ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE ALGORITMOS DE DETERMINACIÓN DE RUTAS PARA EL ROBOTINO® DE FESTO

TESIS PREVIA A LA OBTENCIÓN DEL GRADO DE MAGÍSTER EN AUTOMATIZACIÓN Y CONTROL ELECTRÓNICO INDUSTRIAL

ING. VÍCTOR DANIEL ZAMBRANO PÉREZ
vic.zambrano.vdzp@gmail.com

DIRECTOR: NELSON SOTOMAYOR, MSc.
nelson.sotomayor@epn.edu.ec

QUITO, Junio 2015

DECLARACIÓN

Yo, Ing. Víctor Daniel Zambrano Pérez, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Ing. Víctor Daniel Zambrano Pérez

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el Ing. Víctor Daniel Zambrano Pérez, bajo mi supervisión.

_____

Nelson Sotomayor, MSc.
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A mi madre Rosario por brindarme su apoyo incondicional en todo momento de mi vida.

A mi padre Jorge por darme la oportunidad de culminar la carrera que desee.

A mis hermanos Miguel, María Fernanda y María José gracias al apoyo mutuo podemos sacar adelante cualquier dificultad que se nos presente en la vida.

Al MSc. Nelson Sotomayor, por creer en mis conocimientos y habilidades para desarrollar este proyecto.

Al MSc. Bolívar Ledesma y al Ing. Francisco Rosales por darme facilidades en mi trabajo para desarrollar y culminar esta Tesis.

DEDICATORIA

El presente proyecto, lo dedico a mis padres, y a mis hermanos por creer en mí y brindarme su apoyo incondicional.

CONTENIDO

RESUMEN

PRESENTACIÓN

CAPÍTULO 1

1 FUNDAMENTO TEÓRICO 1

1.1 ROBOT OMNIDIRECCIONAL 1

1.2 RUEDAS OMNIDIRECCIONALES 2

1.2.1 RUEDAS OMNIDIRECCIONALES SIMPLES 3

1.2.2 RUEDAS OMNIDIRECCIONALES DOBLES 3

1.2.3 RUEDAS MECANUM 4

1.2.4 RUEDA ESFÉRICA [1] 6

1.3 TIPOS DE ROBOTS OMNIDIRECCIONALES 7

1.3.1 ROBOT OMNIDIRECCIONAL DE TRES RUEDAS 7

1.3.1.1 Rovio [5] 8

1.3.1.2 Robotino® [6] 9

1.3.2 ROBOTS OMNIDIRECCIONALES DE CUATRO RUEDAS 10

1.3.2.1 Robot omnidireccional circular 10

1.3.2.2 Robot omnidireccional con ruedas orientables [7] 11

1.3.2.3 Robot omnidireccional Mecanum 13

1.4 MODELO MATEMÁTICO DE UN ROBOT MÓVIL 14

1.4.1 MODELO CINEMÁTICO DE UN ROBOT OMNIDIRECCIONAL [9] 14

1.4.2 MODELO MATEMÁTICO PARA ROBOT OMNIDIRECCIONAL DE TRES RUEDAS [9] 20

1.5 ROBOTINO® DE FESTO [10] 24

1.5.1 DESCRIPCIÓN GENERAL 25

1.5.2 MÓDULO DE LA UNIDAD DE ACCIONAMIENTO 27

1.5.3 UNIDAD DE CONTROL 28

1.5.4 CÁMARA 29

1.5.5 SENSORES Y DETECTORES 29

1.5.5.1 Sensores Infrarrojos 29

1.5.5.2 Encoders Incrementales 29

1.5.5.3 Sistema anticolisión 29

1.5.5.4 Sensor de proximidad inductivo 30

1.5.5.5 Sensor de reflexión 30

1.5.6 PUNTO DE ACCESO LAN INALÁMBRICO 31

2 DESCRIPCIÓN DE MÉTODOS PARA MAPEO DE AMBIENTES 32

2.1 NAVEGACIÓN PARA ROBOTS MÓVILES 32

2.1.1 MAPEO DE ENTORNO [12] 32

2.1.1.1 Rejillas de Ocupación 33

2.1.1.2 Mapeo por Características 33

2.1.1.3 Mapas Topológicos 34

2.1.1.4 Vector de percepción General 34

2.2 PLANIFICACIÓN DE RUTAS 35

2.3 MÉTODOS PLANIFICADORES DE RUTAS 36

2.3.1 MÉTODO GRAFO DE VISIBILIDAD [14] 36

2.3.1.1 Eliminación de Vértices Cóncavos [14] 39

2.3.1.2 Segmentación Tangencial [14] 39

2.3.1.3 Planificación basada en Subgrafos de Visibilidad [13] 40

2.3.1.4 Entorno Expandido [13] 41

2.3.2 PLANIFICACIÓN POR DESCOMPOSICIÓN DE CELDAS [14] 44

2.3.2.1 Descomposición de Celdas Exactas 45
2.3.2.1.1 Descomposición de Celdas Vertical (CHAZELLE, 1987) 46
2.3.2.1.2 Descomposición por Triangulación de Delaunay [14] 48
2.3.2.2 Descomposición de Celdas Aproximada o Adaptiva [14] 50
2.3.3 PLANIFICACIÓN POR VORONOI 53
2.3.3.1 Diagramas Voronoi 54
2.3.3.1.1 Línea de Barrido (Fortune 1987) [14] 56
2.3.3.1.2 Divide y Vencerás (Shamos y Hoey 1975) [14] 56
2.3.3.1.3 Inserción Incremental (Green y Sibson 1978) [14] 56
2.3.4 PLANIFICADOR POR MAPAS PROBABILÍSTICOS [14] 58
2.3.5 PLANIFICACIÓN POR CAMPOS POTENCIALES 61
2.3.5.1 Mínimos Locales 66
2.3.5.1.1 Eliminación de mínimos por movimientos aleatorios [14] 67
2.3.5.1.2 Eliminación por inclusión de cargas ficticias [11] 68
2.4 SEGUIMIENTO DE TRAYECTORIAS 70
2.4.1 GENERACIÓN DE CAMINOS 70
2.4.1.1 Interpolación Lineal [24] 71
2.4.1.2 Interpolación Polinómica [24] 72
2.4.1.3 Interpolación Spline [23] 73
2.4.2 FUNCIÓN TRAYECTORIA 74
2.4.3 PLANIFICACIÓN DE TRAYECTORIAS [21] 75
2.4.3.1 Restricciones físicas de velocidad 76
2.4.3.2 Restricciones operacionales de velocidad 77
2.4.4 CONTROL DE TRAYECTORIA POR MÉTODOS NUMÉRICOS [20] 78
3 DESARROLLO DEL PROGRAMA DE CONTROL 82
3.1 ESTRUCTURA DEL PROGRAMA DE CONTROL 84
3.2 PROGRAMACIÓN DE LA INTERFACE CON EL USUARIO 86
3.2.1 PANEL FRONTAL DEL HMI 87
3.2.2 DESARROLLO DEL DIAGRAMA DE FLUJO PRINCIPAL 90

3.2.3 HOMOLOGACIÓN DE DATOS 94

3.3 PROGRAMA DESCOMPOSICIÓN DE CELDAS 96

3.3.1 SUBRUTINA GENERADOR DE SÓLIDOS 100

3.3.2 SUBRUTINA ANALIZADOR DE RECTAS 104

3.3.3 REPRODUCCIÓN DEL GRAFO DE CONECTIVIDAD 106

3.3.4 DESCOMPOSICIÓN DE CELDAS MODIFICADO 108

3.3.5 GENERADOR DE LA RUTA MÁS CORTA 110

3.4 PROGRAMA GRAFO DE VISIBILIDAD 114

3.4.1 REPRODUCCIÓN DEL GRAFO DE CONECTIVIDAD 117

3.5 PROGRAMA CAMPOS POTENCIALES 120

3.5.1 SUBRUTINA GENERADORA DE SÓLIDOS 122

3.5.2 SUBRUTINA FUERZA DE ATRACCIÓN 126

3.5.3 SUBRUTINA FUERZA DE REPULSIÓN NETA 131

3.5.4 ELIMINACIÓN DE MÍNIMOS LOCALES 136

3.5.5 FILTRADO DE RUTA 140

3.6 PROGRAMA INGRESO MANUAL DE RUTA 143

3.7 PROGRAMA CONTROL DE RUTAS 145

3.7.1 CONFIGURACIÓN DEL ROBOTINO® DE FESTO 146

3.7.1.1 Configuración del Robotino® para la Conexión con el PC 147

3.7.1.2 Configuración de Encoders 149

3.7.1.3 Control de Motores 150

3.7.2 DIAGRAMA DE FLUJO DE CONTROL DE RUTA 151

3.7.3 CONTROL DE ROTACIÓN 155

3.7.4 CONTROL DE TRASLACIÓN 163

3.8 IMPLEMENTACIÓN DE CONTROL DE TRAYECTORIA 170

3.8.1 RUTINA CONSTRUCTOR DE TRAYECTORIA 173

3.8.2 SISTEMA DE CONTROL DE TRAYECTORIA 181

4 PRUEBAS Y RESULTADOS 187

4.1 ANÁLISIS DE LOS MÉTODOS PLANIFICADORES 187

4.1.1 PRIMERA PRUEBA, MAPA DE ENTORNO ESTILO LIBRE 188

4.1.1.1 Primera prueba aplicada a Grafo de Visibilidad 189

4.1.1.2 Primera prueba aplicada a Descomposición del Celdas 191

4.1.1.3 Primera prueba aplicada a Campos Potenciales 193

4.1.1.4 Análisis comparativo para la primera prueba 194

4.1.2 SEGUNDA PRUEBA, COMPLEJIDAD DE OBSTÁCULOS 196

4.1.2.1 Segunda prueba aplicada a Grafo de Visibilidad 197

4.1.2.2 Segunda prueba aplicada a Descomposición de Celdas 198

4.1.2.3 Segunda prueba aplicada a Campos Potenciales 199

4.1.2.4 Análisis comparativo para la segunda prueba 200

4.1.3 TERCERA PRUEBA, ENTORNO REAL 201

4.1.3.1 Tercera prueba aplicada a Grafo de Visibilidad 203

4.1.3.2 Tercera prueba aplicada a Descomposición de Celdas 205

4.1.3.3 Tercera prueba aplicada a Descomposición de Celdas Modificado 206

4.1.3.4 Tercera prueba aplicada a Campos Potenciales 208

4.1.3.5 Análisis comparativo para la tercera prueba 209

4.1.4 CUARTA PRUEBA, RENDIMIENTO EN FUNCIÓN DEL PROCESADOR 211

4.1.4.1 Prueba con Computador 1 213

4.1.4.2 Prueba con Computador 2 216

4.1.4.3 Prueba con Computador 3 217

4.1.4.4 Análisis comparativo de dependencia de procesador. 217

4.2 PRUEBA DE FUNCIONAMIENTO DE CONTROL DE RUTAS 219

4.3 PRUEBAS DE CONTROL DE RUTAS DEL ROBOTINO® EN ENTORNO REAL 222

4.3.1 PRUEBA DE CONTROL POR GRAFO DE VISIBILIDAD 222

4.3.2 PRUEBA DE CONTROL POR DESCOMPOSICIÓN DE CELDAS 228

4.3.3 PRUEBA DE CONTROL POR CAMPOS POTENCIALES 234

4.3.4 ANÁLISIS DE PRUEBAS DE CONTROL DE RUTAS 240

4.4 PRUEBAS DE CONTROL DE TRAYECTORIA DEL ROBOTINO® EN ENTORNO REAL 241

4.4.1 PRUEBA DE CONTROL DE TRAYECTORIA POR GRAFO DE VISIBILIDAD ……241

4.4.2 PRUEBA DE CONTROL DE TRAYECTORIA POR DESCOMPOSICIÓN DE CELDAS ……246

4.4.3 PRUEBA DE CONTROL DE TRAYECTORIA POR CAMPOS POTENCIALES ……249

4.4.4 PRUEBA DE CONTROL DE TRAYECTORIA CON CORRECCIÓN DE ERROR DE POSICIÓN INICIAL 252

5 CONCLUSIONES Y RECOMENDACIONES 256

5.1 CONCLUSIONES 256

5.2 RECOMENDACIONES 260

RESUMEN

En la actualidad el desafío que enfrenta la robótica móvil es la autonomía de los robots que hace poco eran necesarios en el campo industrial e investigación, pero con el pasar del tiempo llegó a incurrir la vida cotidiana de las personas como el servicio y el entretenimiento.

Debido a la capacidad que tienen los robots en realizar tareas repetitivas sin perder precisión y exactitud en el trascurso del tiempo, además de utilizarlos en ambientes peligrosos y de rescate, a un robot se puede implementar un algoritmo que cumpla con estos fines eliminando el factor humano.

El factor más importante de la autonomía es el desplazamiento seguro y confiable de los robots móviles en un entono ya sea conocido, desconocido, ó dinámico, para lo cual se desarrollaron varias técnicas de navegación, como la utilización de sensores ó directamente ingresar el entorno para que el robot ejecute sus movimientos.

Precisamente este proyecto va orientado a la investigación de características, funcionalidades, aplicaciones, ventajas y desventajas de métodos planificadores y su dependencia respecto a factores externos a ellos.

La programación de estos métodos planificadores tiene como objetivo analizar paso a paso el algoritmo del método planificador, y realizar modificaciones que mejoren su rendimiento en la obtención de trayectoria, para compararlos tanto en simulaciones, como en condiciones reales al ser ejecutados por un robot.

Otro objetivo es analizar el rendimiento de los métodos planificadores bajo circunstancias externas como el computador utilizado, el tiempo de ejecución el algoritmo, la complejidad del mapa de entorno y otros aspectos descritos más adelante.

Con estos requerimientos se pretende diseñar un programa que interactúe de forma sencilla con el usuario (HMI), en el ingreso de un mapa de entorno, configuración y calibración de parámetros, que tenga la capacidad de presentar toda la información necesaria, detallada acerca del rendimiento de los métodos planificadores de rutas, y el seguimiento de las mismas utilizando el Robotino® de Festo.

PRESENTACIÓN

El presente proyecto comprende capítulos descritos a continuación:

El capítulo 1 describe los aspectos más relevantes de la robótica móvil que tiene relación con el desarrollo de esta tesis, de los cuales se resaltan los tipos de robos con ruedas omnidireccionales, diseño estructural, características, movilidad, maniobrabilidad, orientación, posicionamiento, y representación matemática. Para concluir se explica las características y funcionalidades del Robotino® de Festo.

En el capítulo 2 se realiza un breve análisis de los métodos más difundidos y utilizados en el mapeo de ambientes, planificación de rutas, seguimiento de las mismas, y el estudio del control de trayectorias aplicadas a la robótica móvil.

En el capítulo 3 se desarrolla en una plataforma computacional (LabVIEW) los tres métodos planificadores anteriormente mencionados, se incluye la programación de una interface con el usuario (HMI), que tenga la capacidad de presentar la obtención de ruta a través del método planificador seleccionado en un mapa de entorno dibujado, y que pueda ser utilizada para realizar el seguimiento de la misma a través del Robotino® de Festo para ver las capacidades de velocidad del robot según la complejidad de la ruta.

En el capítulo 4, se realiza un análisis comparativo entre los métodos planificadores programados bajo distintas circunstancias, como número de obstáculos, complejidad de los mismos, tiempo de ejecución del método planificador, distancia de la ruta obtenida, movimientos de ejecución del robot, tiempo mínimo de recorrido del robot, errores obtenidos por cada método planificador, con el fin de ver cual método y bajo qué condiciones es la mejor opción para un mapa de entorno específico.

Las conclusiones y recomendaciones se presentan en el capítulo 5, y para concluir.

En los anexos, se incluye un manual de usuario del funcionamiento del HMI, en el cual se incluye la forma de ingreso del mapa de entorno y la ejecución de cada uno de los métodos planificadores. También se incluyen algoritmos de búsqueda de rutas, utilizados de complementos para los planificadores basados en grafos, y por último se presenta el plano y la representación en el HMI de un entorno real aplicado para las pruebas desarrolladas en el capítulo 4.

CAPÍTULO 1

FUNDAMENTO TEÓRICO

El presente capítulo explica las características, funcionalidades, clasificaciones, ejemplos y análisis matemático de los robots omnidireccionales que son parte de objeto de estudio de esta Tesis, para posteriormente controlar un robot omnidireccional comercial como parte del proyecto de planificación de rutas en base a un mapa de entorno conocido.

1.1 ROBOT OMNIDIRECCIONAL

En la robótica móvil, los robots omnidireccionales presentan una de las estructuras más eficientes para realizar movimientos frontales, laterales y rotacionales sobre una superficie plana, una de sus principales características es el tipo y la disposición de sus ruedas (Figura 1.1).

Un robot omnidireccional se define como un vehículo que cuenta con movilidad en cualquier dirección desde un punto arbitrario del sistema de coordenadas sin realizar rotaciones previas al desplazamiento, llegando a su destino con la orientación deseada (sistema holónomos). Esta condición se debe a que la disposición de las ruedas brinda al robot tres grados de libertad en el plano xy, movimiento en el eje x, y y orientación $\emptyset$ o también llamado guiñar (yaw).

Otra definición indica que “un sistema omnidireccional posee la misma cantidad de posicionamientos posibles que la cantidad de coordenadas necesarias para localizar un vehículo de forma única en un plano” [1].

El principal componente de los robots omnidireccionales son los actuadores que posibilitan su desplazamiento; en este caso es su sistema de motorización, transmisión, sus ruedas omnidireccionales y la distribución de las mismas en el chasis del vehículo.

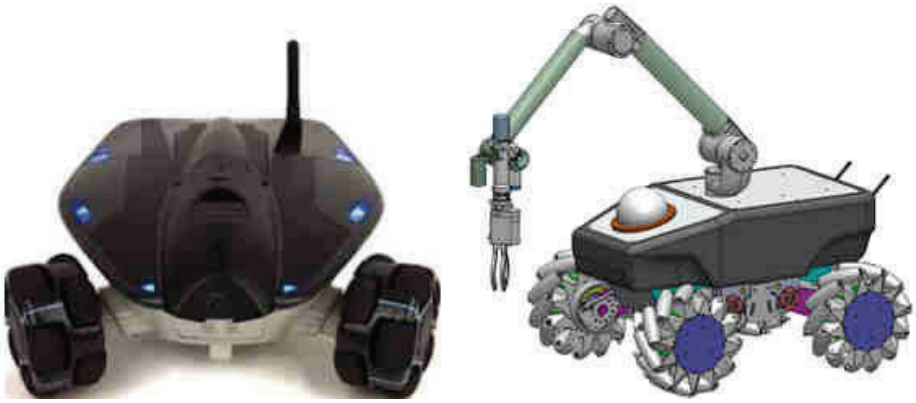


Figura 1.1 Estructura de un robot omnidireccional, [2]

1.2 RUEDAS OMNIDIRECCIONALES

Las ruedas omnidireccionales en todas sus configuraciones se basan bajo un mismo principio general, que tiene que ver con la inclusión de rodillos alrededor de su periferia, aumenta un grado de libertad adquiriendo movilidad en el eje perpendicular al sentido rodadura que tiene una rueda común [1].

Figura 1.2 Rueda omnidireccional universal, [1]

La Figura 1.2 muestra la configuración básica de una rueda omnidireccional, de la cual se desarrollan ruedas omnidireccionales más complejas que mejoran el

desplazamiento, el posicionamiento, la orientación, capacidad de carga así como reducir la complejidad del controlador.

1.2.1 RUEDAS OMNIDIRECCIONALES SIMPLES

La rueda omnidireccional simple (Figura 1.2) cuenta con una rueda principal que conforma la estructura, esta rueda produce el movimiento de avance igual que una rueda común, y para generar los movimientos laterales la rueda cuenta con rodillos colocados uniformemente en el perímetro de la rueda principal, los cuales cuentan con una curvatura para evitar rebotes en el movimiento de avance.

Este tipo de rueda tiene las siguientes características [1]:

- Debido a que los rodillos se encuentran en la periferia de la rueda principal, estos soportan todo el peso de robot limitando su capacidad de carga.

- Los tres grados de libertad que otorga esta rueda permite realizar movimientos del robot en cualquier dirección sin tener la necesidad rotaciones previas.

- Fricción baja, ya que es generada únicamente por el rodillo que entra en contacto con la superficie.

- Mayor sensibilidad a la superficie respecto de otros tipos de ruedas. Al desplazarse lateralmente, los rodillos se comportan de forma similar que una rueda convencional (Figura 1.2).

1.2.2 RUEDAS OMNIDIRECCIONALES DOBLES

Las ruedas omnidireccionales dobles son una variante de la estructura general de una rueda simple, presentando dos rodillos alternados en el perímetro de la rueda principal (Figura 1.3).

Figura 1.3 Ruedas omnidireccionales dobles, [1]

Esta rueda tiene las mismas características de la rueda omnidireccional simple, sin embargo tienen ventajas sobre su predecesora, principalmente la estabilidad de la rueda aumentando la superficie de contacto, y además tener mayor capacidad de carga.

1.2.3 RUEDAS MECANUM

Esta rueda cuenta con un diseño especial que logra tracción en la dirección de avance de la ruedas, y además permite generar movimiento lateral realizando el control de rotaciones sobre los actuadores de todas sus ruedas, permitiendo mayor flexibilidad en ambientes congestionados para robots móviles más sofisticados (Figura 1.4). Para ciertos movimientos ruedas Mecanum permiten al vehículo cambiar su dirección de movimiento sin cambiar su orientación [3].

Esta rueda fue desarrollada por el inventor sueco Bengt Llon de la empresa sueca Mecanum en el año de 1973.

Figura 1.4 Rueda omnidireccional Mecanum, [4]

La rueda Mecanum cuenta con una estructura que tiene una serie de ganchos alrededor del eje de la rueda, estos tienen normalmente una orientación de 45 grados respecto al eje de rotación de la rueda.

En cada gancho se ubica un eje con dos rodillos curvados, los cuales generan los desplazamientos omnidireccionales.

Esta disposición de los rodillos implica que la fuerza aplicada sobre el eje de la rueda principal se divide en dos componentes, uno perpendicular y otro paralelo al eje de los rodillos que varía según la orientación de los mismos.

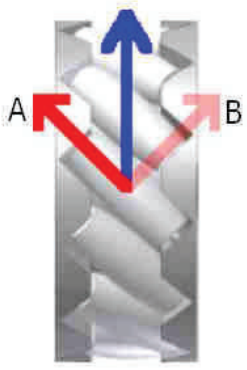


Figura 1.5 Componentes de la fuerza ejercida del motor sobre la rueda Mecanum, [1]

En la Figura 1.5 se observa que la suma de los dos componentes genera el movimiento de avance de la rueda La componente $\vec{A}$ no genera movimiento al rodillo por lo que brinda el movimiento al robot, mientras que la componente $\vec{B}$ proporciona el movimiento al rodillo produciendo su rotación.

1.2.4 RUEDA ESFÉRICA [1]

Como su nombre lo indica este tipo de rueda tiene una estructura esférica (Figura 1.6). Está rueda descansa sobre dos paneles atornillados entre sí, permitiendo estabilizarla.

Para controlar a esta esfera se utilizan varios motores, los cuales mediante fricción transfieren el movimiento a la rueda; esta disposición permite que la rueda tenga movilidad en todas las direcciones por medios activos.

Debido al método de transferencia del movimiento, esta rueda requiere motores de gran potencia capaces de romper la inercia de la rueda, adicionalmente la rueda requiere varios motores, y deben estar coordinados entre sí para ejecutar cada movimiento sin pérdidas de energía debido a posibles fricciones.

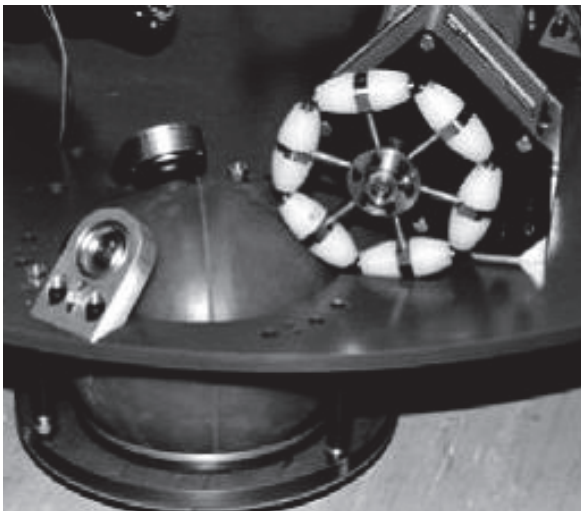


Figura 1.6 Rueda esférica, [1]

1.3 TIPOS DE ROBOTS OMNIDIRECCIONALES

Desde el punto de vista estructural y disposición de ruedas, los robots omnidireccionales se clasifican en robots de 3 y 4 ruedas, los últimos se subdividen en función del tipo de rueda omnidireccional que llevan, y la disposición de ellas en el chasis del robot.

1.3.1 ROBOT OMNIDIRECCIONAL DE TRES RUEDAS

El robot omnidireccional de tres ruedas, estructuralmente se compone de un chasis normalmente circular, cuya disposición de ruedas tiene una apertura de 120º entre sí con orientación tangencial al chasis del vehículo.

Las ruedas utilizadas son las omnidireccionales simples o las dobles, controladas por un motor cada una. El control es sencillo de implementar ya que para cada desplazamiento deseado por parte del robot existe sólo una combinación de velocidades aplicadas a los motores ó las ruedas.

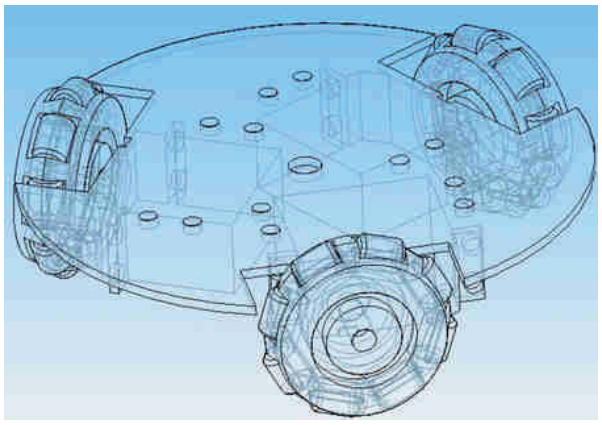


Figura 1.7 Estructura de un robot omnidireccional de tres ruedas, [3]

Esta configuración se aplica a vehículos pequeños debido al tipo de rueda omnidireccional utilizada.

Existen varios fabricantes que desarrollan este tipo de robot, entre los más conocidos se describen los siguientes:

1.3.1.1 Rovio [5]

ROVIO es un robot móvil con cámara web y capacidad de movimiento en cualquier dirección gracias a sus tres ruedas omnidireccionales dobles (Figura 1.8).

Este robot, está diseñado para la vigilancia del hogar a distancia. Se controla vía WiFi, y se puede mover de manera autónoma con el sistema TrueTrack de Northstar, que le permite seguir una serie de rutas predefinidas, con capacidad de recalcular su ruta pese a que la posición del robot cambie.

La cámara web incorporada tiene plena capacidad de audio y video streaming por lo que puede vigilar su casa desde cualquier lugar del mundo donde exista una conexión Web.

Utiliza el protocolo $802.11b/g$ para el acceso WiFi.

Rovio puede ser controlado desde un ordenador, una consola, un Smartphone que posea el acceso a Internet que permita darle instrucciones simples. Esta conjunción de características hace que controlarlo desde la Web sea muy fácil.

También posee el modo “patrulla”, con el cual el Rovio toma fotografías y las envía a lugares específicos que han sido previamente configurados, además posee un sistema de posicionamiento que permite que el robot conozca su ubicación en un ambiente previamente reconocido por sí mismo y pueda conocer trayectorias para su desplazamiento.

Figura 1.8 Robot Rovio, [5]

1.3.1.2 Robotino® [6]

El Robotino® es un robot móvil omnidireccional de tres ruedas desarrollado para el aprendizaje, formación y perfeccionamiento, e incluso una plataforma de investigación y desarrollo para escuelas técnicas en un mismo sistema.

Por su estructura modular, todos los componentes técnicos de Robotino®, los actuadores eléctricos, los sensores y la cámara, no sólo son comprensibles de inmediato, sino que también es fácil de estudiar el comportamiento de su sistema integrado presentado en la Figura 1.9.

Un elevado número de detectores, una cámara y un sistema de mando de alto rendimiento confieren al sistema la "inteligencia" necesaria. Si se programa adecuadamente, logra realizar de forma autónoma las tareas que le han sido asignadas o bien, sirve como plataforma de investigación y desarrollo para sistemas inteligentes.

Figura 1.9 Robotino® de Festo, [6]

1.3.2 ROBOTS OMNIDIRECCIONALES DE CUATRO RUEDAS

A diferencia de los robots omnidireccionales de tres ruedas que se limita a un tipo de disposición y dos tipos de ruedas omnidireccionales la simple y la doble, los robots de cuatro ruedas tienen más de una disposición de ruedas y utilizan más tipos de ruedas, adicionalmente se pueden instalar mecanismos activos que aumentan los grados de libertad de cada una de las ruedas y otros componentes que permitan al robot desplazarse sobre caminos irregulares. Entre los tipos de robots de cuatro ruedas se destacan los siguientes:

1.3.2.1 Robot omnidireccional circular

En este robot la disposición de las ruedas omnidireccionales es igual que la del robot de tres ruedas, estas tienen la apertura de 90º entre sí y orientadas tangencialmente al chasis del robot como se presenta en la Figura 1.10.

Este robot utiliza ruedas omnidireccionales simples o dobles, y cada una de las ruedas es controlada por un motor individual.

Una de las ventajas de este robot es la redundancia en el desplazamiento, ya permite llegar al mismo desplazamiento y orientación con varias combinaciones de velocidades de las ruedas omnidireccionales.

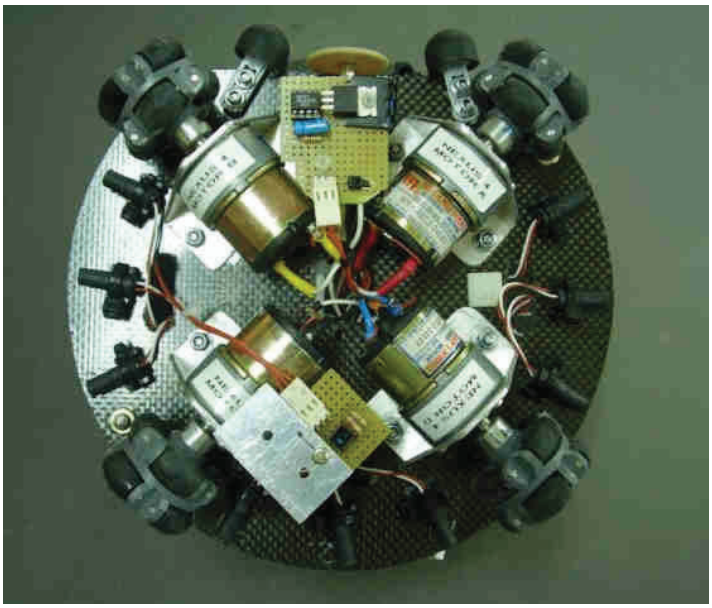


Figura 1.10 Robot omnidireccional de 4 ruedas, [1]

1.3.2.2 Robot omnidireccional con ruedas orientables [7]

El robot con ruedas orientables llamado ORM SOW (Figura 1.11), además de las ruedas omnidireccionales universales, tiene un sistema de motores acoplados a cada eje de la rueda proporcionándole un grado de libertad adicional en cada eje o también puede tener un sistema de dirección común mecánico del cual se logra un grado de libertad pero no en cada eje individual.

Su diseño permite incorporar el sistema de Transmisión continuamente variable (CVT) que le permite mayor maniobrabilidad y eficiencia en aspectos de consumo de energía que vehículos con tracción diferencial.

Este vehículo también tiene ventajas sobre sus antecesores omnidireccionales sobre todo los de tres ruedas, como la capacidad de desarrollar mayores velocidades sin perder estabilidad especialmente en superficies inclinadas.

Figura 1.11 Robot con ruedas orientables, [7]

Debido a su configuración el vehículo se puede considerar como un híbrido entre vehículo diferencial y omnidireccional tomando las capacidades de cada uno de ellos desarrollando las siguientes características:

- Movimiento en línea recta a velocidades altas como un vehículo diferencial, orientando las ruedas al eje paralelo.

- Movimientos omnidireccionales, solamente orientando las ruedas permitiéndole permanecer en su posición y orientación.

- Mayor capacidad de carga que los vehículos omnidireccionales de tres ruedas.

Este vehículo, así como los anteriormente mencionados tienen limitaciones, debido a las ruedas que utilizan, solo pueden desplazarse en superficies

regulares ya que pierden su capacidad omnidireccional en superficies irregulares dependiendo del radio de los rodillos.

1.3.2.3 Robot omnidireccional Mecanum

Como su nombre lo indica, es un robot de cuatro ruedas Mecanum cuya disposición es paralela al chasis. Esta distribución de las ruedas permite al vehículo tener la estructura del chasis más compleja incorporando sistema de amortiguación, barras de suspensión, etc.

Con estos elementos estos vehículos tienen la capacidad de movilizarse en caminos con superficies irregulares sin que esto altere la capacidad omnidireccional, además permite que estos vehículos tengan gran capacidad de carga y pueden desplazarse a velocidades mayores que otros robots omnidireccionales (Figura 1.12).

Este tipo de robots son utilizados para transportación en ambientes industriales que tienen caminos estrechos.

Figura 1.12 Vehículo omnidireccional Mecanum (prototipo de Mitsubishi Motors), [8]

1.4 MODELO MATEMÁTICO DE UN ROBOT MÓVIL

En el campo de la robótica, el modelo matemático de un sistema es una herramienta definitiva de diseño. El paso previo a la modelación de una representación matemática es conocer el entorno de trabajo, capacidad de carga, funcionalidades etc, para obtener ideas generales que llevan a determinar la complejidad del sistema y desarrollar el modelo, por otro lado permite diseñar la estructura, geometría componentes mecánicos móviles, distribución de masa, etc.

Un modelo matemático debe cubrir todos estos ámbitos, por lo cual debe incluir un estudio estructural, lo que implica hacer un análisis dinámico del sistema, conocer la variación de movimientos en función del entorno y de la carga adicional al cuerpo del robot. Este proyecto aborda la parte cinemática de los robots omnidireccionales para determinar los desplazamientos del robot en función de los movimientos de los actuadores.

1.4.1 MODELO CINEMÁTICO DE UN ROBOT OMNIDIRECCIONAL [9]

La cinemática se centra en el análisis del movimiento sin importar las causas que lo originan como fuerzas, rozamiento del suelo, etc. En la robótica móvil la cinemática analiza el desplazamiento de los robots en función de su geometría, tipos de ruedas, disposición de las mismas, articulaciones y engranajes.

Para el análisis del comportamiento cinemático de un robot móvil en general, se parte de la configuración más completa.

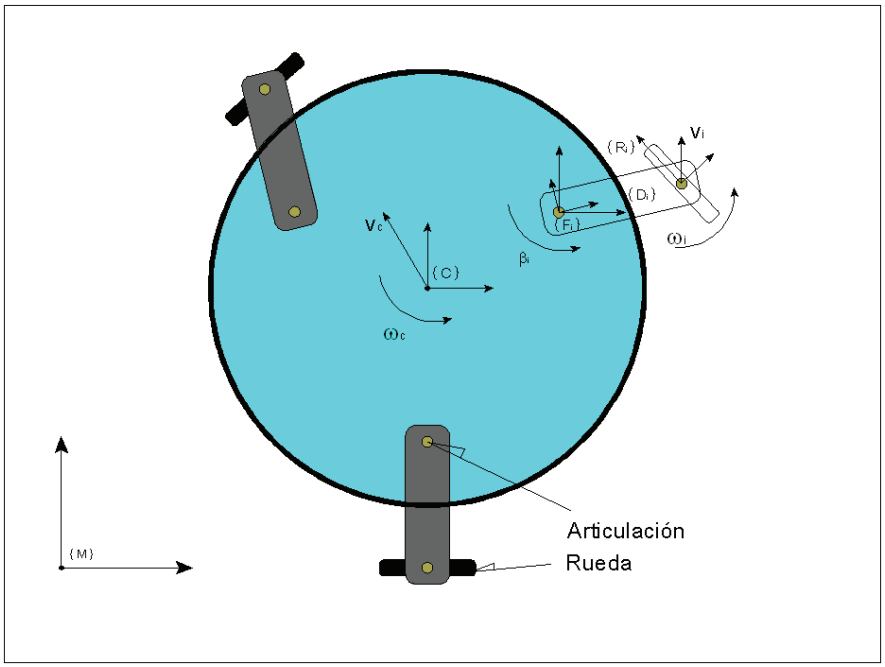


Figura 1.13 Estructura de un robot móvil omnidireccional, [9]

La Figura 1.13 presenta la estructura de un robot móvil circular, este robot tiene n ruedas, y cuenta con articulaciones que permite rotar la i-ésima rueda contra el chasis y del chasis a la articulación.

Para aislar cada componente del robot móvil, se coloca un sistema de referencia individual en cada articulación representado por la siguiente simbología:

- $\{\boldsymbol{M}\}$: Sistema de coordenadas Global; es la referencia de un observador externo al robot. (Ve los desplazamientos del Robot).

- $\{\boldsymbol{C}\}$: Sistema de coordenadas asociada al centro del robot; referencia que determina la posición y orientación del robot respecto al sistema global, además determina las posiciones de las articulaciones.

- $\{\boldsymbol{F_i}\}$: Sistema de coordenadas asociada al punto de anclaje de la articulación de la rueda i-ésima. (Referencia que determina la posición y orientación del anclaje respecto al centro del robot $\{\boldsymbol{C}\}$).

- $\{\boldsymbol{D_i}\}$: Sistema de coordenadas asociada al elemento de fijación de la i-ésima rueda. (Referencia que determina la posición y la orientación de la i-ésima rueda con respecto a la articulación $\{\boldsymbol{F_i}\}$).

- $\{\boldsymbol{R_i}\}$: Sistema de coordenadas asociado al punto de contacto del piso con la i-ésima rueda. (Referencia que determina la posición y orientación del piso respecto a la i-ésima rueda).

Cada sistema de referencia desglosado presenta los movimientos de la i-ésima rueda respecto al sistema de coordenadas global $\{\boldsymbol{M}\}$.

Otra consideración a tomar en cuenta es el tipo de rueda empleada, para un robot omnidireccional se toma la rueda que tenga tres grados de libertad (Figura 1.14).

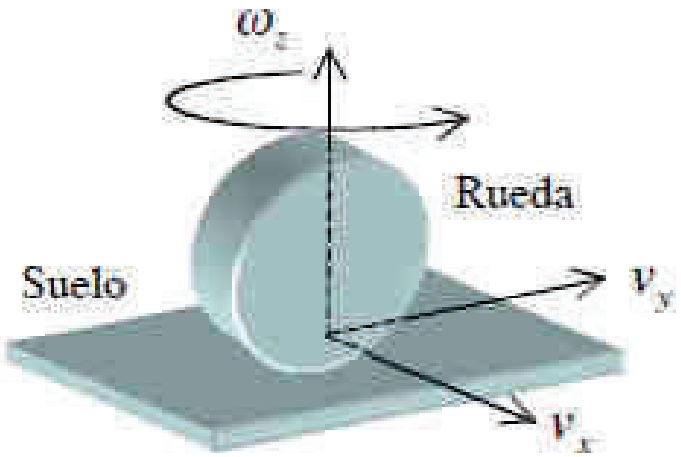


Figura 1.14 Rueda en contacto con el piso, [9]

De la Figura 1.14, se describen los movimientos que puede realizar la rueda:

El desplazamiento de avance de la rueda es representado por el vector $\boldsymbol{V_y}$, el desplazamiento lateral es representado por $\boldsymbol{V_x}$ (este componente existe en las ruedas omnidireccionales), y los deslizamientos rotacionales se representan con $\boldsymbol{\omega_z}$ que es la velocidad rotacional de la rueda, esta se utiliza cuando se produce giros generados por las articulaciones asociadas a la rueda.

Estos componentes de velocidad son referenciados respecto al sistema $\{\boldsymbol{R_i}\}$, es decir el contacto de la rueda respecto al piso.

Donde los movimientos se representan por (Figura 1.13):

- $\boldsymbol{v_i}$: Velocidad lineal generada por la rueda.

- $\boldsymbol{\omega_i}$: Velocidad angular generada por la rueda, respecto al sistema $\{\boldsymbol{R_i}\}$.

- $\dot{\boldsymbol{\beta}}_{\boldsymbol{i}}$: Velocidad angular de la articulación de la i-ésima rueda referida al sistema $\{\boldsymbol{D_i}\}$.

- $\boldsymbol{v_c}$: Velocidad del robot respecto al sistema $\{\boldsymbol{C}\}$.

- ω_c : Velocidad angular del robot respecto al sistema $\{\boldsymbol{C}\}$.

De acuerdo con [9], la velocidad del robot provocado por el sistema formado por la i-ésima rueda y su articulación está dada por:

$$\boldsymbol{v_c} = \boldsymbol{R}_i(\boldsymbol{\theta_i}).\boldsymbol{v_i} + \boldsymbol{\omega_i} x \boldsymbol{p_i} + \dot{\boldsymbol{\beta}}_{\boldsymbol{i}} x \boldsymbol{\lambda_i} \tag{1.1}$$

Donde:

- $\boldsymbol{R}_i(\boldsymbol{\theta_i})$: Es la matriz de rotación en el plano de referencia.

- $\boldsymbol{\lambda_i}$: es el vector posición respecto al sistema $\{\boldsymbol{C}\}$

- $\boldsymbol{p_i}$ y $\boldsymbol{\theta_i}$: son el vector posición y orientación respectivamente del sistema de coordenadas $\{\boldsymbol{R_i}\}$ respecto a $\{\boldsymbol{C}\}$, que viene dada por la sumatoria de la orientación de la i-ésima rueda:

$$\boldsymbol{p_i} = \boldsymbol{\lambda_i} + \boldsymbol{R}\left(\boldsymbol{\alpha_i} + \boldsymbol{\beta_i}\right).\boldsymbol{\delta_i}$$
$$\boldsymbol{\theta_i} = \boldsymbol{\alpha_i} + \boldsymbol{\beta_i} + \boldsymbol{\gamma_i} \tag{1.2}$$

Donde:

- γ_i y δ_i son la variación del ángulo de dirección y la posición respecto al sistema $\{\boldsymbol{R_i}\}$.

Por otro lado, la velocidad angular ($\boldsymbol{\omega_C}$) que provoca la orientación del robot respecto al sistema $\{\boldsymbol{C}\}$, es la diferencia entre las velocidades de la articulación $\dot{\boldsymbol{\beta}}_{\boldsymbol{i}}$, y la i-ésima rueda $\boldsymbol{\omega_i}$ correspondiente:

$$\boldsymbol{\omega_C} = \dot{\boldsymbol{\beta}}_{\boldsymbol{i}} - \boldsymbol{\omega_i} \tag{1.3}$$

Reescribiendo las expresiones anteriores en forma matricial, se obtiene:

$$\underbrace{\begin{pmatrix} v_{Cx} \\ v_{Cy} \\ \omega_C \end{pmatrix}}_{\boldsymbol{V_C}} = \underbrace{\begin{pmatrix} c_i & -s_i & p_{iy} & -\lambda_{iy} \\ s_i & c_i & -p_{ix} & \lambda_{ix} \\ 0 & 0 & 1 & -1 \end{pmatrix}}_{\hat{J}} \underbrace{\begin{pmatrix} v_{ix} \\ v_{iy} \\ \omega_i \\ \dot{\beta}_i \end{pmatrix}}_{\hat{\boldsymbol{q}}_{\boldsymbol{i}}} \tag{1.4}$$

$$\boldsymbol{V_C} = \hat{\boldsymbol{J}}\hat{\boldsymbol{q}}_{\boldsymbol{i}}$$

Donde:

v_{Cx}, y v_{Cy} son las componentes de v_C,

p_{ix}, y p_{iy} las de p_i;

λ_{ix}, y λ_{iy} las de λ_i; y

v_{ix}, y v_{iy} las de v_i

$$s_i = sen(\theta_i)$$
$$c_i = cos(\theta_i) \tag{1.5}$$

La velocidad lineal de la rueda se obtiene por la acción de giro proporcionada por el motor. Para una rueda orientable la matriz de conversión $\boldsymbol{\omega_i}$ se describe como:

$$\dot{\hat{\boldsymbol{q}}}_{\boldsymbol{i}} = \boldsymbol{W_i}\dot{\boldsymbol{q}}_{\boldsymbol{i}} = \begin{pmatrix} 0 & 0 & 0 \\ -r_i & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_{ix} \\ \omega_{iy} \\ \dot{\beta}_i \end{pmatrix} \tag{1.6}$$

Para calcular la velocidad del motor asociado a la i-ésima rueda $\boldsymbol{v_C}$, se desarrolla la matriz Jacobiana en función de las componentes de velocidad de la rueda $\dot{\boldsymbol{q}}_{\boldsymbol{i}}$, combinando la matriz de relación de velocidades del robot (1.4) con la matriz de velocidades de la i-ésima rueda (1.6), obteniendo:

$$\boldsymbol{J_i} = \hat{\boldsymbol{J}}_{\boldsymbol{i}}\boldsymbol{W_i} = \begin{pmatrix} r_i s_i & p_{iy} & -\lambda_{iy} \\ -r_i c_i & p_{ix} & \lambda_{ix} \\ 0 & 1 & -1 \end{pmatrix} \tag{1.7}$$

Esta matriz se aplica a ruedas orientables, para el caso de un robot que no tenga articulaciones la expresión se reduce a:

$$\boldsymbol{J_i} = \hat{\boldsymbol{J}}_{\boldsymbol{i}}\boldsymbol{W_i} = \begin{pmatrix} r_i s_i & p_{iy} \\ -r_i c_i & p_{ix} \\ 0 & 1 \end{pmatrix} \tag{1.8}$$

Tomando en cuenta todas las ruedas del robot, se incluyen los Jacobianos de relación de velocidades en una matriz para incluir las restricciones de todas las ruedas.

$$\begin{pmatrix} I \\ I \\ \vdots \\ I \end{pmatrix} \boldsymbol{V_C} = \begin{pmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \dots & \ddots & \ddots & \dots \\ 0 & \dots & 0 & J_N \end{pmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \vdots \\ \dot{q}_N \end{pmatrix} \tag{1.9}$$

$$\boldsymbol{A V_C} = \boldsymbol{B}\dot{\boldsymbol{q}}$$

Utilizando una aproximación por mínimos cuadrados se calcula la velocidad neta del robot:

$$\boldsymbol{V_C} = \underbrace{(\boldsymbol{A^T A})^{-1}\boldsymbol{A^T B}}_{J}\,\dot{\boldsymbol{q}} \tag{1.10}$$

$$\boldsymbol{V_C} = \boldsymbol{J}\dot{\boldsymbol{q}}$$

Donde:

- $\boldsymbol{J}$: es el Jacobiano que representa el modelo del robot

Definiendo la función $\Omega(\boldsymbol{A})$, que representa el deslizamiento del robot de cada una de sus ruedas:

$$\Omega(\boldsymbol{A}) = \boldsymbol{A}(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}} - \boldsymbol{I} \tag{1.11}$$

Considerando que el robot funciona en forma adecuada (sin deslizamientos) la expresión se reduce a:

$$\Omega(\boldsymbol{A})\boldsymbol{B}\dot{\boldsymbol{q}} = 0 \tag{1.12}$$

1.4.2 MODELO MATEMÁTICO PARA ROBOT OMNIDIRECCIONAL DE TRES RUEDAS [9]

De las expresiones obtenidas del modelo cinemático para un robot móvil omnidireccional de n ruedas en la sección anterior, se aplican a un robot de tres ruedas no orientables para determinar las velocidades aplicar a los motores para obtener las velocidades y desplazamientos deseados al robot, ya que es el tipo de robot es el selecciona para el control de este proyecto.

