UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

LOCALIZACIÓN Y CONTROL DE UN ROBOT MÓVIL PARA EL SEGUIMIENTO DE TRAYECTORIAS EN EL PLANO UTILIZANDO RETROALIMENTACIÓN VISUAL

T E S I S

QUE PARA OBTENER EL TÍTULO DE

INGENIERO MECATRÓNICO

PRESENTA

JESÚS ALEJANDRO PEÑA CASIMIRO

DIRECTOR DE TÉSIS

DR. VICTOR JAVIER GONZÁLEZ VILLELA

CIUDAD UNIVERSITARIA 2008

Dedicatoria

A mi papá Jesús por ser un gran ejemplo a seguir, por la formación que me ha brindado y por todas las enseñanzas que han permitido mi desarrollo.

A mi mamá Judis por el cariño, la alegría y el cuidado que he recibido en todo momento en el curso de mi formación personal y profesional.

A mi hermanita Titus por el aliento que me ha brindado y la felicidad que hemos compartido desde niños.

A Moni por la compañía y el apoyo a lo largo de una de las mejores etapas de mi vida.

Agradecimientos

Al Dr. Víctor Javier González Villela por la paciencia, la motivación y confianza brindada durante la realización de este trabajo.

A mis profesores de la Facultad de Ingeniería, en especial a los del departamento de Ingeniería Mecatrónica, a quienes debo mi formación profesional.

A mis amigos, quienes también contribuyeron con mi desarrollo.

A PAPIIT por el apoyo brindado para la elaboración de este trabajo.

A la Universidad Nacional Autónoma de México por la excelente educación recibida.

ÍNDICE GENERAL

Capítulo 1. Introducción

1. Introducción

El término robot, fue acuñado en 1920 y procede de la palabra checa *robota*, que significa “siervo” o en ruso “trabajo obligatorio” (Čapek, 1920).

La evolución natural de los procesos industriales y las tecnologías llevó al origen de los robots móviles que son aquellos robots que tienen la capacidad de desplazarse a través del medio para el cual están diseñados.

Los primeros robots móviles aparecen como solución al problema de proporcionar un medio de transporte eficaz para los materiales entre las distintas zonas de la cadena de producción de las industrias. Los primeros en aparecer son los vehículos guiados automáticamente (AGV por sus siglas en ingles), que se mueven por rutas previamente establecidas. El desarrollo de gran parte de los robots móviles se encuentra limitado a entornos industriales donde la navegación de vehículos se realiza con una capacidad sensorial y de razonamiento mínimas, la tarea se estructura en una secuencia de acciones en la que a su término el vehículo supone que ha alcanzado el objetivo para el que está programado, ante cualquier cambio inesperado en el área de trabajo que afecte el desarrollo normal de la navegación, el sistema de navegación del vehículo se encontrará imposibilitado para ejecutar acciones opcionales que le permitan reanudar su labor, lo cual se puede traducir en perdidas económicas.

La tendencia actual apunta hacia el robot móvil autónomo, que es aquel que plantea la capacidad de movimiento sobre entornos no estructurados, de los que se posee un conocimiento incierto, todo reconocimiento del ambiente de trabajo se logra mediante la interpretación de la información suministrada a través de sus sensores y del estado actual del vehículo.

El robot móvil autónomo se caracteriza por una conexión “inteligente” entre las operaciones de percepción y acción, que definen su comportamiento y le permite llegar a la consecución de los objetivos programados sobre entornos con cierto grado de incertidumbre. El grado de autonomía depende en gran medida de la facultad del robot para abstraer el entorno y convertir la

información obtenida en órdenes, de tal modo que, aplicados sobre los actuadores del sistema de locomoción, se garantice la realización eficaz de la tarea. De este modo, las dos grandes características que alejan al robot móvil autónomo de cualquier otro tipo de vehículo se relacionan a continuación de acuerdo con (Lozano, 1990):

:

- Percepción: Determina la relación del robot con su entorno de trabajo mediante el uso de los sensores.

- Razonamiento: Determina las acciones que se han de realizar en cada momento, según el estado del robot y su entorno, para alcanzar las metas asignadas.

Los tipos de robots móviles pueden ser de la más grande diversidad. Se clasifican de acuerdo al medio de locomoción utilizado o al medio en el cual trabajan, por lo que se puede decir que existen robots móviles de ruedas, patas, orugas, aéreos, espaciales, marinos, submarinos, por mencionar algunos. También se pueden clasificar con base en su tamaño, por lo que se puede decir que hay robots, minirobots y microrobots.

El tipo de robot móvil utilizado para la realización de alguna tarea es definido principalmente por las características del ambiente; así, se tiene que para ambientes interiores y exteriores donde el piso es plano se usan por lo general los vehículos de ruedas. Si la evolución del robot requiere de la movilidad en escaleras y pisos altamente irregulares, un robot de patas puede ser preferible. Sin embargo, si el terreno es sumamente irregular, un robot de orugas puede ser la elección aplicable. Hay que considerar que para construir un sistema de robot móvil no siempre es necesario construirlo desde cero, ya que es posible utilizar algún vehículo comercial, al cual se le agregan los elementos necesarios para robotizarlo.

En general los robots autónomos son dispositivos compuestos de sensores que reciben datos de entrada y que pueden estar conectados a una estación de

control. Ésta, al recibir la información de entrada, determina la acción a realizar por parte del robot (Ollero, 1995).

1.1 Motivación del trabajo

El uso de robots móviles se destina a aplicaciones en las que se realizan tareas monótonas, molestas, arriesgadas o imposibles para el trabajador humano, como puede ser el transporte de material peligroso, las excavaciones mineras, la limpieza industrial, la inspección de plantas nucleares, el rescate en zonas de desastre, la búsqueda en ambientes agresivos, la exploración espacial, entre otras. Otro grupo de aplicaciones es donde este tipo de robots complementa la actuación del operador, lo componen las labores de vigilancia, inspección, asistencia a personas incapacitadas, etc. Asimismo en aplicaciones a distancia, donde existe un retraso sensible o posibilidad de fallo en las comunicaciones, también resulta útil el uso de vehículos con cierto grado de autonomía (Segovia et al. 2002)

De acuerdo con (Salichs y Moreno, 2000) el robot móvil autónomo es aquel capaz de pasar la prueba máxima de navegación, que consiste en situar al robot en un ambiente desconocido complejo y dinámico, tras una breve exploración con sus sensores el robot debe ser capaz de llegar a un punto seleccionado, buscando minimizar la función de costo (tiempo, energía, etc.) Es por esta razón que el presente trabajo plantea la solución a dicha prueba utilizando una cámara de video como herramienta de sensado para el robot. Además se busca aprovechar las herramientas existentes para facilitar la tarea de captura, procesamiento y reconocimiento de imágenes, ofreciendo una opción de bajo costo y fácil implementación.

Los robots están revolucionando los procedimientos que se emplean en la agricultura, la minería, el transporte, entre muchas otras áreas, actualmente el desarrollo y avance de áreas estratégicas, como las comentadas, se encuentra sumamente ligado a la evolución de los robots móviles, de ahí la importancia de buscar nuevas técnicas y herramientas para optimizar el desempeño de dichos sistemas.

1.2 Planteamiento del Problema

El problema de la localización y control de robots móviles por medio de la retroalimentación visual y el reconocimiento de patrones ha sido abordado y estudiado bajo la óptica de contextos distintos, sin embargo, en la mayor parte de los casos, los conocimientos de programación así como de procesamiento digital de imágenes requeridos para la implementación de los sistemas suelen ser considerables, dificultando la experimentación, investigación y desarrollo de aplicaciones que utilicen esta tecnología.

1.3 Objetivo

El presente trabajo tiene por objetivo el desarrollo de un sistema sencillo que permita controlar un robot móvil para llegar a un punto definido dentro de un espacio de trabajo a través de retroalimentación visual y con base en la navegación por campos potenciales artificiales (Figura 1.1).

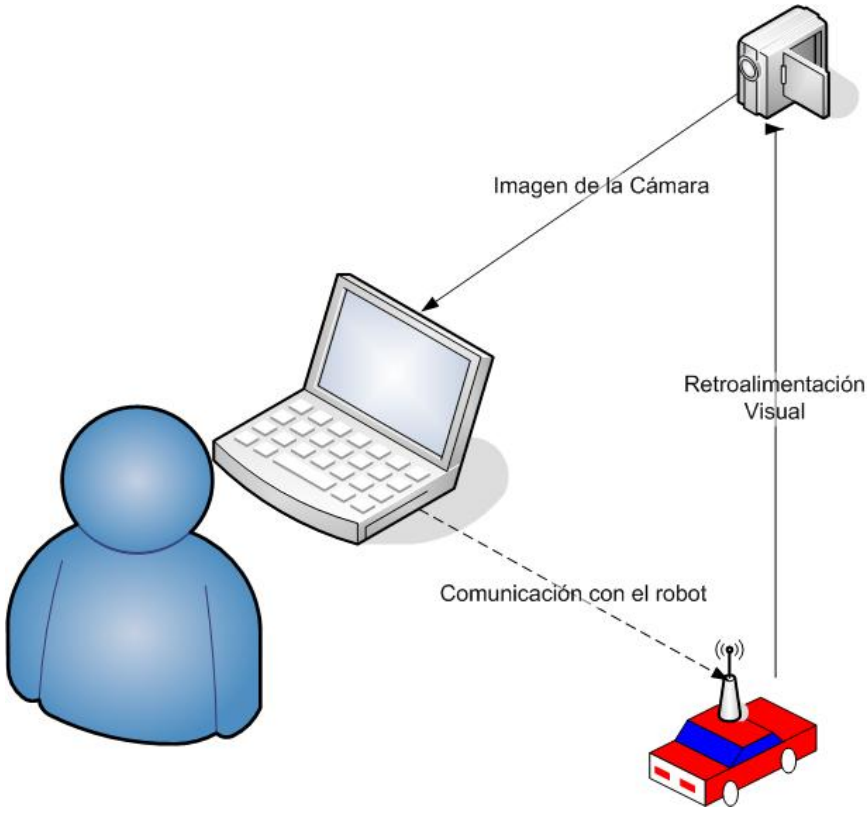


Figura 1.1 – Sistema planteado

1.4 Contribuciones

Las principales contribuciones de este trabajo son:

- Implementación de un sistema de parametrización para el control de un robot móvil, basado en retroalimentación visual, utilizando la plataforma reacTIVision y LabVIEW.

- Implementación del algoritmo para la respuesta de un robot móvil de acuerdo con una retroalimentación visual.

- Comprobación del modelo de campos potenciales artificiales para la evasión de obstáculos en la trayectoria y el seguimiento de objetivos no fijos.

- Sentar la primera etapa de desarrollo del proyecto de robótica móvil para el futuro desarrollo de robots cooperativos y manipuladores móviles.

1.5 Organización

La tesis se encuentra dividida en ocho capítulos, tres apéndices y un apartado final con referencias bibliográficas y de Internet. El primer capítulo expone el marco teórico para el desarrollo de la tesis, plantea el problema, el objetivo y las contribuciones del trabajo.

En el segundo capítulo titulado “El robot móvil” se exponen distintos tipos de robots móviles y se genera un contexto para la justificación y elaboración de un prototipo funcional.

El tercer capítulo aborda la visión por computadora como elemento de retroalimentación, se define la herramienta reacTIVision como parte del sistema, y se describen los elementos necesarios para la operación.

En el capítulo cuatro se desarrolla una interfase de computadora para la interacción de los distintos elementos del sistema, se detallan los mecanismos para la implementación con base en C#.

El capítulo cinco titulado “Navegación y planeación de trayectorias del robot móvil” introduce los conceptos fundamentales que definen la realización de tareas de navegación y planeación por parte de los robots móviles, además de abordar el método de campos potenciales artificiales.

En el sexto capitulo se realiza una breve introducción al control y al modelado cinemático del robot móvil para la configuración del prototipo.

El capítulo siete plantea varias aplicaciones prácticas para el sistema desarrollado, y analiza los resultados experimentales de dichas pruebas.

Por último el capítulo ocho destaca los aspectos y aportaciones más relevantes de la tesis, a la vez que plantea futuras líneas de investigación derivadas del trabajo realizado.

Capítulo 2. El robot móvil

2. El robot móvil

Es necesario definir los tres elementos principales del sistema, el primero en ser abordado es el robot móvil.

2.1 Clasificación de los robots móviles

Una de las distintas clasificaciones que existe en robótica móvil se realiza con base en el sistema de locomoción que utilizan los robots para lograr su desplazamiento, a continuación se comentan las características más significativas de los sistemas más comunes:

2.1.1 Vehículos con ruedas

Los vehículos con ruedas son la solución más simple y eficiente para conseguir la movilidad en terrenos suficientemente duros y no deslizantes. Existen distintos tipos de configuración.

2.1.1.1 Ackerman

Es el sistema utilizado en vehículos de cuatro ruedas convencionales. De hecho, los vehículos robóticos para exteriores resultan normalmente de la modificación de vehículos convencionales tales como automóviles o incluso vehículos más pesados. En este sistema la rueda delantera interior gira un ángulo ligeramente superior al exterior para eliminar el deslizamiento. Las prolongaciones de los ejes de las dos ruedas delanteras intersectan en un punto sobre la prolongación del eje de las ruedas traseras. El lugar de los puntos trazados sobre el suelo por los centros de los neumáticos son circunferencias concéntricas con centro en el eje de rotación. Si no se tienen en cuenta las fuerzas centrífugas, los vectores de velocidad instantánea son tangentes a estas curvas.

El mayor problema de la locomoción Ackerman es la limitación en la maniobrabilidad.

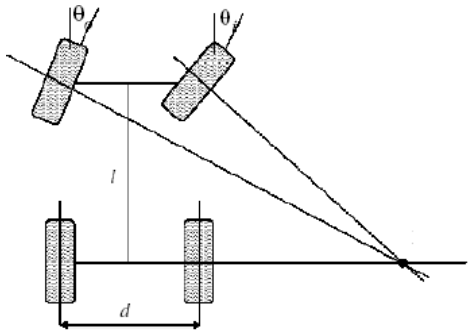


Figura 2.1 – Sistema Ackerman

2.1.1.2 Triciclo Clásico

En este sistema la rueda delantera sirve tanto para la tracción como para el direccionamiento. El eje trasero, con dos ruedas laterales, es pasivo y sus ruedas se mueven libremente. La maniobrabilidad es mayor que en la configuración anterior, pero puede presentar problemas de estabilidad en terrenos difíciles. El centro de gravedad tiende a desplazarse cuando el vehículo se desplaza por una pendiente, causando la perdida de tracción. Debido a su simplicidad, es bastante frecuente en vehículos robóticos para interiores y exteriores pavimentados.

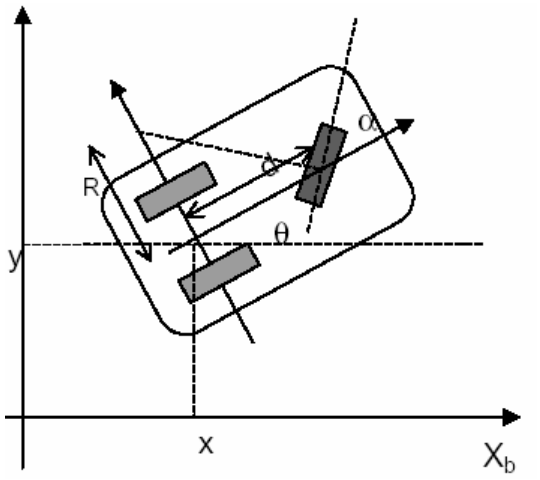


Figura 2.2 – Locomoción de triciclo

2.1.1.3 Direccionamiento Diferencial

El direccionamiento viene dado por la diferencia de velocidades de las ruedas laterales. La tracción se consigue también con estas mismas ruedas. Adicionalmente, existe una o más ruedas para soporte. Esta configuración es la más frecuente en robots para interiores.

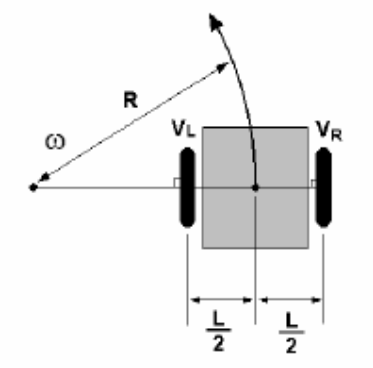


Figura 2.2 – Locomoción diferencial

2.1.1.4 Skid Steer

Se disponen varias ruedas de cada lado del vehículo que actúan de forma simultánea. El movimiento es el resultado de combinar las velocidades de las ruedas de la izquierda con las de la derecha.

Figura 2.3 – Robot "Terregator" con locomoción Skid Steer de la Universidad de Carnegie Mellon

2.1.1.5 Pistas de deslizamiento

Son vehículos tipo oruga en los que tanto la impulsión como el direccionamiento se consiguen mediante pistas de deslizamiento. Puede considerarse funcionalmente análogo a Skid Steer. De forma más precisa, las pistas actúan de forma análoga a ruedas de gran diámetro. La locomoción mediante pistas de deslizamiento es útil en navegación de campo o en terrenos irregulares. En este caso, la impulsión esta menos limitada por el deslizamiento y la resistencia al desgaste es mayor.

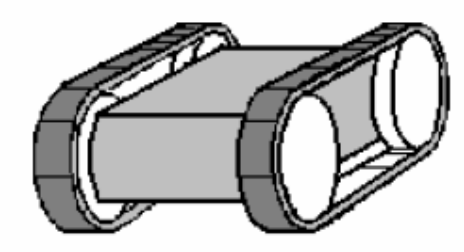


Figura 2.4 – Robot con pistas de deslizamiento

2.1.1.6 Ruedas Síncronas

Consiste en la actuación simultánea de todas las ruedas, que giran de forma síncrona. La transmisión se consigue mediante coronas de engranajes o con correas concéntricas.

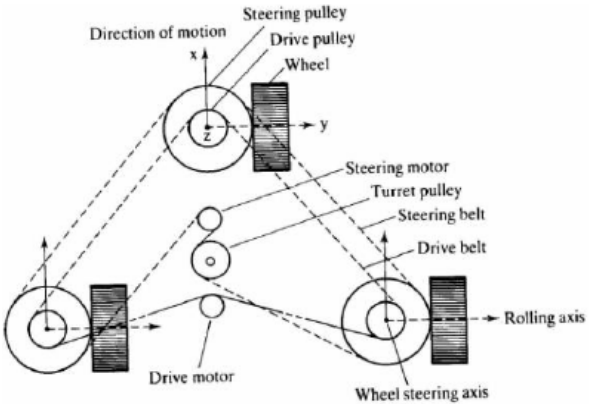


Figura 2.5 – Locomoción con ruedas síncronas

2.1.1.7 Ruedas Omnidireccionales

Esta configuración consiste en el empleo de las denominadas ruedas suecas u omnidireccionales, las cuales permiten conseguir un movimiento en distintos ejes simultáneos del vehículo.

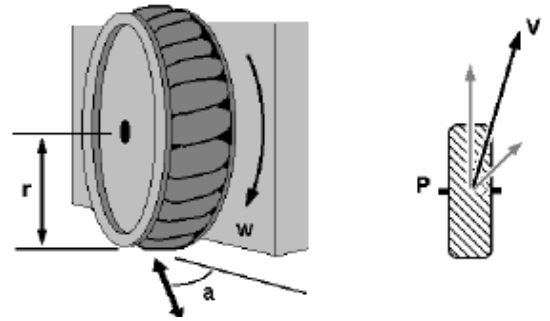


Figura 2.6 – Ruedas omnidireccionales

2.1.2 Locomoción mediante patas

Permiten aislar el cuerpo del robot del terreno empleando únicamente puntos discretos de soporte. Es posible adaptar el polígono de soporte para mantener la estabilidad y pasar sobre obstáculos, ideal para terrenos muy irregulares.

Asimismo, mediante patas, es posible conseguir la omnidireccionalidad, y el deslizamiento en la locomoción es mucho menor.

En los robots con patas la complejidad de los mecanismos necesarios es mayor, así como el consumo de energía. En principio, los problemas de planificación y control son más complejos que en los vehículos con ruedas.

La configuración más común es la de seis patas, sin embargo, la tendencia en la evolución es hacia los robots bípedos, que emulan el sistema de locomoción del hombre, en esta clasificación existen aplicaciones muy interesantes como los robots de patas utilizados para escalar y trepar, o los robots de exploración.

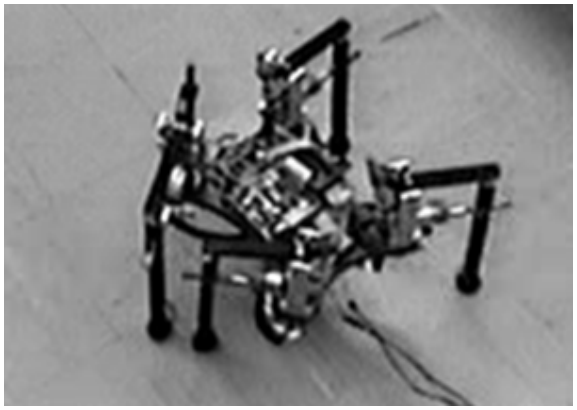


Figura 2.7 – Robot con patas de la Facultad de ingeniería de la UNAM

2.1.3 Configuraciones Articuladas

Las configuraciones articuladas son de interés para terrenos difíciles a los que debe adaptarse el cuerpo del robot. La solución más simple consiste en articular dos o más módulos con locomoción mediante ruedas. Las configuraciones articuladas con gran cantidad de eslabones son apropiadas para caminos estrechos. La seguridad de funcionamiento puede ser mayor debido a la redundancia de su estructura, ofreciendo la oportunidad de intercambiar segmentos.

Figura 2.8 – Robot "MAKRO" de configuración articulada de GMD (Alemania)

2.1.4 Robots submarinos y aéreos

El interés de tales aplicaciones surge como necesidad para tareas tales como la inspección, recogida de datos o mantenimiento de instalaciones en entornos naturales en los que el acceso al hombre resulta muy difícil, o incluso

imposible. Estos vehículos son el resultado de la evolución de vehículos normalmente piloteados por el hombre.

2.2 Configuración del robot

Se opta por una configuración diferencial tras analizar los distintos tipos de sistemas de locomoción. Se elige porque el diseño es simple, fácil de implementar y las restricciones para el desplazamiento son mínimas, además de que el sistema se desarrolla para un ambiente de trabajo que se supone no irregular.

Valorando los elementos disponibles en el Departamento de Mecatrónica de la Facultad de Ingeniería y considerando este trabajo como la primera etapa de un proyecto más amplio, se estima prudente el uso de la plataforma Lego Mindstorms NXT para la construcción del robot móvil prototipo, pues ofrece la ventaja de ser reconfigurable y ser compatible con LabVIEW, software del cual el departamento tiene licencia.

2.3 Lego Mindstorms NXT

La línea Lego Mindstorms nació a partir de un acuerdo entre Lego y el MIT. Según este trato, Lego financiaría investigaciones del grupo de epistemología y aprendizaje del MIT sobre cómo aprenden los niños y a cambio obtendría nuevas ideas para sus productos, que podría lanzar al mercado sin tener que pagar regalías al MIT.

De esta sociedad surge Lego Mindstorms, un sistema integrado de robótica con partes electromecánicas controladas por computadora desarrollado por Lego, el sistema posee elementos básicos de las teorías robóticas como la unión de piezas y la programación de acciones en forma interactiva.

El sistema básico de Lego Mindstorms incluye (figura 2.9) 1 - *Brick* (ladrillo) Inteligente, 2 - Sensor de Contacto, 3 - Sensor de Sonido, 4 - Sensor de Luz, 5

- Sensor Ultrasónico, 6 - Tres servomotores. Además una de las mayores ventajas que ofrece el *brick* Inteligente es la interacción con distintos Lenguajes de programación siendo los principales C y LabVIEW, este último, utilizado para el desarrollo de este trabajo.

Figura 2.9 - Lego Mindstorms NXT (The Lego Group 2006)

El sistema del ladrillo inteligente está desarrollado con el microcontrolador AT91SAM7S256 de Atmel, más información técnica puede ser consultada en el apéndice C.

2.4 LabVIEW

LabVIEW es una plataforma de desarrollo de programas gráficos creada por National Instruments en 1986 como herramienta para la implementación de pruebas virtuales, control de sistemas y diseño en distintas áreas.

Bajo el lema “La potencia está en el software” LabVIEW se ha convertido en una de las herramientas más poderosas para la adquisición de datos, el control de instrumentos, la automatización industrial, el diseño de prototipos de control, así como el diseño embebido, en gran parte debido a la facilidad del software para trabajar con distintas entradas y salidas de diversos sistemas para la adquisición y manipulación de datos.

Los programas de LabVIEW se componen de un Panel Frontal y un Diagrama de Bloques. El Panel Frontal es la interfase con el usuario, en la cual se definen los controles e indicadores que se muestran en pantalla. El Diagrama de Bloques es el programa funcional, aquí se colocan instrumentos virtuales que realizan una determinada función y se interconectan para realizar una tarea más compleja aun.

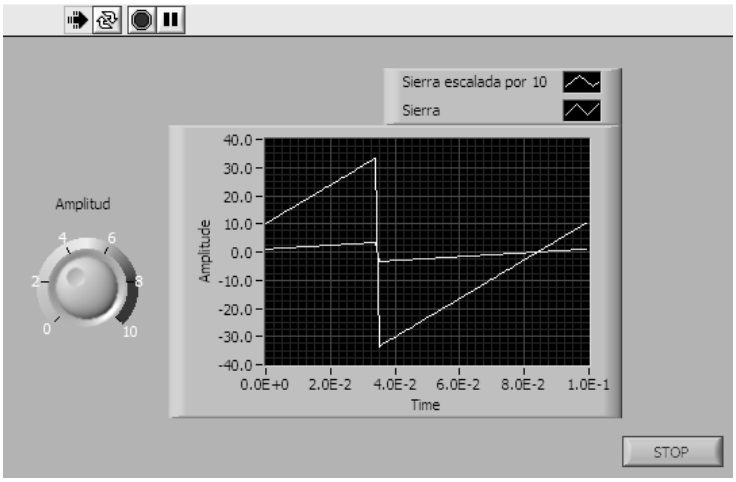


Figura 2.10 - Panel de Control

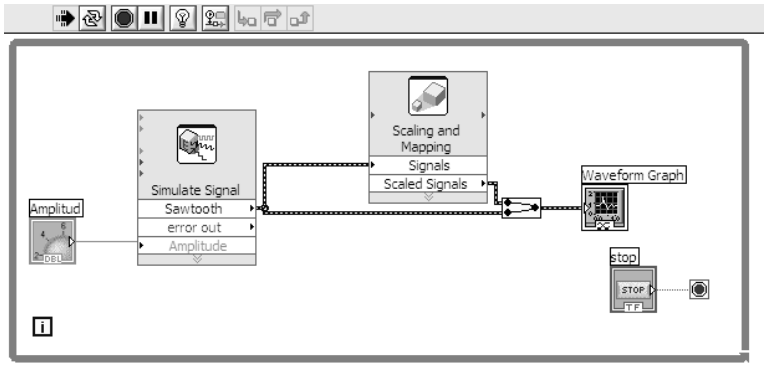


Figura 2.11 - Diagrama de Bloques

La programación de LabVIEW se basa funciones y en bloques de programación conocidos como Instrumentos Virtuales. Las estructuras con las que se puede programar son similares a las existentes en los demás lenguajes (*if, while, case, for*, etc.)

