decisión; cuando se trata de un nodo de instrucción se utiliza el método *procNode* para enviar la pregunta al robot, esperar la respuesta, y enviar al robot la acción que debe realizar según su respuesta recibida. Cuando la respuesta del robot implica avanzar al siguiente nodo en la lista ligada, *procNode* ejecuta la instrucción correspondiente y devuelve el control a *executeSequence*. Por

otro lado, cuando se trata de un nodo de decisión, *executeSequence* consulta al robot y dependiendo de la respuesta que devuelva, el método ejecuta los nodos afirmativos o negativos. Tanto los nodos afirmativos como negativos cuentan con una variable que los identifica como tales haciendo posible recorrer la lista y ejecutar sólo los nodos deseados. La figura 5.10 ilustra el diagrama de flujo del método *executeSequence*.

El cuadro 5.14 muestra el código del método *procNode*, dicho nodo procesa cada nodo de la lista ligada. Básicamente, el método envía la pregunta al robot y la recibe utilizando la clase AIBO, una vez que obtiene la respuesta verifica si ante esa respuesta debe avanzar al siguiente nodo de la lista ligada. Si la acción que tiene que ejecutar el robot es una acción simple ésta es enviada al robot.

```
void procNode(){
      int version=0;
      bool resp;
      char action[255];

      cout<<"Pregunta: "<<(*iSeq).question<<endl;

      //Se envía la pregunta y espera respuesta
      resp = aibo.question((*iSeq).question);

      cout<<"Respuesta: "<<resp<<endl;

      //Enviando acción según sea la respuesta:
      if (resp) { //1
            strcpy(action,(*iSeq).cmdIfTrue);
            if ((*iSeq).nextIfTrue )
                  iSeq++;
      } //1
      else{ //2
            strcpy(action,(*iSeq).cmdIfFalse);
            if ((*iSeq).nextIfFalse )
                  iSeq++;
      } //2
      if (action != 'S')      { //3
            cout<<"Acción: "<<action<<endl;
            aibo.action(action);
            Sleep(500);
      } //3

}
```

Cuadro 5.14 Módulo *procNode*.

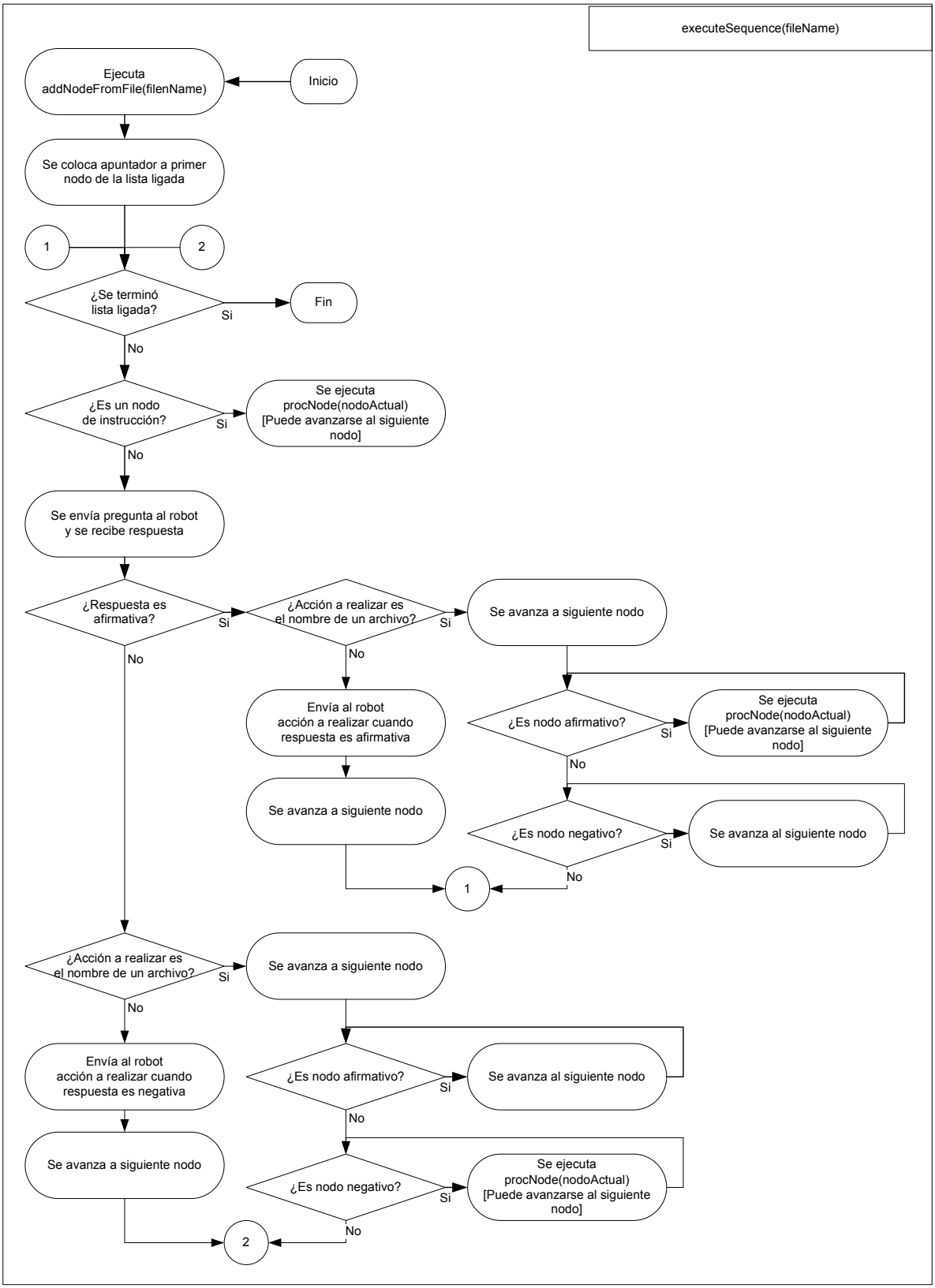


Figura 5.10 Diagrama de flujo del método *executeSequence*.

5.2.2 Interrogación

Cuando se desea que el robot conteste a alguna pregunta, el método *question* se ejecuta; este método utiliza la clase AIBO para enviar la pregunta al robot y recibir la respuesta. Una vez que la respuesta ha sido recibida, el método imprime dicho resultado para que el diálogo de interacción que invocó a HRI pueda obtener la respuesta. El cuadro 5.15 muestra la implementación del método *question*.

```
void question(){
      bool Respuesta=false;
      Respuesta = aibo.question(Ques);
      if (Respuesta)
            cout<<"1";
      else
            cout<<"0";
}
```

Cuadro 5.15 Método *question*.

La tabla 5.4 muestra los identificadores de las preguntas que se pueden realizar al robot. Como se puede observar en la implementación de la clase AIBO (sección 5.1.3), la respuesta del robot a las preguntas es un número mayor a 0 si el objeto es detectado y menor a cero cuando dicho objeto no es identificado por el robot. Para usos prácticos, la clase devuelve un valor booleano indicando si el objeto ha sido visto o no.

Comando de voz	***Identificador***	***Id. Respuesta***
Do you see the ball?	50	Ve la pelota – *True*, No vela pelota - *False*
Do you see the yellow goal?	51	Ve la portería amarilla – *True*, No la ve - *False*
Do you see the cyan goal?	52	Ve la portería cian – *True*, No la ve – *False*
Do you see the ball near?	59	Ve la pelota cerca – *True*, No la ve cerca – *False*
Do you see a red robot?	60	Ve robot rojo – *Trae*, No lo ve – *False*
Do you see a blue robot?	61	Ve robot azul – *True*, No lo ve – *False*

Tabla 5.4 Identificadores para los comandos de interrogación.

Las llamadas percepciones cuentan con identificadores idénticos a la pregunta que les corresponde, dichos identificadores se pueden observar en la tabla 5.5

Comando de voz	*Identificador*
You see the ball	50
You see the yellow goal	51
You see the cyan goal	52
You see a red robot	57
You see a blue robot	60
You see the ball near	61

Tabla 5.5 Identificadores de percepciones.

5.2.3 Entrenamiento

5.2.3.1 Almacenamiento de instrucción de entrenamiento

Para registrar una instrucción de entrenamiento, el programa recibe como parámetros los identificadores de la acción a realizar, del condicionante (“until” o “while”) y de la percepción que se desea asociar a la acción; además, puede incluirse otro identificador para una acción a realizar cuando se termina la instrucción de entrenamiento. La figura 5.11 muestra el diagrama de flujo del método *instruction*, este método se encarga de registrar los parámetros que HRI recibe y de ejecutar la instrucción de entrenamiento.

El método *instruction* abre el archivo Sequence.txt en modo de escritura, luego verifica si recibió como parámetro el identificador para una acción a ejecutar cuando termine la instrucción de entrenamiento; si éste no existe, almacena en la variable correspondiente la palabra “Next” para indicar que se salta a la siguiente instrucción de entrenamiento. Luego, *instruction* verifica si el condicionante recibido corresponde a UNTIL o a WHILE. Después, el método escribe en el archivo el identificador de la acción a realizar, luego un “0” si el condicionante fue UNTIL o un “1” si fue WHILE, en la siguiente línea se escribe el identificador de la percepción que condiciona la ejecución de la acción; finalmente, se escribe en otra línea el identificador de la acción a realizar cuado la instrucción termine, es decir, el valor de la variable *Then*. Una vez que se tiene la información en el archivo, se procede a consultar al robot acerca de la percepción. Si el condicionante fue UNTIL, entonces se enviará al robot la orden de ejecutar la acción hasta que la respuesta del robot a la percepción sea positiva. En caso contrario, cuando el condicionante es WHILE, se enviará al robot la orden de ejecutar la acción mientras la respuesta del robot sea positiva. En ambos casos, cuando se escapa del bucle, se envía al robot el identificador almacenado en la variable Then. El cuadro 5.16 muestra el código del método *instructions*.