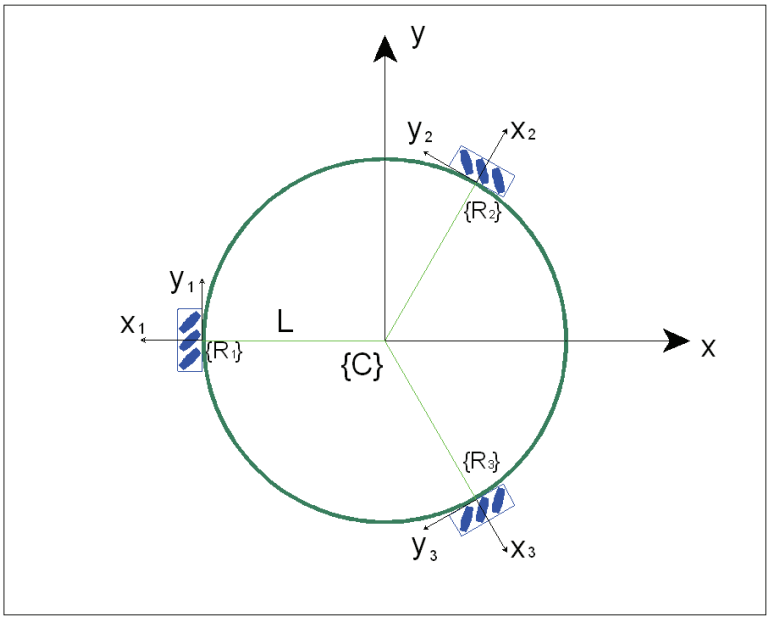


Figura 1.15 Estructura de un robot omnidireccional de 3 ruedas [9]

La Figura 1.15 representa la configuración más utilizada para los robots omnidireccionales en el campo de estudio, (Robotino® de Festo). El robot cuenta con una estructura circular de radio L, donde el centro del círculo es la referencia de todo el sistema $\{\boldsymbol{C}\}$. Las ruedas omnidireccionales están ancladas tangentes al chasis, con una apertura de 120º entre ellas, sin articulaciones para realizar los movimientos rotacionales (ruedas no orientables), lo cual elimina las componentes $\boldsymbol{\beta_i}$, $\boldsymbol{\gamma_i}$ y $\boldsymbol{w_z}$ de las tres ruedas e iguala los sistemas de referencia asociados a las articulaciones de las tres ruedas y del chasis del robot.

La Tabla 1.1 recopila las posiciones, orientaciones, y distribución de las tres ruedas omnidireccionales en función del radio de robot respecto al sistema $\{\boldsymbol{C}\}$.

Tabla 1.1 Parámetros de configuración cinemática, [9]

	Rueda 1	**Rueda 2**	**Rueda 3**
α_i	180^o	60^o	-60^o
β_i	0^o	0^o	0^o
γ_i	0^o	0^o	0^o
δ_i	$(0,0,0)$	$(0,0,0)$	$(0,0,0)$
λ_i	$(-L,0,0)$	$(\frac{L}{2},\frac{L\sqrt{3}}{2},0)$	$(\frac{L}{2},\frac{-L\sqrt{3}}{2},0)$

Para determinar la velocidad lineal y rotacional del robot primero se calcula el Jacobiano aplicado a las tres ruedas:

$$\boldsymbol{J_i} = \begin{pmatrix} R.s_i & r.c_i & \lambda_i \\ -R.c_i & r.s_i & -\lambda_i \\ 0 & 0 & 1 \end{pmatrix} \tag{1.13}$$

Donde:

- R: Radio de la rueda omnidireccional.

- r: Radio de los rodillos de la rueda omnidireccional.

- $S_i = Sen\theta_i$: Función seno de la orientación de la rueda respecto a $\{\boldsymbol{C}\}$.

- $C_i = Cos\theta_i$: Función coseno de la orientación de la rueda respecto a $\{\boldsymbol{C}\}$.

Entonces para la primera rueda el Jacobiano es:

$$\boldsymbol{J_1} = \begin{pmatrix} R.Sen180 & r.Cos180 & 0 \\ -R.Cos180 & r.Sen180 & L \\ 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{J_1} = \begin{pmatrix} 0 & -r & 0 \\ R & 0 & L \\ 0 & 0 & 1 \end{pmatrix} \tag{1.14}$$

Y para la rueda 2 y 3 los respectivos Jacobianos son:

$$\boldsymbol{J_2} = \begin{pmatrix} \frac{\sqrt{3}R}{2} & \frac{r}{2} & \frac{\sqrt{3}L}{2} \\ -\frac{R}{2} & -\frac{\sqrt{3}r}{2} & -\frac{L}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{1.15}$$

$$\boldsymbol{J_3} = \begin{pmatrix} -\frac{\sqrt{3}R}{2} & \frac{r}{2} & -\frac{\sqrt{3}L}{2} \\ -\frac{R}{2} & -\frac{\sqrt{3}r}{2} & -\frac{L}{2} \\ 0 & 0 & 1 \end{pmatrix} \tag{1.16}$$

Integrando los Jacobianos de cada rueda se obtiene la expresión total del sistema, para calcular la velocidad del robot con la aproximación de los mínimos cuadrados.

$$\boldsymbol{J} = \begin{pmatrix} 0 & -\frac{r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & -\frac{R}{a} & \frac{r}{6} & -\frac{L}{a} \\ \frac{R}{3} & 0 & \frac{L}{3} & -\frac{R}{6} & \frac{r}{a} & -\frac{L}{6} & -\frac{R}{6} & -\frac{r}{a} & -\frac{L}{6} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix} \tag{1.17}$$

Donde:

$$a = 2\sqrt{3}$$

Aplicando la condición de no deslizamiento (1.12):

$$\Omega(\boldsymbol{A})\boldsymbol{B}\dot{\boldsymbol{q}} = 0$$

$$\begin{pmatrix} 0 & \frac{2r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & -\frac{R}{a} & \frac{r}{6} & -\frac{L}{a} \\ -\frac{2R}{3} & 0 & -\frac{2L}{3} & -\frac{R}{6} & \frac{r}{a} & -\frac{L}{6} & -\frac{R}{6} & -\frac{r}{a} & -\frac{L}{6} \\ 0 & 0 & -\frac{2}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & -\frac{r}{3} & 0 & -\frac{R}{\sqrt{3}} & -\frac{r}{3} & -\frac{L}{\sqrt{3}} & -\frac{R}{a} & \frac{r}{6} & -\frac{L}{a} \\ \frac{R}{3} & 0 & \frac{L}{3} & \frac{R}{3} & -\frac{r}{\sqrt{3}} & \frac{L}{3} & -\frac{R}{6} & -\frac{r}{a} & -\frac{L}{6} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & -\frac{2}{3} & 0 & 0 & \frac{1}{3} \\ 0 & -\frac{r}{3} & 0 & \frac{R}{a} & \frac{r}{6} & \frac{L}{a} & \frac{R}{\sqrt{3}} & -\frac{r}{3} & \frac{L}{\sqrt{3}} \\ \frac{R}{3} & 0 & \frac{L}{3} & -\frac{R}{6} & \frac{r}{a} & -\frac{L}{6} & -\frac{R}{3} & \frac{r}{\sqrt{3}} & \frac{L}{3} \\ 0 & 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & -\frac{2}{3} \end{pmatrix} . \dot{q} = 0 \quad (1.18)$$

En el determinante se observa que existe combinación lineal entre las tres primeras filas con respecto a las siguientes, despejando las variables no actuadas en las ruedas se obtienen las siguientes expresiones:

$$\omega_{1r} = \frac{R(\omega_{3x} - \omega_{2x})}{\sqrt{3}r} \quad (1.19)$$

$$\omega_{2r} = \frac{R(\omega_{1x} - \omega_{3x})}{\sqrt{3}r} \quad (1.20)$$

$$\omega_{3r} = \frac{R(\omega_{2x} - \omega_{1x})}{\sqrt{3}r} \quad (1.21)$$

$$\omega_{1z} = -\frac{R(\omega_{1x} + \omega_{2x} + \omega_{3x})}{3L} \quad (1.22)$$

$$\omega_{2z} = \omega_{1z} \quad (1.23)$$

$$\omega_{3z} = \omega_{1z} \quad (1.24)$$

Donde:

- $\omega_{1x},\ \omega_{2x},\ \omega_{3x}$: Velocidades de los motores correspondientes a cada rueda.

- $\omega_{1r},\ \omega_{2r},\ \omega_{3r}$: Velocidades de los rodillos correspondientes a cada rueda.

- $\omega_{1z},\ \omega_{2z},\ \omega_{3z}$: Deslizamiento rotacional en el eje vertical de cada rueda.

Al sustituir el resultado de (1.19) en el Jacobiano completo las velocidades globales del robot en función de los actuadores son:

$$v_{Cx} = \frac{R(\omega_{2x}-\omega_{3x})}{\sqrt{3}} \tag{1.25}$$

$$v_{Cy} = \frac{R(2\omega_{1x}-\omega_{2x}-\omega_{3x})}{3} \tag{1.26}$$

$$\omega_C = -\frac{R(\omega_{1x}+\omega_{2x}+\omega_{3x})}{3L} \tag{1.27}$$

1.5 ROBOTINO® DE FESTO [10]

Figura 1.16 Robotino® de Festo, [10]

1.5.1 DESCRIPCIÓN GENERAL

Robotino® es un robot omnidireccional disponible en el mercado de la compañía FESTO DIDACTIC (Figura 1.16) utilizado para la educación, la investigación, entretenimiento, etc. Cuenta con una base omnidireccional, sensores de choques, sensores infrarrojos a distancia, y una cámara a color VGA. El diseño de Robotino® es modular, y puede ser fácilmente equipado con una variedad de accesorios, incluyendo sensores laser, giroscopios, y el sistema NORTHSTAR para el posicionamiento de interiores.

Tabla 1.2 Datos técnicos básicos del Robotino®, [10]

Parámetros	**Valor**
Alimentación	$24Vdc, 4.5\text{A}$
Entradas Digitales	8
Salidas Digitales	8
Entradas Analógicas	$8\ (0-10V)$
Salidas por relé	8

El controlador de Robotino® consiste en un PC embebido con una tarjeta compact flash. Las aplicaciones de demostración pueden ejecutarse directamente desde el panel de control del Robot (Figura 1.18).

Robotino® puede programarse con el software Robotino®View desde un PC. Robotino®View es capaz de transmitir señales al controlador del motor, así como visualizar, cambiar y evaluar valores de los sensores.

La Webcam permite visualizar y evaluar una imagen de cámara en tiempo real con ayuda del Robotino®View. Con ello, pueden implementarse aplicaciones tales como el trazado de rutas y seguimiento de objetos.

Puede accederse al controlador directamente a través conexión inalámbrica (WLAN). Cuando está correctamente programado, Robotino® realiza de forma autónoma las tareas asignadas.

Adicionalmente pueden conectarse actuadores y sensores adicionales a través de una interface de E/S.

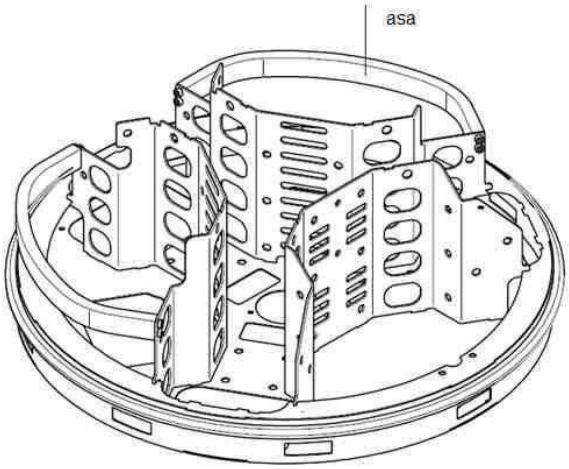


Figura 1.17 Chasis del Robotino®, [10]

En el chasis (Figura 1.17) se sitúan las baterías, las unidades de accionamiento, la cámara Web, los sensores de medición de distancia y los sensores para detección de objetos (Figura 1.18).

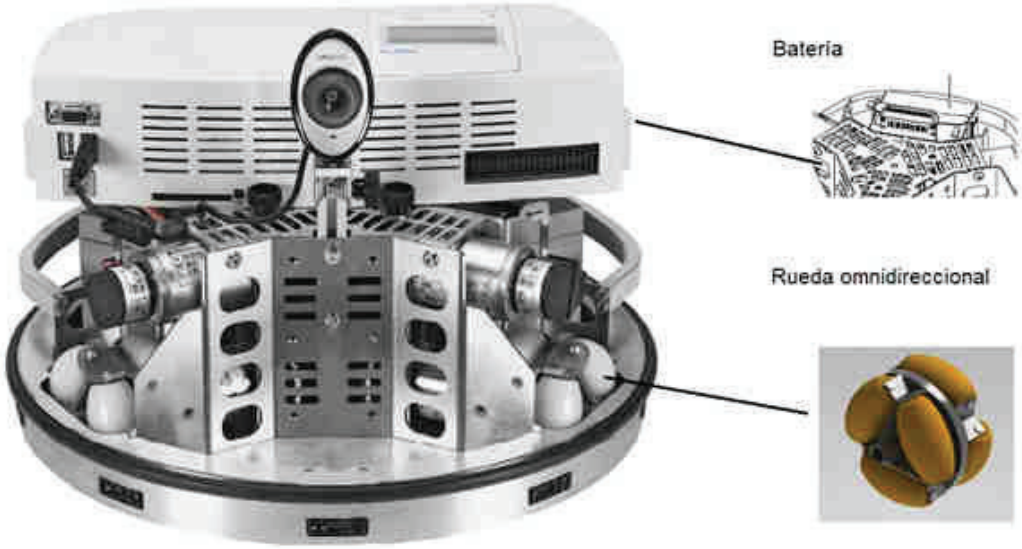


Figura 1.18 Unidad de Accionamiento y Batería, [10]

Los componentes muy sensibles del sistema, tales como el controlador, el módulo de E/A y las interfaces, se hallan situados en el puente de mando (Figura 1.19).

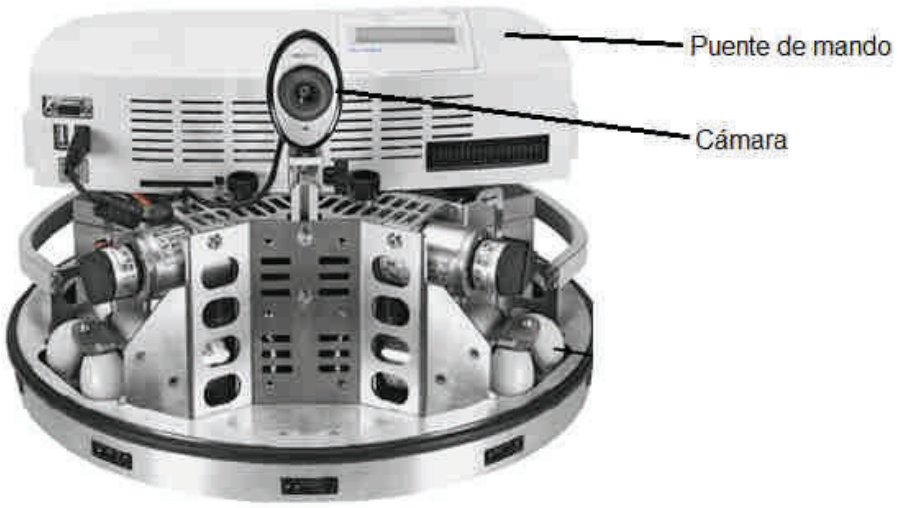


Figura 1.19 Puente de Mando y Cámara, [10]

1.5.2 MÓDULO DE LA UNIDAD DE ACCIONAMIENTO

Robotino® es accionado por 3 unidades de accionamiento omnidireccionales independientes (Figura 1.20). Se hallan montadas formando un ángulo de 120^o entre sí.

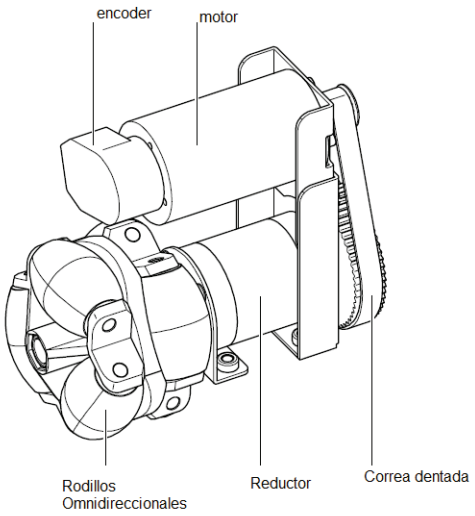


Figura 1.20 Sistema de propulsión del Robotino®, [10]

La velocidad real del motor puede compararse con la velocidad deseada por medio del encoder incremental, y puede regularse con un control PID a través de la placa de circuito de E/S.

Cada motor desarrolla una velocidad de $3600RPM$ con un par de arranque de $20Nm$, en condiciones normales ($24V_{dc}$ y $0.9A$).

1.5.3 UNIDAD DE CONTROL

El controlador del Robotino® consta de tres componentes:

Procesador PC 104, compatible con MOPSIcdVE de $300MHz$, y sistema operativo Linux con kernel en tiempo real, SDRAM $128MB$.

- Tarjeta Compact flash ($256MB$) con API C++ para controlar el Robotino®.

- Punto de acceso LAN inalámbrico.

- La unidad de control está equipada con los siguientes interfaces (Figura 1.21):

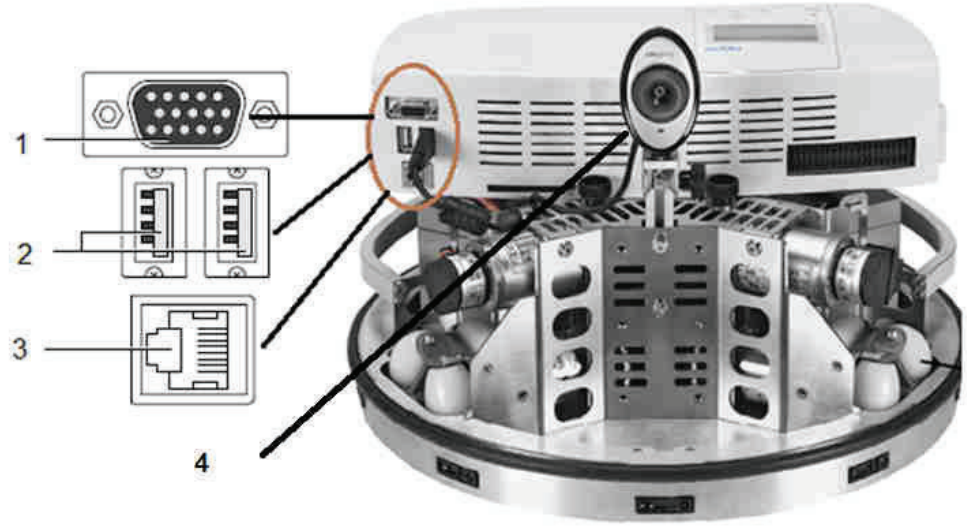


Figura 1.21 Puertos de Comunicación (1) Zócalo conector VGA, (2) Puerto USB, (3) Interface ETHERNET, (4) cámara, [10]

Una ETHERNET, 2 USB, y una VGA, Estas se utilizan para conectar un teclado, un ratón y una pantalla. Con ello puede accederse al sistema operativo y a la librería C++ sin un PC.

1.5.4 CÁMARA

La cámara permite visualizar imágenes de hasta $1024x768$ y generar videos con una resolución máxima $640x480$ a $15fps$. Con la ayuda del Robotino®View se realiza el procesamiento de imágenes, que puede utilizarse para evaluar imágenes (Figura 1.21).

1.5.5 SENSORES Y DETECTORES

El Robotino® tiene integrado sensores para medición de distancia a objetos, velocidad de los motores, orientación del robot y sensores para detección objetos.

1.5.5.1 Sensores Infrarrojos

Robotino® está equipado con nueve sensores de medición de distancia por infrarrojos, que se hallan montados en de chasis formando un ángulo de 40^0 entre sí. Los sensores son capaces de medir distancias con precisión o relativas a objetos, con valores entre 4 y $30cm$.

1.5.5.2 Encoders Incrementales

Los encoders incrementales miden la velocidad de los motores del Robotino® en RPM. A estos encoders se los puede ajustar para realizar control de la velocidad del Robotino®.

1.5.5.3 Sistema anticolisión

El sistema anticolisión del Robotino®, está compuesto por una banda de impacto formada por dos superficies conductoras fijada alrededor de un aro que circunda

el chasis (Figura 1.22) que entran en contacto cuando se aplica una mínima presión a la banda, generado una señal que ingresa a la unidad de control del robot que provoca la detención del Robotino®

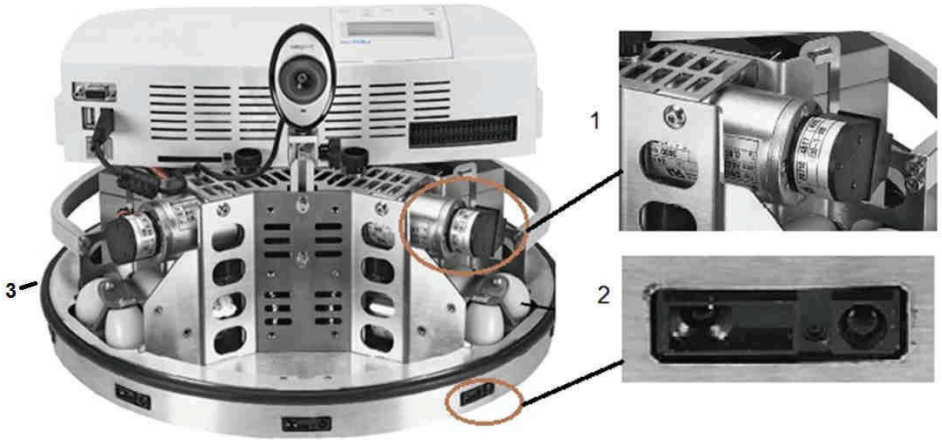


Figura 1.22 Distribución e identificación de los Sensores (1) Encoder, (2) Sensores de Medición de Distancia, (3) Banda de impacto [10]

1.5.5.4 Sensor de proximidad inductivo

El sensor inductivo de proximidad es un componente adicional que sirve para detectar objetos metálicos en el suelo y se utiliza para el control de filo guiado. El sensor emite señales de diferente intensidad dependiendo de si se halla en el medio o en el borde de una tira metálica. Con ello puede controlarse el recorrido de forma perfectamente diferenciada. El sensor envía una señal analógica entre $0V$ y $10V$ para un margen de detección de $0mm$ a $6mm$

1.5.5.5 Sensor de reflexión

El seguimiento de una ruta también puede ser implementado con los dos sensores de reflexión directa (de luz difusa) incluidos. Los cables flexibles de fibra óptica se conectan a una unidad óptica que funciona con luz roja visible. Se

detecta la luz reflejada. Diferentes superficies y colores producen diferentes grados de reflexión. Sin embargo, no se puede detectar diferencias graduales en la luz reflejada.

1.5.6 PUNTO DE ACCESO LAN INALÁMBRICO

El punto de acceso LAN inalámbrico es un componente que permite la configuración con el robot por medio de una dirección de red.

El punto de acceso cumple con los estándares: IEEE $802.11g$, y $802.11b$.

Permite velocidades de transmisión de hasta $54Mbps$ para $802.11g$, y $11Mbps$ para $802.11b$ con un amplio alcance de las transmisiones (hasta $100m$ dentro de edificios).

Permite establecer una red segura con encriptación WEP y función WPA-PSK.

Es rápida y simple de configurar a través de la utilidad de gestión de la web.

CAPÍTULO 2

DESCRIPCIÓN DE MÉTODOS PARA MAPEO DE AMBIENTES

2.1 NAVEGACIÓN PARA ROBOTS MÓVILES

El objetivo de la navegación es otorgar a un robot, o un vehículo autónomo, la habilidad a desplazarse desde un punto arbitrario hasta un destino de forma segura en un entorno que contenga obstáculos.

La navegación de un robot móvil, involucra varios pasos ó procedimientos que tiene que ver con el mapeo del entorno, como la localización del robot, la modelación del mapa de entorno a partir de medios sensoriales, la planificación de rutas libres de obstáculos, y el seguimiento de la misma aplicada a un vehículo que ejecute los movimientos necesarios para recorrer el mapa donde se encuentre [11].

2.1.1 MAPEO DE ENTORNO [12]

Son técnicas que tienen de objetivo representar a un mapa de entorno en algoritmos matemáticos, geométricos, ó en bases de datos que puedan ser procesados.

Para aplicar estas técnicas previamente se recolecta la información del entorno que se quiere modelar, actualmente se utiliza robots para adquirir esta información por medio de sensores, cámaras, etc.

De la información recogida y en función del tipo se sensores utilizados se puede modelar un mapa de entorno. Existen varias técnicas que se describen a continuación.

2.1.1.1 Rejillas de Ocupación

Está técnica discretiza el mapa de entorno en celdas de igual tamaño, y tiene la particularidad de que a cada celda se asigna la probabilidad de la presencia de un obstáculo donde 1 representa que la celda está ocupada.

0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5
0.5	0.5	0.9	0.3	0.3	0.3	0.3	0.3	0.3	0.3	1.0	0.5
0.5	0.5	0.9	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.9	0.5
0.5	0.5	0.8	0.2	0.2	0.2	0.2	0.2	0.2	0.2	1.0	0.5
0.5	0.5	0.9	0.2	0.2	0.9	0.8	0.8	0.7	0.3	1.0	0.5
0.5	0.5	0.8	0.1	0.1	0.9	0.2	0.2	0.2	0.2	0.8	0.5
0.5	0.5	0.9	0.2	0.3	1.0	0.2	0.1	0.2	0.3	0.9	0.5
0.5	0.5	0.9	0.2	0.2	1.0	1.0	0.2	0.2	0.2	1.0	0.5
0.5	0.5	0.8	0.2	0.2	0.2	1.0	1.0	0.1	0.1	0.9	0.5
0.5	0.5	0.9	0.2	0.3	0.2	0.3	1.0	0.2	0.3	0.9	0.5
0.5	0.5	0.8	0.2	0.2	0.2	0.3	1.0	0.2	0.2	1.0	0.5
0.5	0.5	0.9	0.1	0.1	0.1	0.2	0.2	0.2	0.3	1.0	0.5
0.5	0.5	0.8	0.3	0.3	0.9	1.0	0.9	0.9	0.9	0.5	0.5
0.5	0.5	1.0	1.0	1.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

Figura 2.1 Rejillas de Ocupación, [12]

El algoritmo, parte de que en todas las rejillas tienen la probabilidad del 50% que exista un obstáculo (Figura 2.1). Cuando el robot circula a través del mapa, se recalcula la probabilidad con la información recolectada por sus sensores de que exista un obstáculo de acuerdo a la ubicación actual del robot.

2.1.1.2 Mapeo por Características

Esta técnica de mapeo se basa en la modelación de los obstáculos del mapa de entorno utilizando características paramétricas [12], representadas por vectores y nodos ortogonales (Figura 2.2).

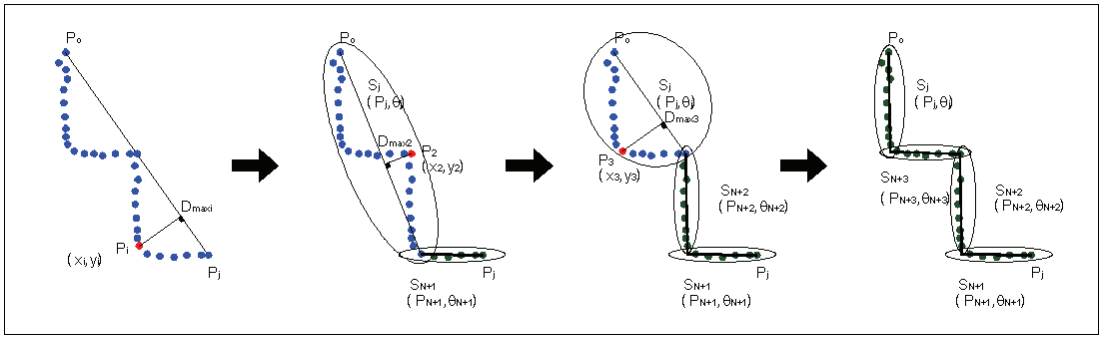


Figura 2.2 Mapas basados en características, [12]

2.1.1.3 Mapas Topológicos

Esta técnica permite mapear un entorno utilizando la navegación, formando un grafo que contenga la información de las métricas del mismo y los nodos (Figura 2.3).

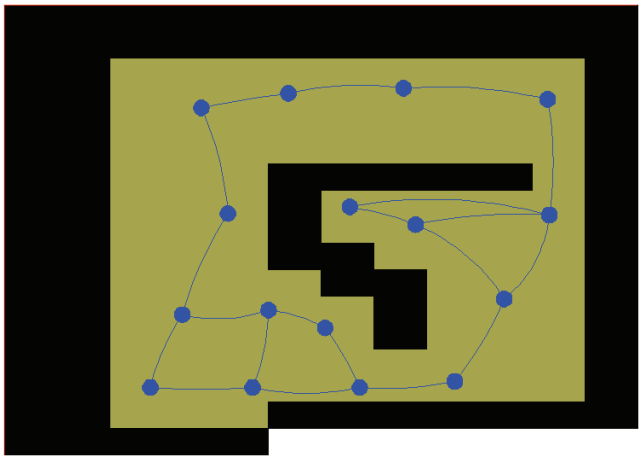


Figura 2.3 Mapas topológicos, [12]

2.1.1.4 Vector de percepción General

Esta técnica se basa en el almacenamiento de la información recolectada por sensores, que posteriormente es procesada y guardada en un vector de percepción que contiene los datos necesarios para determinar la probabilidad de

presencia de obstáculos en el entorno cercano (Figura 2.4). Este proceso es iterativo ya que conforme el robot ejecuta sus movimientos, se actualiza el vector de percepción.

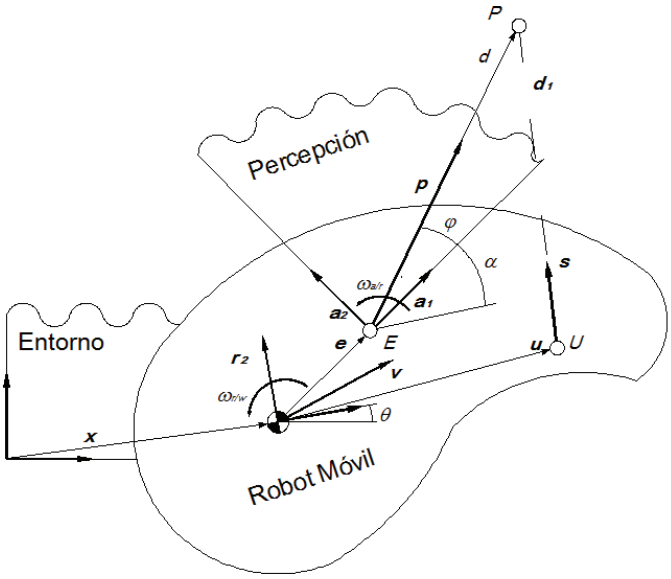


Figura 2.4 Mapa vector de percepción General, [12]

2.2 PLANIFICACIÓN DE RUTAS

El paso siguiente para la navegación de los robots móviles es determinar cuáles son las rutas que el robot debe seguir para llegar a su destino.

La ruta es una secuencia de puntos o configuraciones determinadas por un algoritmo para circular por el mapa de entorno, que son procesados posteriormente para que un robot realice el seguimiento de estos utilizando expresiones matemáticas seguir para alcanzar la meta [13].

Para la planificación de rutas parte desde un mapa de entorno modelado, generando una secuencia nodos coordenados, en el espacio libre se colisión, que definan la ruta [13].

Uno de los requisitos para la planificación es el tipo de mapeo realizado para la modelación del mapa de entorno, el cual determina los métodos a seguir para

garantizar que el robot circule libremente sin colisionarse, además de tener la información de las dimensiones del robot utilizado.

2.3 MÉTODOS PLANIFICADORES DE RUTAS

Los métodos planificadores de rutas, son algoritmos que tienen como objetivo, encontrar una ruta libre de obstáculos dentro de un mapa de entorno preestablecido desde un punto ó configuración inicial P_0 a un destino, también llamado configuración final P_f.

Actualmente existen una gran variedad de métodos planificadores de rutas, cada uno con una manera particular de resolver el problema de planificación individualmente. Adicionalmente al fusionar dos ó más métodos, los nuevos algoritmos brindan ventajas significativas en la planificación.

Los primeros métodos de planificación se caracterizan en la construcción de grafos que rodeen todo el mapa de entorno, y con ayuda de un algoritmo buscador de caminos se obtiene la secuencia de nodos que formen la ruta entre las configuraciones deseadas.

Otro enfoque es la construcción directa del camino en tiempo real utilizando los sensores del robot, para evitar los obstáculos. A continuación se describen los métodos más difundidos y para cada uno se describen sus variantes y sus combinaciones con otros.

2.3.1 MÉTODO GRAFO DE VISIBILIDAD [14]

Los grafos de visibilidad (Nilson 1969), presenta un enfoque geométrico para realizar la planificación de rutas. Este método consiste en obtener una ruta libre de obstáculos a partir de la generación de un grafo de conectividad construido por este algoritmo dentro de un mapa de entorno conocido, donde se incluyen las

configuraciones inicial P_0 y final P_f. Este es uno de los métodos más aceptados debido a su simple aplicación.

El requerimiento para aplicar este algoritmo es la representación del mapa de entorno en forma poligonal, es decir que el espacio libre de colisiones y los obstáculos estén constituidos por polígonos.

El reto principal que aborda este método planificador es la construcción del grafo de conectividad, para lo cual se toma los vértices de los obstáculos y también las configuraciones inicial P_0 y final P_f y se incluyen en el grafo formando sus nodos. La formación del grafo se origina enlazando todos los nodos que tienen línea de vista entre sí (Figura 2.5 b).

A nivel computacional el método planificador utiliza matrices de almacenamiento para ingresar las coordenadas de los nodos, la secuencia de los enlaces formados entre ellos y el costo de los enlaces.

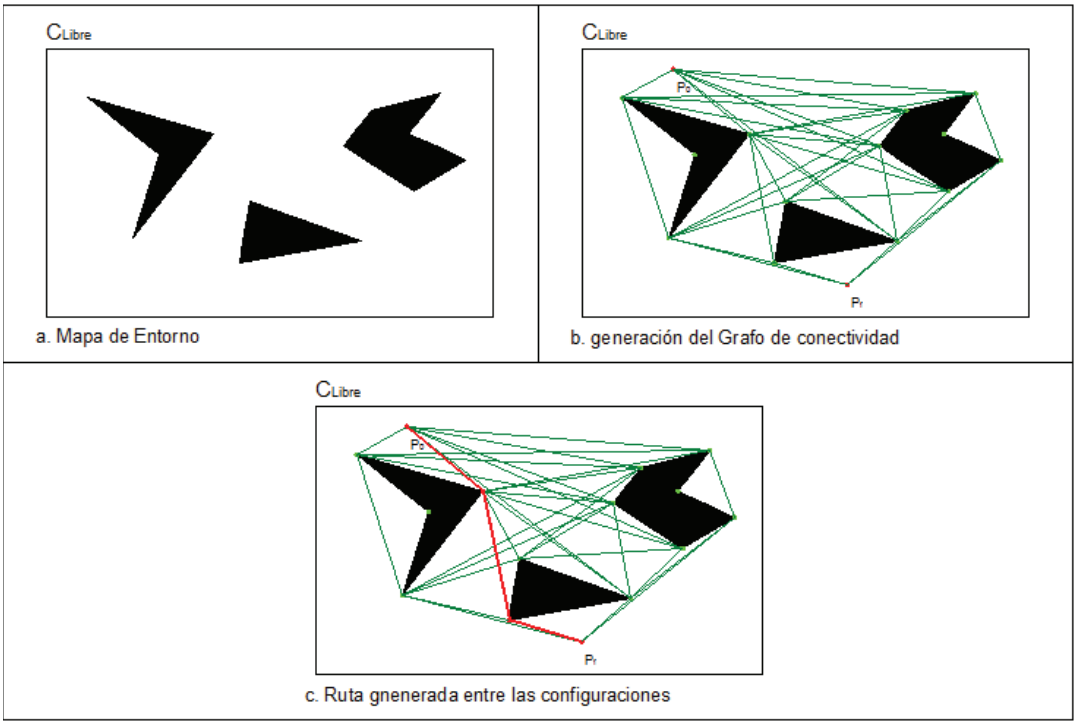


Figura 2.5 Proceso de planificación de rutas por Grafo de Visibilidad

El proceso de construcción del grafo de conectividad empieza seleccionando un nodo del mapa de entorno (para el Grafo de Visibilidad es tomar un vértice de cualquier obstáculo). Este se enlaza primero con el nodo más cercano analizando previamente si existe un obstáculo entre ellos. En el caso que existiera línea de vista, se considera un enlace exitoso y se guarda la información del enlace entre los nodos en una matriz que representa al grafo de conectividad, en el caso que exista un obstáculo no se toma en cuenta el enlace, el algoritmo continúa con el mismo proceso con el resto de los nodos del grafo.

Una vez terminado el proceso con el primer nodo se toma el siguiente y se aplica el mismo criterio pero descartando el nodo ya tomado en cuenta para el grafo de conectividad, esto se realiza con el fin de evitar redundancias en los enlaces. Al final al grafo se incluyen los enlaces que constituyen la periferia de los obstáculos (Figura 2.5 b).

Concluido la construcción del grafo de conectividad, este puede contener más de una ruta entre las configuraciones inicial P_0 y final P_f. Para determinar la ruta definitiva se toma en cuenta el costo de todos los enlaces del grafo de conectividad (habitualmente la distancia Euclídea), y con un algoritmo de búsqueda de rutas como el algoritmo Dijkstra ó Algoritmo A* (Anexo B), se determina una ruta entra las configuraciones, ó también se puede determinar la ruta con otros criterios basados en la minimización de número de nodos, esta selección básicamente depende de la maniobrabilidad del robot.

La principal limitación del método planificador es la dependencia directa del tiempo de procesamiento en función de la complejidad del mapa de entorno, pero con la evolución de los procesadores y la inclusión de algoritmos en la actualidad el método planificador puede resolver entornos complejos en poco tiempo, entre las técnicas para disminuir el tiempo de procesamiento se describen las siguientes:

2.3.1.1 Eliminación de Vértices Cóncavos [14]

Los obstáculos que tienen vértices cóncavos se caracterizan por maximizar la distancia entre enlaces formados, aumentando el tiempo de procesamiento. Y en todos los casos no sería solución al momento de aplicar el algoritmo de búsqueda de rutas.

La eliminación de estos vértices implica optimizar la construcción del grafo de conectividad (Figura 2.6).

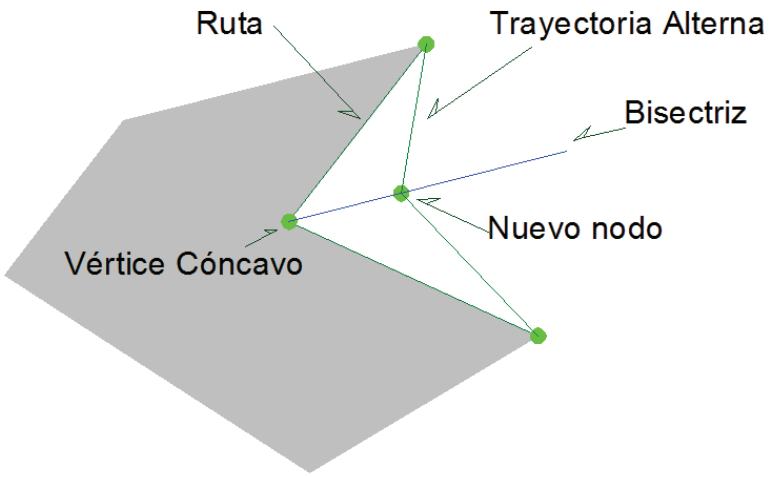


Figura 2.6 Vértice cóncavo, [14]

La solución es la inclusión de un vértice alternativo al obstáculo generando una bisectriz. Esto demuestra que no es la solución óptima y por lo tanto los arcos del grafo perteneciente a dicho vértice jamás están en la solución.

2.3.1.2 Segmentación Tangencial [14]

La segmentación tangencial disminuye el procesamiento creando nuevos enlaces entre dos obstáculos que sean vecinos, evitando enlaces redundantes cuando los obstáculos tienen más de un vértice para unir dos nodos.

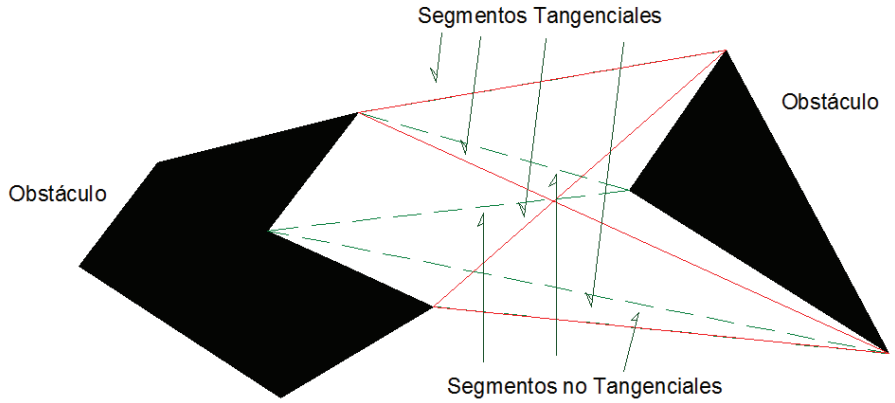


Figura 2.7 Segmento tangente, [14]

De acuerdo con [14] y la Figura 2.7, se considera segmento tangencial cuando un enlace entre dos nodos de obstáculos diferentes no interseca los obstáculos.

2.3.1.3 Planificación basada en Subgrafos de Visibilidad [13]

Los subgrafos de visibilidad son una simplificación del método planificador original. Este subgrafo busca la primera ruta entre las configuraciones P_0 y, P_f libre de colisión como solución definitiva pero no necesariamente es la ruta óptima.

A diferencia del método de grafo de visibilidad, este integra la construcción del grafo de conectividad con la búsqueda de ruta en la misma ejecución.

Para generar la ruta libre de obstáculos, el robot toma directamente la configuración inicial P_0 y se enlaza con el primer nodo visible (preferentemente el más cercano) incorporándose a la solución (Figura 2.8 a), luego el proceso se repite con el siguiente nodo hasta realizar el enlace con la configuración final P_f y completar la ruta definitiva (Figura 2.8 d).

En caso que un nodo no pueda enlazarse, se desecha el enlace de ese nodo con el anterior y busca otra ruta enlazándose con otro nodo (Figura 2.8 c).

Esta variante del grafo de visibilidad tiene la ventaja de disminuir el tiempo de procesamiento para encontrar la ruta pero no necesariamente la óptima (por lo general la más corta).

Al llevar el método de grafos de visibilidad a la práctica el robot colisionaría con los vértices de los obstáculos entonces se considera al robot como un cuerpo sólido y no adimensional.

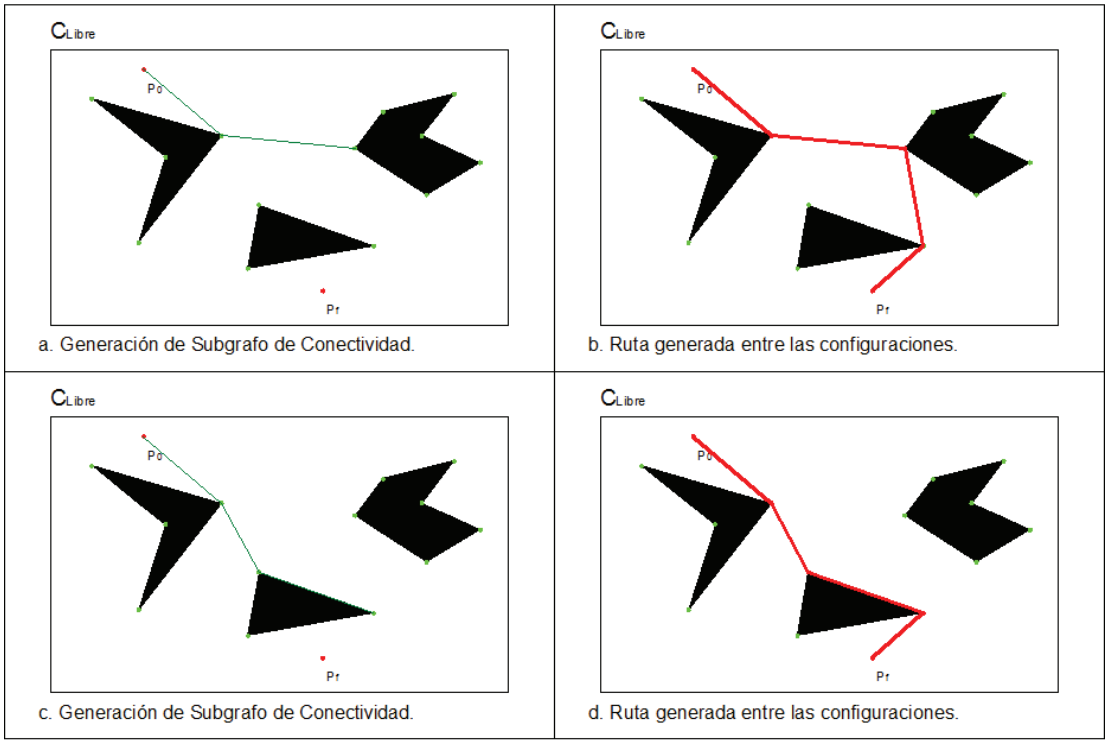


Figura 2.8 Proceso de creación de rutas

2.3.1.4 Entorno Expandido [13]

El entorno expandido es la amplificación de los obstáculos del mapa de entorno que compense la restricción de las dimensiones del robot para trasladarse.

