

ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CARRERA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

**“DESARROLLO DE UN LABORATORIO VIRTUAL BÁSICO
DE ROBÓTICA MÓVIL CON TUTOR INTELIGENTE”**

Previa a la obtención del Título de:

INGENIERA DE SISTEMAS E INFORMÁTICA

POR: MARIANELA ELIZABETH ORTIZ BELTRÁN

SANGOLQUI, Diciembre del 2006

CERTIFICACIÓN

Certifico que el presente trabajo fue realizado en su totalidad por la Srta. MARIANELA ELIZABETH ORTIZ BELTRÁN, como requerimiento parcial a la obtención del título de INGENIERA EN SISTEMAS E INFORMÁTICA.

Sangolquí, Diciembre del 2006

Ing. César Villacís
DIRECTOR DE TESIS

DEDICATORIA

A mi abuelito Gerardo, quien fue ejemplo de vida abriéndose paso con su trabajo y tesón inusitados, con aquel trajinar de su presencia infatigable de sol a sol en tanto en su trabajo con en sus tareas cotidianas; quien sembró mi mente ideales de grandeza con sus consejos sabios, cuando desde niña escuchaba los relatos de sus bondades compartidas.

A Wilson Patricio, mi tío, su recuerdo lo tengo siempre presente, como un hermano mayor con sus ocurrencias y cuidados lleno de alegría mi infancia.

Porque tomada de sus manos inicie el aprendizaje en esta vida, ahora casi todo lo que soy se lo debo a su ejemplo de tenacidad y valor, todo el esfuerzo y entrega expuestos en este trabajo es dedicado a la memoria de los dos.

Marianela Elizabeth

AGRADECIMIENTOS

Mi profundo reconocimiento a la Escuela Politécnica de Ejército, al Departamento de Ciencias de la Computación, que forman verdaderos profesionales útiles a la sociedad y competentes para el desarrollo del país.

Mi gratitud a los señores Director y Codirector de tesis, que sin su valiosa guía y apoyo este trabajo no hubiese llegado a feliz término.

A mis amigos, quienes con plena confianza en mí, me dieron ánimos para continuar con el desarrollo de este proyecto pese a la complejidad y las adversidades.

Por ultimo, es necesario agradecer a toda mi familia el apoyo prestado durante el tiempo que ha durado este trabajo.

Marianela Elizabeth

ÍNDICE

INDICE DE CONTENIDOS

Resumen	1
---------------	---

CAPÍTULO 1

INTRODUCCIÓN

1.1 Tema	2
1.2 Antecedentes	2
1.3 Justificación e Importancia	3
1.4 Objetivos	5
1.5 Alcance	6

CAPÍTULO 2

MARCO TEÓRICO

2.1 Introducción	8
2.2 E-learning y software educativo	9
2.2.1 ¿Qué es el E-learning?	9
2.2.2 Ventajas y desventajas del E-learning	10
2.2.3 Tipos de aprendizaje	11

2.2.4	Factores de diseño de programas de E-learning	13
2.2.5	Enfoques metodológicos para el desarrollo de software educativo	14
2.3	Entornos virtuales de experimentación (E-Labs)	16
2.3.1	Tipos de laboratorios virtuales	16
2.3.2	Laboratorios electrónicos	18
2.3.3	Laboratorios remotos vs. laboratorios virtuales	19
2.4	Tutores inteligentes	19
2.4.1	Asistentes personales	20
2.4.2	Laboratorios virtuales y asistentes personales	20
2.4.3	Sistemas Tutores Inteligentes	20
2.5	Fundamentos de Robótica	22
2.5.1	Conceptos y elementos de robótica	22
2.5.2	Tipos de robots	23
2.5.3	Robots móviles	26
2.5.4	El problema de la planificación de trayectorias	29
2.5.5	Métodos clásicos de planificación	31
2.5.6	Método de Grafo de Visibilidad	35
2.6	Geometría Computacional	37
2.6.1	Definición	37
2.6.2	Objetivos	38
2.6.3	Aplicaciones	39

2.7	Metodología MSF	41
2.7.1	Microsoft Solution Framework (MSF)	41
2.7.2	Características de MSF	41
2.7.3	Modelos	42
2.7.4	Modelo del Proceso de desarrollo de aplicaciones	44

CAPÍTULO 3

METODOLOGÍA DEL PROYECTO

3.1	Fase de visión	46
3.1.1	Planteamiento del problema	46
3.1.2	Visión de la solución	47
3.1.3	Metas del proyecto	49
3.1.4	Matriz de tradeoffs de proyecto	50
3.1.5	Alcance	50
3.1.6	Lista de riesgos	51
3.1.7	Esquema de la solución	52
3.1.8	Visión aprobada	53
3.2	Fase de planeación	54
3.2.1	Requerimientos del sistema	54
3.2.2	Diseño conceptual	55
3.2.3	Diseño lógico	59
3.2.4	Diseño físico	64
3.2.5	Planeación del proyecto aprobada	74

3.3	Fase de desarrollo	75
3.3.1	Código fuente	75
3.3.2	Parámetros de configuración	92
3.3.3	Pruebas de inspección de código	93
3.3.4	Reporte de bugs	95
3.3.5	Alcance completo	96
3.4	Fase de Estabilización	97
3.4.1	Especificación de test	97
3.4.2	Consideraciones y restricciones	101
3.4.3	Código fuente y ejecutables	102
3.4.4	Release aprobado	102
3.5	Fase de implantación	103
3.5.1	Información de soporte y operación	103
3.5.2	Manual de usuario	104
3.5.3	Siguientes pasos	105
3.5.4	Deployment completo	105

CAPÍTULO 4

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

4.1	Conclusiones	106
4.2	Recomendaciones	108
4.3	Trabajos Futuros	109

ANEXOS

Anexo A	Diagramas UML	110
	Diagramas de Casos de Uso	110
	Diagramas de Secuencia	112
	Diagramas de Clases	114
	Diagramas de Actividad	116
Anexo B	Esquema del Tutorial de Robótica Móvil	119
	Mapa del Sitio	119
	Contenidos	120
Anexo C	Algoritmos	125
	Sumas de Minkowski	125
	Método de Grafo de Visibilidad	128
	Test de intersección	129
	Algoritmo de Dijkstra	130
Anexo D	Manual de Usuario	133
	Instalando del sistema	133
	Ingresando al sistema	134
	El entorno de trabajo	135
	Utilizando el laboratorio virtual	143
	Errores	150

BIBLIOGRAFÍA

BIBLIOGRAFÍA	153
--------------------	-----

ÍNDICE DE TABLAS

Tabla 3.1	Matriz de tradeoffs	50
Tabla 3.2	Lista de riesgos	52
Tabla 3.3	Listado de requerimientos del sistema	54
Tabla 3.4	Perfil del usuario	55
Tabla 3.5	Caso de Uso Navegación por contenidos	56
Tabla 3.6	Caso de Uso Simulación	57
Tabla 3.7	Caso de Uso Teleoperación	57
Tabla 3.8	Caso de Uso Programación	58
Tabla 3.9	Clase Figura	78
Tabla 3.10	Clase Vértice	79
Tabla 3.11	Clase Línea	80
Tabla 3.12	Clase Polígono	81
Tabla 3.13	Clase Robot	82
Tabla 3.14	Clase Nodo	83
Tabla 3.15	Clase Form1	84
Tabla 3.16	Clase Agent	86
Tabla 3.17	Clase form1 Subsistema Programación	87
Tabla 3.18	Clase Móvil	89
Tabla 3.19	Clase Video	90
Tabla 3.20	Clase Form1 Subsistema Teleoperación	91
Tabla 3.21	Prueba de navegación completa	93
Tabla 3.22	Prueba de cálculos y algoritmos	94
Tabla 3.23	Prueba de validación de datos	94
Tabla 3.24	Prueba de comunicación con el robot	95

ÍNDICE DE FIGURAS

Figura 2.1	Robots Androides	23
Figura 2.2	Robots Móviles	24
Figura 2.3	Robot Médico	25
Figura 2.4	Robot industrial	25
Figura 2.5	Elementos del sistema robótico móvil	27
Figura 2.6	Planificación de trayectorias	30
Figura 2.7	Diagrama de Voronoi	32
Figura 2.8	Diagrama de modelado del espacio libre	33
Figura 2.9	Descomposición en celdas y grafo de conectividad	34
Figura 2.10	Campos de potencial	34
Figura 2.11	Grafo de visibilidad en un entorno de dos obstáculos	35
Figura 2.12	Grafo de visibilidad y camino mínimo	37
Figura 2.13	Modelo de proceso de desarrollo de aplicaciones	44
Figura 3.1	Esquema de la solución	53
Figura 3.2	Diagrama de casos de uso	56
Figura 3.3	Diagrama de estructura del sistema	58
Figura 3.4	Diseño de la interfaz del tutorial	59
Figura 3.5	Diseño de la interfaz de simulación automática	60
Figura 3.6	Diseño de la interfaz de práctica guiada	61
Figura 3.7	Diseño de la interfaz de teleoperación	62
Figura 3.8	Diseño de la interfaz de programación	63
Figura 3.9	Componentes de la solución	63
Figura 3.10	Interfaz de la pantalla principal	67

Figura 3.11	Interfaz del tutorial	68
Figura 3.12	Interfaz de Modo de ingreso	69
Figura 3.13	Interfaz del módulo de simulación automático	70
Figura 3.14	Interfaz del modo de simulación de práctica guiada	71
Figura 3.15	Interfaz del módulo de teleoperación	72
Figura 3.16	Interfaz del módulo de programación	73
Figura 3.17	Vista arquitectónica de la implantación de la solución	73
Figura 3.18	Datos de prueba	98
Figura 3.19	Gráfico del proceso de planificación de trayectorias	100
Figura 3.20	Salida esperada	101
Figura A.1	Diagrama de casos de uso Navegación por contenidos	110
Figura A.2	Diagrama de casos de uso Simulación	110
Figura A.3	Diagrama de casos de uso Teleoperación	111
Figura A.4	Diagrama de casos de uso Programación	111
Figura A.5	Diagrama de secuencia del Sistema	112
Figura A.6	Diagrama de secuencia Navegación por contenidos	112
Figura A.7	Diagrama de secuencia Simulación	113
Figura A.8	Diagrama de secuencia Teleoperación	113
Figura A.9	Diagrama de secuencia Programación	114
Figura A.10	Diagrama de clases Contenidos	114
Figura A.11	Diagrama de clases Simulación	115
Figura A.12	Diagrama de clases Teleoperación	115
Figura A.13	Diagrama de clases Programación	116
Figura A.14	Diagrama de actividad del sistema	116
Figura A.15	Diagrama de actividad Navegación por contenidos	117

Figura A.16	Diagrama de actividad Simulación	117
Figura A.17	Diagrama de actividad Teleoperación	118
Figura A.18	Diagrama de actividad Programación	118
Figura B.1	Mapa del sitio del Tutorial de Robótica Móvil	119
Figura C.1	Sumas de Minkowski de dos polígonos	125
Figura C.2	Ángulo que forma el vector \overrightarrow{PQ} con el eje de abscisas	126
Figura C.3	Grafo con caminos y longitud	130

RESUMEN

Este proyecto fin de carrera aborda la creación de un ambiente de aprendizaje basado en la actividad de los estudiantes a través de un laboratorio virtual con un tutor inteligente, para esto es necesario el estudio de los principios, fundamentos, técnicas y herramientas que se utilizan en el desarrollo de estos sistemas, por ello, en primera instancia se contempla los conceptos teóricos correspondientes.

Un laboratorio virtual es un ambiente heterogéneo y distribuido para la solución de problemas; presenta ventajas como reducción de costos, fomenta la transferencia de tecnología, permite incorporar estrategias didácticas, tutores, ayudas en línea, etc.; entre otras, de acuerdo a esto, la presente tesis expone la aplicación de varias disciplinas informáticas como e-Learning, Computación Gráfica y Robótica para proporcionar una solución al problema de construir un laboratorio virtual con tutor inteligente como apoyo al aprendizaje de conceptos y programación de robots móviles.

El documento expone también el proceso de construcción del sistema, en donde se explica y desarrolla cada una de las etapas de la metodología planteada para el despliegue de este proyecto.

La aplicación desarrollada, es decir el laboratorio virtual básico con tutor inteligente para la enseñanza de robótica móvil, permite la simulación de un robot móvil para la definición, realización y ejecución de prácticas de planificación de trayectorias, y cuenta con un agente tutor inteligente que cumple el rol de asistente sencillo que monitorea el uso del equipo virtual; a su vez el sistema incorpora un módulo tutorial en donde se imparten los conocimientos relacionados a esta rama de la ciencia en sus aspectos básicos e iniciales. Cuenta además con un entorno básico de programación y otro de teleoperación libre del robot móvil real equivalente implementado con el set educacional LEGO Mindstorms.

CAPITULO I

INTRODUCCIÓN

1.1 Tema

Desarrollo de un Laboratorio Virtual Básico de Robótica Móvil con Tutor Inteligente.

1.2 Antecedentes

El costo de los laboratorios reales ha impulsado el desarrollo de herramientas abiertas de software para simular, visualizar y programar equipos o herramientas.

Un laboratorio virtual es un ambiente heterogéneo y distribuido para la solución de problemas; presenta ventajas como reducción de costos, fomenta la transferencia de tecnología, permite incorporar estrategias didácticas, tutores, ayudas en línea, etc.; entre otras.

Los tutores inteligentes, por otro lado, son elementos que tienen como objetivo principal reproducir el comportamiento de un tutor humano que puede adaptarse al ritmo de aprendizaje del estudiante.

El estilo de enseñanza de los tutores inteligentes puede complementar la labor de los laboratorios virtuales.

De acuerdo a lo anteriormente expuesto, la presente tesis propone una solución al problema de construir un laboratorio virtual con tutor inteligente como apoyo al aprendizaje del control y programación de robots móviles.

1.3 Justificación e Importancia

Los ejercicios de laboratorio son imprescindibles en la formación universitaria en ingeniería. Los Laboratorios Virtuales amplían los servicios de educación con el objetivo de disponer y compartir diferentes recursos para el desarrollo de experimentos como si estuvieran en el mismo sitio donde se encuentran los recursos reales.

Por otra parte, los robots han demostrado gran utilidad para realizar trabajo repetitivo; así mismo, tienen la ventaja de desempeñar actividades bajo ambientes agresivos para un ser humano. El costo tan elevado de los robots impide que muchas instituciones educativas los tengan disponibles en un laboratorio real.

Debido a lo anterior, se considera que los robots presentan características importantes para la enseñanza e investigación y por esta razón se justifica el esfuerzo de desarrollar un laboratorio virtual para robótica móvil que

permita la programación y control de estos dispositivos en beneficio de un mayor número de estudiantes.

Los principales beneficios que se vislumbran al combinar los recursos de laboratorios virtuales orientados hacia la enseñanza de la robótica móvil son los siguientes:

- Lograr que el estudiante tenga un mejor entendimiento a través de “aprender haciendo”.
- Incorporar material didáctico y explicaciones previas al desarrollo de los experimentos.
- Dar seguimiento al estudiante durante el desarrollo del experimento y brindar una evaluación posterior y una adecuada retroalimentación.
- Disminuir los costos y tiempo de enseñanza, debido a que se pueden acelerar los tiempos de los experimentos.
- Evitar accidentes que puedan dañar a los estudiantes o al equipo real que es caro y costoso.

Con todas estas afirmaciones y asumiendo que las lecciones prácticas en la universidad deben jugar un papel muy importante en el sistema educacional, especialmente en los estudios técnicos y que éstas deben ser lo más realistas posibles de forma que los estudiantes tengan un contacto directo con los equipos y métodos que van a utilizar en sus carreras profesionales, la presente Tesis contempla la solución a este problema y presenta un laboratorio virtual complementado con un tutor inteligente para

la enseñanza de robótica móvil, el cual permitirá la simulación de un robot móvil en el aspecto de la planificación de trayectorias, la tele-operación y programación del real equivalente, este proyecto beneficiará y servirá de apoyo al área de Robótica de la Facultad de Ingeniería Electrónica.

1.4 Objetivos

Objetivo General

Desarrollar un laboratorio virtual básico de robótica móvil con tutor inteligente.

Objetivos Específicos

- Definir las características del Laboratorio Virtual Básico de robótica móvil.
- Diseñar e Implementar un simulador de Planificación de trayectorias para robots móviles con sus prácticas.
- Implementar un entorno de teleoperación del robot móvil y de programación del mismo.
- Aplicar la metodología MSF para la construcción del Laboratorio virtual básico de robótica móvil.

- Revisar los fundamentos de robótica móvil, e-learning, software educativo y geometría computacional.
- Ensamblar un robot móvil de tipo vehículo con el kit de robótica de Lego Mindstorms.

1.5 Alcance

- Diseñar y desarrollar un laboratorio virtual con herramientas de Visual Studio.NET y LEGO Mindstorms, que permitirá la simulación de robots móviles; para el nivel de enseñanza universitario.

Este entorno virtual de experimentación permitirá que los estudiantes realicen diferentes prácticas de planificación de trayectorias con obstáculos poligonales convexos.

El modelo matemático a utilizarse contempla el caso general tanto en el ingreso de polígonos como ampliación de los mismos, dejando de lado las particularidades de proximidad mínima e intersecciones entre ellos.

El modelo matemático también contempla el uso de teoría de grafos, búsqueda de rutas mínimas, geometría computacional.

- Implementar un robot móvil tele – operado para realizar movimientos básicos a través del software de simulación a desarrollar con herramientas de Visual Studio.NET y LEGO Mindstorms.

El uso del kit de robótica Lego Mindstorms es una forma fácil y sencilla para construir robots, e inclusive viene acompañado de un software de programación.

- Desarrollar una interfaz de usuario tipo página web donde el usuario pueda acceder a documentación sobre robótica y programación.
- Implementar un entorno básico de programación, el cual permitirá que el usuario realice sus propios programas, compile el código y lo baje al robot.

Capítulo II

Marco Teórico

2.1 Introducción

Para la creación de sistemas de apoyo al aprendizaje de materias dotadas de una componente práctica no muy fuerte como la enseñanza de la Robótica Móvil por ejemplo, requiere de algo más, de un elemento que permita al estudiante poner en práctica todos los conocimientos que vaya adquiriendo.

Este papel en las enseñanzas tradicionales lo desempeña el *laboratorio de prácticas*, el cual, inexorablemente, requiere de la presencia física del estudiante para poder manipular los diferentes equipos existentes en un entorno controlado bajo la supervisión del profesor. Por consiguiente, trasladando este entorno práctico a la enseñanza a distancia, el elemento necesario para abordar la realización de prácticas sobre Robótica Móvil es la existencia de un *laboratorio virtual y de teleoperación, que permita al alumno practicar de una forma lo más similar posible a como si estuviese en las dependencias del laboratorio, dándole la posibilidad de manejar las simulaciones o interactuar con el equipo real.*

En particular en este capítulo, se estudiará el E-learning como nuevo paradigma de aprendizaje basado en el uso de medios electrónicos, se analizarán los llamados Tutores Inteligentes, es decir, aquellos que tratan

de reproducir el comportamiento de un tutor humano. Se profundizará también en las posibilidades que ofrecen los entornos virtuales de experimentación para el entrenamiento y el uso de la Computación Gráfica como aportes complementarios para la creación de este tipo de sistemas de aprendizaje o laboratorios virtuales.

2.2 E-learning y Software Educativo

La formación y el aprendizaje es uno de los principales dominios que, ya desde los orígenes de la Computación, se viene tratando de mejorar mediante la aplicación de las tecnologías de la información y las comunicaciones, aunque en muchos casos sin demasiado éxito. Uno de los motivos de este fracaso radica en la complejidad inherente al desarrollo de este tipo de sistemas, que tiene un carácter fuertemente multidisciplinar, y muchas veces requiere de la integración de múltiples técnicas y tecnologías provenientes de áreas muy diversas, incluso dentro de la propia Informática, como puedan ser la Computación Gráfica, el estudio de la Interacción Hombre-Máquina, la Ciencia Cognitiva, las Telecomunicaciones, la Simulación, etc.

2.2.1 ¿Qué es E-Learning?

El e-Learning es un conjunto de recursos formativos muy interactivos con actividades de aprendizaje bien estructuradas, con servicios de formación

orientados a resultados y accesibles en un entorno Internet/Intranet. Es el aprendizaje adquirido con información mediante el uso de alguna tecnología sea por red (Internet, Intranet o Extranet) o CD`s roms. La información se basa en la entrega de contenidos, mediante un esquema estructurado, ejercicios prácticos, casos de estudio, evaluaciones y simulaciones.

En otras palabras el e-Learning dirige todos los recursos para generar aprendizaje. La sociedad está en continua evolución, ya sea en ciencia, historia y como sociedad, y la educación no es un ámbito que se debe dejar de lado.

Es un cambio en el paradigma de la educación el que se debe afrontar y asimilar con la mayor rapidez posible. El e-Learning es una herramienta fundamental para conectarse al Aprendizaje virtual, Expandir su conocimiento, creatividad y estar a la vanguardia profesionalmente.

2.2.2 Ventajas y Desventajas del E-Learning

A grandes rasgos se puede citar las siguientes ventajas y desventajas del E-learning:

Ventajas

- Acceso y flexibilidad horaria.
- Justo a tiempo (JIT)

- Reducción Costos.
- Alto grado de interactividad.
- Fomenta el desarrollo personal.
- Mayor colaboración y conectividad entre estudiantes
- Independencia de la calidad del instructor

Desventajas

- Económica.
- Motivación.
- Cultural.

Algunos investigadores como Brandon Hall, Forbes, Masie y Lewis han desarrollados estudios sobre el impacto del e-Learning en el sistema educacional, empresarial y otros. Los resultados demuestran que la implementación de e-Learning no es sencilla. Para que los resultados sean realmente efectivos y eficientes, se deben tener en cuenta muchos factores desde su desarrollo hasta su implementación. Otros señalan que una de las limitaciones al introducir el e-Learning, fue la inversión inicial de costo y tiempo para la preparación y obtención de las herramientas tecnológicas, así como los conocimientos necesarios para la creación de cursos.

2.2.3 Tipos de Aprendizaje

El aprendizaje se puede dividir en los siguientes tipos:

COLABORATIVO: vía Intranet o Internet

- Sincrónico
- Asincrónico

De AUTOFORMACIÓN :

- EAO (Enseñanza Asistida por Ordenador)
- CBT(Computer based Training)

Cursos de Autoformación

En este tipo de cursos se supone que toda la información, la ejercitación, y evaluación se realiza solo exclusivamente con la interacción con el computador, el que brinda retroalimentación correctiva o formativa, guiando como un tutor virtual al aprendiz interesado.

Apoyo a la Formación Presencial

Son básicamente simulaciones, ejercicios o actividades basadas en el computador, pero que se utilizan como complemento de una actividad presencial, ya sea durante, antes o después.

Test y Evaluaciones

Es este un punto alto de la técnica instruccional basada en el computador ya que agiliza la respuesta que se da al test, guía individualmente y puede dar retroalimentación inmediata sobre los

resultados que ha procesado tanto la organización como el propio aprendiz.

Asistencia Electrónica al Desempeño

Esta forma de Capacitación Basada en el Computador consiste en una guía para resolver problemas en forma rápida y efectiva.

Se trata de garantizar que el usuario pueda responder eficazmente a situaciones inmediatas, guiado por un programa que contiene respuesta organizada a la manera de las FAQ (Frequent Asked Questions).

Aprendizaje Colaborativo

El aprendizaje colaborativo no solo hace posible el aprendizaje estandarizado sino que además facilita el aprendizaje organizacional al permitir:

- Comunicar
- Compartir conocimientos
- Distribuir conocimientos
- Generar Conocimientos

2.2.4 Factores de Diseño de Programas de e-Learning

- Objetivos de aprendizaje
- Calidad de los Contenidos

- Navegación Flexible y amistosa
- Interacciones inteligentes
- Retroalimentación constante
- Dar control al usuario
- Personalizar
- Uso adecuado de la multimedia.
- Proporcionar herramientas efectivas
- Dar apoyo a la implementación

2.2.5 Enfoques metodológicos para el desarrollo de software educativo.

Enfoque “instructivo”

A lo largo de una vida estudiantil, surgen muchos procesos de enseñanza más o menos efectivos. Incluso, el refuerzo en el proceso de aprendizaje en algunas materias, se realiza en muchos casos a través de un seguimiento más estrecho entre un tutor y el estudiante. Estos procesos centran su actividad en la transmisión de conocimiento del profesor al alumno. Las capacidades que un buen profesor presenta se centran, en muchos casos, en una estructuración y presentación acertada del conocimiento. Además de disponer de un cierta variedad de técnicas, más o menos atractivas, para mantener la atención del estudiante y facilitar la transmisión del conocimiento deseada.

Enfoque “constructivo”

Como ya hemos apuntado anteriormente, existe otra propuesta diferente al enfoque instructivo anterior y que Papert bautizó como *construtivismo*. En el planteamiento que propone entiende el aprendizaje como un proceso activo de construcción de conocimiento. Este planteamiento, asegura además que la “mejor” forma de aprender consiste en dedicarse a construir de forma consciente algo, algún objeto. En los sistemas desarrollados según esta propuesta el estudiante lleva el control de la actividad docente, construyendo su propia sesión de aprendizaje y fijando y asegurando sus propios objetivos de aprendizaje. Los sistemas hipermedia se adaptan perfectamente a las exigencias de este tipo de planteamientos. Estos sistemas permiten que el usuario:

- Acceda a la información de la base de conocimiento que desee.
- Disponga de una gran variedad de formas de acceso a la información.
- Redefina la estructura y contenido del material a utilizar.

El grado de libertad que ofrecen a los estudiantes estas herramientas, aun siendo su mayor ventaja, puede llegar a ser su peor enemigo. La gran movilidad del estudiante por el hiperespacio puede desorientarle de tal manera que pierda el rumbo adecuado para alcanzar sus objetivos. Por ello es necesario ofrecer al estudiante herramientas complementarias que le permita reorientar su proceso de aprendizaje.

2.3 Entornos virtuales de experimentación (e-labs)

Los Entornos virtuales de experimentación son nuevos tipos de prácticas y adquisición del aprendizaje mediante el uso de simuladores; Desarrollo de nuevas competencias y habilidades prácticas, por parte de los estudiantes, en laboratorios virtuales de investigación.

Una de las definiciones de “laboratorios virtuales” que se ha aplicado a la enseñanza a distancia es la aquella que los definen como “simulaciones de prácticas manipulativas que pueden ser hechas por la/el estudiante lejos de la universidad y el docente”. Los laboratorios virtuales son imitaciones digitales de prácticas de laboratorio o de campo, reducidas a la pantalla de la computadora (simulación bidimensional) o en sentido estricto, a una visión más realista con profundidad de campo y visión binocular, que requiere que la persona se coloque un casco de realidad virtual.

2.3.1 Tipos de laboratorios virtuales

Para plantear este documento, se ha clasificado en estos tres tipos generales la variedad de laboratorios virtuales:

Laboratorios virtuales software. Son laboratorios virtuales desarrollados como un programa de software independiente destinado a ejecutarse en la máquina del usuario, y cuyo servicio no requiere de un servidor Web. Es el caso de programas con instalación propia, que pueden

estar destinados a plataformas Unix, Linux, M.S. Windows... e incluso necesitar que otros componentes de software estén instalados previamente, pero que no necesitan los recursos de un servidor determinado (como bases de datos o módulos de software de servidor) para funcionar. También determinados laboratorios virtuales pensados inicialmente como aplicaciones accesibles a través de un servidor Web se pueden considerar de este tipo si funcionan localmente y no necesitan recursos de un servidor en concreto.

Laboratorios virtuales Web. En contraste con los anteriores, este tipo de laboratorios se basa en un software que depende de los recursos de un servidor determinado. Esos recursos pueden ser determinadas bases de datos, software que requiere ejecutarse en su servidor, la exigencia de determinado hardware para ejecutarse. Esto es, no son programas que un usuario pueda descargar en su equipo para ejecutar de localmente de forma independiente.

Laboratorios remotos. Se trata de laboratorios remotos que permiten operar remotamente cierto equipamiento, bien sea didáctico como maquetas específicas, o industrial, además de poder ofrecer capacidades de laboratorio virtual. En general, estos laboratorios requieren de equipos servidores específicos que les den acceso a las máquinas a operar de forma remota, y no pueden ofrecer su funcionalidad ejecutándose de forma local. Otro motivo que hace dependientes estos laboratorios de sus servidores es la habitual gestión de usuarios en el servidor.

2.3.2 Laboratorios electrónicos.

Una de las soluciones de **eLearning** más interesantes en la educación universitaria son los laboratorios electrónicos, o e-laboratorios. Estos tienen el mismo objetivo que los laboratorios tradicionales: dar a los estudiantes la oportunidad de poner en práctica las habilidades y conocimientos recientemente adquiridos realizando diferentes prácticas y experimentos a través de un uso ilimitado y repetido.