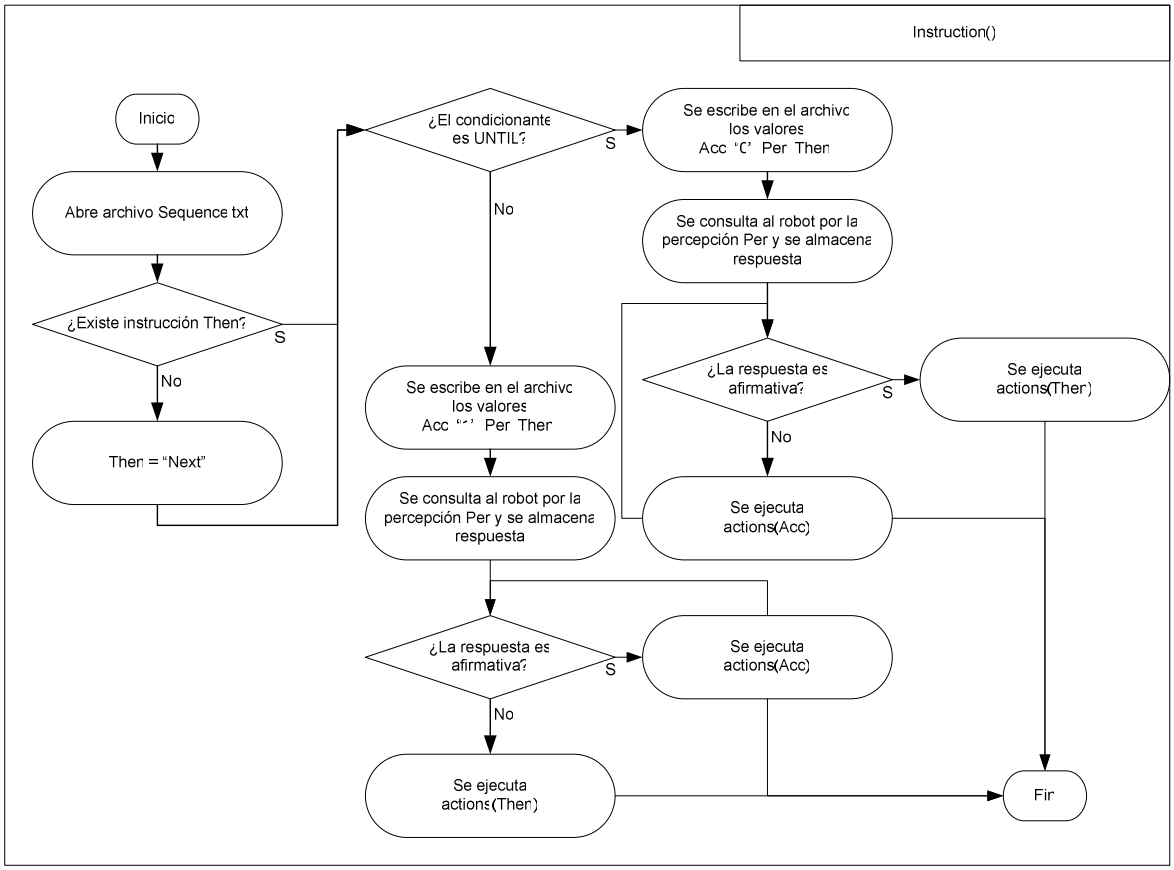


Figura 5.11 Diagrama de flujo de *instruction*.

```
void instruction(){
      bool Respuesta=false;
      FILE *f;
      f = fopen("Sequence.txt","a");
      if (!strcmp (Then,"NULL")){ //Then == NULL
            strcpy (Then,"Next");
      if (Dec == UNTIL){
            fprintf( f, "%s\n",Acc);fprintf( f, "%s\n","0");
            fprintf( f, "%s\n",Per);fprintf( f, "%s\n",Then);
            Respuesta = aibo.question(Per);
            While (¡Respuesta)
                  actions(Acc);
            Actions(Then);
      }
      else{
            //WHILE
            fprintf( f, "%s\n",Acc);fprintf( f, "%s\n","1");
            fprintf( f, "%s\n",Per);fprintf( f, "%s\n",Then);
            Respuesta = aibo.question(Per);
            While (Respuesta)
                  actions(Acc);
            Actions(Then);
      }
      fclose(f);
}
```

Cuadro 5.16 Método *intructions*.

5.2.3.2 Almacenamiento de una instrucción de decisión

Cuando se opera en el modo de instrucción de decisión, HRI obtiene como parámetros identificadores de la percepción a que hace referencia la instrucción de decisión y de las acciones a realizar cuando se cumple y cuando no se cumple la percepción. El método *instIF* se encarga de almacenar dicha instrucción en el archivo "Sequence.txt". En el archivo se almacenan primero una letra 'D' que indica que en ese punto comienza una decisión, luego se escribe el identificador de la percepción seguido del de la acción a realizar cuando la percepción ocurra; finalmente, se escribe en el archivo el identificador de la acción a realizar cuando no se cumpla la percepción.

Una vez que la información ha sido registrada, el mismo método consulta mediante la clase AIBO la percepción y ejecuta la acción que corresponda a la respuesta obtenida del robot. El cuadro 5.17 muestra el código del método, mientras que la figura 5.12 ilustra el diagrama de flujo para el mismo método.

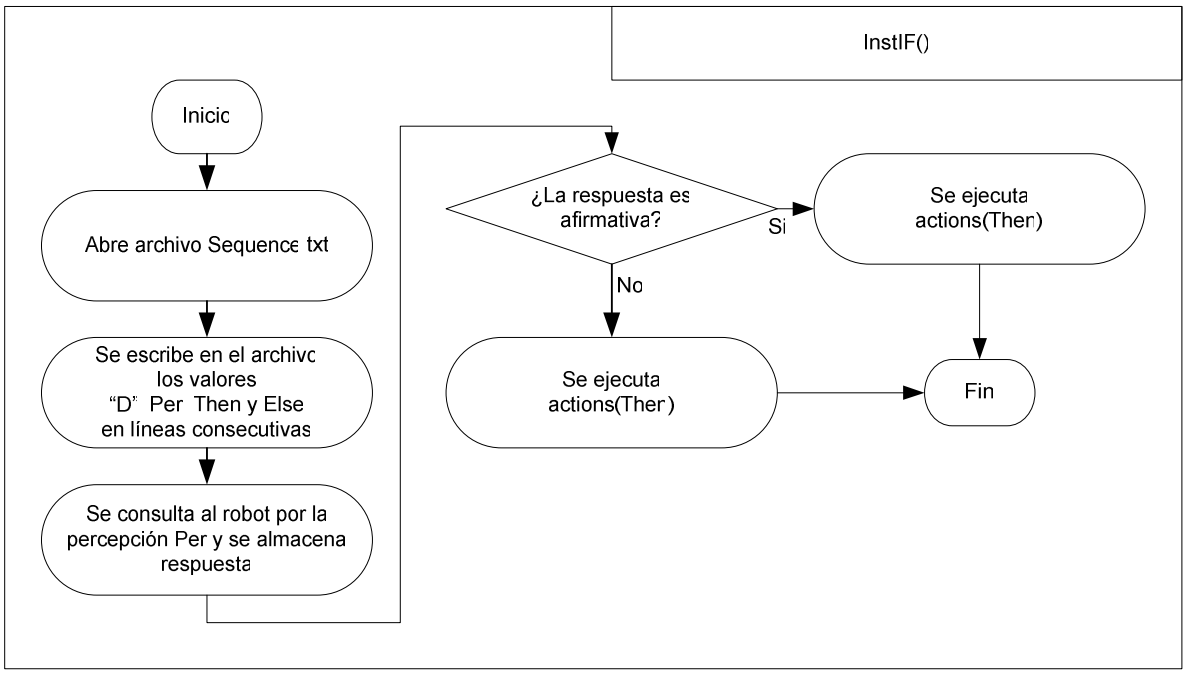


Figura 5.12 Diagrama de flujo de *intsIF*.

```
void instIF(){
      bool Respuesta=false;
      FILE *f;
      f = fopen("Sequence.txt","a");
      fprintf( f, "%s\n","D");
      fprintf( f, "%s\n",Per);
      fprintf( f, "%s\n",Then);
      fprintf( f, "%s\n",Else);

      Respuesta = aibo.question(Per);

      if (Respuesta){
            cout<<"Then: "<<Then<<endl;
            actions(Then);
      }
      else{
            cout<<"Else: "<<Else<<endl;
            actions(Else);
      }
      fclose(f);
}
```

Cuadro 5.17 Método *intIF*.

CAPÍTULO 6

PRUEBAS Y RESULTADOS

Una vez terminada la implementación del sistema, se realizaron una serie de pruebas con el fin de evaluar tanto el correcto funcionamiento de éste como el desempeño que ofrece al usuario. Este capítulo presenta los escenarios de prueba y los resultados obtenidos.

Se realizaron dos grupos de pruebas: el primer grupo corresponde a pruebas individuales a los módulos del sistema, es decir, primero se probó que el robot ejecutara los comportamientos deseados, luego se probó que los diálogos de entrenamientos funcionaran correctamente y finalmente se verificó que las comunicaciones entre la computadora y el robot trabajaran adecuadamente.

El segundo grupo de pruebas se realizó con el objetivo de evaluar el desempeño del sistema en las tres capacidades principales que debe ofrecer al usuario: acciones, preguntas y entrenamiento. Primero, se midió el tiempo de respuesta del sistema; para ello, se utilizó el diálogo de interacción básica para indicar al robot la ejecución de una serie de acciones. Otra prueba midió el tiempo de respuesta del sistema ante un conjunto de preguntas.

Finalmente, las pruebas se centraron en los resultados de los entrenamientos. Se probó entrenando dos comportamientos complejos partiendo de acciones simples; el primero de esto comportamientos consistió en indicar al robot que buscara la pelota y se acercara a ella para tomarla con las patas delanteras; el segundo empieza justo donde el anterior termina, con el robot en posesión de la pelota, y consistió en instruir al robot para que buscara la portería y efectuara un tiro hacia ella cuando la tuviera a la vista. Una vez desarrollados estos comportamientos se desarrolló un entrenamiento donde se generó un comportamiento complejo derivado de la unión de los dos últimos. Por último, se probó la estructura de decisión entrenando al robot para que tome la decisión de pasar la pelota a otro robot o tirar a gol dependiendo de la detección de un obstáculo enfrente de él.

6.1 SISTEMA DE INTERACCIÓN HUMANO-ROBOT

En esta sección se presenta una breve descripción del procedimiento de prueba del robot, de los diálogos de interacción y del módulo HRI. Cada uno de estos sistemas fue probado por separado para verificar el correcto funcionamiento y garantizar que las pruebas presentadas en la sección 6.2 se realizaran correctamente.

6.1.1 Robot

Esta serie de pruebas buscaba garantizar que el comportamiento del robot fuera el adecuado ante el envío de comandos de acción o de interrogación. Para realizar estas pruebas se utilizó el software EKMetaComm, dicho software fue implementado para probar comunicaciones entre la computadora y el robot AIBO durante el desarrollo de [WEI 2006].

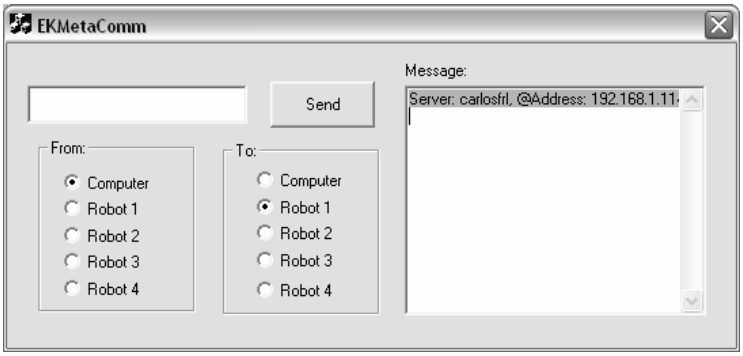


Figura 6.1 Software EKMetaComm.

La primera prueba consistió en enviar mediante la aplicación EKMetaComm una serie de instrucciones al robot, estas instrucciones fueron ejecutadas casi de manera inmediata. Para enviar las instrucciones, EKMetaComm provee de un cuadro de texto para colocar el identificador de la acción que se desea que el robot realice; el software cuenta con un botón para enviar la instrucción al robot. En la parte derecha de la pantalla, es posible observar un registro de los comandos enviados; en este caso, como lo que se envió son órdenes de acción, no se registró ninguna respuesta del robot. Los identificadores de la acciones fueron mostrados en la tabla 5.2.

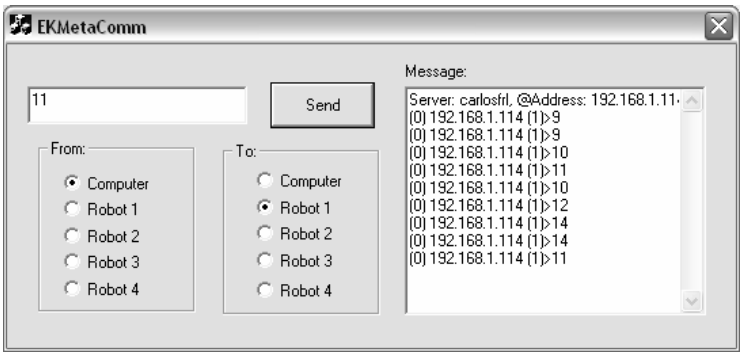


Figura 6.2 EKMetaComm enviando instrucciones de acción.