La amplificación de las dimensiones de los obstáculos se calcula en función de la geometría del robot a utilizar. Si al robot se considera como un círculo de radio r, entonces la amplificación de los obstáculos y la contracción del contorno del mapa cambiarían al menos en el radio del robot.

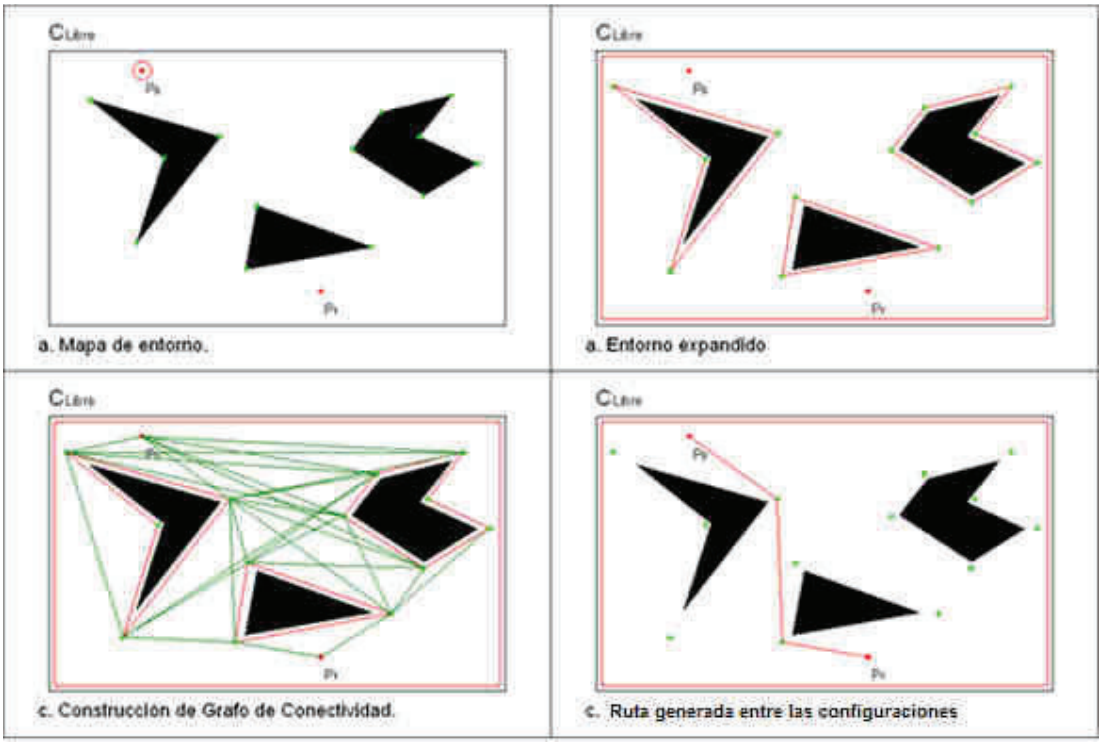


Figura 2.9 Método del entorno expandido

Al implementar el criterio del contorno expandido, se constituye un paso previo al cálculo del grafo de visibilidad.

Existen casos que al ejecutar la ampliación de los obstáculos puede generar solapamiento de entre obstáculos ó contra el contorno del mapa. Este es un indicativo que el robot no puede trasladarse por esos caminos limitando rutas que pueden ser más cortas entre las configuraciones, pero en virtud de la disminución del tiempo de cómputo ya que el número de nodos disminuye y por ende el grafo de conectividad.

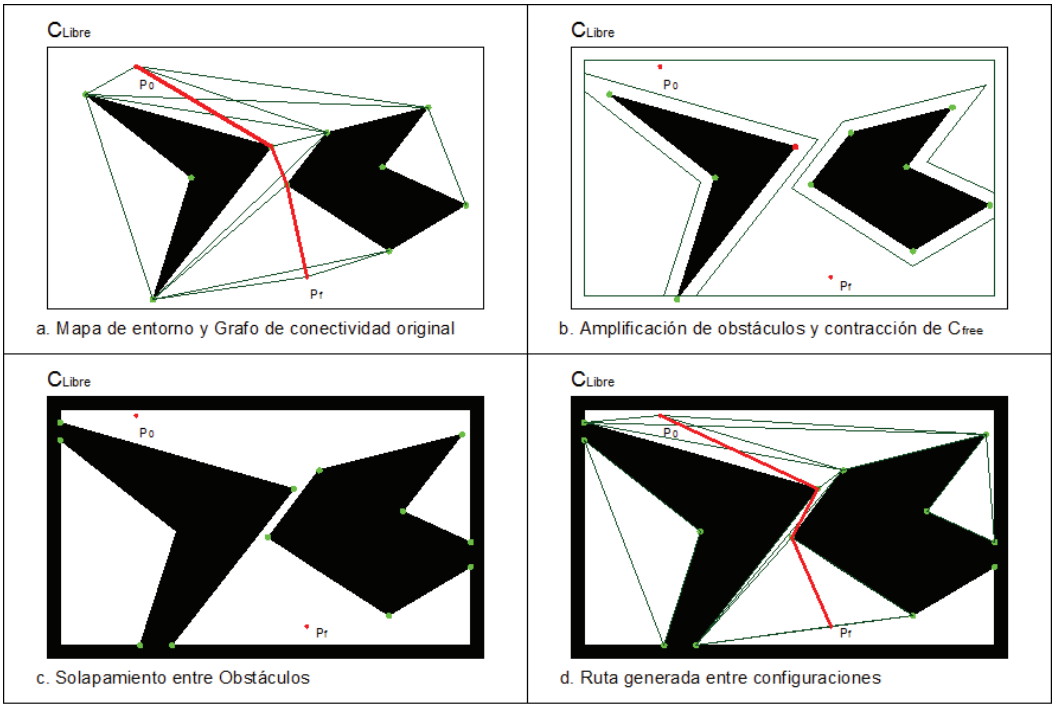


Figura 2.10 Solapamiento entre obstáculos

La Figura 2.10 presenta diferentes escenarios de la obtención del grafo de conectividad cuando se incluye la amplificación de los obstáculos.

De acuerdo con la Figura 2.11, la ruta obtenida con el criterio del entorno expandido, elimina las posibles colisiones del robot al trasladarse por espacios estrechos.

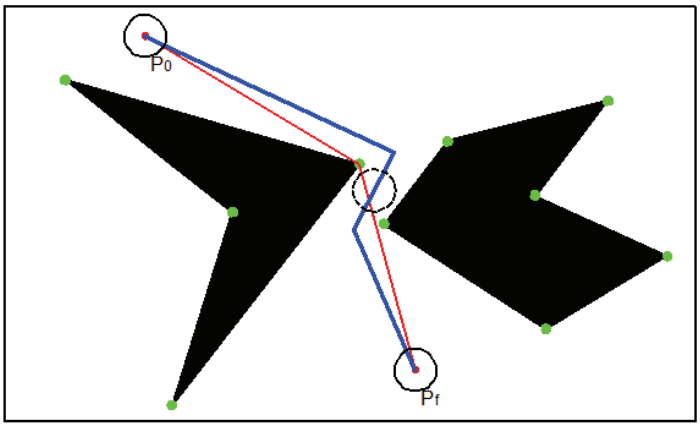


Figura 2.11 Comparación entre caminos

2.3.2 PLANIFICACIÓN POR DESCOMPOSICIÓN DE CELDAS [14]

Este método planificador al igual que el de grafo de visibilidad es un algoritmo basado en la construcción de un grafo de conectividad, pero este grafo no se forma a partir de los vértices de los obstáculos, sino crea sus propios nodos descomponiendo el espacio libre de obstáculos del mapa de entorno C_{free} en secciones.

Por concepto. Este método se divide en dos etapas:

- Segmentación del espacio libre

- Generación del Grafo de conectividad

La primera etapa implica descomponer el espacio libre en celdas, las cuales son representadas por nodos que formen el grafo de conectividad (Figura 2.12 b).

Una vez determinadas las celdas y los nodos, el siguiente paso es la construcción de un grafo de conectividad (Figura 2.12 c). Este grafo se forma a partir de la unión de nodos de celdas adyacentes a diferencia del método del grafo de visibilidad que maneja línea de vista entre todos los nodos.

Terminado el grafo de conectividad, se incluyen las configuraciones inicial P_0 y final P_f, y con un algoritmo de búsqueda de rutas (algoritmo Dijkstra) se calcula el camino entre las configuraciones (Figura 1.12 d).

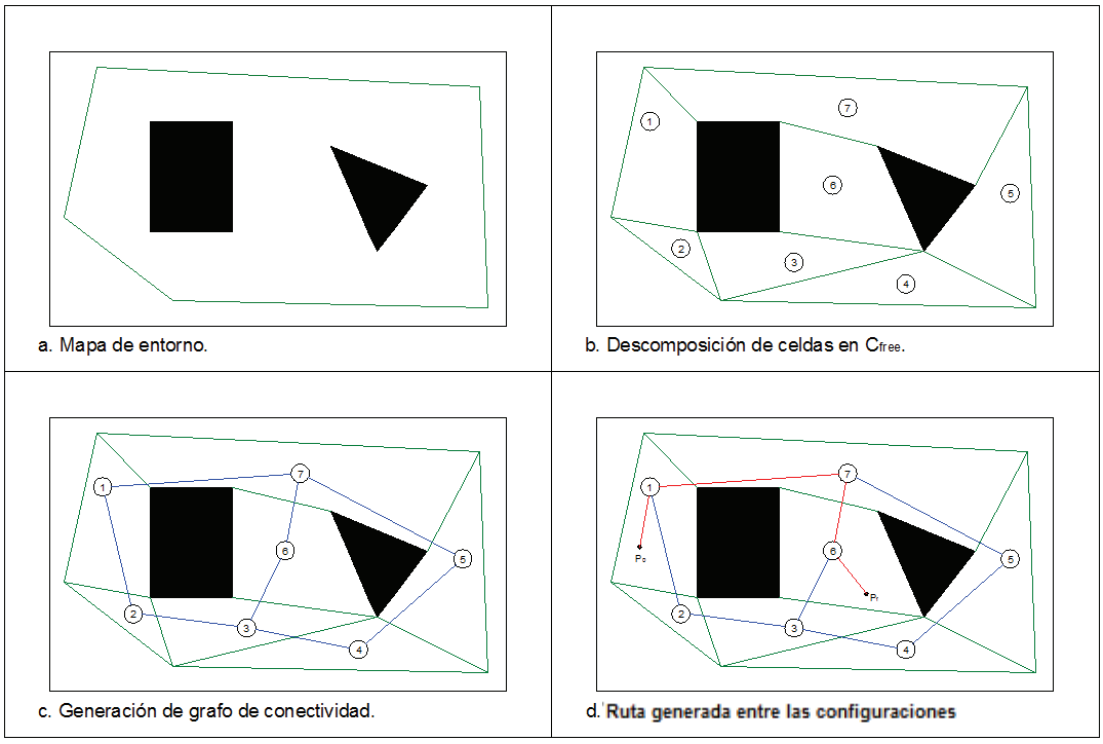


Figura 2.12 Ejecución del método planificador

Aunque se puede descomponer el espacio libre en celdas libremente, existen algoritmos que descomponen de forma eficiente, ordenada y proporcionada de los cuales los siguientes son los más utilizados.

2.3.2.1 Descomposición de Celdas Exactas

Este tipo de descomposición fragmenta el espacio libre de obstáculos C_{free}, en su totalidad Figura 2.13.

Las celdas obtenidas son representadas por nodos. Dos de estos son enlazados sí y solo sí las correspondientes celdas son adyacentes formando el grafo de conectividad.

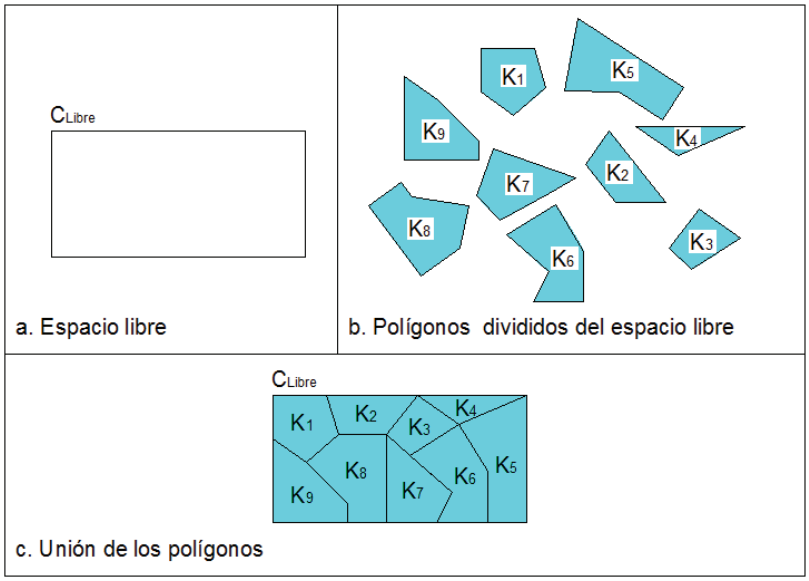


Figura 2.13 Descomposición de celdas exactas

Existen varias técnicas de descomposición del espacio libre de las cuales se mencionan las siguientes:

2.3.2.1.1 Descomposición de Celdas Vertical (CHAZELLE, 1987)

La descomposición vertical, fragmenta el especio libre de colisión C_{free}, en celdas triangulares y/o trapezoidales trazando rectas verticales desde los vértices de cada obstáculo a lo largo del mapa de entorno como se presenta en Figura 2.14.

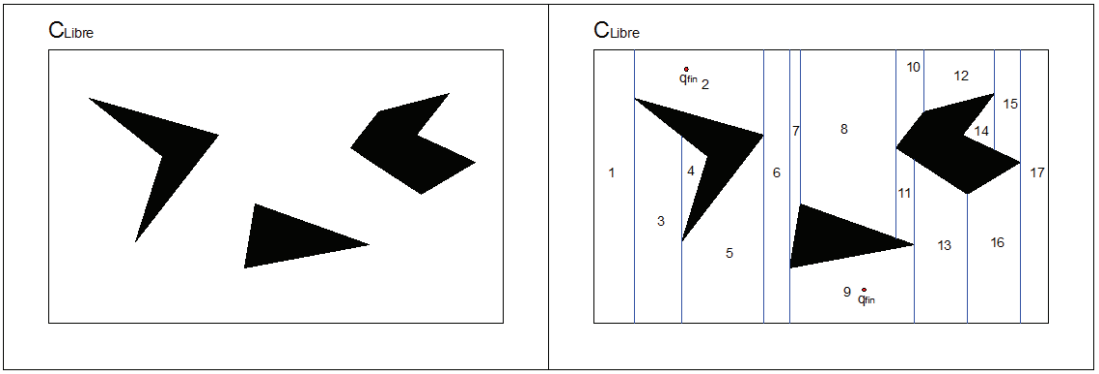


Figura 2.14 Formación de rectas verticales

Los nodos que representan a las celdas son colocados en los puntos medios de cada recta generada (Figura 2.15 a), para luego dibujar el grafo de conectividad que contiene la ruta entre las configuraciones inicial y final; este grafo se forma enlazando nodos adyacentes que no tengan un obstáculo en medio de ellos. (Figura 2.15 b)

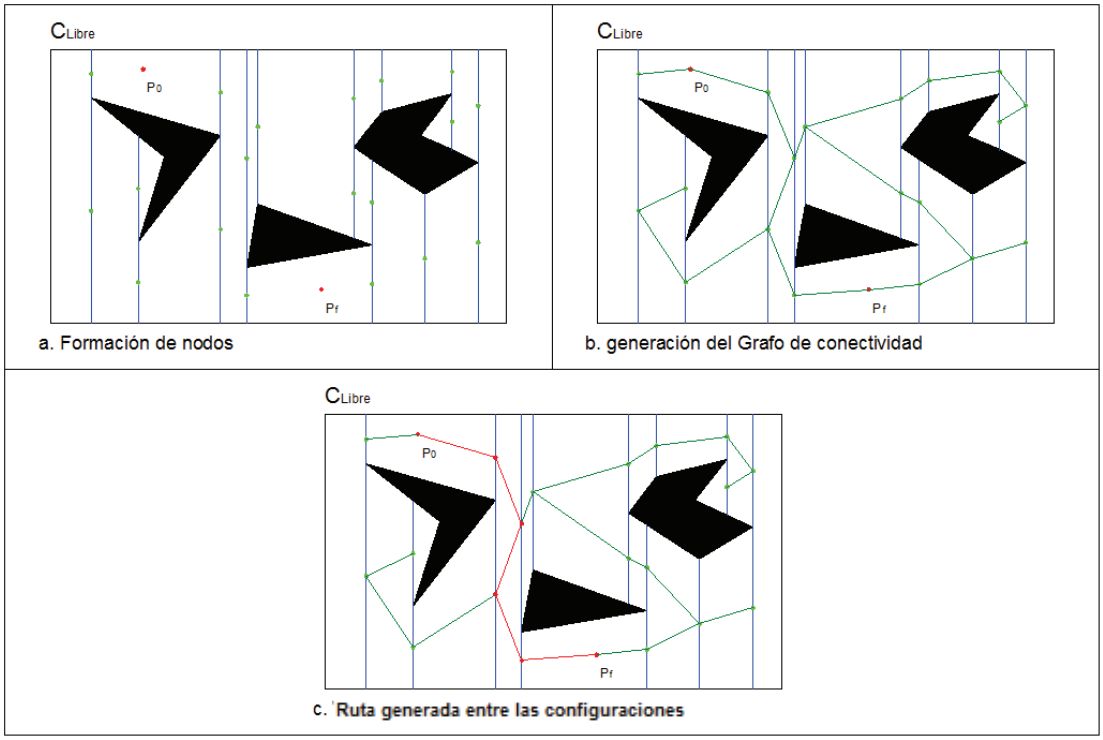


Figura 2.15 Generación del grafo de conectividad y obtención de ruta

Utilizando un algoritmo de búsqueda de rutas se determina el camino entre las configuraciones inicial P_0 y final P_f (Figura 2.15 c).

A nivel computacional un requisito para aplicar esta técnica es el ordenamiento de los vértices, del mapa, lo que implica tiempo de procesamiento [14], lo más recomendable es hacer un barrido a lo largo del eje horizontal de los vértices de cada polígono y trazar la rectas verticales.

2.3.2.1.2 Descomposición por Triangulación de Delaunay [14]

Este método de descomposición exacta, forma celdas triangulares en el espacio libre de colisión C_{free} utilizando la técnica de Triangulación de Delaunay a los vértices de los obstáculos del mapa de entorno.

La triangulación de Delaunay consiste en hallar la triangulación en una nube de puntos que sean próximos entre sí formando aristas. Esta técnica se aplica en sistemas de dos y tres dimensiones. Para un sistema de dos dimensiones el método cumple las siguientes propiedades.

"Sea P una nube de puntos en un plano, existe una triangulación de Delaunay en P sí y solo sí existe una circunferencia circunscrita de cualquier triangulación que no contenga puntos en la circunferencia considerada" [15].

Una triangulación ilegal se obtiene cuando tres puntos que forman una circunferencia circunscrita contiene en su interior uno o varios puntos que pertenecen a P.

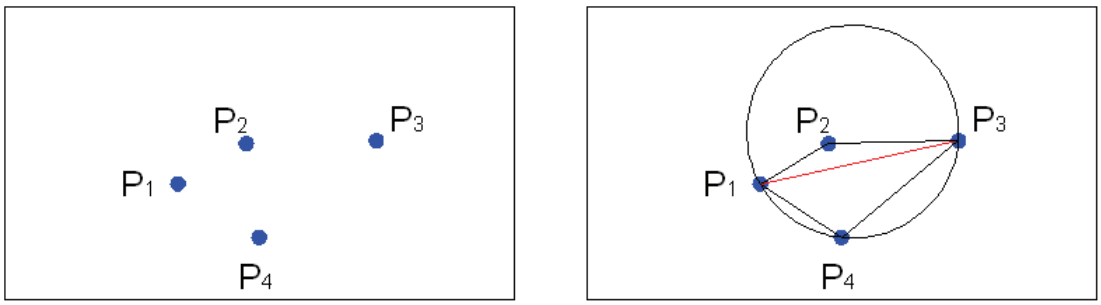


Figura 2.16 Triangulación ilegal, [15]

De la Figura 2.16 el punto P_2 se encuentra dentro del círculo circunscrito por el triángulo formado por los vértices P_1, P_3, y P_4, por lo que no cumple con la triangulación de Delaunay. Para arreglar este problema se reestructura la triangulación cambiando los puntos del plano. En el caso de la Figura 2.16 los

puntos P_1 y P_3 no deben tener una arista de triangulación debido a que puede contener los puntos P_2 y P_4 .

Seleccionando los puntos P_2 yP_4 para formar una arista de la triangulación, se toma el punto P_1 y se traza el circulo circunscrito, (Figura 2.17), el punto P_4 no pertenece al círculo completando la triangulación. De la misma forma se realiza la otra triangulación con los puntos P_2, P_4, y P_3.

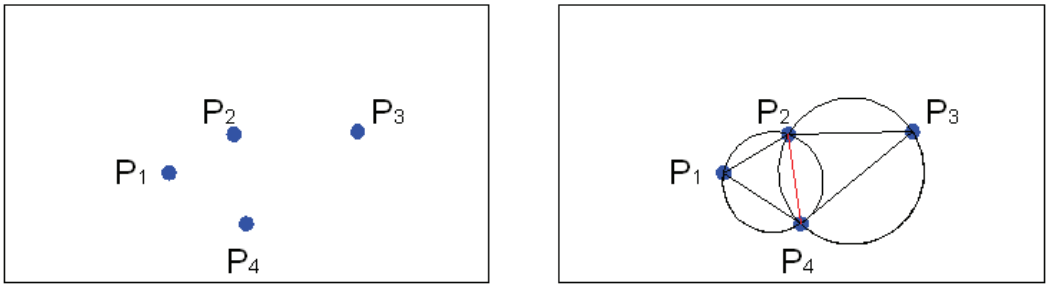


Figura 2.17 Triangulación legal, [15]

Aplicando la técnica en la descomposición de celdas, los puntos del plano son representados por los vértices de los obstáculos y la triangulación se aplica sobre la espacio libre C_{free} formando las celdas.

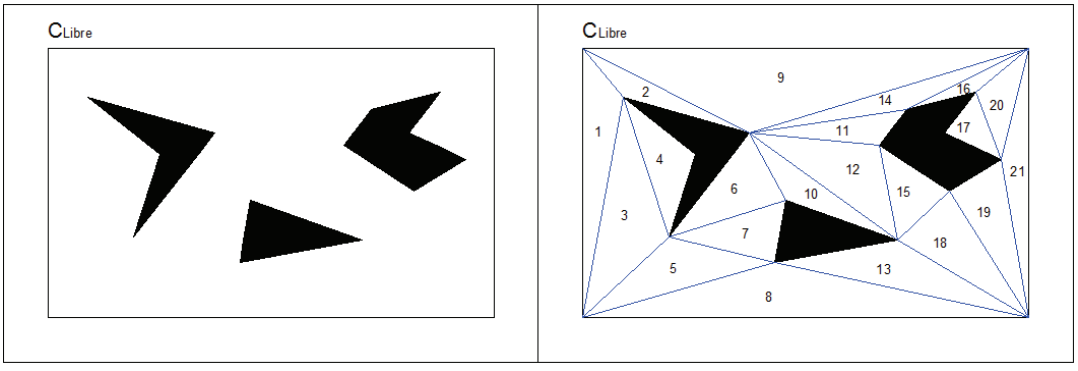


Figura 2.18 Formación de triangulaciones

Con las celdas formadas se colocan los nodos, se pueden tomar los puntos medios de las triangulaciones y formar los nodos (Figura 2.19 a). Luego se traza el grafo de conectividad uniendo los nodos de celdas adyacentes (Figura 2.19 b).

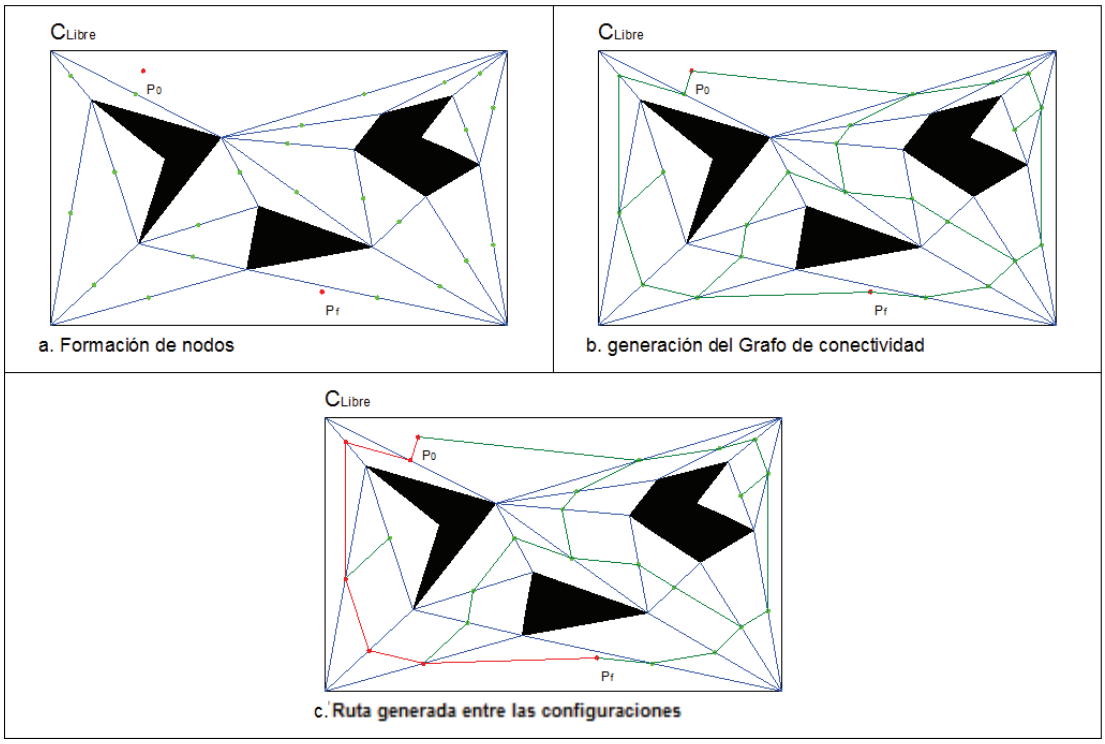


Figura 2.19 Generación del grafo de conectividad y obtención de la ruta

Esta técnica tiene la ventaja de obtener el grafo de conectividad que está más distante de los obstáculos del mapa de entorno por lo tanto se tiene un margen de seguridad mayor que la técnica vertical.

2.3.2.2 Descomposición de Celdas Aproximada o Adaptiva [14]

La descomposición adaptiva fragmenta el mapa de entorno, en celdas utilizando bloques preestablecidos, generalmente bloques semejantes geométricamente con el mapa de entorno, ó bloques rectangulares (Figura 2.20 a). Las celdas que ocupan los obstáculos con desechadas, mientras las que se encuentran en el espacio libre C_{free} forman parte del grafo de conectividad.

Debido a que la descomposición del mapa de entorno se utiliza bloques predefinidos, que al descomponer el mapa existen bloques que contienen

obstáculos y espacio libre, por lo cual se utiliza un técnica de descomposición sucesiva o “quadtree” (Figura 2.20 b) para realizar múltiples divisiones de las del mapa de entorno en celdas mejorado la resolución y obtener una mejor respuesta [14].

Las celdas que descomponen el mapa de entorno son clasificadas de acuerdo a la ocupación de espacio libre, obstáculos, y mixtas. Están son representadas por un código de color para identificarlas.

El proceso de descomposición comienza dividiendo el mapa de entorno en cuatro rectángulos semejantes. Estos rectángulos son analizados si contienen el espacio libre, un obstáculo o si son mixtos. En el caso que una celda contenga el espacio libre se concluye que el rectángulo forma parte del grafo de conectividad, sí el rectángulo inscribe a un obstáculo, es desechado, y en el caso que una celda sea mixta, se vuelve a dividir y continua con el proceso de análisis anterior.

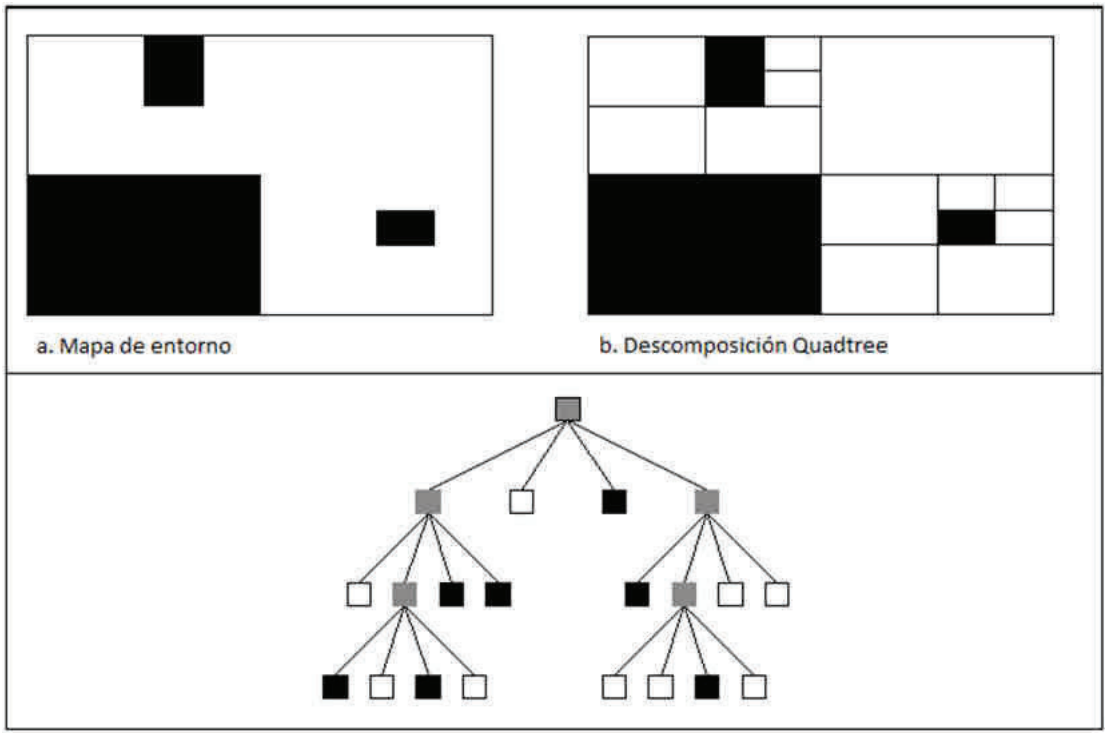


Figura 2.20 Proceso de descomposición sucesiva, [14]

Al ser una técnica jerarquizada puede representarse utilizando un árbol para cada división de un rectángulo gris. La descomposición quadtree realiza la división del mapa (Figura 2.20 c), y la representación se da en el árbol de jerarquía. Al tener el mapa obstáculos rectangulares el algoritmo termina al cuarto nivel de división.

Cuando se aplica a polígonos irregulares el número de divisiones rectangulares aumenta (Figura 2.21) y por ende las iteraciones, pero se puede incluir un mecanismo para limitar el número de iteraciones para terminar el proceso de descomposición.

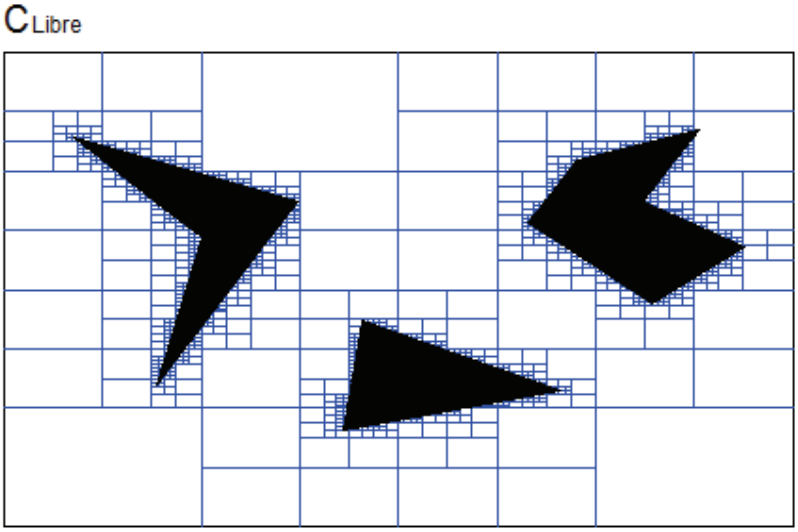


Figura 2.21 Formación de quadtree

Al completar el algoritmo se determinan los nodos que representa a cada celda, lo más común es colocar los nodos en los puntos medios de un lado de cada celda y se forma el grafo de conectividad de la misma forma que para el método de descomposición exacta como se presenta en la Figura 2.22.

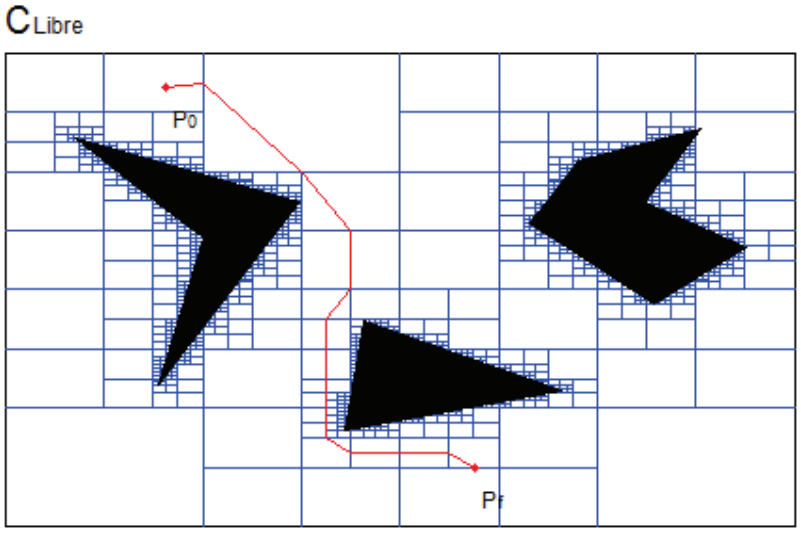


Figura 2.22 Generación de la ruta entre configuraciones

Del grafo de conectividad se calcula la ruta entre las configuraciones inicial y final utilizando un método de búsqueda de rutas como el algoritmo Dijkstra o A* (Anexo B).

Este método es muy utilizado a nivel computacional para identificación de imágenes, representando las celdas con pixeles como el valor mínimo de rectángulo.

2.3.3 PLANIFICACIÓN POR VORONOI

El algoritmo Voronoi se utiliza como un método planificador completo, y también es aplicado como un complemento para otros métodos planificadores [14].

Como método planificador el algoritmo Voronoi consiste en la construcción de un grafo de conectividad en el mapa de entorno utilizando el algoritmo del mismo nombre, para luego trazar una ruta entre las configuraciones inicial y final utilizando un algoritmo de búsqueda de rutas. Al igual que los métodos anteriores descritos, este requiere que el mapa de entorno tenga representación poligonal.

A diferencia del método por Grafo de Visibilidad donde el objetivo es buscar todos los enlaces posibles entre los nodos formados por los vértices de los obstáculos, el método Voronoi se basa precisamente en lo contrario, el algoritmo busca maximizar la distancia entre los obstáculos para construir el grafo de conectividad, obteniendo una ruta los más alejada posible de los obstáculos y siendo el método eficaz para aplicaciones donde se tracen caminos en lugares estrechos o tengan demasiados obstáculos.

2.3.3.1 Diagramas Voronoi

Los diagramas Voronoi constituyen una de las herramientas principales de la geometría computacional. En la actualidad estos diagramas tienen aplicaciones en varias ramas relacionadas en la búsqueda de lugares que tengan características de interés, también tiene aplicaciones para construir caminos lo más alejados posibles de un punto.

El diagrama Voronoi es un grafo que divide el entorno en regiones proximidad formadas por los nodos correspondientes a dicha región (Figura 2.23).

Según [16] “los diagramas Voronoi son estructuras que contiene la información de proximidad de un conjuntos de puntos descomponiendo el plano en regiones”.

De los dos conceptos del diagrama se comprueba las aplicaciones que tiene los diagramas Voronoi. Para planificación de rutas lo que interesa del diagrama Voronoi es construir un grafo que contenga el camino entre las configuraciones del cual se maximiza la distancia entre los obstáculos.

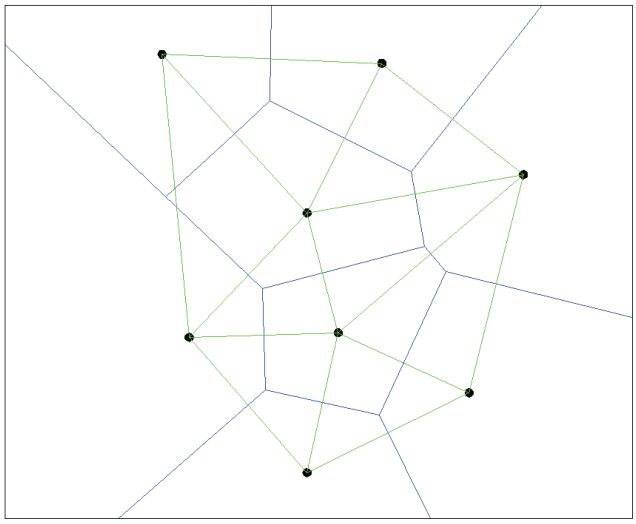


Figura 2.23 Diagrama Voronoi, [16]

Para la construcción del diagrama Voronoi, primero se traza un segmento entre dos nodos seleccionados, ($\overline{P_1P_2}$). A este segmento se traza una bisectriz que divide el plano en regiones Voronoi (Figura 2.24 a).

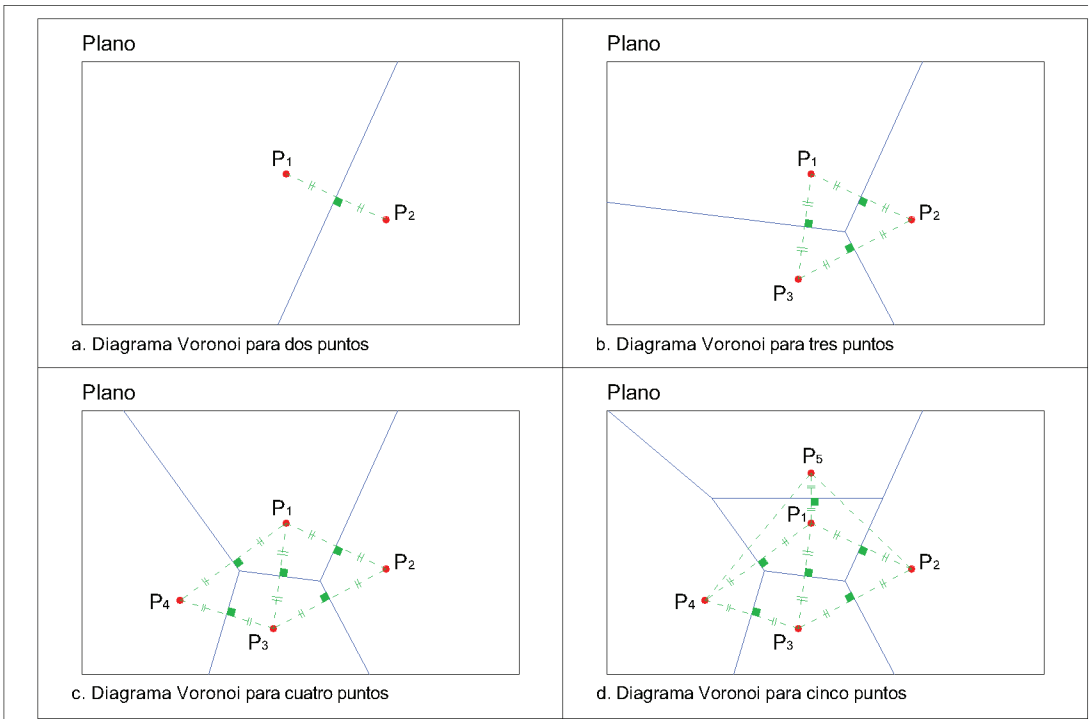


Figura 2.24 Formación de regiones con diagrama Voronoi

Cuando se agrega un nodo se traza las bisectrices del los segmentos formados por los nodos $\overline{P_3P_2}$ y $\overline{P_1P_3}$ (Figura 2.24 b). La mediatriz del triángulo formado por los nodos evaluados, es el punto límite de los segmentos del Grafo formado por el método. Al adicionar otro punto el proceso de repite con el siguiente triángulo formado hasta completarlo (Figura 2.24 d).

A nivel computacional existen procedimientos que permiten la construcción de un diagrama Voronoi en sistemas que tienen gran cantidad de puntos y/o obstáculos dimensionales, de los cuales se destacan:

2.3.3.1.1 Línea de Barrido (Fortune 1987) [14]

La línea de barrido consiste en construir el diagrama Voronoi, utilizando una recta vertical que recorre el mapa de entorno de izquierda a derecha analizando los puntos del mapa, construyendo el diagrama que se actualiza de conforme la línea pasa por los puntos.

2.3.3.1.2 Divide y Vencerás (Shamos y Hoey 1975) [14]

Esta técnica se basa en dividir los puntos ordenadamente hasta obtener una pareja de cada uno, a continuación se utiliza el criterio general del método que es obtener la bisectriz entre el segmento formado por la pareja de puntos, el proceso continua tomando pares de puntos de todo el mapa de entorno, para luego recursivamente tomar estas parejas con otras para formar otro diagrama hasta completar todas las subdivisiones.

2.3.3.1.3 Inserción Incremental (Green y Sibson 1978) [14]

La inserción incremental, utiliza la propiedad de dualidad del método Voronoi con la triangulación de Delaunay.

El método consiste en la obtención de la triangulación de tres nodos vecinos del mapa, y así incrementar hasta completar la triangulación. Luego se toman los

lados de las triangulaciones para trazar las mediatrices de los triángulos formados para obtener el diagrama Voronoi.

El método Voronoi originalmente utiliza puntos para la división en regiones del espacio de configuraciones, para ello se realizaron adaptaciones para utilizarlo cuando los obstáculos son sólidos con dimensiones y orientación definida. A este método se llama Voronoi Generalizado [14].

En el diagrama Voronoi Generalizado, dos nodos se equidistan utilizando rectas para determinar la región Voronoi, en el caso que se determine la región entre un punto y una recta se utiliza parábolas, y entre rectas se utiliza una recta para determinar el seccionamiento de mapa de entorno en regiones, Aunque los diagramas de Voronoi pueden ser determinados cuando los obstáculos son curvos, el grafo formado presenta alta complejidad debido a las expresiones utilizadas para resolverlo, por lo que solo se utilizan expresiones lineales y parabólicas para construir el grafo.

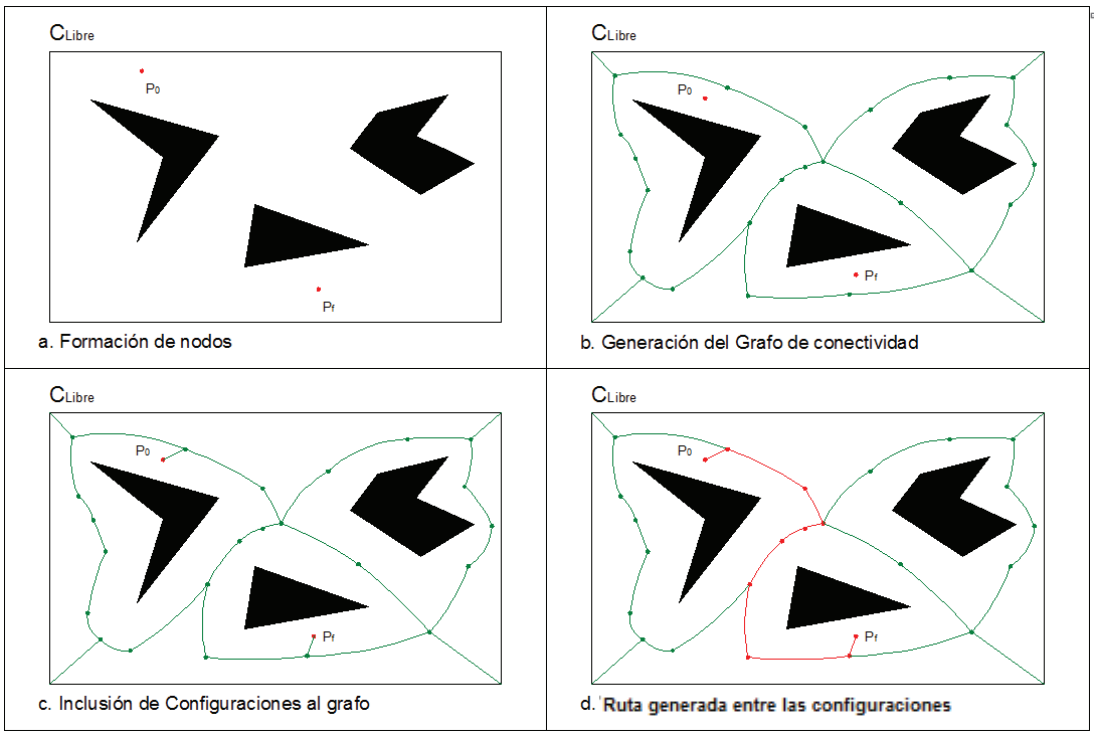


Figura 2.25 Generación del Grafo de conectividad y obtención de la ruta

La Figura 2.25 presenta el proceso de obtención de la ruta en un mapa de entorno con el algoritmo Voronoi. Primero se traza el grafo aplicado a los polígonos utilizando rectas y parábolas (Figura 2.25 b). Seguidamente se enlazan las configuraciones con el grafo trazando rectas hacia los nodos más cercanos del grafo.