El estudiante puede acceder a los equipos del laboratorio a través de un navegador y conectarse a entornos de entrenamiento simulados o reales, pudiendo ejercitar habilidades diversas sin riesgo alguno. Así los estudiantes pueden aprender mediante prueba y error: sin miedo a sufrir o provocar un accidente; sin avergonzarse si requieren realizar varias veces la misma práctica, ya que la pueden repetir sin límite; sin temor a dañar alguna herramienta o equipo. Además, también pueden asistir al laboratorio cuando ellos lo quieran, y hasta escoger aquellas áreas del laboratorio que resultan más significativas para realizar prácticas sobre su trabajo.

Es posible llevar a cabo experimentos de forma estructurada o incluso más abierta, en la que los estudiantes desarrollan habilidades de resolución de problemas, observación interpretación y análisis de los resultados, de forma similar a la que los investigadores realizan.

2.3.3 Laboratorios remotos vs. Laboratorios virtuales

Existen dos aproximaciones diferentes a la hora de implementar un e-laboratorio: laboratorios virtuales y remotos. En un laboratorio remoto existe un equipo físico al cual acceden los estudiantes a través de un navegador. En cambio, un laboratorio virtual no tiene ningún referente físico: los estudiantes utilizan un simulador que reproduce una situación real o implementa una herramienta CAD.

2.4 Tutores Inteligentes

Una de las aproximaciones actuales en los campos vinculados a los ITS y la IA, como los agentes software y los sistemas multi-agente, es el diseño de asistentes personales que supervisen las acciones del usuario en un entorno informático para proporcionar ayuda, llamados asistentes personales.

De forma más específica, los asistentes personales son agentes de interfaz que cooperan con el usuario para alcanzar un objetivo. Esta cooperación estrecha entre el usuario y el agente permite al usuario aumentar considerablemente su rendimiento.

2.4.1 Asistentes personales

Una clase especial de Tutores Inteligentes son los denominados asistentes personales que supervisan directamente las acciones del usuario en un entorno informático para proporcionar ayuda. De forma más específica, los asistentes personales son agentes de interfaz que cooperan con el usuario para alcanzar un objetivo. Como se ha demostrado en recientes estudios, esta colaboración estrecha entre el usuario y el agente aumenta considerablemente el rendimiento del primero.

2.4.2. Laboratorios virtuales y asistentes personales

Un laboratorio virtual se puede complementar con el uso de asistentes personales. Los usuarios realizan una serie de experimentos bajo la supervisión de un asistente que le guía a través de los pasos necesarios para completar un proyecto, experimento o práctica, comunicándose con él a medida que es necesario.

2.4.3. Sistemas Tutores inteligentes

A finales de la década de los 60, un grupo de investigadores empezó a explorar el gran potencial que tenían las aproximaciones que representaban características cognitivas y el aprendizaje humano, a través de estructurar la información. Este enfoque, anclado en la Inteligencia

Artificial (IA), dio lugar a lo que se conoce hoy como Sistemas Tutores inteligentes (ITS).

Los investigadores de ITS adoptaron la idea de que los ordenadores podían construirse para entender a estudiantes y dominios de conocimiento, e inferir a partir de las interacciones del estudiante la estrategia de enseñanza más apropiada. Sin embargo, diversos factores estancaron el desarrollo de los ITS como por ejemplo el escaso conocimiento que se tenía entonces sobre la cognición y aprendizaje humano. A medida que el conocimiento en estas áreas y en otras como la IA han avanzado, los ITS también lo han hecho. Así, en la actualidad los tutores inteligentes no han recibido todavía una aceptación general debido principalmente a la complejidad de diseño y definición de los mismos que limitan su aplicabilidad en la práctica.

El problema de la complejidad del diseño de los tutores inteligentes se debe en parte a la arquitectura monolítica en la que se basa. Un paradigma de diseño de estos sistemas, y otros dentro de la inteligencia artificial, trata de resolver este problema aplicando una estrategia del tipo “divide y vencerás”, que ha dado lugar a lo que se conoce como sistemas multiagente. Como su nombre lo indica, este tipo de sistemas está formado por una serie de agentes, que funcionan como entidades computacionales al estilo de los objetos, los cuales disponen de una autonomía completa y que se comunican entre sí para llevar a cabo una tarea concreta.

2.5 Fundamentos de Robótica

2.5.1 Conceptos y Elementos de Robótica

El término Robótica procede de la palabra robot. La robótica es, por lo tanto, la ciencia o rama de la ciencia que se ocupa del estudio, desarrollo y aplicaciones de los robots.

Los robots son dispositivos compuestos de sensores que reciben datos de entrada y que pueden estar conectados a la computadora. Esta, al recibir la información de entrada, ordena al robot que efectúe una determinada acción.

Puede ser que los propios robots, dispongan de microprocesadores que reciben el input de los sensores y que estos microprocesadores ordenen al robot la ejecución de las acciones para las cuales está concebido. En este último caso, el propio robot es a su vez una computadora.

En la actualidad, los avances tecnológicos y científicos no han permitido todavía construir un robot realmente inteligente, aunque existen esperanzas de que esto sea posible algún día.

Hoy por hoy, una de las finalidades de la construcción de robots en su intervención en el proceso de fabricación, estos robots, son los encargados de realizar los trabajos repetitivos en las cadenas de procesos de

fabricación como por ejemplo: pintar al spray, moldear a inyección, soldar carrocerías de automóvil, trasladar materiales, etc.

2.5.2 Tipos de robots

Androides

Una visión ampliamente compartida es que todos los robots son “androides”. Los androides son artilugios que se parecen y actúan como seres humanos.

Los robots de hoy en día vienen en todas las formas y tamaños, pero a excepción de los robots que aparecen en las ferias y espectáculos, no se parecen a las personas y por tanto no son androides.

Actualmente, los androides reales sólo existen en la imaginación y en las películas de ficción.



Fig. 2.1 Robots Androides

Móviles

Los robots móviles están provistos de patas, ruedas u orugas que los capacitan para desplazarse de acuerdo su programación.

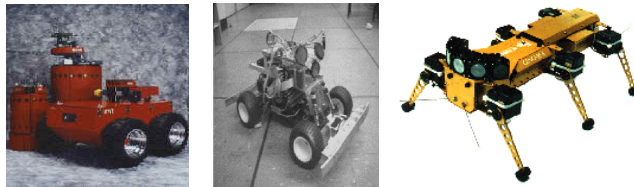


Fig. 2.2 Robots Móviles

Elaboran la información que reciben a través de sus propios sistemas de sensores y se emplean en determinado tipo de instalaciones industriales, sobre todo para el transporte de mercancías en cadenas de producción y almacenes. También se utilizan robots de este tipo para la investigación en lugares de difícil acceso o muy distantes, como es el caso de la exploración espacial y las investigaciones o rescates submarinos.

Médicos

Los robots médicos son, fundamentalmente, prótesis para disminuidos físicos que se adaptan al cuerpo y están dotados de potentes sistemas de mando. Con ellos se logra igualar con precisión los movimientos y funciones de los órganos o extremidades que suplen.



Fig. 2.3 Robot médico

Industriales

Los robots industriales son artulugios mecánicos y electrónicos destinados a realizar de forma automática determinados procesos de fabricación o manipulación. Los robots industriales, en la actualidad, son con mucho los más frecuentemente encontrados. Japón y Estados Unidos lideran la fabricación y consumo de robots industriales siendo Japón el número uno.



Fig. 2.4 Robot industrial

Teleoperadores

Hay muchos "parientes de los robots" que no encajan exactamente en la definición precisa. Un ejemplo son los teleoperadores. Dependiendo de

cómo se defina un robot, los teleoperadores pueden o no clasificarse como robots.

Los teleoperadores se controlan remotamente por un operador humano. Cuando pueden ser considerados robots se les llama "telerobots". Cualquiera que sea su clase, los teleoperadores son generalmente muy sofisticados y extremadamente útiles en entornos peligrosos tales como residuos químicos y desactivación de bombas.

2.5.3 Robots Móviles

Los robots han llegado a ser algo común en la industria, principalmente debido a la calidad y productividad en la automatización de las tareas industriales. Esto ha guiado a una importante innovación tanto en la investigación y desarrollo de los sistemas robóticos desde el advenimiento de los modernos microprocesadores. La mayoría de éstos sistemas son robots manipuladores u otros sistema fijos diseñados para trabajar en un espacio de trabajo limitado. En la actualidad la robótica se esta expandiendo hacia nuevas áreas y de hecho uno de estos campos es el área de la robótica móvil.

Fundamentación de los Robots móviles

Existe un gran número de diferencias entre los requerimientos de los tradicionales robots fijos y los requerimientos de los sistemas robóticos

móviles, uno de estas consideraciones primarias es la incertidumbre del espacio en el cual operará el vehículo. Para lo sistemas robóticos fijos un pequeño espacio de trabajo puede ser diseñado para facilitar que la tarea sea realizada. Para un sistema robótico móvil, es mucho más difícil realizar una ingeniería del espacio y a menudo se requiere que el sistema se adapte y opere en un amplio inestructurado y dinámico ámbito. Esto conlleva a la necesidad de desarrollar sensores y manejar la incertidumbre.

Los robots móviles de acuerdo a su definición deben poseer algún medio de locomoción, esto puede ser en forma de patas, ruedas, alas o algún otro mecanismo, el mismo que debe estar en función del espacio o ambiente en el cual el robot operará.

Elementos de un sistema robótico móvil

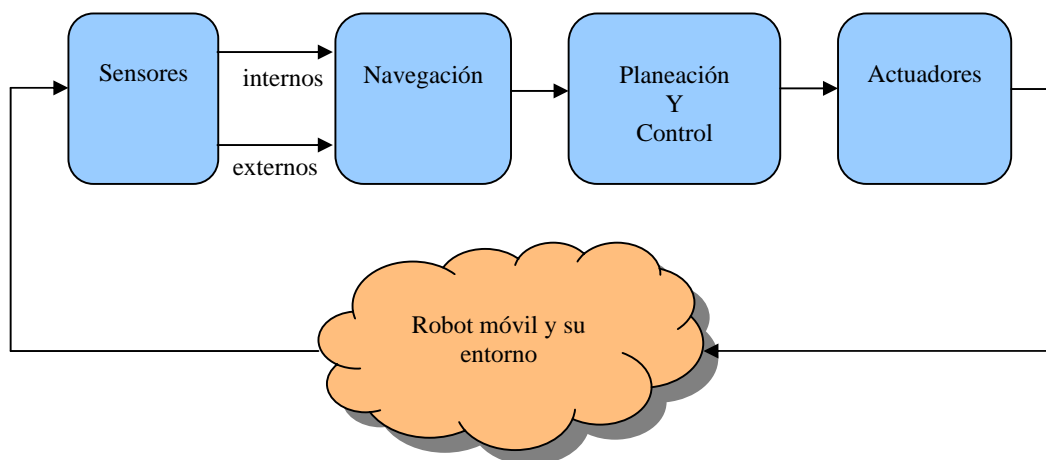


Fig. 2.5 Elementos del sistema robótico móvil

Sensores

El robot debe poseer algún medio para medir o percibir el entorno en que opera para anticipar y responder a cambios en el mismo, los sensores ayudan a la navegación y a la planeación. Estos sensores pueden ser internos o externos para proporcionar estimaciones sobre la posición del robot

Navegación

La navegación consiste en estimar su posición utilizando los datos obtenidos a través de los sensores, empleando para ello algoritmos y formulaciones matemáticas.

Planeación y Control

Guiar al robot móvil hacia la meta deseada requiere dos competencias fundamentales, primero, la habilidad de planear un camino usando la mejor información disponible acerca del entorno y segundo, la habilidad de controlar el movimiento del robot a lo largo del camino planeado evitando los obstáculos.

Los métodos usados para la planificación de la trayectoria del robot móvil son altamente dependientes de la aplicación, en robots que operan en un entorno conocido y estructurado como por ejemplo una fábrica, es posible planificar estas trayectorias con anterioridad al desarrollo del robot.

Entonces, la planificación de trayectorias se convierte en el problema de determinar la ruta más corta desde la localización actual del robot hasta la meta. En un robot que opera en un entorno no estructurado y parcialmente conocido, es necesario, que la planificación de la trayectoria sea realizada en línea.

Actuadores

Son aquellos usados para realizar el movimiento del robot a través de la trayectoria establecida en el módulo anterior.

2.5.4 El Problema De La Planificación De Trayectorias para Robots Móviles

La planificación se define como la búsqueda de una ruta libre de obstáculos desde una posición inicial hasta otra final a través del entorno de trabajo del robot móvil. Esta operación se realiza mediante el uso de la información que se posee del entorno actualmente, la descripción de la tarea de navegación y algún tipo de metodología estratégica. Así, el planificador se define por el modelo del entorno y el algoritmo de búsqueda utilizado. El caso más simple consiste en considerar un entorno conocido en su totalidad, estático y modelado de forma geométrica mediante polígonos. Con estas apreciaciones resulta factible la aplicación un algoritmo de búsqueda en grafos, que emplee cierta función de coste para la obtención de la ruta.

Esquemáticamente, se puede definir el problema de la planificación como:
 Dada una posición inicial y una orientación y una posición final y su respectiva orientación de A en W, se debe obtener una trayectoria especificando una secuencia de posiciones y de orientaciones evitando contacto con los B_i , comenzando con la posición y orientación inicial y terminando en la posición final y su correspondiente orientación. Se asume que W y los obstáculos B_i son poliedros o polígonos fijos y A es un poliedro o polígono móvil con orientación fija. Se asume que el robot es el único objeto móvil en el espacio de trabajo y se ignoran las propiedades dinámicas del robot.

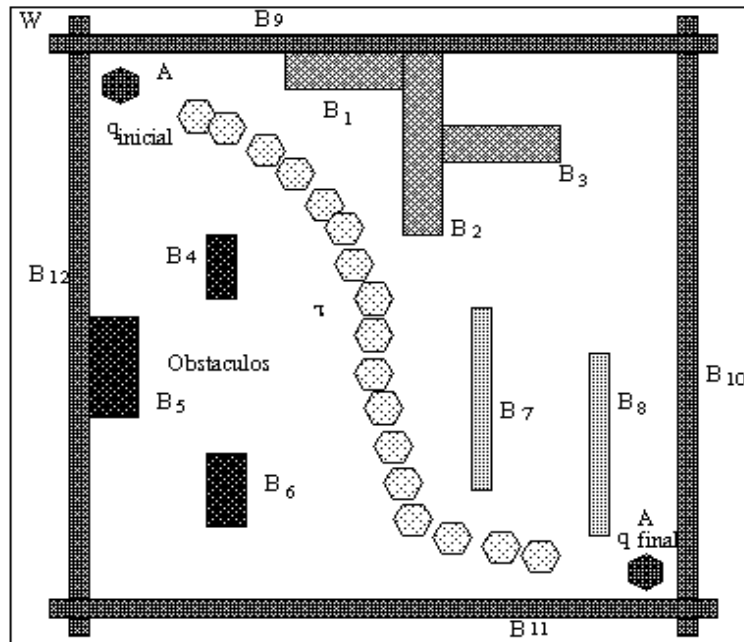


Fig. 2.6 Planificación de trayectorias

El problema de planificación de movimiento tiene que resolverse para cualquier tipo de robot que quiera desplazarse en un espacio físico.

2.5.5 Métodos Clásicos de Planificación

Todos ellos se fundamentan en una primera fase de construcción de algún tipo de grafo sobre el espacio libre, según la información poseída del entorno, para posteriormente emplear un algoritmo de búsqueda en grafos (por ejemplo tipo A*) que encuentra el camino óptimo según cierta función de coste.

- **Planificación basada en Grafos de Visibilidad**

Los grafos de visibilidad (Nilsson, 1.969) proporcionan un enfoque geométrico útil para resolver el problema de la planificación. Supone un entorno bidimensional en el cual los obstáculos están modelados mediante polígonos. Para la generación del grafo este método introduce el concepto de *visibilidad*, según el cual define dos puntos del entorno como *visibles* si y solo si se pueden unir mediante un segmento rectilíneo que no intersecte ningún obstáculo (si dicho segmento resulta tangencial a algún obstáculo se consideran los puntos afectados como visibles). En otras palabras, el segmento definido debe yacer en el espacio libre del entorno. Así, si se considera como nodos del grafo de visibilidad la posición inicial, la final y todos los vértices de los obstáculos del entorno, el grafo resulta de la unión mediante arcos de todos aquellos nodos que sean visibles.

- **Planificación basada en diagramas de Voronoi.**

Al contrario que los métodos basados en grafos de visibilidad, la planificación basada en diagramas de Voronoi sitúa la ruta lo más alejada posible de los obstáculos. Con ello elimina el problema presentado por los grafos de visibilidad de construir rutas semilibres de obstáculos.

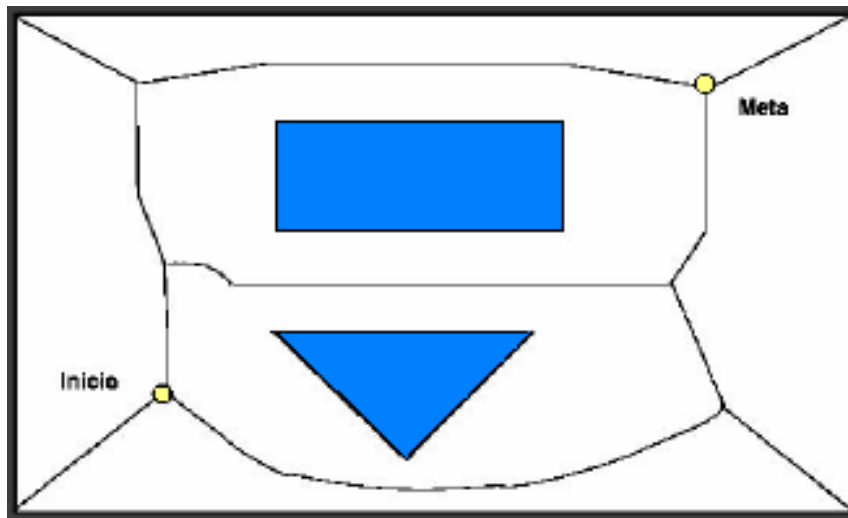


Fig. 2.7 Diagrama de Voronoi

- **Planificación basada en modelado del espacio libre.**

Se aplica a arquetipos de entornos con obstáculos poligonales, y la planificación en este caso se realiza mediante el modelado del espacio libre. Esta acción se lleva a cabo por los denominados cilindros rectilíneos generalizados (CRG). Al igual que los diagramas de Voronoi, con el uso de los CRG se pretende que el vehículo navegue lo más alejado de los obstáculos. De forma que la ruta que lleve al robot desde una configuración inicial hasta otra final estará compuesta por una serie de CRG

interconectados, de tal modo que la configuración de partida se encuentre en el primer cilindro de la sucesión y la final en el último.

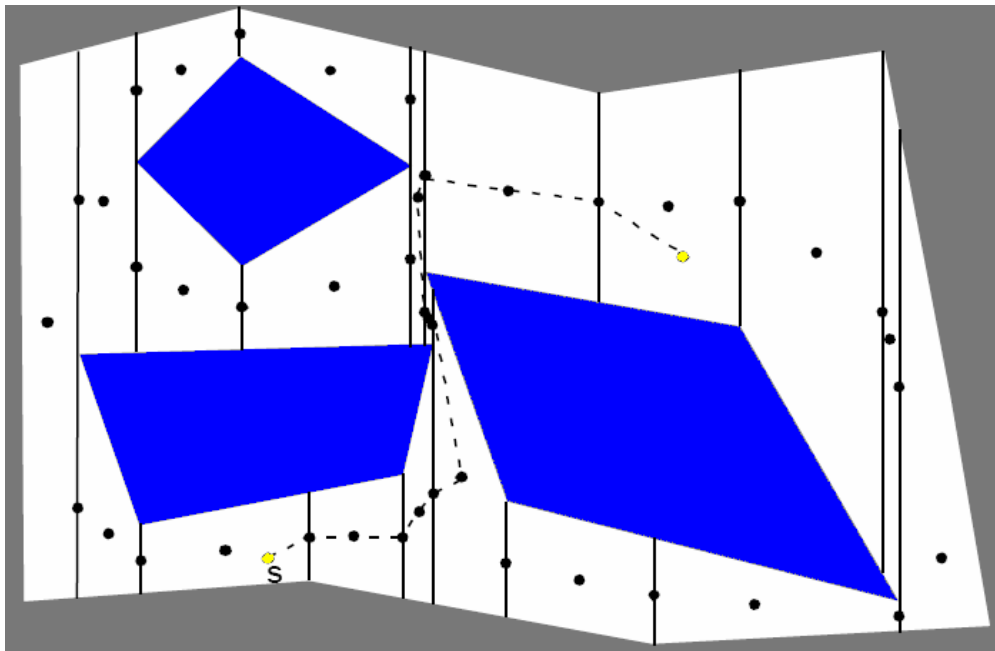


Fig. 2.8 Diagrama de modelado del espacio libre

- **Planificación basada en la descomposición en celdas.**

Este tipo de métodos se fundamenta en una descomposición en celdas del espacio libre. Así, la búsqueda de una ruta desde una postura inicial qa hasta otra final qf , consiste en encontrar una sucesión de celdas que no presente discontinuidades, tal que la primera de ellas contenga a qa y la última a qf . Al contrario que los métodos expuestos a lo largo de este apartado, no encuentra una serie de segmentos que modele la ruta, sino una sucesión de celdas; por ello, se hace necesario un segundo paso de construcción de un *grafo de conectividad*, encargado de definir la ruta.

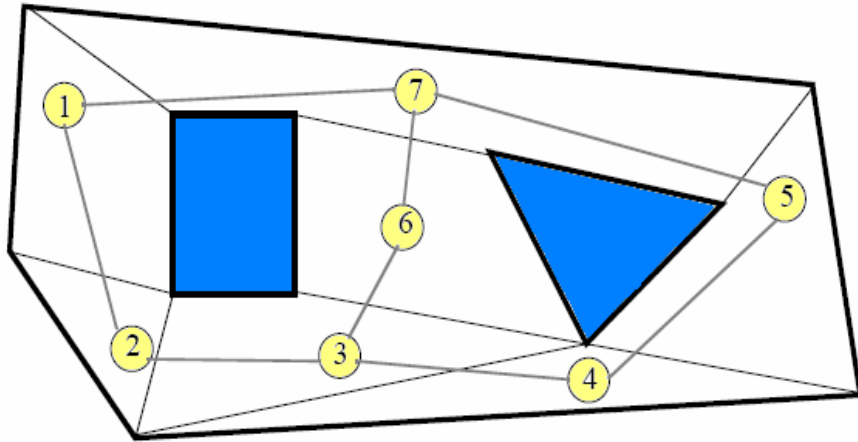


Fig. 2.9 Descomposición en celdas y grafo de conectividad

- **Planificación basada en campos potenciales.**

Los métodos basados en campos potenciales poseen una concepción totalmente distinta a los expuestos más arriba al estar basados en técnicas reactivas de navegación. El ámbito de uso de esta técnica se centra en la planificación local en entornos desconocidos, como puede ser el sorteo en tiempo real de obstáculos o de los que no se tiene constancia.

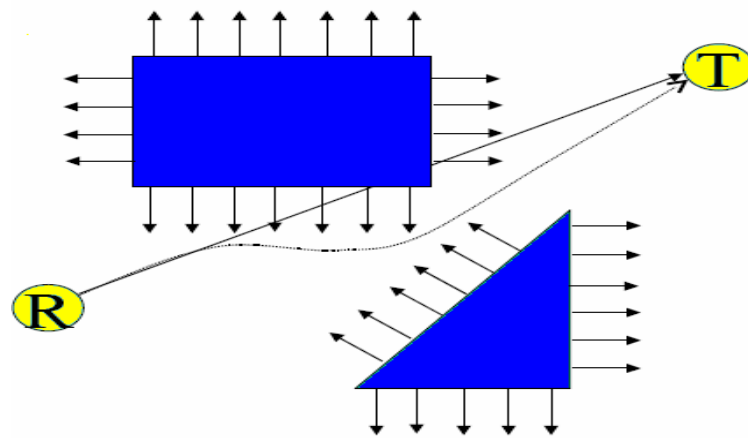


Fig. 2.10 Campos de potencial

2.5.6 Método de Grafo De Visibilidad

Uno de los métodos más antiguos, une vértices de obstáculos (polígonos) mediante rectas que no atraviesan obstáculos.

Los grafos de visibilidad proporcionan un enfoque geométrico para solventar el problema de la planificación. Este método se encuentra muy extendido debido a que opera con modelos poligonales de entorno, con lo que existen algoritmos que construyen esta clase de grafos con un coste computacional relativamente bajo. Este método necesita modelos de entornos definidos con polígonos, y puede trabajar tanto en el plano como en el espacio.

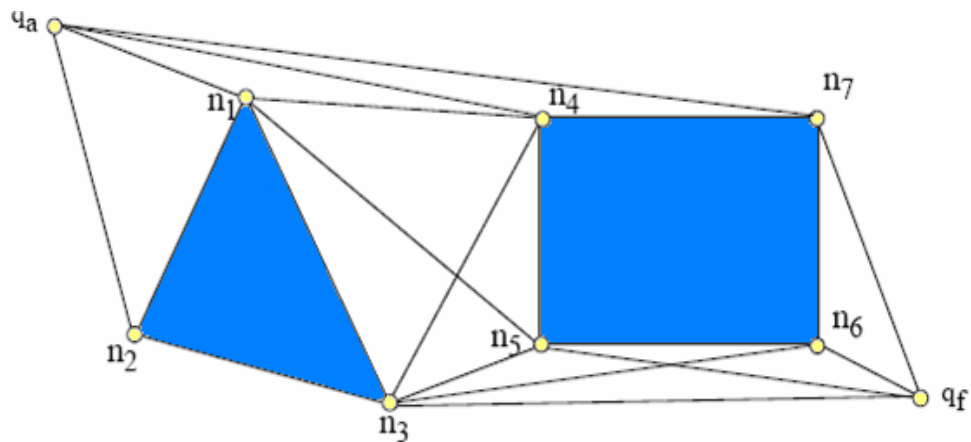


Fig. 2.11 Grafo de visibilidad en un entorno de dos obstáculos.

A continuación se formaliza el primer tipo, ya que el segundo queda fuera del ámbito de esta tesis, se resume brevemente el algoritmo global basado en el grafo de visibilidad para encontrar el camino mínimo entre dos puntos dados.

Los pasos a realizar en este algoritmo son:

1. Determinación de la posición final.
2. Determinar las coordenadas de los vértices de los obstáculos.
 - (a) Si el contorno de los obstáculos no es un polígono convexo hay que calcular el polígono convexo envolvente.
 - (b) Se ensanchan los polígonos convexos resultantes con las dimensiones del robot. (Sumas de Minkowski)
 - (c) Analizar si existe solape entre obstáculos tras el ensanchamiento de los polígonos y en caso de ser axial se unen como si fuesen un único obstáculo y se aplica de nuevo el cálculo envolvente.
3. Construir el grafo de visibilidad. Para todos los obstáculos entre el robot y el objetivo, empezando por la posición del robot se van trazando las rectas que unen los vértices de distintos obstáculos y se comprueban si cortan o no las aristas de algún obstáculo. En caso negativo, se van almacenado las distancias entre vértices y coordenadas respectivas generalmente en una estructura adecuada.
4. Búsqueda del camino óptimo. Para ello se utilizara cualquiera de los algoritmos de cálculo de distancia para grafos: Dijkstra, Bellman, etc.

Tras todos los pasos se obtendrá como resultado el camino mínimo a seguir libre de obstáculos.