La tabla 6.1 muestra los resultados obtenidos al enviar los comandos de acción al robot. El robot ejecuta la totalidad de los comandos siempre y cuando los comandos se envíen cuando el robot no está ejecutando otra acción; cuando lo anterior no ocurre el robot ignora la instrucción recibida. Una excepción es la acción de detenerse (identificador "0"), ésta interrumpe cualquier ejecución de acción y coloca al robot en la posición estándar de caminata.

Acción	Identificador	Robot
Girar a la derecha	9	Ejecutó el movimiento
Girar a la derecha	9	Ejecutó el movimiento
Girar a la izquierda	10	Ejecutó el movimiento
Tirar	11	Ejecutó el movimiento
Girar a la izquierda	10	Ejecutó el movimiento
Tomar pelota	12	Ejecutó el movimiento
Girar a la izq. con pelota	14	Ejecutó el movimiento
Girar a la izq. con pelota	14	Ejecutó el movimiento
Tirar	11	Ejecutó el movimiento
Ir a pelota	20	Ejecutó el movimiento
Detenerse	0	Se detuvo
Caminar	1	Ejecutó el movimiento
Detenerse	0	Se detuvo
Bloquear	22	Ejecutó el movimiento

Tabla 6.1 Resultados de la prueba del robot con acciones.

Para realizar las pruebas de comunicación con preguntas, se utilizó una configuración en la que el robot pudiera ver tanto la portería color cian como al robot rojo y a la pelota. Con este escenario se envió una serie de comandos de interrogación al robot y se evaluó si las respuestas recibidas desde el robot correspondían a los objetos que pueden ser vistos por él en este escenario. La figura 6.3 muestra la configuración descrita.

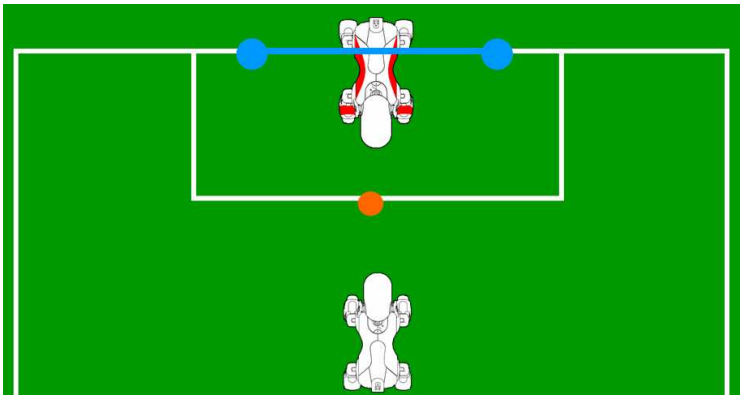


Figura 6.3 Configuración para prueba de preguntas al robot.

Ante el envío de un comando de interrogación, el robot responde con un número entero que representa la cantidad de píxeles que percibe del objeto. Si el robot responde con un número inferior a cero significa que no percibe el objeto. La figura 6.4 muestra la pantalla de EKMetaComm donde es posible observar como a una pregunta sigue una respuesta del robot.

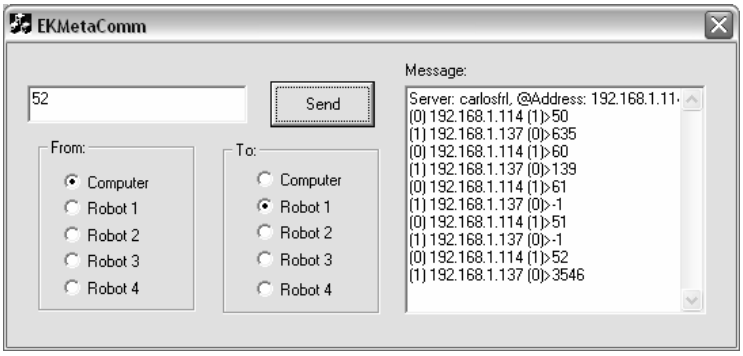


Figura 6.4 Respuestas del robot en EKMetaComm.

La tabla 6.2 muestra los resultados obtenidos de la prueba con comandos de interrogación.

Pregunta	Identificador	Respuesta	Resultado
¿Ve la pelota?	50	635	Correcto
¿Ve un robot rojo?	60	139	Correcto
¿Ve un robot azul?	61	-1	Correcto
¿Ve la portería amarilla?	51	-1	Correcto
¿Ve la portería cian?	52	3546	Correcto
¿Ve la pelota cerca?	59	-1	Correcto
¿Ve la pelota?	50	652	Correcto

Tabla 6.2 Resultados de la prueba del robot con pregunta.

6.1.2 Interacción Humano-Computadora

Para realizar pruebas a los diálogos de interacción, el código TCL de invocación a HRI debe ser retirado o comentado en todos los nodos tipo *Action* en que esté presente. La figura 6.5 muestra el código TCL comentado dentro de un nodo tipo *Action*.

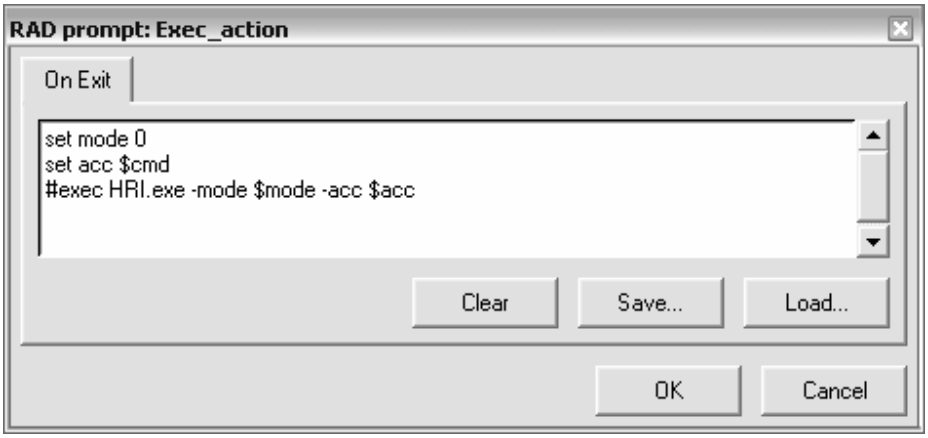


Figura 6.5 Configuración del nodo *Exec_action* para pruebas.

El carácter '#' al inicio de la tercera línea indica que esa línea no será ejecutada cuando el flujo del diálogo de interacción alcance al nodo.

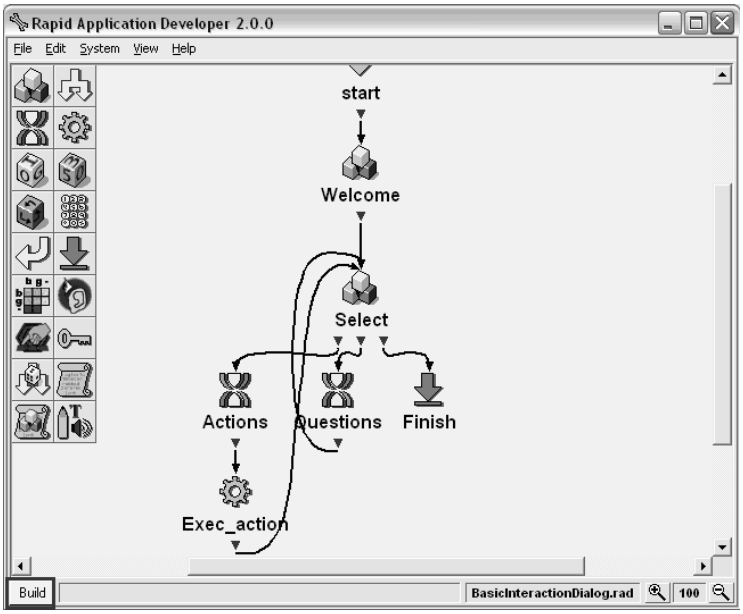


Figura 6.6 Ubicación del botón *Build*.

Una vez que se han comentado la totalidad de los nodos donde se invoca al programa HRI, el diálogo de interacción debe ser primero construido pulsando sobre el botón *Build* en la parte inferior izquierda de la ventana, al hacer lo anterior el nombre del botón cambia a *Run*; pulsando este botón, el dialogo comienza a ejecutarse.

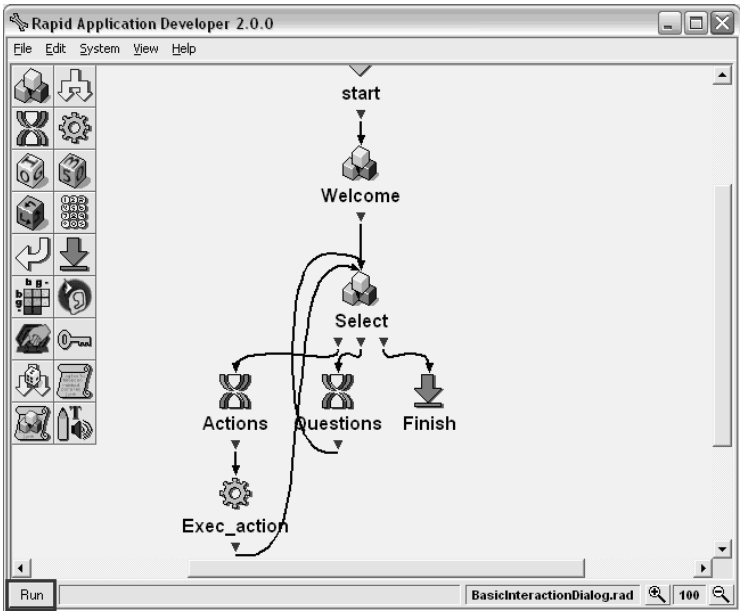


Figura 6.7 Ubicación del botón *Run*.

El diálogo de interacción puede ser ejecutado hasta asegurarse que el reconocimiento de los comandos de voz es el indicado para cada nodo tanto en los diálogos principales (*diálogo de interacción básica* y *diálogo de entrenamiento*) como en los subdiálogos *Actions*, *Questions* y *Perceptions*.

6.1.3 Comunicación entre la computadora y el robot

Para probar las comunicaciones entre el módulo HRI y el robot se realizaron una serie de pruebas similares a las descritas en el apartado 6.1.1. Se envió al robot, desde el software HRI en el modo de ejecución cero, una serie de instrucciones con el fin de determinar si los mensajes estaban llegando correctamente al robot. También se enviaron un conjunto de preguntas y se espero la respuesta del robot, el escenario de pruebas fue el mismo que en las pruebas de la sección 6.1.1 y corresponde a la figura 6.3.

Los resultados obtenidos para estas pruebas son idénticos a los obtenidos anteriormente, la acción siempre es ejecutada y las preguntas son, también, siempre contestadas correctamente. Las tablas 6.3 y 6.4 muestran los resultados de las pruebas de este apartado.