Completado el grafo de conectividad, al igual que los métodos planificadores basados en grafos, se busca la ruta entre las configuraciones con el algoritmo Dijkstra.

2.3.4 PLANIFICADOR POR MAPAS PROBABILÍSTICOS [14]

También llamado "Probabilistic Roadmap". Este método genera nodos aleatoriamente en todo el espacio libre de colisiones del mapa de entorno. Tiene aplicaciones en sistemas que tengan articulaciones múltiples, y se basa en la creación de grafos de conectividad completos, o por regiones que contengan las configuraciones inicial P_0 y final P_f.

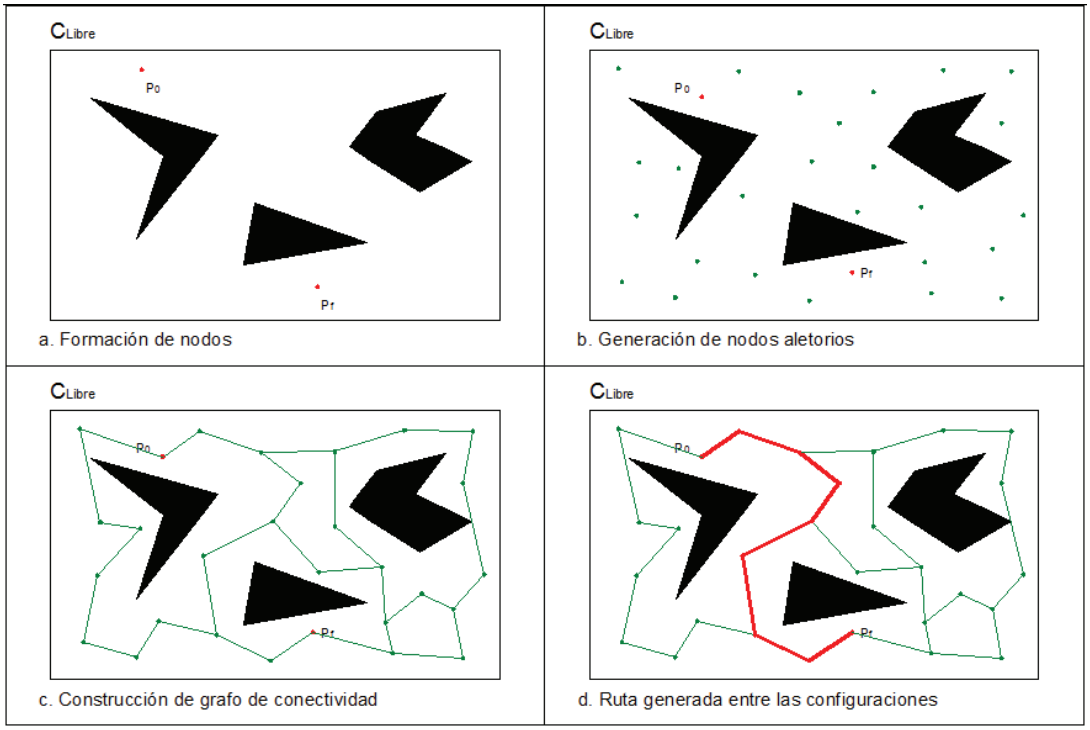


Figura 2.26 Proceso de obtención de ruta del PRM para un grafo completo

Debido a la naturaleza del método, la distribución de los nodos debe ser uniforme, en el caso que se construya un solo grafo de conectividad que incluya a las configuraciones (Figura 2.26).

El criterio para construir el grafo puede basarse en línea de vista de un nodo con todos los restantes tal como el método de grafo de visibilidad, construir enlaces con los nodos vecinos (Método de Descomposición de celdas) Figura 2.26 d, o una combinación de estos para generar grafos locales dentro del mapa.

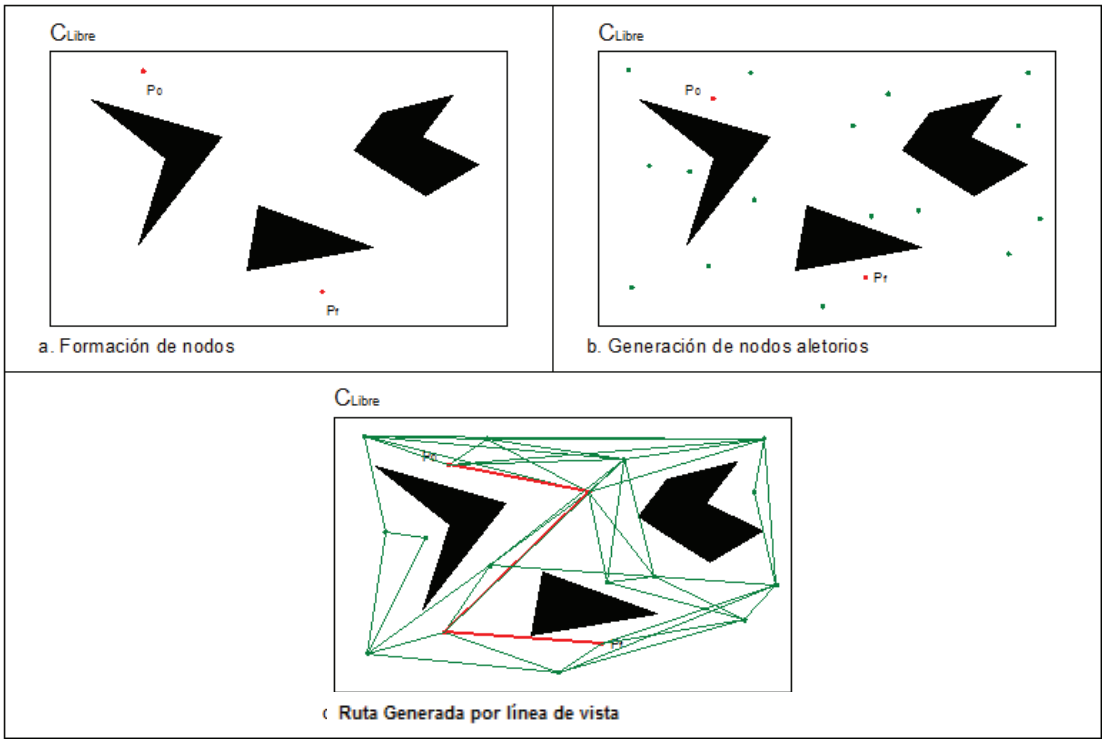


Figura 2.27 Métodos de construcción del grafo de conectividad

La principal ventaja al aplicar este método de planificación es la evitación de realizar un análisis completo del mapa de entorno para poder determinar los nodos que conformen el grafo de conectividad, y simplemente el método se limita que a cada nodo generado este ó no dentro de un obstáculo para seleccionarlo.

La Figura 2.27, presenta una distribución heterogénea de los nodos, presentándose un problema en la generación de un grafo de conectividad al no existir una solución para alcanzar la configuración final, por lo que hay que considerar la distribución de los nodos en el mapa.

Otro inconveniente es la lentitud del método cuando se construyen varios grafos de conectividad locales, debido a que se ejecuta repetidamente el método planificador para encontrar los enlaces entre las configuraciones inicial y final.

Pero en virtud de la lentitud del método aplicándose para construir varios grafos de conectividad, se obtiene mayor información del mapa de entorno, para determinar más de una sola ruta entre las configuraciones (Figura 2.28).

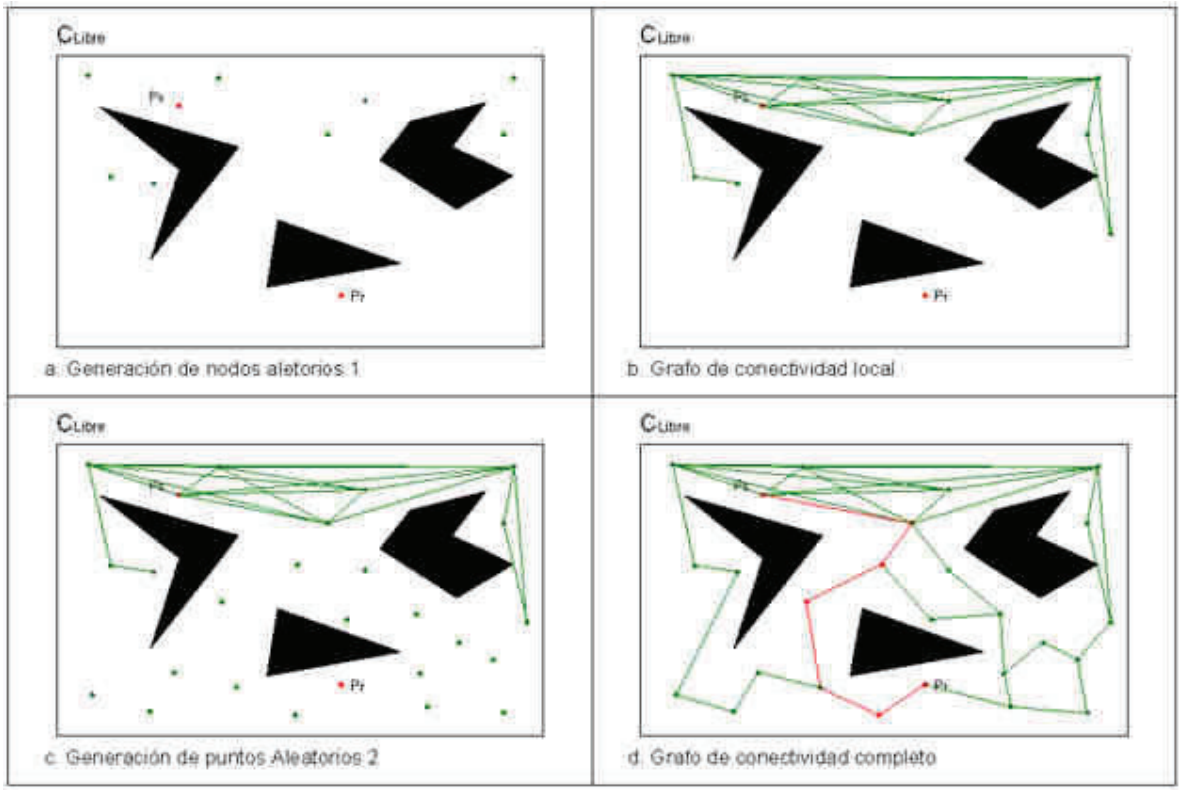


Figura 2.28 Construcción de grafos locales y obtención de la ruta

2.3.5 PLANIFICACIÓN POR CAMPOS POTENCIALES

Originalmente creado para evitar colisiones para un robot manipulador, este método desarrollado en 1985 por Khatib, es un algoritmo planificador aplicable cuando no se dispone de un mapa de entorno conocido pero el robot utiliza sus sensores para detectar los obstáculos durante la ejecución de sus movimientos [14].

El método de campos potenciales es un método de planificación de rutas, en el cual el robot es sometido a un campo de fuerzas artificiales que obliguen a desplazarse hacia la configuración final P_f evitando los obstáculos.

El campo potencial artificial es representado por este grupo de fuerzas aplicadas al robot. Por un lado existe una fuerza de atracción la cual dirige al robot hacia la configuración final utilizando la ruta más corta y por otro lado existe una serie de fuerzas repulsivas cuya función es cambiar la trayectoria del robot cuando se aproxime a un obstáculo evitando colisionarse.

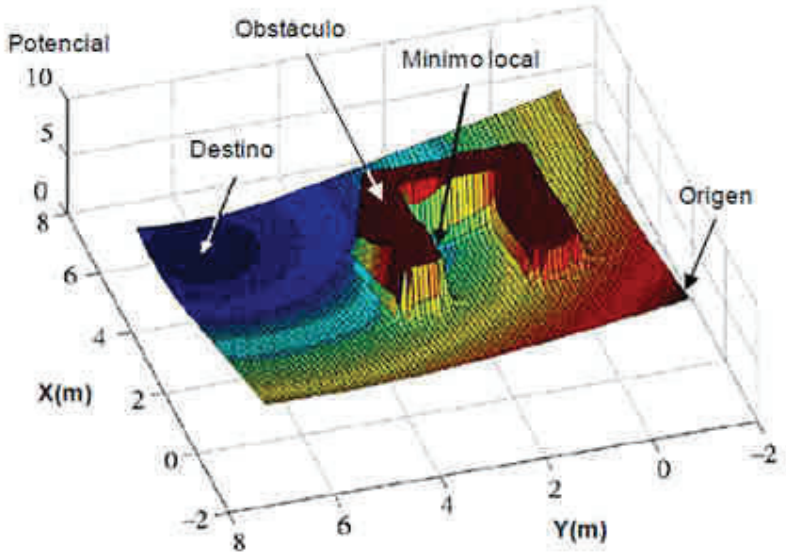


Figura 2.29 Campos potenciales generado por un entorno, [14]

Este método planificador tiene la habilidad de aplicarse en un sistema de 3 dimensiones para aplicaciones con brazos robóticos, y también se utiliza ampliamente en la robótica móvil.

Matemáticamente el campo potencial aplicado a un robot se representa por la siguiente expresión general:

$$\vec{\boldsymbol{U}}(q) = \vec{\boldsymbol{U}}_{atr}(q) + \vec{\boldsymbol{U}}_{rep}(q) \tag{2.1}$$

Donde:

- $\vec{\boldsymbol{U}}(q)$: Campo potencial resultante.

- $\vec{\boldsymbol{U}}_{atr}(q)$: Campo potencial generado por fuerza de atracción sobre el punto de destino.

- $\vec{\boldsymbol{U}}_{rep}(q)$: Campo potencial generado por fuerza de repulsión resultante producida por los obstáculos.

El método planificador genera el camino utilizando el gradiente del campo potencial. Este gradiente se calcula en la posición instantánea que se encuentre el robot P_i, para agregar este segmento al extremo de la ruta obteniendo una nueva posición para el robot.

De acuerdo con este criterio este método es iterativo por lo que se traduce en la siguiente expresión matemática para la generación del camino:

$$\vec{\boldsymbol{P}}_{(i+1)} = \vec{\boldsymbol{P}}_{(i)} + k\nabla\big(\vec{\boldsymbol{U}}(q)\big) \tag{2.2}$$

Donde:

- $\vec{\boldsymbol{P}}_{(i)}$: Posición actual del robot.

- $\vec{\boldsymbol{P}}_{(i+1)}$: Siguiente posición del robot.

- k: Constante de ajuste.

- $\nabla(U(q))$: Gradiente de la función potencial.

Las primeras versiones del método, la fuerza de atracción se define como una función parabólica donde la variable que determina la fuerza es la distancia entre la posición instantánea del robot y la configuración final.

$$U_{atr} = k\rho_g^2(p) \tag{2.3}$$

Donde:

$\rho_g(q)$: Distancia instantánea entre el robot y la configuración final y se representa como:

$$\rho_g = \left\|\vec{\boldsymbol{P}}_f - \vec{\boldsymbol{P}}_i\right\| \tag{2.4}$$

Desarrollando el gradiente de la expresión del potencial de atracción se tiene:

$$\vec{\boldsymbol{F}}_{atr} = -\vec{\nabla}U_{atr}(p) \tag{2.5}$$

$$\vec{F}_{atr} = -\left(\frac{\partial}{\partial x}k\left(\left(p_{fx} - p_{xi}\right)^2 + \left(p_{fy} - p_{yi}\right)^2\right)\vec{\boldsymbol{\imath}} + \frac{\partial}{\partial y}k\left(\left(p_{fx} - p_{xi}\right)^2 + \left(p_{fy} - p_{yi}\right)^2\right)\vec{\boldsymbol{J}}\right)$$

$$\vec{\boldsymbol{F}}_{atr} = -2k\left(\left(p_{fx} - p_{xi}\right)\vec{\boldsymbol{\imath}} + \left(p_{fy} - p_{yi}\right)\vec{\boldsymbol{J}}\right)$$

$$\vec{\boldsymbol{F}}_{atr} = -2k\left(\left(p_{fx} - p_{xi}\right)\vec{\boldsymbol{\imath}} + \left(p_{fy} - p_{yi}\right)\vec{\boldsymbol{J}}\right)$$

$$\vec{\boldsymbol{F}}_{atr} = -2k\left(\vec{\boldsymbol{P}}_f - \vec{\boldsymbol{P}}_i\right) \tag{2.6}$$

La fuerza de atracción es directamente proporcional a la distancia entre la posición instantánea del robot y la configuración final, creciendo conforme más lejos esté el robot de su destino.

Para calcular la fuerza de repulsión primero se define del campo de repulsión, el cual resulta de la sumatoria de todos los campos repulsivos generados por todos los obstáculos del mapa de entorno.

$$U_{rep}(q) = \sum_{k=1}^{r} U_{CBk}(q) \tag{2.7}$$

Donde:

- $U_{CBk}(q)$: Campo de repulsión de cada obstáculo.

El campo potencial repulsivo debe ser diseñado para lograr condiciones de estabilidad del robot y para crear en cada punto de la superficie del obstáculo una barrera que disminuya rápidamente conforme el robot se aleje del obstáculo.

Una de las primeras expresiones del campo potencial repulsivo para un obstáculo (Khatib 1985) es representada por:

$$U_{CBk}(x) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{f(x)} - \frac{1}{f(x)}\right)^2, & f(x) \leq f(x_0) \\ 0, & f(x) > f(x_0) \end{cases} \tag{2.8}$$

La región de influencia del campo potencial está limitada por las superficies $f(x) = 0$ y $f(x) = f(x_0)$.

Donde:

- x_0: Es un punto dado en la periferia del obstáculo.

- η : Es una constante de escala.

La desventaja de esta función es la complejidad de aplicarlo sobre obstáculos asimétricos donde la separación de superficies de los obstáculos puede variar ampliamente.

Utilizando el criterio de la distancia más corta el campo potencial repulsivo es (Latombe 1991):

$$U_{CBk}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{\rho_k} - \frac{1}{\rho_0}\right)^2, & \rho_k(q) \leq \rho_0 \\ 0, & \rho_k(q) > \rho_0 \end{cases} \qquad (2.9)$$

Donde:

- ρ_0 : Es la distancia de influencia del campo potencial y

- ρ_k: Distancia más corta al obstáculo identificado con el valor k.

- $\rho_k(q) = min_{q\epsilon' CB_k} \|\vec{\boldsymbol{q}} - \vec{\boldsymbol{q}}'\|$ (2.10)

La fuerza de repulsión asociada a cada obstáculo es:

$$\vec{\boldsymbol{F}}_{CB_k} = -\vec{\boldsymbol{\nabla}} U_{CB_k}(q) \qquad (2.11)$$

$$\vec{\boldsymbol{F}}_{CB_k} = -\vec{\boldsymbol{\nabla}} U_{CB_k}(q) \begin{cases} \eta\left(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho_k^2(q)}\vec{\boldsymbol{\nabla}}\rho_k(q), & \rho_k(q) \leq \rho_0 \\ 0, & \rho_k(q) > \rho_0 \end{cases}$$

Entonces la fuerza de repulsión generada por todos los obstáculos es:

$$\vec{\boldsymbol{F}}_{rep}(q) = \sum_{k=1}^{r} \vec{\boldsymbol{F}}_{CB_k}(q) \qquad (2.12)$$

Donde:

- $\vec{\boldsymbol{F}}_{CB_k}(q)$: Es la fuerza de repulsión de cada obstáculo.

Con las expresiones determinadas tanto en atracción como en repulsión la fuerza aplicada al robot $\vec{\boldsymbol{F}}_N$ es:

$$\vec{\boldsymbol{F}}_N = \vec{\boldsymbol{F}}_{rep} + \vec{\boldsymbol{F}}_{atr} \tag{2.13}$$

De la expresión anterior, la fuerza neta dirige al robot a su destino evadiendo los obstáculos, esta fuerza neta $\vec{\boldsymbol{F}}_N$ varía en función de la posición del robot. Una vez que el robot llegue a su objetivo la fuerza aplicada se vuelve nula $\vec{\boldsymbol{F}}_N = 0$. Y la expresión de desplazamiento se reduce a:

$$\vec{\boldsymbol{P}}_{(i+1)} = \vec{\boldsymbol{P}}_{(i)} \tag{2.14}$$

Manteniendo al robot en la configuración final P_f.

2.3.5.1 Mínimos Locales

Los mínimos locales son configuraciones que tienen diferentes coordenadas a la configuración final P_f del mapa de entorno donde se anula la acción del campo potencial o la acción de la fuerza virtual resultante (Figura 2.30). Esta condición impide el desplazamiento del robot en el caso de que la ejecución sea en tiempo real, o detiene la ejecución del método en el caso de simulaciones cuando el mapa de entorno sea previamente desarrollado.

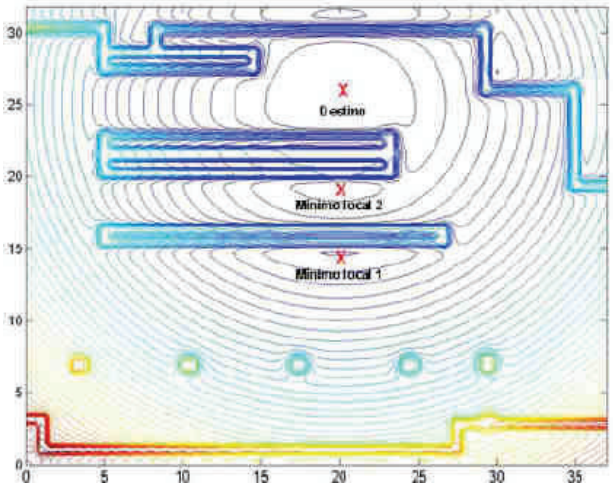


Figura 2.30 Formación de mínimos locales en mapa de entorno conocido, [11]

Los mínimos locales específicamente se forman en áreas cercanas a un obstáculo que se encuentre en medio de la posición instantánea del robot y la configuración final (ó mínimo global para el caso del método planificador) produciendo la igualdad entre la fuerza de atracción y la fuerza de repulsión neta:

$$\vec{F}_{rep} = \vec{F}_{atr} \neq 0$$

Esta condición del método planificador es una limitante o desventaja por lo que se han desarrollado varias técnicas para salir de los mínimos locales tanto en tiempo real como en el caso de tener un mapa de entorno conocido.

2.3.5.1.1 Eliminación de mínimos por movimientos aleatorios [14]

Cuando el algoritmo cae en un mínimo local, inmediatamente esta técnica trata de salir de esta posición realizando movimientos aleatorios con la precaución de detectar si estos movimientos producen posibles colisiones con los obstáculos más cercanos, para luego de una serie de movimientos volver a aplicar el método planificador original de campo potencial (Figura 1.31).

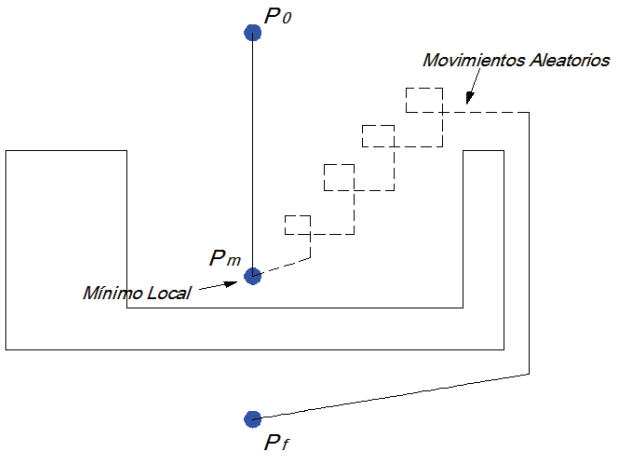


Figura 2.31 Desplazamientos aleatorios por causa de un mínimo local, [14]

2.3.5.1.2 Eliminación por inclusión de cargas ficticias [11]

Esta técnica coloca cargas repulsivas ficticias en un lugar determinado que no formen parte del mapa de entorno, que obliguen al robot a salir o alejarse de un mínimo local en la ejecución en tiempo real, ó eliminar los mínimos locales en el caso de ejecución con un mapa de entorno conocido.

La inclusión de cargas ficticias implica analizar todo el tiempo la evolución del campo potencial y el desplazamiento. Cuando el potencial neto llega a cero ($\vec{\boldsymbol{F}}_{rep} = -\vec{\boldsymbol{F}}_{atr}$) (ó un valor muy pequeño en el caso de iteraciones), se añaden estas cargas:

De acuerdo con [11], la expresión para ubicar a una carga ficticia es:

$$\boldsymbol{P}_q = k_p(\boldsymbol{P_i} - \vec{\boldsymbol{F}}_{rep} + \vec{\boldsymbol{u}}_l) \tag{2.15}$$

Donde:

- P_q: Posición de la carga ficticia.

- P_i: Posición instantánea del método planificador.

- $\vec{F}_{rep}$: Fuerza de repulsión neta provocada por los obstáculos.

- $\vec{u}_l$: Vector unitario perpendicular a la dirección de la fuerza de atracción.

- k_p: Constante que determina la distancia donde se coloca la carga ficticia.

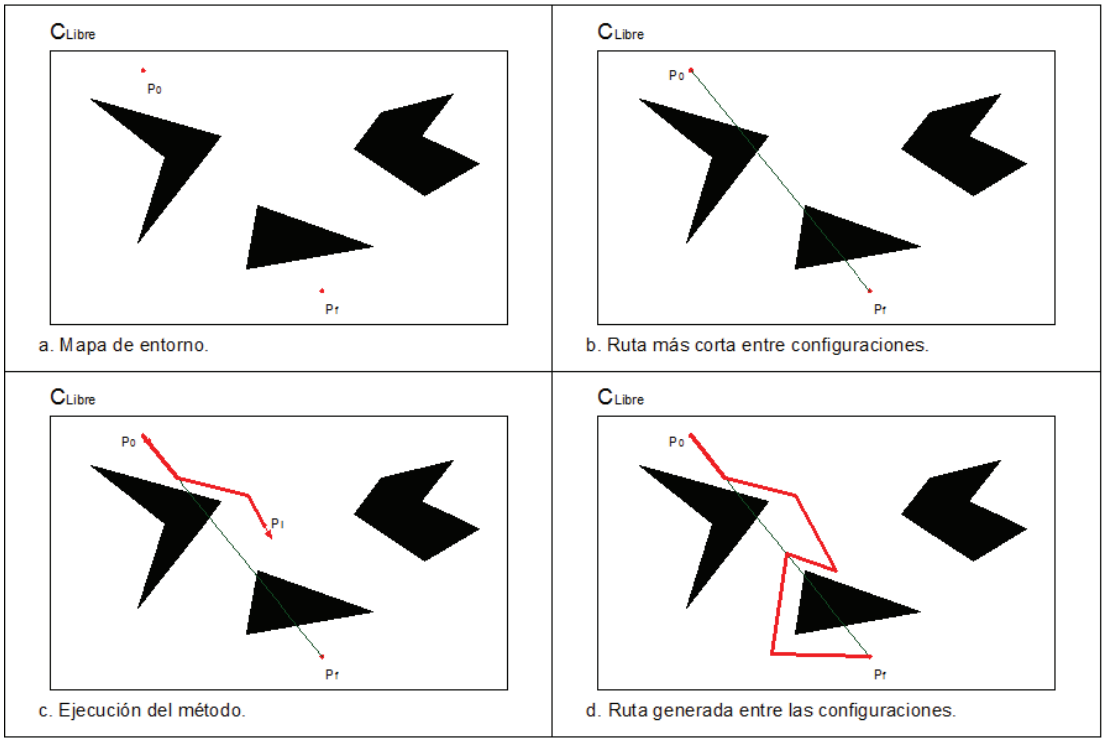


Figura 2.32 Camino formado por seguimiento de paredes

Para determinar la fuerza de repulsión de la carga ficticia se puede utilizar las mismas expresiones que para los obstáculos pero de manera general se puede aplicar:

$$U_{cf} = \frac{k}{2}\left(\frac{1}{\|P_i - P_q\|}\right)^{m_c} \tag{2.16}$$

Donde:

- k y m_c: son constantes de ajuste.

Las cargas ficticias generalmente se las añade para cambiar la dirección de la fuerza neta en la misma dirección del obstáculo obteniendo una ruta por seguimiento de paredes (Figura 2.32).

En el siguiente capítulo se desarrollara una variante del método basado en la creación de fuerzas virtuales utilizando el concepto original de generación de campo potencial por distribución discreta de carga aplicado sobre una partícula.

2.4 SEGUIMIENTO DE TRAYECTORIAS

El objetivo de la robótica móvil es controlar el robot para que ejecute desplazamientos seguros en un mapa que tenga obstáculos tanto fijos, móviles o un sistema de robots que compartan el mismo entorno de trabajo, para lo cual en esta sección se enfoca en el procedimiento para el diseño del control sobre los robots para que cumpla con su trabajo de acuerdo a las rutas generadas.

2.4.1 GENERACIÓN DE CAMINOS

De los métodos planificadores de rutas analizados anteriormente tienen en común que su respuesta es la obtención de una secuencia de configuraciones que lleven al robot hacia la configuración final a través de un mapa de entorno con obstáculos de forma segura.

Esta ruta contiene la mínima información necesaria para diseñar un controlador para alcanzar la configuración final, utilizando la posición como la variable a ser controlada.

Este tipo de controlador tiene la particularidad de realizar el control por partes dividiendo el problema del objetivo de alcanzar la configuración final en sub objetivos que es alcanzar el siguiente nodo de la ruta. Más adelante es esta tesis se realiza el control de ruta aplicando este criterio.

El control de camino realiza el seguimiento del camino sin considerar el tiempo de ejecución para completar todo su recorrido. Para realizar un seguimiento donde la variable del tiempo es un factor determinante se recurre al control de trayectoria para alcanzar las posiciones deseadas en los tiempos deseados, para lo cual el requisito para este control es la construcción de la trayectoria.

La generación de caminos es el primer paso para la formación de trayectorias, para lo cual se considera una función paramétrica $\vec{\boldsymbol{P}}(\lambda)$ [21], que se puede discretizar para obtener una sucesión de puntos coordenados de dependan de λ, y que tengan separación armónica. Tradicionalmente se utiliza la interpolación para generar una función parametrizada a partir de una ruta de los cuales los más utilizados son los siguientes:

2.4.1.1 Interpolación Lineal [24]

La interpolación lineal es la formación de una recta que representa a los puntos evaluados (Figura 2.33).

$$P(t) = mt + b \tag{2.17}$$

Esta expresión determina un camino recto a partir de los nodos involucrados donde el parámetro t determina las posiciones faltantes.

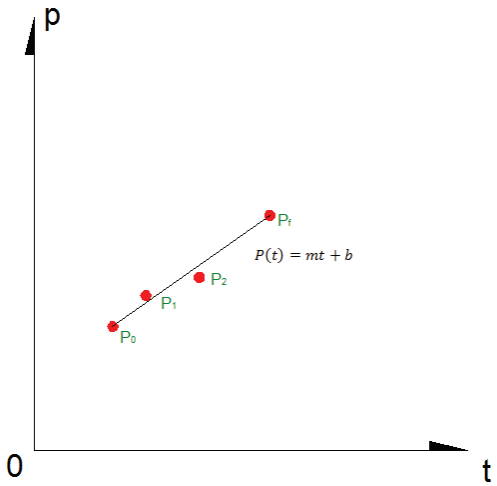


Figura 2.33 Interpolación lineal

2.4.1.2 Interpolación Polinómica [24]

La interpolación polinómica es la aproximación de la ruta a una función polinómica que tiene la siguiente forma general:

$$P(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + \cdots + a_n t^n \tag{2.18}$$

Para el cálculo del polinomio que representará un conjunto de puntos existen varias técnicas entre las cuales se destacan:

- Interpolación directa: se basa en el desarrollo de un sistema de ecuaciones para determinar las constantes del polinomio de grado n, para un conjunto de n nodos.

$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} 1 & t_0 & \dots & t_0^n \\ 1 & t_1 & \dots & t_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \dots & t_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \tag{2.19}$$

- Interpolación de Lagrange: parte del polinomio que tiene la forma

$$P(t) = \sum_{j=0}^{n} y_j l_j(t) \tag{2.20}$$

Donde l_j, son polinomios que dependen de los nodos tabulados y de las bases polinómicas

$$l_j = \prod_{i=0, i\neq j}^{k} \frac{t - t_i}{t_j - t_i} \tag{2.21}$$

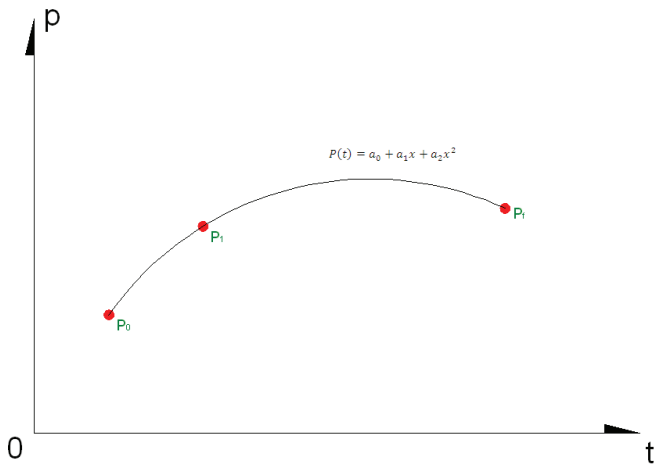


Figura 2.34 Ejemplo de Interpolación parabólica

La Figura 2.34 presenta la interpolación parabólica de un sistema que tiene tres nodos. La expresión polinómica se reduce a una de segundo orden que puede ser calculada a partir de la siguiente expresión:

$$P(t) = a_0 + a_1 t + a_2 t^2 \tag{2.22}$$

2.4.1.3 Interpolación Spline [23]

La interpolación Spline se realiza por segmentos a la cual se le asigna una función polinómica para cada segmento obligando de la curva pase por todos los puntos manteniendo la continuidad a lo largo del rango de acción de la ruta (Figura 2.35).

Este tipo de interpolación tiene la ventaja de utilizar funciones polinómicas de grado inferior, simplificando la complejidad de cálculo de las constantes de una función más compleja.

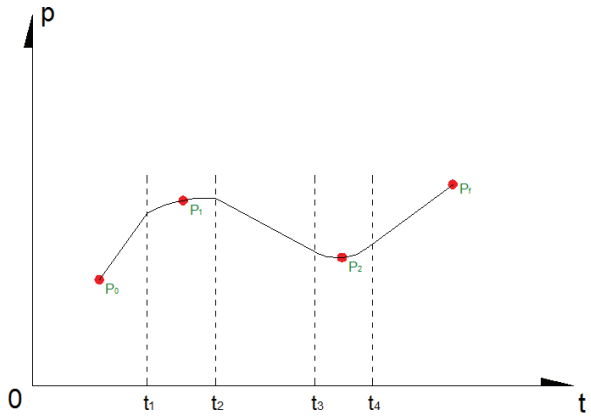


Figura 2.35 Interpolación Spline

$$P(t) = \begin{cases} a_1 t + a_0 & \forall \quad t_0 \leq t < t_1 \\ b_2 t^2 + b_1 t + b_0 & \forall \quad t_1 \leq t < t_2 \\ c_1 t + c_0 & \forall \quad t_2 \leq t < t_3 \\ d_2 t^2 + d_1 t + d_0 & \forall \quad t_3 \leq t < t_4 \end{cases} \tag{2.23}$$

2.4.2 FUNCIÓN TRAYECTORIA

Desde el punto de vista físico la trayectoria es un lugar geométrico que representa el cambio de posiciones de una partícula o cuerpo en el tiempo.

Matemáticamente la trayectoria es una función vectorial parametrizada en el tiempo expresada por:

$$\vec{\boldsymbol{r}} = f_x(t)\vec{\boldsymbol{\imath}} + f_y(t)\vec{\boldsymbol{\jmath}} + f_z(t)\vec{\boldsymbol{k}} \tag{2.24}$$

A diferencia de la ruta la trayectoria tiene toda la información del comportamiento cinemático de un cuerpo, como la velocidad y la aceleración.

La velocidad es una variable física que se calcula a partir del desplazamiento medido en el tiempo, y su representación general es:

$$\vec{\boldsymbol{v}}(t) = \frac{d}{dt}\vec{\boldsymbol{r}}(t) \tag{2.25}$$

La velocidad también es una función que depende del tiempo, y determina el comportamiento del cuerpo, más adelante en este proyecto se aplicará este concepto para el control de trayectoria.

En el campo de la robótica normalmente se utiliza tablas que contiene la información de la trayectoria generadas por la interpolación para obtener los nodos de la misma, a las cuales se incluye el parámetro del tiempo para alcanzarlas Tabla 2.1. (Discretización de la función camino) [21].

Tabla 2.1 Representación de trayectoria en posiciones y tiempos

Punto	x	y	t
0	x_o	y_o	t_o
1	x_1	y_1	t_1
2	x_2	y_2	t_2
	.	.	.
i	x_i	y_i	t_i
.	.	.	.
n	x_n	y_n	t_n

Estos tiempos se calculan a partir de las capacidades del robot y de su entorno de trabajo, las velocidades y aceleraciones seguras para que el robot cumpla la trayectoria satisfactoriamente.

2.4.3 PLANIFICACIÓN DE TRAYECTORIAS [21]

La planificación de trayectorias (Muñoz 1995), consiste en construir una trayectoria discreta a partir de una ruta, donde la determinación del camino tenga características específicas, para luego generar la trayectoria a partir de la planificación de un perfil de velocidades en las que se considera las condiciones mecánicas, cinemáticas, dinámicas del robot, y de las características de la trayectoria construida, para realizar el seguimiento de maximizando la velocidad del robot.

Tabla 2.2 Planificación de trayectorias (Muñoz 1995)

Punto λ	$\vec{P}(\lambda)$	$\vec{\boldsymbol{v}}(\lambda)$	t
0	$\vec{\boldsymbol{P}}_o$	$\vec{\boldsymbol{v}}_o$	t_o
1	$\vec{\boldsymbol{P}}_1$	$\vec{\boldsymbol{v}}_1$	t_1
2	$\vec{\boldsymbol{P}}_2$	$\vec{\boldsymbol{v}}_2$	t_2
	.	.	.
i	$\vec{\boldsymbol{P}}_i$	$\vec{\boldsymbol{v}}_i$	t_i
.	.	.	.
n	$\vec{\boldsymbol{P}}_n$	$\vec{\boldsymbol{v}}_n$	t_n

Este planificador de trayectoria se divide en dos pasos. El primero es la construcción de la trayectoria y el segundo es la generación del perfil de velocidades en función de las restricciones del robot y la trayectoria.

Para la construcción de la trayectoria, se utiliza la interpolación por splines (propuesta por Muñoz) para generar una expresión parametrizada $\vec{\boldsymbol{P}}(\lambda)$, que no tenga aristas suavizando el camino, y que depende el tiempo t. Esta función se discretiza en segmentos iguales incluyendo las curvaturas del camino.

El siguiente paso es la conformación del perfil de velocidades que asocie la función $\vec{\boldsymbol{P}}(\lambda)$ y formar la trayectoria, para lo cual se parte de las restricciones de velocidad.

$$\forall\, \vec{\boldsymbol{P}}(\lambda_i) \;\rightarrow\; \vec{\boldsymbol{V}}(s_i) \tag{2.26}$$

Donde:

- s_i: Es la longitud entre el punto $i-esimo$ y su anterior del camino.

2.4.3.1 Restricciones físicas de velocidad

Las restricciones de velocidad determinan el perfil de velocidades aplicadas al robot para cada punto de la trayectoria, las restricciones físicas de velocidad están relacionadas con las capacidades y la estructura del robot, de las cuales se

destacan, las restricciones mecánicas (ME) en la que se involucra la parte motriz del robot que incluye velocidades y aceleraciones $\vec{\boldsymbol{v}}_{ME}(\lambda)$.

Otra componente de restricción de velocidad es el modelo cinemático del robot (MC), ya que está asociado a la capacidad de movilidad como un sistema en movimientos de traslación, rotación y la combinación de ellos $\vec{\boldsymbol{v}}_{MC}(\lambda)$. Y las restricciones por parte del comportamiento dinámico (CD), ocasionado por el tiempo de reacción que tiene el sistema ante cambios de la trayectoria como en el caso de las curvaturas producidos por fuerzas e inercias del robot obteniendo el perfil de velocidad del comportamiento dinámico $\vec{\boldsymbol{v}}_{CD}(\lambda)$.

2.4.3.2 Restricciones operacionales de velocidad

Estas restricciones se relacionan con el ambiente de trabajo del robot, la restricción por distancia al objetivo (DO) se calcula la velocidad máxima a la que debe pasar el robot cerca de los obstáculos en función de la capacidad de frenado del robot cuando se tiene un mapa de entorno con obstáculos móviles o un entorno multi-robot $\vec{\boldsymbol{v}}_{DO}(\lambda)$, y la otra restricción es la de velocidad de seguridad $\vec{\boldsymbol{v}}_{S}(\lambda)$, la cual planifica la velocidad en función de la actividad que realiza el robot y de la trayectoria que sigue el robot la cual se puede planificar por tramos

Con estos parámetros que restringen la velocidad del robot, se asigna a cada punto del camino generado se asigna una velocidad para alcanzar el siguiente (Tabla 2.3), para determinar la velocidad para cada punto se utiliza un gráfico velocidad-espacio (Shiller y Gwo, 1991), que asigna una velocidad segura para cada punto del camino en función del la longitud s.

Tabla 2.3 Formación de perfil de velocidades

Punto λ	$\vec{\boldsymbol{v}}_{ME}(\lambda)$	$\vec{\boldsymbol{v}}_{MC}(\lambda)$	$\vec{\boldsymbol{v}}_{DC}(\lambda)$	$\vec{\boldsymbol{v}}_{DO}(\lambda)$	$\vec{\boldsymbol{v}}_{S}(\lambda)$
0	$\vec{\boldsymbol{v}}_{ME}(0)$	$\vec{\boldsymbol{v}}_{MC}(0)$	$\vec{\boldsymbol{v}}_{DC}(0)$	$\vec{\boldsymbol{v}}_{DO}(0)$	$\vec{\boldsymbol{v}}_{S}(0)$
1	$\vec{\boldsymbol{v}}_{ME}(1)$	$\vec{\boldsymbol{v}}_{MC}(1)$	$\vec{\boldsymbol{v}}_{DC}(1)$	$\vec{\boldsymbol{v}}_{DO}(1)$	$\vec{\boldsymbol{v}}_{S}(1)$
2	$\vec{\boldsymbol{v}}_{ME}(2)$	$\vec{\boldsymbol{v}}_{MC}(2)$	$\vec{\boldsymbol{v}}_{DC}(2)$	$\vec{\boldsymbol{v}}_{DO}(2)$	$\vec{\boldsymbol{v}}_{S}(2)$

	.	.	.	.	.
i	$\vec{\boldsymbol{v}}_{ME}(i)$	$\vec{\boldsymbol{v}}_{MC}(i)$	$\vec{\boldsymbol{v}}_{DC}(i)$	$\vec{\boldsymbol{v}}_{DO}(i)$	$\vec{\boldsymbol{v}}_{S}(i)$
.	.	.	.	.	.
n	$\vec{\boldsymbol{v}}_{ME}(n)$	$\vec{\boldsymbol{v}}_{MC}(n)$	$\vec{\boldsymbol{v}}_{DC}(n)$	$\vec{\boldsymbol{v}}_{DO}(n)$	$\vec{\boldsymbol{v}}_{S}(n)$

Trazando curvas a partir de las velocidades obtenidas por las restricciones, se obtiene una curva del perfil de velocidades para toda la trayectoria de los cuales los la región en la que convergen todas las velocidades en un punto representa la velocidad asociada a la trayectoria $\vec{\boldsymbol{V}}(\lambda)$ (Figura 2.36).

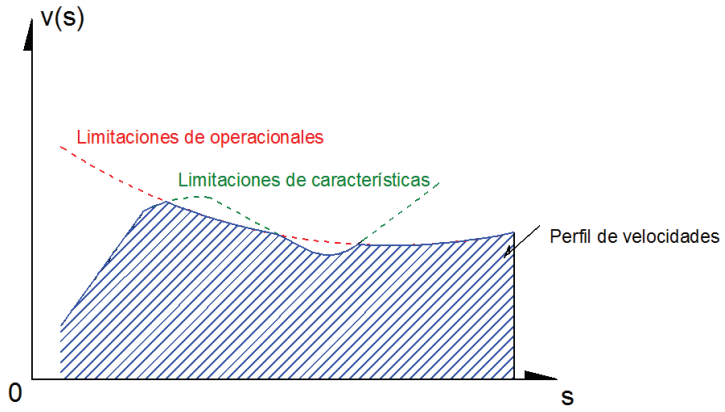


Figura 2.36 Limitaciones de velocidad en el plano espacio-velocidad [21]

Matemáticamente la comparación es:

$$\vec{\boldsymbol{V}}(\lambda) = \min\left(\vec{\boldsymbol{v}}_{ME}(\lambda), \vec{\boldsymbol{v}}_{MC}(\lambda), \vec{\boldsymbol{v}}_{CD}(\lambda), \vec{\boldsymbol{v}}_{CO}(\lambda), \vec{\boldsymbol{v}}_{s}(\lambda)\right) \tag{2.27}$$

2.4.4 CONTROL DE TRAYECTORIA POR MÉTODOS NUMÉRICOS [20]

Un método numérico son algoritmos matemáticos y/o lógicos (operaciones simples) que permiten aproximar matemáticamente un proceso real complejo que no puede ser modelo por una expresión matemática exacta.

El control de trayectorias para robot por métodos numéricos consiste en encontrar la acción de control en función de la trayectoria deseada y el modelo dinámico del robot, conociendo el valor de las variables en tiempo discreto.

Considerando la siguiente ecuación matricial:

$$\dot{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, t) \tag{2.28}$$

Y tiene la siguiente condición inicial:

$$\boldsymbol{y}(0) = \boldsymbol{y}_0 \tag{2.29}$$

Donde:

- $\boldsymbol{y}$: Es la posición del robot.
- $\boldsymbol{u}$: Acción de control.
- t: tiempo de ejecución del sistema.