2.4.1 Instrumento Virtual

El conjunto de funciones en LabVIEW en un bloque de programación es llamado Instrumento Virtual (o VI por sus siglas en inglés), porque su apariencia y operación imita a instrumentos físicos como pueden ser osciloscopios y multímetros.

Los Instrumentos virtuales son el equivalente a las funciones compuestas en otros lenguajes de programación, se encuentran ligados regularmente con entradas y salidas.

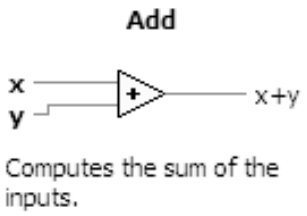


Figura 2.12 – Función Add de LabVIEW

Los Instrumentos Virtuales pueden ser obtenidos de alguna librería de LabVIEW o creados por el usuario como el conjunto de distintas funciones, declaraciones, instrumentos o estructuras de programación.

LabVIEW permite importar librerías de funciones de propósito específico para ciertos componentes o periféricos, tal es el caso del Toolkit para Lego Mindstorms NXT que agrega dos nuevas ventanas de componentes para la programación y la adquisición de datos a través del ladrillo inteligente.

La primera librería NXT Toolkit contiene estructuras e instrumentos para programar el ladrillo en lazo abierto, el programa creado con estos instrumentos es descargado al ladrillo inteligente y las salidas no pueden ser monitoreadas.

Figura 2.13 – Librería NXTToolkit

La segunda librería incluye instrumentos para la comunicación directa entre LabVIEW y el ladrillo inteligente, esta librería es bastante útil para el monitoreo, el control en tiempo real.

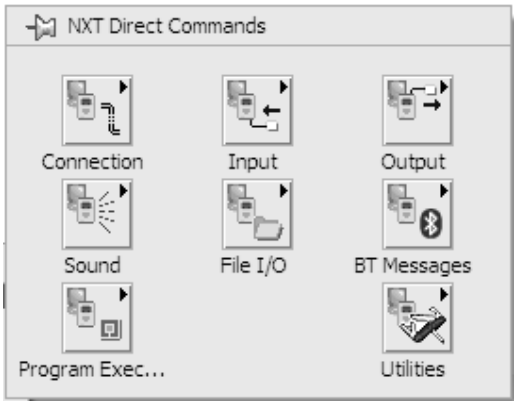


Figura 2.14 – Librería NXT Direct Commands

El Toolkit de Lego mindstorms para LabVIEW se descarga de la página de Nacional Instruments.

2.5 Robot prototipo

Como se discutió previamente el prototipo se construye con base en una arquitectura diferencial, esta configuración requiere dos llantas de tracción y en

este caso de un soporte adicional para la estabilidad, con el soporte extra y las llantas existentes se forma un triángulo, el centro de gravedad total del móvil debe encontrarse dentro de este triángulo para mantener la estabilidad y el equilibrio. Si se adhiere otro soporte se debe agregar suspensión a las otras llantas a fin de mantener un contacto permanente sobre los cuatro puntos de la base. En el prototipo de trabajo se agrega un solo soporte, una rueda loca de tipo castor.

Figura 2.15 – Prototipo construido con Mindstorms NXT

2.5.1 Comunicación inalámbrica del robot prototipo vía Bluetooth

Para comunicar inalámbricamente el robot prototipo con el sistema de control generado por la interfase se elige el protocolo de comunicación Bluetooth, pues en el departamento se cuenta con los elementos necesarios para ocuparlo, además de ser compatible con el Lego Mindstorms NXT.

La especificación de Bluetooth define un canal de comunicación, la frecuencia de radio con la que trabaja está en el rango de 2.4 a 2.48 GHz con posibilidad de transmitir en *Full Duplex* con un máximo de 1600 bits por segundo. La potencia de salida para transmitir a una distancia máxima de 10 metros es de 1[mW], mientras que la versión de largo alcance transmite entre 100[mW] y 1[W].

Por medio de LabVIEW se puede realizar un enlace Bluetooth entre la interfase y el robot, como se abordará más adelante en este trabajo.

Capítulo 3. Retroalimentación con video

3. Retroalimentación con video

La visión es la facultad por la cual se percibe el mundo exterior. Muchos organismos simples tienen receptores luminosos capaces de reaccionar ante determinados movimientos y sombras, pero la verdadera visión supone la formación de imágenes en el cerebro que permiten la abstracción del entorno y la posibilidad de interacción con el mismo. En el presente trabajo se incorpora visión al robot móvil por medio de una cámara de video, que es el elemento tecnológico más cercano a la visión que tenemos los seres humanos.

Tradicionalmente el reconocimiento y procesamiento de imágenes por computadora se asocia con un proceso difícil que consume una buena cantidad de los recursos del sistema, además de requerir equipos especializados de alto costo. Sin embargo, en la actualidad existen herramientas que facilitan dicha tarea, reduciendo el gasto de recursos computacionales y el costo en equipo de video, una de estas herramientas es la plataforma reacTIVision descrita a continuación.

3.1 ReacTIVision

ReacTIVision es una plataforma de visión por computadora para la interacción con elementos tangibles.

ReacTIVision es una plataforma de visión por computadora de código abierto principalmente diseñada para la construcción de interfases tangibles bidimensionales con el usuario.

El ReacTIVision fue desarrollado como elemento sensor principal para la reacTable (Jordà et al. 2005), un instrumento musical electroacústico tangible. Utiliza marcadores visuales espacialmente diseñados (símbolos fiducial) que pueden ser asociados a objetos físicos. Los marcadores son reconocidos y localizados por un algoritmo de visión por computadora optimizado para el diseño particular de los marcadores, mejorando la velocidad promedio y la

robustez del proceso de reconocimiento. Estos símbolos marcadores fiducial permiten distinguir cientos de identidades únicas, añadiendo la posibilidad del preciso cálculo del ángulo de rotación del marcador en un plano de 2 dimensiones.

ReacTIVision y sus componentes están estructurados por la combinación de distintas licencias de software libre como son GPL, LGPL, BSD y puede ser obtenido como ejecutable o como código fuente abierto del sitio SourceForge.

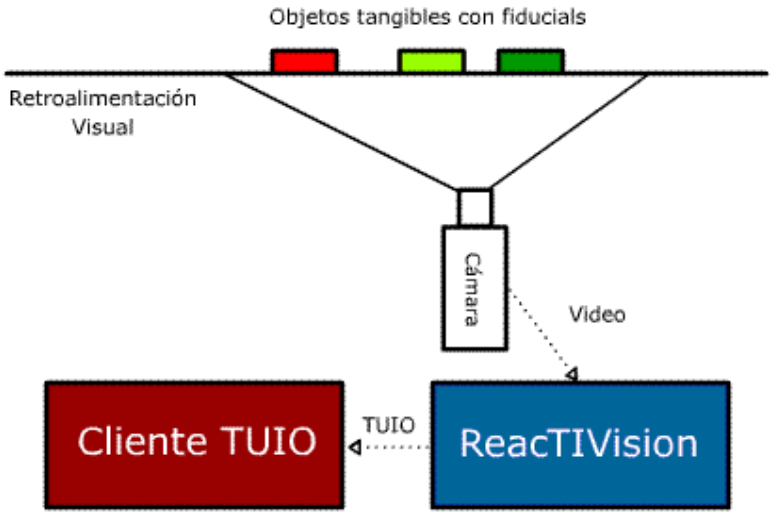


Figura 3.1 – Aplicación de ReacTIVision

3.1.1 Arquitectura

ReacTIVision esta diseñado como una aplicación de construcción fragmentada y no como un objeto de una librería de código. Cada componente del sistema esta implementado como un proceso ejecutable independiente. La comunicación entre los componentes se logra utilizando un protocolo publicado (TUIO). Este diseño simplifica el uso del mismo facilitando la integración con distintos ambientes de programación. La arquitectura también permite la ejecución de los componentes de la plataforma en distintas máquinas, lo cual puede se bastante útil en ciertos contextos de instalación o aplicaciones de red.

3.1.2 Componente de Reconocimiento

La aplicación reacTIVision adquiere la imagen de una cámara, busca símbolos fiducial en el flujo de video -cuadro por cuadro- y manda los datos sobre los símbolos encontrados a través de un *socket* de red hacia una aplicación receptora. La aplicación está diseñada de forma modular, haciendo fácil la inclusión de nuevas imágenes para el reconocimiento y componentes para el procesamiento.

El código está programado en C++ Portable combinado con otros componentes para la adquisición. Este software también se encuentra de manera independiente como código abierto con el nombre PortVideo.

3.1.3 Componente de Comunicación

ReacTIVision define su propio protocolo de comunicación TUIO (Kaltenbrunner et al. 2005) que fue específicamente diseñado para las necesidades de las interfases tangibles basadas en una mesa: codificando y transmitiendo los atributos de los artefactos tangibles que se encuentran sobre la superficie de la misma.

Para poder ofrecer una mejor y más confiable comunicación con aplicaciones de clientes locales y remotos las capas del protocolo redundan los mensajes sobre una estructura de transporte UDP. TUIO define un conjunto abierto de instrucciones para el control. Estos mensajes constantemente transmiten la presencia, posición y ángulo de los símbolos reconocidos, así como sus parámetros derivados. En el cliente, estos mensajes redundantes son decodificados a funciones genéricas como agregar símbolo, actualizar símbolo y remover símbolo, cada una correspondiente al evento físico aplicado sobre cada uno de los objetos.

3.1.4 Componente Cliente

Para facilitar el desarrollo de interfases, el cliente de reacTIVision puede ser programado en distintos lenguajes como son C++, C#, Java, SuperColloder, Max/MSP y Flash. El cliente se encarga de decodificar los mensajes para trasladarlos a alguna aplicación gráfica o simplemente imprimirlos en la consola.

3.1.5 Símbolos Fiducial

La geometría de los símbolos fiducial ha sido rediseñada por medio de algoritmos genéticos para ser reconocida, hasta llegar a la amiba, reduciendo el tamaño de los marcadores y haciendo más eficientes los algoritmos de seguimiento.

Para el reconocimiento de los símbolos fiducial la imagen de origen es convertida a blanco y negro con un algoritmo adaptativo de mapeo. Esta imagen es segmentada en regiones gráficas adyacentes de blanco y negro alternante, dentro de estas se buscan tres estructuras únicas contenidas en los símbolos fiducial. Finalmente las tres estructuras identificadas son comparadas con un diccionario que le asigna un ID numérico en particular de acuerdo con la semejanza obtenida.

3.1.6 Fiducial Amiba

La compacta geometría del fiducial amiba fue obtenida a través de algoritmos genéticos. Al utilizar también un algoritmo genético para el reconocimiento se optimiza la búsqueda del fiducial utilizando un conjunto de funciones para el reconocimiento de figuras, formas, tamaños, localización de centroides y orientación. ReacTIVision funciona con un conjunto de 90 símbolos distintos escogidos de 128 conjuntos de tres patrones. Los patrones se conforman de nodos hoja, la limitación en la cantidad de los mismos es para favorecer la

exclusión en el reconocimiento, favoreciendo la robustez del algoritmo y evitando la detección de falsos positivos debido al ruido.

La posición del símbolo es calculada como el centroide de todos los nodos hoja (pequeñas manchas circulares), los cuales proveen mayor precisión en el reconocimiento. La orientación del símbolo es calculada como el vector que va del centroide del marcador hacia el centroide de los nodos hoja obscuros distribuidos en la parte superior del símbolo.

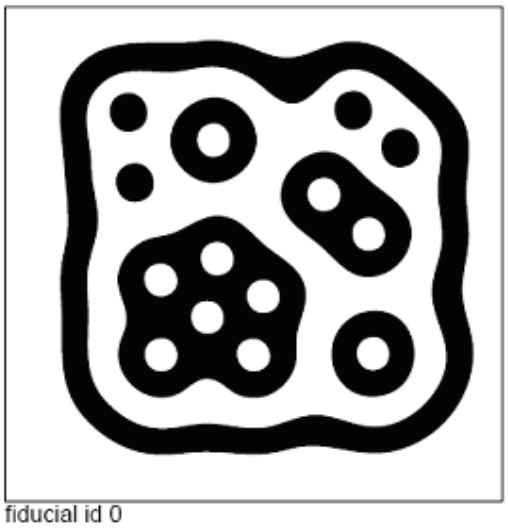


Figura 3.2 – Fiducial amiba

3.1.7 Cámara y Lentes

Gran parte de la tarea de reconocimiento está obviamente ligada a la calidad de la imagen adquirida, para un correcto funcionamiento de la plataforma reacTIVIsion es necesario que la cámara se encuentre bien enfocada sobre la superficie de trabajo, además de tener un tiempo de exposición adecuado a fin de evitar una imagen de captura borrosa o desvanecida, de igual forma la superficie de trabajo debe encontrarse bien iluminada.

ReacTIVision en general trabaja con cualquier tipo de cámara y lente. Para una buena calidad es conveniente el uso de cámara digital o webcam, USB o

FireWire, con resolución mínima de 640x480 a 30 cuadros por segundo. Los lentes de ángulo amplio u ojo de pescado también pueden ser utilizados, sin embargo, es conveniente una calibración por software de la plataforma para obtener un resultado adecuado que elimine las distorsiones y se adapte al patrón de reconocimiento deseado.

3.2 Interpretación de la Información

Es importante la definición de elementos que puedan ser interpretados y analizados por la interfase, con la finalidad de dar un significado a los elementos visualizados por la cámara. En este caso se ocuparan distintos patrones para cada uno de los elementos del medio, siendo el patrón fiducial 0 el correspondiente al robot, el patrón fiducial 1 correspondiente a la meta y cualquier otro patrón fiducial será reconocido como un obstáculo.

Para este trabajo se utilizó la HandyCam Sony modelo DCR-HC28 conectada a la computadora con cable FireWire y con resolución NTSC de 720x480. La pérdida de linealidad asociada con el lente de la cámara se supone despreciable para fines prácticos.

Los valores de posición en X y Y son relativos a la posición en la pantalla de captura y se encuentran entre los valores (0,0) y (1,1). Se aprovecha esta característica para que los valores adquiridos por reacTIVision puedan ser independientes de la lejanía o cercanía de la cámara con el campo de trabajo, en futuros desarrollos con una cámara fija se puede asignar un factor de escala a los valores de posición.

El sistema de referencia utilizado en este trabajo en función de reacTIVision localiza la coordenada (0, 0) en la esquina inferior izquierda de la pantalla de captura y la coordenada (1, 1 x la relación proporcional de la pantalla de captura) en la esquina superior derecha de la pantalla de captura, los ángulos

son medidos de la manera convencional, positivos en el sentido contrario a las manecillas de reloj.

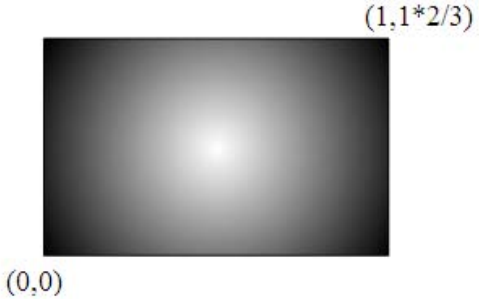


Figura 3.3 – Pantalla de captura de 720x480

Capítulo 4. Interfase

4. Interfase

Normalmente, las tareas en robótica móvil no son fáciles de programar, pues el entorno del robot suele ser desconocido, contrario a lo que ocurre con los manipuladores, por poner un ejemplo, además la operación a distancia agrega un grado extra de dificultad.

Por todo ello, la programación de vehículos robóticos suele realizarse con un juego de instrucciones relativamente pequeño en las que se involucran movimientos en el plano con una relativamente elevada interacción con el entorno. De esta forma, se combinan órdenes básicas a la dirección o velocidad del vehículo con otras cuya ejecución hace necesario el sistema de percepción.

Desde un punto de vista funcional, el sistema de programación o interfase de un robot forma parte de un sistema informático que debe realizar cierto número de tareas tales como: planeación, control automático en tiempo real, comunicación con el operador, comunicación con periféricos y comunicación con otros equipos.

4.1 Arquitectura

Para poder llevar a cabo la retroalimentación visual de la posición del robot es necesario ligar los valores adquiridos por la cámara con las salidas de los motores que controlan la posición del robot, en este caso los valores adquiridos por la cámara son obtenidos a través del reacTIVision y el control de las salidas se realiza a través de LabVIEW.

Los mensajes de reacTIVision deben ser decodificados por un cliente, éste a su vez debe analizar y procesar los datos para enviarlos hacia LabVIEW que se encargará de controlar los motores del robot.

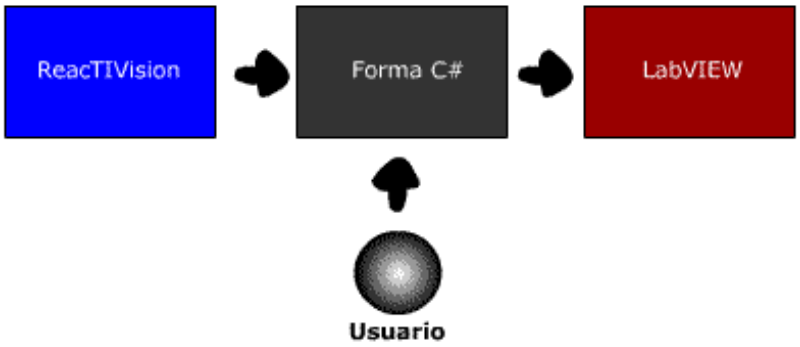


Figura 4.1 – Diagrama de la Interfase

El cliente se desarrolló con Visual C#, debido a la facilidad del entorno de programación para manejar salidas a través de puertos, pensando en el desarrollo a futuro del proyecto. Los datos decodificados por el cliente se ligan con una forma tipo Windows, que hace más familiar la interacción con el usuario, dicha forma controla por medio de un control ActiveX un Instrumento Virtual (VI) de LabVIEW que regula la respuesta de los motores, la forma además registra los datos, que pueden ser enviados a Excel para su análisis. Los programas desarrollados se encuentran en los apéndices A y B.

4.2 Cliente TUIO

El Cliente TUIO (encargado de decodificar los mensajes de reacTIVision) está programado en C# orientado a objetos, y se basa en un mecanismo de funciones que notifica cuando se agregan, cuando se mueven y cuando se quitan objetos fiducial al campo de visión.

La aplicación cliente TUIO fue desarrollada a partir de los class de código abierto provistos con reacTIVision versión 1.4 (pre1). En C#, un class es un componente de proyecto que contiene elementos y funciones modulares, es similar a las librerías o archivos “.h” del lenguaje C.

En general, la lógica de la aplicación se tiene que implementar con la interfase class *TuioListener*, que define varios métodos como son *addTuioObj()*,

updateTuioObj() y *removeTuioObj()*. Estos métodos son llamados por la class *TuioClient* la cual deduce los eventos del flujo continuo de información recibida por la aplicación sensora de reacTIVision. La class *TuioClient* tiene que ser inicializada con el método *connect()* al inicio de la sesión del programa. Es necesario también registrar todos los *TuioListener* que necesitan ser notificados por el *TuioClient* utilizando el método *addTuioListener()*. El *TuioClient* opera en flujo independiente en segundo plano hasta que es terminado con el método *disconnect().*

La comunicación con el *TuioClient* se establece a través de un puerto de comunicación, este puerto puede ser interno para una aplicación local, sin embargo, se puede desarrollar un cliente remoto a través de una red local. En este trabajo se utiliza el puerto interno 3333.

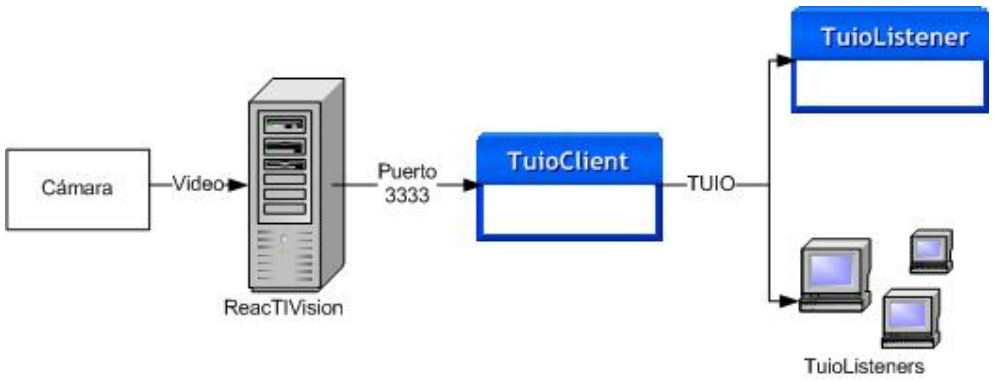


Figura 4.2 – Comunicación con reacTIVision

4.3 Comunicación del *TuioClient* con la forma TuioTesis en C#

Como ya se explicó previamente es necesario ligar el TuioClient con una interfase hacia el usuario, en este caso se realiza - la interfase - con una forma de Windows de nombre TuioTesis, dicha forma permite ligar los valores obtenidos a través del cliente con etiquetas de texto que permiten al usuario el

monitoreo gráfico de los valores, así como la sincronización de eventos asociados a dichos cambios,

Para iniciar el desarrollo del cliente es necesario iniciar un proyecto, el realizado en este trabajo tiene por nombre TUIO_TESIS, es indispensable agregar los elementos necesarios y provistos por reacTIVision para la creación de un cliente local TUIO, los cuales se muestran a continuación:

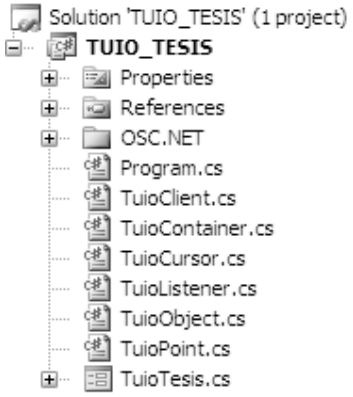


Figura 4.3 – Elementos necesarios para un Cliente

De estos elementos el class Program.cs es muy importante, pues en él se aloja la función principal del proyecto, en ésta se establece la liga con reacTIVsion, se asigna el puerto 3333 como cliente local para la comunicación y se inicia la forma TuioTesis ligada al puerto establecido.

Figura 4.4 – Program.cs

La forma tiene que ligarse al cliente, por lo que es necesario declarar una variable cliente *TuioClient* en la forma y llamar al proceso *connect (port)*. Una

vez realizado, se incluye un *TuioListener* en la forma y finalmente se tiene la posibilidad de utilizar las funciones *addTuioObj()*, *updateTuioObj()* y *removeTuioObj()*.

Sin embargo, como se expuso previamente, el *TuioClient* trabaja en un flujo independiente en segundo plano, el flujo es similar a un *loop* infinito, para poder ligar un control de forma de Windows (en este caso el texto de una etiqueta) al flujo sin generar un error, es necesario llamar a un método que se encarga de invocar a la función del control, a través de un representante, es decir, no interrumpe el flujo, sino que toma los datos del mismo a través de una función que obtiene el valor y lo representa en la etiqueta de la forma.

4.4 ActiveX

ActiveX esta basado en la tecnología *Object Linking and Embedding* (OLE) que es un sistema de objeto y protocolo desarrollado por Microsoft.

OLE permite a un editor encargar a otro la elaboración de parte de un documento y posteriormente volverlo a importar. Por ejemplo, un sistema de publicación de escritorio puede enviar un poco de texto a un procesador de textos o una imagen a un editor de *bitmap* usando OLE. La ventaja principal de OLE, además de que el tamaño del archivo es menor, es la de poder crear un archivo principal. Se puede hacer una referencia a los datos de ese archivo, con lo que todo cambio posterior en el archivo principal se reflejará en el documento referenciado.

Su uso principal es el manejo de documentos compuestos (*compound documents*), pero también puede ser usado para transferir datos entre aplicaciones diferentes usando arrastrar y soltar y operaciones del portapapeles. El concepto de “incrustación” (*embedding*) es también de uso central en páginas web multimedia, las cuales tienden a contener videos, animaciones y archivos de música dentro del código HTML. Sin embargo, OLE usa una arquitectura denominada *fat client* (cliente pesado), la cuál significa

que el tipo de archivo o la aplicación que va a ser incrustada debe estar presente en la máquina en la cuál esta va a trabajar. Por ejemplo, si una hoja de cálculo de Excel está por ser procesada o incluso solo visualizada, debería haber una copia de Excel o un visor de Excel instalado en la máquina del usuario.

OLE 1.0, lanzado en 1990, fue la evolución de su antecesor Dynamic Data Exchange (DDE), concepto que Microsoft desarrolló en las primeras versiones de Windows. Mientras DDE fue restringido a transferir una limitada cantidad de información entre dos aplicaciones, OLE fue capaz de mantener enlaces activos entre dos documentos o incluso incrustar un tipo de documento dentro de otro.

Servidores y clientes OLE se comunican con las bibliotecas del sistema usando Tablas de Funciones Virtuales o *virtual function tables* o VTBLS. La VTBL es una estructura de funciones punteros que el sistema de biblioteca puede usar para comunicarse con el cliente o con el servidor. Las bibliotecas del servidor y el cliente OLESVR.dll y ALECLI.dll, fueron originalmente diseñadas para comunicarse entre ellas usando mensajes Windows WM_DDE_EXECUTE.

Posteriormente OLE 1.0 se convirtió en una arquitectura para componentes de software (*software componentry*) más conocida como COM y luego DCOM.