Acción	Identificador	Robot
Girar a la derecha	9	Ejecutó el movimiento
Girar a la derecha	9	Ejecutó el movimiento
Girar a la izquierda	10	Ejecutó el movimiento
Tirar	11	Ejecutó el movimiento
Girar a la izquierda	10	Ejecutó el movimiento
Tomar pelota	12	Ejecutó el movimiento
Girar a la izq. con pelota	14	Ejecutó el movimiento
Girar a la izq. con pelota	14	Ejecutó el movimiento
Tirar	11	Ejecutó el movimiento
Ir a pelota	20	Ejecutó el movimiento
Detenerse	0	Se detuvo
Caminar	1	Ejecutó el movimiento
Detenerse	0	Se detuvo
Bloquear	22	Ejecutó el movimiento

Tabla 6.3 Resultados de la prueba de acciones usando HRI.

Pregunta	Identificador	Respuesta	Resultado
¿Ve la pelota?	50	628	Correcto
¿Ve un robot rojo?	60	147	Correcto
¿Ve un robot azul?	61	-1	Correcto
¿Ve la portería amarilla?	51	-1	Correcto
¿Ve la portería cian?	52	3482	Correcto
¿Ve la pelota cerca?	59	-1	Correcto
¿Ve la pelota?	50	634	Correcto

Tabla 6.4 Resultados de la prueba de pregunta usando HRI.

6.2 ACCIÓN, INTERROGACIÓN Y ENTRENAMIENTO

En esta sección se presentan las principales pruebas sobre las capacidades del sistema. Primero, se exponen los resultados de las pruebas de comandos de acción; después, los de las pruebas con comandos de interrogación bajo diferentes escenarios; finalmente, se presentan los resultados de las distintas pruebas al sistema de entrenamiento.

Las dos primeras pruebas (acciones e interrogación) buscan evaluar tanto el correcto funcionamiento del sistema como el tiempo de respuesta que ofrece al usuario.

6.2.1 Acción

Un ejemplo de la secuencia de ejecución de una acción puede observarse en la figura 6.8; en ella, las pantallas A y C muestran al diálogo de interacción básica mientras que la pantalla B es una captura del subdiálogo *Actions*. En la pantalla A se puede observar la ejecución del nodo *Select*, este nodo permite elegir entre dos opciones: ordenar acciones o realizar preguntas. Con el comando de voz "action" se selecciona ordenar acciones al robot. El flujo del diálogo avanza al subdiálogo *Actions*; en él es posible seleccionar distintas acciones utilizando los comandos de voz enlistados en la tabla 5.2; por ejemplo, se puede ordenar al robot ejecutar el movimiento de tiro utilizando el comando de voz "kick". Tras seleccionar la acción a ejecutar, el flujo regresa al *diálogo de interacción básica*; en él, la ejecución de la instrucción se da en el nodo *Exec_action* (pantalla C).

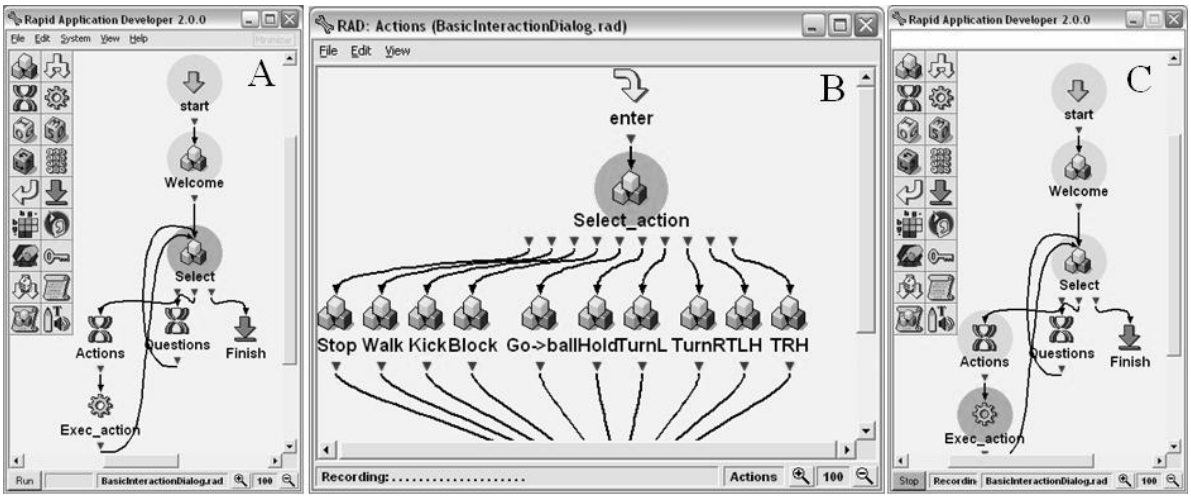


Figura 6.8 Ejecución de una acción.

La prueba consistió en ordenar al robot la ejecución de una serie de acciones básicas: primero, se ordenó al robot rotar a la izquierda sobre su eje; en seguida, bloquear; luego, ejecutar el tiro; después, rotar a la derecha; a continuación, tirar de nuevo; posteriormente, volver a girar a la derecha; luego, bloquear una vez más; después, rotar a la izquierda y finalmente, tirar.

La tabla 6.5 muestra el diálogo de interacción entre el usuario y el sistema utilizado en esta prueba, la prueba puede ser observada en el video HRI-01-Actions.wmv en [RAM 2009].

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Basic Interaction Dialog	
Select	System	Select option	
Select	User	Action	
Select_action	User	**Left**	
TurnL			Gira a la izquierda
Select	System	Select option	
Select	User	Action	
Select_action	User	**Block**	
Block			Ejecuta bloqueo
Select	System	Select option	
Select	User	Action	
Select_action	User	**Kick**	
Kick			Ejecuta tiro
Select	System	Select option	
Select	User	Action	
Select_action	User	**Right**	
TurnR			Gira a la derecha
Select	System	Select option	
Select	User	Action	
Select_action	User	**Kick**	
Kick			Ejecuta tiro
Select	System	Select option	
Select	User	Action	
Select_action	User	**Right**	
TurnR			Gira a la derecha
Select	System	Select option	
Select	User	Action	
Select_action	User	**Block**	
Block			Ejecuta bloqueo
Select	System	Select option	
Select	User	Action	
Select_action	User	**Left**	
TurnL			Gira a la izquierda
Select	System	Select option	
Select	User	Action	
Select_action	User	**Kick**	
Kick			Ejecuta tiro
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.5 Diálogo de la prueba de tiempos con acciones.

La primera columna indica el nodo del *diálogo de interacción básica* en que se encuentra el flujo del sistema, la segunda columna muestra al actor que habla en dicho nodo, la tercera y cuarta columna indican la conversación oral que se presenta y el comportamiento del robot para cada nodo.

En esta prueba se midió el tiempo de reacción del sistema, es decir, el tiempo transcurrido entre el final del comando de voz y el inicio de la ejecución de la orden por parte del robot. La prueba se realizo 3 veces, los resultados se muestran en la tabla 6.6.

Acción	Tiempo de reacción (segundos)		
	Prueba 1	Prueba 2	Prueba 3
Rotar izquierda	0.83	0.82	0.81
Bloquear	0.84	0.79	0.83
Tirar	0.81	0.83	0.77
Rotar derecha	0.83	0.79	0.81
Tirar	0.78	0.84	0.79
Rotar derecha	0.78	0.83	0.79
Bloquear	0.88	0.78	0.81
Rotar izquierda	0.77	0.79	0.79
Tirar	0.78	0.81	0.80
PROMEDIO	0.8111	0.8088	0.8000

Tabla 6.6 Tiempos de respuesta para la prueba con acciones.

El promedio de tiempo de reacción para las tres pruebas fue de 0.8066 segundos mientras que la desviación estándar fue de 0.0261 segundos. Estos últimos datos nos demuestran que el tiempo de reacción es aceptable y no presenta grandes variaciones.

6.2.2 Interrogación

Un ejemplo de una típica secuencia de ejecución para realizar preguntas al robot es mostrado en las distintas ventanas de la figura 6.9. El flujo comienza en el nodo *Select*, aquí el usuario puede elegir entre ordenar una acción o realizar una pregunta; utilizando el comando de voz "question", el usuario seleccionar realizar una pregunta al robot. El flujo avanza al subdiálogo *Questions*; en el nodo *Select_question* (pantalla B) el usuario debe seleccionar alguna de las preguntas disponibles utilizando los comandos de voz mostrados en la tabla 5.3; por ejemplo, el usuario puede preguntar al robot si detecta la pelota utilizando el comando de voz "Do you see the ball?". El nodo MQ invoca al programa HRI para enviar la pregunta al robot y recibir la respuesta. Los nodos *Yes* y *No* (pantalla D) se encargan de emitir la respuesta mediante síntesis de voz.

La prueba a las capacidades de interrogación del sistema consistió en realizar una serie de preguntas al robot y esperar la respuesta. El módulo HRI envía la pregunta al robot y espera la respuesta. En el robot, EK2008 procesa el mensaje recibido y genera una respuesta tras consultar la información proveniente del módulo de procesamiento de visión.

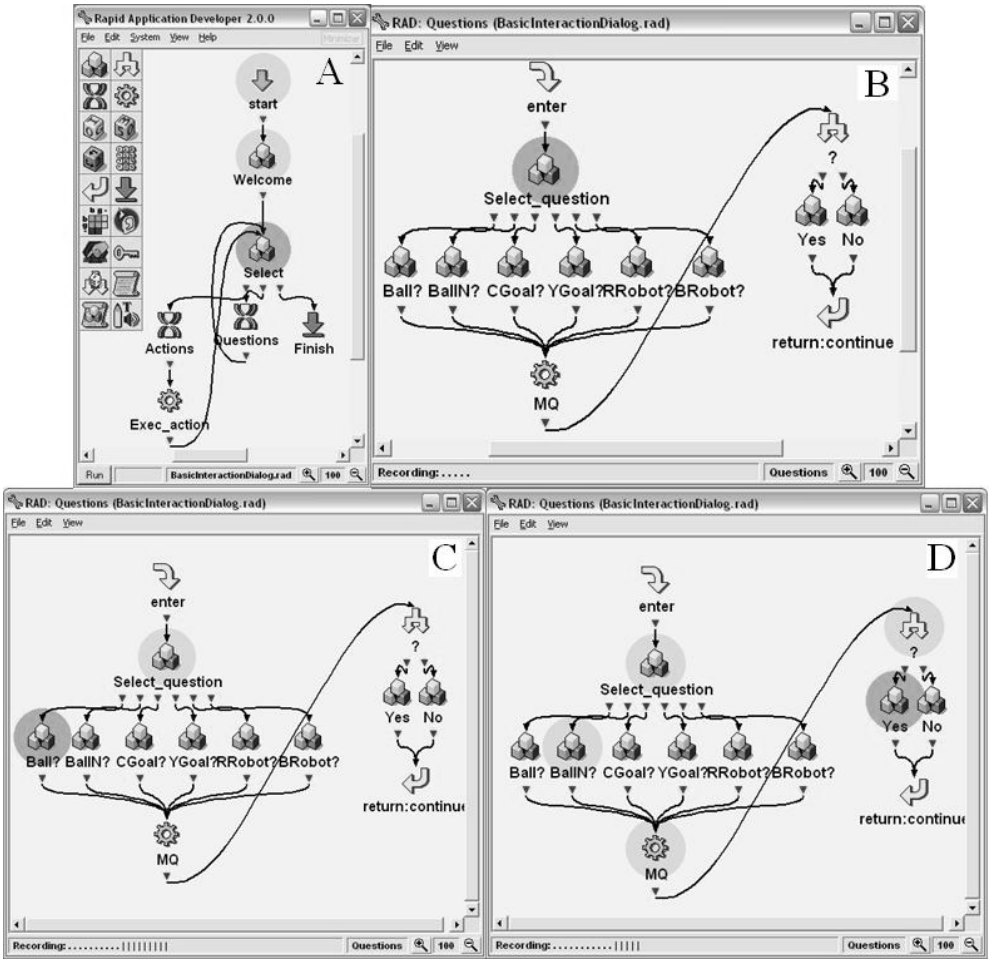


Figura 6.9 Ejecución de una pregunta.