Discretizando la expresión (2.28) sabiendo que $t = kT_0$, donde T_0 es el periodo de muestreo y k un variable entera positiva. La expresión se transforma en:

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + \int_{kT_0}^{(k+1)T_0} \boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, t)dt \tag{2.30}$$

Aplicando el método Euler se puede aproximar la expresión (2.30), para determinar el control aplicado al robot ($\boldsymbol{u}_k$) y alcanzar la posición deseada, para lo cual se conoce la posición actual del robot ($\boldsymbol{y}_k$), el periodo de muestreo (T_0), y la siguiente posición del robot ($\boldsymbol{y}_{k+1}$), que se la obtiene a partir de la trayectoria deseada.

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k + T_0\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, t) \tag{2.31}$$

La función $\boldsymbol{f}(\boldsymbol{y}, \boldsymbol{u}, t)$, se incluye a partir del modelo cinemático o dinámico del robot utilizado, el cual se le discretiza aplicando la aproximación de Euler (2.31).

Tomando en cuenta el modelo cinemático del robot omnidireccional de tres ruedas en forma matricial [9] se tiene.

$$\begin{pmatrix} v_x \\ v_y \\ \omega_r \end{pmatrix} = \begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{pmatrix} \begin{pmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \end{pmatrix} \tag{2.32}$$

Donde:

- $v_x,\ v_y,\ w_r$: Componentes de velocidad del robot omnidireccional de tres ruedas.

- $u_{\omega_1},\ u_{\omega_2},\ u_{\omega_3}$: Velocidad aplicada a los motores del robot.

Reescribiendo la expresión (2.31) en forma matricial aplicando el modelo cinemático del robot omnidireccional de tres ruedas (2.32), se obtiene:

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \\ \omega_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \\ \omega_k \end{pmatrix} + To \begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{pmatrix} \begin{pmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \end{pmatrix} \tag{2.33}$$

Donde:

- $x_{k+1},\ y_{k+1},\ \omega_{k+1}$: son las componentes de la posición y orientación del robot a la que se quiere alcanzar.

- $x_k,\ y_k,\ \omega_k$: Son las componentes de la posición y orientación del robot actual.

Reescribiendo la expresión en forma compacta:

$$\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + To\boldsymbol{A}\boldsymbol{U}_k \tag{2.34}$$

Donde:

- $\boldsymbol{X}_{k+1}$: es la posición de la trayectoria a la que el robot debe llegar

Para calcular las velocidades aplicadas a los motores se despeja la matriz $\boldsymbol{U}_k$:

$$\boldsymbol{U}_k = \frac{1}{To}(\boldsymbol{X}_{k+1} - \boldsymbol{X}_k)\boldsymbol{A^{-1}} \tag{2.35}$$

$$\begin{pmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \end{pmatrix} = \frac{1}{To}\begin{pmatrix} xd_{k+1} - x_k \\ yd_{k+1} - x_k \\ \omega d_{k+1} - \omega_k \end{pmatrix}\begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{pmatrix}^{-1} \tag{2.36}$$

CAPÍTULO 3

DESARROLLO DEL PROGRAMA DE CONTROL

En el presente capítulo se desarrollan tres métodos planificadores, para implementarlos en un programa computacional. Adicionalmente se desarrolla el programa de control de ruta y control de trayectoria para el Robotino® de Festo aplicando la ruta obtenida por el método planificador seleccionado en un ambiente real.

Para la selección de la plataforma de programación primero se toma en cuenta los siguientes requerimientos:

- Plataforma de programación que cuente con lenguaje de alto nivel.

- Ingreso, procesamiento, y presentación en un entorno gráfico de las variables obtenidas para el control del robot.

- Capacidad de generar archivos y bases de datos.

- Capacidad de comunicación por medio del Wireless con el Robotino® de Festo.

- Visualización del interface con el Usuario (HMI) de sencilla configuración.

Con estas consideraciones la plataforma de programación LabVIEW 10.0 se ajusta a todos los requerimientos incluyendo la capacidad de comunicarse con el Robotino® de Festo para monitorear sus sensores, encoders y controlar sus motores.

La plataforma de programación LabVIEW permite el desarrollo de programas de control de cualquier categoría dentro de un entorno gráfico. La plataforma cuenta

con dos paneles, el panel frontal cumple el papel de ser la interface con el usuario (HMI), a la que se le coloca los instrumentos de control, de visualización, registros auxiliares e interfaces con otros programas.

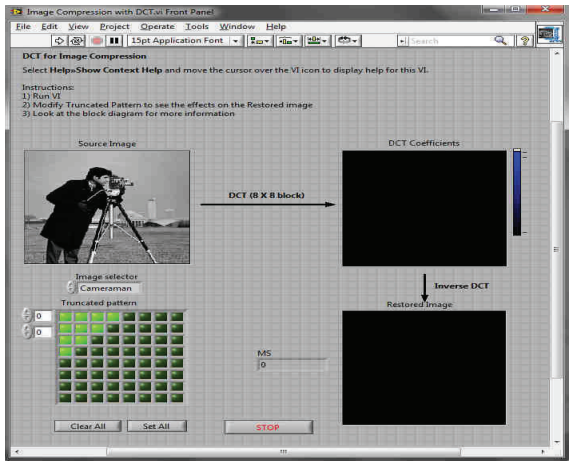


Figura 3.1 Panel Frontal de LabVIEW 10.0

El panel de diagrama de bloques, cuenta con funciones de programación, lógicas, matemáticas, transformación de variables, procesamiento de señales, manejo de dispositivos externos, manejo de protocolos de comunicación, librerías para sistemas de control, manejo de programas instalados, utilización de base de datos y archivos, entre otros.

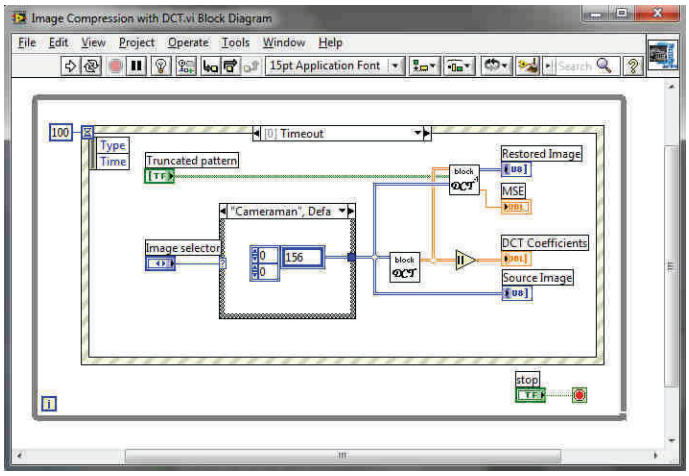


Figura 3.2 Diagrama de Bloques de LabVIEW 10.0

También cuenta con librerías que facilitan la programación para la planeación de trayectorias, simulación de brazos robóticos y otras funciones que se abordarán más adelante en este capítulo.

3.1 ESTRUCTURA DEL PROGRAMA DE CONTROL

Para el diseño del programa de control se toma de punto de partida un diagrama de bloques general que incorpore todas las rutinas a desarrollarse y otros componentes fundamentales.

En base al objetivo de esta Tesis, requerimientos de los métodos planificadores, la ejecución de la ruta en el Robotino® de FESTO, y las capacidades de la plataforma de programación, el software de planificación y control se desarrolló con la siguiente estructura:

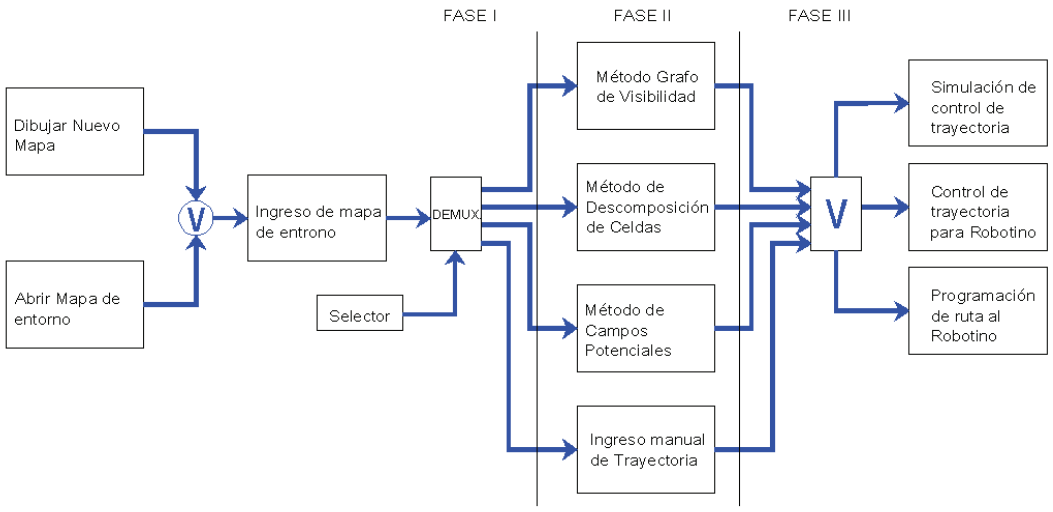


Figura 3.3 Diagrama de bloques general del programa de control

El diagrama de bloques de la Figura 3.3 presenta las opciones y la conformación de las rutinas de los métodos planificadores el cual se divide en tres fases facilitando la posterior programación.

La primera fase o de modelado del mapa de entorno, se encarga en creación y edición de un mapa.

- *Dibujar nuevo mapa:* es la opción que tiene el usuario para introducir un nuevo mapa.

- *Abrir mapa de entorno*: opción que permite abrir un archivo previamente guardado.

- *Grabar mapa de Archivo*: opción que guarda el mapa de entorno terminado, y la ruta obtenida.

- *Ingreso de mapa de entorno*: esta opción visualiza el mapa de entorno, y controla los otros bloques en la edición.

En la segunda fase o generadores de ruta, se escoge el método planificador de ruta a través del multiplexor de la primera fase, y ejecuta el método de planificación seleccionado. Se compone de los siguientes bloques:

- *Método Grafo de Visibilidad*.

- *Método de Descomposición de Celdas.*

- *Método de Campos Potenciales.*

- *Ingreso manual de Ruta.*

La tercera fase o de control de ruta y trayectoria, se encarga de la construcción del camino a partir de la ruta obtenida por el método planificador, y reproducirlos en un simulador de control de trayectorias y el control para el Robotino®, tanto para ruta como para el control de trayectoria.

3.2 PROGRAMACIÓN DE LA INTERFACE CON EL USUARIO

La interface con el usuario o HMI incorpora las herramientas para dibujar, ingresar, editar, guardar el mapa de entorno, visualizar variables, ubicación de la configuración inicial P_0 y final P_f de la ruta, ejecución de métodos de planificación de rutas, el control de ruta y trayectoria para el Robotino® de FESTO. Para lo cual se parte del siguiente diagrama de flujo basado en el diagrama de bloques general (Figura 3.3)

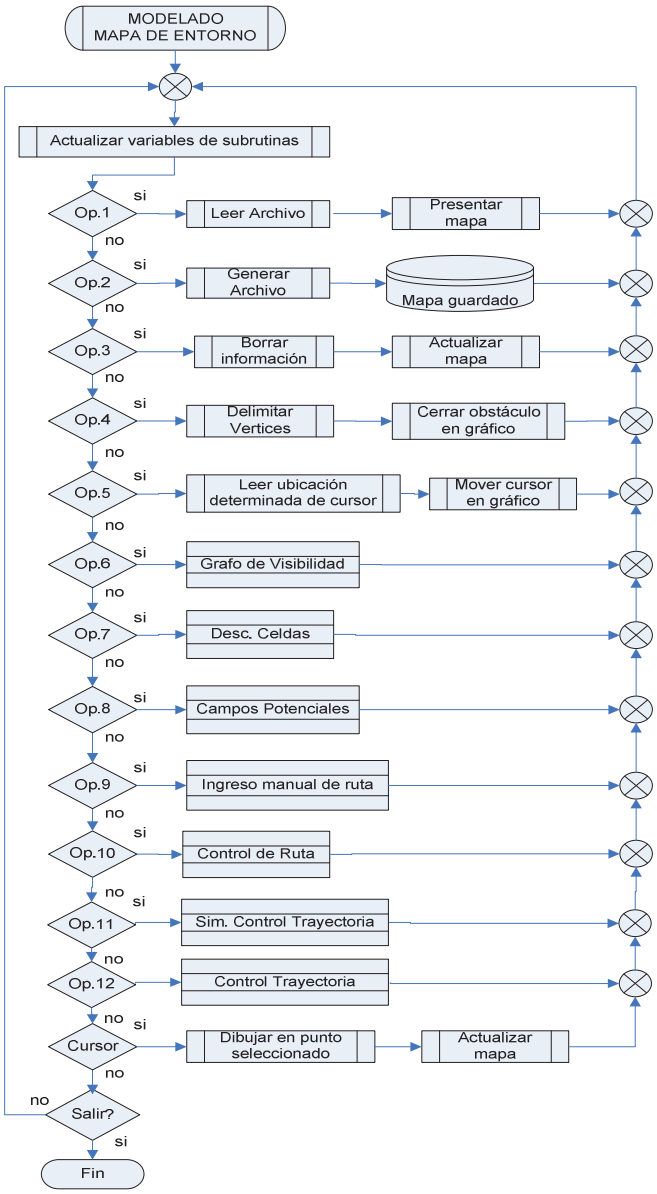


Figura 3.4 Diagrama de flujo del programa principal

Este diagrama de flujo de la Figura 3.4 incorpora todas las funciones del programa principal para el diseño del HMI con las selecciones y presentación del mapa de entorno.

3.2.1 PANEL FRONTAL DEL HMI

Considerando los datos que ingresan, los controladores, los visualizadores y funcionalidades que presenta el diagrama de flujo de la Figura 3.4, el HMI para la presentación de la información de los métodos planificadores y control sobre el Robotino®, es el siguiente:

Figura 3.5 Panel Frontal del programa principal (HMI)

La Figura 3.5 presenta el HMI, cuyos controles y visualizadores son:

- Configuración inicial P_0.
- Configuración final P_f.
- Mapa de entorno, (dibujo y parámetros de relevancia).

- Ruta calculada por el método planificador.

- Selectores de edición de mapa y ejecución de métodos planificadores.

Para la visualización del mapa de entorno y la ruta generada, se utiliza un plano (GRAPH x, y) presentado en la Figura 3.6.

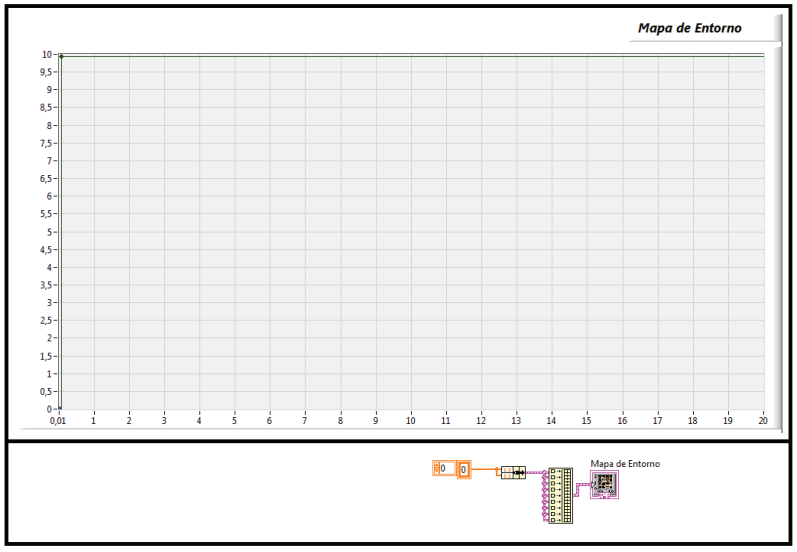


Figura 3.6 Plano de visualización de mapa de entorno y ruta

El plano de visualización tiene la capacidad de presentar los gráficos utilizando variables tipo ARRAY, y puede ser configurado para presentar varias curvas a la vez, configuración de colores y tipos de líneas, permitiendo ver el mapa de entorno y el método planificador de rutas en el mismo gráfico.

Las variables tipo ARRAY permiten el ingreso de información en cualquier formato. La Figura 3.7 presenta el empleo de estas variables para el ingreso de las configuraciones inicial y final.

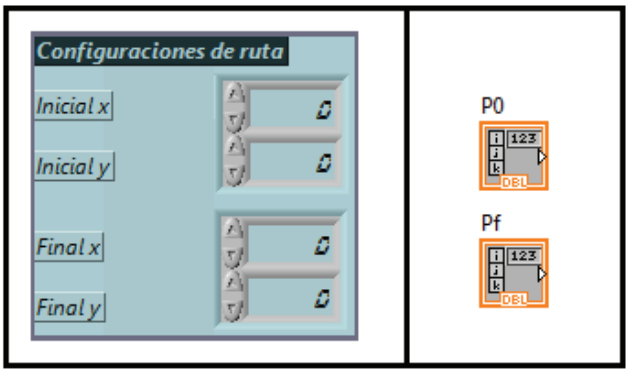


Figura 3.7 Ingreso de configuración inicial y final

Estos controles están identificados y configurados para ingresar por medio del teclado o utilizando los botones de incremento o disminución de las variables.

En el diagrama de flujo del programa principal de la Figura 3.4, los condicionales son representados por pulsantes, y cada uno sigue la función encomendada (Figura 3.7 y 3.8).

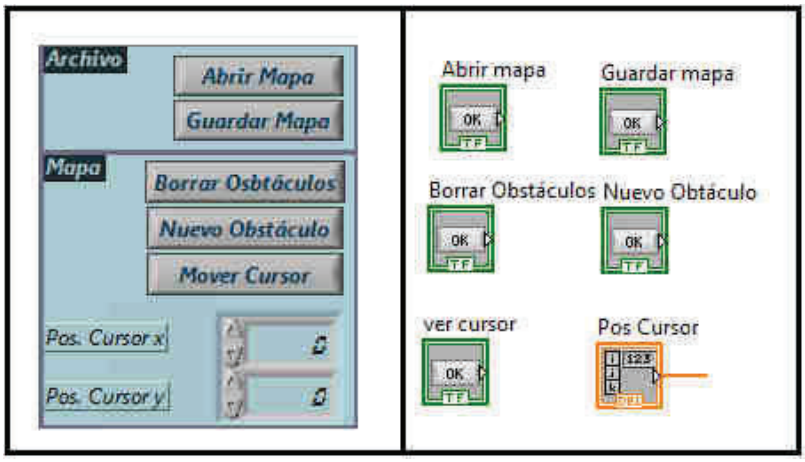


Figura 3.8 Pulsantes para edición de mapa de entorno

Los pulsantes de la Figura 3.8 son utilizados para la edición del mapa de entorno, y los de la Figura 3.9 para selección del método de planificación y programación del Robotino® de FESTO.

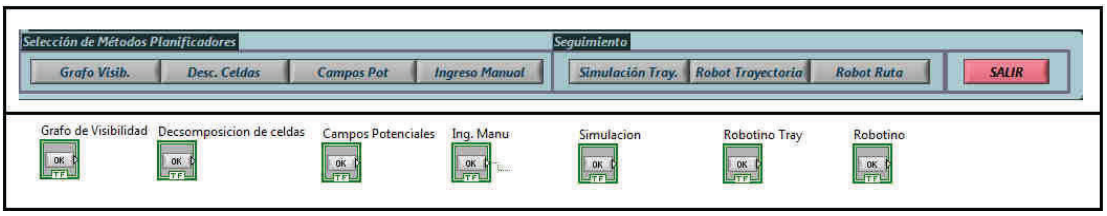


Figura 3.9 Pulsantes para ejecución de subrutinas

Además de los datos presentados en el plano, también se toma en cuenta otros parámetros que determinan el rendimiento del método planificador, para ello se incorporaron dos indicadores de distancia y tiempo.

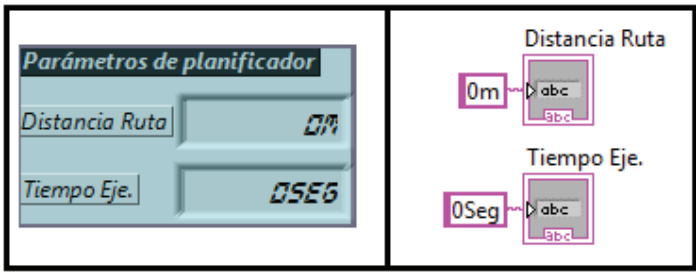


Figura 3.10 Visualización de parámetros adicionales

Estos indicadores presentan la distancia total de la ruta obtenida y el tiempo de ejecución del método planificador.

3.2.2 DESARROLLO DEL DIAGRAMA DE FLUJO PRINCIPAL

El diseño de HMI (Figura 3.5) permite el ingreso y visualización de los datos. La lógica del programa de control se desarrolla en base al diagrama de flujo del programa principal (Figura 3.4).

El programa se compone de un lazo WHILE que lo ejecuta a menos que se quiera salir, esta opción se ejecuta presionado el botón de salir (Figura 3.11).

Los botones de selección para la edición del mapa de entorno, y de llamado a los planificadores son las opciones del diagrama de flujo (Figura 3.4), que son incorporadas al programa con un bloque de estructura de eventos.

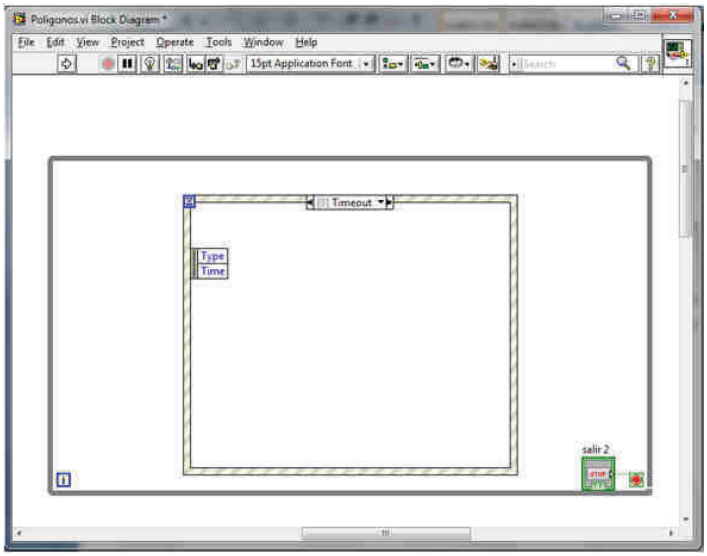


Figura 3.11 Estructura de diagrama de flujo

Las opciones del diagrama de flujo tienen una función específica que se describen a continuación:

- *Opción 1 (Abrir Mapa):* se abre un archivo previamente creado (.dat) y se carga en el plano del panel frontal.

- *Opción 2 (Guardar mapa):* Toma todos los datos de los gráficos y los coloca en un archivo (.dat), para posteriormente guardarlos en la dirección que el usuario desee (Figura 3.12).

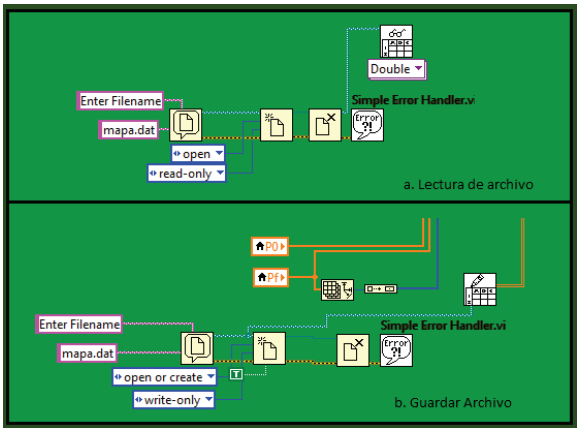


Figura 3.12 Diagrama de bloques para generador de archivos

- *Opción 3 (Borrar Obstáculos):* Borra la información de las variables del mapa de entorno.

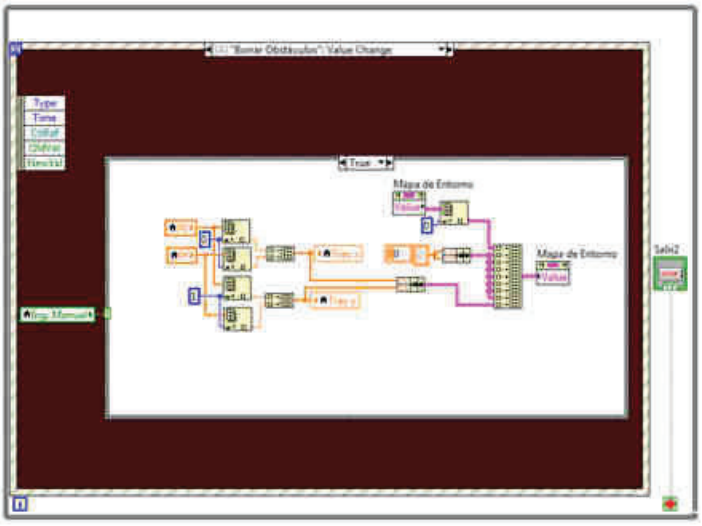


Figura 3.13 Diagrama de bloques para borrar obstáculos

- *Opción 4 (Nuevo Obstáculo):* Termina automáticamente el obstáculo que está dibujando e inicia el siguiente.

- *Opción 5 (Mover Cursor):* Mueve el cursor a la posición determinada por el control de coordenadas.

La Figura 3.13 presenta el diagrama de bloques de la opción mover cursor, que tiene la misión de borrar el mapa de entorno.

- *Opción 6 (Grafo Visib.):* Llama a la rutina que ejecuta el algoritmo de planificación de ruta grafo de visibilidad.

- *Opción 7 (Desc. Celdas):* Llama a la rutina que ejecuta el algoritmo de planificación de ruta Descomposición de Celdas Exactas.

- *Opción 8 (Campos Pot.):* Llama a la rutina que ejecuta el algoritmo de planificación de ruta Campos Potenciales.

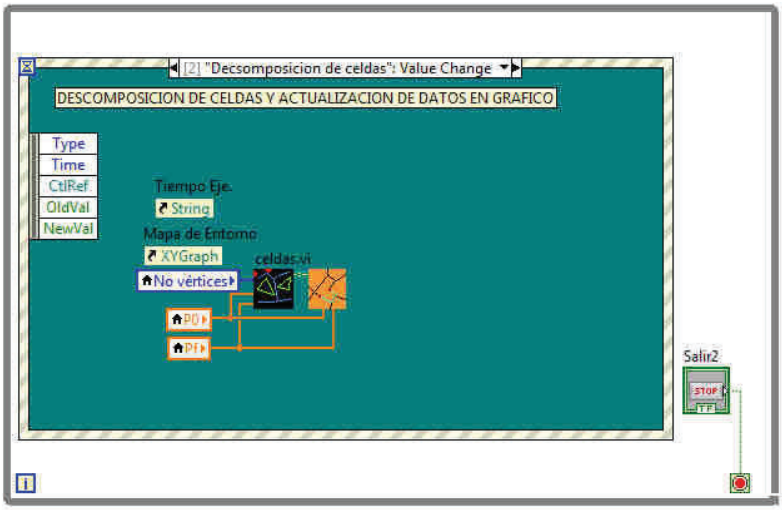


Figura 3.14 Diagrama de bloques de Campos Potenciales

La Figura 3.14 muestra la opción que llama a la rutina que ejecuta el algoritmo de planificador de ruta con los parámetros requeridos.

- *Opción 9 (Ingreso Manual):* Permite dibujar en el plano una ruta utilizando el cursor, y editarlo utilizando los mismos comandos que el ingreso de los obstáculos.

- *Opción 10 (Robot):* Llama a la rutina que envía a los datos al robot para reproducir los movimientos.

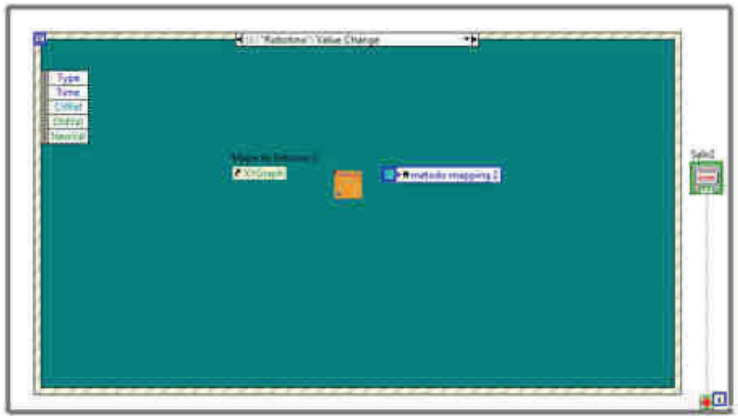


Figura 3.15 Diagrama de bloques para ejecución al Robotino® de Festo

- *Opción 11 (Simulación Tray):* Rutina que realiza el control de trayectoria utilizando un simulador del Robotino®.

- *Opción 12 (Robot Trayectoria):* Rutina que realiza el control de trayectoria a partir de la ruta obtenida por el método planificador al Robotino®.

Al ver el estado de las opciones de selección, el programa actualiza todas las variables y continúa con la ejecución del lazo principal. En caso de llamado a rutina el programa actualiza los valores de sus variables durante la ejecución.

3.2.3 HOMOLOGACIÓN DE DATOS

Debido al diseño del diagrama de flujo (Figura 3.4), las rutinas de planificación de rutas y la de control para el Robotino® de FESTO necesitan que la información proveniente de las configuraciones inicial P_0, final P_f y la del mapa de entorno sea compatible, y a su vez la información de la ruta obtenida también lo sea con el HMI, y con la rutina de control de ruta y de trayectoria para incrementar el rendimiento y estandarizar el procedimiento de programación de los planificadores de rutas.

Para el proceso de homologación los datos deben tener solamente la información mínima necesaria del mapa de entorno, dejando el análisis de identificación del mapa de entorno a las rutinas de planificación de rutas.

Según la teoría tanto en el método de Grafo de Visibilidad como el de Descomposición de Celdas (Capítulo 2) los obstáculos deben ser representados como polígonos, y el método de campos potenciales se puede utilizar estos para representar obstáculos, por lo tanto el ingreso del mapa de entorno al programa se los realiza por medio de este esquema.

Para dibujar un polígono en el plano, los vértices son puntos coordenados que determinan la posición del polígono en el plano y su geometría.

La Figura 3.16 demuestra que la secuencia de los vértices determina en este caso sí el polígono es un rectángulo o un polígono complejo cóncavo irregular.

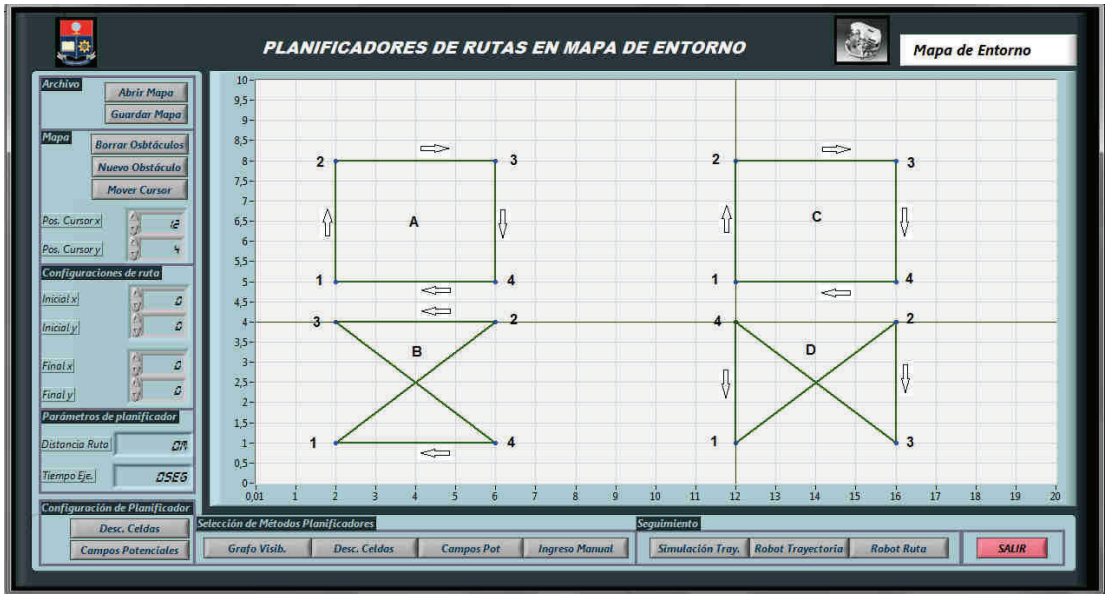


Figura 3.16 Secuencia de vértices

Entonces la información necesaria para identificar el mapa de entorno completo es:

- Número de Obstáculos.

- Número y secuencia de vértices de cada obstáculo.

- Ubicación de estos vértices en el plano.

Y para generar la ruta se necesita las configuraciones inicial P_0 y final P_f.

Toda la información se ingresa en registros de tipo ARRAY, donde el número de obstáculos se obtiene al pulsar el botón *nuevo obstáculo*, la secuencia al momento de dibujar y guardar en estos registros en el mismo orden y las coordenadas son los valores numéricos de los vértices guardados.

Los valores presentados pertenecen al rectángulo derecho de la Figura 3.16, en los registros están los vértices y el orden está en concordancia con la forma final presentada en el plano, además se presenta el número de vértices del polígono y el número total de obstáculos dentro del mapa.

Con estas consideraciones para el ingreso de información del mapa de entorno y las configuraciones, se procede al desarrollo de los algoritmos de planificación de rutas.

3.3 PROGRAMA DESCOMPOSICIÓN DE CELDAS

El algoritmo Descomposición de Celdas en todas sus clasificaciones se basa en la creación de celdas sobre la zona libre de obstáculos, la formación de los nodos que representa cada celda, la construcción del grafo de conectividad, y el ordenamiento de los vértices de los obstáculos. Entonces se escoge el método de Descomposición de Celdas Exactas Vertical para incorporar al programa.

Según el diagrama de flujo del programa principal (Figura 3.4) los programas de planificación de rutas son rutinas que contiene el HMI (Figura 3.5), que son ejecutados cuando se seleccionan por parte del usuario del programa.

Con estas premisas se desarrolla el programa de control en base al siguiente diagrama de flujo que explica el procedimiento de ejecución del programa.

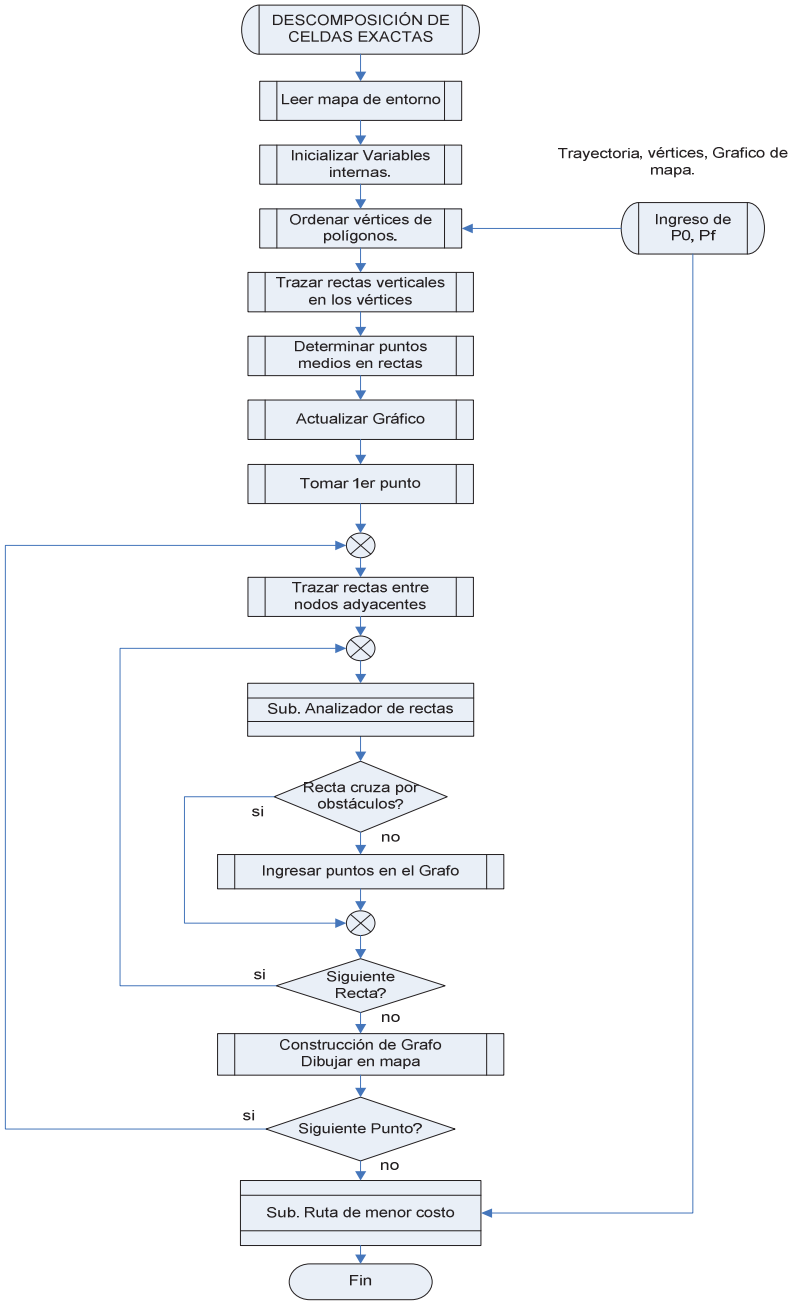


Figura 3.17 Diagrama de flujo para Descomposición de Celdas Vertical

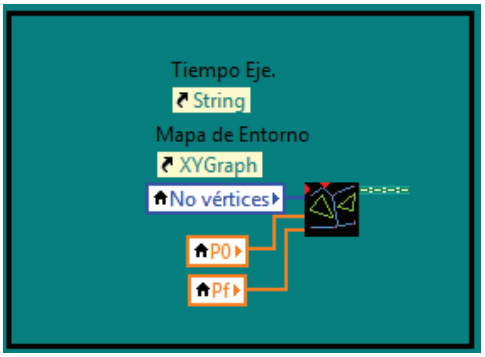


Figura 3.18 Bloque de la rutina de Descomposición de Celdas

La Figura 3.18 presenta la rutina del método de Descomposición de celdas Exactas Vertical. Al programa se ingresa los datos requeridos del mapa de entorno y configuraciones inicial P_0 y final P_f.

El panel frontal de la subrutina (Figura 3.19) presenta las variables de ingreso, las de procesamiento, y las de salida, presentando la construcción del grafo en el HMI.

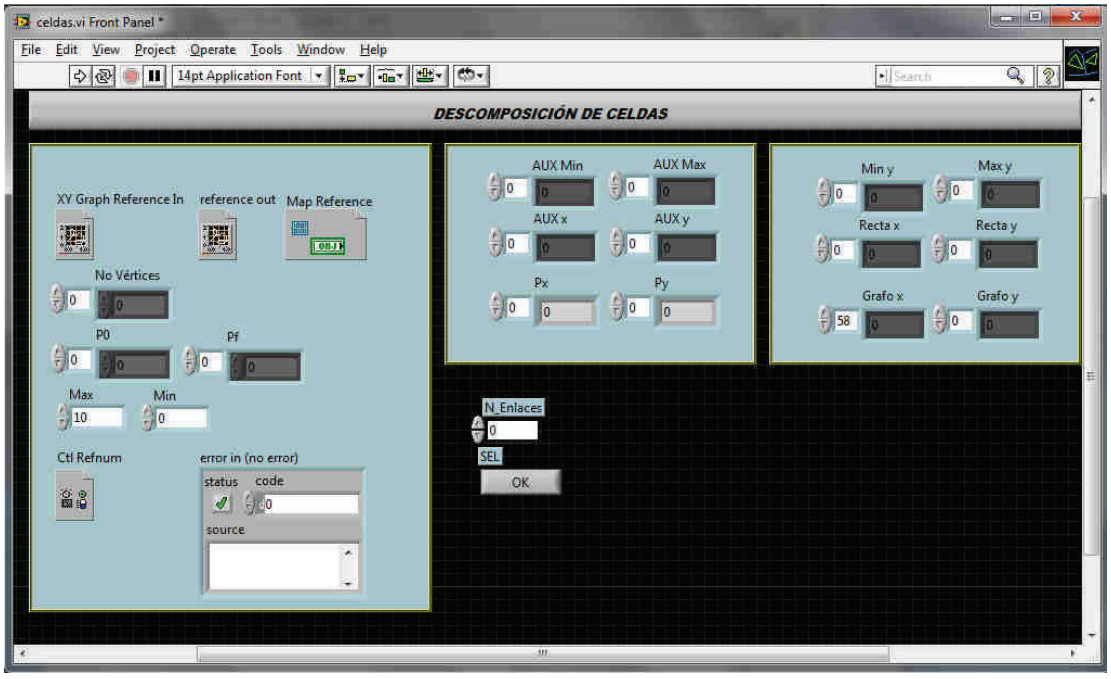


Figura 3.19 Panel frontal de rutina de Descomposición de Celdas

El programa toma los vértices del mapa de entorno, ordena todos los polígonos de izquierda a derecha del plano para posteriormente almacenarlos en una variable tipo ARRAY.

Ordenar los vértices disminuye la complejidad de programación, y estandariza el proceso, permitiendo ingresar aleatoriamente los obstáculos en el plano del HMI.

Con los polígonos ordenados, se trazan las rectas verticales y generar las celdas, formadas por polígonos triangulares o trapezoidales limitados por el contorno del plano y por los obstáculos.

Una recta vertical tiene la siguiente función característica.

$$x = x_v \tag{3.1}$$

Donde x_v es una constante proporcionada por la posición x del vértice de un obstáculo.

Estas rectas son dibujadas en el plano del programa principal (Figura 3.20).

Figura 3.20 Generación de puntos adyacentes para construcción de grafo

Determinadas las rectas se calculan los puntos medios de cada una y son agregadas al plano, estos puntos son los nodos del grafo de conectividad.

Por lo general se obtiene un nodo por recta trazada, pero hay que tomar en cuenta las siguientes condiciones.

De los vértices del triángulo formado, los extremos deben formar dos nodos debido a que se generan las celdas hacia arriba y hacia abajo del obstáculo.

Otra condición se da en el caso que un polígono se dibuje debajo de otro, en ese caso se considera el contorno del segundo polígono ya que cambia los límites de la rectas verticales.

Obtenidos los nodos de las celdas hay que ordenarlos para formar el grafo de conectividad.

Antes de analizar la generación del grafo, se describe las subrutinas que hacen posible la ejecución del programa.

3.3.1 SUBRUTINA GENERADOR DE SÓLIDOS

La subrutina Generador de Sólidos es el programa base para la construcción de grafos, tanto para el método de Descomposición de Celdas y del Grafo de Visibilidad. Este programa se encarga de determinar el estado de pertenencia de un punto del plano respecto a un polígono dibujado.

El programa primero toma las coordenadas de los vértices y la secuencia de un polígono del mapa de entorno.

Con esta información se ordena los vértices del polígono y se determinan la orientación de los lados de los polígonos utilizando ecuaciones de rectas entre los vértices.

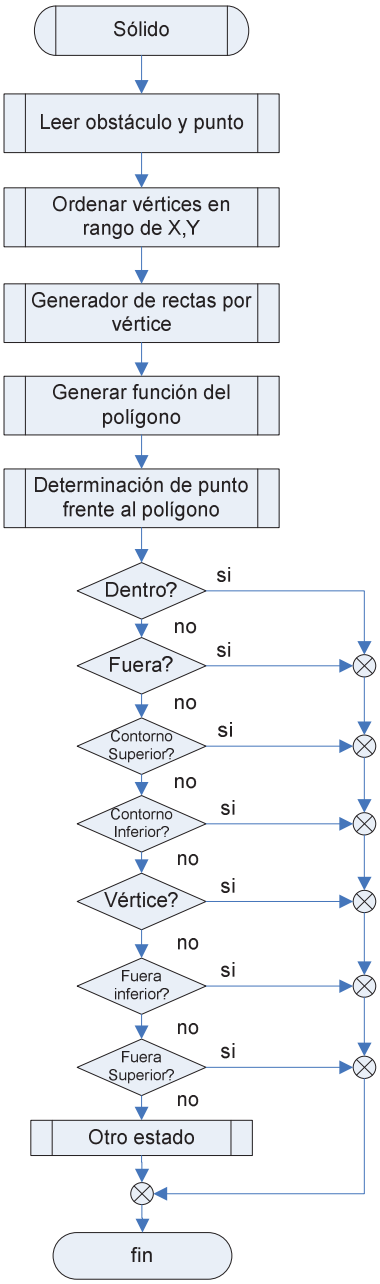


Figura 3.21 diagrama de flujo del Generador de sólidos

Considerando el polígono de la Figura 3.22:

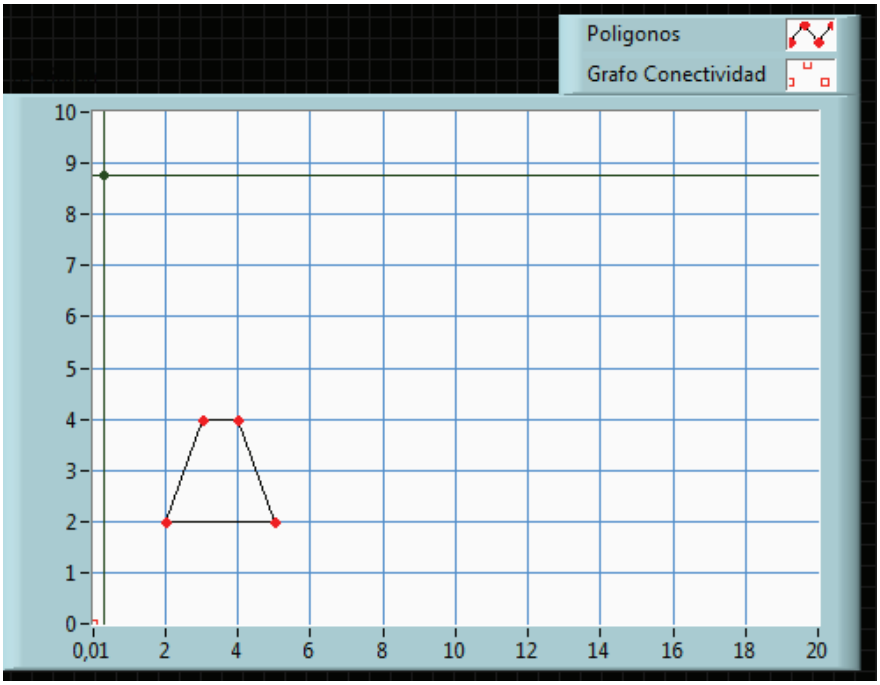


Figura 3.22 Ejemplo de la subrutina

Las rectas generadas son:

$$\begin{aligned} y_1 &= 2x_1 - 2; && \forall \quad 2 \leq x_1 \leq 3 \\ y_2 &= 4\,; && \forall \quad 3 \leq x_2 \leq 4 \\ y_3 &= -2x_3 + 12; && \forall \quad 4 \leq x_3 \leq 5 \\ y_4 &= 2; && \forall \quad 2 \leq x_4 \leq 5 \end{aligned} \tag{3.2}$$

Para la construcción de la expresión matemática del polígono se ordena las ecuaciones de tal manera que el programa considere siempre que el polígono se dibujó de izquierda a derecha.

$$y_p = \begin{cases} 2x - 2 & \forall \quad 2 \leq x \leq 3 \\ 2 & \forall \quad 2 \leq x \leq 5 \\ 4 & \forall \quad 3 \leq x \leq 4 \\ -2x + 12 & \forall \quad 4 \leq x \leq 5 \end{cases} \tag{3.3}$$

Este ordenamiento permite facilitar el análisis ya que el polígono pertenece al rango comprendido entre 2 y 5.