En 1996, Microsoft renombró la tecnología OLE 2.0 como ActiveX.

4.5 Comunicación de la forma TuioTesis en C# con un VI de LabVIEW vía ActiveX

Para controlar las salidas de los motores del robot Lego Mindstorms NXT se requiere establecer una liga de transferencia entre la forma TuioTesis de C# y el Instrumento Virtual (VI). La comunicación entre ambos componentes se realiza vía ActiveX.

En el proyecto TUIO_TESIS se incluye una referencia de componente de software al objeto COM de ActiveX hacia la librería de LabVIEW, para poder hacer uso de recursos compartidos entre la forma del proyecto y el Instrumento Virtual.

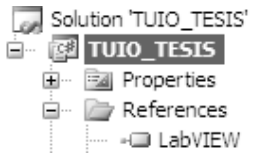


Figura 4.5 – Referencia COM a LabVIEW

En la inicialización de la forma se deben declarar variables de la aplicación LabVIEW y el Instrumento Virtual (LabVIEW.Application lv y LabVIEW.VirtualInstrument vi) ambas variables deben ser inicializadas y referenciadas a la ubicación del VI de la siguiente manera:

```
//se abre aplicación de LabVIEW
lv = new LabVIEW.ApplicationClass();

//se indica el camino al VI
string vipath = lv.ApplicationDirectory.ToString() + @"\tesis\controLego.vi";

//se vincula con el VI
vi = lv.GetVIReference(vipath, "", false, 0);

//se inicia el VI de manera asincrona
vi.Run(true);
```

El nexo entre los controles del panel y los valores de la forma puede ser establecido modificando las propiedades del *vi* declarado. Es importante correr el VI de manera asíncrona, pues de esta forma se pueden modificar los controles de la aplicación en tiempo real sin generar error.

4.6 Exportación del registro de datos a Excel vía ActiveX

Para poder tener el registro de los datos obtenidos durante una prueba con el sistema se agrega la posibilidad de guardar y exportar datos a Excel, para este fin se requiere establecer un vínculo entre la forma TuioTesis de C# y un libro de trabajo de Excel.

En el proyecto TUIO_TESIS se incluye una referencia de componente de software al objeto COM de ActiveX hacia la librería de control y operación de Excel, como se hizo previamente con LabVIEW.

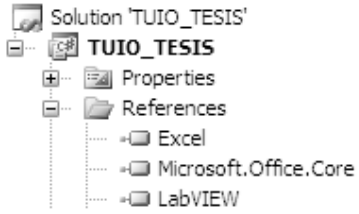


Figura 4.6 – Referencia COM a Excel

Una vez que la referencia ha sido añadida al proyecto es necesario crear una nueva aplicación (en la inicialización se declara una Excel.Application AplicacionExcel) y para poder usar la aplicación creada se realiza:

```
//se abre una nueva aplicación
AplicacionExcel = new Excel.ApplicationClass();

//se indica el camino hacia el archivo
string workbookPath = "c:Tesis.xls";
Excel.Workbook Librodetrabajo =

//función para abrir el libro de trabajo
AplicacionExcel.Workbooks.Open(workbookPath, 0, false, 5, "", "", false,
Excel.XlPlatform.xlWindows, "", true, false, 0, true, false, false);

//se selecciona la hoja en la que se va a escribir
Excel.Sheets Hojaexcel = Librodetrabajo.Worksheets;
string hojaactual = "Hoja1";
Excel.Worksheet Hojadetrabajo =
(Excel.Worksheet)Hojaexcel.get_Item(hojaactual);
```

```
//se llena con un arreglo de valores registrados
Hojadetrabajo.get_Range("A1", "B100").Value2 = registros;
```

Los valores registrados deben ser almacenados en el arreglo que se exporta a Excel, cabe también la posibilidad del registro en Excel en tiempo real, sin embargo, el gasto de recursos aumenta y se incrementa la posibilidad de errores.

4.7 Control de Lego Mindstorms NXT con LabVIEW

Para transmitir los valores de las variables de control hacia los motores se requiere de un Instrumento Virtual que ligue los datos provenientes de la forma TuioTesis con las salidas a motor del *brick* inteligente.

Previamente se explicó la forma de ligar la forma con un VI, los valores controlables del VI son los que se encuentran en el panel de control y como se expuso previamente LabVIEW tiene librerías para el control de Lego Mindstorms NXT.

Para establecer una conexión con el robot vía bluetooth para el control en tiempo real es necesario unir el VI “Find NXT”, que como su nombre lo indica busca la presencia de un robot NXT en la región de alcance, con el VI “Create NXT Object”, que crea un objeto para la manipulación y monitoreo en sincronización con la computadora.

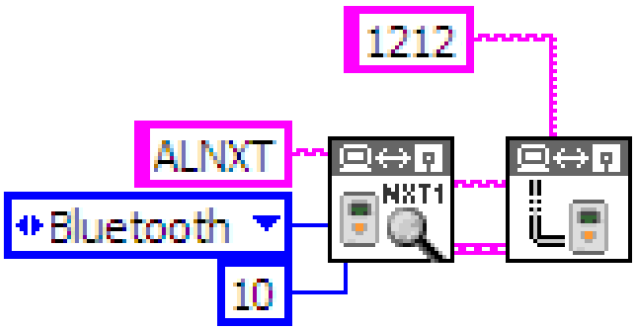


Figura 4.7 – Conexión Bluetooth con el NXT

Los argumentos necesarios son el tipo de conexión, el tiempo de búsqueda, el NXT particular que se busca y la contraseña para la conexión. Por otro lado para terminar con la conexión, sólo es necesario conectar el VI “Destroy NXT Object” que termina la conexión y destruye el objeto virtual creado.

Figura 4.8 – Desconexión Bluetooth con el NXT

Para controlar las salidas de los motores en tiempo real, se utiliza el VI “Motor Unlimited” que permite controlar el sentido, puerto y poder de la salida, así como el VI “Motor Stop” que para el motor seleccionado, y lo deja energizado o neutral.

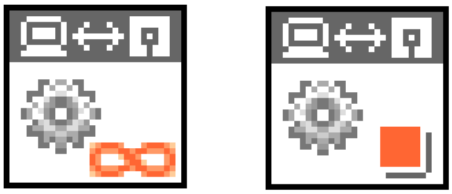


Figura 4.9 – Control del motor

Para conocer el valor de rotación o posición del motor se utiliza el VI “Get Output Values” que permite monitorear las salidas sensibles por el NXT.

Figura 4.8 – Monitoreo de salidas

Con estos elementos se puede realizar el control a distancia del robot Lego Mindstorms NXT.

Capítulo 5. Navegación y planeación de trayectorias del robot móvil

5. Navegación y planeación de trayectorias del robot móvil

La navegación se define como la metodología de planeación que permite guiar el curso de un robot móvil a través de un entorno con obstáculos para llegar a un punto objetivo. Existen diversos esquemas para la navegación y planeación, pero todos ellos poseen en común el afán por llevar el vehículo a su destino de forma segura y eficiente. La capacidad de reacción ante situaciones inesperadas debe ser la principal cualidad para desenvolverse, de modo eficaz, en entornos no estructurados.

Las principales tareas involucradas en la navegación de un robot móvil son la percepción del entorno a través de sus sensores, de modo que le permita crear una abstracción adecuada del mundo, la planificación de una trayectoria libre de obstáculos para alcanzar el punto destino seleccionado, y el guiado del vehículo a través de la referencia construida. De forma simultánea, el vehículo puede interaccionar con ciertos elementos del entorno.

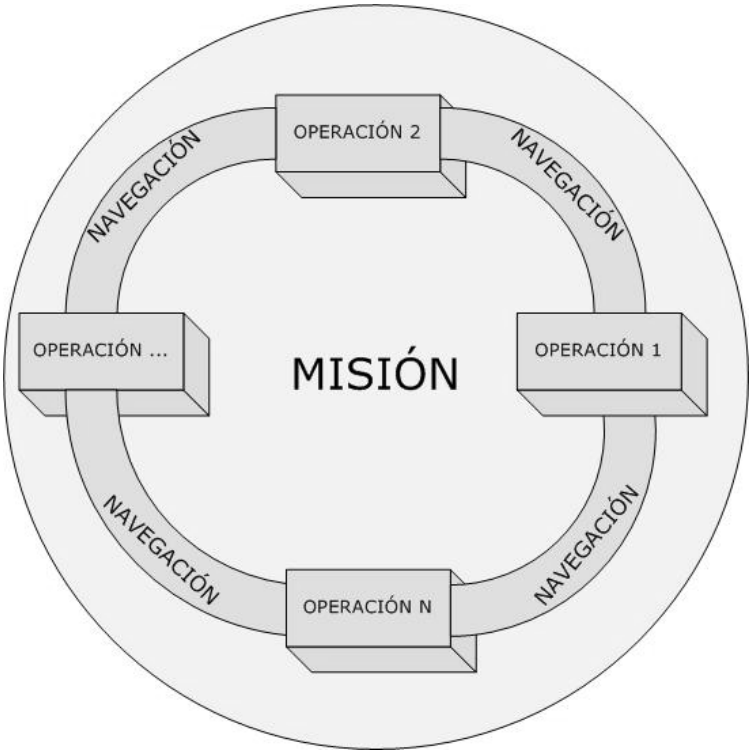


Figura 5.1- Esquema de la misión de un robot móvil

El robot móvil se caracteriza por realizar una serie de desplazamientos y por llevar a cabo una interacción con distintos elementos de su entorno de trabajo, que implican el cumplimiento de una serie de objetivos impuestos según cierta especificación. Así, se establece formalmente el concepto de misión en el ámbito de los robots móviles como la realización conjunta de una serie de objetivos de navegación y operación (Levi, 1987), como se muestra en la figura 5.1

Para un robot móvil realizar una tarea de navegación significa recorrer un camino que lo conduzca desde una posición inicial hasta otra final, pasando por ciertas posiciones intermedias o submetas. El problema comúnmente se divide en las siguientes etapas:

- Percepción del mundo: Mediante el uso de sensores externos, creación de un mapa o modelo del entorno donde se desarrollará la tarea de navegación, en esta etapa se sitúa al robot en el medio.

- Planificación de la ruta: Se crea una secuencia ordenada de objetivos o submetas que deben ser alcanzadas por el vehículo. Esta secuencia se calcula utilizando el modelo o mapa de entorno, la descripción de la tarea que debe realizar y algún tipo de técnica para la consecución de dicho objetivo.

- Seguimiento del camino: Efectúa el desplazamiento del vehículo según el camino generado mediante el adecuado control de los actuadores del vehículo. En el caso de ambientes dinámicos el seguimiento puede ser alterado por un cambio en la percepción del medio, que lleva a una iteración de las etapas previamente mencionadas para alcanzar el objetivo deseado.

Estas tareas pueden llevarse a cabo de forma separada, aunque en el orden especificado. La complejidad del sistema necesario para desarrollar la tarea depende principalmente del conocimiento que se posee del entorno de trabajo.

5.1 Tipos de Arquitecturas

La arquitectura de un robot determina el funcionamiento que tiene el mismo al enfrentarse con su entorno para poder realizar las tareas de percepción, planificación y seguimiento, a continuación se describen:

5.1.1 Modelos tradicionales

En estos modelos se tiene una representación del medio ambiente, se planean las acciones y movimientos del robot por adelantado, estos modelos tienen una organización serie, suelen ser útiles en entornos controlados o virtuales, como los videojuegos.

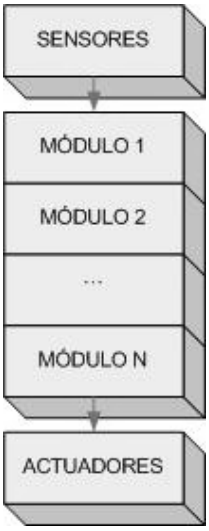


Figura 5.1 – Arquitectura *pipeline* (Modelo tradicional)

No son adecuados para entornos dinámicos o para robots que presentan errores en la ejecución de movimiento o errores de sensado.

5.1.2 Modelos reactivos

Se basan en el comportamiento de los insectos, para estos no es necesaria una representación del medio ambiente, no utilizan la planeación de acciones

ni de movimiento, son adecuados para entornos dinámicos y con errores de sensado, los comportamientos funcionan en paralelo. Algunos ejemplos de este tipo de modelo son las máquinas de estado, las redes neuronales, las máquinas vectoriales y la robótica BEAM, entre otros.

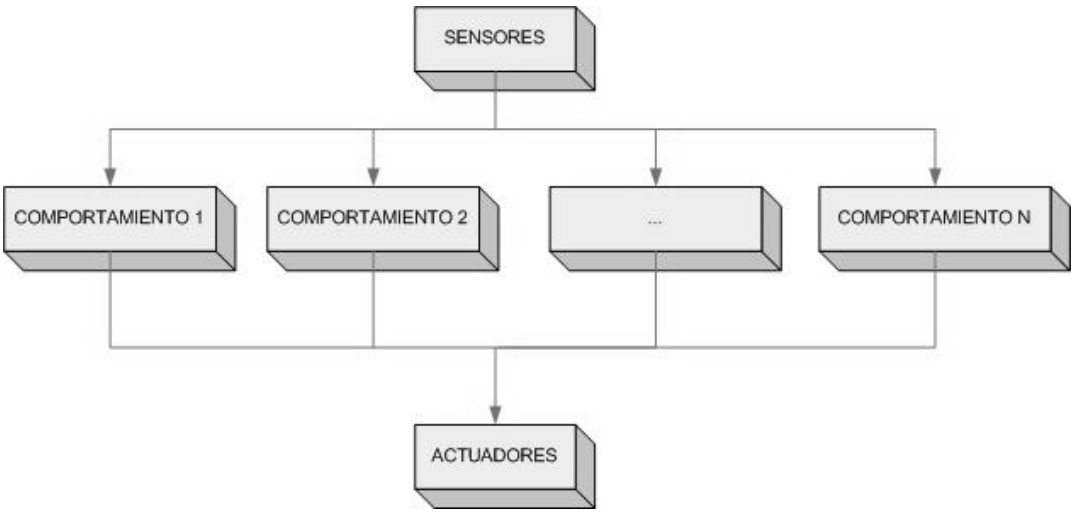


Figura 5.2 – Modelo reactivo

5.1.3 Modelos híbridos

Combinan las arquitecturas tradicional y reactiva para suplir las deficiencias de cada una de ellas. Es el modelo ideal si se busca un modelo que permita la planeación bajo entornos dinámicos.

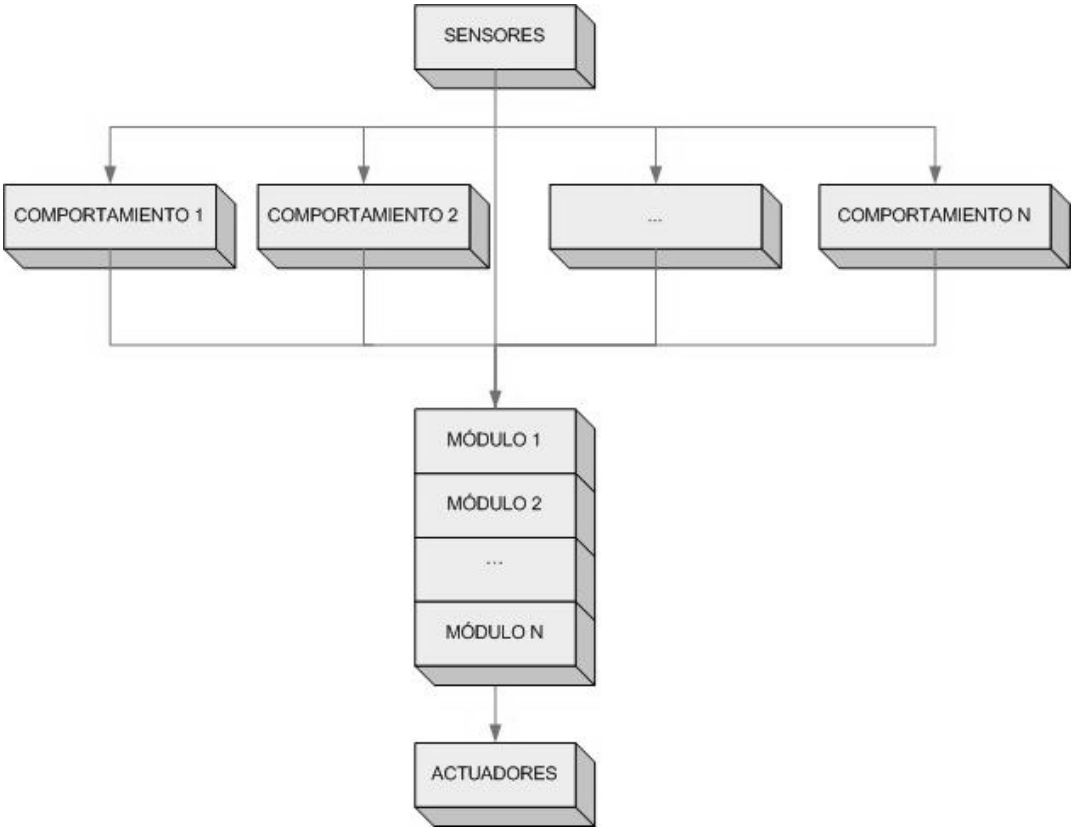


Figura 5.3 – Modelo Híbrido

5.2 Campos Potenciales

La idea de robótica basada en comportamientos se sustenta en la noción de robots construidos para seguir ciertas normas, como lo retrata Asimov en su obra “Yo robot” con las tres leyes universales de la robótica, esta idea sostiene que los comportamientos se basan en simples ciclos de estímulo respuesta en los que las acciones son potenciadas o inhibidas de acuerdo con la programación establecida.

El comportamiento basado en campos potenciales artificiales (abordado por Latombe y Arkin entre otros) es un método que genera una respuesta rápida a la información obtenida de los sensores. Constituye también un método de arquitectura híbrida para la planificación en el que se prima la reactividad, integrándose las funciones de planificación con las de control de movimientos.

Se trata de un modelo híbrido que realiza una planeación para el movimiento y cambia el modelo planeado al alterarse el entorno.

La mejor forma de llegar a un entendimiento de los campos potenciales es a través de una analogía; los campos potenciales son como las formas de las colinas que guían la trayectoria de una partícula a través de un sendero. La idea básica del comportamiento es que la respuesta de la partícula depende de la combinación de las formas en el campo. A diferencia de las colinas, donde la topología está determinada por las condiciones ambientales, la topología de los campos potenciales artificiales que un robot experimenta en un espacio es determinada por el programador. Específicamente el programador genera distintos comportamientos, cada uno asignado a una tarea en particular, después representa cada uno de estos como un campo potencial y finalmente la representación total del espacio surge de la combinación de los campos generados.

El bloque fundamental de los campos potenciales es el vector de acción, que corresponde al comando a seguir por el robot. Cada comportamiento genera una salida deseada que se representa por medio de un vector. Por ejemplo el comportamiento *SeekGoal* (buscar meta) tiene como objetivo llevar al robot hacia una meta identificada. La salida del comportamiento es un vector que dirige al robot hacia la meta identificada. El conjunto de vectores crea un campo que representa potenciales sintéticos de energía para el robot.

El campo potencial asociado con el comportamiento *SeekGoal* es un ejemplo de campo atractivo (Figura 5.4).

Otro comportamiento muy útil para la navegación de robots móviles es *AvoidObstacle* (evitar obstáculo), tiene como objetivo generar alejamiento de un objeto, un campo repulsivo, ligado a un radio de influencia, fuera del cual el potencial repulsivo es cero (Figura 5.5).

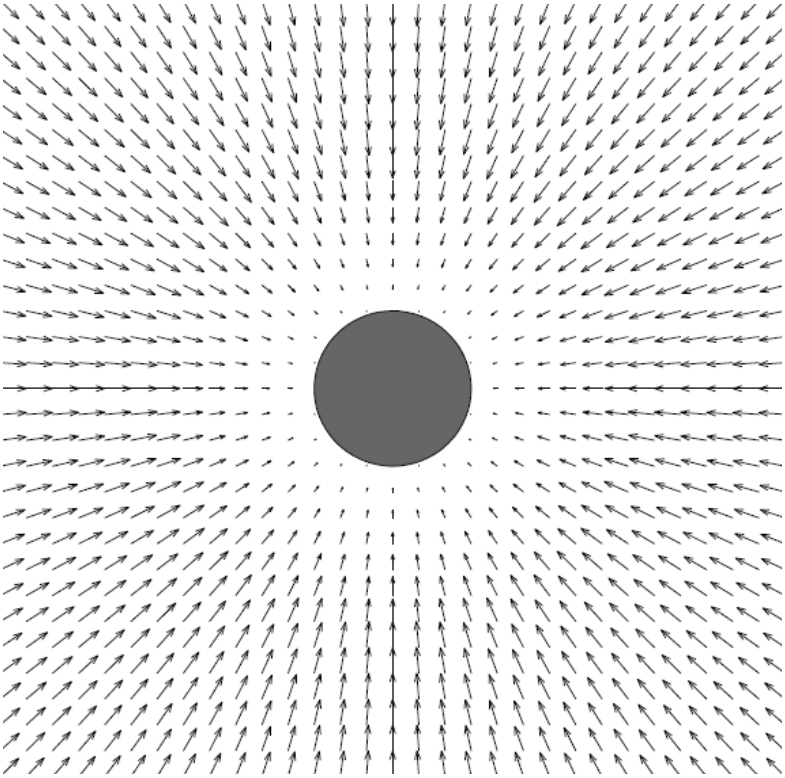


Figura 5.4 – Campo atractivo correspondiente al comportamiento *SeekGoal*

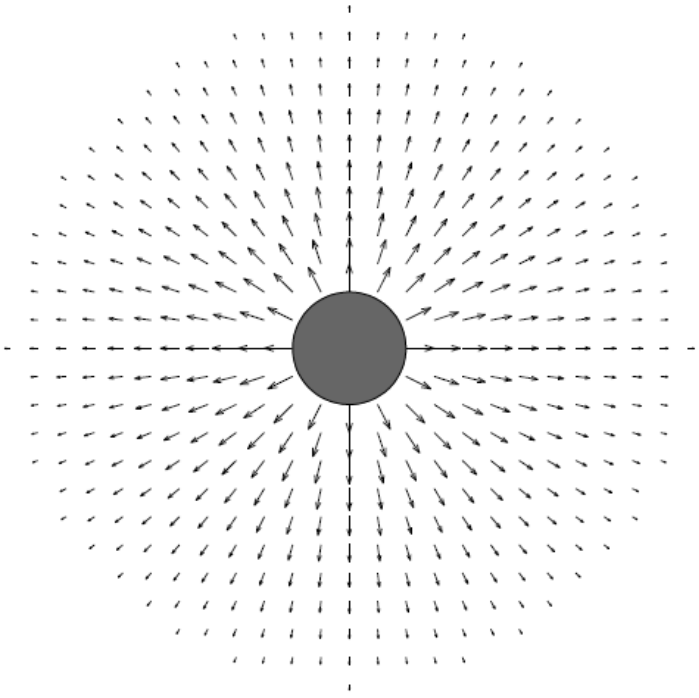


Figura 5.5 – Campo repulsivo correspondiente al comportamiento *AvoidObstacle*

Para una navegación integral es necesario considerar metas y obstáculos, sumando campos atractivos y repulsivos se puede generar un campo potencial artificial que cumpla con los dos comportamientos de forma simultánea y eficiente, como se muestra en la figura 5.6.

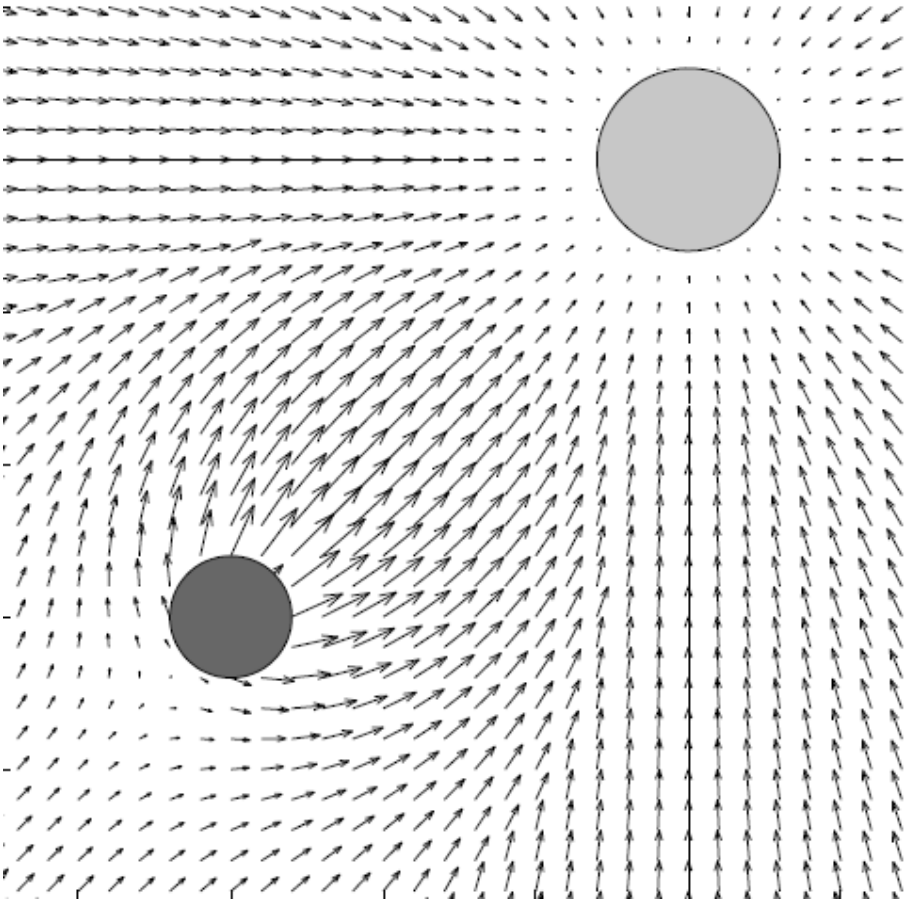


Figura 5.6 – Campo potencial artificial

5.2.1 Modelo matemático de campos potenciales

Para poder llevar a cabo la aplicación de los campos potenciales artificiales es necesario establecer el modelo matemático del campo para un espacio determinado.