Esta prueba se realizó bajo dos escenarios, en el primero (figura 6.10) el robot tiene frente a él la pelota, el robot azul y la portería color cian. Se preguntó al robot si detectaba la pelota, la portería cian, la portería amarilla, el robot rojo y, por último, el robot azul.

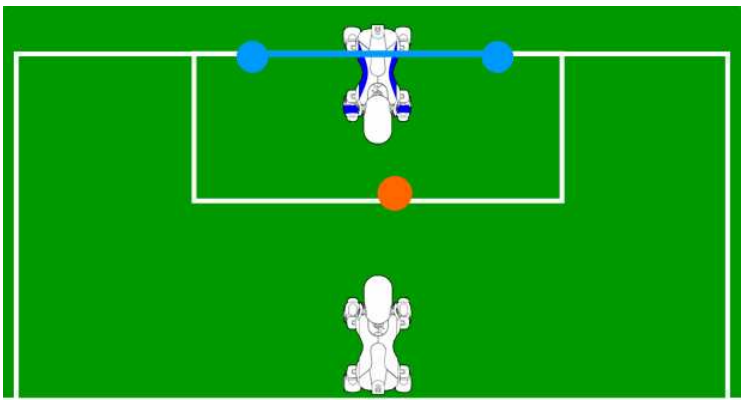


Figura 6.10 Escenario 1 de la prueba con preguntas.

El diálogo de interacción del escenario 1 se muestra en la tabla 6.7.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Basic Interaction Dialog	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the ball?**	
Ball?			
MQ			Verifica y responde
Yes	System	**Yes**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the cyan goal?**	
CGoal?			
MQ			Verifica y responde
Yes	System	**Yes**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the yellow goal?**	
YGoal?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see a red robot?**	
RRobot?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see a blue robot?**	
BRobot?			
MQ			Verifica y responde
Yes	System	**Yes**	
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.7 Diálogo de la prueba con preguntas, escenario 1.

Al igual que en la prueba anterior, la prueba de preguntas para el escenario descrito se realizó en tres ocasiones obteniendo en cada una de ellas siempre las respuestas correctas. Los tiempos de respuesta fueron medidos entre el momento es que se terminó de formular la pregunta y el momento en que los nodo *Yes* y *No* sintetizan la voz correspondiente a la respuesta.

En este caso el promedio en el tiempo de respuesta fue de 1.3086 segundos con una desviación estándar de 0.0229.

Objeto	Tiempo de reacción (segundos)		
	Prueba 1	Prueba 2	Prueba 3
Pelota	1.28	1.33	1.34
Portería cian	1.31	1.30	1.29
Portería amarilla	1.29	1.27	1.33
Robot rojo	1.29	1.31	1.30
Robot azul	1.33	1.33	1.33
PROMEDIO	1.3000	1.3080	1.3180

Tabla 6.8 Tiempos de respuesta de la prueba (escenario 1) con preguntas.

El escenario 2 y su respectivo diálogo son mostrados en la figura 6.11 y la tabla 6.9, respectivamente. Aquí se realizaron las mismas preguntas pero en distinto orden.

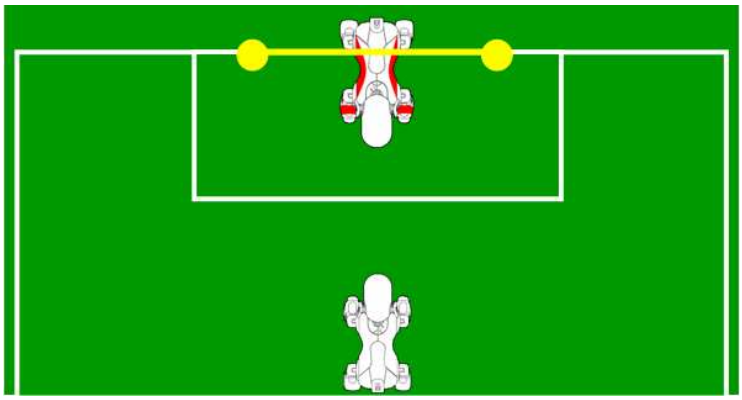


Figura 6.11 Escenario 2 de la prueba con preguntas.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Basic Interaction Dialog	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the ball?**	
Ball?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see a red robot?**	
RRobot?			
MQ			Verifica y responde
Yes	System	**Yes**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the cyan goal?**	
CGoal?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the yellow goal?**	
YGoal?			

MQ			Verifica y responde
Yes	System	**Yes**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see a blue robot?**	
BRobot?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Question	
Select_question	User	**Do you see the ball?**	
Ball?			
MQ			Verifica y responde
No	System	**No**	
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.9 Diálogo de la prueba con preguntas, escenario 2.

La tabla 6.10 contiene los tiempos de respuesta para las tres pruebas de este escenario.

Objeto	Tiempo de reacción (segundos)		
	Prueba 1	Prueba 2	Prueba 3
Pelota	1.32	1.32	1.28
Robot rojo	1.28	1.31	1.31
Portería cian	1.33	1.29	1.31
Portería amarilla	1.31	1.28	1.32
Robot azul	1.31	1.34	1.31
Pelota	1.29	1.29	1.28
PROMEDIO	1.3066	1.3050	1.3016

Tabla 6.10 Tiempos de respuesta de la prueba (escenario 2) con preguntas.

En este nuevo escenario el promedio de los tiempos de respuesta es 1.3044 segundo y la desviación estándar es de 0.0185. Considerando ambos escenarios, promedio es de 1.3063 segundos y la desviación estándar de alrededor de 0.0199.

Los datos de los anteriores experimentos muestran que se tienen tiempos de respuesta cortos y con poca variabilidad.

6.2.3 Entrenamiento

La figura 6.12 muestra una típica ejecución de una instrucción de entrenamiento. En la pantalla A, el nodo *Select* permite al usuario seleccionar dos opciones: proporcionar una instrucción de entrenamiento o proporcionar una instrucción de decisión. Con el comando de voz "instruction" el usuario selecciona proporcionar al sistema una instrucción de entrenamiento, la ejecución continua hacia el subdiálogo *Actions* (Pantalla B). El nodo *Select*_action del mencionado subdiálogo permite elegir cualquier de las acciones básicas que el robot puede ejecutar mediante los comandos de voz mostrados en la tabla 5.2.

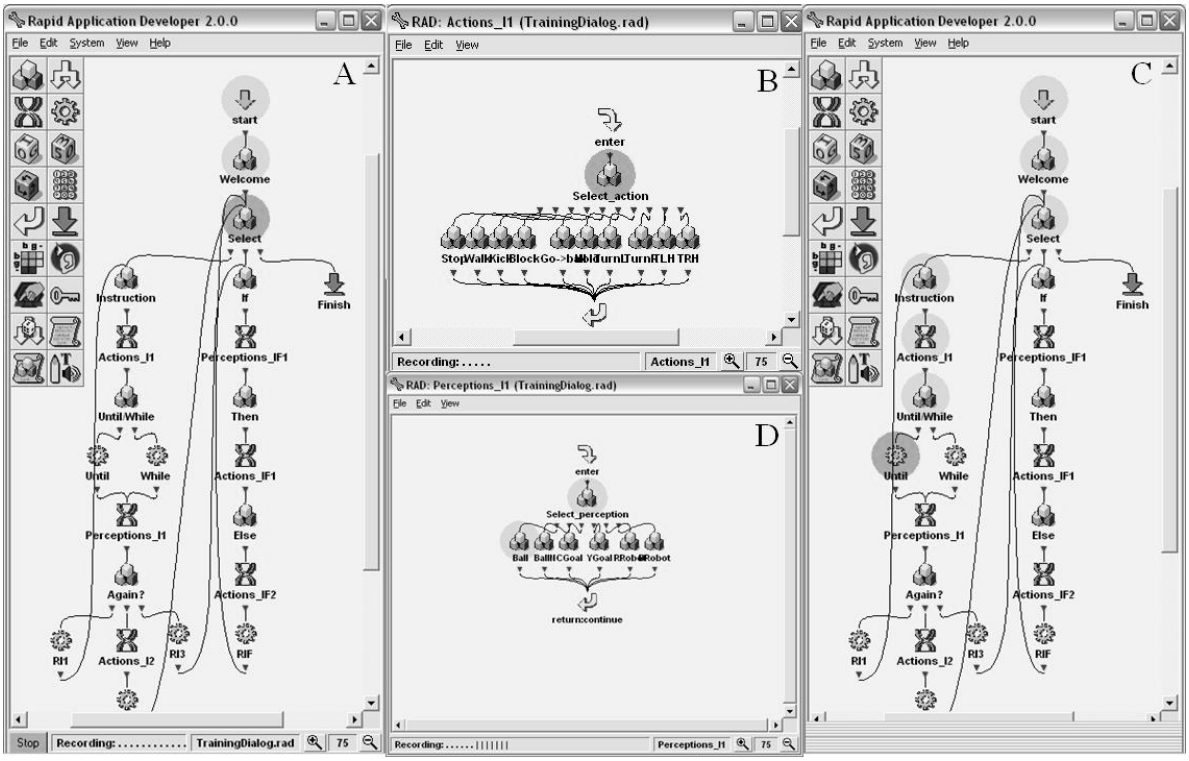


Figura 6.12 Ejecución de una instrucción de entrenamiento.

Tras seleccionar la acción a realizar, el flujo continua hacia el nodo *Until/While*, en dicho nodo se pueden seleccionar mediante los comandos de voz "until" o "while" el condicionante para relacionar la percepción con la acción. Finalmente, el flujo del *diálogo de entrenamiento* alcanza al subdiálogo *Perceptions*; en éste se debe elegir la percepción que condiciona la acción, por ejemplo con el comando de voz "you see the ball".

Para cada instrucción de entrenamiento, el sistema toma los tres elementos (percepción, condición y acción) proporcionados por el usuario y define con ellos un relación que permite que la percepción dispare la acción. La relación es almacenada en el archivo utilizando los identificadores de la percepción, la condición y la acción. Así, cuando un comportamiento es invocado, la relación permite que una acción se ejecute hasta que se presente la percepción o mientras se presente.

Por otro lado, la figura 6.13 muestra la secuencia de ejecución para una instrucción de decisión. En el nodo *Select* del *diálogo de entrenamiento*, el comando de voz "If" sirve para seleccionar una instrucción de decisión; cuando dicho comando de voz es utilizado, el flujo avanza hacia un subdiálogo del tipo *Perceptions*. El nodo *Select_perception* permite seleccionar una percepción cuya respuesta desencadenará alguna de las dos acciones que a continuación se seleccionan. Una vez que se ha seleccionado una percepción, por ejemplo "you see a blue robot", el flujo avanza al nodo *Then* del diálogo de entrenamiento (pantalla C) en el que es necesario que el usuario emita el comando de voz "then". El punto anterior conducirá el flujo hacia el subdiálogo *Actions_IF1* en el que se debe elegir la acción a ejecutar cuando la percepción se produzca; tras esto, el flujo conduce al nodo *Else* en el que es obligatorio emitir el comando de voz "else". Finalmente, el flujo avanza al subdiálogo *Actions_IF2* en el que se deberá elegir la acción a ejecutar cuando la percepción no se cumpla.