Todos los polígonos ya sean regulares o irregulares tienen la particularidad de tener un número par de soluciones en y para cada valor en x a excepción de los vértices, tomando en cuenta esta consideración se puede saber el rango que pertenece al interior del polígono.

Para el ejemplo planteado existen dos respuestas en y para cada valor en x a excepción de los extremos del polígono.

Sea $x = 3$, enonces:

$$y_{sup} = 4$$
$$y_{inf} = 2$$

Y por lo tanto el rango del polígono para $x = 3$ es: $2 \leq y \leq 4$

Con este análisis, se obtiene el rango para determinar el interior y el exterior del polígono.

Después la rutina toma las coordenadas de un punto ingresado al registro, este punto se analiza si está dentro del rango del polígono en el eje x, si pertenece al rango se evalúa en y, caso contrario el punto está fuera del polígono.

Para evaluar en y, se iguala la coordenada x del punto con las rectas del polígono que integren este punto dentro de su rango, se calcula los valores de y, y si el valor en y del punto está dentro del rango se considera al punto dentro del polígono. Si el punto está fuera del rango se analiza si el valor es mayor o menor a los límites del rango concluyendo que el punto está sobre o debajo del polígono respectivamente, y en el caso que el valor sea igual a las rectas se concluye que el punto está en la periferia del polígono.

Los estados del punto son representados por valores numéricos.

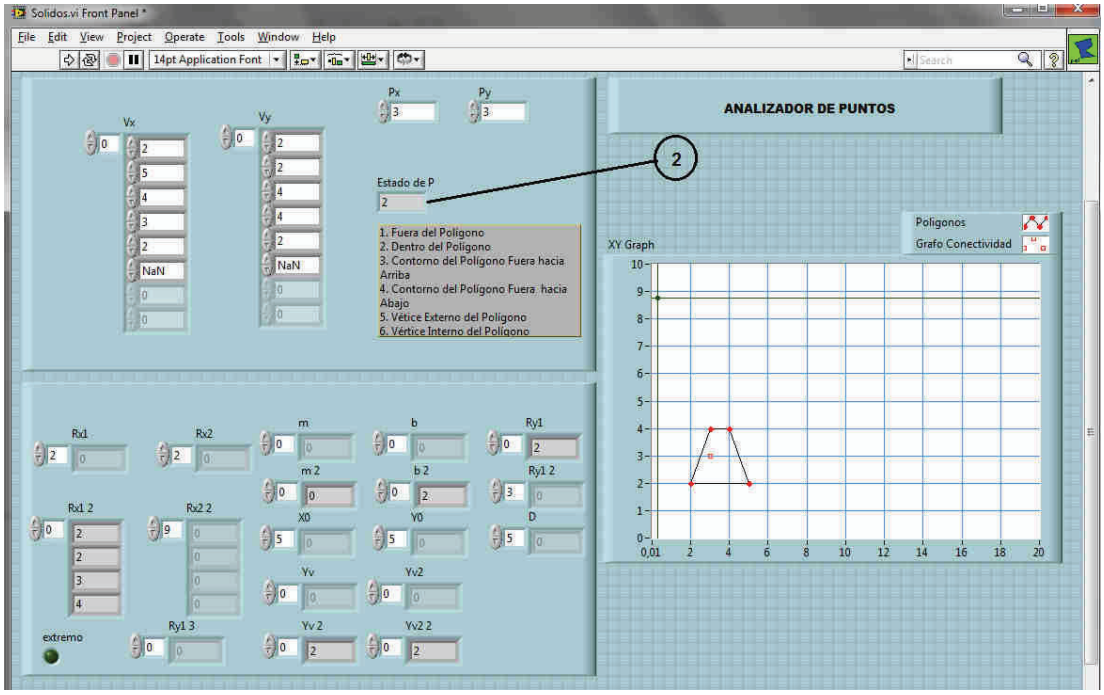


Figura 3.23 Panel frontal de la subrutina Sólidos

En el caso del ejemplo se concluyó que el punto está dentro del polígono indicando el número 2.

El panel frontal de la subrutina SÓLIDOS (Figura 3.23) presenta el ingreso de los vértices del polígono, las coordenadas del punto para evaluación, las variables auxiliares y el estado del punto respecto al polígono, también presenta un gráfico que ayuda en la visualización y permite confirmar el estado del punto.

Esta subrutina tiene la función de analizar si rectas generadas cortan los obstáculos, trazar rectas verticales, conocer el contorno de un polígono, etc.

3.3.2 SUBRUTINA ANALIZADOR DE RECTAS

El objetivo del programa es conocer si dos nodos forman o no un enlace entre ellos en el mapa de entorno, por medio de corte de rectas generadas por el enlace y los lados de los obstáculos.

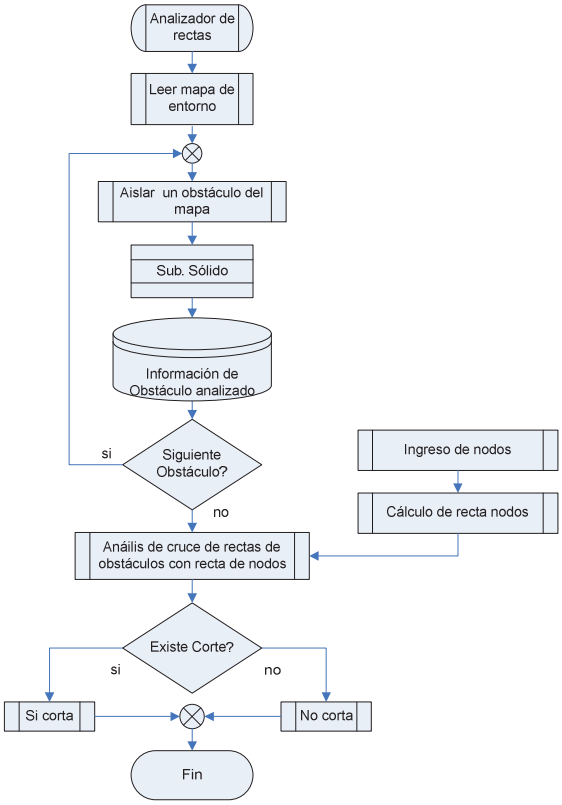


Figura 3.24 Diagrama de flujo de rutina Analizador de Rectas

De acuerdo con el diagrama de flujo de la Figura 3.24, la rutina toma la información del mapa de entorno y saca los vértices de cada uno de los obstáculos individualmente. La rutina SOLIDO realiza el análisis de cada obstáculo y determina las ecuaciones de las rectas con sus respectivos rangos de cada lado.

Cuando el programa termina de calcular todas las rectas, calcula una recta adicional correspondiente a los nodos ingresados por el método planificador.

Con la información de las ecuaciones de las rectas, se calcula el punto de corte de los nodos con todas las demás rectas que formen el mapa de entorno, y por medio de los rangos de las rectas de los obstáculos se concluye si existe corte o no. Finalmente el programa envía el resultado para luego realizar el enlace entre los nodos (Figura 3.25).

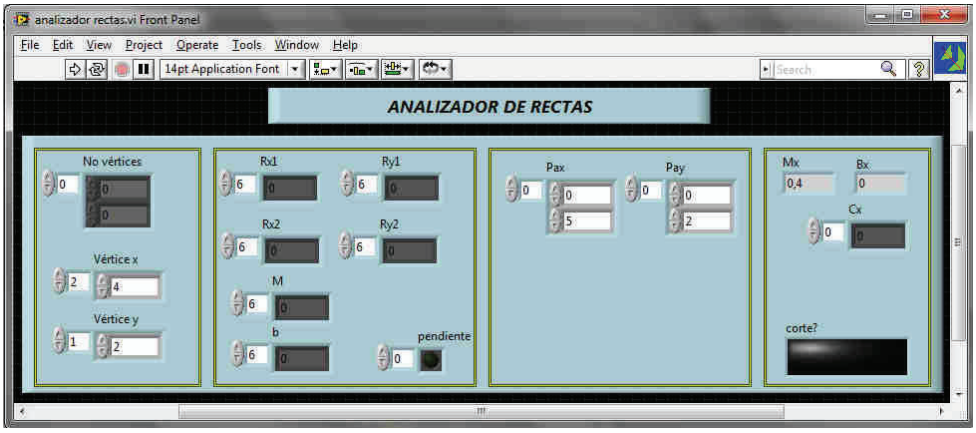


Figura 3.25 Panel frontal de subrutina Analizador de Rectas

3.3.3 REPRODUCCIÓN DEL GRAFO DE CONECTIVIDAD

De acuerdo con el diagrama de flujo de la rutina Descomposición de Celdas (Figura 3.17), el lazo exterior explica el proceso de ordenamiento de puntos y la construcción del grafo de conectividad.

Se toma el primer nodo ordenado previamente, (en el caso de Figura 3.20 el nodo superior izquierdo del plano), se traza una recta con el siguiente nodo ordenado. Para que el programa identifique que este nodo sea adyacente e incorpore como solución válida para la generación del grafo de conectividad, en primer lugar no debe existir corte de esta recta con ningún obstáculo del mapa de entorno, y en segundo lugar que sea el más próximo al primer nodo que conforme la recta.

Con la subrutina ANALIZADOR DE RECTAS, se determina si los dos nodos seleccionados tienen línea de vista. En el caso que exista corte de la recta con algún obstáculo el programa descarta los dos nodos y toma el siguiente y repite el proceso hasta encontrar una recta que no corte con ningún obstáculo.

Cuando encuentra un par de nodos válidos, estos son incorporados al grafo y se dibuja en el HMI el enlace entre ellos, luego la rutina toma el siguiente nodo y repite el proceso. Para evitar posibles repeticiones de selección de un par de nodos válidos el programa descarta el nodo que ya ha sido evaluado y que haya tenido una respuesta válida.

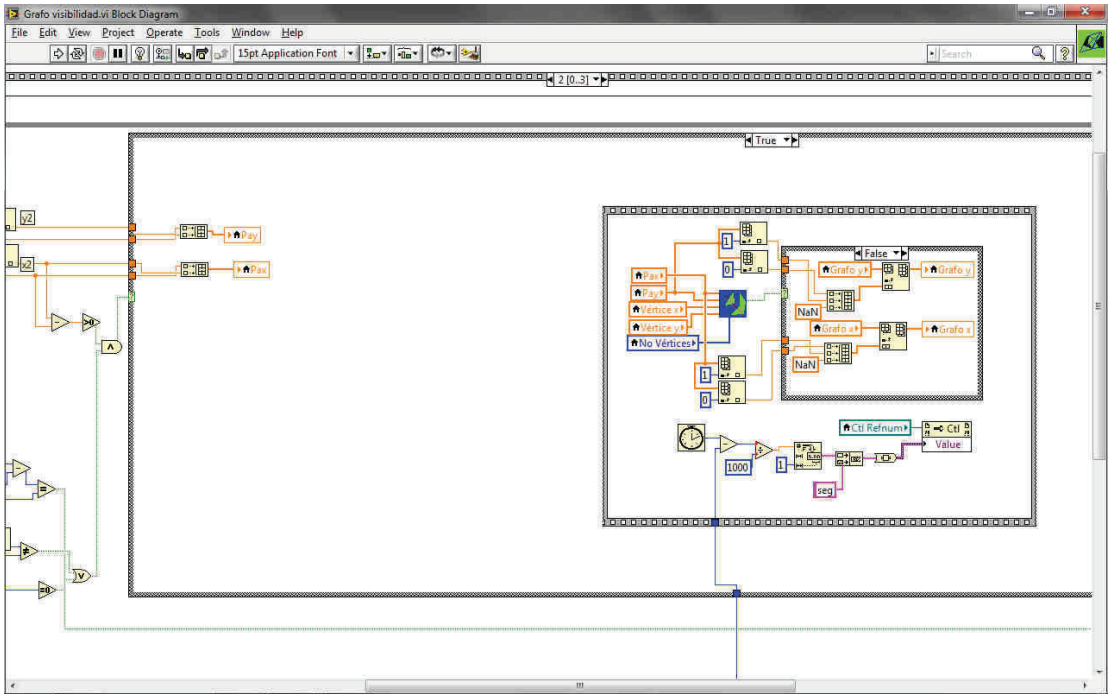


Figura 3.26 Diagrama de bloques para reproducción de grafo de conectividad

El diagrama de bloques de la Figura 3.26 traduce la obtención del grafo de conectividad del diagrama flujo (Figura 3.18) en el LabVIEW.

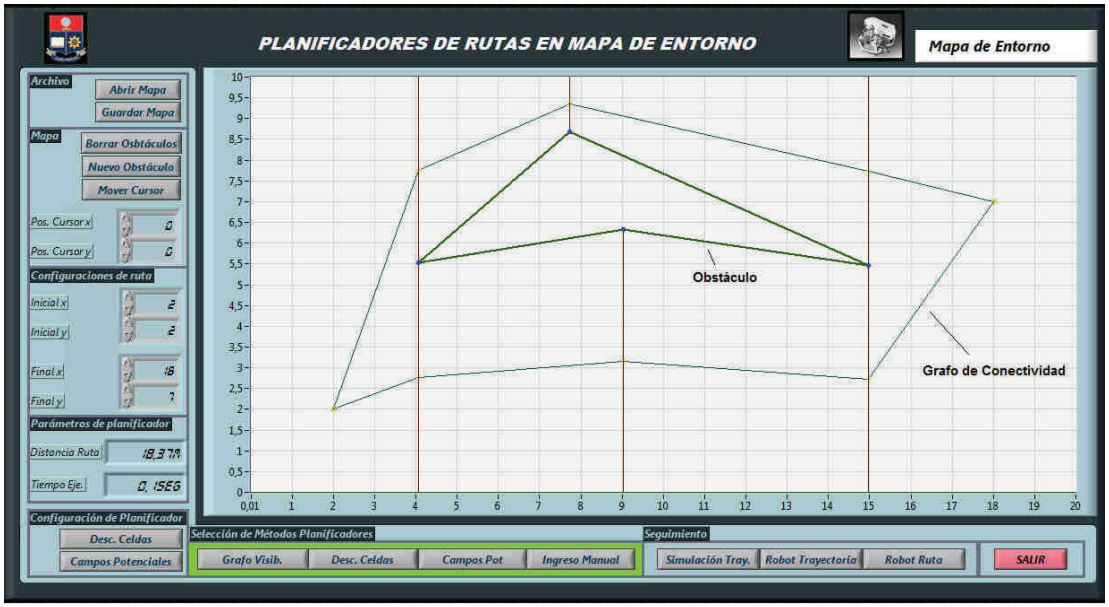


Figura 3.27 Reproducción del Grafo de Conectividad

Evaluados todos los nodos, el grafo se completa, y se procede al cálculo de la ruta entre las configuraciones inicial y final.

3.3.4 DESCOMPOSICIÓN DE CELDAS MODIFICADO

El método de Descomposición de Celdas tiene la particularidad de formar enlaces son los nodos adyacentes para formar el grafo (Figura 3.27), teniendo un limitaciones para formar rutas más simples y con menos quiebres, para lo cual se plantea una rutina que genere sub grafos de conectividad entre celdas que no necesariamente sean vecinas pero que simplifique la ruta final formada.

La construcción de estos enlaces se basa en la inclusión de posibles nuevos enlaces formados por los nodos obtenidos por el método planificador que no sean vecinos pero que forme un enlace libre de obstáculos, para lo cual se plantea un control para la formación de posibles enlaces formado sub grafos de conectividad por línea de vista, pero evitando la formación de un grafo de conectividad completo por línea de vista como el método de Grafo de Visibilidad.

Para realizar este control se utiliza una subrutina que permita escoger entre el método original y el modificado para lo cual se utiliza un una ventana que tenga estas funciones (Figura 3.28).

Figura 3.28 Panel frontal generador de enlaces

La rutina que forma los grafos de conectividad modificado forma parte del mismo programa del método planificador (Figura 3.19), y el funciona bajo el siguiente diagrama de flujo.

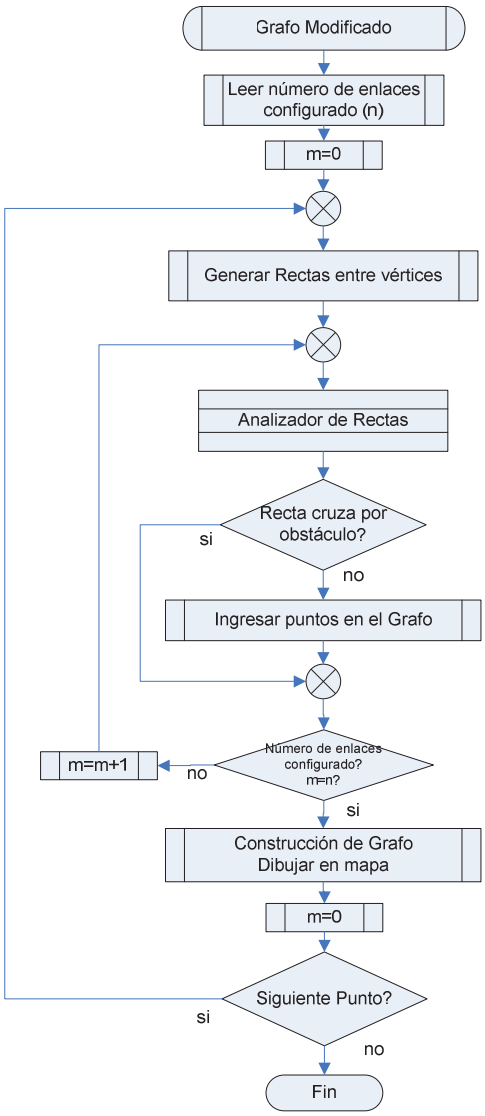


Figura 3.29 Diagrama de flujo para formación de subgrafos

El programa toma los nodos formados previamente por el método planificador (Figura 3.17), en el caso que se seleccione la opción del la variante del método planificado el programa toma el valor del número de enlaces el panel (Figura 2.28), y luego toma el primer nodo formado y evalúa si existe enlace o no con el siguiente nodo con la rutina ANALIZADOR DE RECTAS, con la ayuda de un contador el programa controla que solo se evalúen los nodos ingresados y se

reinicia cuando termina con nodos vecinos deseados. Terminado el primer nodo toma el siguiente y repite el proceso descartando los enlaces con los nodos ya evaluado para no obtener enlaces repetidos.

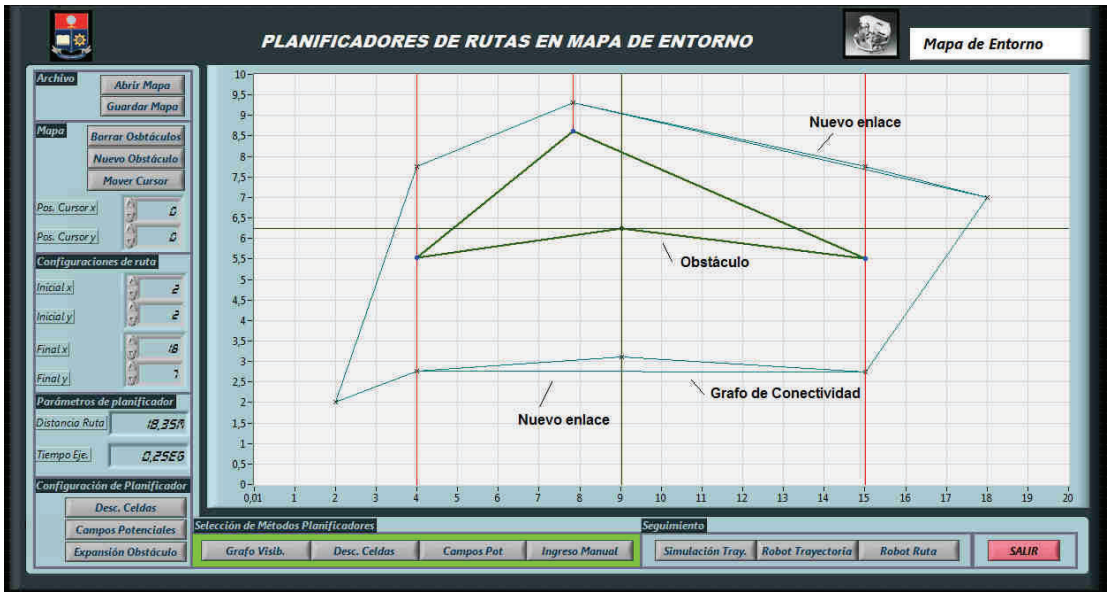


Figura 3.30 Formación de grafo de conectividad

El grafo de conectividad formado por el algoritmo implementado incluye nuevos enlaces con los nodos adyacentes que tiene línea de vista, el control del número de enlaces adicionales se ingresa en el panel frontal de la rutina.

Estos nuevos enlaces formados DE LA Figura 3.30 dan más opciones para los algoritmos de búsqueda de rutas y obtener rutas más simples y cortas.

3.3.5 GENERADOR DE LA RUTA MÁS CORTA

Este programa calcula la ruta más corta entre la configuración inicial y final P_0 y la final P_f a partir del grafo de conectividad.

Se compone de dos rutinas:

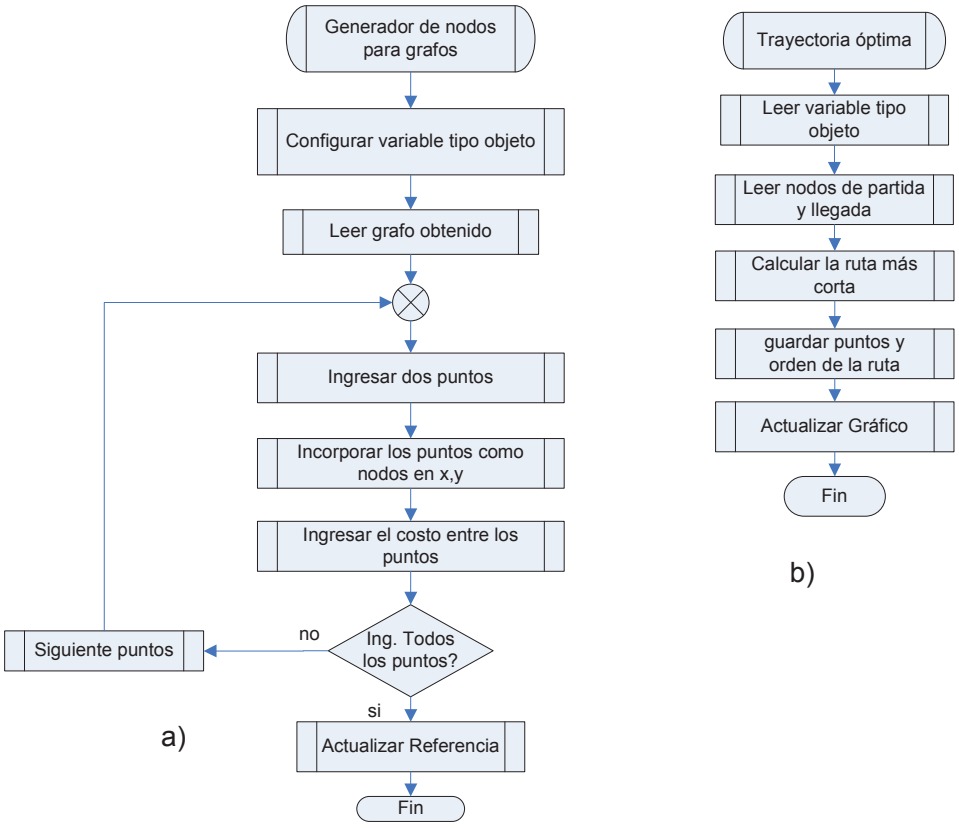


Figura 3.31 Diagrama de flujo de programa generador de ruta

Los diagramas de flujo de la Figura 3.31 presentan dos rutinas para la obtención de ruta más corta, la primera trata del ingreso del grafo de conectividad en un registro del LabVIEW llamado MAP REFERENCE, y el segundo es el cálculo de la ruta con este registro utilizando el algoritmo A*.

De acuerdo con el diagrama de flujo (Figura 3.31 a), primero se toma y configura una variable tipo objeto para manejar los bloques de la librería de planeación de rutas del LabVIEW (Figura 3.32).

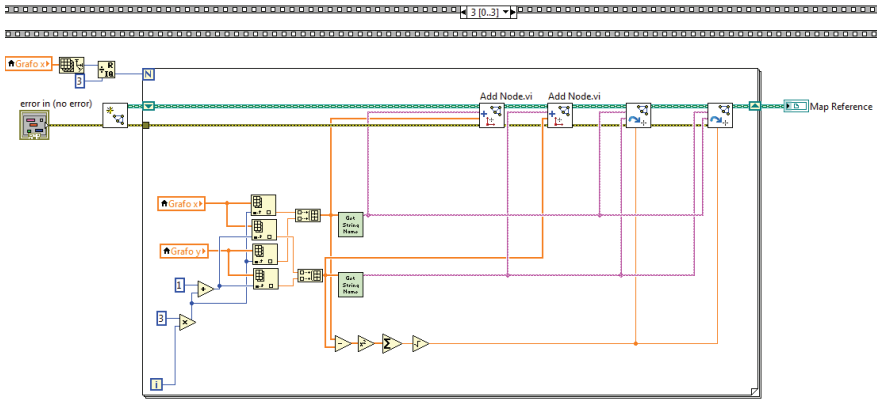


Figura 3.32 Diagrama de bloques para ingreso de grafo de conectividad

Para ingresar dos nodos del grafo de conectividad al registro MAP REFERENCE, hay que incorporar los enlaces del método planificador con su respectivo costo.

El costo es un número que representa el valor de la distancia entre los dos nodos ingresados, este valor puede ser cualquiera pero para este caso de estudio el costo es la distancia entre los nodos que forman el enlace del grafo de conectividad.

El lazo FOR de programa hace que el programa siga ejecutándose hasta que ingrese todos los nodos del grafo de conectividad formado por el método planificador.

El segundo diagrama de flujo de la Figura 3.31 calcula la ruta más corta entre las configuraciones inicial y final.

Este programa forma parte del HMI, para utilizarlo en las dos rutinas de planificación de rutas.

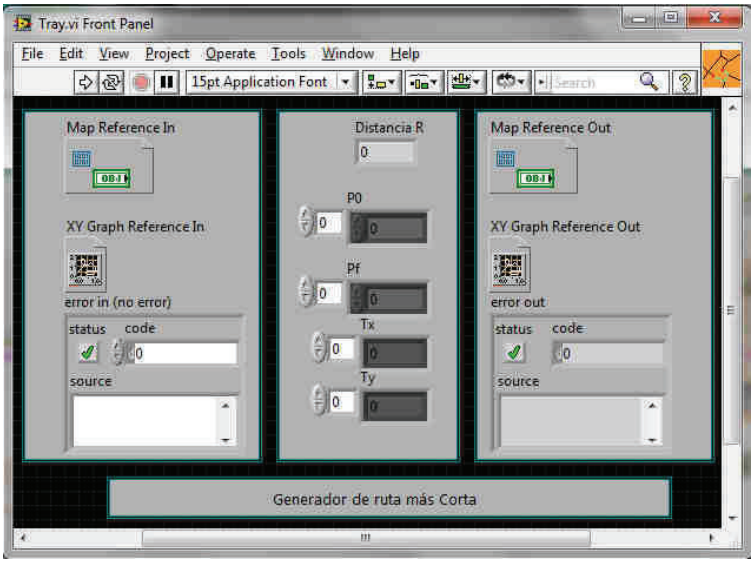


Figura 3.33 Panel frontal del programa buscador de rutas

En el panel frontal de la Figura 3.33 se muestra las variables de ingreso y de procesamiento. Para calcular la ruta se toma los valores de las configuraciones, y con la ayuda del bloque de adquisición de nodos se agrega los nodos de inicio y final de trayectoria (Figura 3.34).

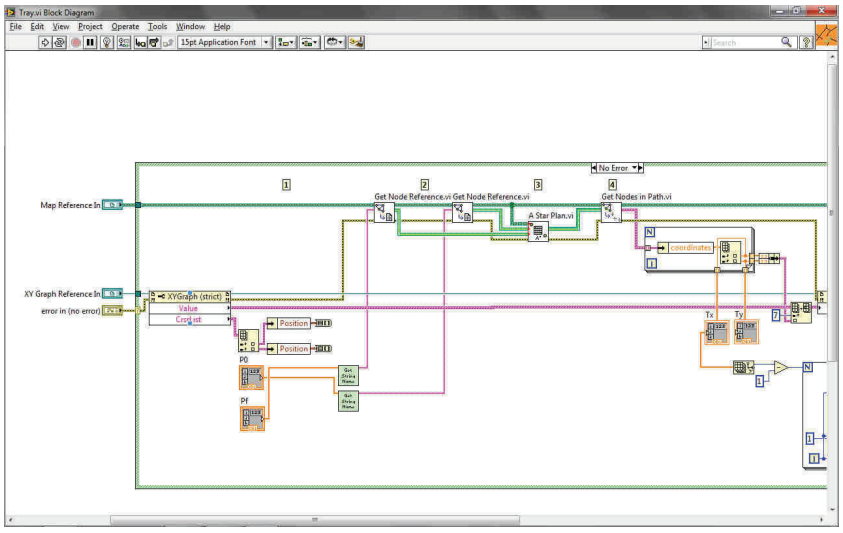


Figura 3.34 Diagrama de bloques para generar la ruta más corta

El programa evalúa el costo de todos los enlaces y la secuencia del grafo para determinar los pares de nodos que incluyen la ruta más corta entre las configuraciones inicial y final. Con estos nodos el programa calcula la distancia de la ruta.

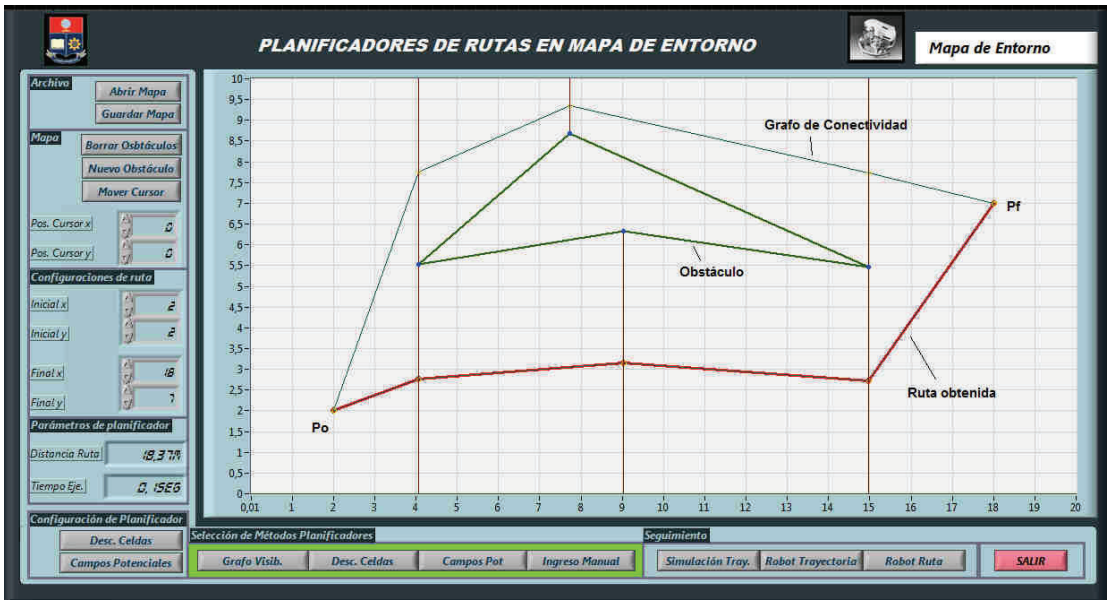


Figura 3.35 Obtención de la ruta entre las dos configuraciones

Con el cálculo de la ruta más corta, el HMI actualiza la gráfica, los indicadores de distancia, tiempo de ejecución y presenta en la Figura 3.35 la ruta entre las dos configuraciones terminando la ejecución del método de planificación de rutas.

3.4 PROGRAMA GRAFO DE VISIBILIDAD

El algoritmo Grafo de Visibilidad se basa en la construcción de un grafo de conectividad, igual que el método Descomposición de Celdas, pero con la diferencia que los nodos del grafo son los vértices de los obstáculos que tienen línea de vista con todos los demás.

Debido a la similitud con el programa de Descomposición de Celdas descrito en el Subcapítulo 3.3, el diagrama de flujo es prácticamente idéntico, a excepción de la concepción de los nodos y el entrelazamiento de ellos (Figura 3.36).

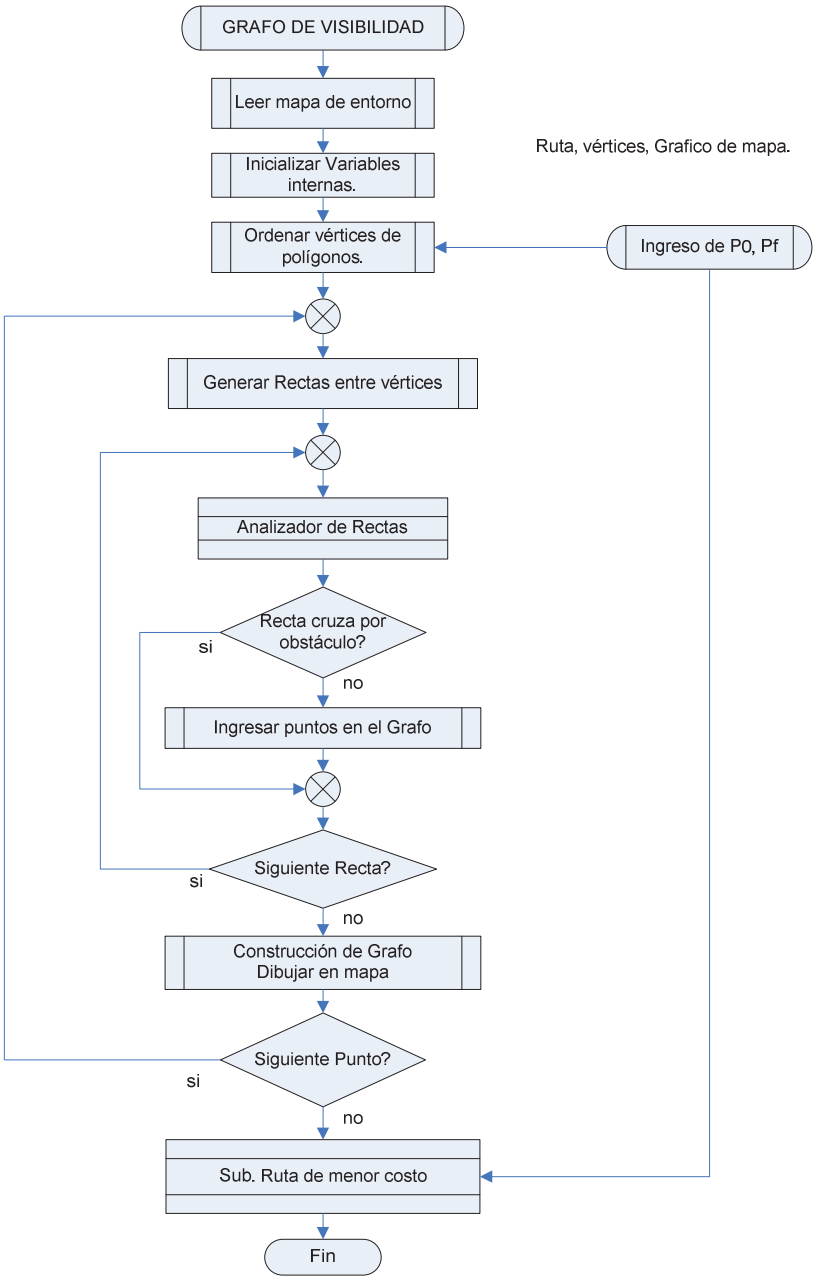


Figura 3.36 Diagrama de bloques del programa Grafo de Visibilidad

LA rutina del método Grafo de Visibilidad de la Figura 3.37 requiere la información de los vértices, la secuencia de ellos y las configuraciones inicial P_0 y final P_f, de igual forma que el método de Descomposición de Celdas para su ejecución.

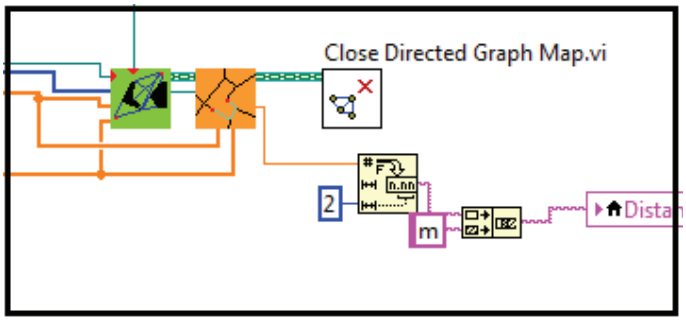


Figura 3.37 Bloque del programa Grafo de Visibilidad

El programa comienza ordenando los vértices de los polígonos individualmente de izquierda a derecha para generar el grafo en esa dirección, al mismo tiempo la configuración inicial P_0 y final P_f se incluyen formando parte de los registros de los nodos del mapa de entorno.

Figura 3.38 Panel frontal de la programa Grafo de Visibilidad

El panel frontal de la Figura 3.38 presenta los registros de las variables utilizadas para generación del grafo. Los registros (Vértice x, Vértice y), tienen los valores de los nodos del grafo que son simplemente los vértices de los polígonos del mapa de entorno y de las configuraciones.

3.4.1 REPRODUCCIÓN DEL GRAFO DE CONECTIVIDAD

La reproducción del grafo de conectividad se explica con el siguiente ejemplo:

Figura 3.39 Mapa de entorno dibujado

La Figura 3.39 presenta un mapa de entorno formado por dos polígonos que tienen los siguientes vértices:

$P1$:

$V_1 = (6.7\,;2)$

$V_2 = (10.6\,;0{,}6)$

$V_3 = (8.5\,;2{,}8)$

$P2$:

$V_1 = (10.2\,;8.7)$

$V_2 = (6.2\,;3.9)$

$V_3 = (13.1\,;3.8)$

$V_4 = (9.3\,;5.14)$

El siguiente paso del diagrama de flujo es ordenar los vértices, las configuraciones en y guardar en los registros (vértice x, vértice y) formando los nodos del grafo:

Tabla 3.1 Nodos obtenidos en mapa de entorno

Nodo	p_0	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8
x	1	6.2	6.7	8.5	9.3	10.2	10.6	13.1	17
y	7	3.9	2	2.8	5.14	8.7	0.6	3.8	5

Se toma el primer nodo de la rutina, en este caso es el punto $(1\,,7)$, y se trazan las rectas con el resto de nodos de la tabla.

Tabla 3.2 Selección de nodos para cálculos de rectas

1	7	6.2	3.9
		6.7	2
		8.5	2.8
		9.3	5.14
		10.2	8.7
		10.6	0.6
		13.1	3.8
		17	5

Las rectas calculadas, son ingresadas a la subrutina ANALIZADOR DE RECTAS, el cual evalúa la existencia de obstáculos para formar o no el enlace entre nodos para posteriormente integrarlos al grafo de conectividad (Figura 3.40). Si la recta no cruza por ningún obstáculo el programa dibuja la recta válida en el HMI caso contrario no la dibuja y continúa con el siguiente nodo.

Otra consideración es enlazar nodos que comparten un lado de un obstáculo e incluirlos en el grafo de conectividad ya que es una solución válida del método.

Cuando se evalúan todas las rectas con el primer nodo, el programa toma el siguiente y repite el proceso en este caso es el punto $(6.7, 2)$, para el caso del ejemplo toma los 6 nodos restantes calcula todas las rectas y las evalúa.

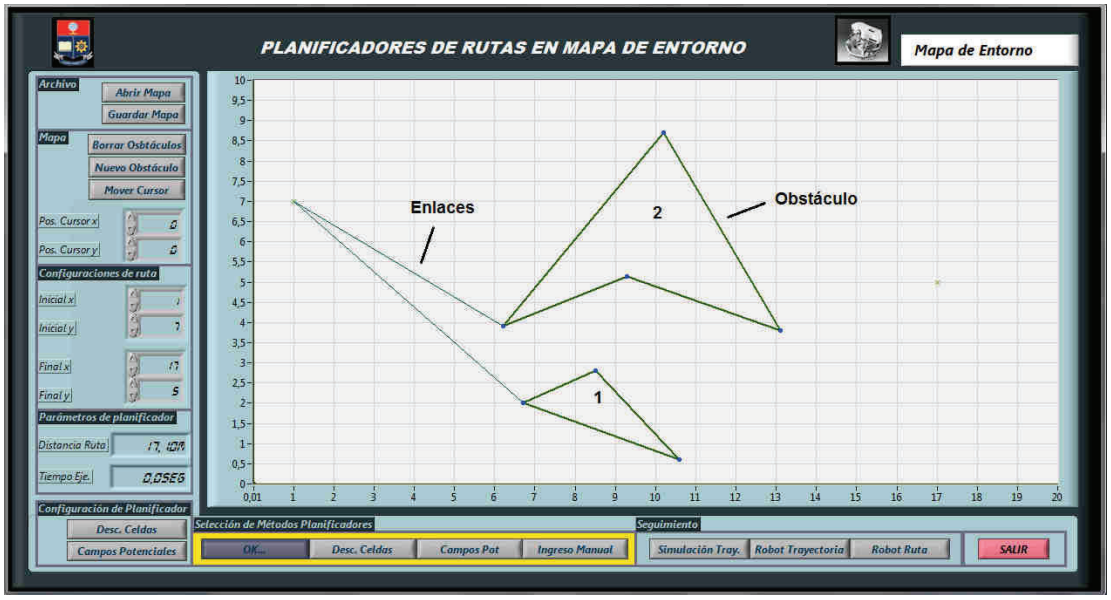


Figura 3.40 Rectas válidas para el nodo (1, 7)

La Figura 3.40 presenta las rectas válidas para el primer nodo construyendo el grafo.

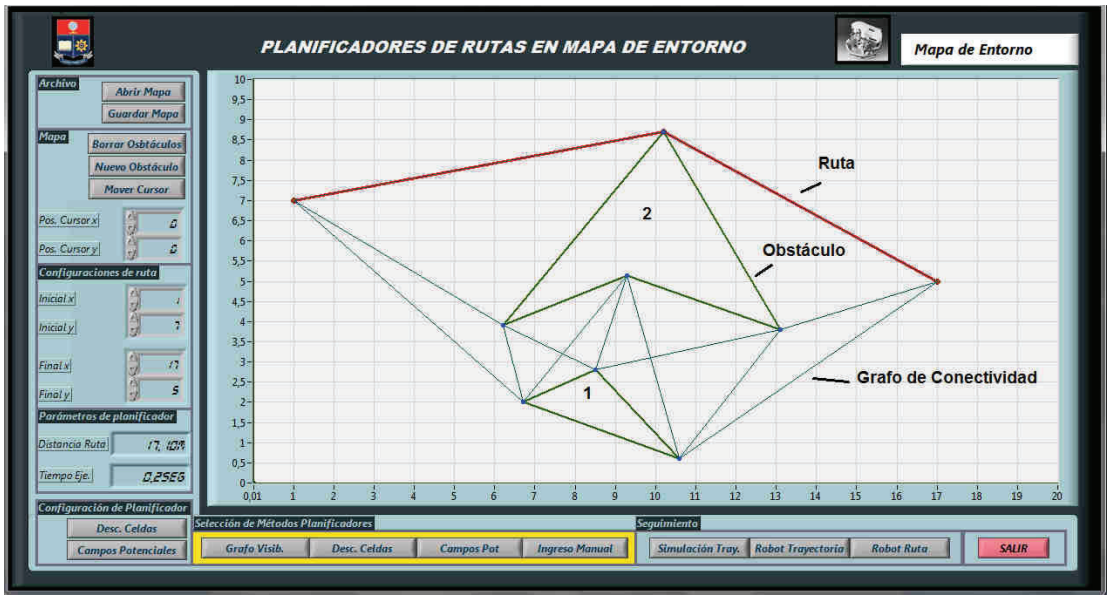


Figura 3.41 Obtención de la ruta entre las dos configuraciones

Una vez terminado el grafo (Figura 3.41), se llama la subrutina GENERADOR DE RUTA MÁS CORTA, y traza la ruta más corta entre las dos configuraciones, calcula la distancia y el tiempo de duración para el método.

Los puntos de la ruta se almacenan en el mismo gráfico, para su posterior uso en el programa Control de Trayectoria y ruta o guardar en un archivo.

3.5 PROGRAMA CAMPOS POTENCIALES

El algoritmo de Campos Potenciales es un método de planificación que tiene un enfoque diferente a los dos anteriores. Este método no se basa en la construcción de un grafo para determinar la ruta óptima, sino que el método traza directamente la ruta a partir de una fuerza virtual aplicada al robot atrayéndolo a la configuración final P_f.

Esta fuerza virtual se calcula a partir de la fuerza de atracción $\vec{F}_{atr}$ producida por la configuración final P_f, y por la fuerza generada por el campo de repulsión de los obstáculos del mapa de entorno en condiciones normales. Los mínimos locales es una limitación del método bajo ciertas condiciones impidiendo que el programa termine el cálculo de la ruta, para lo cual se más adelante en este capítulo se desarrolla el algoritmo para eliminar esta limitante.

El diagrama de flujo de la Figura 3.43 toma en cuenta las consideraciones del método descritas anteriormente, la información del mapa de entorno, y la compatibilidad con el HMI.