Supongamos un campo parabólico de dos dimensiones, que determinará el movimiento de nuestro robot. (Figura 5.7)

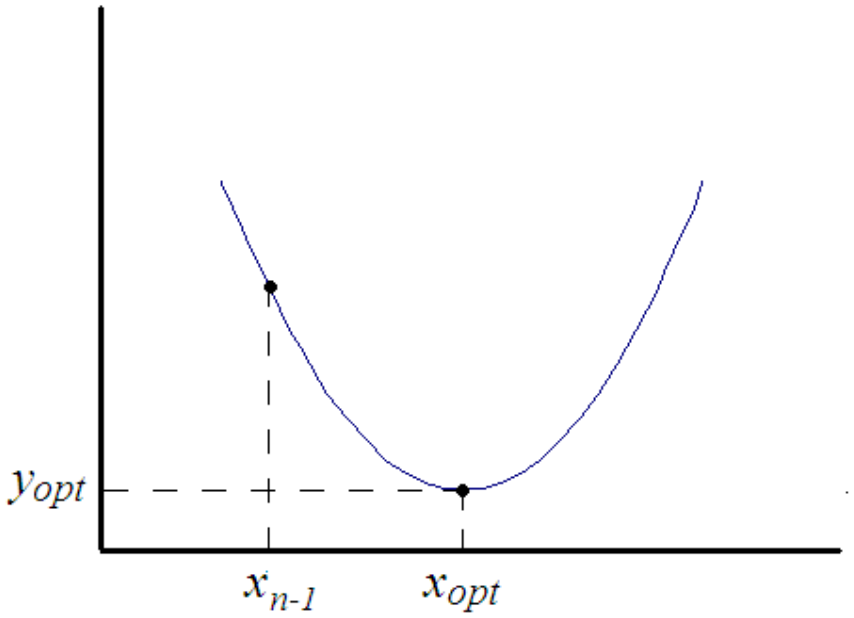


Figura 5.7 – Campo parabólico

Donde nuestro objetivo es llegar al punto (x optimo, y optimo) por el mejor camino a través del campo.

$$y = y_{opt} + (x - x_{opt})^2 \tag{5.1}$$

$$x_n = f(x_{n-1}) = x_{n-1} - \delta \cdot \frac{dy}{dx} \tag{5.2}$$

siendo δ el incremento con respecto a la posición anterior. Ahora derivando 5.1 y sustituyéndola en 5.2 con δ =1/2.

$$x_n = x_{n-1} - \frac{1}{2} \cdot 2(x - x_{opt}) = x_{opt} \tag{5.3}$$

como se observa, el robot se mueve por la pendiente más pronunciada.

En el plano:

$$\overline{q}_n = (x_n, y_n)$$

$$\overline{q}_{n+1} = \overline{q}_n + \delta \cdot f(\overline{q}_{n-1}) \tag{5.4}$$

Donde $f(\overline{q}_{n-1})$ es un vector unitario de fuerzas en la dirección del gradiente de un campo potencial.

$$\bar{f}(\overline{q}) = \frac{\overline{F}(\overline{q})}{\left|\overline{F}(\overline{q})\right|} \tag{5.5}$$

Considerando que

$$\overline{F}(q) = \nabla U(q) \tag{5.6}$$

recordando la definición de gradiente

$$\nabla U(q) = \left(\frac{\partial u}{\partial x} i + \frac{\partial u}{\partial y} j\right)$$

Entonces para un campo potencial artificial constituido por fuerzas atractivas y repulsivas.

$$U(\overline{q}) = U_{atr}(\overline{q}) + U_{rep}(\overline{q}) \tag{5.7}$$

$$\overline{F}(\overline{q}) = \overline{F}_{atr}(\overline{q}) + \overline{F}_{rep}(\overline{q}) \tag{5.8}$$

En el caso atractivo el campo puede modelarse con dos comportamientos, uno de tipo cónico cuando el robot se encuentra lejano al obstáculo y se requiere de un avance con pendiente pronunciada y otro de tipo parabólico al aproximarse

al destino, para amortiguar la llegada. La lejanía o cercanía es definida por una distancia arbitraria asociada con la constante *kr*.

Campo parabólico **(*distancia al destino < kr*)**

$$U_{atr}(\bar{q}) = \frac{1}{2} E_1 \left| \bar{q} - \bar{q}_{destino} \right|^2 \tag{5.9}$$

Aplicando 5.6 para 5.9

$$U_{atr}(x, y) = \frac{1}{2} E_1 ((x - x_d)^2 + (y - y_d)^2)$$

$$\nabla U_{atr}(x, y) = E_1 (x - x_d) i + E_1 (x - x_d) j$$

$$\boxed{\bar{F}_{atr}(\bar{q}) = E_1 \cdot (\bar{q} - \bar{q}_{destino})} \tag{5.10}$$

Campo cónico **(*distancia al destino > kr*)**

$$U_{atr}(q) = E_2 \left| \bar{q} - \bar{q}_{destino} \right| \tag{5.11}$$

Aplicando 5.6 para 5.11

$$\boxed{\bar{F}_{atr}(\bar{q}) = \frac{E_2 \cdot (\bar{q} - \bar{q}_{destino})}{\left| \bar{q} - \bar{q}_{destino} \right|}} \tag{5.12}$$

El campo de fuerzas atractivas queda finalmente definido por 5.10 y 5.12 y por las constantes atractivas E_1 y E_2.

Para el campo repulsivo es necesaria la presencia de obstáculos en el espacio de trabajo, el obstáculo es modelado en función de un radio de influencia del campo repulsivo generado por el obstáculo, que se establece como un valor constante *ki*. Fuera de ese radio de influencia la repulsión es nula.

De acuerdo con (Khatib, 1989) el vector de repulsión generado por un obstáculo puede ser definido:

$$U_{rep}(\overline{q}) = \frac{1}{2}\eta \cdot (\frac{1}{\overline{q} - \overline{q}_{obstáculo}} - \frac{1}{d_o})$$

(5.13)

donde d_o representa la distancia al obstáculo y n la constante de repulsión. Por último aplicando 5.6 el campo de fuerzas repulsivas queda:

$$\overline{F}_{rep}(\overline{q}) = -\eta \cdot (\frac{1}{|\overline{q} - \overline{q}_{obstáculo}|} - \frac{1}{d_0})(\frac{1}{|\overline{q} - \overline{q}_{obstáculo}|^2}) \cdot \frac{(\overline{q} - \overline{q}_{obstáculo})}{|\overline{q} - \overline{q}_{obstáculo}|}$$

(5.14)

cuando $\boldsymbol{d_o < ki}$

Los valores obtenidos por medio de los campos potenciales artificiales nos brindan la trayectoria óptima para llegar a nuestro objetivo por la ruta más eficiente de acuerdo con los parámetros establecidos en el modelo, quedando este determinado por los valores de las constantes atractivas y repulsivas.

Capítulo 6
Modelo de control del robot móvil

6. Modelo de control del robot móvil

El control ha desempeñado un papel vital en el avance de la ingeniería y de la ciencia, en particular el control automático se ha convertido en una parte importante e integral de los procesos modernos industriales y de fabricación. Por ejemplo, el control automático es esencial en el control numérico de las máquinas herramienta de las industrias de manufactura, es indispensable en el diseño de sistemas automáticos para el desarrollo de procesos industriales eficientes, y es una herramienta muy útil en la industria aeroespacial y en el diseño automotriz, solo por mencionar algunas aplicaciones.

Un proceso se define como una operación o desarrollo natural progresivamente continuo, marcado por una serie de cambios graduales que se suceden unos a otros de forma relativamente fija y que conducen a un resultado o propósito determinados; o en términos de robótica se define como una operación artificial que se hace de forma progresiva y que consta de una serie de acciones o movimientos controlados, sistemáticamente dirigidos hacia un resultado o acción.

Un sistema es una combinación de componentes que actúan juntos y realizan un objetivo determinado.

El control se encarga de mantener un equilibrio o de llevar a un punto de operación en particular a los sistemas y procesos, éste se realiza a través de la manipulación de variables. Una de ellas es la variable controlada, que como su nombre lo indica es la cantidad o condición que se mide y controla, otra es la variable manipulada que es la cantidad o condición que el controlador modifica para afectar el valor de la variable controlada. Normalmente, la variable controlada es la salida del sistema. Controlar significa medir el valor de la variable controlada del sistema y aplicar la variable manipulada al sistema para corregir o limitar la desviación del valor medido respecto del valor deseado.

Mediante el control automático de procesos se pretende concebir y realizar sistemas que permitan gobernar un proceso sin la intervención de agentes externos, en particular el hombre.

El objetivo del control en robótica móvil es que el vehículo ejecute de forma autónoma movimientos previamente planificados o que reaccione de forma apropiada a la percepción del entorno.

6.1 Modelado de robots móviles

Para poder realizar el control del robot móvil es necesario conocer el modelo geométrico cinemático y dinámico del robot. Se entiende por modelo geométrico del robot la relación entre los valores de las variables asociadas a la posición y orientación de un sistema de referencia, solidario al robot, que se define teniendo en cuenta la tarea que se pretende desarrollar con el robot. En vehículos robóticos, el sistema de referencia se elige normalmente asociado al punto de guía deseado, la cinemática, como es sabido, es el estudio del movimiento sin considerar las fuerzas que lo producen. Por tanto, se trata de estudiar tanto las propiedades geométricas como las temporales del movimiento. En estos términos, se considera, además del problema puramente geométrico involucrado en el posicionamiento estático, las variaciones en el tiempo de las posiciones y orientaciones; es decir, de las velocidades y aceleraciones.

En el modelado geométrico cinemático se involucra esencialmente el estudio de las relaciones existentes entre las variables de acción y el espacio de trabajo, o espacio operacional, que suele ser un espacio cartesiano. El modelado dinámico se sugiere para futuros trabajos.

6.2 Modelado cinemático de robots móviles

Para facilitar el modelado del sistema se adoptan las siguientes hipótesis simplificadoras:

- El robot se mueve sobre una superficie plana.

- Los ejes de guiado son perpendiculares al suelo.

- Se supone que las ruedas son laterales fijas y se mueven con rodadura pura, es decir el deslizamiento es despreciable.

- El robot no tiene partes flexibles.

- Durante un periodo de tiempo suficientemente pequeño en el que se mantiene constante la consigna de dirección, el vehículo se moverá de un punto a otro a lo largo de un arco de circunferencia.

- El robot se comporta como un sólido rígido.

Ahora para realizar el modelado del desplazamiento del móvil en el plano es conveniente considerar al robot como un monociclo simple en un círculo osculador, con un sistema de referencia móvil local {L} que se desplaza en un sistema de referencia fijo general {G}.

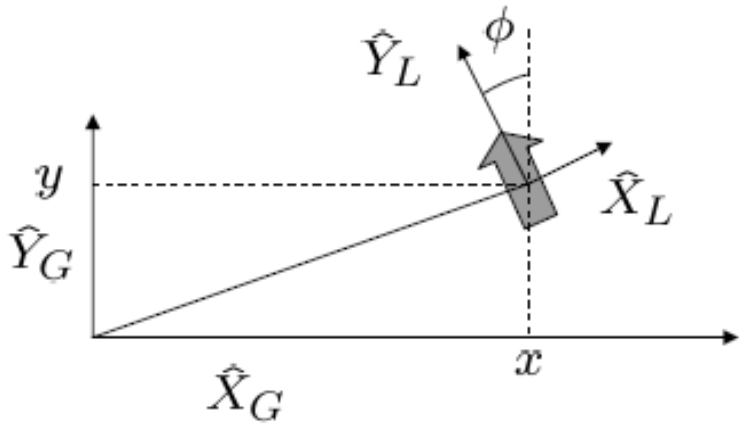


Figura 6.1 – Sistemas de referencia Global y Local

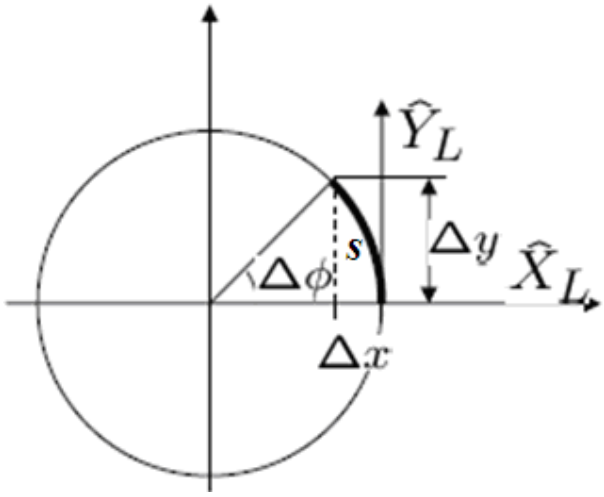


Figura 6.2 – Círculo osculador

Se supone que el vehículo se desplaza en un intervalo de control según un arco de circunferencia *s*. Esta suposición es válida para intervalos de control suficientemente pequeños. Entonces la velocidad lineal del vehículo viene dada por:

$$v = \frac{\Delta s}{\Delta t} \tag{6.1}$$

y la velocidad angular por:

$$\omega = \frac{\Delta \phi}{\Delta t} \tag{6.2}$$

La distancia equivalente a la longitud del arco recorrido por el robot en el mismo intervalo de tiempo viene dada por:

$$\Delta s = R\Delta\phi \tag{6.3}$$

siendo R el radio de giro o radio de la circunferencia que describe el punto guiado. Ahora se define curvatura como la inversa del radio de giro:

$$\gamma = \frac{1}{R} = \frac{\Delta\phi}{\Delta s} \tag{6.4}$$

Las ecuaciones de movimiento en el sistema {L} de la figura 6.1 en la posición inicial son:

$$\Delta x_L = -(R - R\cos(\Delta\phi)) \tag{6.5}$$

$$\Delta y_L = Rsen(\Delta\phi) \tag{6.6}$$

Si la orientación inicial del vehículo con respecto al sistema {G} es de ϕ, el movimiento en el sistema {G} se determina rotando ϕ:

$$\begin{aligned} \Delta x &= R(\cos(\Delta\phi) - 1)\cdot\cos(\phi) - Rsen(\Delta\phi)sen(\phi) \\ \Delta y &= R(\cos(\Delta\phi) - 1)\cdot sen(\phi) + Rsen(\Delta\phi)\cos(\phi) \end{aligned} \tag{6.7}$$

Suponiendo que el intervalo de control es lo suficientemente pequeño, también lo será el cambio de orientación $\Delta\phi$ con lo cual se obtendrá que:

$$\begin{aligned} \cos(\Delta\phi) &\cong 1 \\ sen(\Delta\phi) &\cong \Delta\phi \end{aligned} \tag{6.8}$$

Sustituyendo en las anteriores ecuaciones se tiene que:

$$\begin{aligned} \Delta x &= -R\Delta\phi\cdot sen(\phi) \\ \Delta y &= R\Delta\phi\cdot\cos(\phi) \end{aligned} \tag{6.9}$$

y teniendo en cuenta 6.3

$$\Delta x = -\Delta s \cdot sen(\phi)$$
$$\Delta y = \Delta s \cdot \cos(\phi) \qquad (6.10)$$

Dividiendo ambas ecuaciones entre Δt, teniendo en cuenta 6.1 y haciendo tender Δt a cero se llega a:

$$x' = -v \cdot sen(\phi) \qquad (6.11)$$

$$y' = v \cdot \cos(\phi) \qquad (6.12)$$

ecuaciones a las que puede añadirse:

$$\phi' = \omega \qquad (6.13)$$

lo cual nos proporciona la variación de la orientación.

6.3 Modelo de control de la configuración diferencial

En la figura 6.3 se representa la locomoción con guiado diferencial. En este caso, las variables de control son las velocidades de las ruedas laterales.

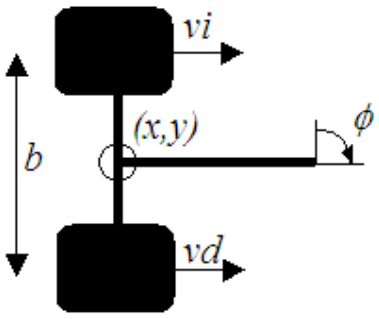


Figura 6.3 – Configuración diferencial

De acuerdo con 6.11, 6.12 y 6.13 el vector de coordenadas globales del punto de guía de un robot móvil puede expresarse como un sistema de la forma:

$$\begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} = \begin{bmatrix} -sen(\phi) \\ \cos(\phi) \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} -sen(\phi) & 0 \\ \cos(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6.14}$$

Como se analizará mas adelante en el capítulo, del planteamiento de campos potenciales se puede obtener la velocidad lineal *v* y angular ω del vehículo, por tanto es conveniente expresar el modelo en función de la velocidad angular de las ruedas como variables de control.

Sean ω_i y ω_d las velocidades de giro de las ruedas izquierda y derecha, respectivamente. Si el radio de la rueda es *c*, las velocidades correspondientes en el modelo de control son $v_i = \omega_i c$ y $v_d = \omega_d c$. En este caso, la velocidad lineal y la velocidad angular vienen dadas por:

$$v = \frac{v_d + v_i}{2} = \frac{(\omega_d + \omega_i) \cdot c}{2} \tag{6.15}$$

$$\omega = \frac{v_d - v_i}{b} = \frac{(\omega_d - \omega_i) \cdot c}{b} \tag{6.16}$$

siendo *b* la distancia que separa las ruedas del vehículo.

Despejando ω_d de 6.15 y sustituyéndola en 6.16

$$\omega_i = \frac{v - (b/2)\omega}{c} \tag{6.17}$$

de donde

$$\omega_d = \frac{v + (b/2)\omega}{c} \tag{6.18}$$

Sustituyendo 6.17 y 6.18, el modelo 6.14 puede expresarse en función de estas variables de control como

$$\begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} = \begin{bmatrix} -(c \cdot sen(\phi))/2 \\ (c \cdot \cos(\phi))/2 \\ -c/b \end{bmatrix} \omega_i + \begin{bmatrix} -(c \cdot sen(\phi))/2 \\ (c \cdot \cos(\phi))/2 \\ c/b \end{bmatrix} \omega_d \tag{6.19}$$

Y por último aplicando el modelo cinemático inverso correspondiente, como la plantea (Ollero, 2007) el modelo en términos de las velocidades de las ruedas, queda:

$$\begin{bmatrix} \omega_i \\ \omega_d \end{bmatrix} = \begin{bmatrix} -(c \cdot sen(\phi))/2 & (c \cdot \cos(\phi))/2 & -c/b \\ -(c \cdot sen(\phi))/2 & (c \cdot \cos(\phi))/2 & c/b \end{bmatrix} \begin{bmatrix} x' \\ y' \\ \phi' \end{bmatrix} \tag{6.20}$$

Modelo que controla la velocidad angular de las ruedas del robot en función de la velocidad linear proyectada sobre los ejes y la velocidad angular del móvil.

6.4 Campos potenciales en el modelo de control

Para obtener una relación entre los valores obtenidos a través de campos potenciales y el modelo de control (González Villela et al. 2004) propone que el problema de seguir la trayectoria puede ser resuelto con las siguientes reglas:

$$v = \begin{cases} v_{\max} & si \quad d_g > k_r \\ \frac{v_{\max}}{k_r} \cdot d_g & si \quad d_g \leq k_r \end{cases} \tag{6.21}$$

donde d_g es la distancia a la meta y *kr* es la constante previamente definida para campos atractivos.

Por otra parte la velocidad angular se plantea como función del ángulo de error resultado de la resta entre el ángulo de la fuerza resultante del comportamiento de campos potenciales y el ángulo actual del robot.

$$\theta_e = \theta_{campospotenciales} - \theta_{robot} \tag{6.22}$$

La velocidad angular queda determinada por:

$$\omega = \omega_{\max} sen(\theta_e) \tag{6.23}$$

Con 6.21 y 6.23 el comportamiento esperado es:

El robot se dirige a la posición final a la máxima velocidad lineal cuando se encuentra lejos y desacelera al aproximarse, mientras su velocidad angular varía en función del ángulo de error dictaminado por el método de campos potenciales.

El sistema es implementado con un sistema parametrizado de control proporcional con una sola constante *kp*, el error en estado permanente aumenta, pero podrá ser solucionado en desarrollos futuros con el diseño de un controlador digital proporcional integral.

Capítulo 7. Aplicaciones

7. Aplicaciones

En este capítulo se plantean varias aplicaciones prácticas para el sistema desarrollado, los datos reportados son resultados de las pruebas experimentales.

7.1 Entorno de las pruebas

Para poder reconocer el robot móvil y los otros elementos del medio (meta y obstáculos) durante las pruebas, es necesario adherir el símbolo fiducial correspondiente en la parte superior, haciendo coincidir el centroide del símbolo con el centro de giro del robot, el centro del obstáculo o el centro de la meta según corresponda. Cabe aclarar que la orientación de cero grados del móvil corresponde al robot en horizontal con frente hacia la derecha (en vista superior). La medida de los símbolos fiducial para estas pruebas fue de 7x7[cm], la distancia entre ruedas b = 0.11[m] y el radio de la rueda c = 0.029[m].

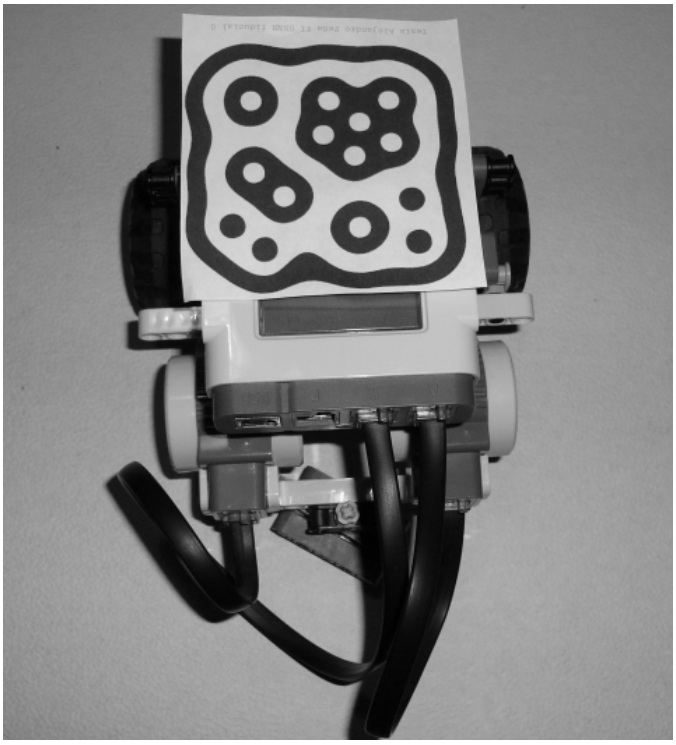


Figura 7.1 – Robot con símbolo fiducial 0 adherido

Para estas pruebas se utilizó la HandyCam Sony modelo DCR-HC28 conectada a la computadora con cable FireWire con resolución NTSC de 720x480, montada a 1.5[m] de altura. El área de trabajo enfocada por la cámara es aproximadamente de 95[cm] de ancho por 62[cm] de alto.

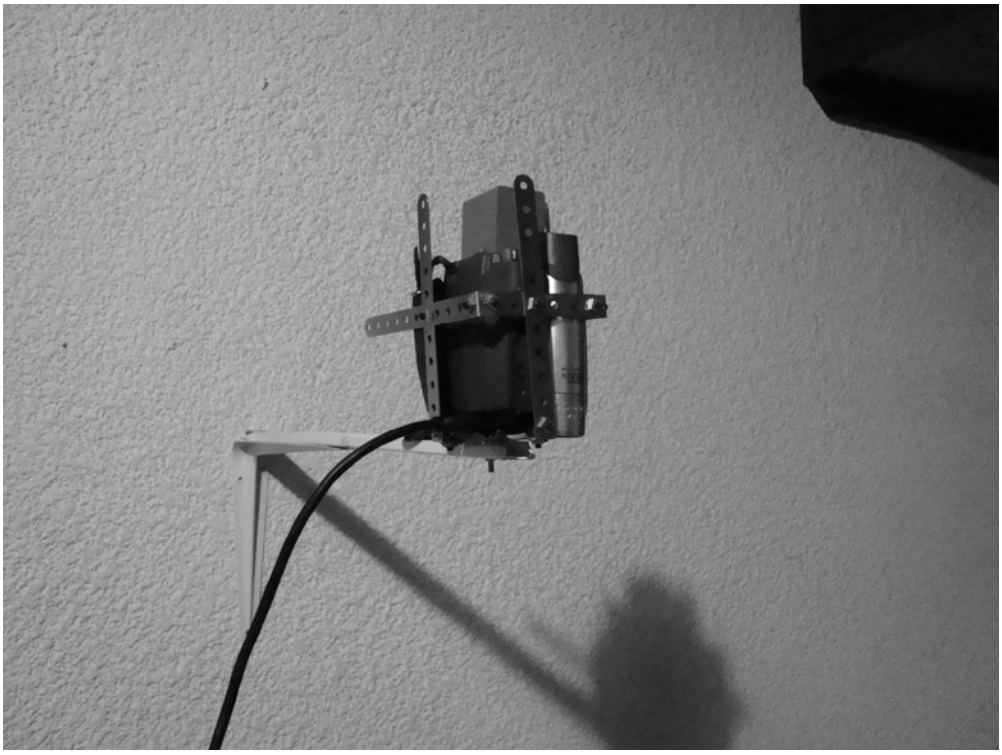


Figura 7.2 – Cámara montada

Las constantes de atracción para las pruebas son E_1 = 5, E_2 = 10, la constante de cercanía *kr* = 0.2, la constante de repulsión η = 1.5, mientras que la constante del radio de influencia *ki* = 0.5.

El periodo de muestreo de las gráficas es de 200 [ms], es decir, una frecuencia de muestreo de 5 [Hz], pues representa de forma suficientemente clara los cambios en los valores de las variables registradas.

Los videos de las pruebas realizadas pueden ser consultados en el disco adjunto o en la siguiente dirección de Internet citada en referencias.