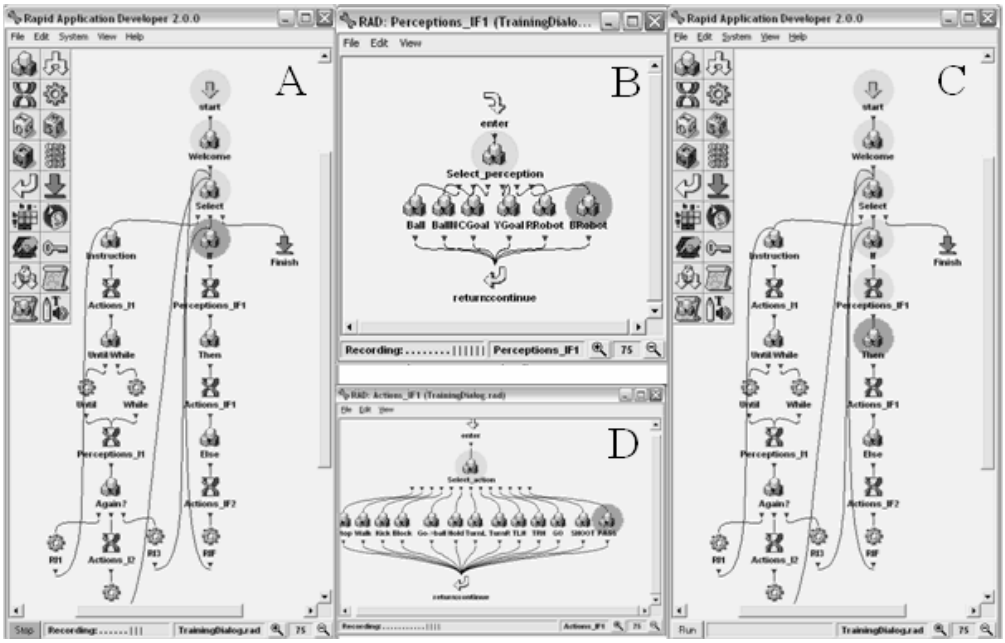


Figura 6.13 Ejecución de una instrucción de decisión IF.

6.2.3.1 Comportamiento *Go*

Se realizaron dos pruebas de entrenamiento básico de comportamientos; el primero de ellos es llamado *Go* y consiste en entrenar al robot para que busque la pelota y mientras la vea se acerque a ella; luego, el robot debe tomar la pelota con sus patas frontales. El diálogo de entrenamiento y el subdiálogo *Perceptions* son usados en este y el resto de los experimentos. El diálogo entre el sistema y el usuario para esta prueba de entrenamiento se muestra en la tabla 6.11.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Right**	
TurnR			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball**	
Ball			
Again?	System	What do you want to do now?	
Again?	User	**New instruction**	Gira a la derecha hasta reconocer la pelota
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Go to the ball**	
TurnR			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball near**	
BallN			
Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**Hold**	Se acerca a pelota hasta estar cerca de ella y toma la pelota.
Hold			
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.11 Diálogo del entrenamiento del comportamiento *Go*.

En [RAM 2009] se puede encontrar el video HRI-03-Go.wmv, el video muestra algunas de pruebas para el comportamiento *Go*. Las pruebas se realizaron con el robot partiendo de 5 diferentes posiciones en la cancha, tres de ellas cercanas a la mitad del campo y dos en las esquinas más próximas a la portería que se ataca. La pelota en las pruebas siempre se colocó frente a la portería, tal como muestra la figura 6.14.

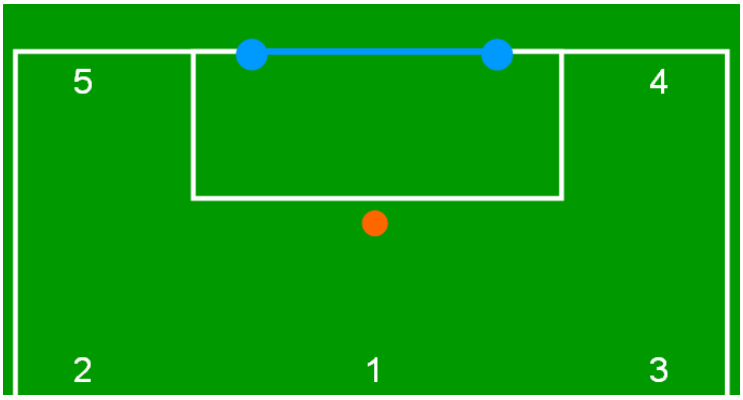


Figura 6.14 Ubicación de las áreas de inicio para las pruebas del comportamiento *Go*.

Adicionalmente, la tabla 6.12 muestra los resultados en la ejecución de las pruebas.

Prueba	Posición de inicio	Tiempo de ejecución (segundos)	Resultado
1	1	14.06	Sujetó la bola
2	1	14.21	Sujetó la bola
3	2	14.28	Sujetó la bola
4	2	14.18	Sujetó la bola
5	3	12.98	Sujetó la bola
6	3	13.86	Sujetó la bola
7	4	24.81	Sujetó la bola
8	4	14.39	Sujetó la bola
9	5	15.72	NO sujeto la bola
10	5	16.21	Sujetó la bola

Tabla 6.12 Resultados de las pruebas de la secuencia *Go*.

6.2.3.2 Comportamiento *Shoot*.

El segundo comportamiento entrenado, denominado *Shoot*, comienza con el robot en posesión de la pelota. En este comportamiento, el robot debe verificar si identifica la portería de color cian; si no puede verla, el robot debe rotar con la pelota y verificar nuevamente si identifica la portería; por el contrario, si la puede reconocer, debe ejecutar el movimiento de tiro con el objetivo de anotar gol.

Al igual que en el experimento anterior, un video con un fragmento de las pruebas puede encontrarse en [RAM 2009] bajo el nombre HRI-04-Shoot.wmv. La tabla 6.13 muestra el diálogo entre el usuario y el sistema. Los resultados de las pruebas pueden observarse en la tabla 6.14.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Right with ball**	
TRH			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the cyan goal**	
CGoal			
Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**Kick**	Gira a la derecha hasta reconocer la portería cian, luego tira
Kick			
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.13 Diálogo de entrenamiento del comportamiento *Shoot*.

Prueba	Posición de inicio	Tiempo de ejecución (segundos)	Resultados
1	1	3.61	No anotó
2	1	9.71	Giro innecesario
3	1	11.59	Giro innecesario
4	1	3.40	Anotó
5	3	5.40	Poste
6	3	11.33	No anoto
7	1	5.48	Anotó
8	1	6.28	Anotó
9	2	9.60	Anotó
10	2	8.92	Anotó

Tabla 6.14 Resultados del comportamiento *Shoot*.

En este caso, la variación de tiempos y resultados dependió de la zona de inicio de la secuencia, la figura 6.15, muestra la ubicación de las posiciones de inicio.

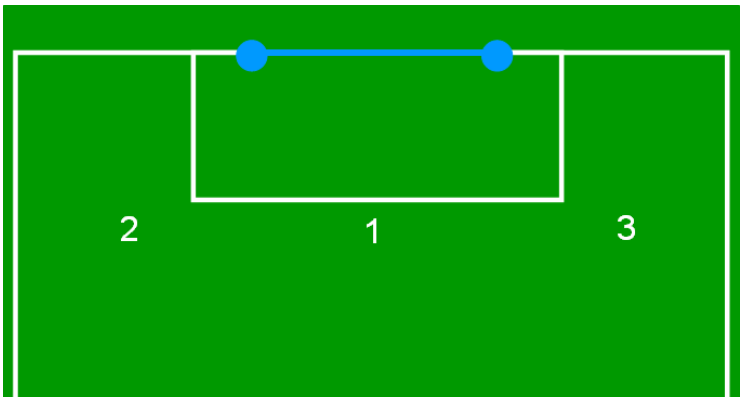


Figura 6.15 Posiciones de inicio de la prueba del comportamiento *Shoot*.

Con los resultados de las últimas dos pruebas demostramos que el modelo basado en un autómata situado resulta efectivo pues, al tiempo que permite que las percepciones disparen las acciones del robot, la transición entre los estado de éste controlan el comportamiento.

6.2.3.3 Comportamiento *Go & Shoot.*

Para probar el entrenamiento de comportamientos complejos se utilizaron los comportamientos entrenados en la sección anterior (*Go* y *Shoot*) y se creó un nuevo comportamiento llamado *Go & Shoot*.

La idea en este nuevo experimento es aprovechar los comportamientos aprendidos anteriormente para desarrollar otros más complejos. El comportamiento consiste en indicar al robot que ejecute el comportamiento *Go*, es decir, que busque la pelota, se acerque a ella y la tome, seguido del comportamiento *Shoot*, o sea, que una vez que tenga la pelota busque la portería girando hasta que tire hacia ella. Para lo anterior los comportamientos *Go* y *Shoot* tuvieron que ser añadidos manualmente al subdiálogo *Actions*.

La figura 6.16 muestra una captura del subdiálogo *Actions* en el que ya se han añadido manualmente los comportamientos.

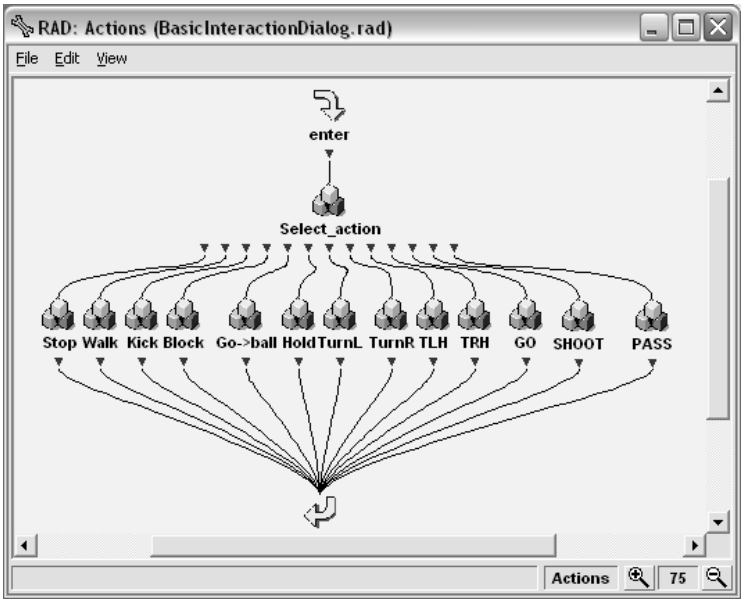


Figura 6.16 Subdiálogo *Actions* con comportamientos complejos.

El diálogo entre el usuario y el sistema para el entrenamiento se muestra en la tabla 6.15.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**GO**	
GO			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball near**	
BallN			
Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**SHOOT**	Busca la pelota y se acerca a ella cuando la ve, la toma y tira a gol cuando ve la portería azul
SHOOT			
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.15 Diálogo del entrenamiento del comportamiento complejo *Go & Shoot*.

El tiempo de entrenamiento, en condiciones ideales, es de alrededor de 33 segundos. Esta prueba pretende destacar la ventaja de estructurar comportamientos basados en otros comportamientos. El mismo comportamiento puede ser entrenado utilizando sólo acciones (tabla 6.16), el tiempo de entrenamiento toma cerca de 1 minuto con 38 segundos si se realiza solo con acciones.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Right**	
GO			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball**	
Ball			
Again?	System	What do you want to do now?	
Again?	User	**New instruction**	Gira a la derecha hasta reconocer la pelota.
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Go to the ball**	
Go->Ball			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball near**	
BallN			

Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**Hold**	Se acerca a la pelota y la toma cuando está cerca
Hold			
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Right whit ball**	
TRH			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the cyan goal**	
CGoal			
Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**Kick**	Gira con la pelota y tira cuando ve la portería cian
Kick			
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.16 Entrenamiento equivalente a *Go & Shoot.*

Las pruebas se realizaron utilizando tres posiciones iniciales para la pelota (figura 6.17, marcadas en color naranja) y tres posiciones para el robot (figura 6.17, marcadas en color blanco).