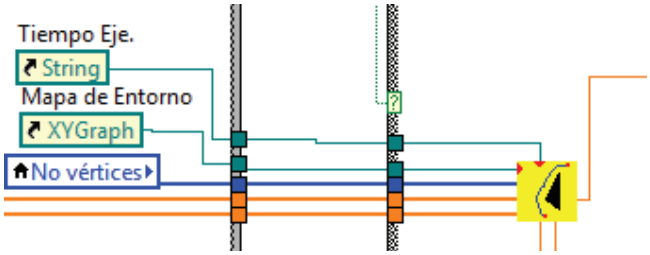


Figura 3.42 Bloque del programa Campos Potenciales

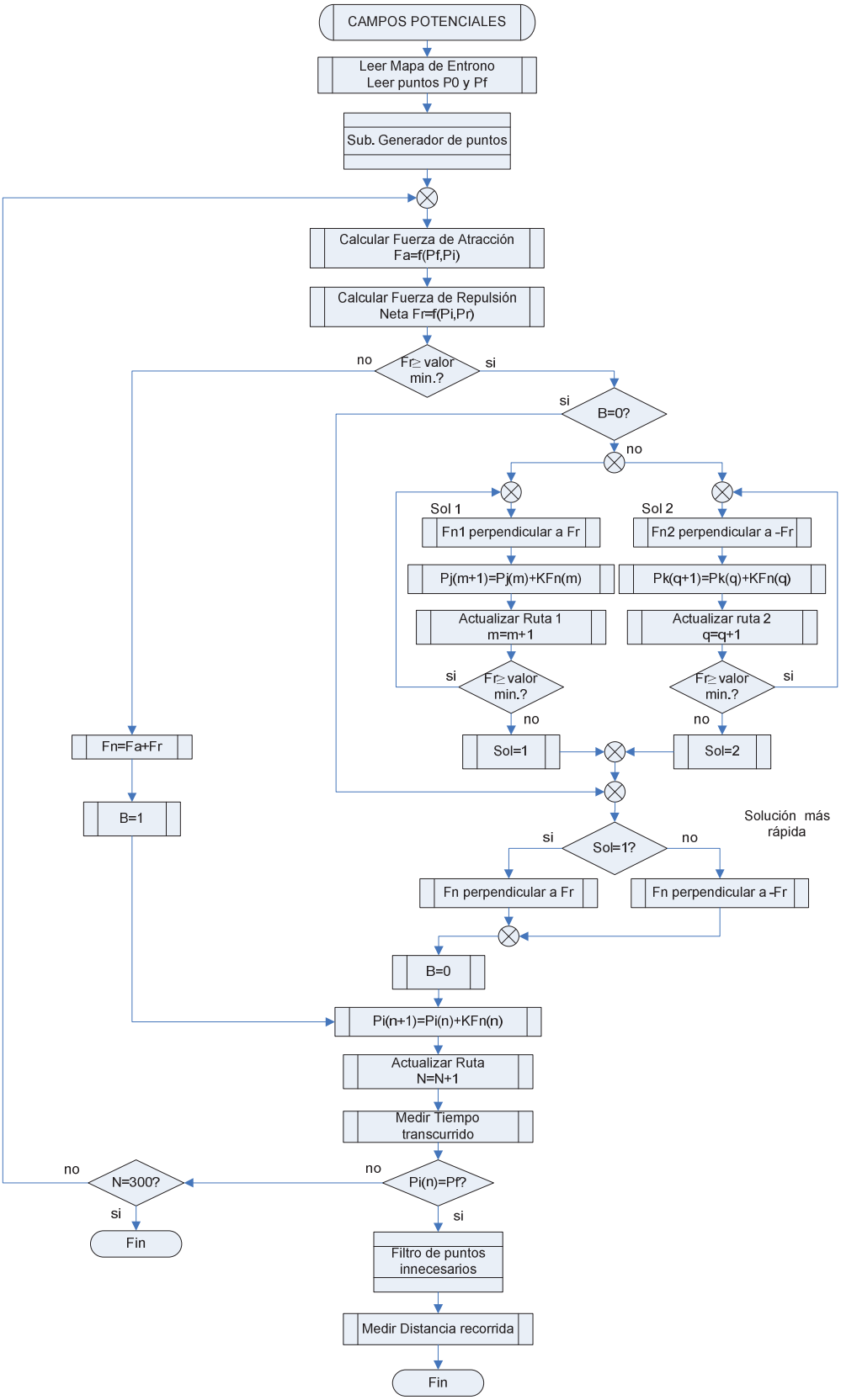


Figura 3.43 Diagrama de flujo de Campos Potenciales

El programa primero toma la información del mapa de entorno y de las configuraciones como de muestra en la Figura 3.42.

Figura 3.44 Panel frontal Programa Campos Potenciales

El panel frontal del algoritmo (Figura 3.44) no forma parte del HMI, solamente procesa las variables y envía el resultado de la ruta obtenida.

De la información obtenida del mapa de entorno y las configuraciones el siguiente paso es el dimensionamiento de la fuerza de atracción $\vec{F}_{atr}$ y la de repulsión $\vec{F}_{rep}$.

La fuerza de repulsión neta $\vec{F}_{rep}$ se genera a partir de la sumatoria de fuerzas de cada obstáculo pero la única información de ellos son los vértices y la secuencia, entonces es necesario transformar los polígonos en obstáculos sólidos para que exista el campo de repulsión.

3.5.1 SUBRUTINA GENERADORA DE SÓLIDOS

De acuerdo con la Teoría Electromagnética, el campo eléctrico que produce un sólido se genera a partir de la distribución de cargas puntuales (Figura 3.45), que compone la geometría del sólido [17].

$$\vec{F} = \sum_{i=0}^{i} k \frac{q_1 q_i}{(r_i)^2} \vec{u}_r \tag{3.4}$$

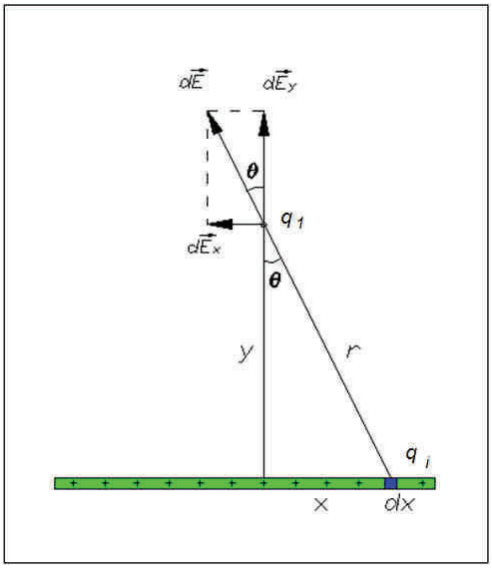


Figura 3.45 Distribución continua de carga

La rutina Generadora de Sólidos (Figura 3.47) se encarga de crear los puntos adicionales que simulen la distribución de carga y generar el campo eléctrico uniforme del obstáculo.

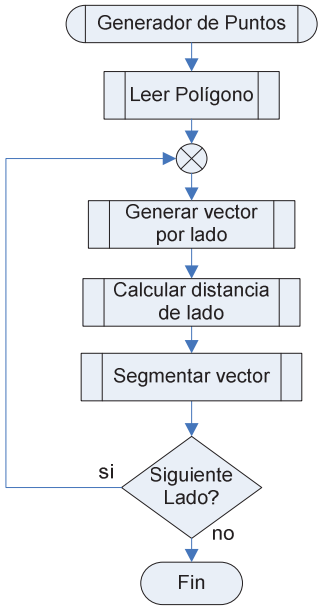


Figura 3.46 Diagrama de Flujo de rutina generadora de puntos

Los puntos se crean a partir de vectores trazados entre vértices consecutivos del polígono como se demuestra en el diagrama de flujo de la Figura 3.46, y la Figura 3.48.

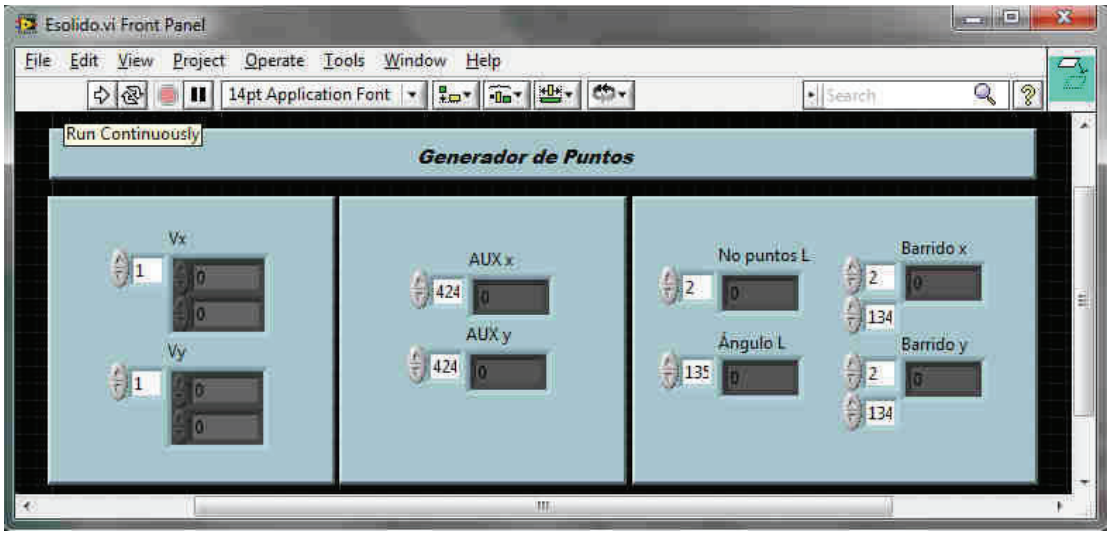


Figura 3.47 Subrutina generadora de puntos

Estos vectores se calculan restando vértices consecutivos del obstáculo utilizando la expresión:

$$\Delta\vec{v} = (P_{xb}, P_{yb}) - (P_{xa}, P_{ya}) \tag{3.5}$$

Donde:

- $\Delta\vec{v}$: Vector generado por aristas consecutivas de un polígono.
- $P_{xb},\ P_{yb}$: Coordenadas de la arista final.
- P_{xa}, P_{ya}: Coordenadas de la arista inicial.

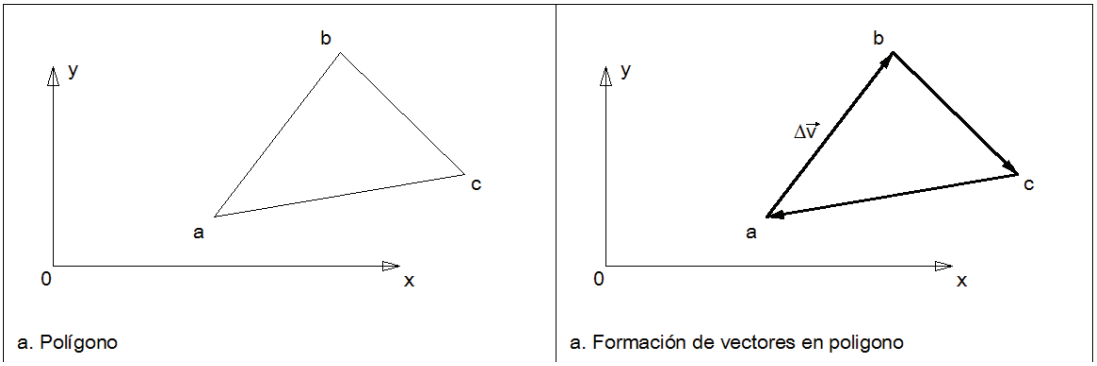


Figura 3.48 Generación de vectores.

Descomponiendo el vector $\Delta\vec{v}$ en módulo y unitario:

$$\Delta\vec{v} = |\Delta v|\vec{\mu}_{\Delta v} \tag{3.6}$$

Donde:

$$|\Delta v| = \sqrt{\Delta vx^2 + \Delta vy^2} \tag{3.7}$$

$$\vec{\mu}_{\Delta v} = \frac{\Delta\vec{v}}{\Delta v} \tag{3.8}$$

El módulo del vector es segmentado en partes iguales, y se construye nuevos vectores segmentados con la misma dirección del vector original.

$$n = \frac{\Delta v}{0.01} \tag{3.9}$$

$$\Delta\vec{w}_i = n_i \Delta v \vec{\mu}_{\Delta v} \tag{3.10}$$

Donde:

- n_i: Es un valor que varía entre 0 y n en pasos de $0.01m$
- $\Delta\vec{w}_i$: Vector segmentado a $0.01m$ del vector $\Delta\vec{v}$.

Luego se suman los vectores segmentados con el primer vértice y se obtienen los puntos.

$$\left(P_{xi}, P_{yi}\right) = \Delta\vec{\boldsymbol{w}}_i + \left(P_{xa}, P_{ya}\right) \tag{3.11}$$

El proceso se repite con todos los lados del polígono, y continúa para todos los obstáculos dibujados.

La Figura 3.49 presenta la ejecución de la subrutina para la obtención de un obstáculo sólido a partir de la información obtenida por el HMI.

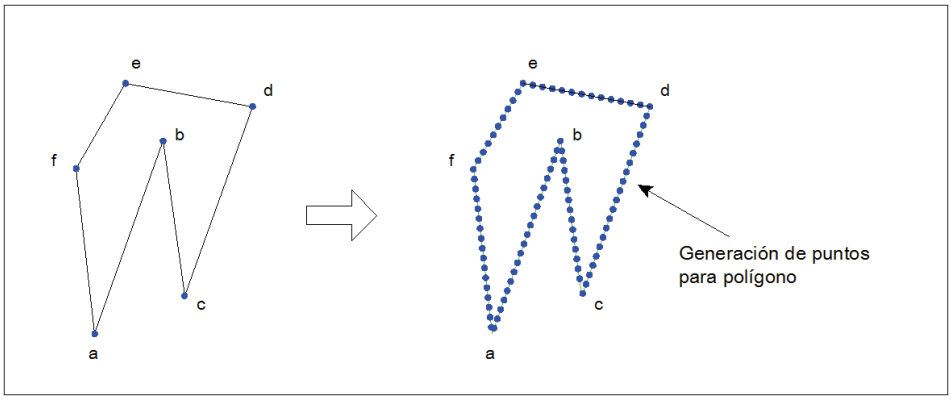


Figura 3.49 Generación de puntos para obstáculos

Continuado con el diagrama de flujo de Campos Potenciales (Figura 3.43) el siguiente punto es la generación de la fuerza de atracción aplicada al Robotino®.

3.5.2 SUBRUTINA FUERZA DE ATRACCIÓN

El método de campos potenciales se basa en la fórmula general:

$$\vec{\boldsymbol{F}}_N = \vec{\boldsymbol{F}}_{rep} + \vec{\boldsymbol{F}}_{atr} \tag{3.11}$$

La fuerza de atracción $\vec{\boldsymbol{F}}_{atr}$, es la fuerza que ejerce la configuración final P_f sobre la posición actual del robot para atraerlo. Esta fuerza es una función que depende de la distancia entre la posición final P_f y la posición instantánea del robot P_i.

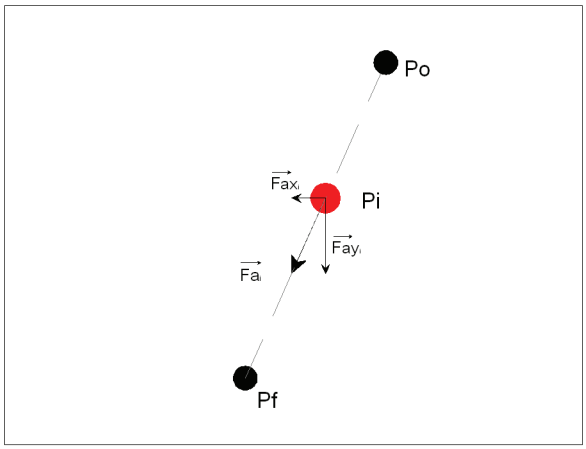


Figura 3.50 Fuerza de atracción

De la Figura 3.50, la fuerza de atracción se descompone en componentes (x, y).

$$\vec{\boldsymbol{F}}_{atr} = F_{atrx}\vec{\boldsymbol{\imath}} + F_{atry}\vec{\boldsymbol{\jmath}} \tag{3.12}$$

$$F_{atrx} = F_{atr}(\cos(\theta_{ra})) \tag{3.13}$$

$$F_{atry} = F_{atr}(\text{sen}(\theta_{ra})) \tag{3.14}$$

Donde:

$$\cos(\theta_{ra}) = \frac{x_{Pi} - x_{Pf}}{D_{PiPf}} \tag{3.15}$$

$$\text{sen}(\theta_{ra}) = \frac{y_{Pi} - y_{Pf}}{D_{PiPf}} \tag{3.16}$$

$$D_{PiPf} = \sqrt{\left(x_{Pi} - x_{Pf}\right)^2 + \left(y_{Pi} - y_{Pf}\right)^2} \tag{3.17}$$

D_{PiPf}: Distancia entre posición final y posición instantánea del robot.

De las expresiones anteriores se obtiene la dirección de la fuerza de atracción.

La función que describe la fuerza de atracción generalmente es directamente proporcional a la distancia entre la configuración final y la posición instantánea del robot ecuación (2.6). La Figura 3.51 presenta la curva de la fuerza de atracción de una función proporcional.

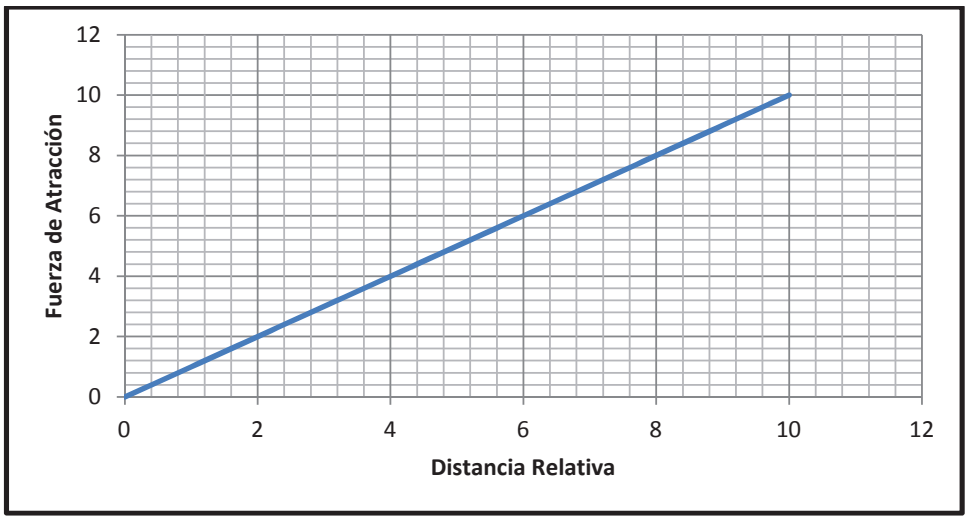


Figura 3.51 Fuerza de atracción proporcional

$$F_{atr} = KD_{PiPf} \tag{3.18}$$

La curva de la ecuación (3.18) indica que la fuerza se incrementa linealmente cuando más lejos está la posición instantánea de la configuración final P_f. Pero al tomar esta fuerza como un valor de desplazamiento, este es alto cuando el robot está lejano de la configuración final.

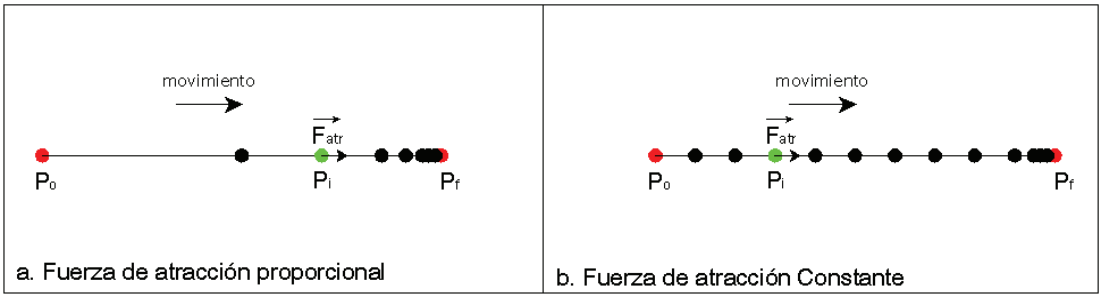


Figura 3.52 Desplazamiento a partir de acción de fuerzas de atracción

La mejor alternativa para mantener el desplazamiento estable del robot en todo su trayecto es generar una fuerza de atracción constante. La siguiente expresión representa a la fuerza de atracción que cumple con los requerimientos de la Figura 3.52 b.

$$F_{atr} = k(1 - e^{-wD_{PiPf}}) \tag{3.19}$$

Donde:

- k: Constante para desplazamiento

- w: Constante de amortiguamiento

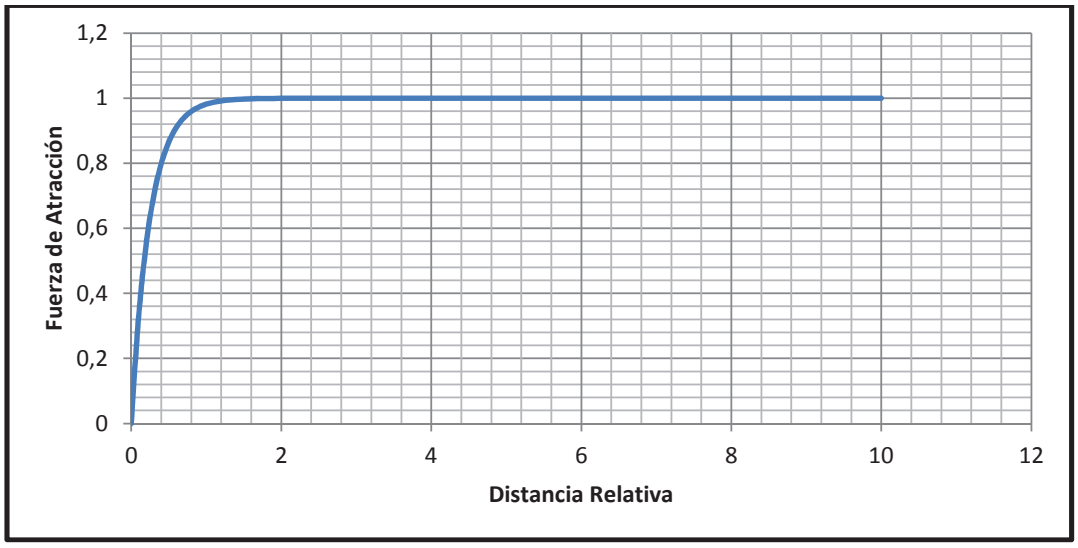


Figura 3.53 Fuerza de atracción

La ecuación 3.19 busca mantener la fuerza de atracción constante cuando la distancia entre la configuración final y la del robot es grande hasta que se encuentren cercanos entre sí, en este caso la fuerza disminuye rápidamente a 0, disminuyendo el error de posición cuando el robot llegue a la configuración final.

Los valores de la constante se determinan a partir de las dimensiones del plano del HMI, y son las siguientes:

$$x = 20m \tag{3.20}$$
$$y = 10m$$

Tomando en cuenta la relación entre el desplazamiento y la longitud del plano en el eje x un valor adecuado para la constante k es:

$$k = 0.15m \tag{3.21}$$

Este valor representa un desplazamiento del 0.75% respecto al plano del HMI obteniendo una buena resolución para el desplazamiento.

La constante de amortiguamiento no actúa salvo que la posición instantánea esté cerca de la configuración final, se calcula a partir de la siguiente consideración:

Sea la fuerza de atracción $(1 - e^{-1})$ cuando la distancia entre el robot y la configuración final es el desplazamiento máximo ($k = 0.15m$).

$$1 - e^{-1} = (1 - e^{-0.15w}) \tag{3.22}$$
$$w = 6.667$$

Estos valores mantienen la fuerza constante escalado el desplazamiento a 0.75% del plano cuando la distancia es mayor a $0.15m$ entre la configuración final y la posición del robot, y se reduce rápidamente cuando es menor, permitiendo un desplazamiento menor para llegar a la configuración final minimizando los errores.

$$F_{atr} = 0.15(1 - e^{-6.7D_{PiPf}}) \tag{3.23}$$

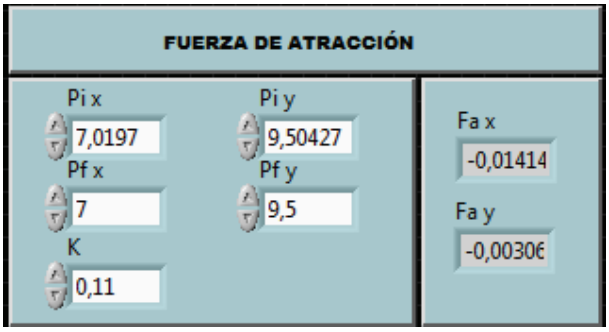


Figura 3.54 Subrutina de fuerza de atracción

El programa de la fuerza de atracción (Figura 3.54), ejecuta las expresiones 3.13, 14 y 23; el valor se guarda en los registros F_{ax} y F_{ay}

3.5.3 SUBRUTINA FUERZA DE REPULSIÓN NETA

La fuerza de repulsión neta $\vec{F}_{rep}$ se forma a partir de la sumatoria de fuerzas que generan los obstáculos, en este caso de los puntos obtenidos de la subrutina generadora de puntos.

$$\vec{F}_{rep} = \sum_{j=0}^{m} \vec{F}_{rj} \tag{3.24}$$

Donde:

- m: Es el número de puntos de todos los obstáculos del mapa de entorno.

Figura 3.55 Subrutina de fuerza neta de repulsión

La subrutina toma las coordenadas de los puntos del mapa de entorno y calcula la fuerza neta de estos en función de la posición instantánea del robot P_i.

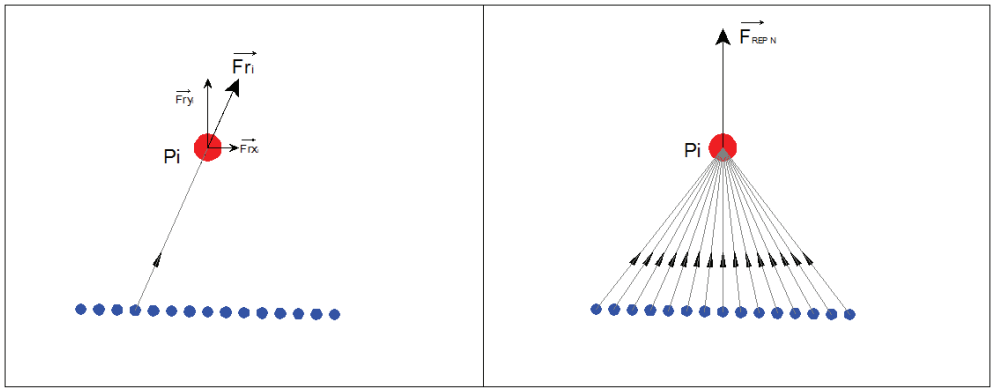


Figura 3.56 Acción de fuerza de repulsión neta

La Figura 3.56 presenta la fuerza de repulsión ejercida por un solo punto del obstáculo, y la fuerza neta ejercida por una recta formada por los puntos generados por el programa GENERADOR DE SÓLIDOS, que simula la distribución uniforme de fuerza produciendo un campo repulsivo virtual radial.

El cálculo de la descomposición de la fuerza en componentes utiliza las expresiones 3.13, 3.14, 3.15 y 3.16. Para el cálculo de la fuerza repulsiva de un solo punto, se utiliza la siguiente expresión:

$$\vec{\boldsymbol{F}}_{Rep} = -le^{-vD_{PiPj}} \cdot \vec{\boldsymbol{u}}r \tag{3.25}$$

Donde:

- l: Constante de proporcionalidad.

- v: Constante de amortiguamiento.

Esta expresión tiene la propiedad de disminuir rápidamente a cero cuando el robot se aleja del obstáculo, permitiendo que la acción de la fuerza de atracción sea la única fuerza que produce desplazamiento. Pero conforme se acerca al obstáculo se incrementa la fuerza de repulsión neta tal que modifica el módulo y la dirección de la fuerza neta aplicada al robot permitiéndole alejarse pero no de forma brusca estabilizando la generación de la ruta.

Figura 3.57 Fuerza de repulsión aplicada a un punto

La Figura 3.57 presenta las curvas de las fuerzas de repulsión cuando se aplica a un punto ubicado con una variación de 2% de la distancia más corta. Estas curvas demuestran la disminución significativa de la fuerza cuando se evalúa del punto vecino, más cercano de la posición del robot.

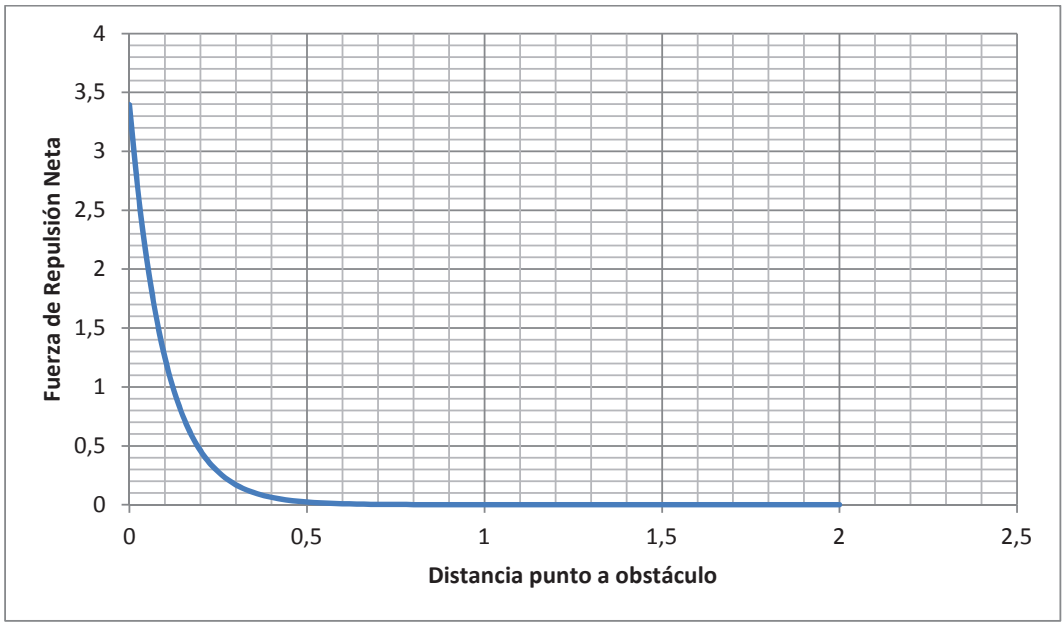


Figura 3.58 Fuerza de repulsión neta

Tomando en cuenta todos los puntos de la recta de la Figura 3.56, la fuerza de repulsión neta tiene una curva similar a la evaluada en un punto del obstáculo, y su valor máximo es aproximadamente 3.5 veces la fuerza máxima producida por el punto más cercano al robot (Figura 3.58).

Teniendo una fuerza máxima definida para una recta, y la disminución de la magnitud de la fuerza para los puntos vecinos al más cercano con la posición del robot se tienen las siguientes características:

- Estabiliza los cambios de posición del robot suavizando la ruta.

- La dirección de la fuerza es prácticamente perpendicular a los lados de los obstáculos.

- Se puede despreciar las dimensiones de los obstáculos y su forma.

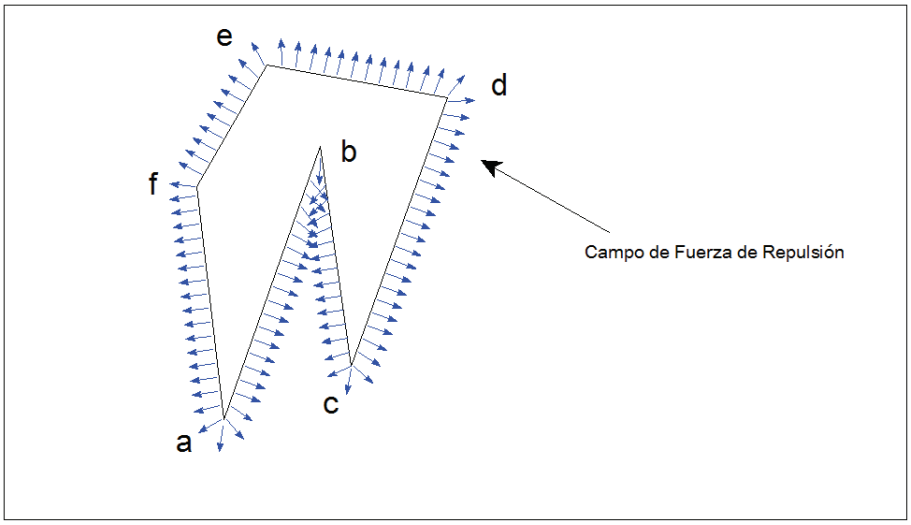


Figura 3.59 Formación de campo repulsivo virtual

Con estas características el programa se simplifica de tal forma que la rutina se limita a calcular la fuerza en un punto y repetir el progreso completando el cálculo

para todo el mapa de entorno, produciendo el efecto del campo de repulsión constante (Figura 3.59).

Para el cálculo de la constante de amortiguamiento v se toma el mismo valor que el de la constante w, y el valor de l se toma de la ecuación de la fuerza (3.25).

Considerando una distancia mínima de acercamiento frontal entre el obstáculo y el robot de $10cm$ generando un mínimo local ($\vec{\boldsymbol{F}}_{atr} = -\vec{\boldsymbol{F}}_{Rep}$) el valor de la constante l de la fuerza de repulsión neta es:

$$le^{-6.7x0,1} = 0.15(1 - e^{-6.7D_{PiPf}}) \tag{3.26}$$

Debido a la naturaleza de la fuerza de atracción se puede considerar que la componente $(1 - e^{-6.7D_{PiPf}}) = 1$ cuando el robot está lejos de la configuración final (Figura 3.50), entonces:

$$l = 0.29$$

Esté valor se divide para 3.5, para ajustar la expresión de la fuerza de repulsión a un solo punto, debido a que la expresión considerada es la fuerza neta.

$$l = \frac{0.29}{3.5} = 0.086 \tag{3.27}$$

La fuerza de repulsión individual es:

$$\vec{\boldsymbol{F}}_{Rep} = -0.086e^{-6.7D_{PiPj}} \cdot \vec{\boldsymbol{u}}r \tag{3.28}$$

Con los cálculos de las fuerza de atracción y repulsión se determina la fuerza neta aplicada sobre el robot con la fórmula general:

$$\vec{\boldsymbol{F}}_N = \vec{\boldsymbol{F}}_{rep} + \vec{\boldsymbol{F}}_{atr} \tag{3.11}$$

Tomando en cuenta las constantes de proporcionalidad de las fuerzas de atracción y repulsión neta, el desplazamiento del robot se calcula con la ecuación vectorial en diferencias:

$$\left(P_{x(n+1)}\vec{\boldsymbol{\imath}} + P_{y(n+1)}\vec{\boldsymbol{\jmath}}\right) = \left(P_{x(n)}\vec{\boldsymbol{\imath}} + P_{y(n)}\vec{\boldsymbol{\jmath}}\right) + \left(F_{Nx(n)}\vec{\boldsymbol{\imath}} + F_{Ny(n)}\vec{\boldsymbol{\jmath}}\right) \tag{3.29}$$

Donde la condición inicial es la configuración inicial P_0 del robot representada por:

$$\left(P_{x(n)}\vec{\boldsymbol{\imath}} + P_{y(n)}\vec{\boldsymbol{\jmath}}\right) = \left(P_{x(0)}\vec{\boldsymbol{\imath}} + P_{y(0)}\vec{\boldsymbol{\jmath}}\right) \tag{3.30}$$

El diagrama de flujo de la rutina de Campos Potenciales aplica la ecuación (2.29) y actualiza los valores de posición en el HMI generando la ruta.

3.5.4 ELIMINACIÓN DE MÍNIMOS LOCALES

Los mínimos locales se producen cuando la fuerza de atracción y la fuerza de repulsión neta se anulan entre si $\vec{\boldsymbol{F}}_N = 0$, donde la posición instantánea del robot es diferente de la configuración final P_f.

Esta condición se da cuando un obstáculo se encuentra entre la posición instantánea del Robotino® P_i y la configuración final atravesados por un lado del obstáculo que su orientación sea perpendicular al camino (Figura 3.60 a).

El diagrama de flujo del método de Campos Potenciales (Figura 3.43) tiene la rutina de eliminación de mínimos locales en el caso que la fuerza neta disminuya significativamente en magnitud.

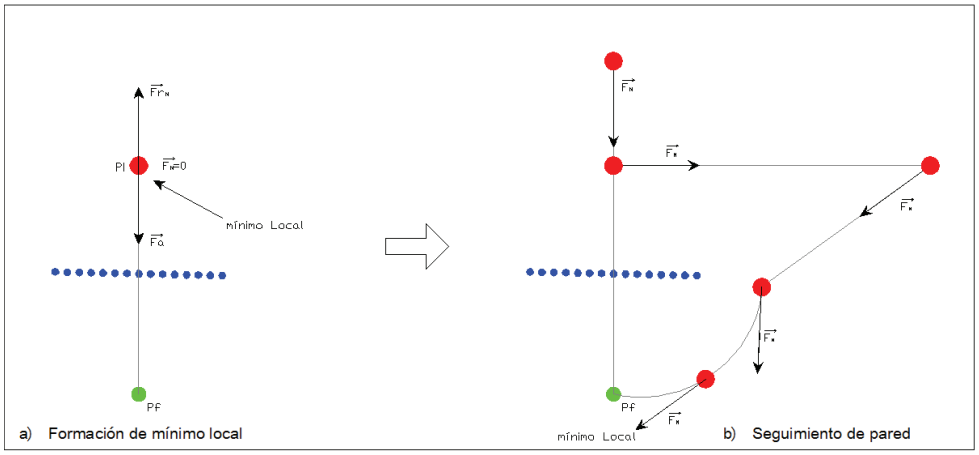


Figura 3.60 Eliminación de mínimos locales

Dado que la expresión de la fuerza de repulsión neta tiene la característica de perpendicularidad a la orientación de un obstáculo (Figura 3.56), para la Figura 3.60 a, el valor de la fuerza neta disminuye en sus componentes generando un mínimo local cuando se acerca a un obstáculo que intercepta el camino a la configuración final. Para eliminar el mínimo local se utiliza una técnica similar a la de seguimiento de paredes del obstáculo pero no incorporando un obstáculo virtual como en la técnica original, sino aprovechando la característica de perpendicularidad de la fuerza de repulsión redirigiéndola a lo largo del obstáculo.

Del diagrama de flujo del método planificador (Figura 3.43), cada iteración el programa primero evalúa el módulo de la fuerza de repulsión y compara su valor actual con el 15% de la fuerza de atracción instantánea. Con esta comparación se conoce si el robot se acerca a un obstáculo.

Cuando la fuerza de atracción es mucho mayor que la de repulsión neta el programa realiza el desplazamiento del punto instantáneo P_i, utilizando la ecuación (3.29) aplicando previamente la fórmula general de la fuerza neta (3.11).

En el caso de que la fuerza neta se acerque a 0, el programa lo detecta comparando la fuerza de repulsión con el valor mínimo, e inmediatamente desprecia las fuerzas de atracción y repulsión en el punto evaluado confirmando que la posición instantánea está próxima a un obstáculo.

En ese instante el programa genera una nueva fuerza $\vec{F}_{Cm}$, que aprovecha la perpendicularidad de la fuerza de repulsión neta con el obstáculo y el valor de la misma, se traza esta perpendicularmente a la fuerza de repulsión.

$$\vec{F}_{Cm} = (\mp F_{Ny(n)}\vec{\imath} \pm F_{Nx(n)}\vec{\jmath}) \tag{3.31}$$

$$\left(P_{x(n+1)}\vec{\imath} + P_{y(n+1)}\vec{\jmath}\right) = \left(P_{x(n)}\vec{\imath} + P_{y(n)}\vec{\jmath}\right) + \left(-F_{Ny(n)}\vec{\imath} + F_{Nx(n)}\vec{\jmath}\right) \tag{3.32}$$

$$\left(P_{x(n+1)}\vec{\imath} + P_{y(n+1)}\vec{\jmath}\right) = \left(P_{x(n)}\vec{\imath} + P_{y(n)}\vec{\jmath}\right) + \left(F_{Ny(n)}\vec{\imath} - F_{Nx(n)}\vec{\jmath}\right) \tag{3.33}$$

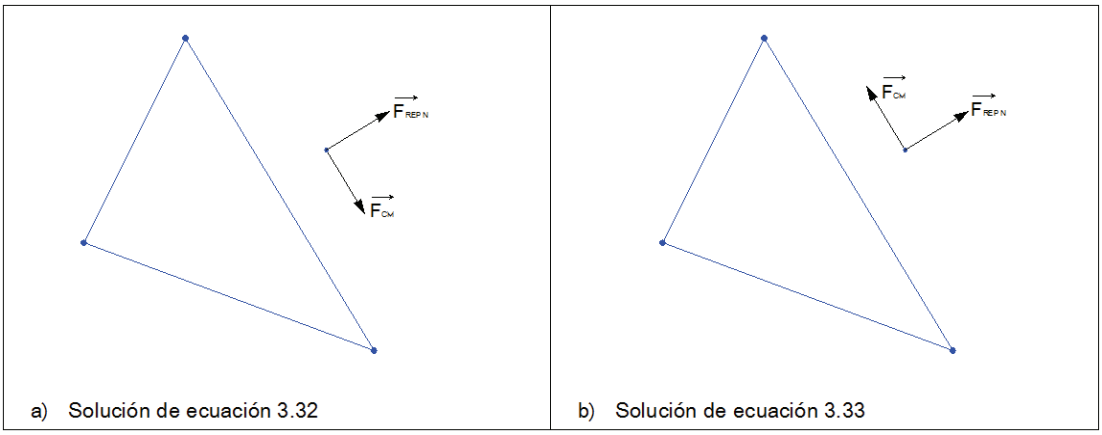


Figura 3.61 Dimensionamiento de Fuerza compensadora

La fuerza compensadora tiene dos soluciones para evadir un obstáculo según la ecuación 3.31, lo que implica que existen dos rutas para evadir un obstáculo.

De acuerdo al diagrama de flujo del método de campos potenciales (Figura 3.43), el programa ejecuta las dos soluciones de evasión pero no forman parte de la solución definitiva, son ejecutadas localmente.

El objetivo de estas dos formas de cálculo es encontrar la solución más rápida para evitar el obstáculo. Una vez encontrada, el programa selecciona la solución más rápida y aplica a la configuración instantánea evadiendo el obstáculo, y

continúa normalmente aplicando la fórmula general de la fuerza neta hasta que se encuentre con otro obstáculo.

Puede existir ocasiones que el programa no obtenga las dos posibles rutas debido a que pueden haber obstáculos que tengan formas complejas o cerradas que ocasionan que el programa no encuentre solución. Por esta razón el programa cuenta con un limitador de ejecución a un número determinado de veces (300 ejecuciones) y evita que el programa ejecute una ruta en la que no exista una solución válida.

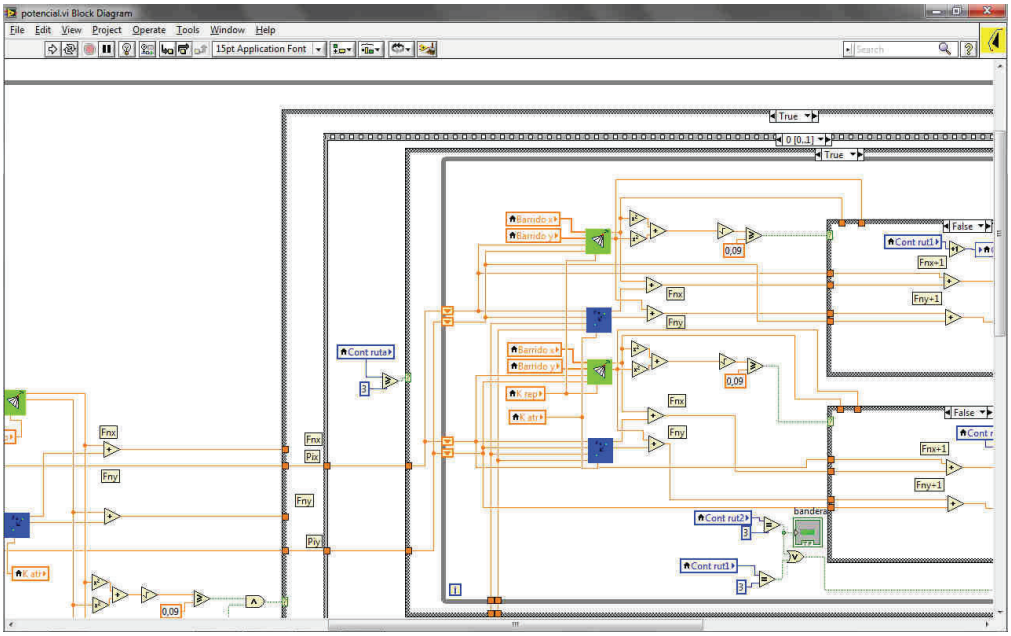


Figura 3.62 Diagrama de bloques de generación de ruta

Determinadas las rutas el programa dibuja en el HMI un punto por iteración.