7.2 Misión “llegar a la meta”

En esta misión el objetivo del robot es llegar a la meta por el camino más eficiente en función de campos potenciales artificiales, no existe obstáculo, la constante de proporcionalidad *kp* = 2. (Ver figura 7.3)

7.2.1 Prueba 1

El robot parte del reposo en la posición relativa (0.8922, 0.1204) ángulo inicial 180.1°, la meta se encuentra en la posición relativa (0.1083, 0.3041), las variables medidas se grafican a continuación:

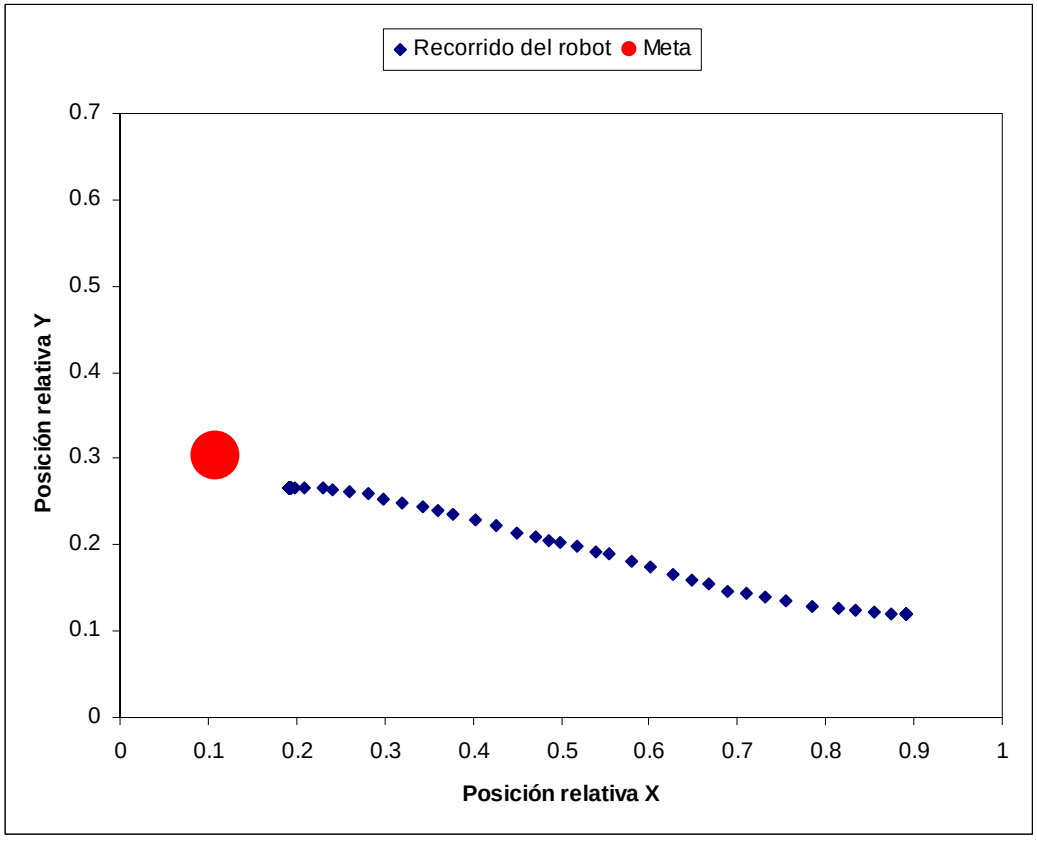


Figura 7.3 – Recorrido del robot para “llegar a la meta” prueba 1

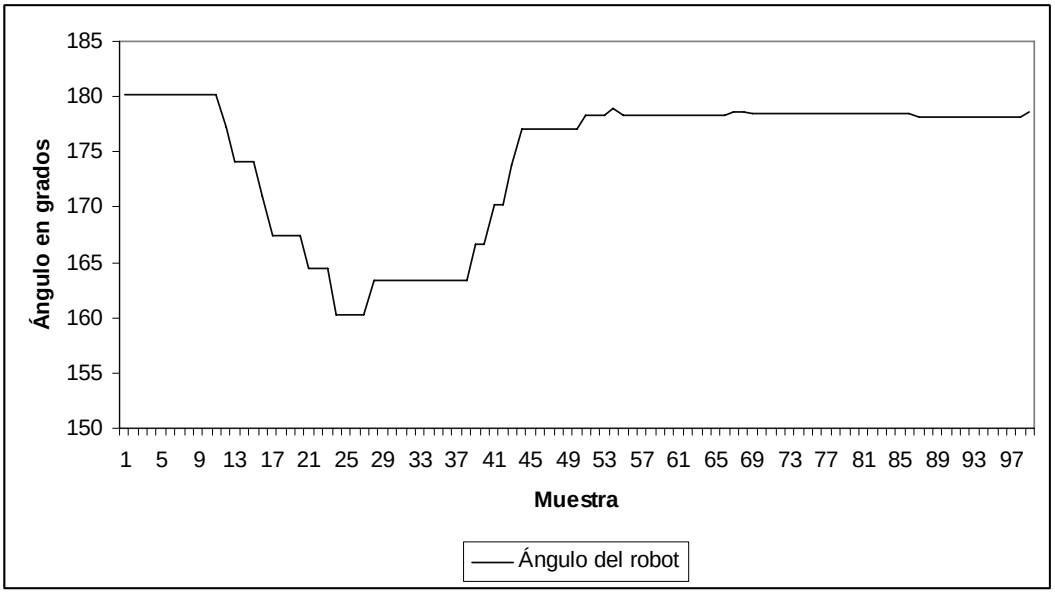


Figura 7.4 – Posición angular, en el tiempo, del robot en la misión "llegar a la meta" prueba 1

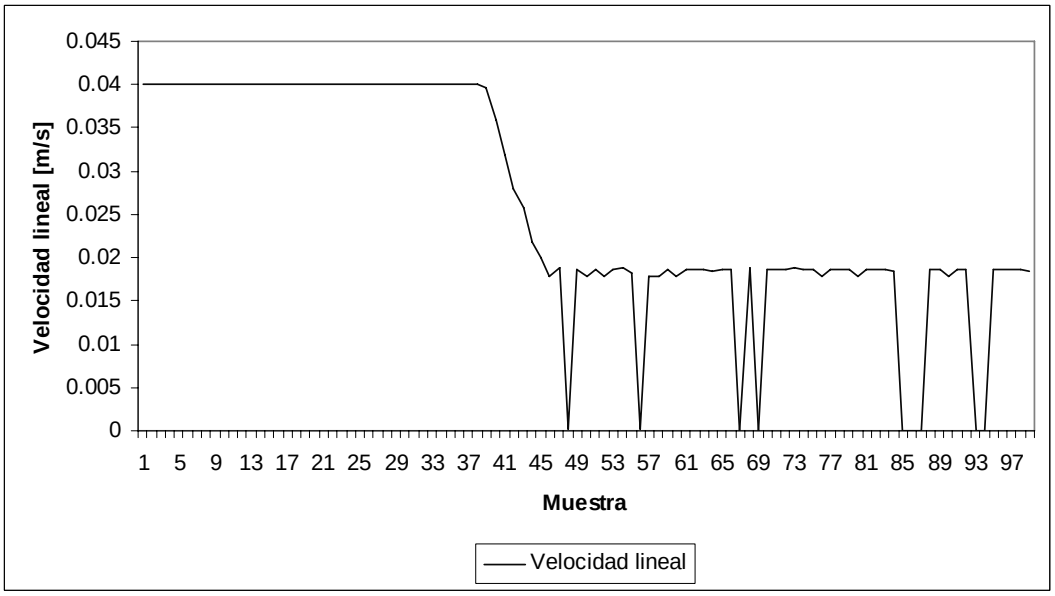


Figura 7.5 – Velocidad lineal del robot en la misión "llegar a la meta" prueba 1

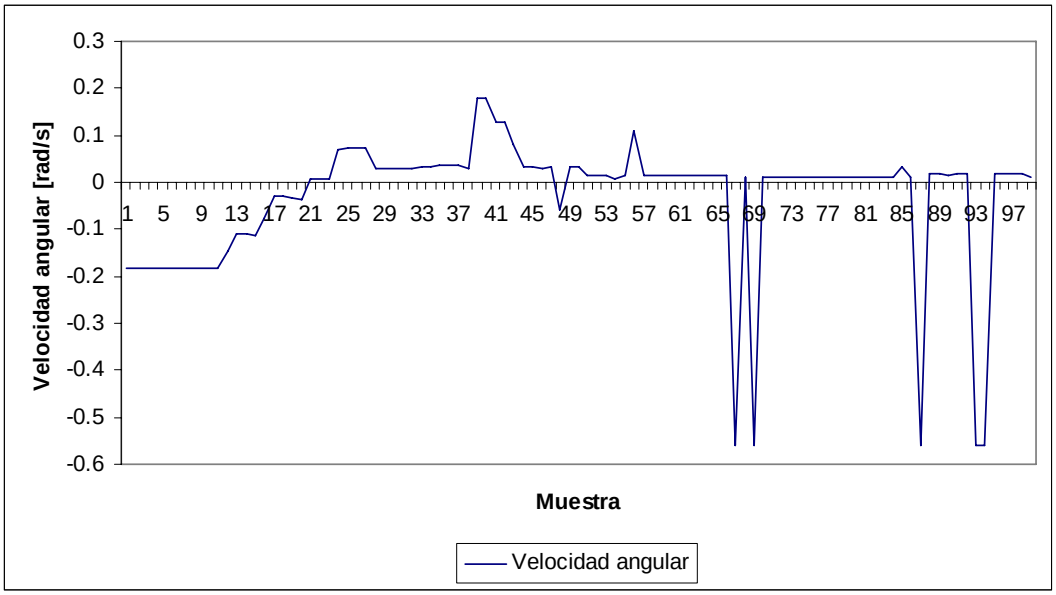


Figura 7.6 – Velocidad angular del robot en la misión "llegar a la meta" prueba 1

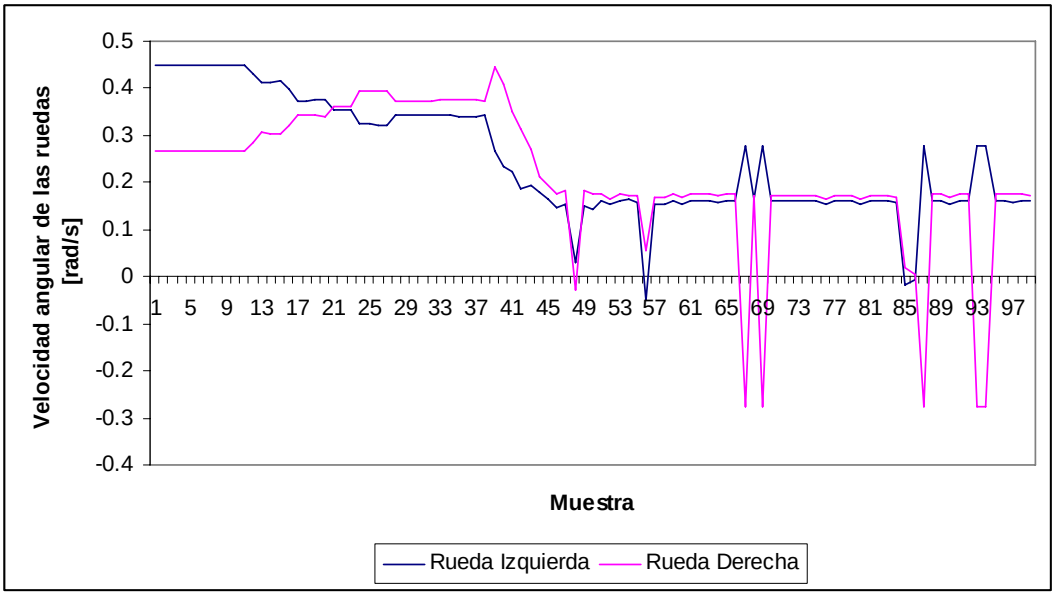


Figura 7.7 – Velocidad angular de las ruedas del robot en la misión "llegar a la meta" prueba 1

Como se observa, el robot busca la trayectoria más corta y suave para llegar a la meta, la velocidad lineal de control permanece constante, pero conforme se acerca al objetivo disminuye hasta valores por debajo del umbral de movimiento de los motores, como se observa en las demás gráficas.

7.2.2 Prueba 2

El robot parte del reposo en la posición relativa (0.8540, 0.2831) pero ahora con ángulo 334.26°, la meta se encuentra en la posición relativa (0.1093, 0.5808), las variables medidas:

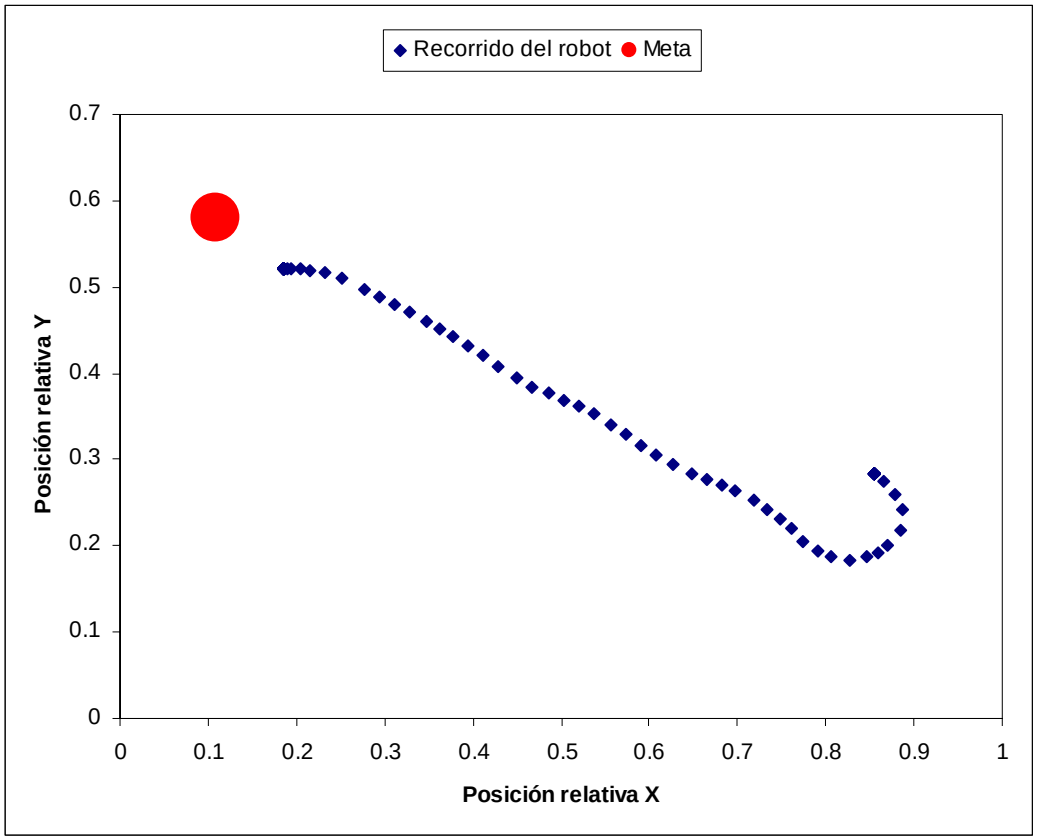


Figura 7.8 – Recorrido del robot para "llegar a la meta" prueba 2

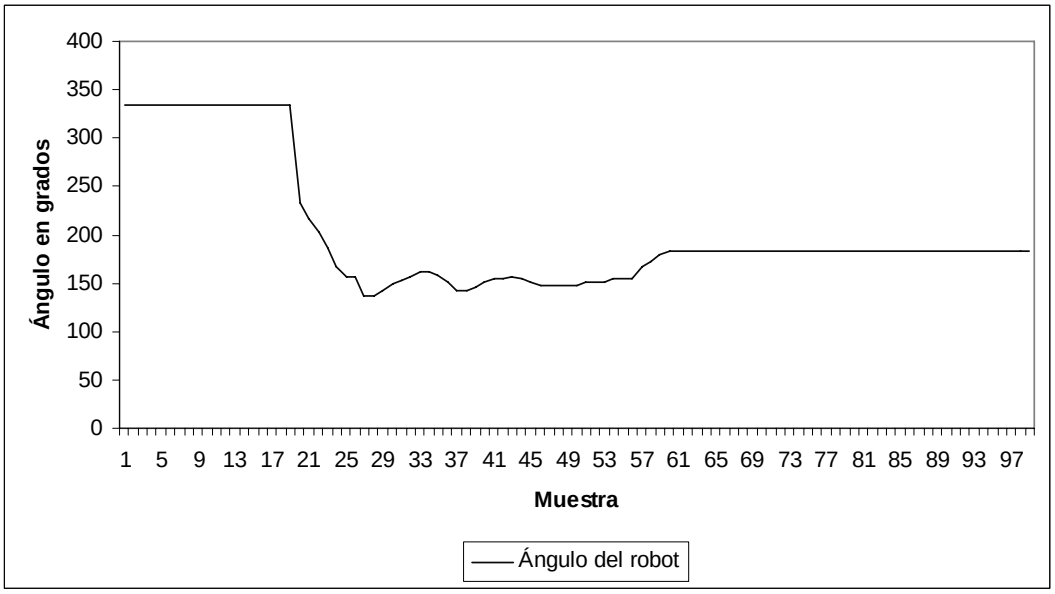


Figura 7.9 – Posición angular, en el tiempo, del robot en la misión "llegar a la meta" prueba 2

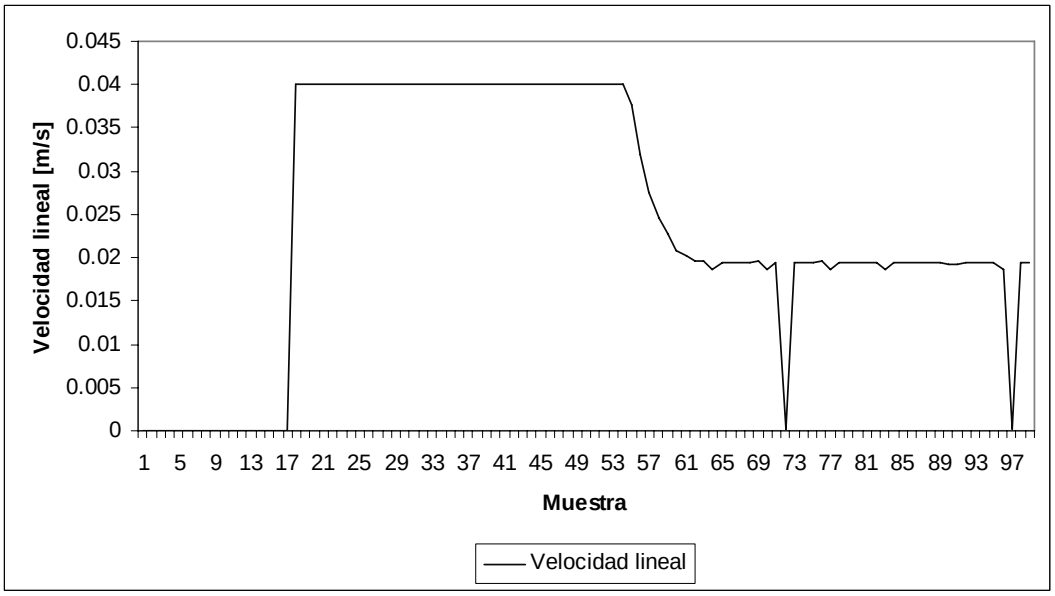


Figura 7.10 – Velocidad lineal del robot en la misión "llegar a la meta" prueba 2

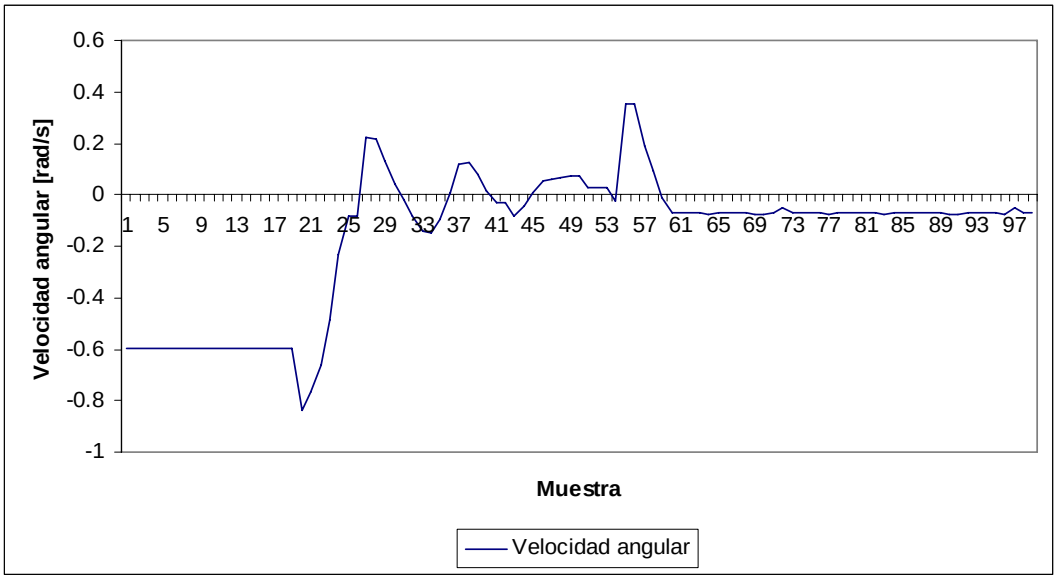


Figura 7.11 – Velocidad angular del robot en la misión "llegar a la meta" prueba 2

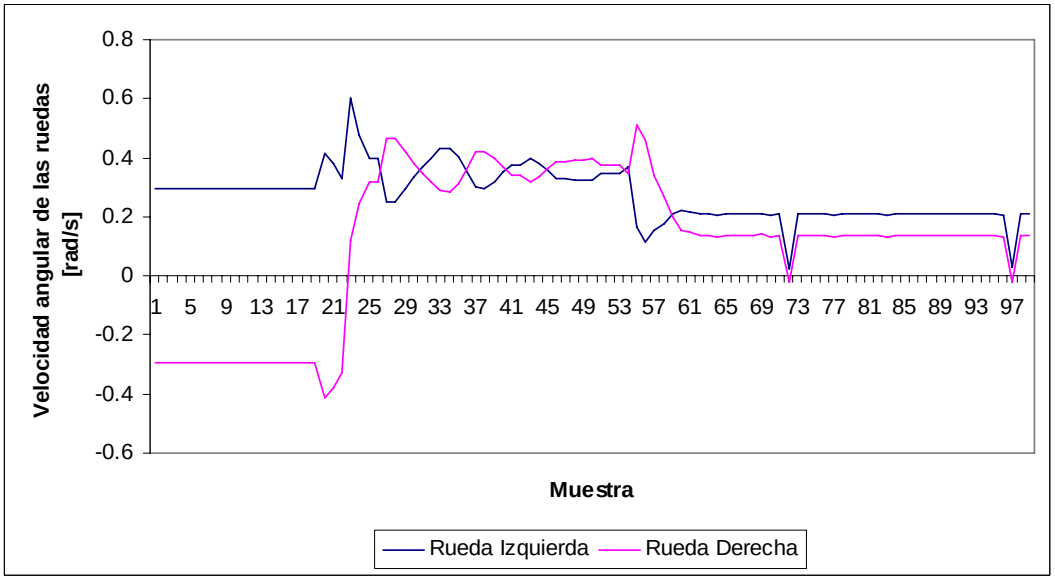


Figura 7.12 – Velocidad angular de las ruedas del robot en la misión "llegar a la meta" prueba 2

Al iniciar en orientación contraria a la trayectoria que lleva a la meta se parte con una velocidad lineal mínima, lo que hace que el móvil tenga que girar para alinearse a la trayectoria más eficiente. De la gráfica de velocidad lineal se hace evidente la disminución de los valores al acercarse al objetivo como consecuencia del campo parabólico. La trayectoria es ligeramente oscilatoria debido a los valores de la velocidad angular.

7.3 Misión “llegar a la meta con obstáculo”

En esta misión el objetivo del robot es llegar a la meta por el camino más eficiente en función de campos potenciales artificiales, evadiendo un obstáculo.