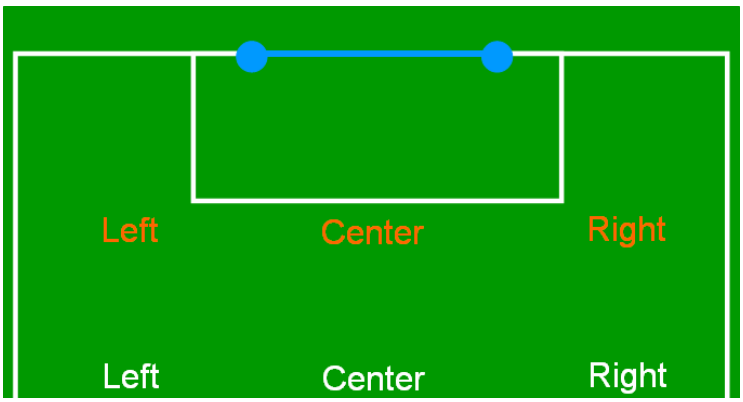


Figura 6.17 Posiciones de inicio de la pelota y el robot para la prueba del comportamiento *Go & Shoot*.

Existen dos videos, HRI-06-GoShoot.wmv y HRI-07-GoShoot.wmv, en [RAM 2009] donde se pueden observar las pruebas realizadas.

La tabla 6.17 muestra los resultados de la primera ejecución de la prueba desglosando los tiempos en cada uno de los estados que componen el comportamiento.

Prueba	Robot	Pelota	Tiempo (segundos)					Resultado
			GO		SHOOT			
	Position inicial	Posición inicial	Busca pelota	Va a pelota	Busca port.	Tira	Total	
1	Izq.	Izq.	0	8	2	4	14	Gol
2	Centro	Izq.	0	10	2	4	16	Gol
3	Der.	Izq.	4	14	2	3	23	No anotó
4	Izq.	Centro	2	10	1	4	17	No anotó
5	Centro	Centro	0	10	0	4	14	Gol
6	Der.	Centro	7	14	0	3	24	Gol
7	Izq.	Der.	2	15	4	3	24	Gol
8	Centro	Der.	6	11	5	4	26	Gol
9	Der.	Der.	0	10	5	3	18	Gol

Tabla 6.17 Resultados del primer experimento del comportamiento *Go & Shoot*.

La tabla 6.18 muestra los resultados de la segunda ejecución de la prueba desglosando los tiempos de cada uno de los comportamientos entrenados que componen este comportamiento complejo.

Prueba	Robot	Pelota	Tiempo Go	Tiempo Shoot	Tiempo Total	Resultado
1	Centro	Izq.	13	8	21	No anotó
2	Centro	Centro	9	3	12	Gol
3	Centro	Der.	13	7	20	No anotó
4	Izq.	Izq.	10	8	18	Gol
5	Izq.	Centro	26	3	29	Gol
6	Izq.	Der.	16	7	23	No anotó
7	Der.	Izq.	18	4	22	Gol
8	Der.	Centro	20	8	28	No anotó
9	Der.	Der.	10	7	17	No anotó

Tabla 6.18 Resultados del segundo experimento del comportamiento *Go & Shoot*.

Los tiempos son altos cuando las posiciones de inicio de robot y pelota son opuestas, es decir, cuando el robot tiene que cubrir una mayor distancia para tomar la pelota. Pese a tener una tasa de anotación de solo 61.11%, los movimientos del robot denotan la intención de realizar siempre los comportamientos indicados; por tanto, mejoras el sistema de percepción del robot, permitirían obtener mejores resultados.

6.2.3.4 Comportamiento *Go & Decisión*.

Para probar las instrucciones de decisión, primero se entrenó un comportamiento denominado ‘Pass’; en él, el robot busca otro robot con uniforme rojo, y tira en dirección a él para darle la oportunidad al otro robot de manipular la pelota. El comportamiento *Pass*, al igual que *Shoot*,

inicia con el robot en posesión de la pelota.

El diálogo entre el usuario y el sistema para el entrenamiento del comportamiento *Pass* se muestra en tabla 6.19.

Nodo en diálogo	Actor	Conversación	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**Right with ball**	
TRH			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see a red robot**	
RRobot			
Again?	System	What do you want to do now?	
Again?	User	Action	
Select_action	User	**Kick**	El robot gira hasta identificar un robot rojo, luego tira.
Kick			
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.19 Diálogo de entrenamiento del comportamiento *Pass*.

Una vez entrenado el comportamiento *Pass*, se entrenó uno nuevo utilizando para ello una instrucción de decisión, se nombró a dicho comportamiento *Go & Decision*. El nuevo comportamiento tiene como objetivo que el robot busque la pelota, vaya hacia ella, la tome y si no tiene enfrente un robot con uniforme azul tire hacia la portería; en el caso contrario, cuando el robot identifica al robot con uniforme azul enfrente de él, debe buscar al robot con uniforme rojo y para pasarle la pelota.

El diálogo de entrenamiento para este comportamiento es mostrado en tabla 6.20. Las figuras 6.18 y 6.19 muestran los escenarios ante los que le sería útil al robot la capacidad de decisión. En la figura 6.18 el robot blanco debe tirar hacia gol dado que el robot azul no está enfrente de él; por otro lado, en la figura 6.19 el robot blanco debe buscar al robot rojo e intentar darle un pase.

Nodo en diálogo	Actor	Diálogo	Robot
Welcome	System	Welcome to the Training Dialog	
Select	System	Select option	
Select	User	Instruction	
Select_action	User	**GO**	
GO			
Until/While	User	**Until**	
Until			
Select_perception	User	**You see the ball near**	
BallN			
Again?	System	What do you want to do now?	
Again?	User	**If**	El robot va hacia la pelota y la toma.
If			
Select_perception	User	**You see a blue robot**	
BRobot			
Then	User	**Then**	
Select_action	User	**PASS**	
PASS			
Else	User	**Else**	
Select_action	User	**SHOOT**	
SHOOT			El robot verifica sus sensores, si ve un robot azul, gira y busca un robot rojo, tira cuando lo encuentra; si no ve el robot azul, busca la portería y tira.
Select	System	Select option	
Select	User	Finish	
Finish	System	Good Bye	

Tabla 6.20 Diálogo de entrenamiento del comportamiento *Go & Decision*.

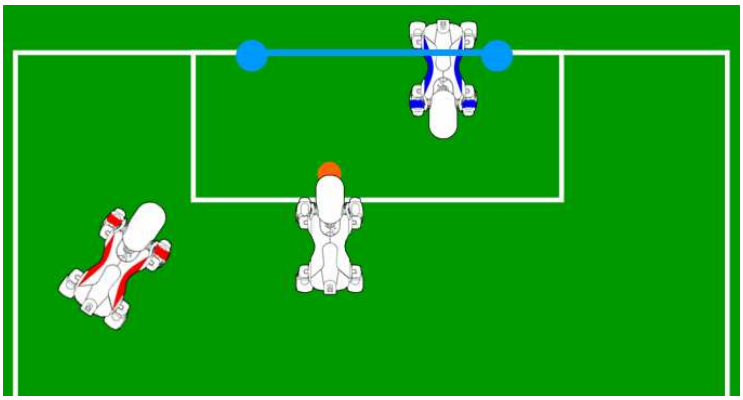


Figura 6.18 Escenario 1 para la prueba *Go & Decisión*.

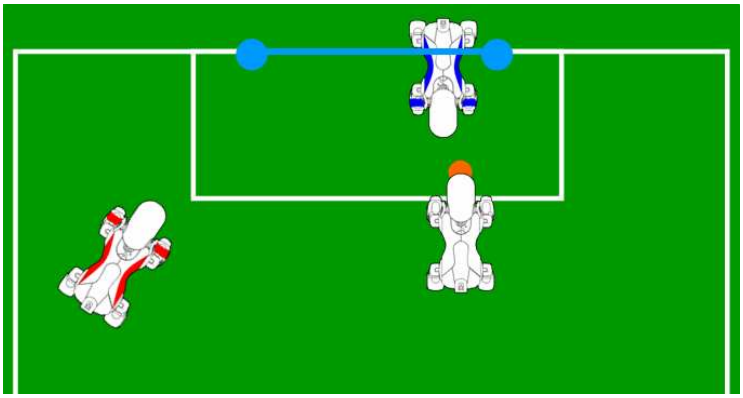


Figura 6.19 Escenario 2 para la prueba *Go & Decisión*.

El video HRI-08-GoDecision.wmv muestra las pruebas realizadas a dicho comportamiento.

Prueba	Go	Pass	Shoot	Total	Resultados
1	7.44	10.57	X	18.01	Pase
2	11.36	12.76	X	24.12	No tomó bola
3	13.76	9.34	X	23.10	No tomó bola
4	8.85	8.67	X	17.50	No tomó bola
5	7.60	10.83	X	18.43	Pase
6	8.21	X	2.97	16.18	Gol
7	10.73	X	3.22	13.95	No tomó bola
8	10.14	X	2.90	13.09	No tomó bola
9	8.49	X	3.10	11.59	Gol
10	10.58	X	3.62	14.20	Gol

Tabla 6.21 Resultados del comportamiento *Go & Decisión*.

Las pruebas realizadas fueron exitosas en la medida que, como se podrá observar en el video HRI-08-GoDecision.wmv en [RAM 2009], el robot hace los movimientos necesarios para desarrollar los comportamientos; lo anterior significa que el autómata que controla el desarrollo del comportamiento esta bien definido. Desafortunadamente, existió una tendencia del robot a no sujetar adecuadamente la bola, esto es, la ejecución de bajo nivel del comportamiento falló pues las percepciones no desencadenaron adecuadamente las acciones.

Las ultimas dos pruebas demuestran la capacidad que posee el sistema para crecer utilizando comportamientos entrenados con anterioridad para desarrollar nuevos comportamientos más complejos.

CAPÍTULO 7

CONCLUSIONES Y LÍNEAS FUTURAS

En este capítulo final se exponen tanto las conclusiones obtenidas del desarrollo del sistema así como las líneas futuras.

7.1 CONCLUSIONES PARTICULARES DEL SISTEMA

7.1.1 Sobre la arquitectura del sistema

La arquitectura modular del sistema permite la modificación y ampliación de las capacidades de interacción modificando únicamente los diálogos de interacción sin afectar ni el módulo de conexión con el robot ni el sistema de control del robot. Así, el sistema desarrollado puede ser utilizado, total o parcialmente, como base para el desarrollo de nuevas capacidades de interacción. La modularidad del sistema permitió, además, la ejecución de pruebas independientes para cada módulo permitiendo tener la certeza del funcionamiento correcto de cada módulo.

En el robot, el uso del software desarrollado para las competencias de *RoboCup* y el conocimiento adquirido en las mismas permitieron una rápida implementación tanto del módulo de comunicaciones como de los movimientos requeridos y las consultas de información. Además, la peculiar arquitectura que rige el desarrollo bajo el lenguaje OPEN-R permite una organización apropiada del código, pues lo separa en subsistemas fáciles de identificar y verificar.