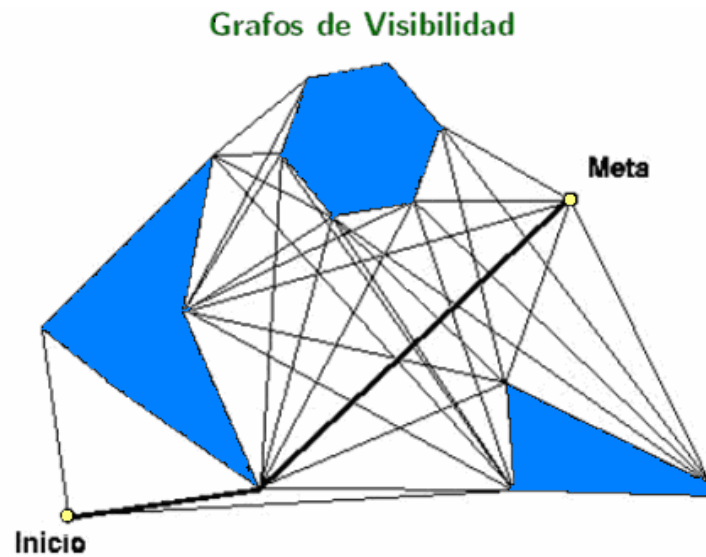


Fig. 2.12 Grafo de visibilidad y camino mínimo

2.6 Geometría computacional

2.6.1 Definición

La Geometría Computacional es una disciplina que se ocupa del diseño y análisis de algoritmos que resuelven problemas geométricos. En ella concurren elementos puramente matemáticos con cuestiones y herramientas propias de la informática. Se desprende por tanto que es un campo bastante reciente (20 - 30 años). Sin embargo, sus antecedentes pueden encontrarse en la Grecia Clásica, hace 2600 años, donde los problemas geométricos que se planteaban los matemáticos de la época eran abordados desde el punto de vista constructivo, es decir algorítmico.

2.6.2 Objetivos.

El objetivo general es el desarrollo de herramientas (estructuras de datos, análisis, diseño e implementación de algoritmos) para cualquier problema que sea susceptible de una representación geométrica. Los objetivos particulares dependen del área de aplicación y del problema a analizar. Así, para fijar ideas, la Geometría Computacional proporciona herramientas para los tópicos siguientes:

- (1) Gestión de grafos.
- (2) Descomposiciones de los objetos geométricos en objetos mas simples.
- (3) Identificación, Búsqueda, Localización de primitivas básicas.
- (4) Agrupamiento de datos en primitivas intermedias.
- (5) Estimación de relaciones topológicas de adyacencia.
- (6) Optimización geométrica. Programación lineal.
- (7) Tratamiento de Imagen.
- (8) Inspección. Identificación de patrones.
- (9) Generación de entornos. Simulación.
- (10) Robótica.
- (11) Análisis del movimiento.

Algunas aplicaciones relacionadas con estos tópicos son:

- (1) Grafos de Visibilidad, Grafos equilibrados, Quadtrees, Octrees.

- (2) Triangulaciones, trapezoidizaciones.
- (3) Reconocimiento, Generación de mallas, Detección de hechos significativos.
- (4) Envolvertes convexas.
- (5) Etiquetado según atributos para la interpretación semántica.
- (6) Toma de decisiones. Manufactura.
- (7) Imágenes Biomédicas, Sistemas de Información Geográfica.
- (8) Control de calidad asistido.
- (9) Reconstrucción 3D a partir de múltiples vistas. Animación asistida.
- (10) Planificación de movimientos. Cinemática y Dinámica computacionales.
- (11) Control de tráfico. Navegación automática.

2.6.3 Aplicaciones

Uno de los aspectos más interesantes de la Geometría Computacional es la gran aplicabilidad de sus resultados.

El significado del término computación se ha expandido notoriamente desde la introducción de los ordenadores, hará ahora unos cincuenta años.

Atendiendo a los objetos que procesan, destacan tres tipos de aplicaciones de los ordenadores. La primera generación va a ser la de los cálculos numéricos, aplicados sobre todo a problemas científicos y técnicos. La segunda, propiciada por necesidades más comerciales y administrativas,

incorporaba largas listas de datos (por ejemplo alfabéticos), con vistas a cómo leer, almacenar, modificar, seleccionar, e imprimir esos datos.

Todo esto naturalmente pervive, y con fuerza, pero se vive una tercera generación de aplicaciones dominada por el procesamiento de información geométrica y gráfica, presente en áreas tan diversas como son la medicina, la cartografía, el control de robots o el diseño artístico. La Geometría Computacional ha emergido, ciertamente, por la necesidad de dar respuesta a esta nueva y creciente demanda.

Se podría decir que las aplicaciones van a preceder la disciplina, y ahora que ésta tiene ya un núcleo teórico sólidamente constituido, como sus vertientes prácticas corresponden a tecnología de máxima vanguardia, la demanda de resultados continúa con la misma fuerza y exigencia que al principio. Por eso se dice que, en Geometría Computacional, las aplicaciones tienen un protagonismo esencial.

Dentro de los campos de aplicación más directamente relacionados con la disciplina destacaremos:

- La Informática Gráfica,
- El Diseño y Fabricación Asistida por Ordenador (CAD/CAM),
- La Caracterización y Reconocimiento Automático de Formas (Pattern Recognition),
- El Diseño VLSI,

- La Visión Artificial,
- La Cartografía y
- La Robótica.

Esta última en especial objeto de la presente tesis, utiliza una combinación del Algebra y la Geometría Computacional, puesto que para la planificación de movimientos requiere una combinación de técnicas procedentes de ambas áreas, donde es posible mostrar la utilidad de una aproximación matemática al problema.

2.7 Metodología MSF

2.7.1 Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

2.7.2 Características de MSF

MSF tiene las siguientes características:

- **Adaptable:** es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.
- **Escalable:** puede organizar equipos tan pequeños entre 3 o 4 personas, así como también, proyectos que requieren 50 personas a más.
- **Flexible:** utilizada en el ambiente de desarrollo de cualquier cliente.
- **Tecnología Agnóstica:** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología.

2.7.3 Modelos

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación.

- **Modelo de Arquitectura del Proyecto:** Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- **Modelo de Equipo:** Este modelo ha sido diseñado para mejorar el rendimiento del equipo de desarrollo.

Proporciona una estructura flexible para organizar los equipos de un proyecto. Puede ser escalado dependiendo del tamaño del proyecto y del equipo de personas disponibles.

- **Modelo de Proceso:** Diseñado para mejorar el control del proyecto, minimizando el riesgo, y aumentar la calidad acortando el tiempo de entrega. Proporciona una estructura de pautas a seguir en el ciclo de vida del proyecto, describiendo las fases, las actividades, la liberación de versiones y explicando su relación con el Modelo de equipo.
- **Modelo de Gestión del Riesgo:** Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riesgos que puedan provocar.
- **Modelo de Diseño del Proceso:** Diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.
- **Modelo de Aplicación:** Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para

diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables, y pueden ser usados en un solo ordenador o incluso en varios servidores.

2.7.4 Modelo de Proceso de Desarrollo de Aplicaciones MSF

El Modelo de Proceso de Desarrollo de MSF describe un ciclo de vida que puede ser usado para desarrollar software de manera exitosa, estableciendo el orden en el cual se deben realizar las actividades. Como puede verse en la Figura 1.12, este modelo consiste en cinco fases distintas, cuyos nombres dependen del tipo del proyecto en el que se aplica. Cada fase del proceso de desarrollo culmina con un hito visible, tal como se describe a continuación:



Fig. 2.13 Modelo de Proceso de Desarrollo de Aplicaciones MSF.

- a) **Fase 1:** Visión. En esta fase el equipo y el cliente definen los requerimientos del negocio y los objetivos generales del proyecto. La fase culmina con el hito *Visión y Alcance aprobados*.

- b) **Fase 2:** Planeación. Durante la fase de planeación el equipo crea un borrador del plan maestro del proyecto, además de un cronograma del proyecto y de la especificación funcional del proyecto. Esta fase culmina con el hito *Plan del proyecto aprobado*.

- c) **Fase 3:** Desarrollo. Esta fase involucra una serie de *releases* internos del producto, desarrollados por partes para medir su progreso y para asegurarse que todos sus módulos o partes están sincronizados y pueden integrarse. La fase culmina con el hito *Alcance completo*.

- d) **Fase 4:** Estabilización. Esta fase se centra en probar el producto. El proceso de prueba hace énfasis en el uso y el funcionamiento del producto en las condiciones del ambiente real. La fase culmina con el hito *Release Readiness aprobado*.

- e) **Fase 5:** Implantación: En esta fase el equipo implanta la tecnología y los componentes utilizados por la solución, estabiliza la implantación, apoya el funcionamiento y la transición del proyecto, y obtiene la aprobación final del cliente. La fase termina con el hito *Implantación completa*.

CAPITULO III

METODOLOGÍA DEL PROYECTO

3.1 FASE DE VISION

3.1.1 Planteamiento del problema

En la actualidad, habitualmente, la enseñanza de una asignatura técnica necesita el uso de materiales de laboratorio caros, por lo que, en muchas ocasiones, éstos son insuficientes para todos los alumnos. Además, se requiere que los alumnos vayan a los laboratorios a realizar las prácticas, debiéndose acomodar a los horarios ofrecidos. Así, surge la necesidad de métodos alternativos, como los laboratorios virtuales.

Un laboratorio virtual provee una forma de realizar experimentos sobre un equipamiento que no está físicamente presente, permite la creación de espacios virtuales donde los estudiantes puedan llevar a cabo las prácticas de laboratorio.

Los laboratorios virtuales son herramientas de apoyo a la enseñanza en ciertas áreas, en éste caso de la Investigación de la Robótica Móvil, a partir de estos hechos, y si se añade la participación de un tutor inteligente para

complementar la labor de los laboratorios virtuales se pueden obtener ventajas como:

- La información siempre está disponible y completa en forma de lecciones o ayuda.
- Retroalimentación inmediata cuando hay errores o conceptos mal entendidos.

El problema consiste en construir un laboratorio virtual en donde se expliquen los conceptos teóricos de la robótica móvil, se aborde el problema de la planeación de trayectorias que consiste en: dada una posición inicial y una posición final, obtener una trayectoria de menor distancia posible evitando colisiones con los obstáculos, asumiendo que el robot es el único objeto móvil en el espacio de trabajo e ignorando las propiedades dinámicas del robot; a través de un simulador. El laboratorio debe permitir además realizar la programación del robot en un lenguaje de alto nivel validando para ello el código ingresado y la teleoperación del mismo pudiendo visualizar en pantalla su movimiento.

3.1.2 Visión de la solución

Se requiere implantar un laboratorio virtual de robótica móvil para inicializar a los estudiantes dentro de los conceptos de Robótica Móvil y garantizar su aprendizaje. Los laboratorios virtuales ofrecen muchas ventajas para la enseñanza de materias tecnológicas, tales como la flexibilidad de horarios y acceso de los estudiantes a un equipamiento caro y limitado.

El principal propósito de este sistema es proveer una herramienta educativa entretenida para la audiencia que consiste en estudiantes de ingeniería, el cliente esta en proceso de acceder a un robot capaz de ser controlado remotamente, programar dicho robot, visualizar una simulación de un método de planificación de trayectorias y además acceder a contenidos teóricos más relevantes de la robótica.

Esta aplicación permitirá a los estudiantes controlar y aprender acerca de los robots móviles, para esto se creará un tutorial en formato html de fácil navegación para los estudiantes, para impartir la teoría correspondiente.

Se incluirá un simulador de la planificación de trayectorias de robots móviles para el método de planificación de grafo de visibilidad que permita dos modos de acceso: simulación automática y práctica guiada con interacción del estudiante, supervisada por un tutor.

Se proveerá de un entorno tanto para programar y descargar código al robot, que se comportará como un compilador, el lenguaje en el que se programará al móvil será C#, y otro entorno para realizar una teleoperación libre, en el cual se podrá visualizar los movimientos del robot a través de una cámara de video.

El robot móvil al que se hace referencia en la programación y teleoperación será un vehículo ensamblado usando el kit de robótica de Lego Mindstorms.

3.1.3 Metas del proyecto

Se espera que con el proyecto se puedan cumplir las siguientes metas:

- Generar una herramienta de software educativo de calidad que permita:
 - Implementación de la simulación del cálculo de la planificación de trayectorias de robot móvil basado en grafos de visibilidad.
 - Creación de un entorno para la teleoperación libre del robot con movimientos básicos.
 - Establecimiento de un modo de práctica guiada por un tutor inteligente.
 - Definición de un entorno de programación de robots en lenguaje C#.
 - Presentación de los contenidos teóricos sobre robótica móvil en formato CAI de fácil navegación.

- Desde el punto de vista docente:
 - Que el alumno pueda poner en práctica los conocimientos adquiridos en la asignatura de robótica mediante sencillos ejemplos simulados.
 - Que la herramienta ayude al alumno a comprender la teoría subyacente bajo los algoritmos de planificación y seguimiento de trayectorias, los modelos cinemáticos y la evitación de obstáculos.

3.1.4 Matriz de tradeoffs del proyecto

En la siguiente tabla se muestra la matriz de tradeoffs del proyecto de laboratorio virtual de robótica móvil con tutor inteligente:

	Restricción por optimizar	Restricción negociable	Restricción aceptada
Recursos			X
Cronograma		X	
Requerimientos	X		

Tabla 3.1 Matriz de tradeoffs

En la Tabla 3.1 se indica que durante esta Fase, se aceptó los recursos con los que se disponía para llevar a cabo el proyecto, mientras que el cronograma propuesto quedó abierto a posibles cambios junto con el conjunto de requerimientos finales del producto, que pueden aparecer durante las siguientes fases.

3.1.5 Alcance

El laboratorio virtual de robótica móvil debe:

1. Proporcionar los contenidos teóricos de la robótica móvil.
2. Simular la navegación de un robot en un entorno conocido con una serie de obstáculos donde el alumno puede modificar la

configuración del entorno, el punto de origen y destino; y visualizar interactivamente el efecto que la modificación de estos tiene sobre la trayectoria elegida para ir del punto inicial al final y sobre el seguimiento que el robot hace a dicha trayectoria. El método para la planificación de trayectorias utilizado será el grafo de visibilidad.

3. Permitir la interacción del alumno con el simulador a través de una práctica guiada supervisada por un tutor inteligente en donde paso a paso vaya desarrollando un ejercicio de planificación.
4. Facultar la programación del robot móvil Lego en lenguaje C#.
5. Posibilitar la teleoperación del vehículo robótico con movimientos básicos.

3.1.6 Lista de riesgos

Se ha identificado una lista de tres riesgos que poseen el potencial de causar que el proyecto realice modificaciones en el cronograma preestablecido.

A cada riesgo fue asignando una probabilidad de ocurrencia y una medida del impacto, multiplicando estos factores dan como resultado el riesgo total, calculado el riesgo total de los tres riesgos se asume que se atenderán primero las condiciones cuyo riesgo total sea más alto.

A continuación se definen los posibles riesgos que pueden aparecer durante el desarrollo del proyecto:

Condición/ Descripción	Falta de información acerca del tema	Complejidad de los algoritmos	Limitaciones de la interfaz que controla el robot
Consecuencia	Retraso en la entrega del producto final	Retraso en la implementación	Limitadas directivas para manipular el robot móvil
Probabilidad	Alto (3)	Alto(3)	Medio (2)
Impacto	Alto (3)	Alto(3)	Medio (2)
Prioridad	Alta (3)	Alta(3)	Baja (1)
Mitigación	Identificar las personas o instituciones encargadas de proveer información y establecer contactos	Asignar más tiempo a la implementación	Buscar un mejor componente
Riesgo Total	27	27	4

Tabla 3.2 Lista de riesgos.

3.1.7 Esquema de la solución

A continuación se presenta el esquema de la solución en el cual se puede observar que el Laboratorio virtual de robótica móvil se compondrá de cuatro subsistemas, el subsistema Tutorial que contendrá la teoría subyacente a la robótica móvil en formato HTML, los subsistemas de

Simulación de planificación de trayectorias, Entorno de programación y de teleoperación de tipo aplicación Windows forms, estos dos últimos tendrán comunicación con el robot móvil.

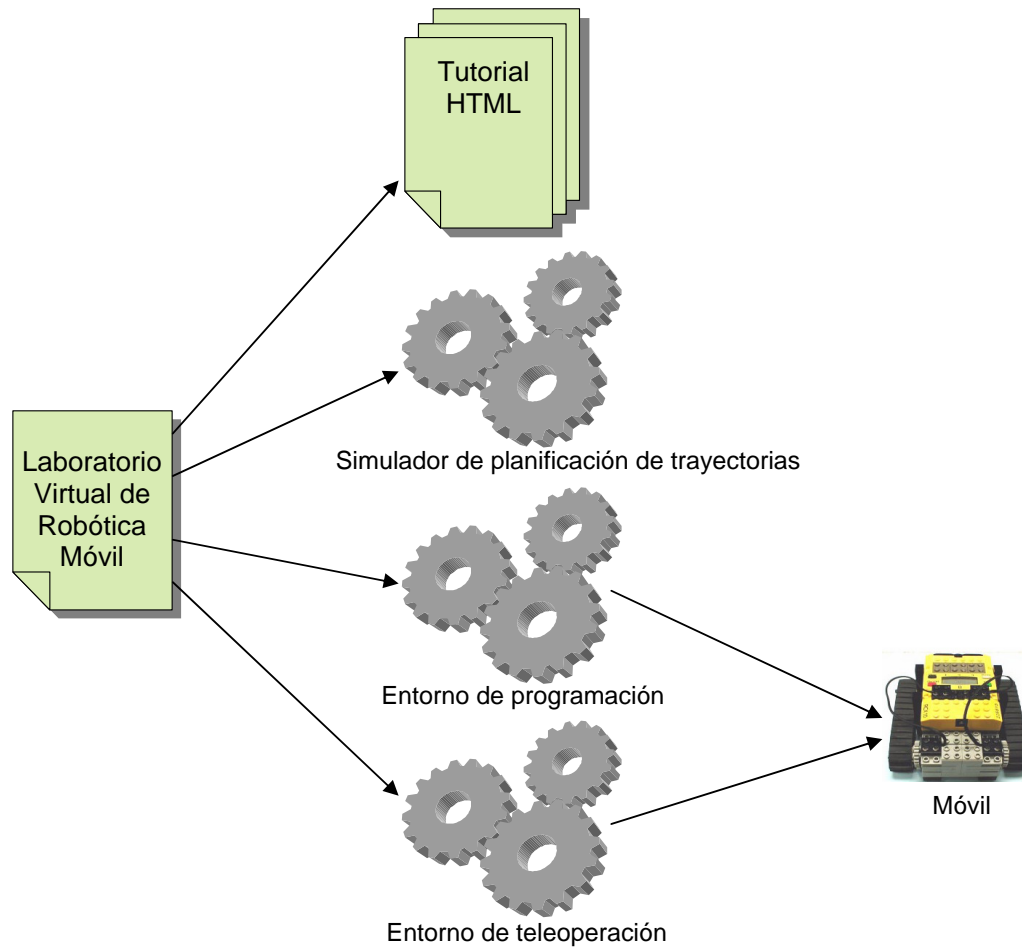


Fig. 3.1 Esquema de la solución

3.1.8 Visión aprobada

Al culminar la fase de visión, en este punto, se ha clarificado la dirección del proyecto, las características que incluirá y no incluirá la solución.

3.2 FASE DE PLANEACION

3.2.1 Requerimientos del sistema

De la fase anterior se desprenden los requisitos del sistema tras haber analizado el planteamiento del problema y establecido la visión de la solución, éstos se muestran a continuación:

No. Requerimiento	Descripción Requerimiento
R1	Mostrar contenidos teóricos estructurados
R2	Imprimir apartados
R3	Realizar evaluaciones de aprendizaje
R4	Ingresar datos (punto inicial, final y polígonos)
R5	Ampliar los obstáculos
R6	Generar el grafo de visibilidad
R7	Calcular el camino mínimo
R8	Validar la comunicación con el móvil
R9	Realizar movimientos básicos
R10	Visualizar por video al móvil
R11	Ingresar código
R12	Compilar código
R13	Ejecutar código

Tabla 3.3 Listado de requerimientos del sistema

3.2.2 Diseño Conceptual

Perfil de los usuarios

De la fase anterior se identificó que en este sistema existirá un único usuario, cuyo perfil se describe a continuación:

Ubicación:	Nivel de educación universitario
Capacitación:	Experiencia en el uso de computadores y software en general. Conocimientos previos de cálculo. Habilidad y experiencia en la programación orientada a objetos.
Expectativas:	Contar con una herramienta que facilite la comprensión de la teoría y fundamentos matemáticos de la robótica móvil, en particular de la planeación de trayectorias.

Tabla 3.4 Perfil del usuario

Escenarios de uso

Se han identificado cuatro escenarios de uso del sistema, de acuerdo al Diagrama de casos de uso de la figura 3.2, cada uno de ellos se describe a continuación.

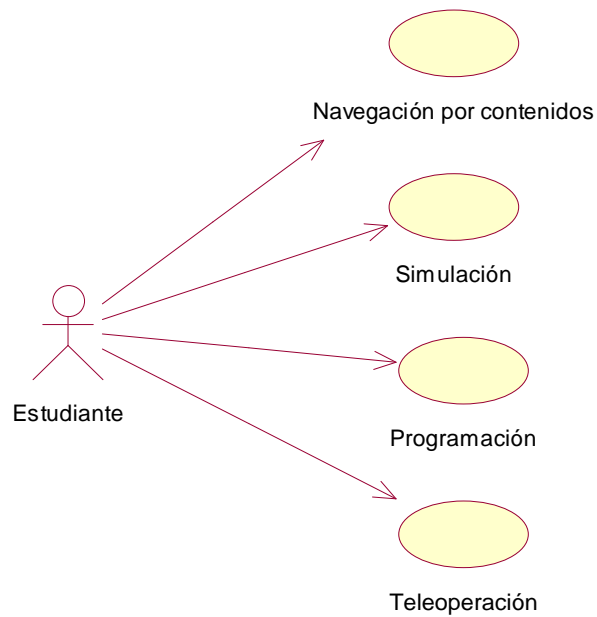


Fig. 3.2 Diagrama de Casos de Uso

Descripción de los casos de uso

Nombre	Navegación por contenidos
Actores	Estudiante
Función	Mostrar los contenidos teóricos del curso de robótica móvil
Descripción	El estudiante puede acceder libremente a los tópicos del curso, imprimirlos si lo desea, al finalizar un apartado puede tomar un test de evaluación para verificar su aprendizaje.
Referencias	R1, R2, R3
Precondición	Haber instalado correctamente el sistema.
Poscondición	El usuario adquiere los conocimientos necesarios sobre robótica móvil y esta en la capacidad de manipular los otros módulos que componen el sistema.

Tabla 3.5 Caso de Uso Navegación por contenidos

Nombre	Simulación
Actores	Estudiante
Función	Realizar la simulación de la planificación de trayectorias
Descripción	El estudiante puede escoger entre los modos automático y práctica guiada, en el primero ingresará datos, obtendrá el resultado y visualizará la simulación, en el segundo, interactuará paso a paso con el simulador guiado por un tutor.
Referencias	R4, R5, R6, R7
Precondición	Tener las bases teóricas sobre planificación de trayectorias
Poscondición	Simulación y/o práctica realizada satisfactoriamente.

Tabla 3.6 Caso de Uso Simulación

Nombre	Teleoperación
Actores	Estudiante
Función	Manejo del robot móvil en un entorno libre
Descripción	El estudiante luego de establecer comunicación con el móvil puede operarlo o guiarlo a través de movimientos básicos por un entorno libre de obstáculos y visualizando a través de una cámara de video los movimientos ejecutados por el robot.
Referencias	R8, R9, R10
Precondición	Tener la torre IR y la cámara de video conectada al computador y el robot móvil encendido.
Poscondición	El robot móvil realiza los movimientos definidos por el usuario y se visualizan a través de la cámara de video.

Tabla 3.7 Caso de Uso Teleoperación

Nombre	Programación
Alias	
Actores	Estudiante
Función	Compilar y ejecutar el código ingresado por el estudiante.
Descripción	El estudiante ingresa las instrucciones que luego compilará, si no existen errores, el código será ejecutado y descargado al robot móvil.
Referencias	R10, R11, R12, R13
Precondición	Conocimiento de programación orientada a objetos.
Poscondición	Código descargado al robot y ejecutado por el mismo.

Tabla 3.8 Caso de Uso Programación

Modelo Conceptual

A continuación se presenta el modelo conceptual con los paquetes que conforman la solución:

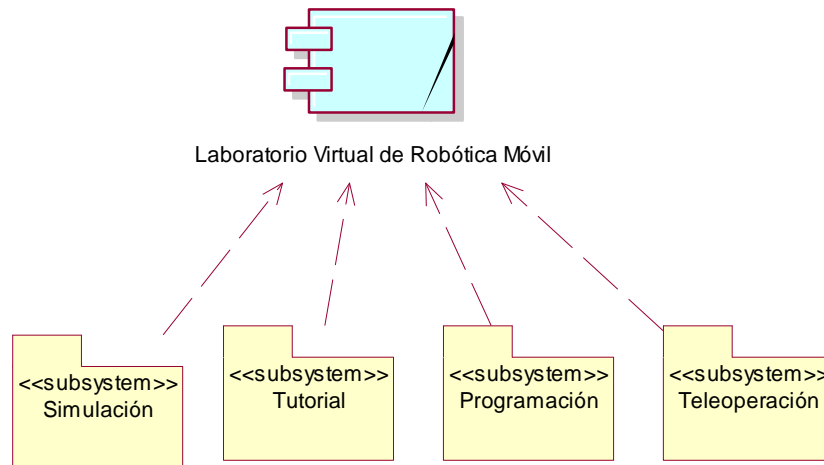


Fig. 3.3 Diagrama de estructura del sistema

Los diagramas restantes que conforman el modelado de la solución se pueden apreciar en el anexo A.

3.2.3 Diseño Lógico

Diseño de la interfaz de usuario

Módulo Tutorial

La interfaz de este subsistema se desarrollará tomando en cuenta normas y estándares para tener un tutorial de tipo web consistente. Los lineamientos a seguir son:

- Usar lenguaje HTML estándar.
- Tener en consideración el peso de las páginas para su adecuado despliegue.
- Diagramar las páginas usando tablas para estructurar tanto la información como los elementos gráficos.
- Usar una combinación de colores adecuada y gráficos alusivos al tema.

En la siguiente figura se muestra el diseño de la interfaz del tutorial.

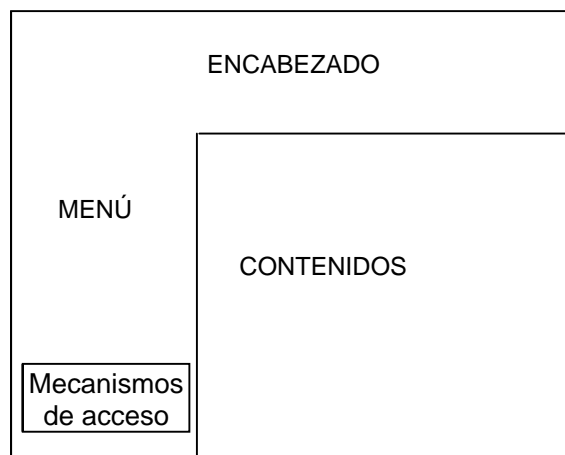


Fig. 3.4 Diseño interfaz de contenidos

Módulo de Simulación

Este módulo será de tipo Windows forms, se compone de dos partes, la simulación automática y la práctica guiada, en ambos casos el diseño será del tipo estándar de Windows procurando una disposición adecuada de los elementos gráficos dentro de la ventana para no sobrecargarla.

El modo de ingreso de datos será a través del Mouse debido al contexto de la aplicación, en el caso de la práctica guiada se utilizará además entrada de datos por teclado.

Las siguientes figuras muestran los diseños para estas interfaces.