El robot parte del reposo en la posición relativa (0.8390, 0.2435) con ángulo 156.47°, la meta se encuentra en la posición relativa (0.0986, 0.5821), el obstáculo se encuentra en la posición relativa (0.5622, 0.4494) y la constante de proporcionalidad $kp = 2$

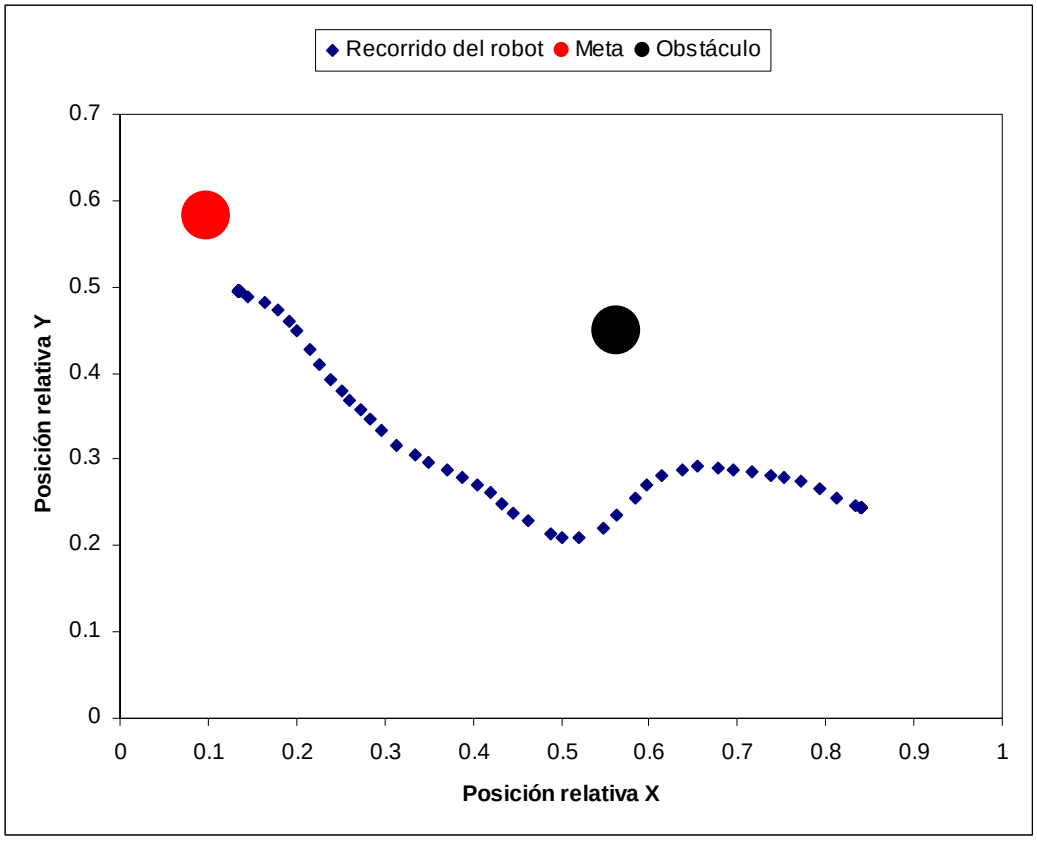


Figura 7.13 – Recorrido del robot para “llegar a la meta con obstáculo”

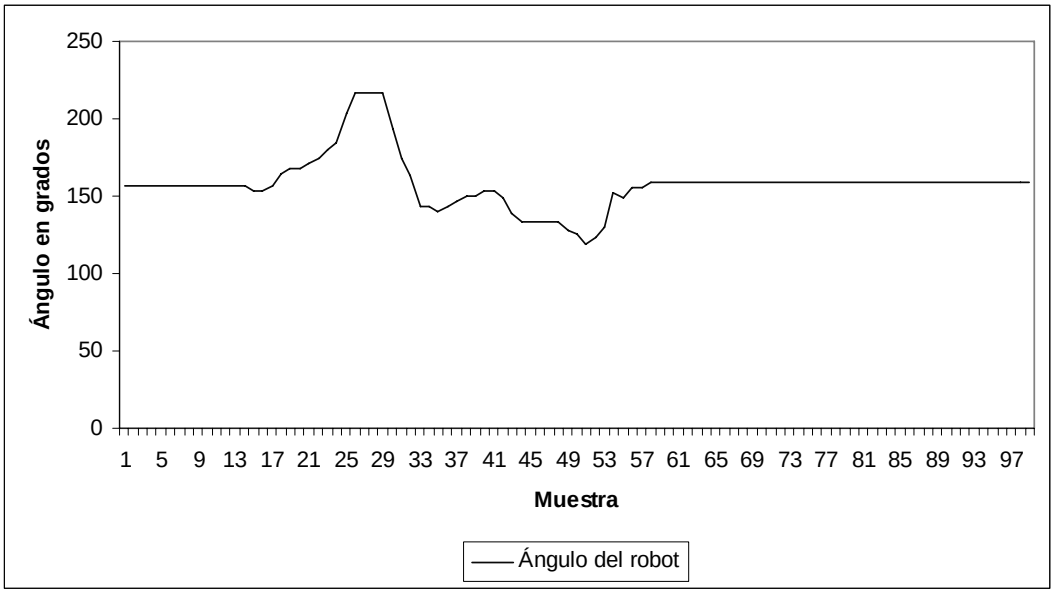


Figura 7.14 – Posición angular, en el tiempo, del robot en la misión "llegar a la meta con obstáculo"

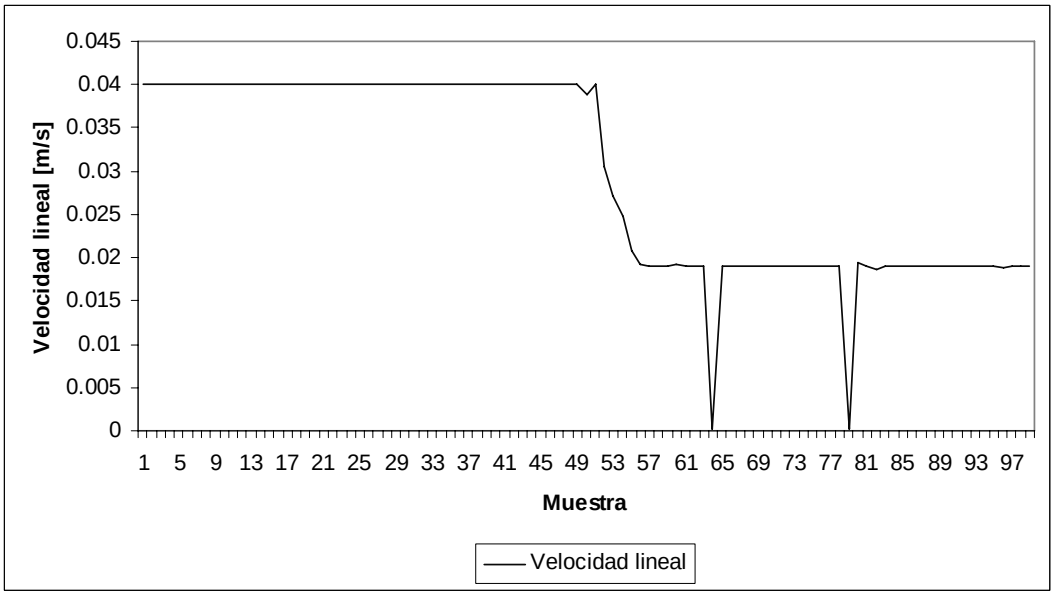


Figura 7.15 – Velocidad lineal del robot en la misión "llegar a la meta con obstáculo"

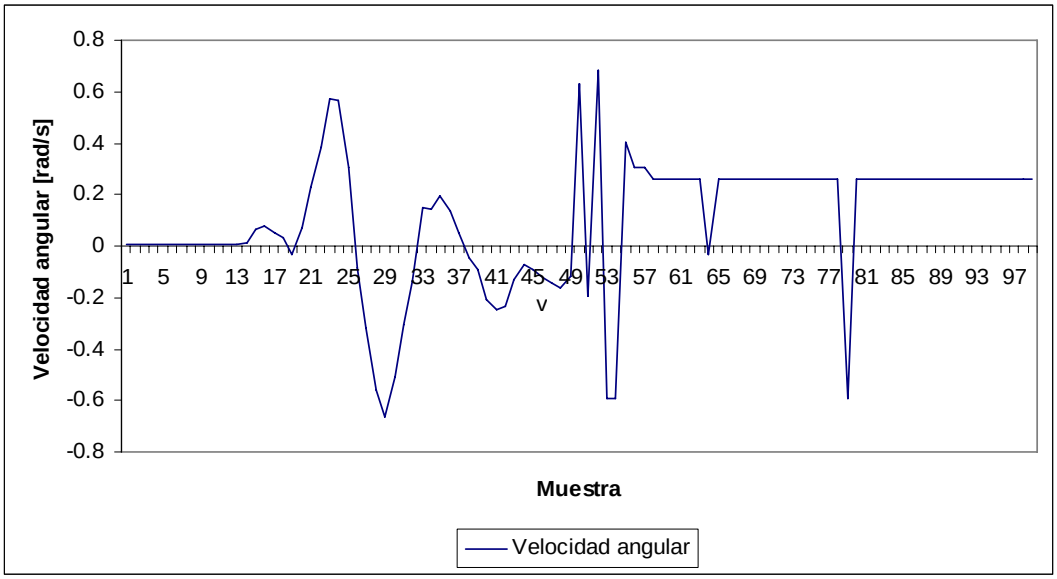


Figura 7.16 – Velocidad angular del robot en la misión "llegar a la meta con obstáculo"

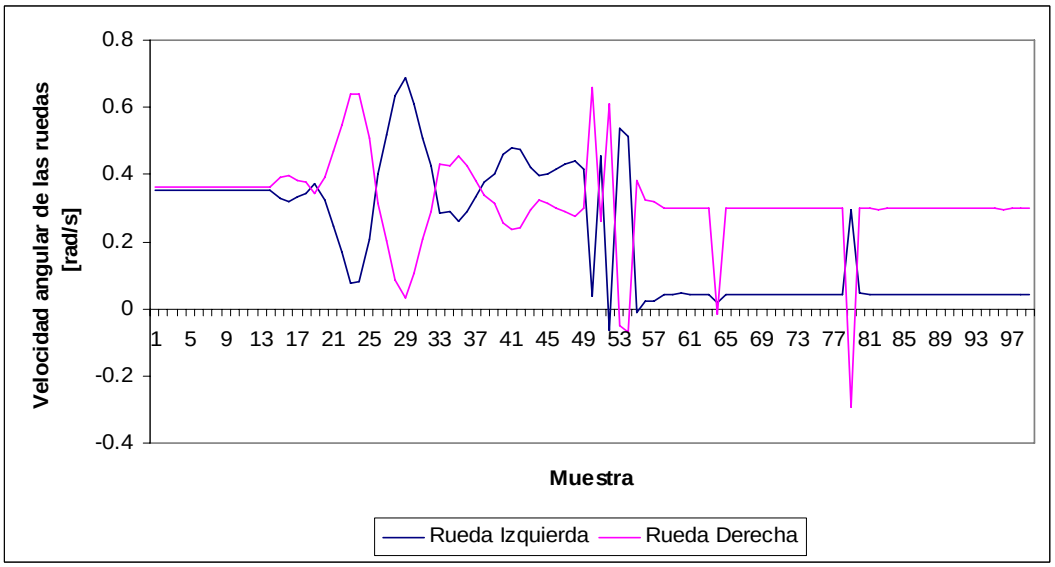


Figura 7.17 – Velocidad angular de las ruedas del robot en la misión "llegar a la meta con obstáculo"

Como se observa, el robot se dirige por el camino más eficiente hacia la meta, sin embargo, al entrar en el radio de influencia del obstáculo recibe una fuerza repulsiva considerable, que lo hace girar de forma abrupta para rodear al obstáculo sin perder de vista el objetivo final. Si la constante de repulsión es muy fuerte para esta prueba se crea un ligero efecto orbital o de sobrepaso.

7.4 Misión “capturar la meta”

En esta misión el objetivo del robot es llegar a la meta que ahora es móvil, por el camino más eficiente en función de campos potenciales artificiales, no hay obstáculos. En el caso analizado el registro del movimiento del robot parte de la posición relativa (0.5581, 0.4611) con ángulo 182.62°, la meta es otro robot móvil que se mueve en círculos, en el registro la meta móvil parte de la posición relativa (0.4140, 0.5617), la constante de proporcionalidad *kp* = 2.

El registro se hace para aproximadamente una vuelta del robot meta.

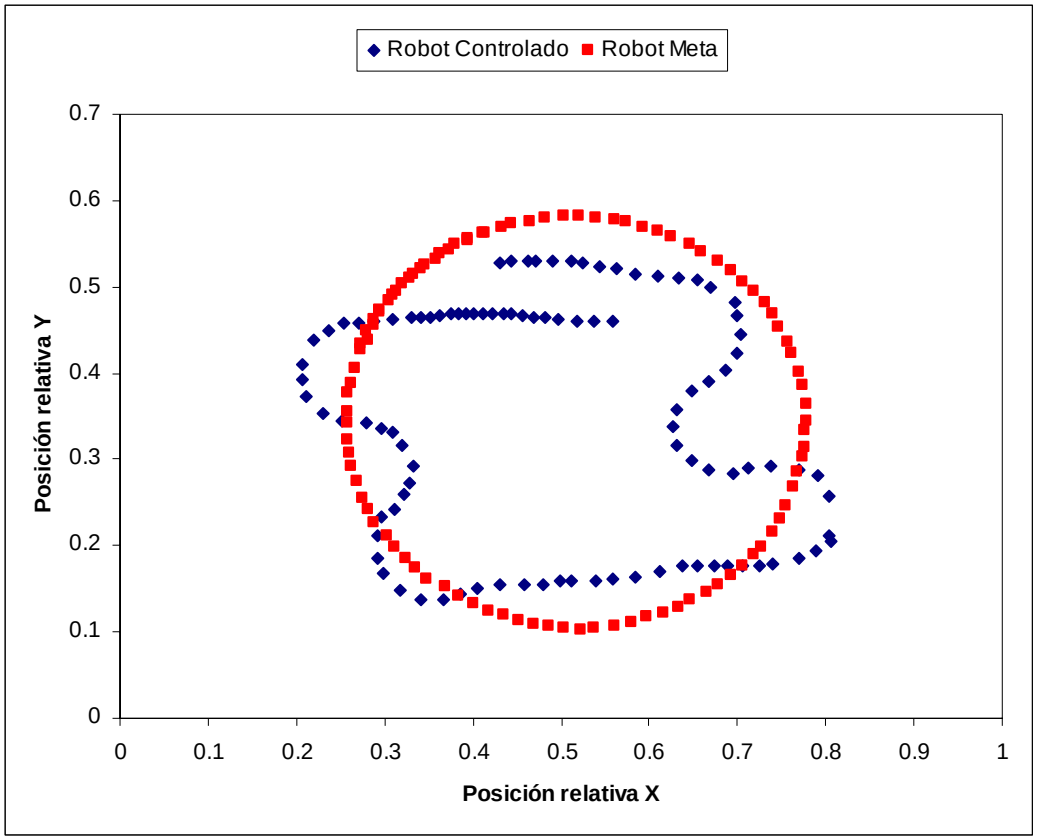


Figura 7.18 – Recorrido del robot (en azul) para “capturar la meta” (en rojo)

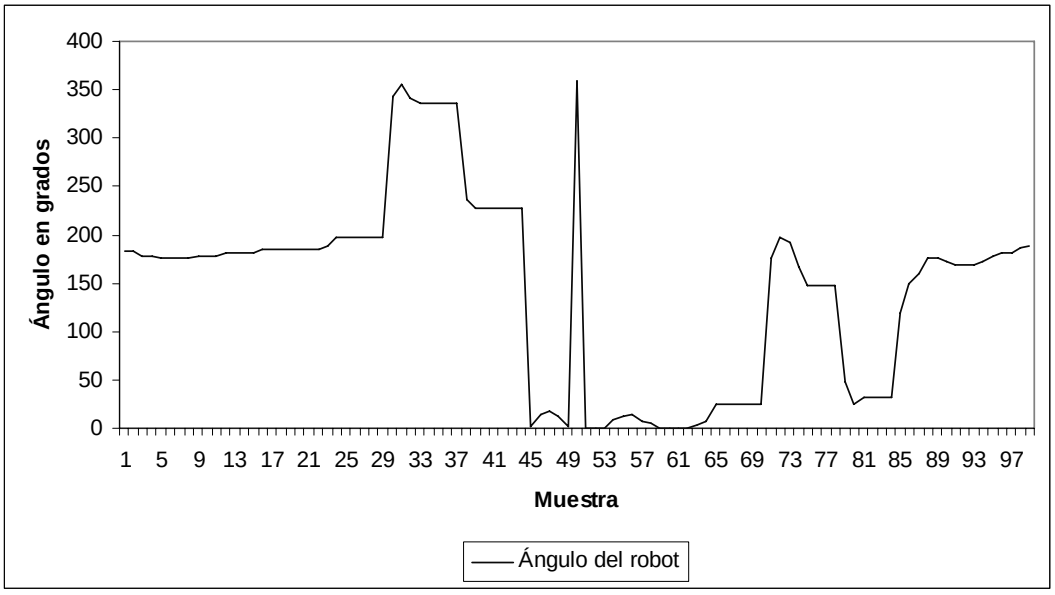


Figura 7.19 – Posición angular del robot controlado en la misión "capturar la meta"

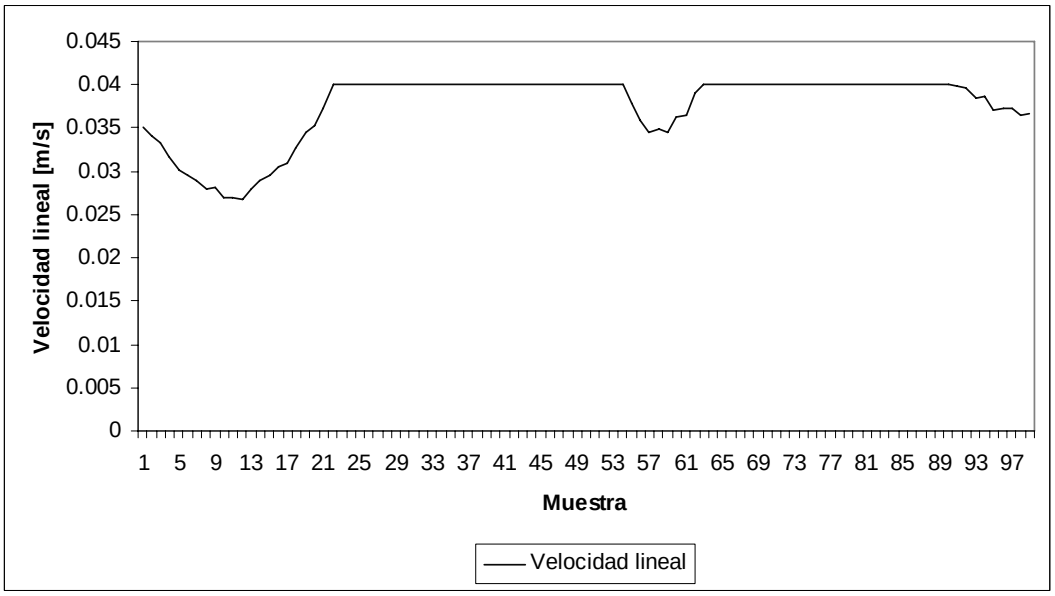


Figura 7.20 – Velocidad lineal del robot controlado en la misión "capturar la meta"

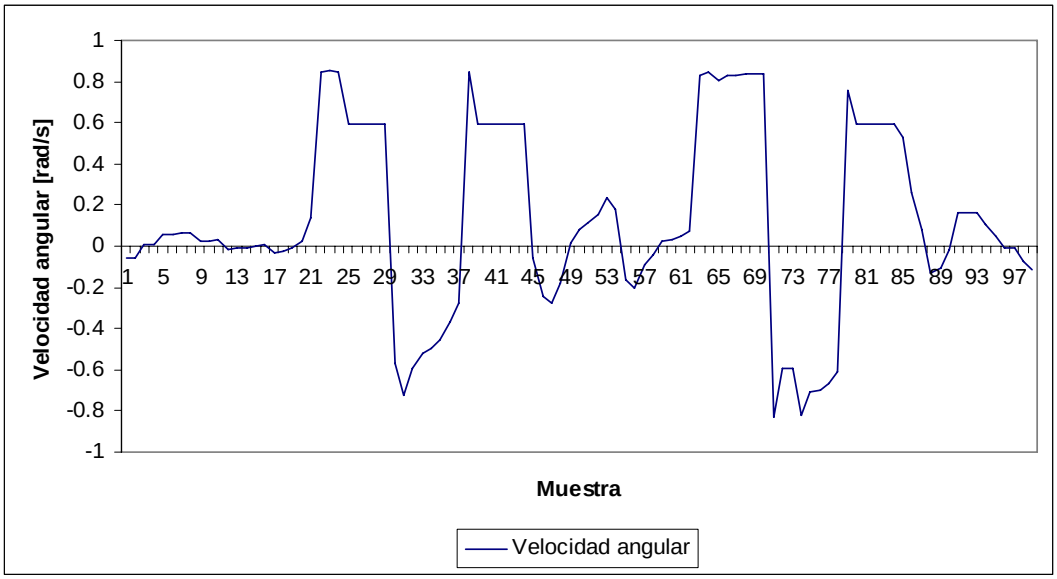


Figura 7.21 – Velocidad angular del robot en la misión "capturar la meta"

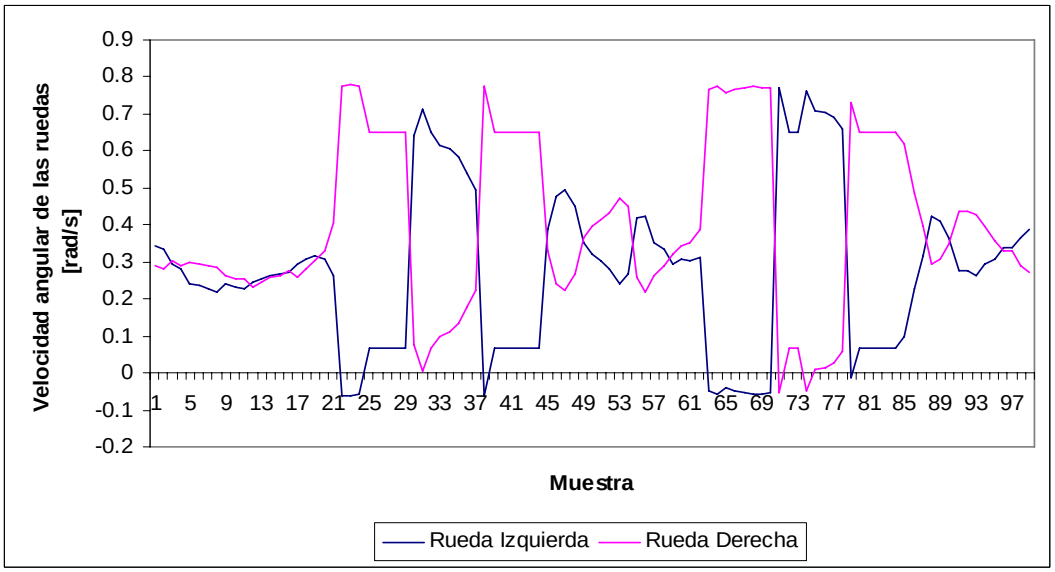


Figura 7.22 – Velocidad angular de las ruedas del robot en la misión "capturar la meta"

Esta es una de las principales aplicaciones de campos potenciales, el objetivo y por ende la trayectoria, va cambiando conforme se mueve la meta montada en el segundo robot, en el video se aprecia cómo el robot controlado persigue aumentando y disminuyendo la velocidad en función de la cercanía o lejanía del robot meta, lo que da la impresión de inteligencia. Las tres disminuciones de la gráfica de velocidad lineal coinciden con los acercamientos del robot controlado al robot meta.

7.5 Misión “capturar la meta con obstáculo”

En esta misión el objetivo del robot es llegar a la meta móvil por el camino más eficiente, en función de campos potenciales artificiales, también se agrega un obstáculo. En el caso analizado el registro del movimiento del robot parte de la posición relativa (0.5306, 0.0985) con ángulo 11.99°, la meta es otro robot móvil que se mueve en círculos, en el registro parte de la posición relativa (0.6925, 0.4659), el obstáculo se encuentra en la posición relativa (0.5431, 0.2978), la constante de proporcionalidad *kp* = 1 para que la respuesta sea más atenuada.

El registro se hace para aproximadamente una vuelta del robot meta.

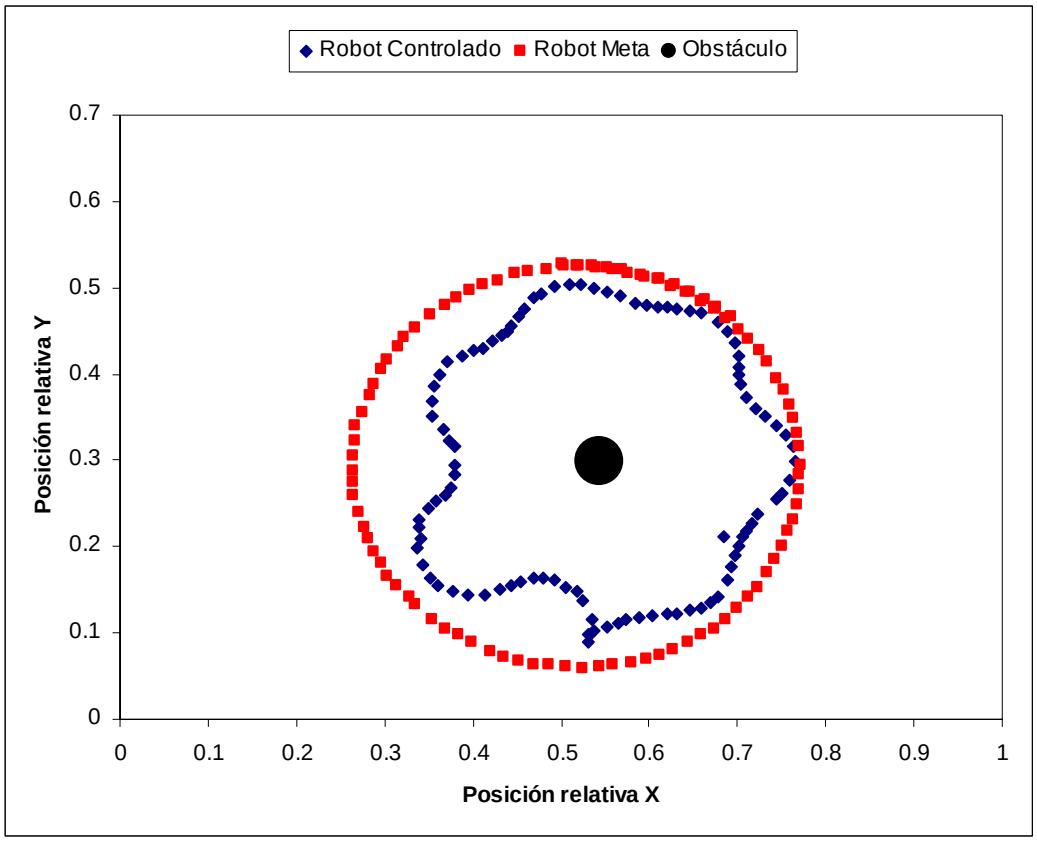


Figura 7.23 – Recorrido del robot (en azul) para “capturar la meta (en rojo) con obstáculo” (en negro)

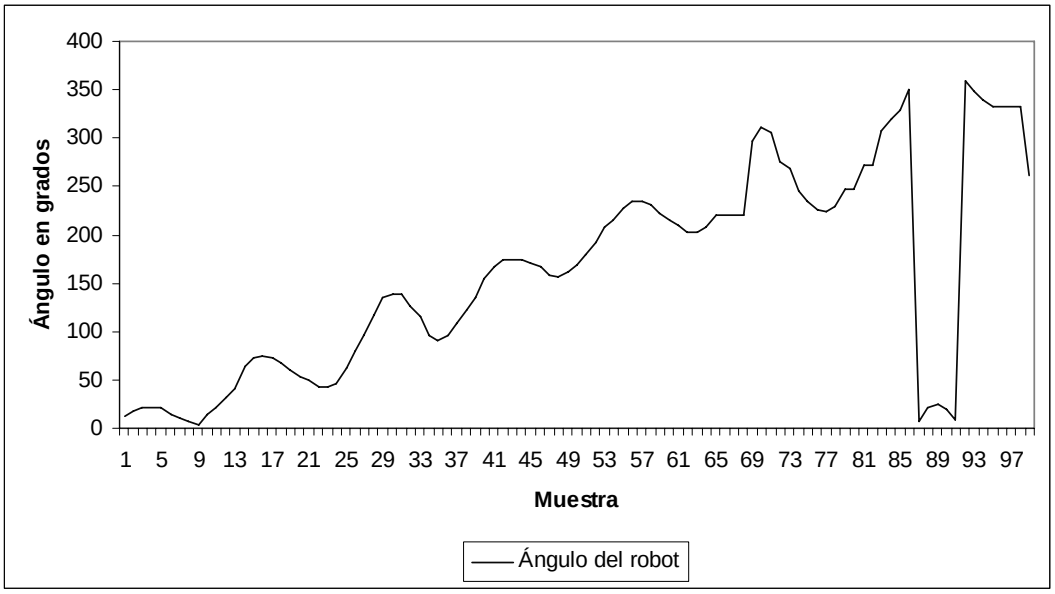


Figura 7.24 – Posición angular del robot controlado en la misión "capturar la meta con obstáculo"