Por otro lado, el uso de la aplicación *Rapid Application Developer* permitió evitar el desarrollo de un sistema de reconocimiento de voz propio; sin embargo, dada la estructura que se imponen a la implementación de los diálogos de interacción, *Rapid Application Developer* introduce un factor de rigidez en la interacción entre el usuario y el sistema. A pesar de que *RAD* obliga al usuario a hablar pausadamente, se puede afirmar que este software y la

implementación de los diálogos de interacción ofrecen una buena aproximación al tipo de conversación deseada.

7.1.2 Sobre la funcionalidad que ofrece al usuario

Se alcanzaron niveles muy aceptables en el desempeño de las diversas tareas que desempeña el sistema: enviar instrucciones de acción, realizar preguntas y proporcionar capacidad de entrenamiento, todo ello basado en voz.

Las instrucciones de acción tienen un tiempo de respuesta aceptable, por debajo de un segundo, mientras que el tiempo del conjunto pregunta-respuesta apenas sobrepasa un segundo. Aunque ambos son susceptibles a interferencia en las comunicaciones o a la perdida de la misma, mientras se tenga una conexión estable entre el sistema y el robot, los tiempos de respuesta no varían de forma considerable.

EK2008 proporcionó al robot un sistema confiable que se ve reflejado tanto en la ejecución correcta de las acciones que se ordenan al robot como en la acertada respuesta a preguntas que se realizan al mismo.

Por otro lado, el entrenamiento y posterior ejecución de comportamientos fue implementado siguiendo el modelo de un autómata de estados finitos llevado a la práctica mediante una lista ligada. Se aprendió que la construcción basada en un acumulado de acciones simples para la construcción de los comportamientos complejos resulta ser una técnica eficaz.

Adicionalmente, el uso de comportamientos complejos integrados por otros comportamientos entrenados con anterioridad proporciona la posibilidad de expandir la complejidad de los comportamientos del robot de manera rápida.

Un punto a resaltar es la adición de la capacidad de decisión, ante escenarios complejos la toma de decisiones se hace imprescindible para el desarrollo de comportamientos útiles.

7.2 LÍNEAS FUTURAS

Si bien el sistema cubre con los objetivos planteados al inicio del trabajo, es evidente que puede ser mejorado en diversos aspectos en trabajos posteriores. Algunos de estos aspectos se enlistan a continuación.

- Primero, rediseñar los diálogos de entrenamiento de modo que la conversación entre el sistema y el usuario sea más natural. Se debe buscar que el uso de la herramienta RAD no implique una rigidez en la forma de hablar al usar el sistema.
- Segundo, explorar las capacidades de programación en el software RAD buscando una mejor integración entre los componentes del sistema. En el presente trabajo sólo se utilizaron instrucciones básicas de TCL.
- Tercero, ampliar las capacidades en las tomas de decisiones a escenarios que requieran la valoración de dos o más percepciones. La instrucción de decisión implementada demostró ser útil, extender su poder puede ser de provecho para logar entrenar comportamientos aún más complejos.
- Cuarto, aumentar las capacidades de control para poder interactuar con varios robots. El objetivo de la presente tesis era desarrollar el sistema de interacción para un robot; sin embargo, se debe evaluar la conveniencia de la interacción con un mayor número de ellos con el objetivo de desarrollar comportamientos que involucren estrategias de cooperación entre ellos.

Además, se puede trasladar el sistema a otras plataformas robóticas, por ejemplo el robot Aldebaran Nao. El cambio de plataforma no implica un rediseño y nueva implementación del sistema complejo sino sólo el desarrollo de un módulo equivalente a EK2008 mediante el cual se implemente la ejecución de acciones y se provea de comunicación con el resto del sistema de interacción.

Finalmente, dada el incremento en las capacidades de computo de los robots, es posible implementar totalmente en un robot un sistema similar al aquí presentado eliminando la necesidad del uso de una computadora como intermediario entre el usuario y el robot.

7.3 CONCLUSIONES GENERALES

El sistema de interacción desarrollado en el presente trabajo cumplió con los objetivos planteados en el primer capítulo, esto es, desarrollar un sistema que proporcionara al usuario la capacidad de emitir ordenes mediante comandos de voz, hacer preguntas al robot utilizando el mismo medio y, finalmente, enseñar al robot comportamientos basados en la ejecución de acciones simples o comportamientos previamente entrenados.

El uso del paradigma del autómata situado permitió implementar un sistema que reacciona de manera adecuada a los cambios en el ambiente al tiempo que se contó con una estrategia de alto nivel para controlar el desarrollo del comportamiento.

Se demostró que bajo un contexto perfectamente definido, como lo son los juegos de fútbol de *RoboCup*, la interacción basada en voz puede ser una buena herramienta para el entrenamiento de comportamientos útiles.

Sin duda, el desarrollo de los sistemas de interacción humana utilizando voz como medio proporcionará en el futuro la posibilidad de una comunicación más natural con los robots, en particular con aquellos enfocados a brindar servicios; así, se puede evitar el uso interfaces difíciles de operar o poco naturales para el usuario.

BIBLIOGRAFÍA

[ASO 1997] Asoh, H., Hayamizu, S., Isao, H., Motomura, Y., Akaho, S., and Matsui, T., *Socially embedded learning of office-conversant robot jijo-2*, in Proceedings of IJCAI-97. IJCAI, Inc. http://dli.iiit.ac.in/ijcai/IJCAI-97-VOL2/PDF/013.pdf

[BAR 2001] Bartneck, C. and Okada, M. *Robotic User Interfaces*, Proceedings of the Human and Computer Conference (HC2001), Pages 130-140. http://bartneck.de/publications/2001/roboticUserInterfaces/bartneckHC2001.pdf

[BLA 2004] Black, A.W. and Lenzo, K. A., *Multilingual text-to-speech synthesis***,** IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. (ICASSP '04), Volume 3, Page 761, May 2004.

[BRO 1991] Brooks, A., *Intelligence without representation*, Artificial Intelligence, Volume 47, Pages 139-160.

[CAR 1993] Carlson, R. *Models of Speech Synthesis*, Colloquium on Human-Machine Communication by Voice, Irvine, California, February 8-9, 1993, National Academy of Sciences, USA.

[CAM 1997] Campbell, J.P., *Speaker recognition: a tutorial*, Proceedings of the IEEE, Volume 85, Issue 9, Pages 1437-1462, Sep 1997.

[COO 2002] Cook, S., *Speech Recognition HOWTO*. http://www.faqs.org/docs/Linux-HOWTO/Speech-Recognition-HOWTO.html

[CSL 2002] *CSLU Speech Tools Rapid Application Development (RAD)*. http://cslu.cse.ogi.edu/toolkit/index.html

[DES 1999] Deshmukh, N., Ganapathiraju, A., and Picone J., *Hierarchical Search for Large Vocabulary Conversational Speech Recognition*, IEEE Signal Processing Magazine, Volume 1, Issue 1, Pages 84-107. 1999.

[DOD 1985] Doddington, G.R., *Speaker recognition—Identifying people by their voices*, Proceedings of the IEEE, Volume 73, Issue: 11, Pages: 1651- 1664, 1985.

[DO1 2005] Dominey, P.F., and Boucher, J.D., *Developmental stages of perception and language acquisition in a perceptually grounded robot*, Cognitive Systems Research, Volume 6, Issue 3, Pages 243-259, 2005.

[DO2 2005] Dominey, P.F., and Boucher, J.D., *Learning to talk about events from narrated video in a construction grammar framework*, AI, Vol 167, No 1-2, pp 31-61, 2005.

[FUR 1981] Furui, S., *Cepstral analysis technique for automatic speaker verification*, IEEE Transactions on acoustic, speech and signal processing, Volume 29, Pages 254-277. 1981.

[HEW 1992] Hewett, Baecker, Card, Carey, Gasen, Mantei, Perlman, Strong and Verplank, *ACM SIGCHI Curricula for Human-Computer Interaction*. http://sigchi.org/cdg/cdg2.html

[IBM 2009] International Business Machines Corp, *Embedded ViaVoice*, https://www-01.ibm.com/software/pervasive/embedded_viavoice/

[JUR 2007] Jurafsky, D., and Martin J., *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition.* Prentice Hall Series in Artificial Intelligence. 2007

[KAE 1986] Kaelbling, L. P., *An architecture for intelligent reactive systems*, In Georgeff, M. P. and Lansky, A. L., editors, Reasoning About Actions &Plans - Proceedings of the 1986 Workshop, Pages 395-410. Morgan Kaufmann Publishers: San Mateo, CA.

[MAJ 2002] Maja, J., *Situated Robotics*, Encyclopedia of Cognitive Science, Nature Publishers Group, Macmillian Reference Ltd., 2002.

[MAR 2005] Martinez, J.A., *Diseño del sistema de localización para robots autónomos*, Tesis Licenciatura, ITAM, México, Abril 2005.

[MAR 2008] Martínez A., *Desarrollo de un modelo de localización para múltiples robots móviles, autónomos y colaboradores dentro de la liga Four-Legged de RoboCup*, Tesis Licenciatura, ITAM, México, 2008.

[NUA 2009] Nuance Communications Inc, *Nuance 8.5*, http://www.nuance.com/recognizer/nuancerecognizer/

[PCS 2002] *Practical C++ Socket*. http://cs.ecs.baylor.edu/~donahoo/practical/CSockets/practical/

[RAM 2009] Ramos, C., *Videos de pruebas del sistema de interacción Humano-Robot* http://robotica.itam.mx/EKCoaching/

[RO1 2008] *RoboCup Oficial Site*, http://www.robocup.org/

[RO2 2008] *RoboCup Standard Platform League. AIBO Rules*. http://www.tzi.de/spl/pub/Website/Downloads/AiboRules2008.pdf

[RO3 2008] *RoboCup Standard Platform League. Nao Rules*. http://www.tzi.de/spl/pub/Website/Downloads/NaoRules2008.pdf

[ROS 1995] Rosenschein, S. and Kaelbling, L. P., *A Situated View of Representation and Control*, Artificial Intelligence, Vol. 73, Pages 149 – 173. 2005.

[SCH 2003] Schroeter, J., *Text-to-Speech Synthesis* en "Electrical Engineering Handbook", CRC Press and IEEE Press. 2003. http://www.research.att.com/~ttsweb/tts/papers/2005_EEHandbook/tts.pdf

[STU 2002] Russell, S.J. and Norvig, P., *Robotics* en Artificial Intelligence: A Moder Approach, Prentice Hall, 2002.

[TES 2005] Tesler P., *Universal robotics: the history and workings of robotics.* http://www.thetech.org/exhibits/online/robotics/universal/index.html

[THR 2003] Thrun, S., *Towards a Framework for Human-Robot Interaction*, Computer Science Department, Stanford University. http://robots.stanford.edu/papers/Thrun03f.pdf

[WEI 2006] Weitzenfeld, A., and Dominey, P., *Cognitive Robotics: Command, Interrogation and Teaching in Robot Coaching*, RoboCup 2006: Robot Soccer World Cup X, G. Lakemeyer et al (Eds), LNCS 4434, pp 379-386, Springer ISSN 0302-9743.

[WEI 2007] Weitzenfeld, A., Martínez, A., Muciño, B., Serrano, G., Ramos, C., and Rivera C., *EagleKnights 2007: Four-Legged League, Team Description Paper, ITAM, Mexico*.

[WEI 2008] Weitzenfeld, A., Ramos, C., and Dominey, P., *Coaching Robots to Play Soccer via Spoken-Langua*ge, RoboCup Symposium 2008, July 14-20, Suzhou, China.