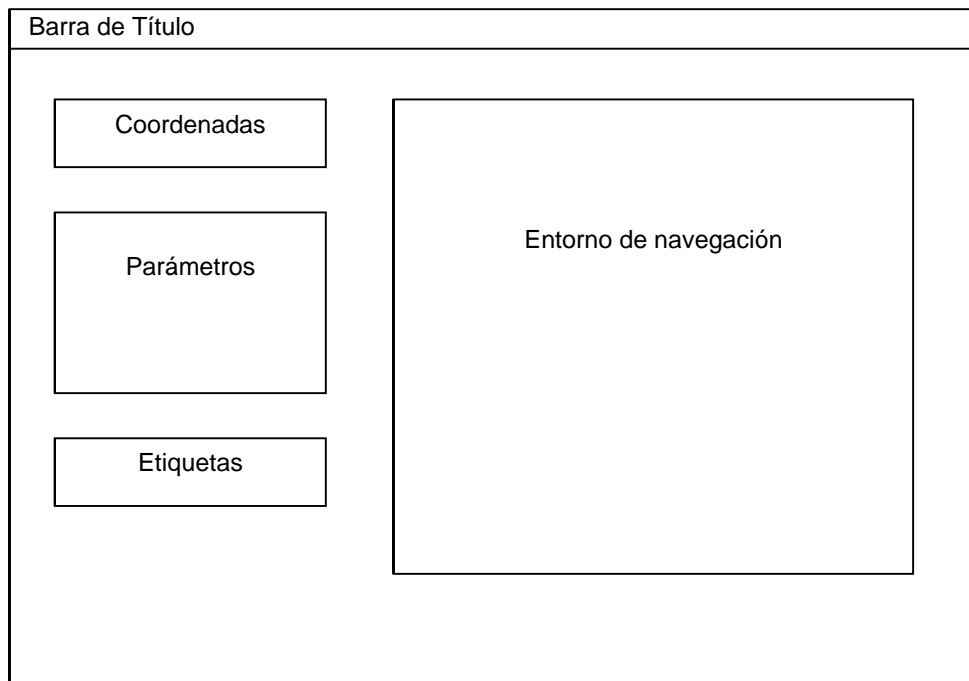


Fig. 3.5 Diseño interfaz de simulación automática

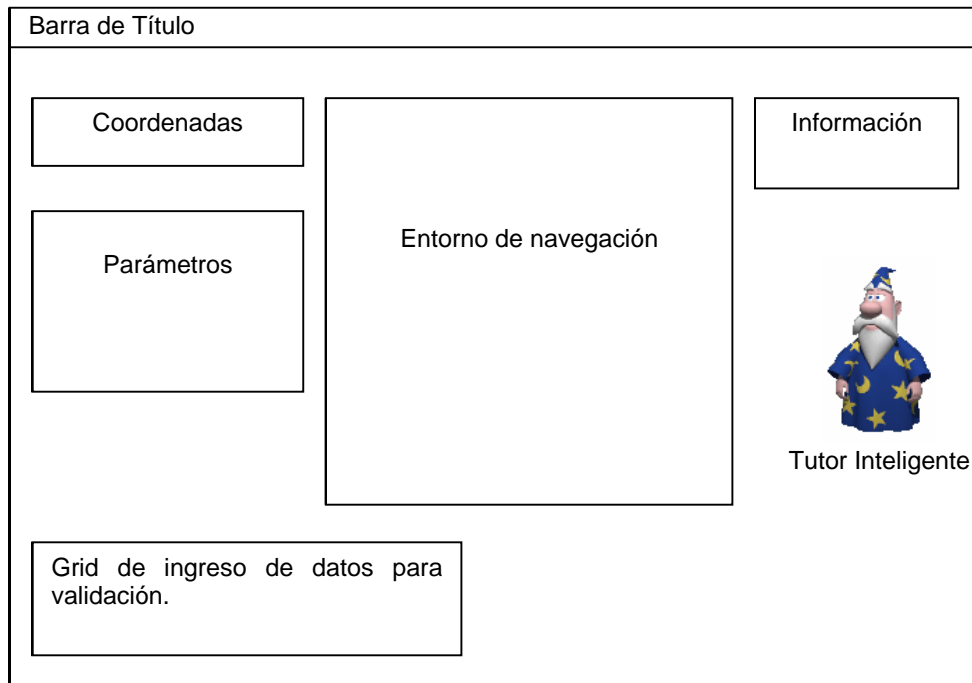


Fig. 3.6 Diseño interfaz de simulación - práctica guiada

Módulo de Teleoperación

La interfaz de este subsistema constará de una sola ventana en la cual se dispondrá de botones de control del vehículo robótico, un espacio de simulación en el cual se visualice la posición del mismo dentro del entorno de navegación y un control para obtener imágenes desde una cámara de video que permita observar la teleoperación del real equivalente.

En esta interfaz la iteración con el usuario se realizará a través del Mouse, a continuación se presenta el aspecto de la misma.

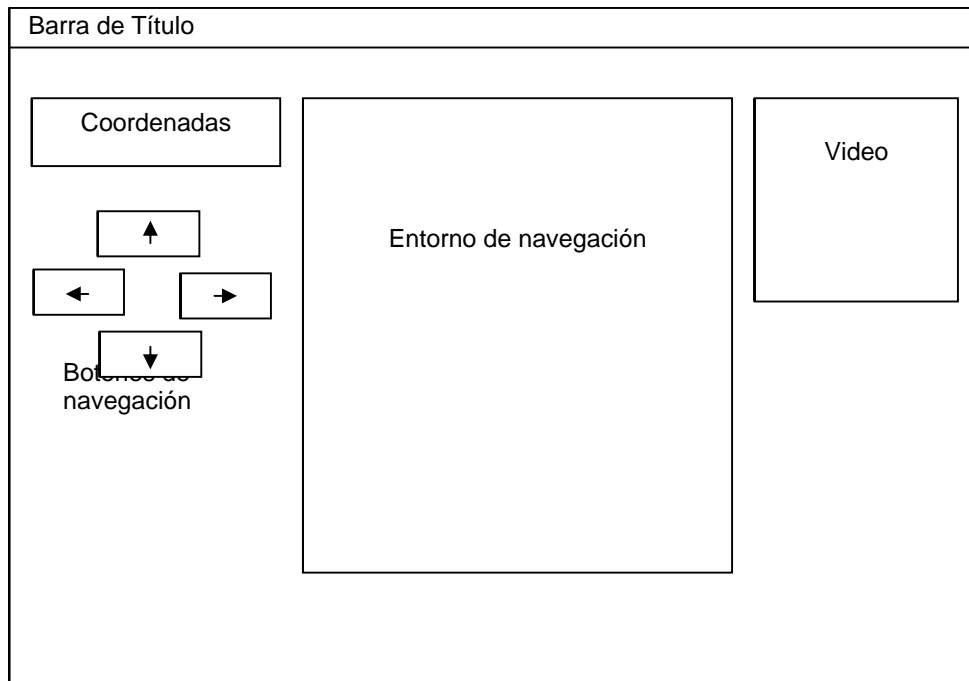


Fig. 3.7 Diseño interfaz de Teleoperación

Módulo de Programación

En este módulo la interfaz constará de una única ventana en la cual se utilizará un menú para operaciones con archivos, un control de texto enriquecido para el ingreso de las instrucciones en lenguaje c#, y cajas de texto para obtener datos necesarios para el programa.

Contará además con botones de comando para realizar la compilación y ejecución de los programas del usuario.

El esquema de esa interfaz se presenta como sigue:

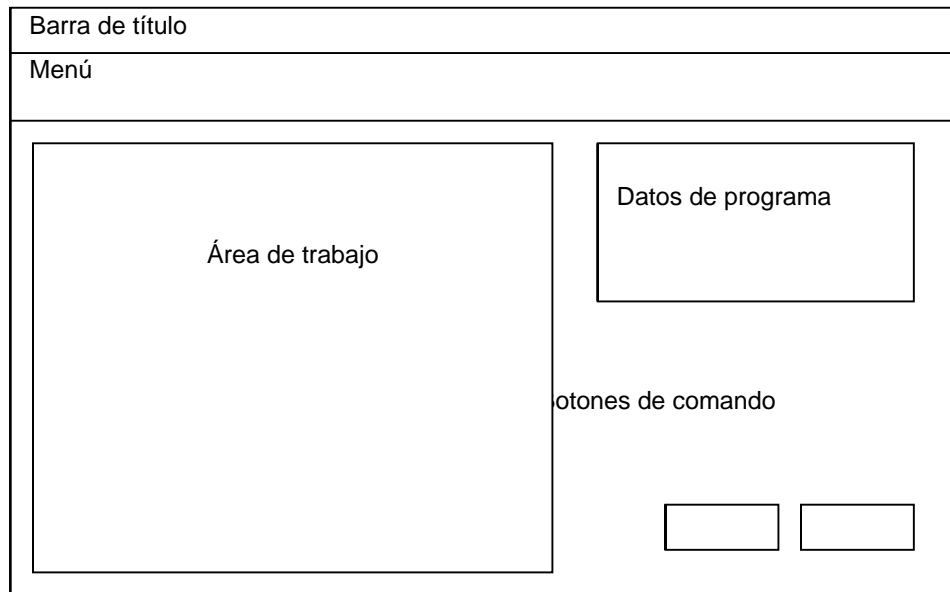


Fig. 3.8 Interfaz de programación

Componentes de la solución

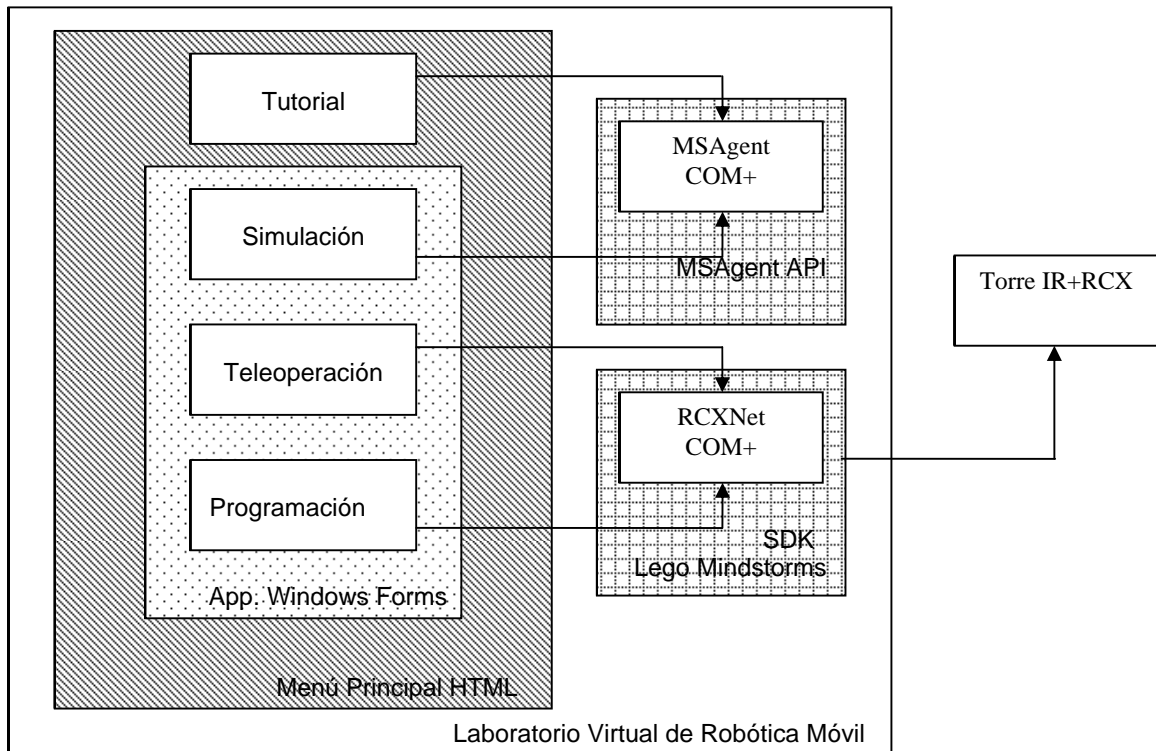


Fig. 3.9 Componentes de la solución

Como se puede apreciar en la figura anterior, el diagrama de los elementos que conforman la solución depende de los escenarios de uso que a su vez conforman cuatro subsistemas que la componen.

Los subsistemas Tutorial y Simulación harán uso de un control DLL ActiveX que descansa completamente en la tecnología COM y que hace uso de componentes del MSAgent API.

Los subsistemas de Teleoperación y Programación, por otro lado, harán uso de la DLL RCXNET, para lo cual es necesario contar con el SDK de Lego Mindstorms para permitir la interoperatividad con el robot móvil.

3.2.4 Diseño Físico

Restricciones de tecnología

1. El curso tutorial de robótica móvil será desarrollado en HTML.
2. Los módulos de simulación, programación y teleoperación serán desarrollados en Visual C#.NET.
3. Los módulos que requieran conectarse con el robot móvil lo harán a través de la librería RCXNET, pero para ello se debe tener preinstalado el SDK de Lego Mindstorms.
4. Para la obtención del video se lo hará a través de la API de Windows.

5. El tutor inteligente se programará utilizando un ActiveX MSAgent de tecnología COM+ para el caso de las aplicaciones Windows y el tutorial.

Descripción de herramientas

Macromedia Dreamweaver

Es la opción profesional para la creación de aplicaciones y sitios Web. Ofrece una combinación muy útil de herramientas de diseño visual, funciones de desarrollo de aplicaciones y soporte para la edición de código. Todo ello permite crear sitios y aplicaciones atractivos basados en estándares de forma rápida.

Visual C#.Net

Es una herramienta que tiene soporte RAD (Rapid Application Development) y provee características que lo convierten en un poderoso lenguaje orientado a objetos.

MSAgent

Los Microsoft Agent son sistemas de servicios programables que presentan características animadas dentro de una interfaz. Se puede utilizar esta tecnología como ayudante, asistente o entretenimiento. Como algunos controles, el *MS Agent* se puede implementar en páginas *Web*.

En síntesis, los *Agents* otorgan una nueva forma de interacción con el usuario, llamada conversacional. Esta interacción se asemeja a la comunicación humana utilizando voz, gestos y además no solamente responde a la entrada a través de teclado y el ratón, sino que también tiene la opción de reconocimiento de discurso o voz.

Lego MindStorms

LEGO Mindstorms™ surgió de la creación del MIT Media Laboratory, el kit de Lego incluye múltiples piezas, conectores, sensores, luces, cables y motores para la construcción del robot deseado. Incluye además el ladrillo programable de LEGO, también llamado RCX (Robotic Command Center), sobre el cual se colocan las piezas y demás dispositivos para construir el robot. El RCX es una microcomputadora con su CPU, RAM, puerto de comunicación serie por infrarrojo, botones, visor, puertos de entrada y salida. La programación se realiza desde el PC y el programa se almacena en el RCX a través de la torre de comunicación por infrarrojos conectada al PC en el puerto serie o USB.

Implementación de la interfaz de usuario

A continuación se muestra las interfaces de usuario implementadas para los diferentes módulos que componen la solución.

Pantalla Principal

Como se puede observar en la siguiente imagen, la pantalla principal contiene enlaces para acceder a los diferentes subsistemas que componen la solución.

El asistente personal realizará una introducción al laboratorio virtual. Esta interfaz se implementó en formato HTML.



Fig. 3.10 Interfaz Pantalla principal

Módulo Tutorial

La interfaz para el Tutorial de Robótica Móvil esta implementada en HTML estándar consta de un conjunto de frames.

En el frame superior se encuentra el encabezado, en el izquierdo se encuentran los enlaces para acceder a las diferentes lecciones que componen el tutorial y en su parte inferior mecanismos de acceso que facilitan la navegación. En el frame central se muestra el contenido de las lecciones.



Fig. 3.11 Interfaz del Tutorial

Módulo de Simulación

Al iniciar este módulo aparecerá una pequeña pantalla a través de la cual se puede escoger entre los dos modos de simulación.

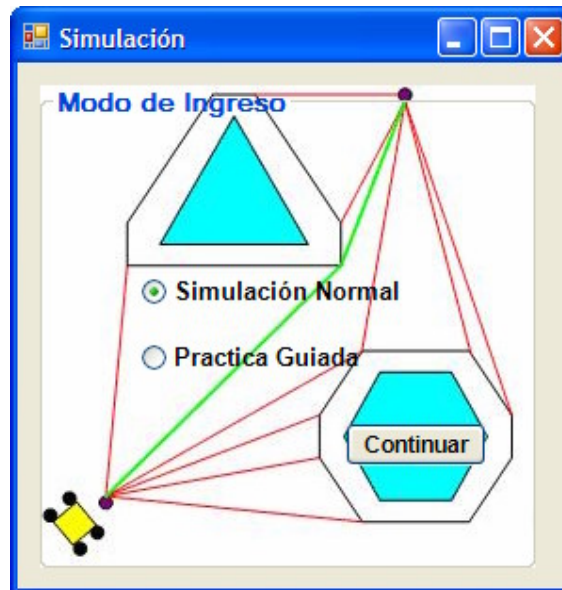


Fig. 3.12 Interfaz de Modo de Ingreso

Interfaz del Modo Automático

La interfaz del subsistema de Simulación se implementó con controles estándar, para visualizar las coordenadas, ingresar los parámetros de la situación y obtener los resultados.

El espacio en donde se desenvuelve la simulación es un control Picture box de 500x500 píxel de tamaño.

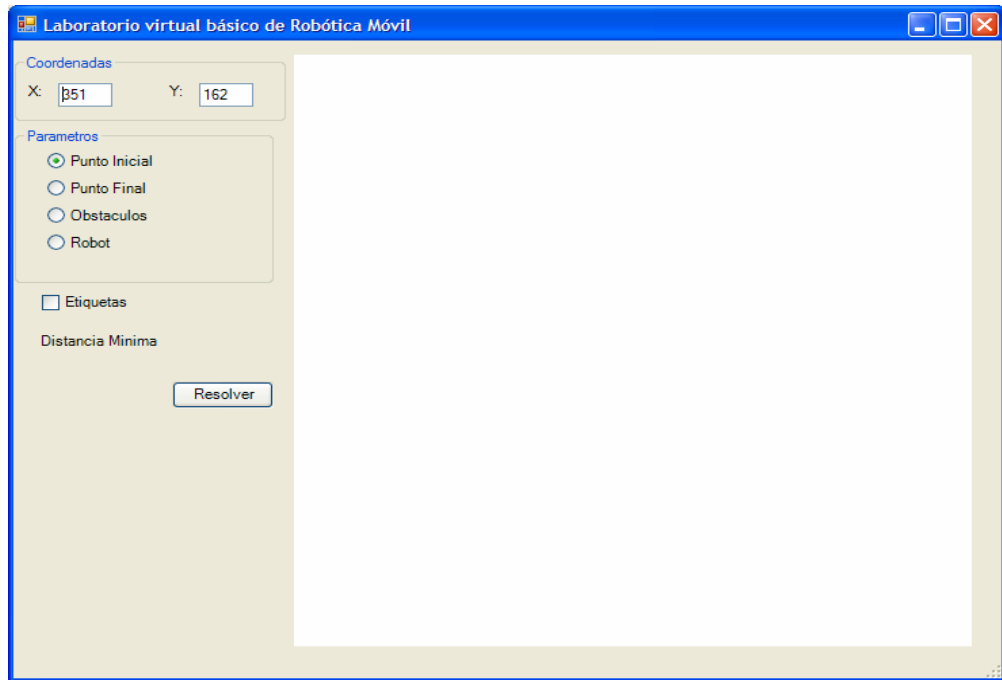


Fig. 3.13 Interfaz del Módulo de simulación – Modo automático

Interfaz del Modo de Práctica Guiada

En la práctica guiada la interfaz es similar a la anterior, salvo que existen controles adicionales para el ingreso de datos, visualización e interacción con el tutor inteligente de tipo asistente personal.

Al Picture box que representa el espacio de simulación se le ha asociado un menú para acceder a las operaciones del método de planificación de trayectorias.

Este tutor inteligente es un carácter animado MsAgent que regula el proceso de la práctica.

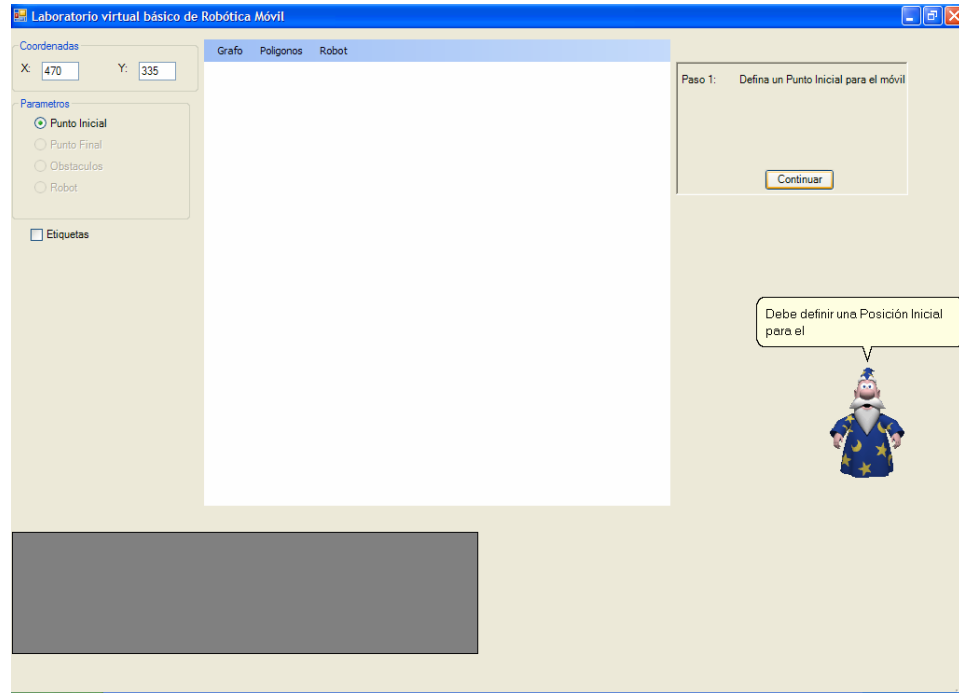


Fig. 3.14 Interfaz del Módulo de simulación – Modo práctica guiada

Módulo de Teleoperación

En la interfaz del módulo de teleoperación se incluyen controles estándar para visualización de coordenadas y operación del robot móvil, una vez más el espacio de navegación esta representado por un control Picture box.

Existe otro control picture box que servirá para la captura de video.

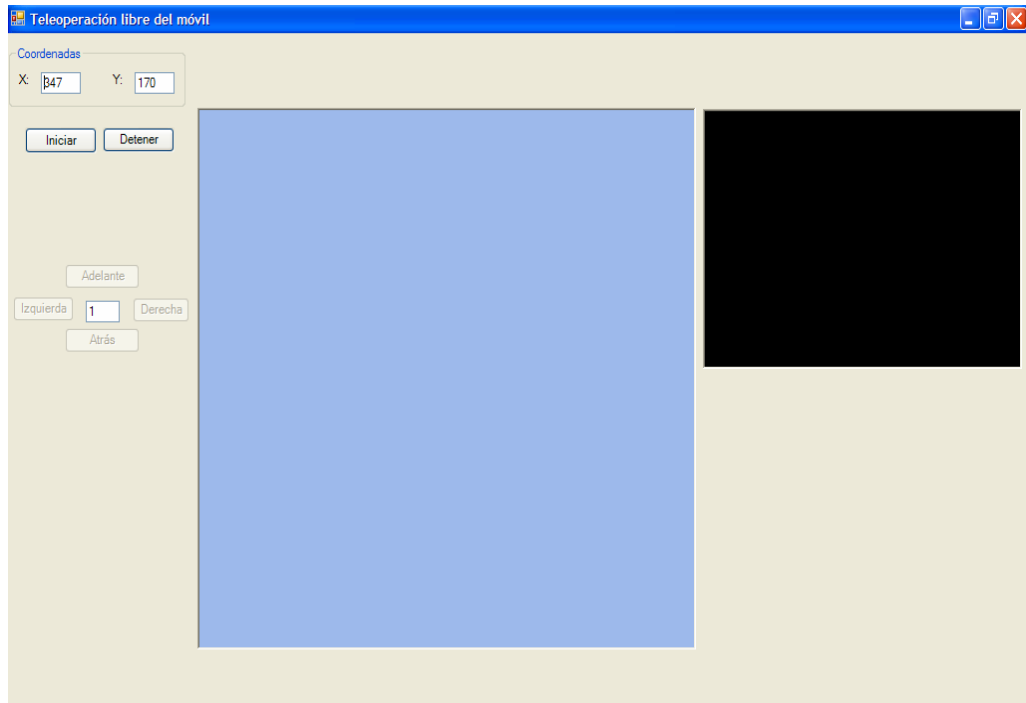


Fig. 3.15 Interfaz módulo de Teleoperación

Módulo de Programación

La interfaz del módulo de Programación contiene controles Windows estándar para ingresar los datos del programa y realizar la compilación y ejecución del mismo.

La ventana tiene un menú asociado donde se encuentran las opciones de Archivo; el área de edición es un control label multilínea.

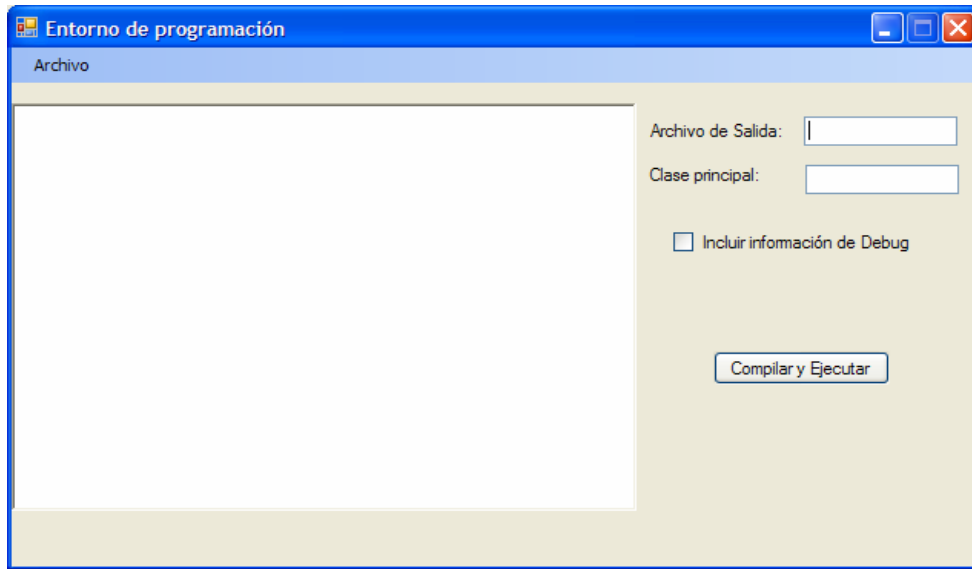


Fig. 3.16 Interfaz módulo de Programación

Arquitectura de la solución

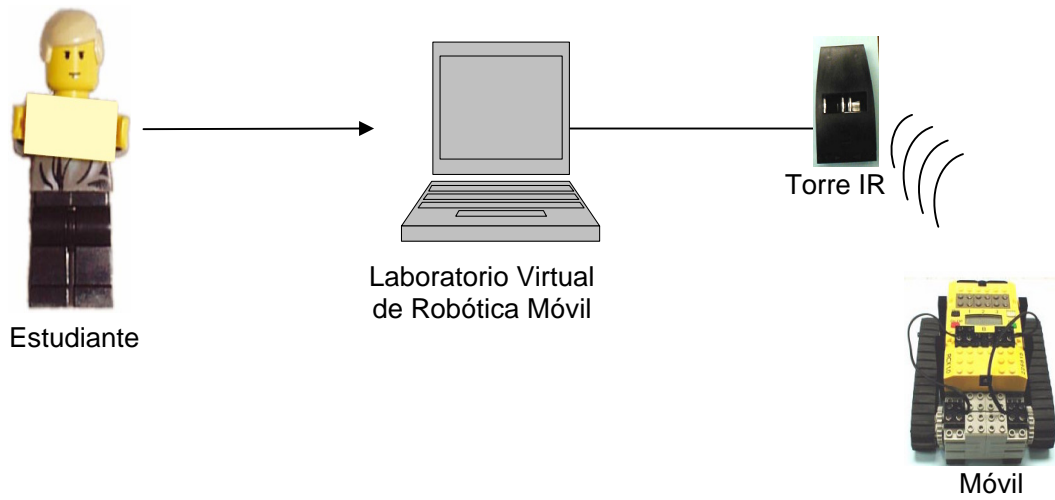


Fig. 3.17 Vista arquitectónica de la implantación de la solución

La figura anterior muestra la vista arquitectónica de la implantación de la solución en la cual el estudiante accede al laboratorio virtual a través del PC, en el caso de la teleoperación del robot éste establece comunicación al móvil a través de la torre infrarroja.

3.2.5 Planeación del Proyecto Aprobada

Al alcanza este hito de finalización de la fase de planeación se han definido las especificaciones funcionales de lo que va a ser entregado en el proyecto.

3.3 FASE DE DESARROLLO

3.3.1 Código fuente

SUBSISTEMA TUTORIAL

Programación de las páginas

Las páginas se codificaron en lenguaje HTML estándar, la interfaz esta formada por un conjunto de marcos y se uso VBScript en algunos casos y en otros JScript para las acciones especiales en botones y enlaces.

Para la estructuración de los contenidos se tomó en cuenta los lineamientos que establece el E-Learning, esto es, se incluye un plan de lección, bienvenida, desarrollo del contenido, una actividad de recreación y autoevaluación.

El mapa del sitio y los contenidos del tutorial de robótica móvil se pueden encontrar en el Anexo B.