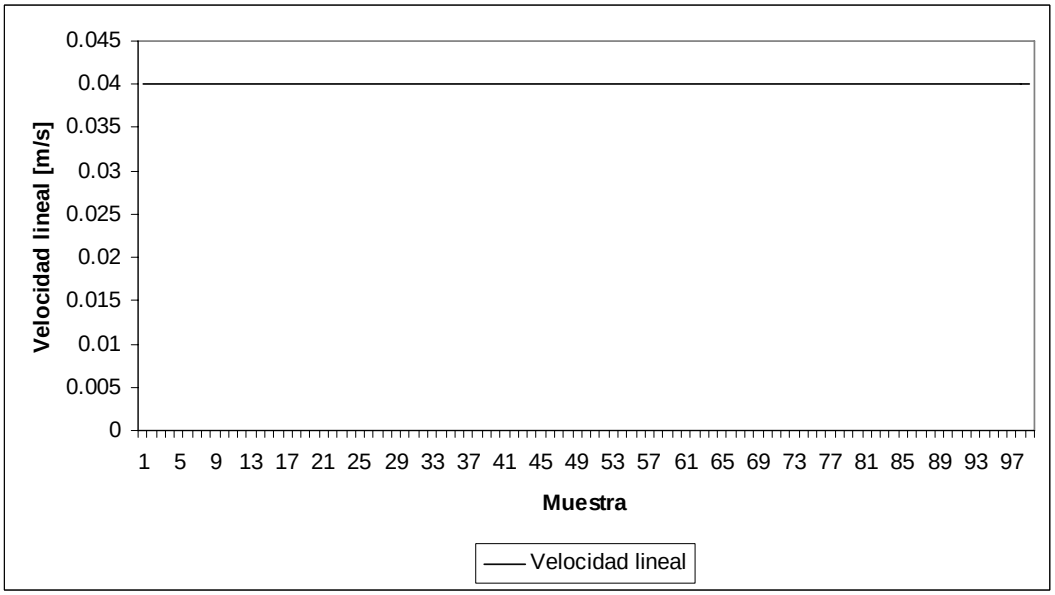


Figura 7.25 – Velocidad lineal del robot controlado en la misión "capturar la meta con obstáculo"

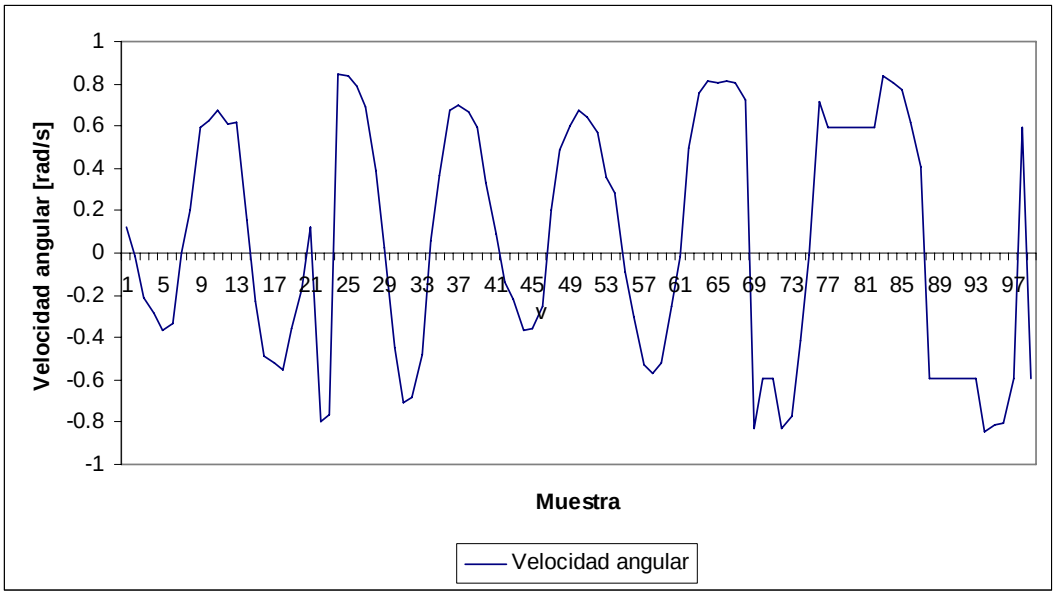


Figura 7.26 – Velocidad angular del robot controlado en la misión "capturar la meta con obstáculo"

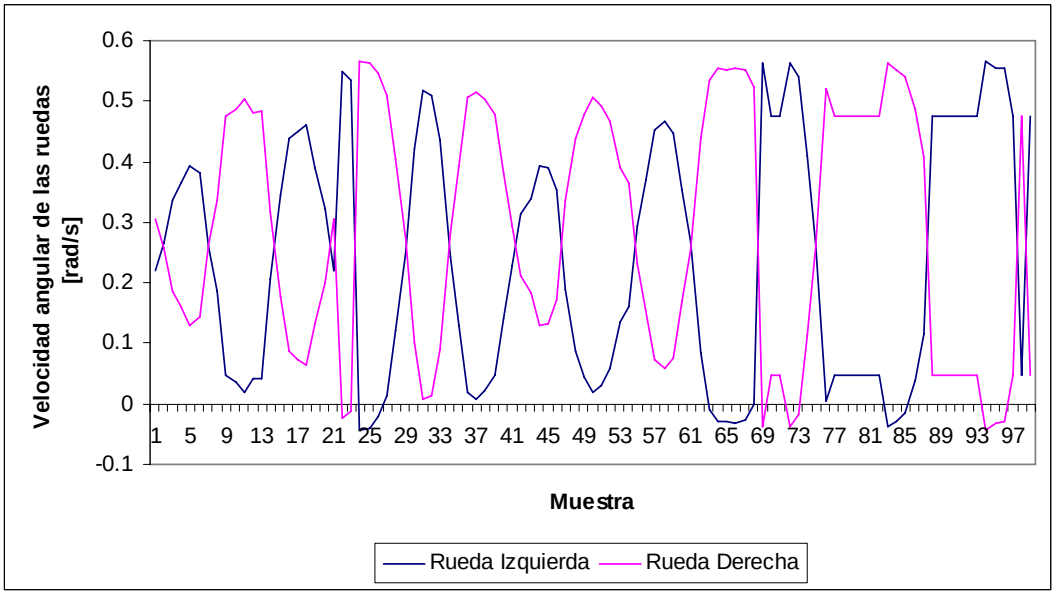


Figura 7.27 – Velocidad angular de las ruedas del robot controlado en la misión "capturar la meta con obstáculo"

Como se puede observar, la respuesta es más atenuada, además al colocar al robot controlado separado del robot meta por el obstáculo, se observa el comportamiento registrado, la trayectoria más eficiente para llegar al objetivo es cruzando por el medio, sin embargo, al encontrarse el obstáculo en el centro, se crea un efecto de "rebote"

Capítulo 8. Conclusiones y trabajo futuro

8. Conclusiones y trabajo futuro

8.1 Conclusiones

El objetivo del presente trabajo de crear un sistema para la localización y control de un robot móvil para el seguimiento de trayectorias en el plano utilizando retroalimentación visual se cumplió de acuerdo con el planteamiento inicial, como se muestra en los resultados experimentales. Cabe mencionar que el análisis y entendimiento de la problemática, junto con las soluciones propuestas, dan la pauta para poder diseñar e implementar sistemas más sofisticados y adecuados que respondan con mayor provecho a los requerimientos que pudieran plantearse en el futuro, esto se debe a que los algoritmos se diseñaron con la idea de migrar a otra plataforma robótica más flexible y completa.

El sistema desarrollado sirvió para resolver la prueba del robot móvil autónomo, el robot se colocó en un ambiente desconocido y dinámico, tras una exploración con la cámara el robot fue capaz de llegar a un punto seleccionado, buscando el camino más eficiente, en términos del comportamiento programado.

Se corroboró que la navegación a través de campos potenciales artificiales, es muy útil para dirigir un robot a través de entornos dinámicos en los que el objetivo puede ser móvil o donde la presencia de obstáculos obstruye el desplazamiento, además se enriqueció el modelo con la retroalimentación visual en tiempo real.

Se observa de los resultados experimentales, que el funcionamiento en conjunto de los componentes del sistema fue altamente satisfactorio, esta apreciación deriva del hecho de que todos los elementos diseñados e implementados funcionaron de la manera esperada, generando una respuesta adecuada para los parámetros establecidos, todos los elementos aquí planteados fueron implementados y probados con éxito.

El sistema desarrollado es sencillo, abierto y flexible, se convierte en una herramienta útil para la realización de pruebas y experimentos con robots móviles controlados con retroalimentación visual.

Como experiencia personal es muy gratificante llevar un proyecto, desde la idea hasta la implementación en un prototipo funcional, la experiencia me permitió desarrollar un proyecto que integra los conocimientos que adquirí a lo largo de mi carrera, además de permitirme culminar mi formación como Ingeniero Mecatrónico de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México, de la cual me siento sumamente orgulloso.

8.2 Trabajo Futuro

A corto plazo se tiene contemplado el trabajo del sistema con robots prototipo, desarrollados en el departamento de Mecatrónica de la Facultad de Ingeniería, en lugar del robot Lego Mindstorms NXT.

Sería muy productivo agregar un control de tipo PID digital al modelo de control para mejorar la velocidad de respuesta y disminuir el error en estado permanente.

La experiencia adquirida con el presente trabajo se piensa aprovechar para realizar el control coordinado de robots utilizando retroalimentación visual.

Se sugiere la implementación de este sistema para el control de manipuladores robóticos, abriendo la posibilidad de integrar sistemas móviles a los mismos para crear manipuladores móviles.

Se propone la utilización del sistema desarrollado para pruebas didácticas con distintos modelos de control de robots móviles o también la realización de control visual de robots utilizando técnicas de visión estereo.

Apéndices

Apéndice A – Tuio Tesis

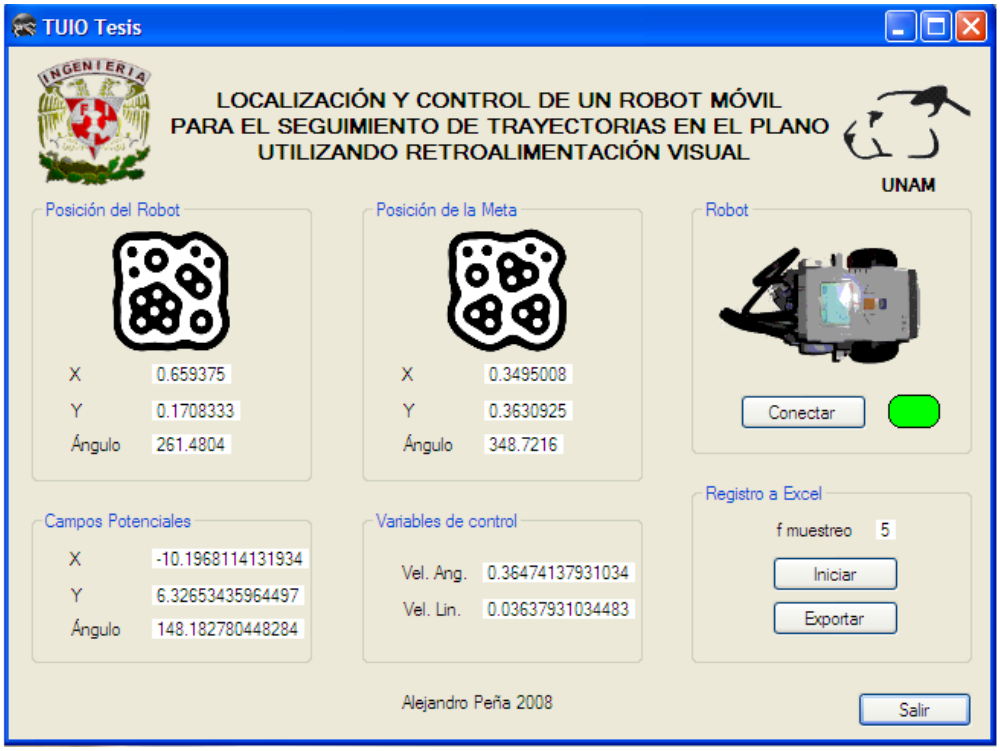


Figura A.1 – Diseño visual de la forma

Program.cs

```
**********************************************************************

//Programa que contiene la función principal del cliente Tuio Tesis y
//que inicia la forma ligada con el puerto de comunicación de
//ReacTIVision

//Referencias
using System;
using System.Collections.Generic;
using System.Windows.Forms;
using TUIO;

namespace TUIO_TESIS
{
    static class Program
    {

      //La entrada principal para el programa
        [STAThread]

      //Función principal, recibe cadena para puerto
        public static void Main(String[] argv)
        {
```

```
            int port = 0;
            //caso para elección de puerto
            switch (argv.Length)                                    {
                case 1:
                    port = int.Parse(argv[0], null);
                    if (port == 0) goto default;
                    break;
                case 0:
                    //se establece el cliente local con puerto 3333
                    port = 3333;
                    break;
                default:
                    Console.WriteLine("usando: java TuioDemo [port]");
                    System.Environment.Exit(0);
                    break;

            }

            //se inicia la aplicación ligada al puerto local
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new TuioTesis(port));

        }
    }
}
```

TuioTesis.cs

```
**********************************************************************
//Programa de la forma que trabaja como cliente de ReacTIVision, liga
//los valores leidos con los elementos del modelado, para generar una
//salida de acuerdo con los modelos propuestos y exporta los valores
//como elementos para el poserior análisis en Excel.

//Referencias
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Collections;
using TUIO;                                     //Utilizando TUIO
using LabVIEW;                                  //COM labVIEW
using Excel = Microsoft.Office.Interop.Excel;  //COM Office Excel
using System.Reflection;


//Inicia proyecto
namespace TUIO_TESIS
{

    public partial class TuioTesis : Form, TuioListener
```

```
{

    //Variables

    private TuioClient client; //cliente de Reactivision
    int nreg;                  //contador de registros
    float Xrob;                //variables de posición del robot
    float Yrob;
    float Arob;
    float Xgol;               //variables de posición del objetivo
    float Ygol;
    float Agol;
    float Xobs;              //variables de psoición del obstáculo
    float Yobs;
    double atrax, atray, anguloatr;      //variables de atracción
    double repx, repy;                   //variables de repulsión
    double errorangulo;
    double dg, dobs;        //distancia al obj y distancia al obs
    double vellinctrl;      //velocidad lineal de control
    double velangctrl;      //velocidad angular de control
    double velinx,veliny;  //velocidades proyectadas en los ejes
    double wi, wd;          //velocidad angular de las ruedas
    //arreglo del registo Xrob, Yrob, Arob, vellinctrl,
    velangctrl, Xgol, Ygol, Xobs, Yobs
    double[,] registros = new double[100, 9];
    //declaración de aplicación y VI con LabVIEW
    LabVIEW.Application lv;
    LabVIEW.VirtualInstrument vi;
    //declaración de aplicación Excel
    Excel.Application AplicacionExcel;

    //Banderas

    //bandera de conexión con el robot
    bool robotconectado = false;
    //bandera del registro con Excel
    bool registra = false;
    //Bandera de obstáculos
    bool obstaculos = false;

    //Consantes

    double Kr = 0.2;       //Constante de cercanía con el objetivo
    double E1 = 5, E2 = 10;  //Constantes de atracción
    double Ki = 0.5;       //Constante del radio de influencia 0.5
    double Nu = 0.35;             //Constante de repulsión
    double Wmax = 0.8;        //Velocidad angular máxima
    double Vmax = 0.04;        //Velocidad lineal máxima
    double b = 0.11;       //Distancia entre las ruedas
    double c = 0.029;       //Radio de la rueda
    double kprop = 2;   //constante de control proporcional


    //Inicia la forma ligada al puerto
    public TuioTesis(int port)
    {

        //definir cerrar la forma
        this.Closing += new CancelEventHandler(Form_Closing);

        //Inicia la forma
```

```
        InitializeComponent();

        //Conectar la forma con Reactivision
        client = new TuioClient(port);
        client.addTuioListener(this);
        client.connect();

    }


    //Se cierra la forma
    private void Form_Closing(object sender,
    System.ComponentModel.CancelEventArgs e)
    {

        //elimina la conexión TUIO
        client.removeTuioListener(this);
        client.disconnect();
        System.Environment.Exit(0);
        //cierra conexión robot
        if (robotconectado)
            vi.SetControlValue("stop", true);

    }


// Código del TuioListener


//Cuando se agregar un objeto fiducial
    public void addTuioObject(TuioObject o)
    {

        if (o.getFiducialID() == 0) //el fiducial 0 está asignado
                                    //al robot
        {
            Xrob = o.xpos;
            Yrob = (1-o.ypos) * 2 / 3;    //1 - para invertir el
                            //eje y por la proporción de la cámara
            Arob = o.getAngleDegrees();

        }

        else if (o.getFiducialID() == 1)    //el fiducial 1 está
                                            //asignado al objetivo
        {
            Xgol = o.xpos;
            Ygol = (1 - o.ypos) * 2 / 3;
            Agol = o.getAngleDegrees();
        }

            //si no es meta ni robot se considera obstáculo
        else if (o.getFiducialID() == 2)
        {
            obstaculos = true;
            Xobs = o.xpos;
            Yobs = (1 - o.ypos) * 2 / 3;
        }

    }
```

```
//Cuando cambia un objeto fiducial
public void updateTuioObject(TuioObject o)
{
    if (o.getFiducialID() == 0)        //robot
    {
        Xrob = o.xpos;
        Yrob = (1 - o.ypos) * 2 / 3;
        Arob = o.getAngleDegrees();
    }
    else if (o.getFiducialID() == 1)    //el fiducial 1 está
                                       //asignado al objetivo;
    {
        Xgol = o.xpos;
        Ygol = (1 - o.ypos) * 2 / 3;
        Agol = o.getAngleDegrees();
    }
        //obstáculos
    else if (o.getFiducialID() == 2)
    {
        Xobs = o.xpos;
        Yobs = (1 - o.ypos) * 2 / 3;
    }

}


//Desaparece un objeto fiducial
public void removeTuioObject(TuioObject o)
{
    if (o.getFiducialID() == 0)        //robot
    {
        Xrob = o.xpos;
        Yrob = (1 - o.ypos) * 2 / 3;
        Arob = o.getAngleDegrees();

    }
    else if (o.getFiducialID() == 1)    //el fiducial 1 está
                                        //asignado al objetivo;
    {
        Xgol = o.xpos;
        Ygol = (1 - o.ypos) * 2 / 3;
        Agol = o.getAngleDegrees();
    }
    else if (o.getFiducialID() == 2)
    {
        obstaculos = false;
        Xobs = o.xpos;
        Yobs = (1 - o.ypos) * 2 / 3;
    }

}

//Si se desea implementar un cursor

public void addTuioCursor(TuioCursor tuioCursor)
{
    throw new Exception("Por implementar");
}

public void updateTuioCursor(TuioCursor tuioCursor)
```

```
{
    throw new Exception("Por implementar");
}

public void removeTuioCursor(TuioCursor tuioCursor)
{
    throw new Exception("Por implementar");
}


//termina ciclo de update
public void refresh(long timestamp)
{

    //Actualiza etiquetas de la forma
    UpdateLblTxt(lblX, Convert.ToString(Xrob));
    UpdateLblTxt(lblY, Convert.ToString(Yrob));
    UpdateLblTxt(lblAngulo, Convert.ToString(Arob));
    UpdateLblTxt(lblXg, Convert.ToString(Xgol));
    UpdateLblTxt(lblYg, Convert.ToString(Ygol));
    UpdateLblTxt(lblAngulog, Convert.ToString(Agol));

    //ejecuta función de campos potenciales
    campospotenciales();

    //ejecuta función de salidas de control
    salidasdecontrol();

}


//Función que calcula los campos potenciales para las
//condiciones actuales
private void campospotenciales()
{
    //calcula la fuerza de atracción
    fuerzatraccion(Xrob, Yrob, Xgol, Ygol);


    //si existen obstáculos cálcula la repulsión
    if (obstaculos)
    {
        fuerzarepulsion(Xrob, Yrob, Xobs, Yobs);
        //actualiza los valores de campo
        atrax = atrax + repx;
        atray = atray + repy;
    }


    //calcula ángulo evitando division entre 0
    if ((atrax == 0) && (atray == 0)) anguloatr = 0;
    else if ((atrax == 0) && (atray > 0)) anguloatr = (Math.PI
    / 2);
    else if ((atrax == 0) && (atray < 0)) anguloatr = ((3 *
    Math.PI) / 2);
    else if ((atrax > 0) && (atray == 0)) anguloatr = 0;
    else if ((atrax < 0) && (atray == 0)) anguloatr = Math.PI;
    else if ((atrax != 0) && (atray != 0))
    {
        if ((atrax > 0) && (atray > 0)) anguloatr =
        Math.Atan(atray / atrax);
```

```
        else if ((atrax > 0) && (atray < 0)) anguloatr = (2 *
        Math.PI) + (Math.Atan(atray / atrax));
        else if ((atrax < 0) && (atray > 0)) anguloatr =
        (Math.PI) + (Math.Atan(atray / atrax));
        else anguloatr = (Math.PI) + (Math.Atan(atray /
        atrax));
    }
    else
    {
        anguloatr = Math.Atan2(atrax, atray);
    }

    //Convierte el ángulo a grados
    anguloatr = anguloatr * 180 / Math.PI;

    //Calcula el ángulo de error para alimentar el control
    errorangulo = anguloatr - Arob;
    //Para evitar valores negativos del ángulo
    if (errorangulo < 0) errorangulo = errorangulo + 360;

    //imprime los valores a la forma
    UpdateLblTxt(lblXpot, Convert.ToString(atrax));
    UpdateLblTxt(lblYpot, Convert.ToString(atray));
    UpdateLblTxt(lblAngulopot, Convert.ToString(anguloatr));

}


//Función para obtener la Fuerza de atracción basada en campos
potenciales artificiales
private void fuerzatraccion(float Xr, float Yr, float Xg,
float Yg)
{

    //calcula diferencia en cada uno de los ejes
    atrax = Xg - Xr;
    atray = Yg - Yr;

    //distancia al goal y magnitud del vector de atracción
    dg = Math.Sqrt((atrax * atrax) + (atray * atray));

    //fuera del radio atractivo?
    if (dg > Kr)
    {
        //Campo cónico
        // F =(E2 * vector atractivo) / magnitud del vector
        //atractivo
        atrax = (atrax * E2) / dg;
        atray = (atray * E2) / dg;
    }
    // dentro?
    else
    {
        //Campo parabólico
        // F = E1*vector atractivo
        atrax = (atrax * E1);
        atrax = (atrax * E1);
    }
}

//Función para obtener la fuerza repulsiva del campo
```

```
private void fuerzarepulsion(float Xr, float Yr, float Xo,
float Yo)
{
    //distancias lineales
    repx = Xo - Xr;
    repy = Yo - Yr;


    //distancia al obstáculo
    dobs = Math.Sqrt((repx * repx) + (repy * repy));

    //si se encuentra en el radio de influencia del obstáculo
    if (dobs < Ki)              {
        //Frep = -Nu * (1/dobs - 1/dg) * (1/(dobs*dobs)) *
        //(vector repusivo/ magnitud del vector repulsivo)
        repx = -Nu * (1 / dobs - 1 / dg) * (1 / (dobs * dobs))
        * (repx / dobs);
        repy = -Nu * (1 / dobs - 1 / dg) * (1 / (dobs * dobs))
        * (repy / dobs);
    }

    else            // no está dentro del radio de influencia
    {
        // No existe fuerza repulsiva
        repx = 0;
        repy = 0;
    }
}

//Función para calcular las salidas de control para el robot
private void salidasdecontrol()
{

    //Velocidad angular de acuerdo con el modelo
    if (errorangulo <= 90 || errorangulo >= 270)
    {
        velangctrl = Wmax * (Math.Sin(errorangulo * Math.PI /
        180));
    }

    //Corrección para mejor respuesta en ángulos fuera del
    //rango
    else if (errorangulo <= 180)
    {
        velangctrl = Wmax;
    }
    else
    {
        velangctrl = -Wmax;
    }


    //Para la velocidad lineal
    //Para limitar mejorar la respuesta en vueltas cerradas
    if (velangctrl >= 0.9*Wmax || velangctrl <= -0.9*Wmax)
    {
        vellinctrl = 0;
    }
    //aplicando el modelo
    else if (dg > Kr)
    {
```

```
        vellinctrl = Vmax;
    }
    else
    {
        vellinctrl = Vmax*dg/Kr;
    }

    //Proyeccion sobre los ejes de la velocidad
    velinx = velinctrl * Math.Cos(Arob * Math.PI / 180));
    veliny = velinctrl * Math.Sin(Arob * Math.PI / 180));


    //de acuerdo con el modelo de control, estableciendo un
    //control paramétrico proporcional
        wi = (-c*velinx * Math.Sin(errorangulo * Math.PI /
        180) + c*veliny * Math.Cos(errorangulo * Math.PI /
        180) – c/b * velangctrl) * kprop;
        wd = (-c*velinx * Math.Sin(errorangulo * Math.PI /
        180) + c*veliny * Math.Cos(errorangulo * Math.PI /
        180) + c/b * velangctrl) * kprop;
}


//para salir del hilo se crea un delegate generic sin
//argumentos
delegate void GenericDelegate();
// se crea  el metodo que invoke el cuerpo de nuestro método
//anónimo
void UpdateLblTxt(Label lbl, string text)
{
    GenericDelegate dlg = delegate()
    {
    //se aplica a la propiedad texto de etiqueta
        lbl.Text = text;
    };

    lbl.Invoke(dlg);
}


//Salir con el botón
private void btnSalir_Click(object sender, EventArgs e)
{
    //termina conexión
    if(robotconectado)
        vi.SetControlValue("stop", true);
    Close();
}


//Apretar botón conectar y desconectar robot
private void btnConectar_Click(object sender, EventArgs e)
{

    //Para conectar
    if (btnConectar.Text == "Conectar")
    {

        try                  //intenta conexión a LabVIEW
        {
```