Programación del tutor

Para usar Microsoft Agent, la etiqueta OBJECT del Microsoft Agent Control debe ser colocada en la página. El Agent object será referido en el script

usando el nombre asignado al campo ID de la etiqueta OBJECT en este caso "AgentControl".

```
<OBJECT ID="AgentControl" width=0 height=0  
CLASSID="CLSID:D45FD31B-5C6E-11D1-9EC1-00C04FD7081F"  
CODEBASE="#VERSION=2,0,0,0">  
</OBJECT>
```

Para usar la salida Text-to-Speech (TTS), un motor TTS compatible con Microsoft Agent debe estar instalado en la máquina cliente. La licencia de Microsoft Agent incluye una licencia para usar el motor TTS TruVoice de Lernout & Hauspie con Microsoft Agent.

```
<OBJECT ID="TruVoice" width=0 height=0  
CLASSID="CLSID:B8F2846E-CE36-11D0-AC83-00C04FD97575"  
CODEBASE="#VERSION=6,0,0,0">  
</OBJECT>
```

En ambos casos la presencia de la etiqueta OBJECT causará que el control sea automáticamente descargado e instalado si no se encuentra en la máquina cliente cuando la página es procesada. El atributo CODEBASE es usado para especificar la última versión del objeto.

Para la programación de las acciones del MSAgent se utilizó VBScript, los métodos usados más frecuentes son Play, Move y Speak.

SUBSISTEMA SIMULACIÓN

En este apartado se explica el modelo de clases usado para el desarrollo de la aplicación que implementa el cálculo del camino mínimo en un entorno poligonal.

La implementación de este subsistema se basa en funciones de geometría analítica y algoritmos geométricos para el procesamiento del grafo de visibilidad, la ampliación de obstáculos y el cálculo del camino mínimo; éstos pueden ser consultados en el Anexo C.

Este subsistema se compone de las siguientes clases:

- Clase Figura
- Clase Vértice
- Clase Línea
- Clase Polígono
- Clase Robot
- Clase Nodo
- Clase Form1

A continuación se enumeran cada una de las clases haciendo referencia a las propiedades y a los métodos de cada clase, se hace un breve comentario de su significado y su uso.

Clase Figura

Descripción:

Es una clase abstracta que define una figura en el plano de los números enteros positivos.

PROPIEDADES	
Lienzo	Graphics
Color	Color
Colorbg	Color
Matriz	Matrix
Visible	Bool

CONSTRUCTOR	
Figura(lienzo, color, colorbg)	Crea una figura

MÉTODOS	
Dibujar() -> Void	Dibuja la figura. Método abstracto para ser redefinido en clases hijas.
Trasladar(x,y) -> Void	Traslada la figura. Método abstracto para ser redefinido en clases hijas.
Rotar(angle) -> Void	Rota la figura. Método abstracto para ser redefinido en clases hijas.
Mostrar() -> Void	Mostrar una figura en pantalla.
Ocultar() -> Void	Ocultar la figura mostrada.
Mover(x,y) -> Void	Mueve la figura a la ubicación de coordenadas (x,y) y la muestra en pantalla
Girar(angle) -> Void	Gira a figura un ángulo determinado y la muestra en pantalla.

Tabla 3.9 Clase Figura

Clase Vértice

Descripción:

Hereda de la clase Figura, modeliza un punto vértice en el plano de los números enteros positivos.

PROPIEDADES	
Hereda propiedades de la clase Figura	
_posición	PointF

CONSTRUCTOR	
Vértice(lienzo, color, colorbg, x, y):	Crea el vértice de coordenadas (x,y).

MÉTODOS	
Hereda propiedades de la clase Figura	
Dibujar(color) -> Void	Dibuja el vértice con el color dado. Método redefinido.
Trasladar(x,y) -> Void	Traslada el vértice al punto (x,y). Método redefinido.
Rotar(angle) -> Void	Rota el vértice. Método heredado.
ShowLabel()->Void	Muestra etiquetas con los valores de las cordenadas.
HideLabel()-> Void	Ocultas las etiquetas con los valores de las cordenadas.

Tabla 3.10 Clase Vértice

Clase Línea

Descripción:

Hereda de la clase Figura. Modeliza una línea en el plano de los números enteros positivos.

PROPIEDADES	
Hereda propiedades de la clase Figura	
P1	PointF (Punto Inicial)
P2	PointF (Punto final)

CONSTRUCTOR	
Línea(lienzo, color, colorbg, p1,p2):	Crea la línea p1p2

MÉTODOS	
Hereda métodos de la clase Figura	
MiddlePoint() -> PointF	Devuelve el punto medio del segmento.
Dibujar(color) -> Void	Dibuja la línea con el color dado. Método redefinido.
Trasladar(x,y) -> Void	Traslada la línea al punto (x,y). Método heredado.
Rotar(angle) -> Void	Rota la línea. Método heredado.
Pendiente()->Float	Devuelve el valor de la pendiente de la línea
Angulo()-> Double	Devuelve el valor del ángulo de inclinación de la pendiente en grados.
H()-> Float	Devuelve el punto de intersección con el eje coordenado Y.
Distance(PointF p1, PointF p2)->Float	Devuelve la distancia entre dos puntos
IntersectionPoint(Línea l1) -> PointF	Devuelve el punto de intersección entre l y l1
Contains(nPoint p) -> Bool	Devuelve TRUE si el punto pertenece a la línea.
Intersects(Línea l1) -> Bool	Devuelve TRUE si se intersecan las dos líneas

Tabla 3.11 Clase Línea

Clase Polígono

Descripción:

Hereda de la clase Figura. Modeliza un polígono en el plano de los números enteros positivos.

PROPIEDADES	
Hereda propiedades de la clase Figura	
_relleno	Bool
vertices	ArrayList lista de vértices (FPoint)
Convex	si es convexo o no, Bool

CONSTRUCTOR	
Polígono()	Crea un polígono con o puntos, convexo y de dirección positiva

MÉTODOS	
Hereda métodos de la clase Figura	
AgregarVértice(Vértice p) -> Bool	Añade el punto p al polígono
Dibujar(brush color) -> Void	Dibuja el polígono con el color dado. Método redefinido.
Trasladar(x,y) -> Void	Traslada cada vértice del polígono en referencia al punto (x,y). Método redefinido
Rotar(angle) -> Void	Rota el polígono. Método heredado.
IsInside(nPoint p) -> Bool	Comprueba si el punto está contenido dentro del contorno del polígono
Intersects(Línea L) -> Bool	Devuelve TRUE si se intersecan la línea dda con el contorno del polígono
GetAmpliado(Polígono robot) -> Polígono	Aplica las sumas de Minkowski al polígono y lo amplía de acuerdo al polígono del robot
ShowLabels() -> Void	Muestra etiquetas con los valores de las coordenadas.
HideLabels() ->	Ocultas las etiquetas con los valores de las coordenadas.

Tabla 3.12 Clase Polígono

Clase Robot

Descripción:

Hereda de la clase Figura. Modeliza un robot móvil en el plano de los números enteros positivos.

PROPIEDADES	
Hereda propiedades de la clase Figura	
Posición	PointF
Orientación	Float
Desplazamiento	Double
Robot	GraphicsPath
RobotAux	GraphicsPath
Reloj	Timer
Camino	ArrayList
NextPoint	IEnumerator
CONSTRUCTOR	
Robot(lienzo, color, colorbg, p1,p2):	Crea el robot
MÉTODOS	
Hereda métodos de la clase Figura	
Dibujar(color) -> Void	Dibuja el robot móvil con el color dado. Método redefinido.
Trasladar(x,y) -> Void	Traslada el robot al punto (x,y). Método redefinido.
Rotar(angle) -> Void	Rota el robot el ángulo dado. Método redefinido.
Recorrer()->Void	Hace que el robot recorra el camino mínimo
Reloj_Tick()-> Void	Crea la animación del desplazamiento del robot.
Angulo()-> Float	Devuelve el ángulo de giro del robot en los desplazamientos.
Distance(PointF p1, PointF p2)->Float	Devuelve la distancia entre dos puntos
IntersectionPoint(Línea l1) -> PointF	Devuelve el punto de intersección entre l y l1
Contains(nPoint p) -> Bool	Devuelve TRUE si el punto pertenece a la línea.
Intersects(Línea l1) -> Bool	Devuelve TRUE si se intersecan las dos líneas

Tabla 3.13 Clase Robot

Clase Nodo

Descripción:

Es una clase que define un grafo en el plano de los números enteros positivos y permite calcular el camino mínimo dentro del mismo.

PROPIEDADES	
_posición	PointF
Distanciafin	Double
Nodos	ArrayList
Min	Nodo

CONSTRUCTOR	
Nodo(PointF P):	Crea un grafo

MÉTODOS	
Distancetofin() -> Double	Calcula la distancia mínima entre el nodo origen al nodo destino.
AddNodo(Nodo N) -> Void	Añade un nodo a la lista de nodos del camino mínimo.
CrearCamino(Graphics lienzo, ArrayList camino) -> Void	Método recursivo que dibuja los caminos o aristas del grafo.
Dibujar() -> Void	Resalta el camino mínimo encontrado en el grafo.

Tabla 3.14 Clase Nodo

Clase Form1

Descripción:

Es una clase que define las acciones de la interfaz y contiene todos los objetos (instancias de las demás clases) que componen el problema y realiza el procedimiento para hallar la solución.

PROPIEDADES	
Lienzo	Graphics
robot	Robot
Obstáculo	Polígono
Robot	Polígono
Obstáculos	ArrayList
Ampliados	ArrayList
PuntoInicial	Vértice
PuntoFinal	Vértice
Inicio	Nodo
Final	Nodo
Nodos	Nodo
Distancia	Double
Paso	Byte
Marcado	Bool
mouseX	Int
mouseY	Int
Agent	Agente
guiada	Bool

CONSTRUCTOR	
Form1(bool guiada):	Crea el entorno de simulación

MÉTODOS	
Panel1_MouseMove() -> Void	Muestra las coordenadas (x,y) que el Mouse señale dentro del espacio de navegación del móvil.
Cb_labels_CheckedChanged() -> Void	Muestra u oculta las etiquetas con información
dibujarToolStripMenuItem_Click() -> Void	Realice las operaciones para dibujar el grafo de visibilidad.

Panel1_MouseUp() -> Void	Indica el fin de ingreso de un polígono
Panel1_MouseClick() -> Void	Rellena los obstáculos poligonales ingresados con un color definido
Form1_Paint() -> Void	Repinta los elementos dibujados
ampliarToolStripMenuItem_Click () -> Void	Amplia los obstáculos según el algoritmo de Minkowski.
NuevoPaso(byte paso)-> Void	Define los pasos que se seguirán en la práctica guiada.
btnSiguiente_Click() -> Void	Valida el cumplimiento de cada paso de la práctica guiada y permite avanzar al siguiente paso.
calcularCaminoMinimoToolStripMenuItem_Click() -> Void	Calcula el camino mínimo que debe recorrer el móvil.
mnRecorrerCamino_Click() -> Void	Hace que el móvil recorra el camino mínimo calculado.
btnResolver_Click() -> Void	Realiza la solución del problema de forma automática.

Tabla 3.15 Clase Form1

Tutor inteligente

Una aplicación .NET no se comunica directamente con un componente COM, en vez de eso, funciona mediante un envoltorio gestionado del componente, mejor conocido como RCW (Runtime Callable Wrapper) que actúa como un Proxy gestionado para el componente COM no gestionado. Permite la manipulación de los métodos del componente y devolver cualquier valor de retorno en una forma que pueda ser interpretado por el CLR (Common Language Runtime). Para utilizar el MSAgent se debe indicar que referencia COM se va a utilizar en el proyecto, en este caso Microsoft Agent Control 2.0. Una vez el componente es parte del proyecto se puede tratarlo como si fuera una clase más del .NET Framework permitiendo instanciar objetos del mismo.

Clase Agent

Descripción:

Es una clase que define el tutor inteligente que interactuará con el usuario.

PROPIEDADES	
AgetSrv	AgentServer
SrvEx	IAgentEx
CharacterEx	IAgentcharacterEx
dwCharID	Int

CONSTRUCTOR	
Agente_Load()	Carga y muestra al agente

MÉTODOS	
Agente_Hablar (string texto) -> Void	Hace que el agente pronuncie el texto dado.
Agente_Animar (string animación) -> Void	Hace que el agente realice la animación pasada como parámetro.
Agente_Detener () -> Void	Detiene la ejecución del agente.
Agente_Pensar (string texto) -> Void	Hace que el agente visualice el texto dado como un pensamiento.
Agente_Gesto (string gesto) -> Void	Hace que el agente realice una animación con el gesto dado.
Agente_Descargar () -> Void	Descarga el agente de la aplicación

Tabla 3.16 Clase Agent

SUBSISTEMA PROGRAMACIÓN

Este subsistema utiliza la siguiente clase para implementar un pequeño compilador de C# que permite descargar código al robot móvil.

Clase Form1

Descripción:

Es una clase que define el entorno de programación.

PROPIEDADES	
CodeProvider	CSharpCodeProvider
Compiler	ICodeCompiler
Parameters	CompilerParameters

CONSTRUCTOR	
Form1()	Inicializa los componentes.

MÉTODOS	
Button1_click() -> Void	Realiza las operaciones para la compilación y ejecución del código.
guardarToolStripMenuItem_Click() -> Void	Abre el diálogo para guardar el código en un archivo.
salirToolStripMenuItem_Click() -> Void	Cierra la aplicación
toolStripMenuItem1_Click () -> Void	Abre el diálogo para cargar un archivo de código.

SUBSISTEMA TELEOPERACIÓN

El subsistema de teleoperación se implementó como una aplicación Windows forms, permite la teleoperación del robot móvil.

Este subsistema hace referencia a la librería RCXNET cuyos métodos son necesarios para la adquisición de datos y envío de señales al robot móvil LEGO a través de la torre infrarroja.

El subsistema de teleoperación se compone de las siguientes clases:

- Clase Móvil
- Clase Video
- Clase Form1

A continuación se presenta una descripción de cada una de ellas.

Clase Móvil

Descripción:

Modeliza una robot móvil en el plano de los números enteros positivos.

PROPIEDADES	
Lienzo	Graphics
bgColor	Brush
MovilPos	Point
Robot	GraphicsPath
RobotAux	GraphicsPath
TranslateMatrix	Matrix

CONSTRUCTOR	
Móvil(lienzo, bgcolor, x, y)	Crea el robot móvil.

MÉTODOS	
Dibujar() -> Void	Dibuja el robot móvil en el espacio de navegación.
Ocultar() -> Void	Oculto el móvil dibujado en el espacio de navegación.
Mostrar() -> Void	Muestra el móvil en la pantalla.
Mover(int x, int y) -> Void	Mueve el móvil a las coordenadas (x,y)
Girar(single angle) -> Void	Hace que el móvil gire el ángulo dado

Tabla 3.18 Clase Móvil

Clase Video

Descripción:

Es una clase que define el entorno de programación.

PROPIEDADES	
Hwndc	Int
Ws_child	Int
Ws_visible	Int
Wm_user	Short
Wm_cap_driver_conect	Int
Wm_cap_driver_disconnect	Int
Wm_cap_set_preview	Int
Wm_cap_set_previerrate	Int
capCreateCaptureWindowA	Public extern static int
SendMessage	Public extern static int

CONSTRUCTOR	
Video()	Inicializa los componentes.

MÉTODOS	
Ventana(int handle) -> Bool	Crea la ventana de captura del video a través de un manejador.
Conectar() -> Bool	Conecta el driver para empezar la captura
Capturar -> Bool	Captura el video en la ventana creada para el efecto.

Tabla 3.19 Clase Video

Clase Form1

Descripción:

Es una clase que define el entorno de programación.

PROPIEDADES	
WaitTimeTurn	Int
Mv	Móvil
Dirección	Int16
MóvilDib	Int16
objSpirit	SpiritNet

CONSTRUCTOR	
Form1()	Inicializa los componentes.

MÉTODOS	
Form1_Load() -> Void	Carga el video
MoverRobot() -> Void	Mueve el robot móvil tanto en la pantalla como el real equivalente.
btnIniciar_Click -> Void	Establece comunicación con el robot móvil real.
btnDetener_Click () -> Void	Cierra la comunicación con el dispositivo
Panel1_Paint() -> Void	Muestra el robot móvil en pantalla
Panel1_MouseDown() -> Void	Sitúa el robot en el espacio de navegación
Panel1_MouseMove() -> Void	Muestra las coordenadas cartesianas
Arriba_MouseDown() -> Void	Habilita el reloj para la dirección hacia arriba

Abajo_MouseDown() -> Void	Habilita el reloj para la dirección hacia abajo
Izquierda_MouseDown() -> Void	Habilita el reloj para la dirección hacia la izquierda
Derecha_MouseDown() -> Void	Habilita el reloj para la dirección hacia la derecha
Arriba_MouseUp() -> Void	Deshabilita el reloj para la dirección hacia arriba
Abajo_MouseUp() -> Void	Deshabilita el reloj para la dirección hacia abajo
Izquierda_MouseUp() -> Void	Deshabilita el reloj para la dirección hacia la izquierda
Derecha_MouseUp() -> Void	Deshabilita el reloj para la dirección hacia la derecha
Reloj_Tick() -> Void	Realiza la animación del movimiento del robot móvil

Tabla 3.20 Clase Form1

3.3.2 Parámetros de configuración

Para el desarrollo de la solución los siguientes elementos deben estar preinstalados en la máquina host:

- Microsoft Agent API
 - Components Microsoft Agent Core
 - Microsoft Agent Characters: Merlin.
 - Microsoft Speech API 4.0a runtime.
 - Microsoft Speech Recognition Engine.

- Motor Text-to-Speech Lernout and Hauspie para español.
- Software del kit Lego Mindstorms
- Lego Mindstorms SDK 2.5
- Net Framework 1.1 o superior

3.3.3 Pruebas de inspección de código

Se realizó diferentes pruebas de caja blanca y caja negra para determinar si los datos que se esperan obtener cuando se ejecuta algún proceso son los correctos, una para cada subsistema, a continuación se muestran las pruebas y sus resultados.

Prueba	Navegación completa
Ámbito	Subsistema Tutorial
Procedimiento	<ol style="list-style-type: none"> 1. Realizar una navegación por todas las páginas que componen el tutorial. 2. Comprobar que los vínculos no tengan conflictos
Resultados	<p>No se encontraron inconvenientes en la navegación.</p> <p>En cuanto a los vínculos se encontró algunos enlaces rotos y otros que tenían destinos incorrectos, pero el problema fue resuelto fácilmente.</p>

Tabla 3. 21 Prueba de navegación completa

Prueba	Cálculos y Algoritmos
Ámbito	Subsistema Simulación
Procedimiento	<ol style="list-style-type: none"> 1. Analizar si las fórmulas matemáticas que intervienen en el proceso de simulación entregan datos correctos. 2. Comprobar que los algoritmos se ejecuten de forma adecuada. 3. Comprobar que no existan bucles infinitos
Resultados	No existen problemas con los resultados de las fórmulas matemáticas, los algoritmos siguen la secuencia adecuada y no se encontró bucles infinitos, sin embargo se debió validar los casos en que el resultado de éstas fórmulas en un dato no numérico (infinito).

Tabla 3. 22 Prueba de cálculos y algoritmos

Prueba	Validación de datos
Ámbito	Subsistema Programación
Procedimiento	<ol style="list-style-type: none"> 1. Verificar que el subsistema valida correctamente los datos que se deben ingresar través de la interfaz. 2. Comprobar que la compilación de sentencia se realiza adecuadamente. 3. Revisar que el código se descargue correctamente al robot.
Resultados	En este subsistema no se encontró ningún inconveniente.

Tabla 3. 23 Prueba de Validación de datos

Prueba	Comunicación con el dispositivo robótico
Ámbito	Subsistema Teleoperación
Procedimiento	<ol style="list-style-type: none"> 1. Verificar que existe comunicación con el robot. 2. Probar los métodos de la animación en pantalla. 3. Comprobar la adquisición de video.
Resultados	En este subsistema se comprobó su plena funcionalidad, salvo que por las limitaciones del dispositivo la comunicación se corta cuando está fuera del rango de alcance de la torre IR y tanto la animación como el dispositivo se quedan en un ciclo infinito, para solucionar este inconveniente se debió ingresar sentencias para control de errores con lo que se corrigió plenamente el problema.

Tabla 3. 24 Prueba de comunicación con el dispositivo robótico

3.3.4 Reporte de Bugs

En la solución desarrollada se han encontrado los siguientes bugs:

- En la simulación, cuando el móvil realiza el recorrido del camino mínimo se borran las partes del gráfico, cuando esta termina se repinta nuevamente.
- Existe asincronismo entre la simulación, la ejecución del robot y la visualización de la webcam.

Debido a que estos bugs no representan ningún inconveniente en la operación misma de la solución se declara que la solución ha alcanzado el hito de ámbito completo. Para solucionar los mismos se puede plantear una siguiente versión del proyecto.

3.3.5 Alcance completo

La fase de desarrollo culmina con este hito, en este punto todas las características de la solución están completas. La solución esta lista para pruebas externas de estabilización.

3.4 FASE DE ESTABILIZACION

En esta etapa se realizó el seguimiento del sistema por un periodo de tiempo determinado en el cual se comprobó que la solución se ejecuta satisfactoriamente en el ambiente de trabajo real.

3.4.1 Especificación de Tests

Las pruebas realizadas en la fase anterior fueron suficientes para evidenciar la total funcionalidad del sistema solo en el caso del subsistema de simulación por ser el módulo más complejo de la solución de planteo un caso de prueba, con los datos que se describe a continuación:

Entradas

Punto inicial: (60,80)

Punto Final: (420,420)

Obstáculos:

V1 (60,200)

V2 (300,240)

V3 (180,380)

Salidas Esperadas

Nodos del camino mínimo: (60,80); (325,215); (420,420)

Coste del camino mínimo: 523.35

Procedimientos especiales

Ingreso de los vértices de los obstáculos en el sentido de las manecillas del reloj, por requerimiento de ordenación del algoritmo de sumas de Minkoski.

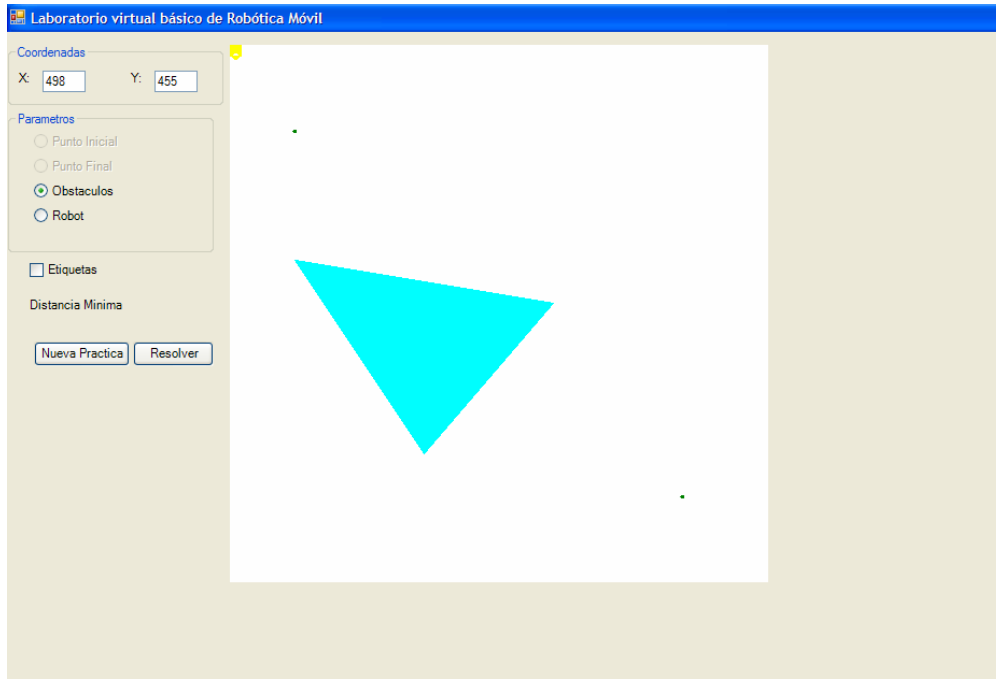


Fig. 3.18 Datos de prueba

Proceso

Una vez ingresados el punto inicial, el punto final y el obstáculo, el subsistema procede a la ampliación del mismo, para ello usa el algoritmo de sumas de Minkowski que permite sumar dos polígonos convexos en este caso el polígono P que representa al obstáculo y el polígono Q que representa al robot.

El polígono Q se considera con centro en el origen de coordenadas (ubicado en el extremo superior izquierdo) y con el valor de sus vértices fijos, este polígono no se visualiza en la interfaz, es definido únicamente para los cálculos. Entonces dados:

	P			Q	
V	X	Y	W	X	Y
1	60	200	1	-25	-25
2	300	240	2	25	-25
3	180	380	3	25	25
4	60	200	4	-25	25
			5	-25	-25

Siguiendo el algoritmo de las sumas de Minkowski se obtuvo los siguientes resultados en las diferentes iteraciones, que corresponden a los vértices del obstáculo ampliado.

	P	\oplus	Q
X			Y
35			175
85			175
325			215
325			265
205			405
155			405
35			225

A continuación el subsistema aplica el algoritmo de Grafo de visibilidad al obstáculo ampliado, encuentra el camino mínimo y calcula el coste del camino mínimo según la fórmula de distancia euclídea. El gráfico siguiente muestra el resultado final del proceso.

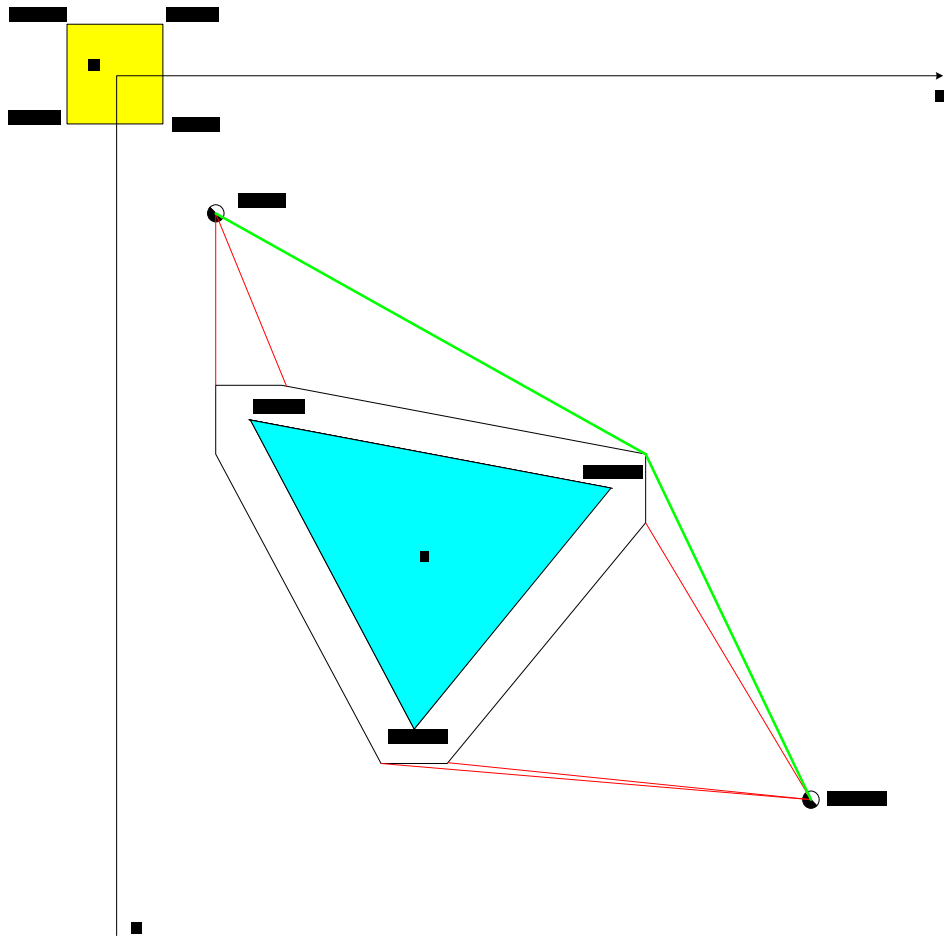


Fig. 3.19 Gráfico del proceso de planificación de trayectorias

Resultados

El resultado del caso de estudio planteado fue el esperado, con lo que se verificó que el sistema arroja resultados reales y estables.

En la siguiente figura se puede comprobar que la salida calculada por el sistema corresponde al proceso seguido para obtener la solución.

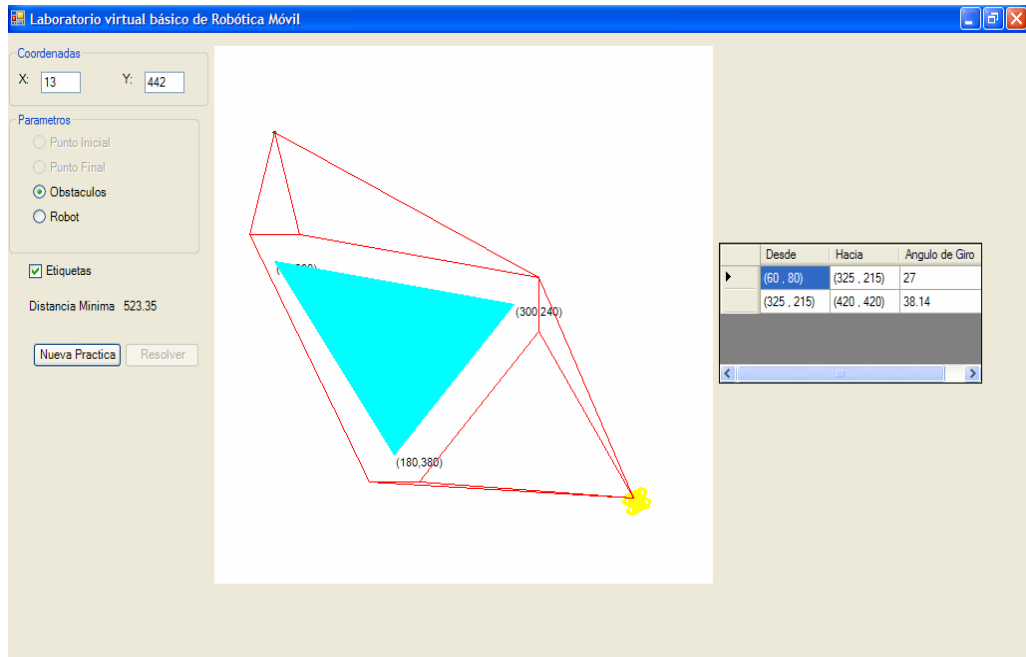


Fig. 3.20 Salida esperada

3.4.2 Consideraciones y restricciones

El sistema se encuentra libre de bugs activos y se encuentra listo para la fase de implantación. En el subsistema de Simulación dada su complejidad se deben tomar en cuenta las siguientes restricciones y consideraciones:

Restricciones:

- El simulador acepta obstáculos convexos únicamente.
- El simulador no contempla solapamientos de obstáculos ni de obstáculos ampliados.

Consideraciones de operación:

- Los vértices de los obstáculos se deben ingresar en el sentido de las manecillas del reloj por requerimientos del algoritmo de Minkowski.
- El primer vértice ingresado debe ser aquel con la coordenada “y” de menor valor.

3.4.3 Código fuente y ejecutables

El código fuente y los ejecutables fueron compilados en un CD que representa el producto final.

2.4.4 Release aprobado

La fase de estabilización culmina en este hito “listo para liberación”. Una vez revisada y depurada la solución está lista para la implantación en un ambiente de producción.

3.5 FASE DE IMPLANTACIÓN

3.5.1 Información de soporte y operación

En esta sección se enumeran los requisitos necesarios para ejecutar la aplicación. Estos requisitos se dividen en requisitos de hardware, requisitos de software y la instalación propia del programa.

Requisitos hardware

Esta aplicación necesita los siguientes requisitos mínimos hardware para ser ejecutada:

- Procesador Pentium II
- 32 Mbytes de memoria RAM
- Configuración gráfica de 16 colores
- RCX 1.1 con torre IR de conexión usb

La configuración óptima para obtener el máximo rendimiento de la aplicación sería:

- Procesador Pentium III 400 Mhz
- 128 Mbytes de RAM
- Configuración Gráfica de 32 bits
- RCX 2.0 con torre IR de conexión usb

Requisitos software

En cuanto a los requisitos software mínimos necesitaremos un entorno con la siguiente configuración:

- Windows 98
- Internet Explorer 5
- Lego Minstorms SDK 2.0
- Microsoft Agent API

Si se quiere obtener un máximo partido se debería tener la siguiente configuración:

- Windows XP
- Internet Explorer 6
- Lego Mindstorms SDK 2.5

3.5.2 Manual de usuario

Esta sección hace referencia al manejo de la aplicación, en el sentido de interfaz de usuario, introducción y resolución de un problema. El documento Manual de Usuario como tal se puede apreciar en el Anexo D.

3.5.3 Siguietes pasos

Se puede pensar en una próxima versión de la solución en donde cubra contenido teórico más profundo en cuanto al tutorial, en la simulación incluir otros métodos de planificación y además la simulación de la cinemática del robot móvil y mejorar las prestaciones de la teleoperación y programación.

En cuanto al hardware se refiere se podría mejorar las prestaciones del robot móvil utilizando una configuración diferencial y sensores. La comunicación inalámbrica se puede optimizar usando un estándar 802.11

3.5.4 Deployment completo

Se ha alcanzado el hito de Deployment Completo con el cual culmina la fase de Implantación, esto quiere decir que la solución ha alcanzado su meta y ha terminado efectivamente todos los procesos y actividades.

CAPITULO VI

CONCLUSIONES, RECOMENDACIONES Y TRABAJS FUTUROS

6.1 Conclusiones

- La experiencia acumulada con el desarrollo de este proyecto permite afirmar que los laboratorios virtuales son una técnica bastante útil en los procesos de enseñanza, ya que facilita el aprendizaje de los contenidos de las materias al permitirle a los estudiantes estudiar en el momento y lugar que crean conveniente, es decir, sin estar obligados a trasladarse a un centro de estudios ni depender de un profesor en un sentido no estricto.
- La inclusión en el terreno pedagógico de un asistente personal en un laboratorio virtual posibilita la supervisión de las acciones del usuario en un entorno informático para proporcionar ayuda. Sin embargo, aunque se ha visto que estos Sistemas Tutores Inteligentes son cada vez más comunes y eficientes, todavía resultan difíciles y caros de construir lo que impide su uso generalizado.

- La Metodología MSF se adapta a proyectos de cualquier dimensión y de cualquier tecnología.

- El modelo de procesos de la metodología *Microsoft Solution Framework* empleado en esta investigación, por utilizar un enfoque iterativo, ofreció la flexibilidad necesaria para manejar exitosamente los cambios de requerimientos que surgieron a lo largo de todo el período del desarrollo. Sin embargo, por ser una metodología tan abierta, fue necesaria la definición de artefactos específicos que permitiesen precisar las necesidades del usuario y las funcionalidades de los componentes del Sistema.

- El entorno de aprendizaje que se propone en esta tesis ofrece la posibilidad de aprender programación y robótica mediante la experimentación a través de un entorno virtual. Entre las ventajas encontradas se pueden destacar los siguientes aspectos:
 - Ventajas del e-learning: sencillez, acceso a información multimedia.
 - Familiarización con la robótica.
 - Familiarización con el lenguaje de programación C#.

- En este trabajo se ha presentado una versión preliminar de una herramienta interactiva que tiene como fin servir de apoyo a los alumnos a la hora de estudiar algoritmos de navegación autónoma. El alumno puede modificar la configuración de un entorno conocido así y estudiar la influencia sobre los algoritmos de control.

- Se aplicaron algoritmos de geometría computacional para resolver el problema de la búsqueda del camino mínimo en grafos y en si en el proceso de simulación.

6.2 Recomendaciones

- Este proyecto se vería beneficiado si se realizará la implementación del tutor de forma netamente “inteligente”, es decir, a través de un ITS (Intelligent Tutoring System) que costa de un sistema experto y otras técnicas de inteligencia artificial.
- Para coincidir en tiempo real el escenario simulado con el escenario real se recomienda desarrollar un módulo de cálculos matemáticos, físicos y trigonométricos, implementado con programación basada en threads.
- Desarrollar un robot móvil con una arquitectura de diferencial y usando sensores para facilitar la localización y posicionamiento.
- Implementar todos los casos para los algoritmos de computación gráfica, contemplando solapamiento de polígonos y sumas de Minkowski de polígonos no convexos.
- Usar algoritmos más óptimos para reducir el tiempo de procesamiento.

- El tiempo necesario para bajar un programa al robot a través de la torre de IR es considerable ($\cong 30$ segundos). La torre de IR y el robot (el RCX) deben estar encarados a 25cm máximo de distancia para permitir una correcta comunicación. Una posible solución sería diseñar una comunicación inalámbrica 802.11.
- Es recomendable que el Departamento de Ciencias de la Computación cree una materia de Geometría Computacional para la enseñanza de algoritmos que resuelven problemas geométricos.

6.3 Trabajos futuros

- Lo que se desea llegar a conseguir con esta herramienta en sus fases posteriores es que el alumno pueda estudiar distintos entornos y algoritmos de navegación, trabajar con varias configuraciones de robots móviles y observar de forma interactiva como la modificación de parámetros del entorno, de la cinemática del robot, del algoritmo de planificación de trayectorias y del algoritmo de seguimiento de caminos incluye en el problema de la navegación autónoma con entorno conocido.