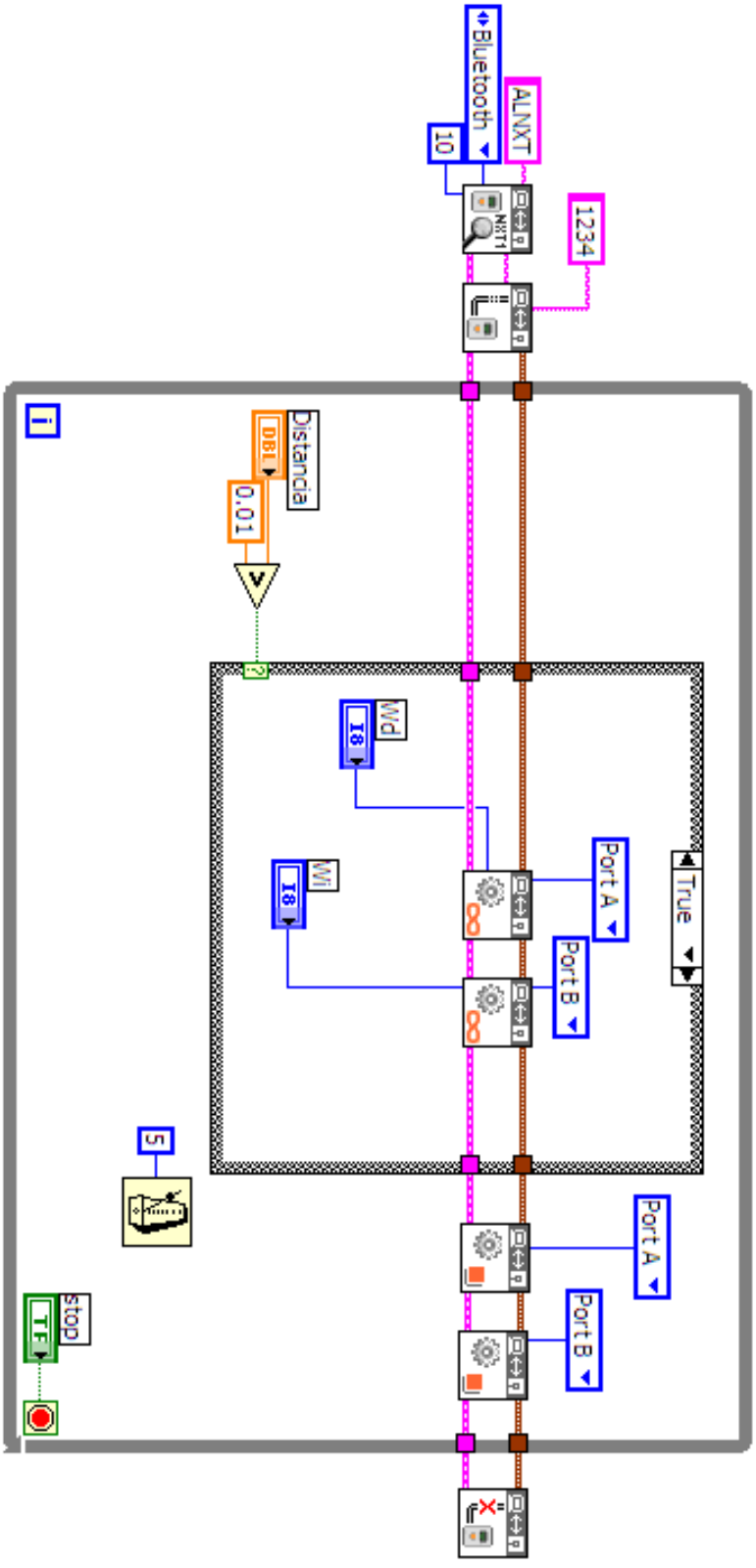
```
            //se abre aplicación de LabVIEW
            lv = new LabVIEW.ApplicationClass();

            //se indica el camino al VI
            string vipath = lv.ApplicationDirectory.ToString()
            + @"\tesis\controLego.vi";

            //se vincula con el VI
            vi = lv.GetVIReference(vipath, "", false, 0);

            //se inicia el VI de manera asincrona
            vi.Run(true);

            robotconectado = true;   //bandera de conexión
                                     //activada
            btnConectar.Text = "Desconectar";
                                     //Cambia texto

            //Manda valores al VI
            vi.SetControlValue("Wi", wi);
            vi.SetControlValue("Wd", wd);
            vi.SetControlValue("Distancia", dg);

        }

        catch (Exception error)     //cacha el error
        {

            robotconectado = false;  //bandera de conexión
                                     //desactivada
            btnConectar.Text = "Conectar";
            MessageBox.Show(error.ToString());
        }

    }
    else
    {
        vi.SetControlValue("stop", true);
        robotconectado = false;  //bandera de conexión
                                 //desactivada
        MessageBox.Show("Desconectado");
        btnConectar.Text = "Conectar"; //Regresa texto
    }

}



//Cuando cambia el texto actualiza datos del VI
private void lblVAngularTray_TextChanged(object sender,
EventArgs e)
{
    if (robotconectado) // si esta habilitada bandera
    {
        //manda variables de control a robot
        vi.SetControlValue("Wi", wi);
        vi.SetControlValue("Wd", wd);
        vi.SetControlValue("Distancia", dg);
    }
}
```

```
// control actualizado al VI
private void lblVelocidaTray_TextChanged(object sender,
EventArgs e)
{
    if (robotconectado)
    {
        //manda variables de control a robot
        vi.SetControlValue("Wi", wi);
        vi.SetControlValue("Wd", wd);
        vi.SetControlValue("Distancia", dg);
    }
}



//Inicia registro para Excel
private void btnInicioExc_Click(object sender, EventArgs e)
{
    //se activa la bandera de registro
    registra = true;
    timReg.Enabled = true;
    nreg = 0;                  //inicia el contador de reg
    // deshabilita exportar mientras registra
    btnExportar.Enabled = false;
}


//Para exportar los datos
private void bntExportar_Click(object sender, EventArgs e)
{
    try   //Intenta exportar
    {

          //se abre una nueva aplicación
          AplicacionExcel = new Excel.ApplicationClass();

          //se indica el camino hacia el archivo
          string workbookPath = "c:Tesis.xls";
          Excel.Workbook Librodetrabajo =
          //función para abrir el libro de trabajo
          AplicacionExcel.Workbooks.Open(workbookPath, 0,
          false, 5, "", "", false, Excel.XlPlatform.xlWindows,
          "", true, false, 0, true, false, false);

          //se selecciona la hoja en la que se va a escribir
          Excel.Sheets Hojaexcel = Librodetrabajo.Worksheets;
          string hojaactual = "Hoja1";
          Excel.Worksheet Hojadetrabajo =
          (Excel.Worksheet)Hojaexcel.get_Item(hojaactual);

          //se llena con un arreglo de valores registrados
          Hojadetrabajo.get_Range("A1", "I100").Value2 =
          registros;
          //cambiar en función del número de registros
          //se apaga bandera
          registra = false;

    }

    catch (Exception laException)  //cacha el error
```

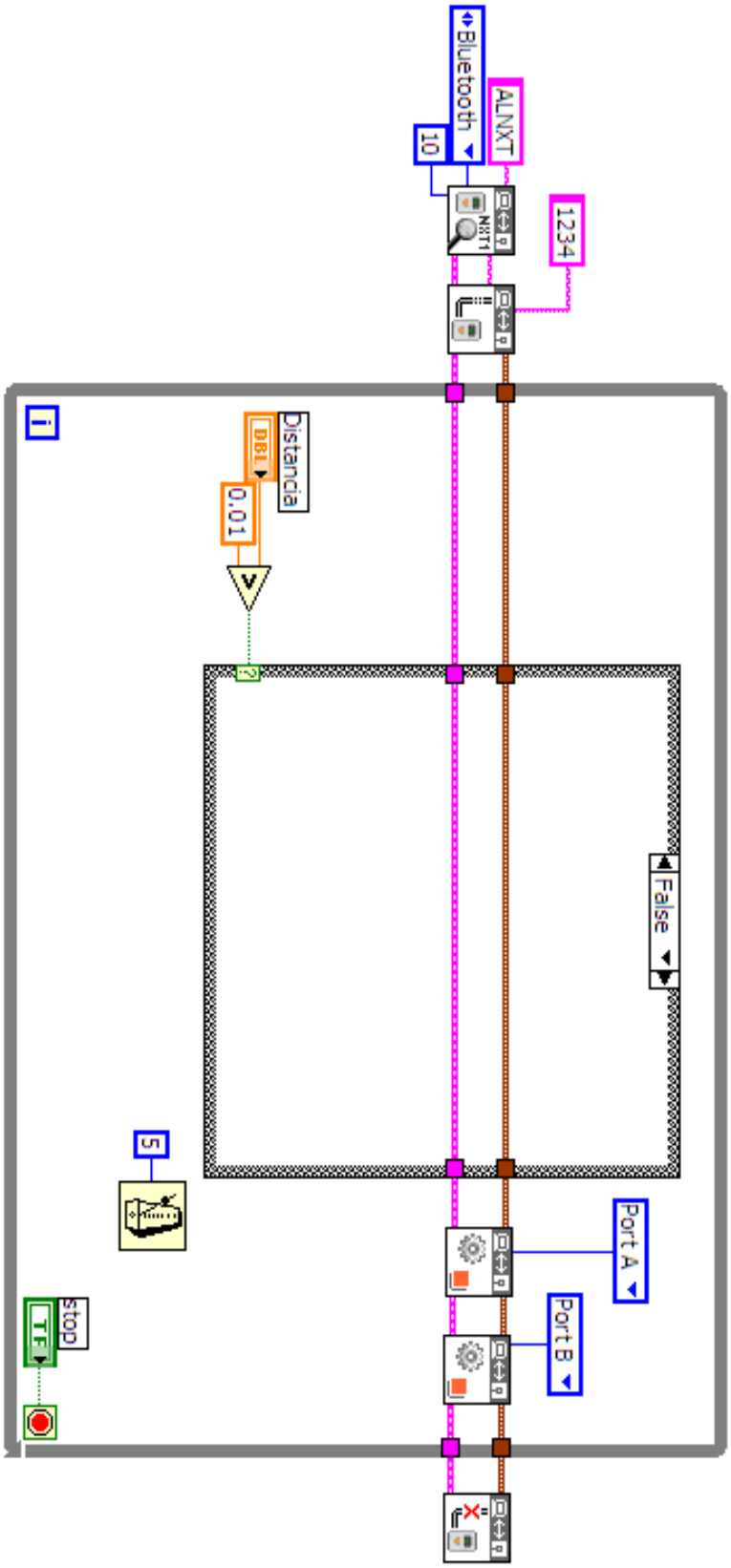
```
            {
                //lo convierte en cadena
                String errorMessage;
                errorMessage = "Error: ";
                errorMessage = String.Concat(errorMessage,
                laException.Message);
                errorMessage = String.Concat(errorMessage, " Line: ");
                errorMessage = String.Concat(errorMessage,
                laException.Source);
                //lo despliega en box
                MessageBox.Show(errorMessage, "Error");
            }

        }


        //Evento para el muestreo
        private void timReg_Tick(object sender, EventArgs e)
        {
            //Revisa si está registrando
            if (registra)
            {
                registros[nreg, 0] = Xrob;
                registros[nreg, 1] = Yrob;
                registros[nreg, 2] = Arob;
                registros[nreg, 3] = vellinctrl;
                registros[nreg, 4] = velangctrl;
                registros[nreg, 5] = Xgol;
                registros[nreg, 6] = Ygol;
                registros[nreg, 7] = Xobs;
                registros[nreg, 8] = Yobs;

                //termina registro e incrementa en 1 índice para
                //arreglo

                nreg = nreg + 1;

                if (nreg >= 99) //termina el muestreo
                {
                    registra = false;
                    timReg.Enabled = false;
                    //habilita exportar
                    btnExportar.Enabled = true;
                }
            }
        }

    }
}
```

Apéndice B – VI del robot NXT

Programa que realiza la conexión Bluetooth con el robot y controla las salidas a motores de acuerdo con las variables enviadas por el cliente.



En el caso negativo solo mantiene la conexión Bluetooth, pero no se envían salidas a motor



Apéndice C – Lego Mindstorms NXT

Las características de *brick* Inteligente del NXT son:

Procesador Principal: Atmel 32-bit ARM, AT91SAM7S256.

- 256 KB FLASH.
- 64 KB RAM.
- 48 MHz.

Co-procesador: Atmel 8-bit AVR, ATmega48.

- 4 KB FLASH.
- 512 Byte RAM.
- 8 MHz.

Comunicación Inalámbrica Bluetooth CSR BlueCoreTM 4 v2.0 +EDR System.

- Soporta comunicación serial (SPP).
- Interno 47 KByte RAM.
- Externo 8 MBit FLASH.
- 26 MHz.

USB 2.0 (12 Mbit/s).

4 puertos de entrada de 6-cables, soporta interfaz análoga y digital.

1 puerto de alta velocidad, IEC 61158 Tipo 4/EN 50170.

3 puertos de salida de 6-cables, interfaz soporta retroalimentación con encoders.

Display de 100 x 64 píxeles LCD blanco y negro.

- Área: 26 X 40.6 mm.

Salida a canal de speaker con 8-bits de resolución.

- Frecuencias 2-16 KHz.

4 botónes.

Conector 6-cables industrial standar RJ12.

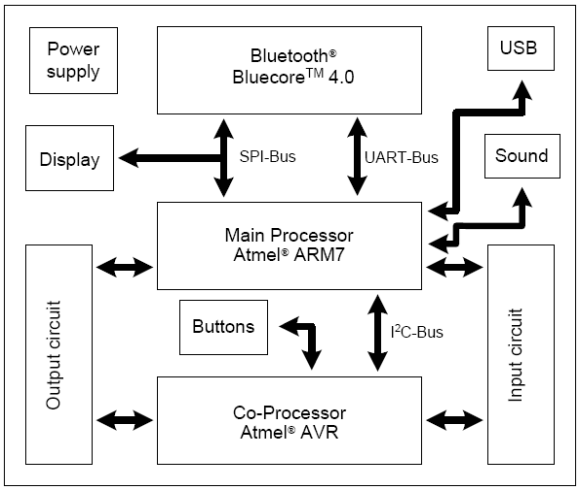


Figura A.2 – Arquitectura del Lego NXT

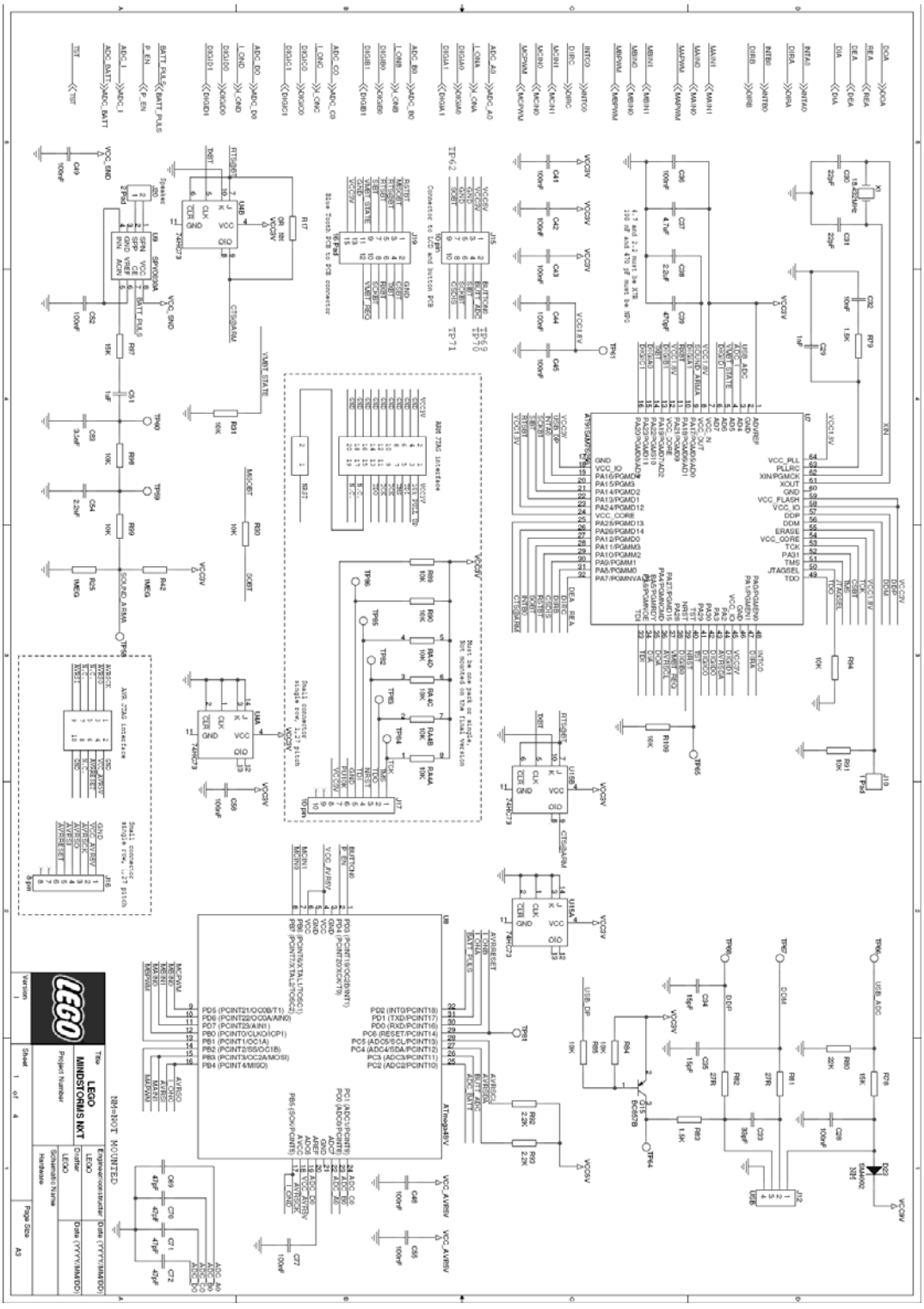


Figura A.3 – Lego Mindstorms NXT Hardware Schematic

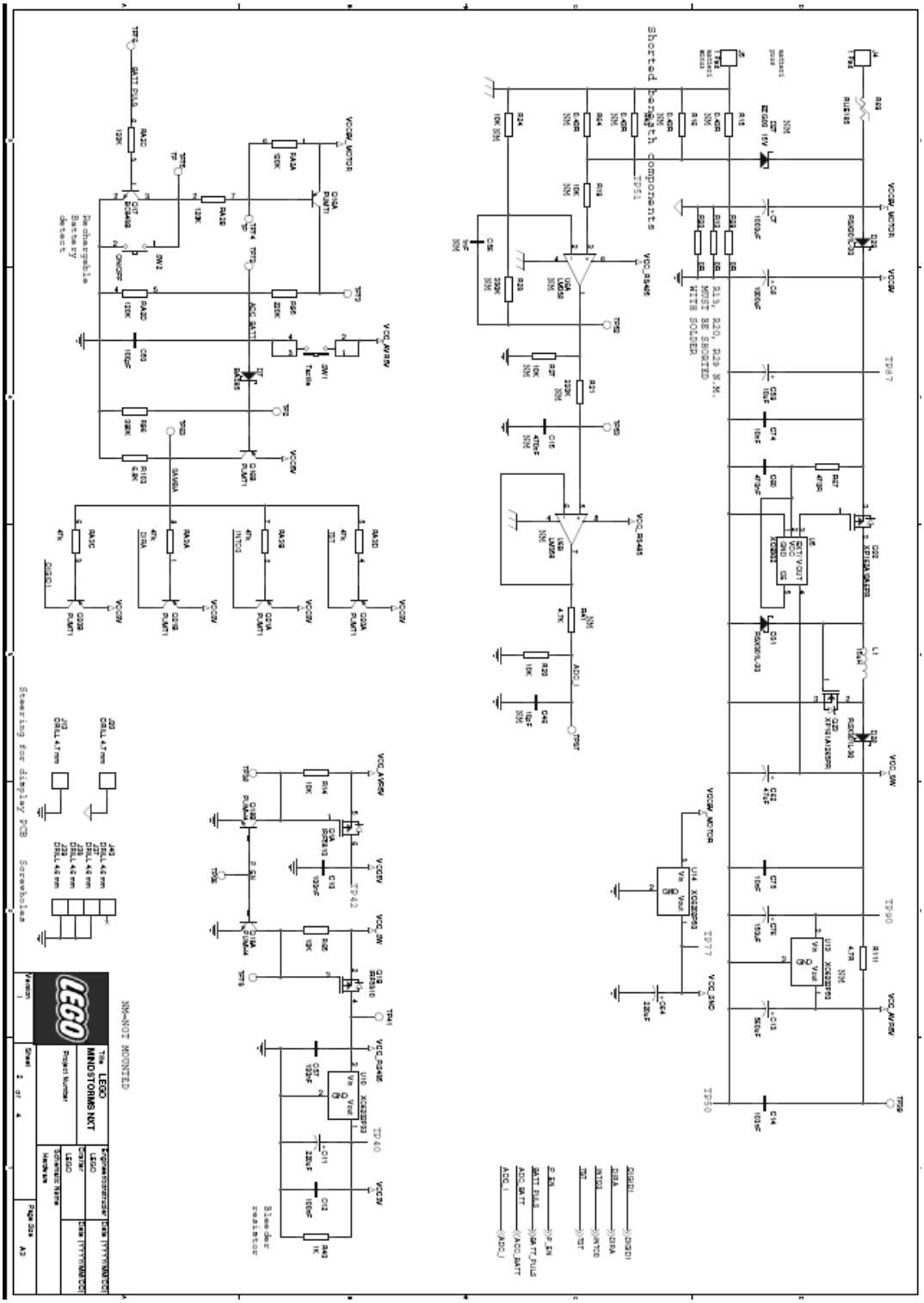


Figura A.4 – Lego Mindstorms NXT Hardware Schematic

Referencias

Bibliografía

- Arkin, R.C. “Behavior-Based Robotics.” MIT Press, Cambridge, Massachusetts, 1998.

- Bencina, R. & Kaltenbrunner, M. “ReacTIVision: A Computer-Vision Framework for Table- Based Tangible Interaction” 2007.

- Čapek, Karel, “Robots Universales de Rossum”, 1920.

- González-Villela, V.J., Parkin, R.M., Lopez-Parra, M., Dorador-González, J.M., et al., "A wheeled mobile robot with obstacle avoidance capability". Ingeniería Mecánica Tecnología y Desarrollo. Revista de la Sociedad Mexicana de Ingeniería Mecánica (SOMIM). 1(5): p. 159-166. ISSN: 1665-7381. 2004.

- Jordà, S. & Kaltenbrunner, M. & Geiger, G. &Bencina, R. “The reacTable*”, Proceedings of the International Computer Music Conference (ICMC2005), Barcelona (Spain), 2005.

- Kaltenbrunner, M. & Bovermann, T. & Bencina,R. & Costanza, E. “TUIO - A Protocol for Table Based Tangible User Interfaces”, Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes (France), 2005.

- Khatib, O., “Real-time obstacle avoidance for manipulators and mobile robots”. Internacional Journal of Robotics Research. Vol 5(1): p 90-98, 1986.

- Latombe J.C. “Robot Motion Planning”. Kluwer academic publishers. ISBN: 0-7923-9129-2, 1991.

- Lego Mindstorms NXT Hardware Developer Kit, The Lego Group 2006.

- Levi P. “Principles of Planing and Control Concepts for Autonomous Mobile Robots”. Proc. of 1987 IEEE International Conference on Robotics and Automation. pp 874-881, 1987.

- Lozano Pérez T. “Foreword: Mobile Robots and Robotics”. Autonomous Robot Vehicles. Editores I.J. Cox y G.T. Wilfong. Springer-Verlag. pp VII-XI. 1990.

- Nehmzow, “Scientic Methods in Mobile Robotics”, Springer. 2006.

- Ollero, A. Mandow, J. Gómez-de-Gabriel, V.F. Muñoz “Control Architecture for Mobile Robot Operation and Navigation”. Robotics & CIM, Vol 11. Elsevier Science 2000.

- Ollero, Anibal, “Robótica, manipuladores y robots móviles” Editorial Alfaomega, España 2007.

- Ollero, Anibal ”Planificación de Trayectorias para Robots Móviles” Tesis Doctoral, 1995.

- Salichs and L. Moreno “Navigation of mobile robots: open questions” United Kingdom, Cambridge University Press. Vol 18, pp. 227–234. 2000.

- Segovia, L. Martínez & J. Alanís,“TRASMAR: A Semi-autonomous robot for Irradiated Materials Transportation”, in Proceedings ofthe 3th International Symposium on Robotics and Automation ISRA’2002, Toluca, México, 2002.

Internet

Página del video de las pruebas documentadas en esta tesis (Octubre/08)

- http://www.youtube.com/watch?v=Gk8yTkBz6u8

Página para la descarga del código de reacTIVision (Febrero/08)

- http://www.sourceforge.net/projects/reactivision

Biblioteca Digital Universitaria, consulta de Tesis (de Enero/08 a Julio/08)

- http://bidi.unam.mx/

Campos potenciales artificiales (Junio/08)

- http://www.revista.unam.mx/vol.6/num1/art02/art02-2.htm

Robótica Móvil y planeación de trayectorias (Junio/08)

- http://webpersonal.uma.es/~VFMM

Robótica Móvil, proyecto Makro (Julio/08)

- http://www.sewerobots.de/en/r_makro.html

Descarga de librerías de Lego para LabVIEW (Julio/08)

- http://zone.ni.com/devzone/cda/tut/p/id/4435.

Información de Lego Mindstorms (Julio/08)

- http://www.lego.com/eng/education/mindstorms/home.asp?pagename=nxtc

Programación con C# y ActiveX (de Julio/08 a Octubre/08)

- http://support.microsoft.com/kb/302096
- http://support.microsoft.com/kb/302084
- http://www.eggheadcafe.com/forumpost.aspx?topicid=2&forumpostid=11691

- http://www.codeproject.com/KB/office/csharp_excel.aspx?fid=23997&df=90&mpp=25&noise=3&sort=Position&view=Quick&fr=26&select=2475791
- http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.7193
- http://www.experts-exchange.com/Programming/Languages/C_Sharp/