ANEXOS

ANEXO A - DIAGRAMAS UML

Diagramas de Casos de Uso

Navegación por contenidos

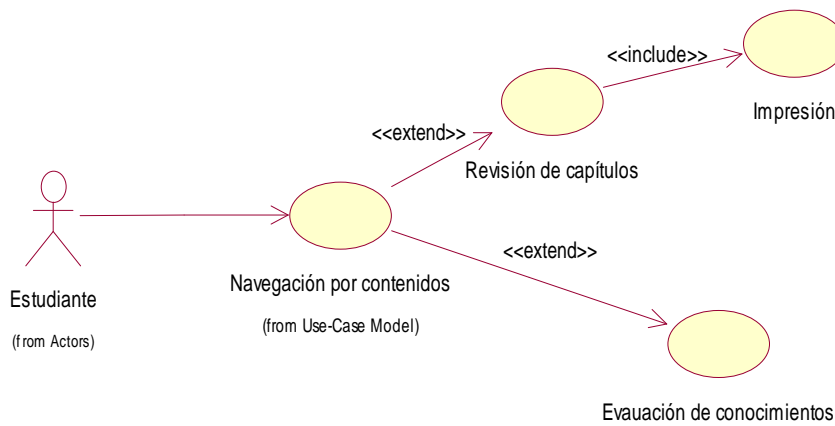


Fig. A.1 Diagrama caso de uso Navegación por contenidos

Simulación

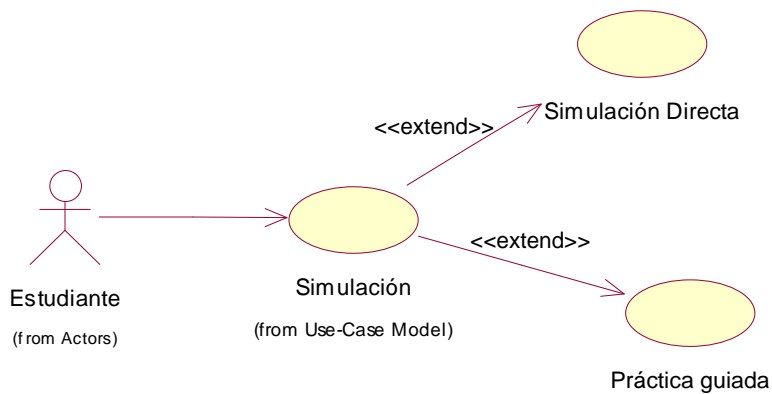


Fig. A.2 Diagrama caso de uso Simulación

Teleoperación

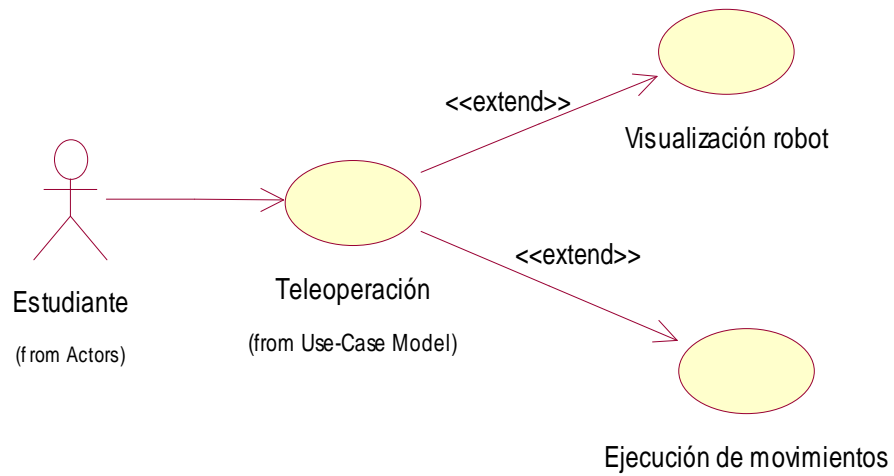


Fig. A.3 Diagrama caso de uso Teleoperación

Programación

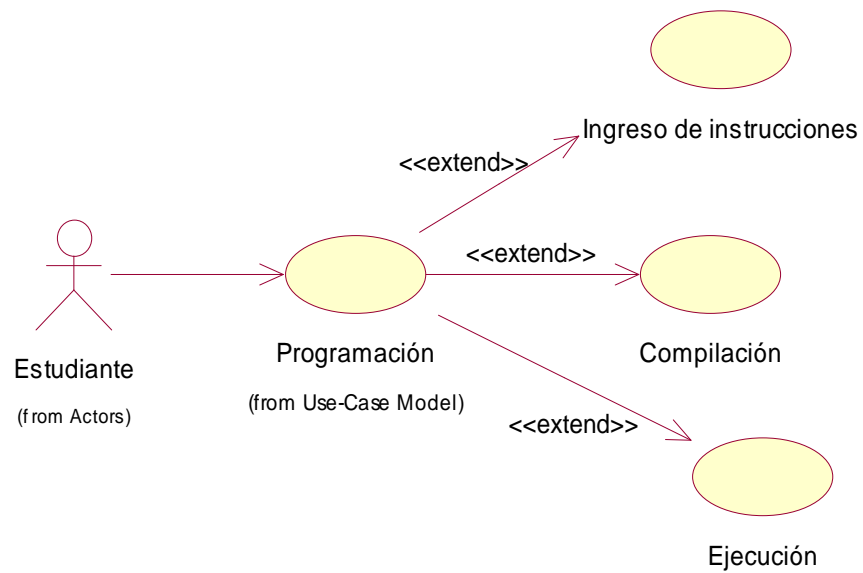


Fig. A.4 Diagrama caso de uso Programación

Diagramas de Secuencia

Diagrama de secuencia del sistema

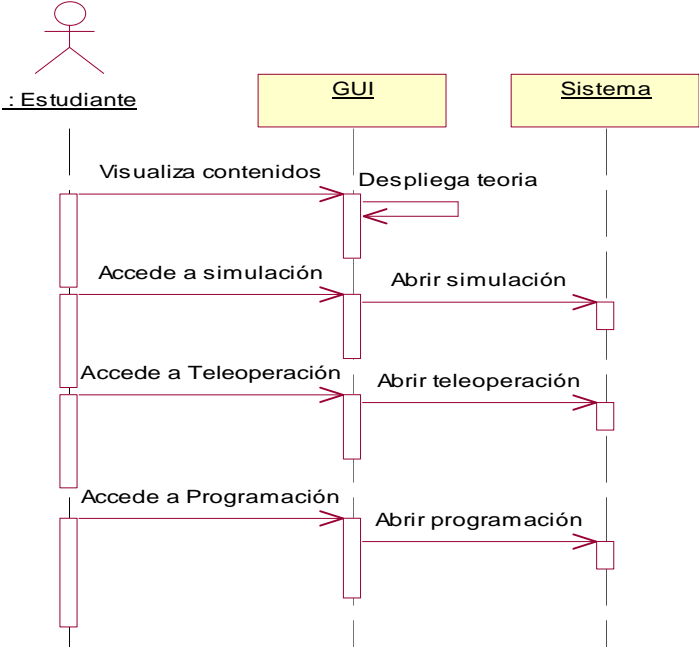


Fig. A.5 Diagrama de secuencia del Sistema

Navegación por contenidos

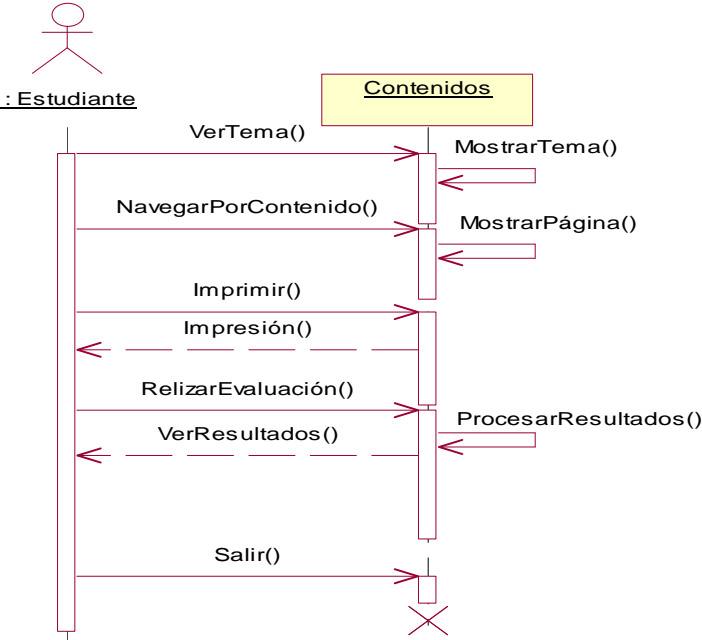


Fig. A.6 Diagrama de secuencia Navegación por contenidos

Simulación

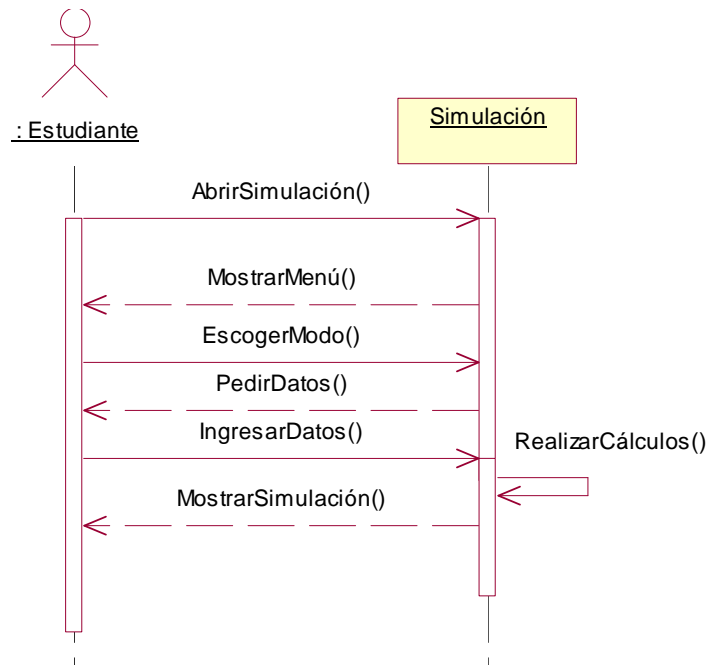


Fig. A.7 Diagrama de secuencia Simulación

Teleoperación

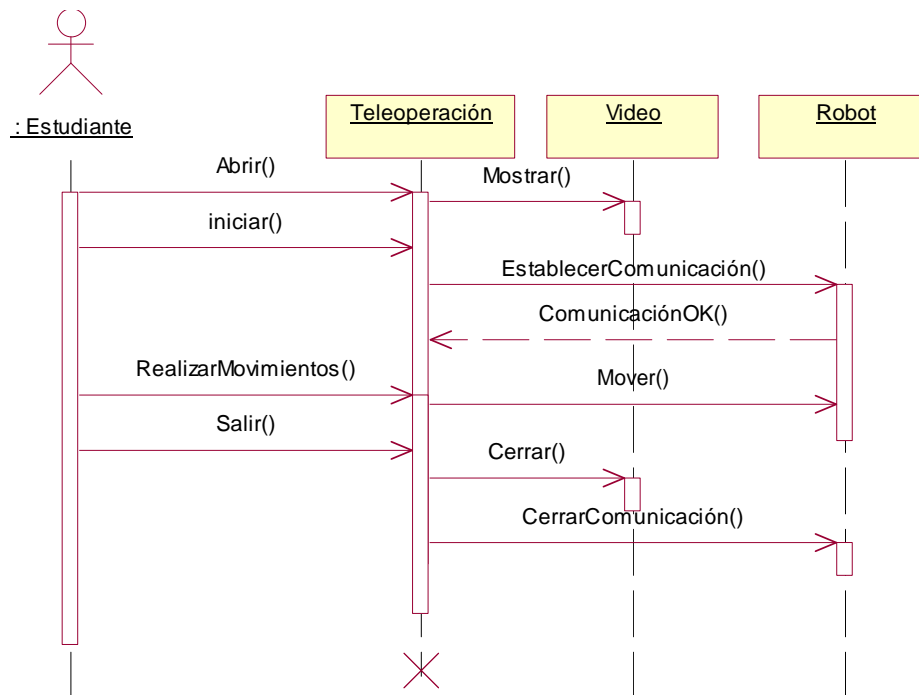


Fig. A.8 Diagrama de secuencia Teleoperación

Programación

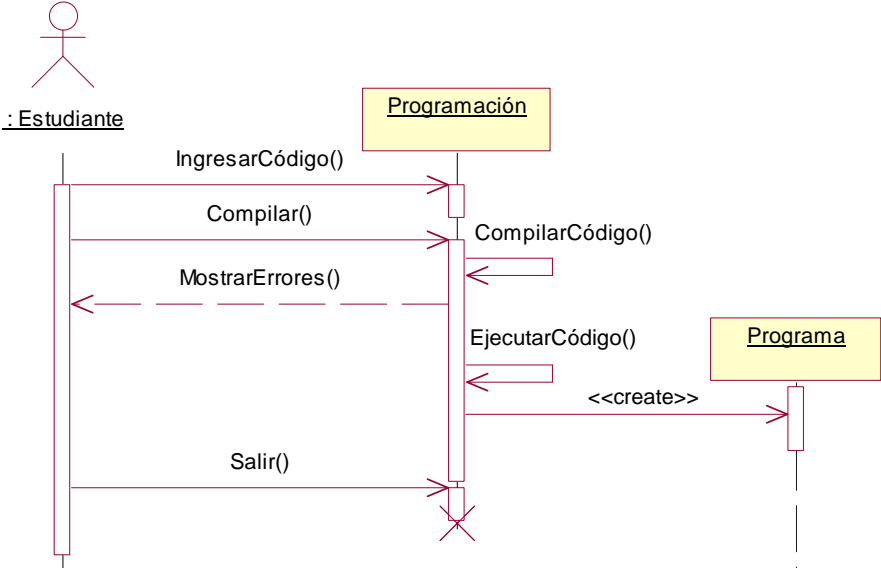


Fig. A.9 Diagrama de secuencia Programación

Diagramas de Clases

Subsistema Contenidos

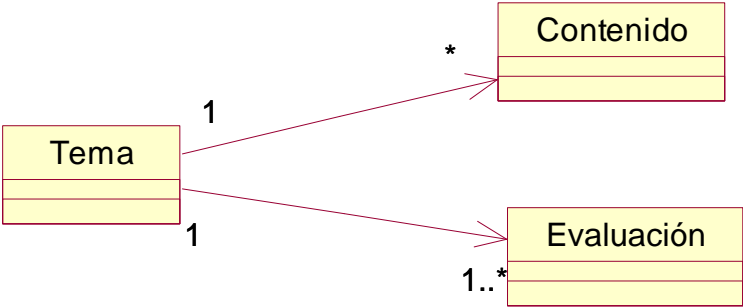


Fig. A.10 Diagrama de Clases Contenidos

Simulación

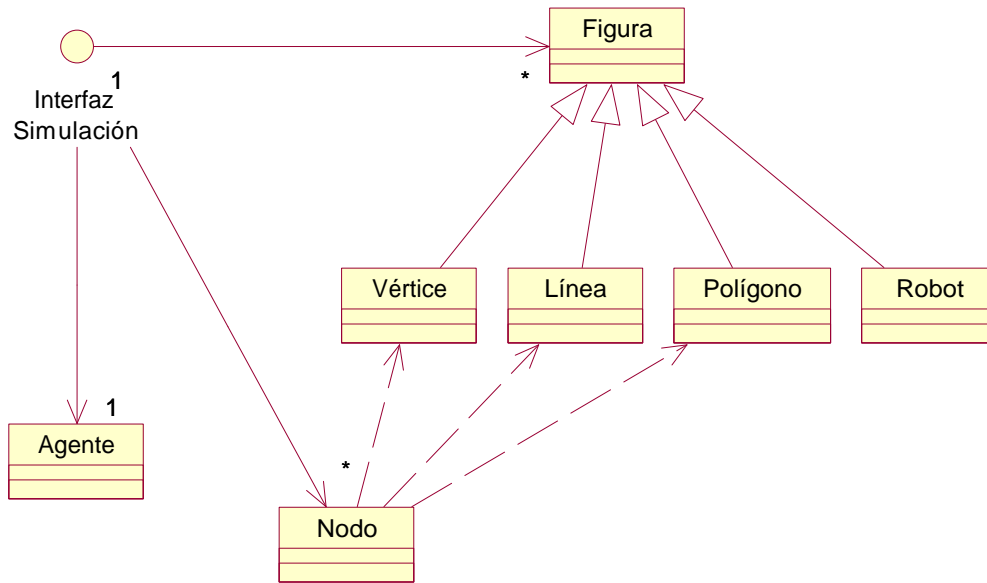


Fig. A.11 Diagrama de Clases Simulación

Teleoperación

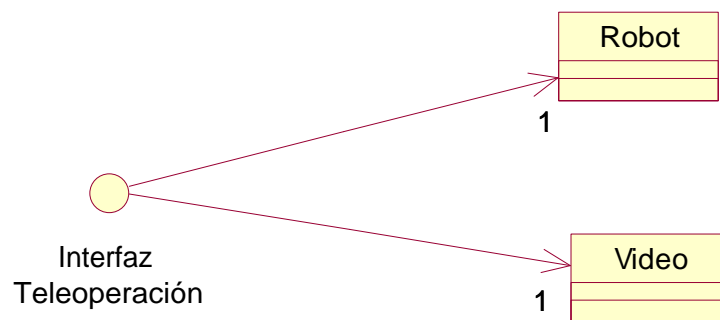


Fig. A.12 Diagrama de Clases Teleoperación

Programación

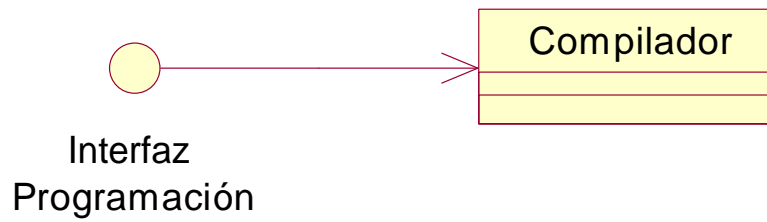


Fig. A.13 Diagrama de clases Programación

Diagramas de Actividad

Diagrama de Actividades del Sistema

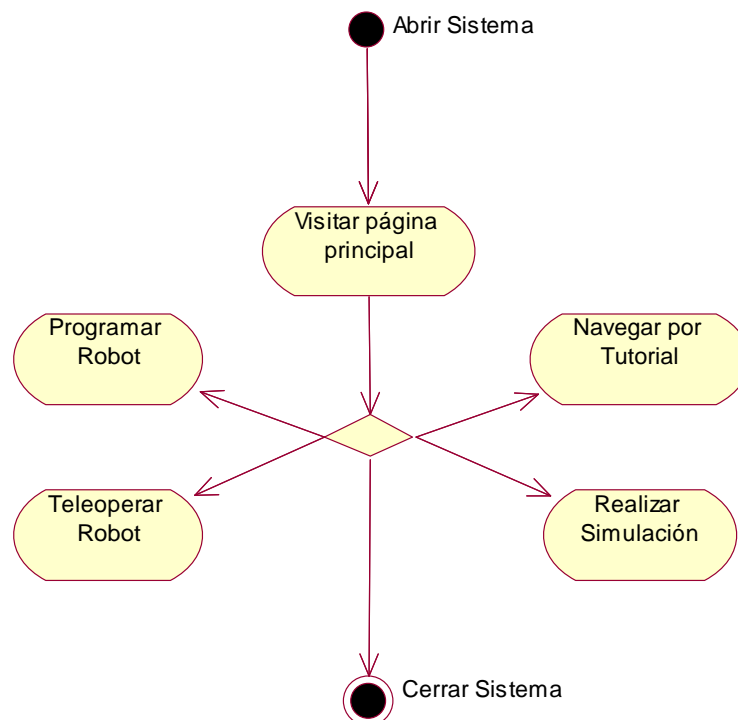


Fig. A.14 Diagrama de actividades del sistema

Navegación por contenidos

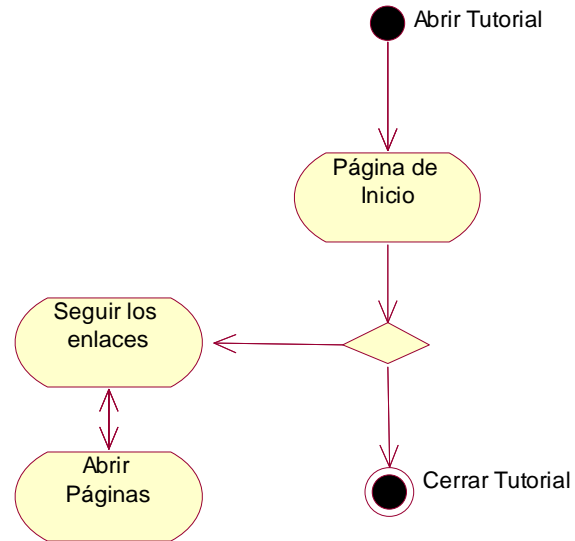


Fig. A.15 Diagrama de actividad Navegación por Contenidos

Simulación

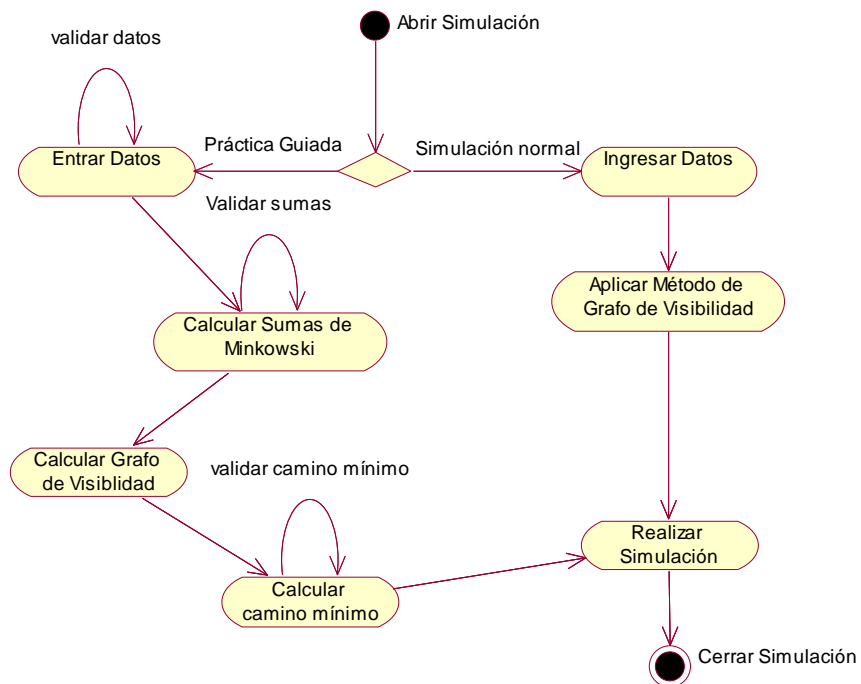


Fig. A. 16 Diagrama de actividad Simulación

Teleoperación

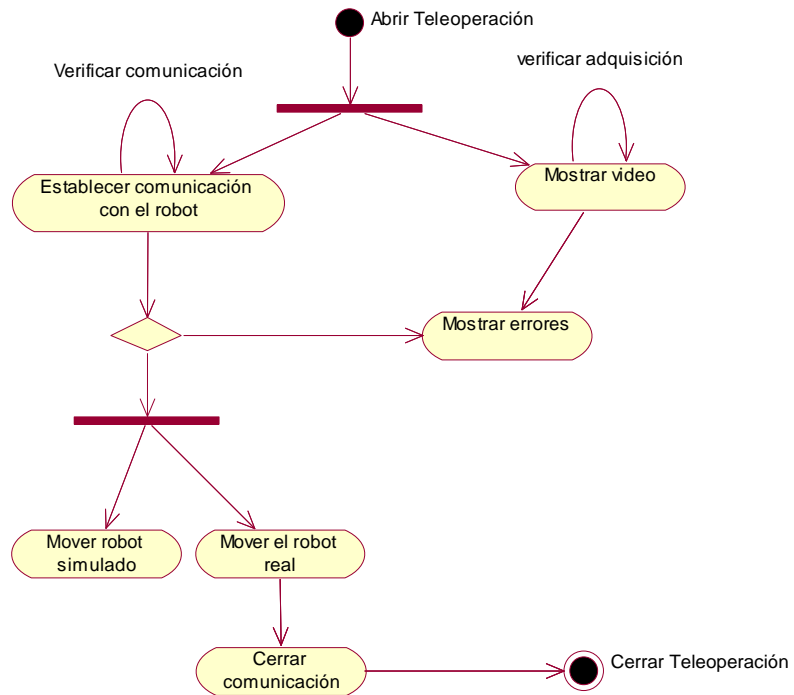


Fig. A.17 Diagrama de actividad Teleoperación

Programación

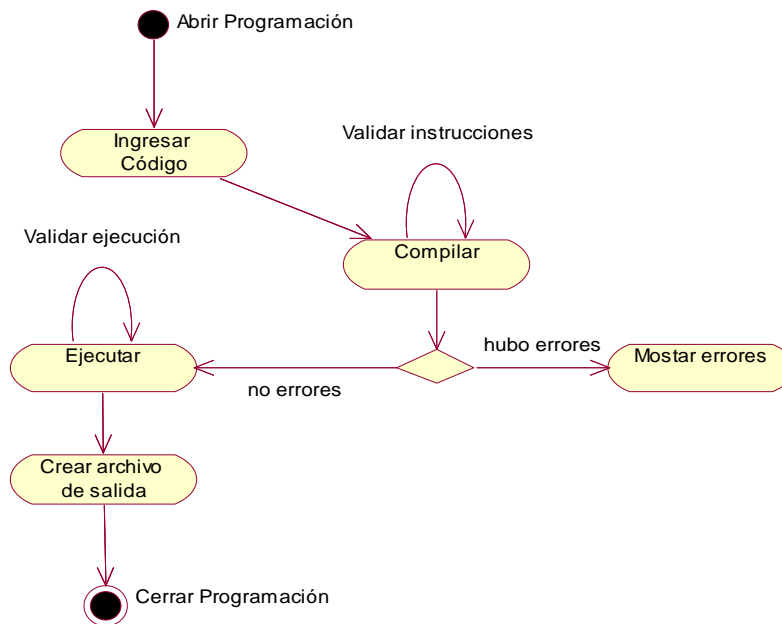


Fig. A.18 Diagrama de actividad Programación

ANEXO B - ESTRUCTURA DEL TUTORIAL DE ROBÓTICA MÓVIL

Mapa del Sitio

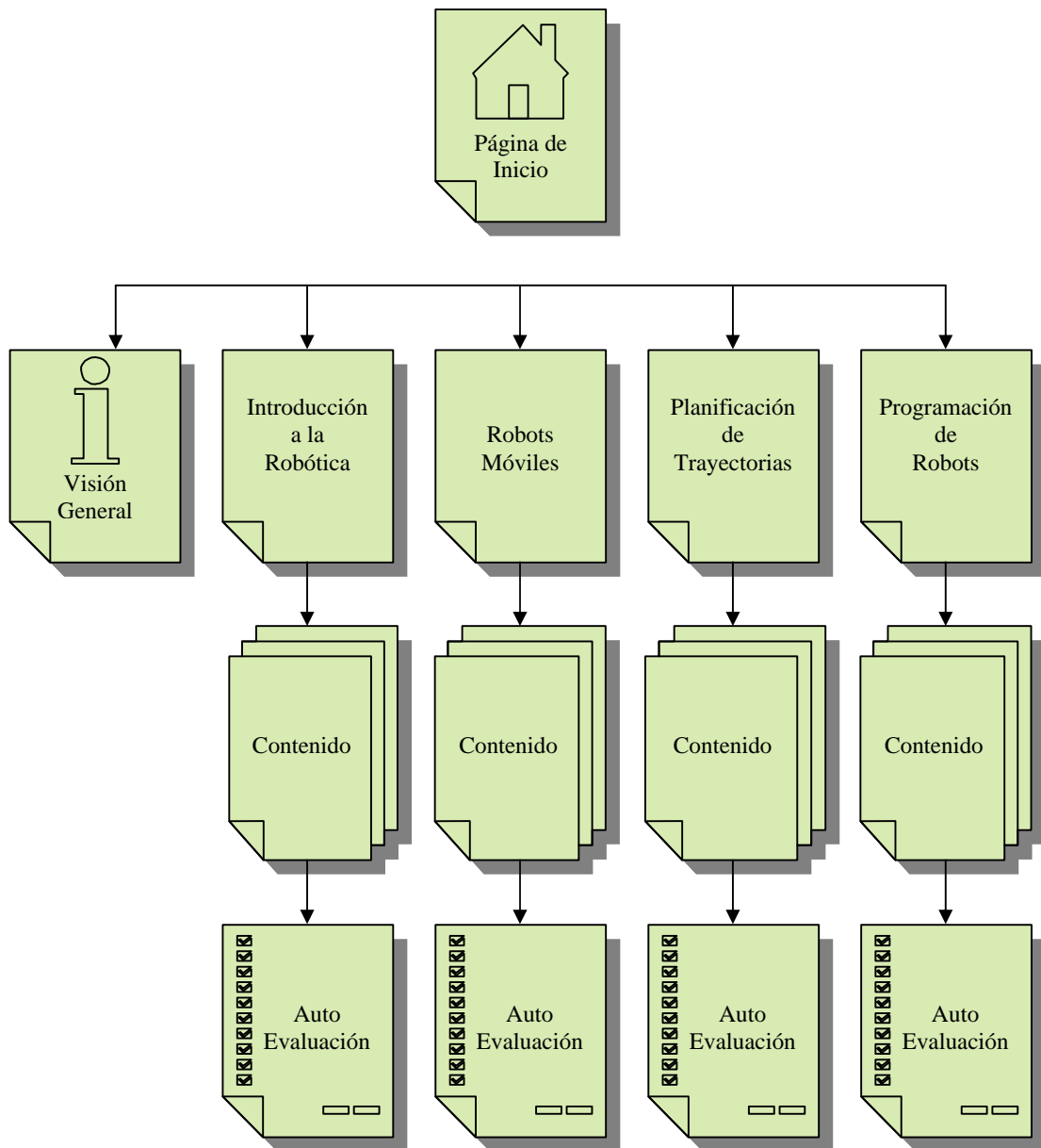


Fig. B.1 Mapa del sitio del Tutorial de Robótica

Contenidos

A continuación se listan los temas y subtemas tratados en cada una de las lecciones del Tutorial de Robótica Móvil. Para ver a detalle los contenidos es necesario revisar el tutorial que consta en el CD de aplicación.

Visión General

Bienvenida

Introducción

Objetivos

Lección 1: Introducción a la Robótica

Plan de Lección

Bienvenida

Introducción

1.1 Definiciones Básicas

1.1.1 ¿Qué es un robot?

1.1.2 Robótica

1.2 Breve historia de la robótica

1.3 Tipos de robots

1.4 Los robots y el trabajo

1.5 Aplicaciones de los robots

1.5.1 Robots industriales

1.5.2 Robots de servicio

1.5.3 Robots de exploración

1.5.4 Otros robots

1.6 Estructura interna de un Robot

1.6.1 Sistema de control (sistema nervioso)

1.6.2 Sensores

1.6.3 Efectores y actuadores

1.6.4 Sistema de locomoción/manipulación

1.7 Cinemática y Dinámica

1.7.1 Grados de Libertad

Desafío

Autoevaluación

Retroalimentación

Lección 2: Robots Móviles

Plan de Lección

Bienvenida

Introducción

2.1 Definición

2.2 Robots Móviles con ruedas

2.2.1 Ruedas

2.2.2 Giro

2.2.3 Arreglos de ruedas

2.2.4 Cinemática

2.2.5 Forma del Robot

2.3 Robots Móviles con patas

2.3.1 Movimiento básico de un hexápodo

2.4 Arquitecturas Básicas de Control

2.4.1 Arquitectura deliberativa

2.4.2 Arquitectura reactiva

2.4.3 Arquitecturas híbridas

2.4.4 Robótica de comportamientos base

2.4.5 Algoritmo Básico de Navegación

Desafío

Autoevaluación

Retroalimentación

Lección 3: Planificación de trayectorias de Robots Móviles

Plan de Lección

Bienvenida

Introducción

1.1 Navegación

1.1.1 Esquema de la navegación

1.1.2 Métodos de planificar un camino

1.2 El problema de la planificación de trayectorias de robots móviles

1.3 Métodos Clásicos de Planificación

1.3.1 Planificación basada en grafos de visibilidad

1.3.2 Planificación basada en diagramas de Voronoi

1.3.3 Planificación basada en el modelado del espacio libre

1.3.4 Planificación basada en descomposición de celdas

1.3.5 Planificación basada en campos de potencial

1.4 Método del Grafo de Visibilidad

1.4.1 Algoritmo de Grafo de Visibilidad

1.4.2 Robot geométrico

1.5 Ejemplo de planificación con grafo de visibilidad

1.6 Funciones de detección, evitación y planificación

1.7 Reacción directa a información de sensores

Desafío

Autoevaluación

Retroalimentación

Lección 4: Programación de Robots

Plan de Lección

Bienvenida

Introducción

1.1 Programación de tareas

1.2 Requerimientos de los lenguajes de programación de robots

1.3 Sistemas operativos

1.4 Clasificación de los lenguajes de programación de robots

1.4.1 Secuenciadores

1.4.2 Extensiones a lenguajes clásicos

1.4.3 Lenguajes específicos

1.4.4 Orientados al robot

1.4.5 Orientados a la tarea

Desafío

Autoevaluación

Retroalimentación

ANEXO C - ALGORITMOS GEOMÉTRICOS

1. Algoritmo de Sumas de Minkowski

Dados dos conjuntos P y $Q \subset \mathbb{R}^2$, la suma de Minkowski de P y Q , denotada por $P \oplus Q$

$\oplus Q$ se define como

$$P \oplus Q = \{p + q : p \in P, q \in Q\}$$

donde $p + q$ es un vector que representa la suma de los vectores p y q . Es decir que dados los puntos $p = (p_x, p_y)$ y $q = (q_x, q_y)$, tenemos que $p + q = (p_x + q_x, p_y + q_y)$.

Es posible aplicar la definición de sumas de Minkowski a los polígonos, ya que un polígono es un conjunto de puntos en el plano.

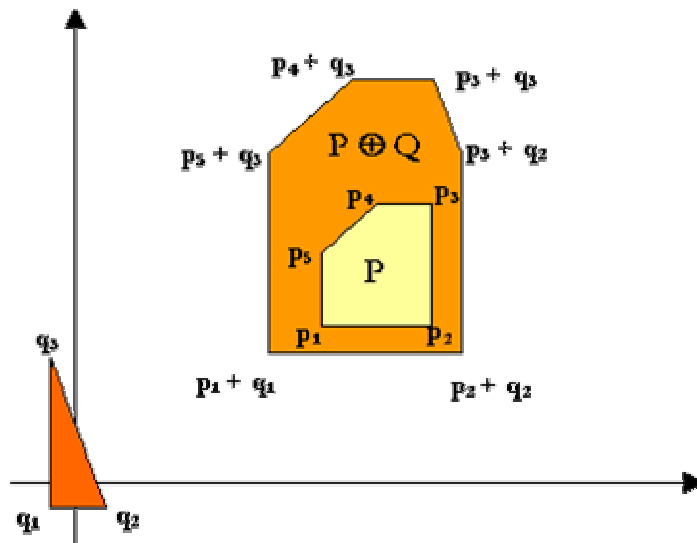


Fig. C.1 Sumas de Minkowski de dos polígonos

Sumas de Minkowski de dos polígonos convexos

Dados dos polígonos convexos P y Q con n y m vértices respectivamente, la suma $P \oplus Q$ se calcula:

Para una dirección dada \vec{d} , un punto extremo en $P \oplus Q$ es la suma de los puntos extremos de P y de Q en la misma dirección \vec{d} . En un polígono convexo, si se cambia la dirección \vec{d} siguiendo el sentido contrario a las agujas del reloj, se obtiene una secuencia de puntos extremos que contienen los vértices del polígono ordenados exactamente como están ubicados si se recorre el borde del polígono en el sentido contrario a las agujas del reloj.

El siguiente algoritmo, recorre las direcciones siguiendo el sentido contrario a las agujas del reloj y usa la observación anterior para recorrer ambos polígonos y encontrar sus puntos extremos.

En este algoritmo la notación $\text{ángulo}(pq)$ denota el ángulo que forma el vector \overrightarrow{pq} con el eje positivo de abscisas.

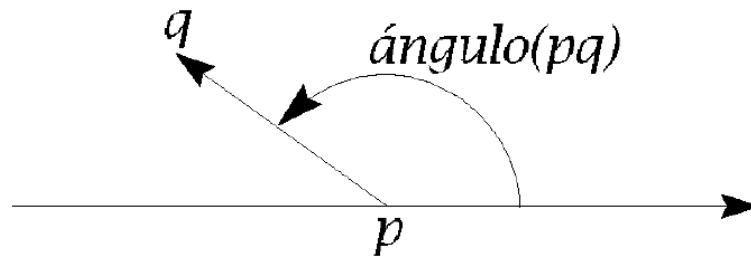


Fig. C.2 Ángulo que forma el vector \overrightarrow{pq} con el eje de abscisas

Algoritmo:

Entrada: Un polígono convexo P con vértices v_1, \dots, v_n , y un polígono convexo Q con vértices w_1, \dots, w_m . Se asume que la lista de vértices estará ordenada siguiendo el sentido contrario al de las agujas del reloj, siendo v_1 y w_1 los vértices con menor coordenada y .

Salida: La suma de Minkowski de $P \oplus Q$

1. $i \leftarrow 1 ; j \leftarrow 1$
2. $v_{n+1} \leftarrow v_1 ; w_{m+1} \leftarrow w_1$
3. **Repetir**
4. Agregar $v_i + w_j$ como un vértice en $P \oplus Q$
5. **Si** $\text{ángulo}(v_i v_{i+1}) < \text{ángulo}(w_j w_{j+1})$
6. **entonces** $i \leftarrow (i + 1)$
7. **sino Si** $\text{ángulo}(v_i v_{i+1}) > \text{ángulo}(w_j w_{j+1})$
8. **entonces** $j \leftarrow (j + 1)$
9. **sino** $i \leftarrow (i + 1)$
10. $j \leftarrow (j + 1)$
11. **Si** $i > n+1$ **entonces** $i \leftarrow n+1$
12. **Si** $j > m+1$ **entonces** $j \leftarrow m+1$
13. **hasta que** $i = n + 1$ y $j = m + 1$

Este algoritmo es sencillo y fácil de implementar. Se ejecuta en un tiempo lineal $O(m+n)$ ya que en cada ejecución de la sentencia de repetición ocurre que o i o j son incrementados sólo hasta alcanzar los valores $n+1$ y $m+1$. Cualquier vértice de la suma de Minkowski $P \oplus Q$ es la suma de dos vértices originales que son

extremos en una dirección común en P y Q . Ya que los polígonos son convexos el testeo del ángulo asegura que esos pares extremos son encontrados.

2. Algoritmo de Grafo de Visibilidad

Entrada: Los polígonos $B_j = \{v_i\}$ definidos por sus vértices.

Salida: El grafo de visibilidad G_v .

Todos los vértices de los polígonos forman el conjunto de vértice del grafo.

Las aristas que delimitan los polígonos pertenecen al conjunto de aristas del grafo.

```
1 Para  $i = 1, \dots, n$  hacer
2     Para cada  $j = 1, \dots, n$  hacer
3         Para cada vértice  $v_r$  del polígono  $P_i$  hacer
4             Para cada vértice  $v_q$  del polígono  $P_j$  hacer
5                 Para cada arista  $a$  de cualquier polígono hacer
6                     Si el arco  $(v_r, v_q)$  intersecta con  $a$  entonces
7                         Ir al paso 4
8                     Fin si
9                 Fin para
10                incluir  $(v_r, v_q)$  en el conjunto de aristas del grafo
11            Fin para
12        Fin para
13    Fin para
14 Fin para
```

3. Algoritmo Test de intersección

El test de intersección se usa en el algoritmo de grafo de visibilidad para verificar si la línea de visibilidad interfecta alguna arista del obstáculo. Los algoritmos utilizados para el Test de Intersección se muestran a contiinuación:

Para verificar si una línea L interseca la línea L1

```
bool Intersects(L,L1)
{
    return (cruza(L1) && cruza(L));
}
```

Dados dos lineas L y L1, se tiene el test de cruce

```
bool cruza(L)
{
    return ((Det(L.P1) * Det(L.P2)) < 0);
}
```

```
float Det(P)
{
    return (((P1.X * P2.Y) + (P.X * P1.Y) + (P2.X * P.Y)) -
            ((P1.X * P.Y) + (P.X * P2.Y) + (P2.X * P1.Y)));
}
```

4. Algoritmo de DIJKSTRA

Este algoritmo calcula los caminos más cortos desde un solo origen a cada uno de los nodos del grafo.

Aplicado al cálculo de camino mínimo en un Entorno Poligonal, como se ha explicado anteriormente en el cálculo del grafo de visibilidad, los nodos del grafo serán los vértices de los obstáculos, los arcos, las rectas del grafo de visibilidad, y la distancia será la Euclídea entre dos puntos.

El algoritmo consiste en lo siguiente:

Sea un grafo $G=(N,A)$, donde N es el conjunto de nodos y A es el conjunto de arcos de G . Cada arco tiene asociada una longitud (o coste o distancia) no negativa.

La longitud de un camino es la suma de las longitudes de los arcos que forman el camino. Cada camino comienza en un nodo, llamado nodo origen, y termina en otro, llamado nodo destino.

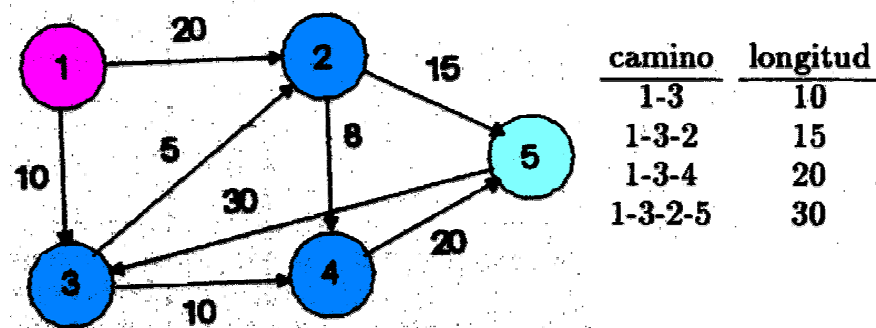


Fig. C.3 Grafo con caminos y longitud

En la figura se adjuntan las distancias mínimas del nodo origen 1 a todos los demás, ordenadas en orden no decreciente. También se incluye el camino que posee cada longitud mínima, aunque esta información no es pedida en el problema.

En cuanto al modo de construir la solución, pueden irse construyendo los caminos mínimos uno a uno. Esto es realizable si se toma como conjunto de candidatos el de nodos; cada vez que se elige un nodo, se dispone de su distancia mínima.

El nodo que se elige en cada momento es aquel candidato que está más cerca del origen, es decir, las distancias mínimas se van hallando en orden no decreciente de longitud. En el ejemplo propuesto, primero se obtiene la distancia mínima (de 10) hasta hasta el nodo 3, después hasta el nodo 2 (de 15) y así sucesivamente.

Implementación del Algoritmo:

```
DistanceToFin(Nodo fin)
{
    if (_distancetofin != -1)
        return _distancetofin;

    if (this.Equals(fin))
    {
        _distancetofin = 0;
        return 0;
    }
}
```



```

if (Nodos.Count == 0)
{
    _distancetofin = double.PositiveInfinity;
    return _distancetofin;
}

double minimo = double.PositiveInfinity;
double distancia;

foreach (Nodo N in Nodos)
{
    distancia = N.DistanceToFin(fin);
    if (distancia < minimo)
    {
        minimo = distancia;
        Min = N;
    }
    if (minimo == 0) break;
}

if (Min != null)
{
    _distancetofin = minimo + Linea.distance(_Posicion,
Min.Posicion);
    return _distancetofin;
}

return double.PositiveInfinity;
}

```

ANEXO D - MANUAL DE USUARIO

1. Instalando el sistema

1.1 Introducción

El Laboratorio virtual básico de Robótica Móvil es una herramienta para el aprendizaje de robótica móvil, el sistema cuenta con un simulador para la planificación de trayectorias, un entorno de programación y de teleoperación de un prototipo móvil.

Es una herramienta con fines de investigación y docencia la cual sirve tanto para asentar conceptos teóricos como para investigar nuevas técnicas y métodos para robótica móvil. Dado que es una herramienta interactiva, facilita mucho su uso y la rápida experimentación por parte del usuario.

1.2 Requerimientos del sistema

Computador Personal que tenga, a lo menos:

- 16 Mb. de memoria principal
- Disco duro con a lo menos 30 Mb. de espacio libre.
- Sistema Operativo Windows 95-98-2000-Me-NT-XP
- No es compatible con Macintosh
- Lego mindstorms 1.1 o superior

- MSAgent API para sistemas operativos diferentes de XP

1.3 Instalación

En cuanto a la instalación de la aplicación, no existe como tal, puesto que al estar desarrollada como una interfaz web se ejecuta directamente desde el navegador bastando con introducir el CD.

2. Ingresando al sistema

2.1 Ingreso al sistema

Inserte el Cd del Laboratorio Virtual, inmediatamente aparecerá la ventana principal en donde podrá acceder a las diferentes opciones.

2.2 Entorno del sistema

Esta es la ventana inicial del laboratorio virtual de robótica móvil, el tutor inteligente será su guía y le explicará a donde le llevarán las diferentes opciones.



3. El entorno de trabajo

3.1 Subsistema Tutorial

Ventana principal

Cuando ingresa al entorno del Tutorial puede observar dos partes importantes:

Menú de Contenido: Le permite desplazarse hacia las diferentes lecciones que componen el tutorial.

Área de Contenido: Es en donde se muestra el contenido de los enlaces del menú, esta área consta además en su parte superior con información

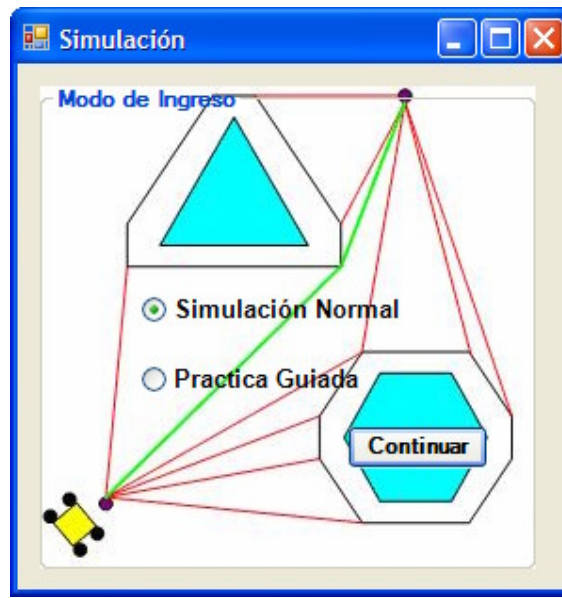
sobre el número de página en la que se encuentra del total que componen la lección y los botones de desplazamiento para moverse entre las páginas.



3.2 Subsistema Simulación

Ventana de ingreso

Al ingresar al área de simulación aparecerá la ventana de ingreso donde debe escoger el modo de entrada a la simulación que puede ser Simulación normal o Práctica Guiada.

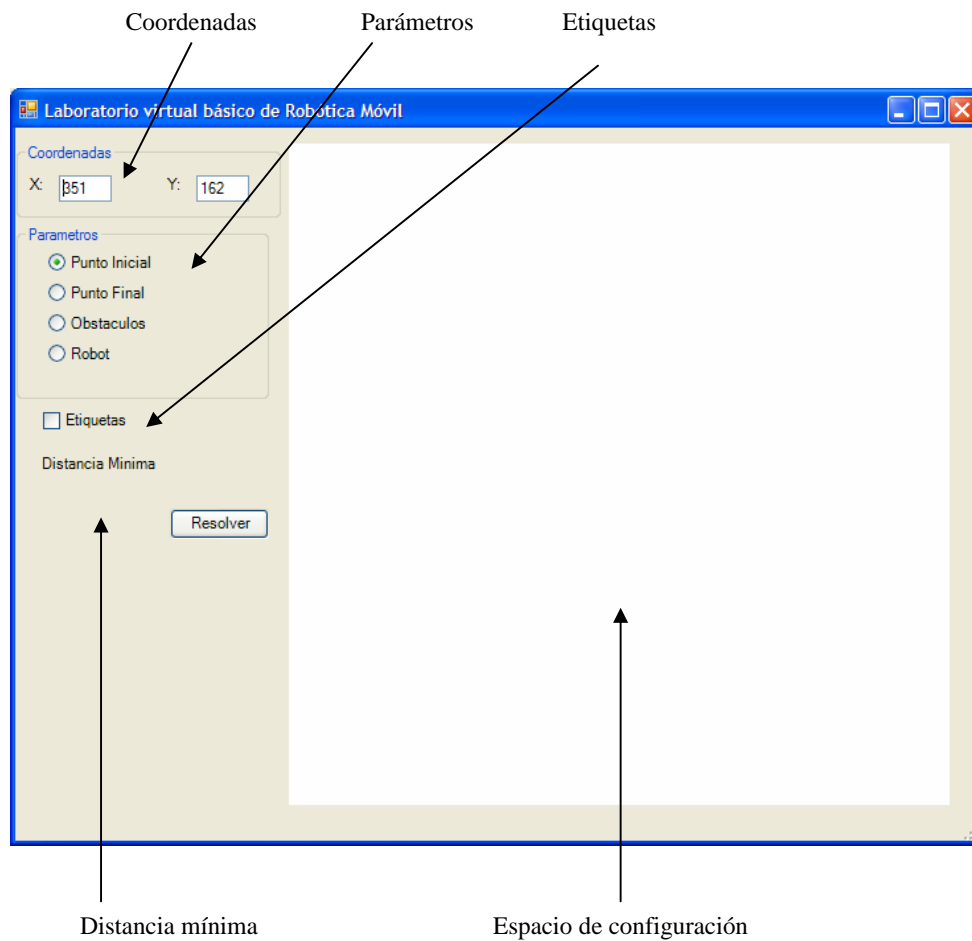


Ventana de ingreso

Modo Automático

Al elegir el modo de simulación normal el entorno de la aplicación es como se aprecia en la figura, en donde se puede observar las siguientes partes:

- **Coordenadas:** Permite visualizar las coordenadas X e Y dentro del espacio de configuración.
- **Parámetros:** Sirven para el ingreso de los datos del problema de planificación de trayectorias.
- **Etiquetas:** Muestra u oculta las coordenadas de los vértices de los obstáculos ingresados.
- **Distancia mínima:** Muestra el coste del camino mínimo calculado.
- **Espacio de configuración:** espacio en donde se desenvuelve le problema.

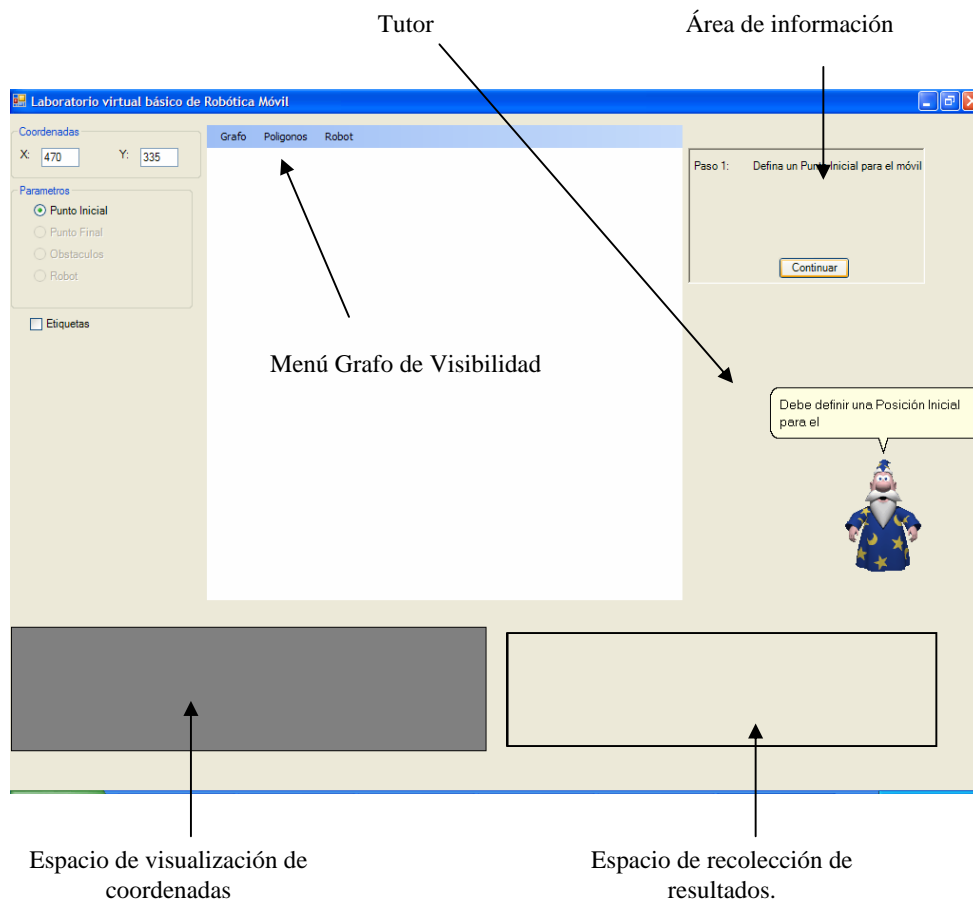


Modo de Práctica Guiada

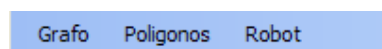
En el modo de práctica guiada la interfaz difiere un poco del modo de simulación normal y adicionalmente se encuentran los siguientes elementos:

- **Menú del Método de grafo de visibilidad:** Permite realizar paso a paso la resolución del problema en la práctica guiada.
- **Área de información:** Indica en que paso se encuentra la práctica.

- **Tutor:** Es un agente animado que le guía paso a paso en el desarrollo de la práctica.
- **Espacio de recolección de resultados:** Permite recibir los datos de cálculos que serán verificados para comprobar que esta realizando bien el ejercicio.
- **Espacio de visualización de coordenadas de obstáculos:** Muestra las coordenadas de cada uno de los vértices de los obstáculos ingresados.



Barra de Menús del Método de grafo de Visibilidad



Menú Grafo

Permite aplicar el grafo de visibilidad y calcular el camino mínimo.

Menú Polígono

Permite aplicar el algoritmo de sumas de Minkowski para ampliar los obstáculos poligonales.

Menú Robot

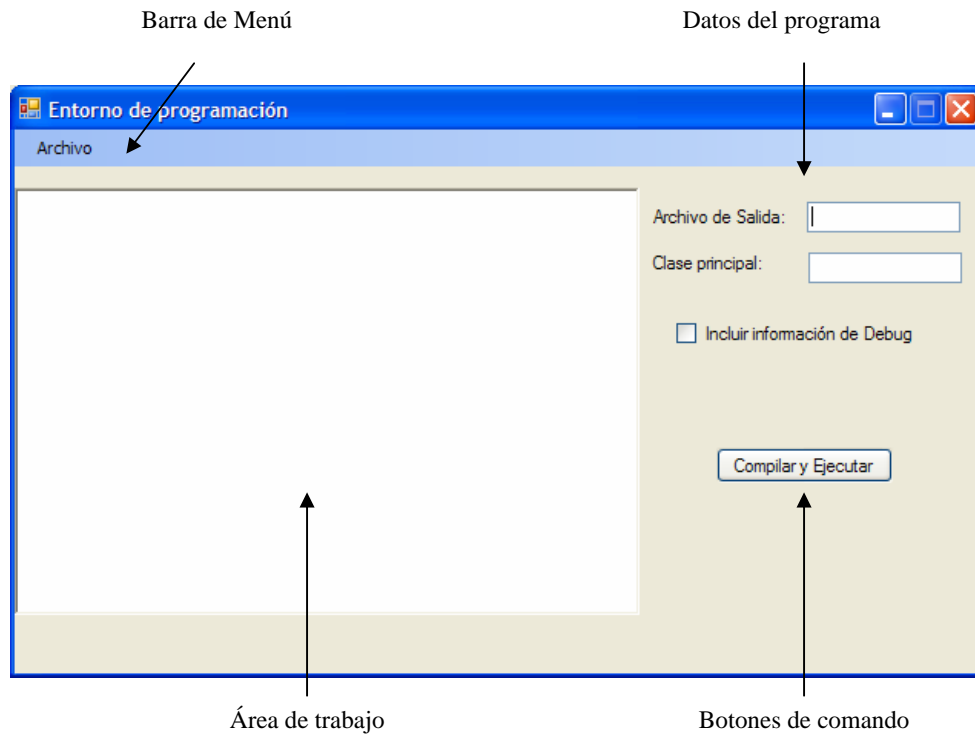
Permite acceder a la opción que realiza la simulación del recorrido del robot por el camino mínimo.

3.3 Subsistema Programación

Ventana principal

En este entorno de programación se pueden apreciar las siguientes partes:

- **Área de trabajo:** En donde se ingresan las instrucciones para la programación del robot móvil.
- **Área de datos del programa:** en donde se ingresan el nombre del archivo de salida a generar y el nombre de la clase principal.
- **Botones de comando:** Permiten la compilación y ejecución del código ingresado en el área de trabajo.
- **Barra de menú:** Permite acceder a las diferentes opciones de manejo de archivos.



Menú Archivo

Abrir Permite cargar un archivo de código al área de trabajo.

Guardar Permite guardar el programa realizado en el área de trabajo.

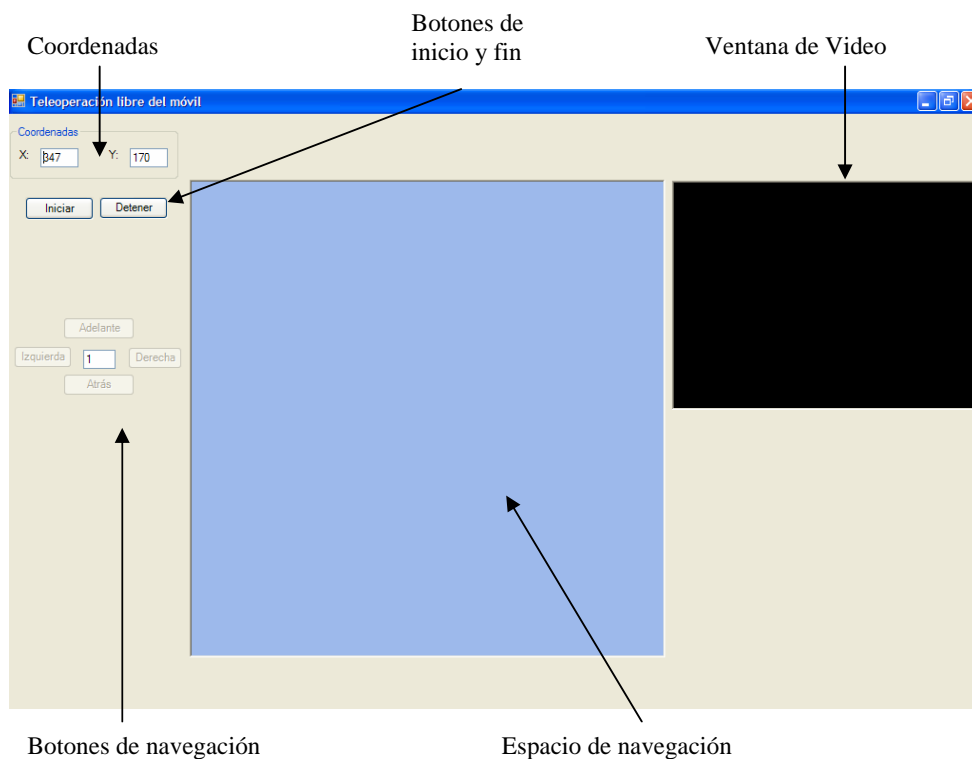
Salir Cierra el entorno de programación.

3.4 Subsistema Teleoperación

Ventana principal

En el entorno de teleoperación se pueden apreciar los siguientes elementos:

- **Coordenadas:** Permite visualizar las coordenadas X e Y dentro del espacio de configuración.
- **Botones de inicio y fin:** El primero inicializa la conexión con el robot móvil y el segundo corta y cierra la comunicación con el dispositivo.
- **Botones de navegación:** Permiten teleoperar el robot móvil en las 4 direcciones preestablecidas.
- **Ventana de video:** Visualiza el robot móvil mediante la adquisición de imágenes de una cámara de video.
- **Entorno de navegación:** Área donde se ubica el móvil simulado y que permite su localización.



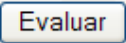
4. Utilizando el laboratorio virtual

4.1 Navegando por el contenido del tutorial

Para navegar por el entorno bastará ir pulsando la opción deseada en el índice de contenidos que aparece en la parte izquierda de cada página o a su vez en los enlaces disponibles.



4.2 Realizar una evaluación

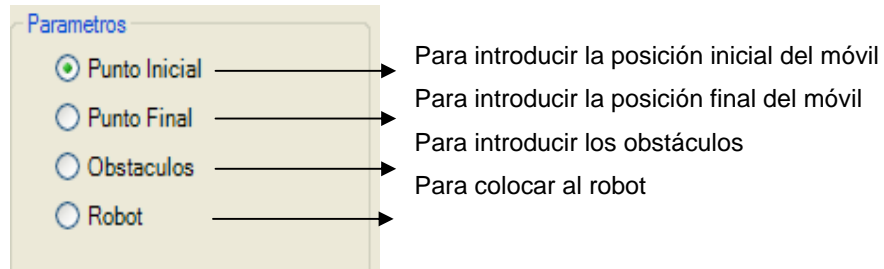
Para realizar la evaluación de los conocimientos teóricos simplemente debe ubicar la misma dentro del tutorial contestar las preguntas que le presente y pulsar el botón  que se encuentra al final, el cual le mostrará los resultados de su evaluación.

4.3 Iniciar una simulación

Para iniciar una simulación acceda al entorno de simulación y escoja la opción de Simulación Normal.

4.3.1 Introducir datos en la simulación

Para introducir los datos a la simulación debe usar el ratón, para ello debe seleccionar el tipo de dato a introducir desde el área de parámetros.

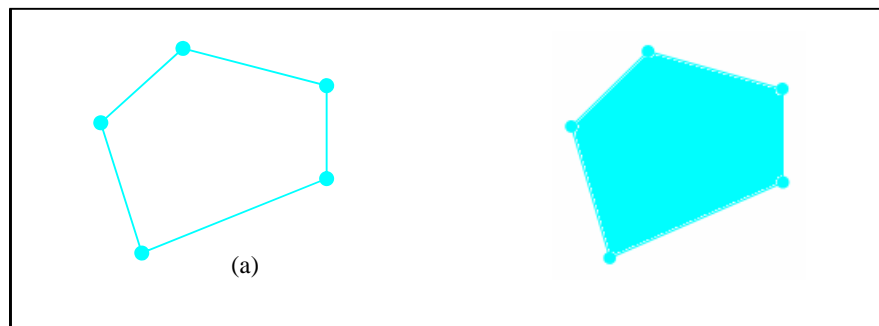


4.3.2 Introducir obstáculos

Para dibujar un polígono basta con marcar los puntos que lo forman a lo largo del espacio de configuración.

Para añadir un punto se hará click con el botón izquierdo del ratón quedando marcada la coordenada del punto en la pantalla. Así procederemos, moviéndonos por el área de dibujo hasta que hayamos añadido el último punto. Ver figura (a).

Pulsando el click con el botón derecho del ratón se concluye con la introducción de puntos al polígono, y es en este momento cuando se rellena de color el polígono resultante. Ver figura (b).

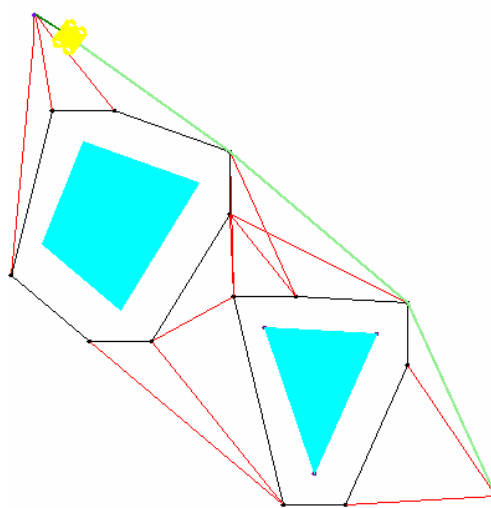


En la parte inferior izquierda de la pantalla puede observar un cuadro con información de cada obstáculo ingresado y los respectivos vértices que lo componen.

	V1	V2	V3	V4
▶ O1	(148,153)	(243,178)	(179,276)	(127,214)
O2	(299,311)	(383,358)	(299,428)	

4.3.3 Obtener la solución

Una vez ingresados los datos del problema, para obtener la solución se pulsa en el botón , en donde se observará pausadamente la ampliación de los obstáculos, la construcción del grafo de visibilidad y el cálculo del camino mínimo.

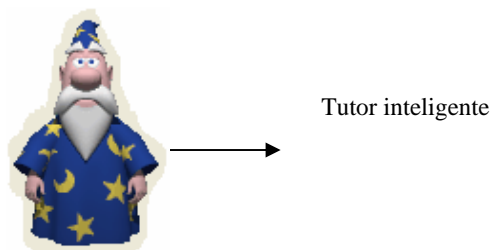


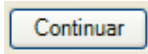
Al empezar la simulación obtendrá en pantalla un cuadro de resumen que contiene los puntos de la trayectoria que debe seguir el móvil y el ángulo que debe girar entre cada punto.

	Desde	Hacia	Angulo de Giro
▶	(64 , 39)	(220 , 150)	35.43
	(220 , 150)	(363 , 271)	4.8
	(363 , 271)	(434 , 426)	25.15

4.4 Iniciar una práctica guiada

Para iniciar una práctica guiada acceda al entorno de simulación y escoja la opción de Práctica Guiada, y pulse Continuar. Una vez que la ventana se haya abierto usted verá al tutor inteligente, el cual lo guiará paso a paso en la realización de la práctica.



Para avanzar al siguiente paso debe pulsar el botón  que se encuentra en el área de información.

4.4.1 Introducir datos

La introducción de datos en la práctica guiada es igual al modo de simulación normal. Ver apartado 4.3.1.

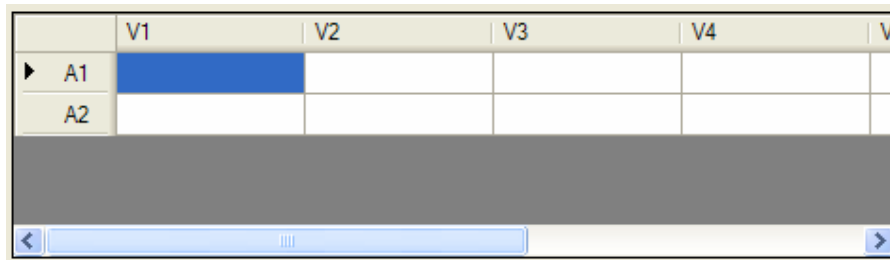
4.4.2 Introducir obstáculos

El ingreso de obstáculos en la práctica guiada es igual al modo de simulación normal. Ver apartado 4.3.2.

4.4.3 Introducir datos solicitados por el tutor

Para comprobar la correcta realización de la práctica guiada debe ingresar en algunos pasos los datos calculados, los cuales serán verificados por el tutor y en el caso de ser correctos podrá avanzar al siguiente paso, de lo contrario deberá revisar sus cálculos.

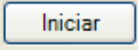
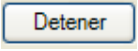
Estos resultados se ingresan en un grid destinado para el efecto y que aparece en la parte inferior derecha de su pantalla.



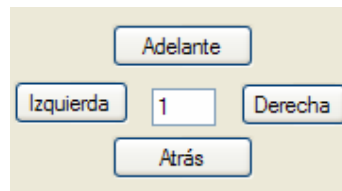
	V1	V2	V3	V4	V
A1					
A2					

4.5 Teleoperando al robot

Una vez abierto el entorno de teleoperación para empezar a guiar libremente al robot móvil debe seguir los siguientes pasos:

2. Colocar el robot simulado en el entorno de navegación.
3. Pulsar e el botón  , para establecer comunicación con el dispositivo real.
4. Utilizar los botones de navegación para dirigir al robot móvil, observe que el robot simulado realiza exactamente los mismos movimientos en el entorno de navegación para ayudarle a su localización.
5. Para finalizar debe presionar el botón  para cerrar la comunicación con el dispositivo real.

Cabe mencionar que en el centro de lo botones de navegación se encuentra un casillero, el cual permite ingresar el tiempo en segundos que el robot ejecutará los movimientos de Adelante y Atrás.



Los movimientos de Izquierda y Derecha son no hacen uso del casillero de tiempo, es decir, son giros fijos de 90 grados en las respectivas direcciones.

4.6 Creando un programa

En el entorno de programación puede realizar cualquier programa de usuario ingresando las sentencias correspondientes en el área de trabajo.

```
using System;
using RCXNET;

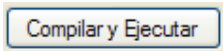
namespace ConsoleTester
{
class Class1
{
[STAThread]

static void Main(string[] args)
{
try
{
SpiritNET obj = new SpiritNET();

SpiritNET.eStatusResult resPort= obj.Status
```

Cuando haya terminado el programa debe asignar un nombre para el archivo de salida que se generará y especificar el nombre de la clase principal que uso en su programa.

Archivo de Salida: ejemplo.exe
Clase principal: Class1

Para compilar y ejecutar el código creado debe asegurarse que los parámetros anteriores fueron ingresados; si no existen errores el código se descarga al robot móvil y se ejecuta en él. Esta operación se realiza pulsando el botón .

Puede grabar sus programas o recuperarlos desde el menú Archivo.



Recuerde que la programación se realiza en lenguaje C# de acuerdo a los mismos convencionalismos del NET Framework.

4.7 Errores

Existen mensajes de error que pueden aparecer durante el uso del sistema.

Los errores que se pueden producir son:

Simulación

1. **“Ingrese el punto inicial”**, cuando no se ha ingresado la posición inicial del móvil.
2. **“Ingrese el punto final”**, cuando no se ha ingresado la posición final del móvil.
3. **“Debe ingresar al menos un obstáculo”**, si no se han ingresado obstáculos en el espacio de configuración.
4. **“El vértice no puede contener una coordenada Y mayor a la del primer vértice”**, si no se ha seguido el orden de ingreso de los vértices en dirección de las manecillas del reloj.
5. **“Debe ampliar los obstáculos utilizando el algoritmo de sumas de Minkowski”**, cuando no se ha accedido la opción que calcula la ampliación de los polígonos.
6. **“El vértice n del obstáculo n no es correcto, revise los cálculos”**, si ha ingresado un valor incorrecto para el vértice de algún polígono ampliado.
7. **“El obstáculo ampliado contiene n vértices, revise los cálculos”**, cuando se ha ingresado un menor número de vértices calculados para u polígono ampliado.

8. **“El vértice n del obstáculo n no contiene un dato válido”**, si el vértice ingresado de un polígono ampliado no contiene el formato de ingreso adecuado (X,Y).
9. **“Calcule el grafo de visibilidad”**, si aun no se ha accedido a la opción que calcula el grafo de visibilidad.
10. **“Aun no ha calculado el camino mínimo”**, si aun no se ha accedido a la opción que calcula el camino mínimo.
11. **“El vértice n del camino mínimo no es correcto, revise los cálculos”**, cuando se ha ingresado un valor incorrecto de algún vértice que pertenece al camino mínimo.
12. **“La distancia del camino mínimo no es correcta, revise los cálculos”**, si se ha ingresado un valor que no corresponde al coste del camino mínimo.
13. **“El vértice n del camino mínimo no contiene un dato válido”**, si el vértice ingresado del camino mínimo no contiene el formato de ingreso adecuado (X,Y).

Teleoperación

1. **“Coloque primero el robot móvil”**, si aun no se ha colocado el robot móvil en el espacio de navegación.
2. **“Ya ha colocado el robot móvil”**, cuando el robot móvil ya ha sido colocado en el espacio de navegación.

3. **“Ha ocurrido un error de comunicación con el robot móvil”**, si el robot móvil ha salido fuera del alcance de la señal de la torre IR.
4. **“No se pudo iniciar la captura”**, si no se pudo capturar las imágenes desde la cámara de video.
5. **“No se pudo encontrar el driver”**, cuando no se encontró el driver de la cámara de video.
6. **“No se pudo crear la ventana de captura”**, si no fue posible crear la ventana de captura del video.

Programación

2. **“El nombre de la Aplicación no puede dejarse en blanco”**, si no ha especificado un nombre para el archivo de salida que se va a generar.
3. **“El nombre de la clase principal no puede dejarse en blanco”**, si no ha especificado el nombre de la clase principal que uso en el programa.
4. **“No se puede ejecutar”**, si hay algún error de enlace y el código no se puede ejecutar.

BIBLIOGRAFIA

- ROSENBERG, M. J. (2000). *E-Learning: Strategies for Delivering Knowledge in the Digital Age*. New York: McGraw-Hill Professional Publishing
- HARTLEY, D. E. (2000). *On-Demand Learning: Training in the New Millennium*. Boston, MA: HRD Press.
- DAVIES, J.R., GERTNER, A. S., LESH, N., RICH, C., RICKEL, J., SIDNER, C. L. (2001). Incorporating Tutorial Strategies into an Intelligent Assistant. Proceedings of Intelligent User Interfaces 2001, January 2001, Santa Fe, New Mexico, USA. Disponible electrónicamente en <http://www.merl.com/projects/collagen/>.
- BRICALL, J. M. et al. (2000). *Informe Universidad 2000*. Disponible electrónicamente en <http://www.crue.upm.es/documen.htm>
- HANSEN, John, THOMSEN, Carsten. *Enterprise development with Visual Studio.NET, UML and MSF*. Apress 2004.
- Microsoft Corporation; *"Microsoft Solution Framework White Paper"*, Microsoft Corporation, Diciembre 1999.

- Microsoft Corporation; “*MSF Process Model v. 3.1. Microsoft Corporation*”,
Disponible en: <http://www.microsoft.com/msf/>. June 2002.
- UML.
<http://programacion.com/tutoriales/UML/>
<http://www.dcc.uchile.cl/~psalinas/uml/>
- FERRARI, Mario, FERRARI, Giulio. *Building robots with Lego Mindstorms*.
Syngress Publishing 2002.
- FERRARI, Mario, FERRARI, Giulio, AGULLÓ, Miguel, et al. *Lego Mindstorms Masterpieces building and programming advanced robots*.
Syngress Publishing 2003.
- *Desarrollo de aplicaciones .NET con Visual C#*. McGraw Hill 2002
- *C# Manual de programación*. MCGraw Hill 2002
- *Superutilidades para C#*. McGraw Hill 2002
- MAHESH, Chand. *Graphics Programming with GDI+*.
- CAZORLA, Miguel; COLOMINA, Otto. *Robótica (Robots autónomos)*.
Universidad Autónoma de España 2005.

- DUDEKAND, G.; JENKIN M. *Computational Principles of Mobile Robotics*, Cambridge University Press, 2000.
- CAÑAS, José; MATELLÁN, Vicente; MONTÚFAR, Rodrigo. *Programación de robots móviles*. Universidad Rey Juan Carlos España. 2005.
- ARKIN R. *Behavior Based Robotics*. The MIT Press, 1998
- MURPHY R. *Introduction to AI Robotics*. The MIT Press, 2000
- GIAMARCHI, Frédéric. *Robots Móviles, estudio y construcción*. Paraninfo. 2000.

APÉNDICE

VITA



Información personal	<p>Nombres: Marianela Elizabeth Apellidos: Ortiz Beltrán Nacionalidad: Ecuatoriana Cédula Identidad: 180321606 - 6 Fecha de Nacimiento: 17-05-1979</p>
Educación	<p><u>Primaria</u> 1985 - 1991 Unidad Educativa Experimental “Pedro F. Cevallos” Ambato – Ecuador</p> <p><u>Secundaria</u> 1991 - 1997 Colegio Nacional Experimental de Señoritas “Ambato” Ambato – Tungurahua Bachiller en Ciencias Físico – Matemático Título Carreras Cortas en Electricidad Miembro del cuadro de honor.</p> <p><u>Superior</u></p> <p>Pontificia Universidad Católica 1995 - 1999 Ambato – Tungurahua Aptitud y Peritaje en Idioma Inglés</p> <p>Escuela Superior Politécnica de Chimborazo 1997 - 2002 Riobamba – Ecuador Analista de Sistemas Informáticos</p> <p>Escuela Politécnica del Ejército 2002 - 2006 Sangolquí – Ecuador Suficiencia en Idioma Inglés Ingeniera en Sistemas e Informática</p>

HOJA DE LEGALIZACION DE FIRMAS

ELABORADA POR

MARIANELA ELIZABETH ORTIZ BELTRÁN

COORDINADOR DE LA CARRERA

ING. RAMIRO DELGADO

Sangolquí, Diciembre del 2006

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.