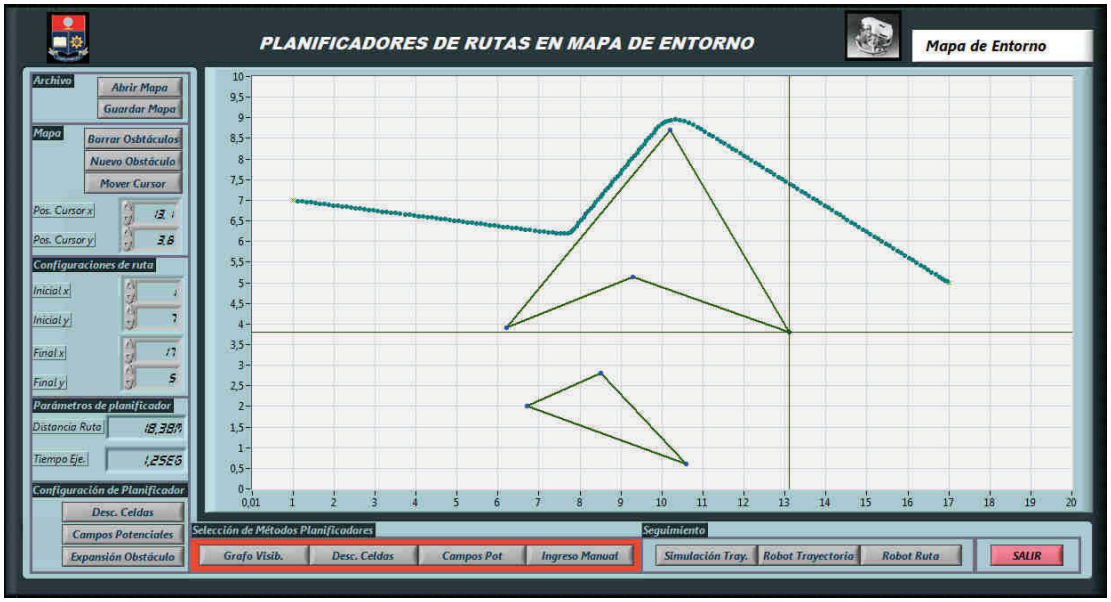


Figura 3.63 Generación de rutas por Campos Potenciales

En la Figura 3.63 se aprecia los cambios de dirección de las posiciones producidos por la fuerza de repulsión y la de compensación por los obstáculos que atraviesan entre la configuración inicial y final.

Los puntos que pasan evadiendo los obstáculos son aquellos en que la fuerza aplicada al robot es la fuerza de compensación y esta es paralela al obstáculo manteniendo la distancia constante sin importar las dimensiones del obstáculo.

3.5.5 FILTRADO DE RUTA

Tanto en el método de Descomposición de Celdas Exactas como en el de Grafo de Visibilidad, la ruta obtenida tiene un número muy bajo de puntos de conexión para completarla, por lo que se realiza un filtrado de la ruta para el método de Campos Potenciales para disminuir la cantidad de puntos que no proveen de información necesaria para el camino obtenido, consiguiendo de esta manera simplificar el programa de control del Robotino® de Festo a un solo algoritmo que sea aplicable para los tres métodos planificadores.

De acuerdo con el diagrama de flujo de la Figura 3.43, la ruta tiene máximo 300 puntos coordenados, ciertamente es un valor alto de coordenadas y la gran mayoría forman líneas rectas (Figura 3.63), haciendo que estos puntos no aporten significativamente en la formación de la ruta. Los puntos se forman cuando el robot se mueve bajo la acción de la fuerza neta o cuando está bajo la fuerza de repulsión produciendo el movimiento paralelo a un lado de un obstáculo.

Para eliminar los puntos que no son relevantes de la ruta generada por el método planificador se tiene el siguiente diagrama de flujo (Figura 3.64):

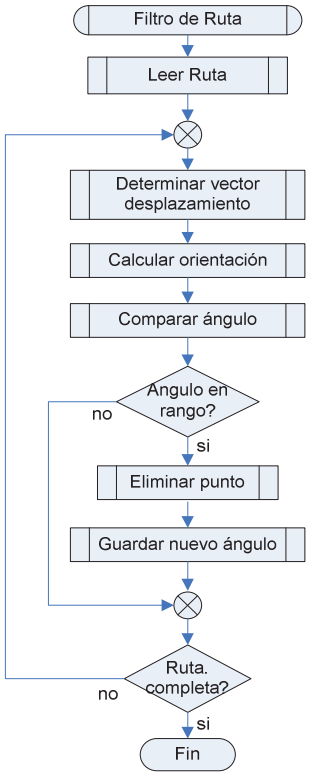


Figura 3.64 Diagrama de flujo de filtrado de ruta

El programa filtra la ruta utilizando un vector desplazamiento $\vec{\boldsymbol{V}}_d$, generado por dos puntos adyacentes de la misma.

$$\vec{\boldsymbol{V}}_d = \vec{\boldsymbol{P}}_i - \vec{\boldsymbol{P}}_{i-1} \tag{3.34}$$

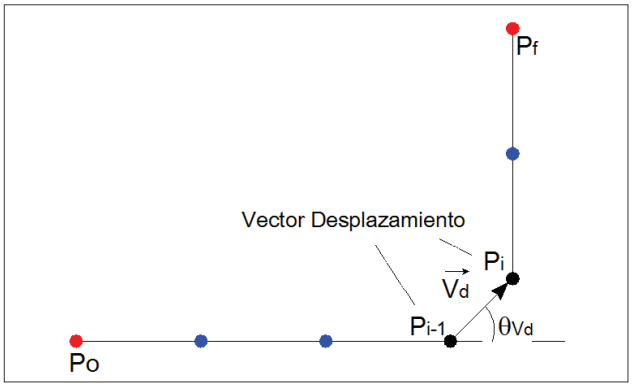


Figura 3.65 Formación del vector de desplazamiento

El ángulo del vector desplazamiento θ_{Vd}, se calcula con la ecuación 3.35:

$$\theta_{Vd} = tan^{-1}\left(\frac{V_{dy}}{V_{dx}}\right) \tag{3.35}$$

El valor de este ángulo es almacenado en un registro para cada vector desplazamiento $\vec{V}_d$, y continua almacenándose hasta que exista un cambio busco de dirección del camino.

El programa toma el vector del par de puntos adyacentes, compara con el guardado, y determina si existe cambio de dirección del camino significativo o no (Figura 3.65). La comparación se realiza utilizando un rango de aceptación del ángulo guardado θ_{Vd} . En el caso que esté en el rango, el programa elimina el punto inicial del vector $\vec{P}_i$ y guarda el nuevo valor de ángulo, caso contrario el programa mantiene el ángulo y no borra el punto considerando que el punto es importante y forma parte de la ruta (Figura 3.66).

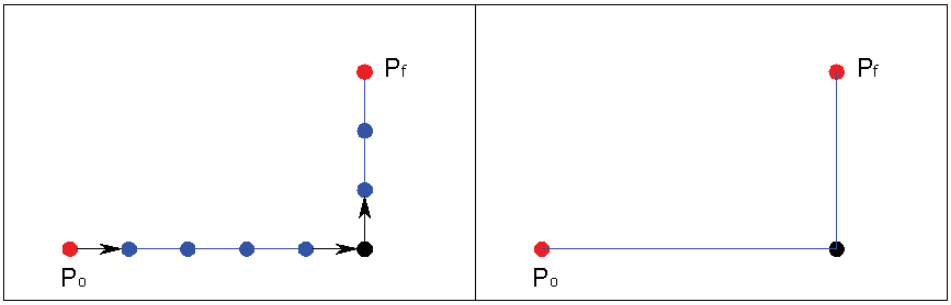


Figura 3.66 Eliminación de puntos innecesarios

Filtrada la ruta el HMI presenta la solución definitiva con los nodos filtrados.

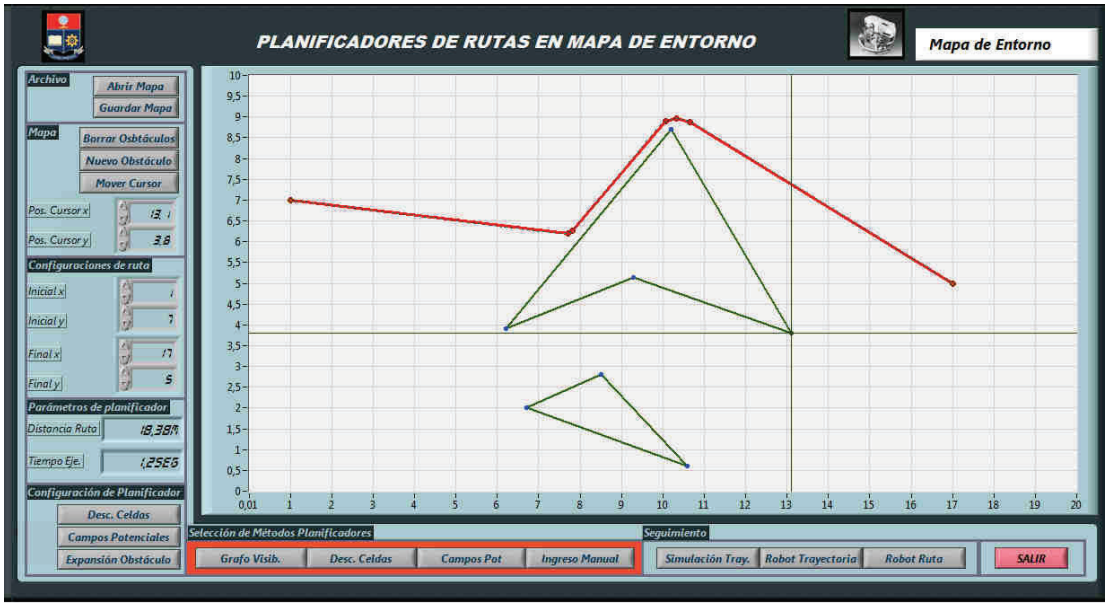


Figura 3.67 Ejecución del programa Campos Potenciales

La Figura 3.67 presenta la solución del algoritmo de campos potenciales, la ruta está filtrada y compone de 5 puntos, manteniendo la información necesaria para ejecutarla utilizando el Robotino®.

3.6 PROGRAMA INGRESO MANUAL DE RUTA

Este programa permite al usuario ingresar una ruta manualmente entre la configuración inicial P_0 y final P_f seleccionada.

El objetivo de este programa es dar una alternativa diferente a la de los métodos descritos anteriormente, en el caso que sea muy complejo generar el grafo o se presenten otros problemas.

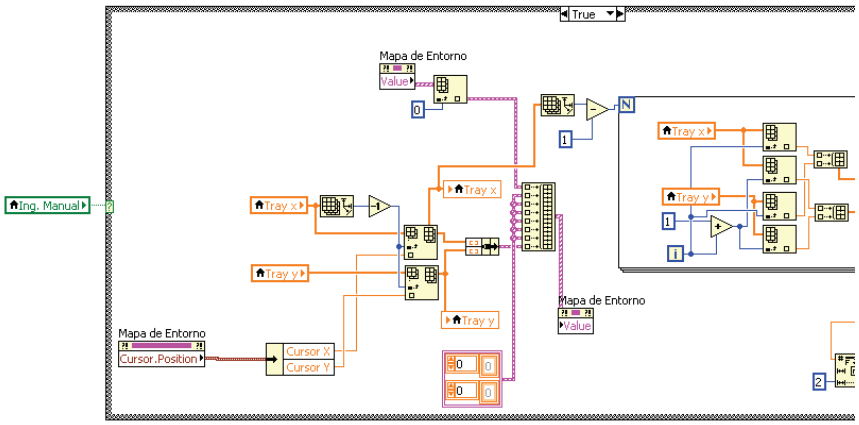


Figura 3.68 Diagrama de bloques para ingreso manual de ruta

Esta rutina es parte del programa principal y la forma de ingreso es igual que la del mapa de entorno, sin tomar en cuenta la geometría de los obstáculos.

Se crea la rutina dentro de la estructura de generación de eventos (Figura 3.11), con interrupción del ingreso manual, previamente se toma los valores de la configuración inicial y final, y coloca en el registro $TRAY\ X$ y $TRAY\ Y$; con la ayuda del bloque de inserción de valores numéricos se ingresa las nuevas coordenadas de los puntos ubicados por el cursor del gráfico y se actualiza el mapa de entorno (Figura 3.68).

Figura 3.69 Ingreso de ruta en un mapa de entorno

El programa calcula la distancia total de la ruta dibujada y se actualiza conforme se sigue ingresando puntos.

Para borrar la ruta se utiliza el botón borrar obstáculos. Activada esta opción no se borra el mapa de entorno.

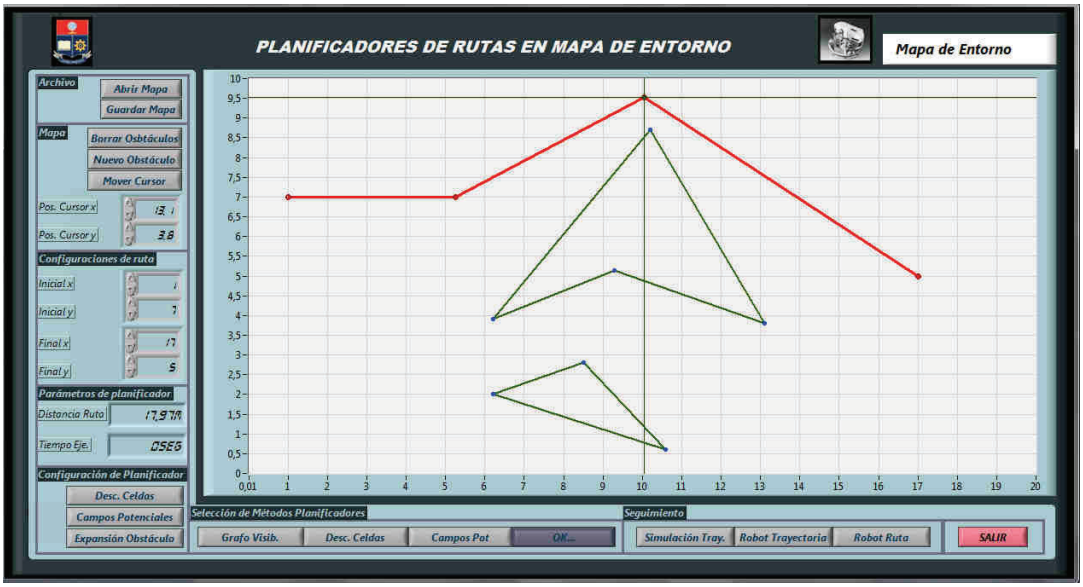


Figura 3.70 Ruta terminada con Ingreso Manual

Las Figura 3.69 y 3.70 presentan el ingreso de una ruta paso a paso, en cada una de ellas se indica la distancia recorrida, de la segunda Figura se ve como el cursor coloca los puntos que forman los vértices de la trayectoria.

Para todos los métodos mencionado el ingreso de la ruta es el mismo, se colocan en el gráfico los valores de los registros $TRAY\ X$ y $TRAY\ Y$, con estos valores y los del mapa de entorno se puede programar al Robotino® de FESTO con la ruta dibujada.

3.7 PROGRAMA CONTROL DE RUTAS

Partiendo del diagrama de bloques del programa principal (Figura 3.3), el control de ruta es la ejecución de los movimientos que debe realizar el Robot para

desplazarse desde la configuración inicial P_0 hacia la final P_f siguiendo la ruta trazada por los métodos de planificación desarrollados.

El control de ruta es una rutina dentro del programa principal (Figura 3.71), que se ejecuta al presionar el pulsante (Robot) del HMI.

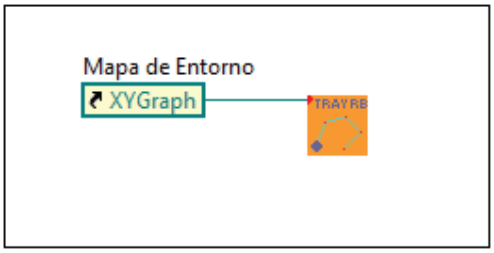


Figura 3.71 Rutina de control de ruta

Antes de construir el diagrama de flujo primero se describe los requerimientos del Robotino® de Festo para controlarlo con LabVIEW.

El Subcapítulo 1.5 (ROBOTINO® DE FESTO), describe las características y funcionalidades del Robotino® en forma general, de las cuales las siguientes se aplicarán al programa de Control de trayectoria.

- Comunicación inalámbrica (Conexión con el LabVIEW).

- Control de motores (Formato de ingreso y Control).

- Lectura de Sensores (Formato de Lectura).

3.7.1 CONFIGURACIÓN DEL ROBOTINO® DE FESTO

En LabVIEW existen librerías instalables compatibles en las existentes en el programa Robotino® VIEW que tiene las mismas características funcionales que se presentan en la Figura 3.72.

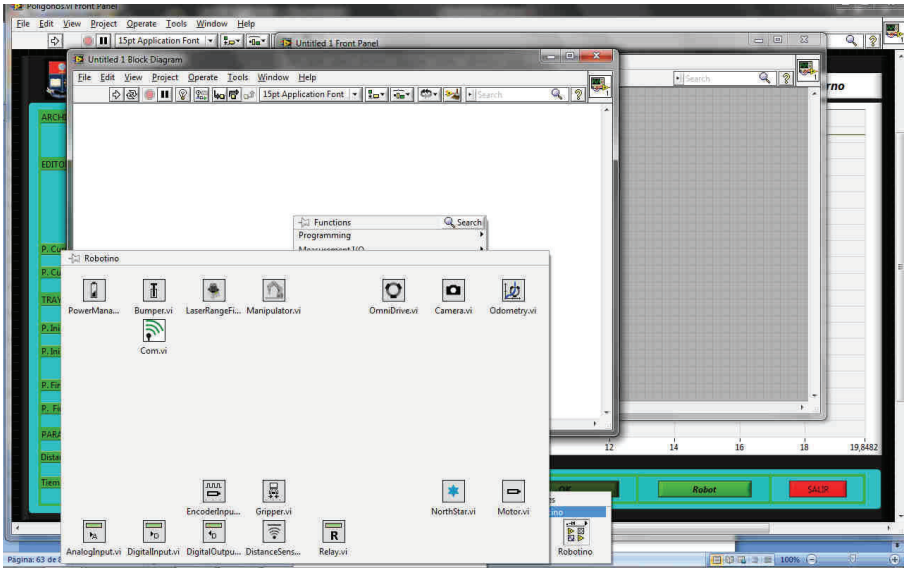


Figura 3.72 Librerías para el control del Robotino®

Estas librerías abarcan las configuraciones del Robotino® para el monitoreo y control del Robotino® de Festo, entre ellos se destacan el control de los motores, conexión con el computador por LAN, lectura de los sensores internos, encoders, estado de entradas y salidas digitales, visualización de la cámara, autonomía, etc.

3.7.1.1 Configuración del Robotino® para la Conexión con el PC

El Robotino® tiene dos tipos de funcionamiento:

- Autónomo (funcionamiento con un programa grabado en su memoria interna).

- Controlado en tiempo real (A través de comunicación Wireless).

Sí se utiliza la comunicación Wireless, al Robotino® se puede configurar como un servidor en una red WLAN, de manera que la conexión con un computador sea Peer to Peer. Al Robotino® se asigna una dirección IP, una máscara de subred y un Gateway, y se ingresa el password que puede ser WEP o WPA, y automáticamente el Robotino® habilita una red inalámbrica para comunicarse (Figura 7.73).

Figura 3.73 Red creada por el Robotino®

El otro método de conexión es configurar al Robotino® como un dispositivo IP que forme parte de una red WLAN establecida, esta arquitectura permite controlarlo desde un computador que no necesariamente tenga el Hardware de comunicación Wireless.

Establecida la comunicación con el Robotino® de Festo, el próximo paso es la conexión con el LabVIEW. El bloque (Com.vi) permite la conexión del Robotino® necesitando la dirección IP y el número de puerto.

La dirección IP se ingresa utilizando una variable tipo STRING, y para conectarlo o desconectarlo se utiliza selectores.

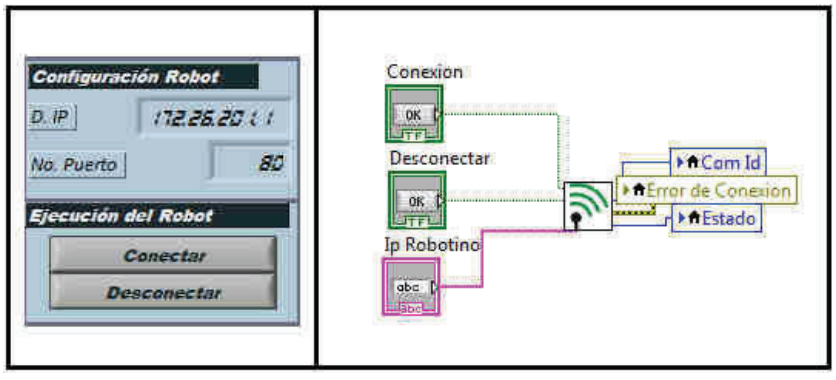


Figura 3.74 Conexión del Robotino® con el PC

El indicador (Com id) de la Figura 3.74 muestra el estado de conexión del Robotino®, cuando el robot está desconectado este indicador presenta el valor -1, y cuando se conecta este indicador presenta un número entero positivo.

El indicador (Error de conexión) permite visualizar si el computador está conectado en la red del Robotino®.

3.7.1.2 Configuración de Encoders

El Robotino® de Festo tiene tres encoders incrementales que indican la posición y la orientación respecto a un punto de referencia.

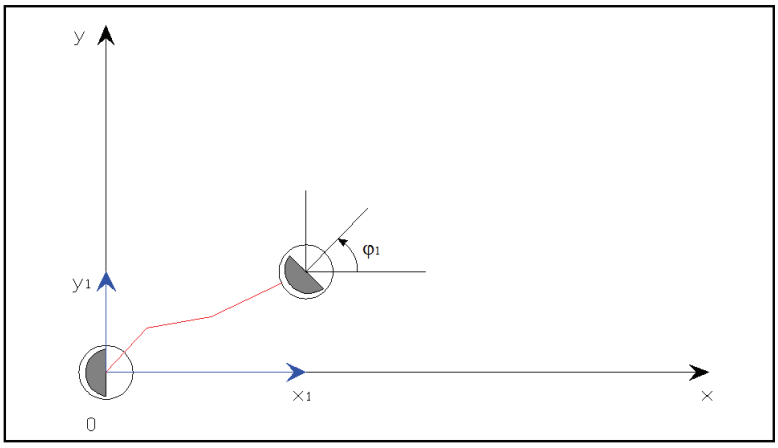


Figura 3.75 Movimientos producidos por el Robotino®

La librería de control del Robotino® tiene un bloque de monitoreo de los encoders llamado ODOMETRY.

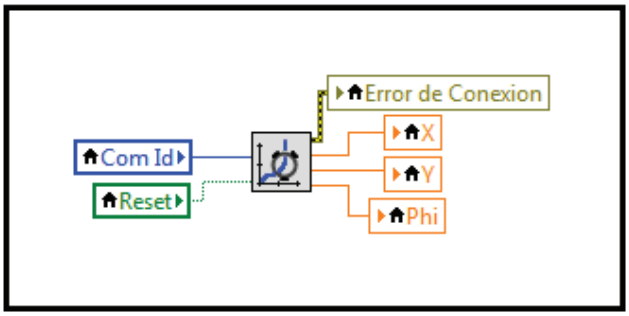


Figura 3.76 Bloque Odómetro

Este odómetro presenta la posición del Robotino® en coordenadas x, y en milímetros y la orientación en grados.

Para leer los valores del Robotino® hay que enviar el número de identificación que se obtiene con el bloque de comunicación (Figura 3.74) y el odómetro presenta los valores de los encoders en tiempo real, además se tiene un reset que encera los valores de todos los encoders.

Los indicadores x, y manejan valores negativos con resolución de $1mm$, y el rango de acción de la orientación (φ) comprende entre $-180^o, 180^o$.

Los encoders x, y están relacionados con la orientación (φ). Al rotar el robot cambia el eje de referencia del robot de acuerdo con las expresiones:

$$x_1 = xcos(\varphi) \tag{3.36}$$

$$y_1 = xsen(\varphi) \tag{3.37}$$

3.7.1.3 Control de Motores

Igual que el monitoreo de los encoders, el Robotino® es controlado utilizando tres movimientos x, y, φ.

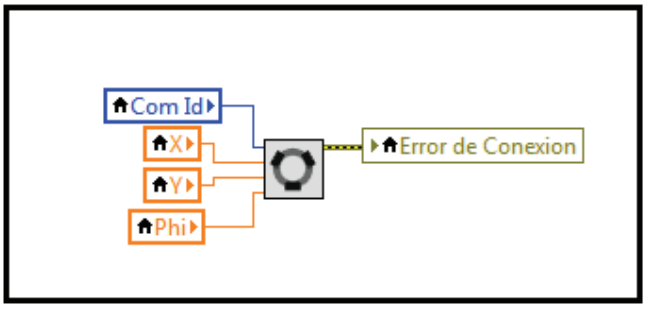


Figura 3.77 Bloque de control de movimientos

El bloque tiene el modelo cinemático del Robotino®, al cual se le ingresa los valores de las velocidades en x, y y la velocidad de rotación ω, los registros están

configurados para tomar los valores de velocidad en mm/s, y la velocidad de rotación en $grados/s$. Para realizar cambios de dirección y retrocesos, se ingresan valores negativos al bloque de control de motores, y para detener al Robotino® se coloca las velocidades en cero.

Las componentes de la velocidad no están relacionadas con los encoders. Por ejemplo si se toma de referencia a la cámara como el frente del Robotino® (Figura 1.22), el movimiento de avance se ejecuta cuando se aplica la velocidad en x, y solo en ese sentido sin importar la orientación del Robotino®.

La velocidad en y ejecuta el movimiento lateral del robot respecto a la cámara del robot en todos los casos.

Esta ejecución de movimientos cambia los ejes de referencia respecto a los valores de los encoders, por lo que hay que realizar conversión de coordenadas.

3.7.2 DIAGRAMA DE FLUJO DE CONTROL DE RUTA

El programa controla los motores del Robotino® para ejecutar los movimientos necesarios y seguir la ruta obtenida por los métodos planificadores.

La homologación de los métodos planificadores desarrollados disminuye la complejidad de programación a un solo algoritmo que sea efectivo con todas las rutas obtenidas.

El diagrama de flujo de la Figura 3.78 presenta las condiciones necesarias y los controladores para ejecutar los movimientos del Robotino®, el cual se describe más adelante.

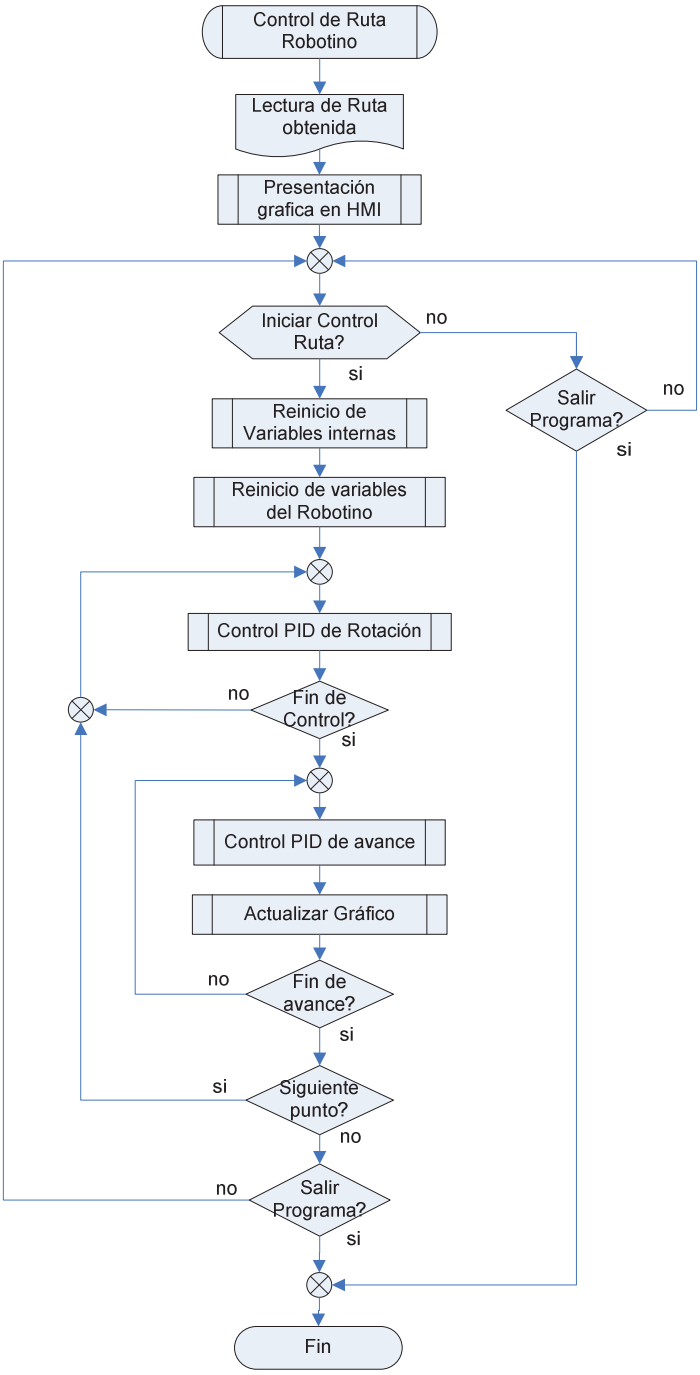


Figura 3.78 Diagrama de flujo para control de ruta

Las rutas formadas tienen la información necesaria para determinar las distancias y cambios de dirección que la componen como los nodos y la secuencia de ellos.

El Robotino® de Festo, tiene la capacidad de ejecutar los movimientos utilizando los mismos cambios de dirección simplemente utilizando rotaciones sobre su eje para mantener el frente sobre el curso de la ruta.

La opción que controla al Robotino® (Figura 3.15) recibe la información del mapa de entorno y de la ruta obtenida. Al ejecutar el control de ruta el HMI abre otra ventana (Figura 3.79), que presenta todas las opciones necesarias de configuración y ejecución del control de ruta.

Figura 3.79 Panel frontal del control de Rutas

El programa primero presenta en un plano de las mismas dimensiones que el dibujado, el mapa de entorno y la ruta entre las configuraciones inicial P_0 y final P_f.

El panel cuenta con las opciones de conexión del Robotino®, display de ingreso de dirección IP y número de puerto e indicadores de estado de conexión.

De acuerdo al diagrama de flujo de control de ruta (Figura 3.78), el programa espera la orden de iniciar el control o salir del mismo. La opción de salir cierra la ventana de control de ruta y retorna al programa principal.

El botón arrancar manda a ejecutar la ruta al Robotino® cuando este se encuentre conectado. Una vez ejecutado el control el programa de control de ruta toma las coordenadas y la secuencia de los nodos, para luego calcular los cambios de dirección (φ) y la distancia entre los nodos (z).

Luego el programa reinicia las variables internas y los valores de los encoders del Robotino® para cada nueva ejecución de control de ruta utilizando el bloque ODOMETRY, luego se escalan los valores de los encoders y realiza la conversión de coordenadas.

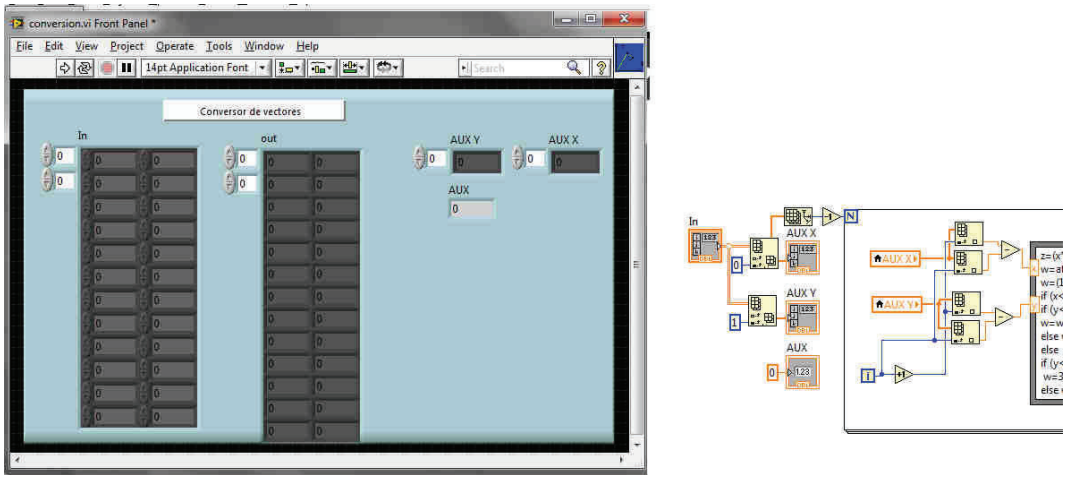


Figura 3.80 Programa de Conversión de coordenadas

Para el escalamiento se utiliza las ecuaciones:

Movimiento en x.

$$x_{rob} = 1000x_{tray} \tag{3.38}$$

Movimiento en y.

$$y_{rob} = 1000y_{tray} \tag{3.39}$$

Las ecuaciones 3.38 y 3.39 escalan la ruta a milímetros, para mantener la resolución de lectura de los encoders el robot.

Para el ángulo de orientación se mantiene el mismo valor.

La ruta escalada se ingresa al programa de conversión de coordenadas y se obtiene vectores equivalentes que tienen la distancia entre puntos (Px_i, Py_i) y el ángulo de orientación φ_i (Figura 3.80). Para los cálculos se utilizan las ecuaciones 3.40 y 3.41

$$z_i = \sqrt{(Px_{i+1} - Px_i)^2 + (Py_{i+1} - Py_i)^2} \tag{3.40}$$

$$\varphi_i = tan^{-1}\left(\frac{Py_{i+1} - Py_i}{Px_{i+1} - Px_i}\right) \tag{3.41}$$

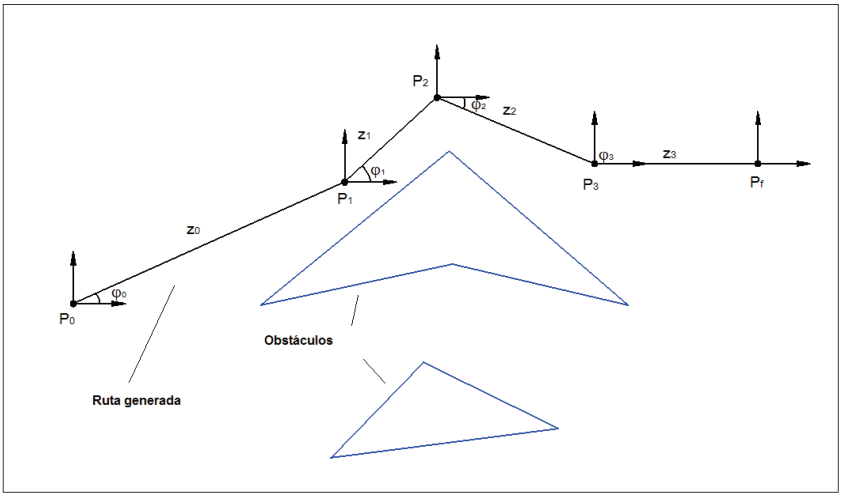


Figura 3.81 Conversión de coordenadas

La Figura 3.81 presenta la conversión de los nodos en vectores que tienen la distancia y la orientación.

3.7.3 CONTROL DE ROTACIÓN

La conversión de la ruta en vectores permite manejar el Robotino® utilizando coordenadas cilíndricas y elimina la componente y, dejando el control de posición para las componentes x y φ.

Para el control de rotación se implementa un sistema de control donde la acción de control es la velocidad de rotación del robot ω, y para el controlador se utiliza un PID. El sistema de control se representa en el diagrama de bloques de la Figura 3.82

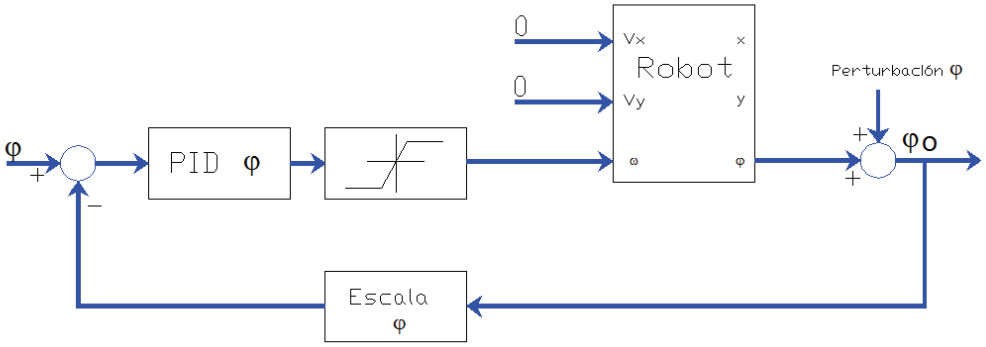


Figura 3.82 Diagrama de bloques de control PID de rotación

El sistema de control en lazo cerrado de la Figura 3.82 tiene como señal de referencia el valor deseado del ángulo que produce el quiebre de la ruta (Figura 3.81), la señal de control está limitada por saturadores que controlan la velocidad máxima de rotación y el sentido de giro.

Para el diseño del controlador PID se parte de la función de transferencia del Robotino® para el giro.

Debido a que el robot es un sistema complejo y por ende difícil de modelarlo, se realizan pruebas directamente con el LabVIEW utilizando una función paso donde la variable de ingreso es la velocidad de rotación controlada con el bloque de control de movimientos (Figura 3.77) y la respuesta es la orientación del Robotino® obtenida con el bloque odómetro (Figura 3.76).

Utilizando una velocidad de rotación de $30^o/s$, la respuesta transitoria de la rotación del Robotino® en lazo abierto se presenta en la Figura 3.83

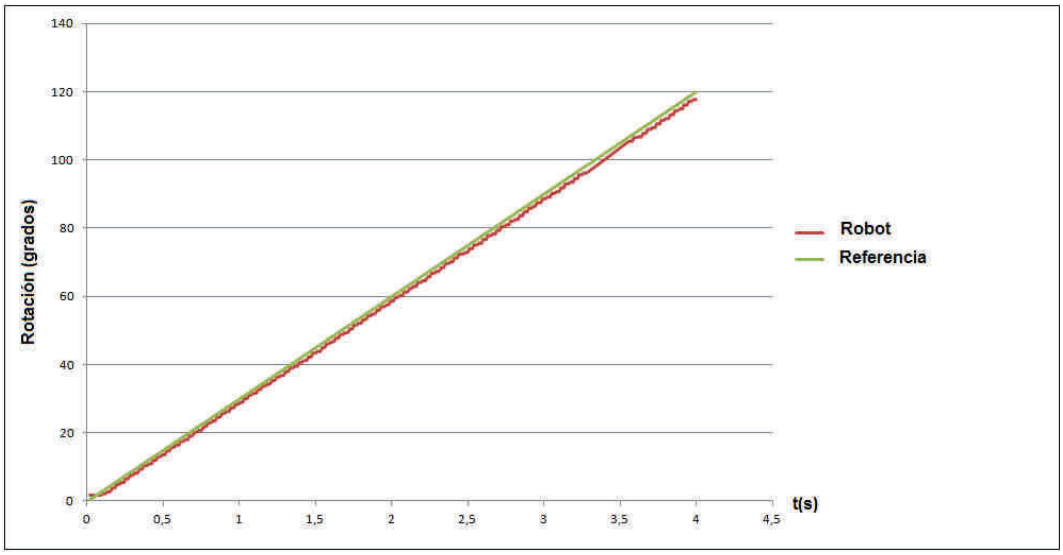


Figura 3.83 Respuesta del Robotino® ante una entrada paso de velocidad de rotación

La respuesta de posición para una entrada paso de $4s$, es proporcional a para la a partir de los $500ms$, tiempo donde se produce el estado transitorio del sistema, para lo cual se deriva la respuesta de la Figura 3.83 para ver la respuesta de velocidad del Robotino® en el estado transitorio (Figura 3.84).

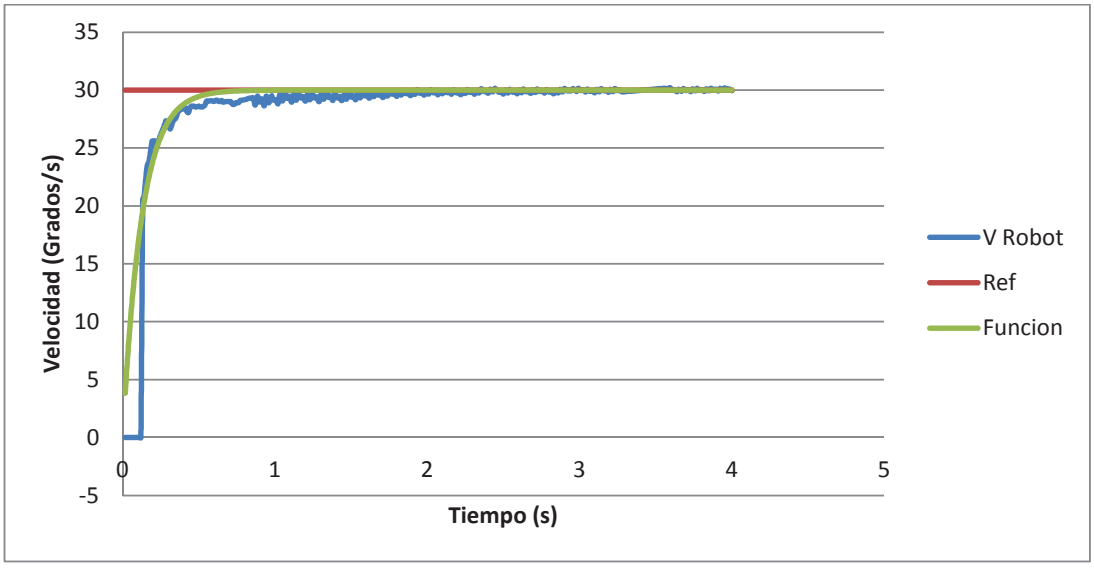


Figura 3.84 Respuesta de velocidad del Robotino® ante una entrada paso

La respuesta de la velocidad en lazo abierto no presenta sobre pico, y a partir de los $500ms$ se vuelve estable (tiempo de establecimiento) a la velocidad ingresada al bloque de control de velocidad. La respuesta de velocidad del Robotino (Figura 3.84) se aproxima a una función de transferencia de primer orden:

$$G_{ROB\,\omega} = k\frac{\frac{1}{\tau_m}}{(s+\frac{1}{\tau_m}))} \tag{3.42}$$

Donde:

- k: Ganancia del sistema
- τ_m: Constante de tiempo

Y con la fórmula de la constante de tiempo:

$$t_s = 4\,\tau \tag{3.43}$$

La función de transferencia del Robotino® es la siguiente conociendo que la ganancia del sistema $k = 1$ (Figura 3.84):

$$G_{ROB\,\omega} = \frac{8}{(s+8)} \tag{3.44}$$

Incorporando un polo en el origen a la función de transferencia de velocidad (3.44), se obtiene la respuesta de la orientación del Robot en lazo abierto.

$$G_{ROB\,\varphi} = \frac{8}{s(s+8)} \tag{3.45}$$

Utilizando Simulink (Figura 3.85) la Respuesta de velocidad de rotación ω_{ROB} y de orientación φ_{ROB} para una velocidad de rotación de $30^o/s$ es:

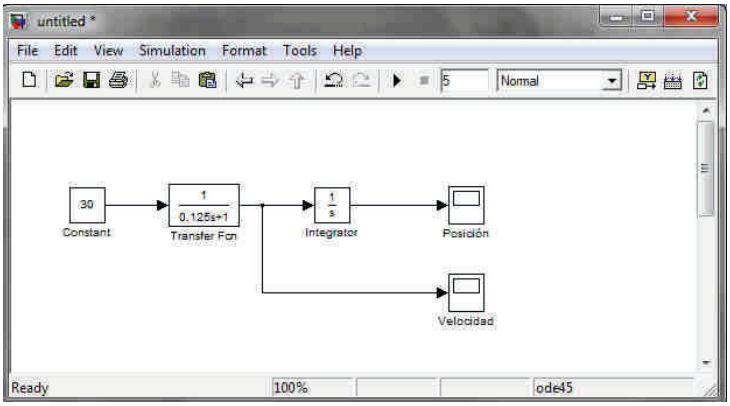


Figura 3.85 Diagrama de bloques de la velocidad del Robotino®

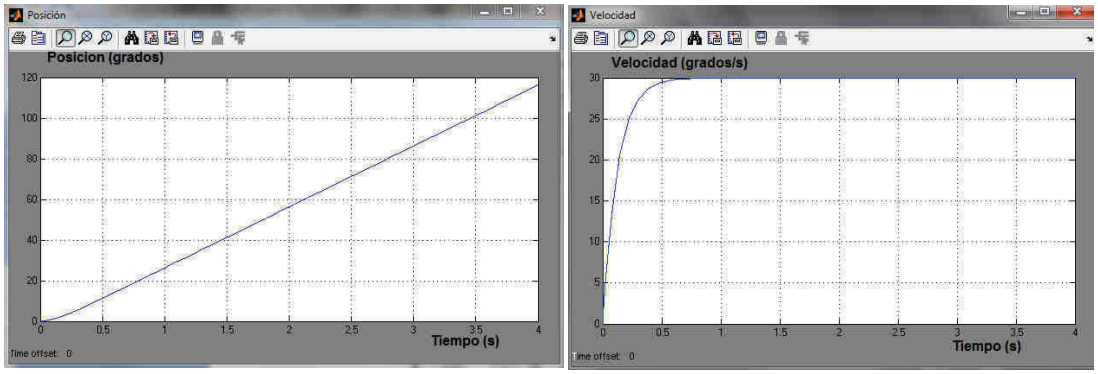


Figura 3.86 Respuesta de posición y velocidad en lazo abierto

La respuesta de la función de transferencia de velocidad de rotación y de orientación calculada (Figura 3.86), se aproxima a la respuesta del Robot medida. Incorporando la función de transferencia de rotación del robot ($G_{ROB\,r}$) a un sistema de control en lazo cerrado el diagrama de bloques es el siguiente:

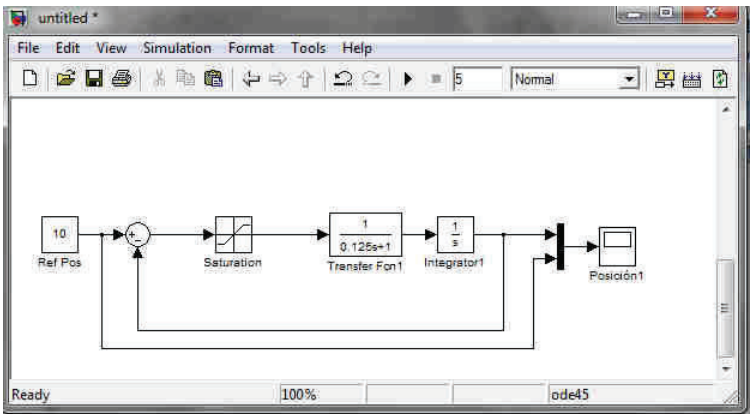


Figura 3.87 Diagrama de bloques para en control en lazo cerrado

El sistema de control de la Figura 3.87 tiene un saturador de velocidad de rotación con límites de $15^o/s$ ya que este valor es el máximo valor confiable para la rotación sin errores significativos debido al rozamiento.

La respuesta de ángulo de rotación del sistema de control de la Figura 3.87 para una rotación 10º es:

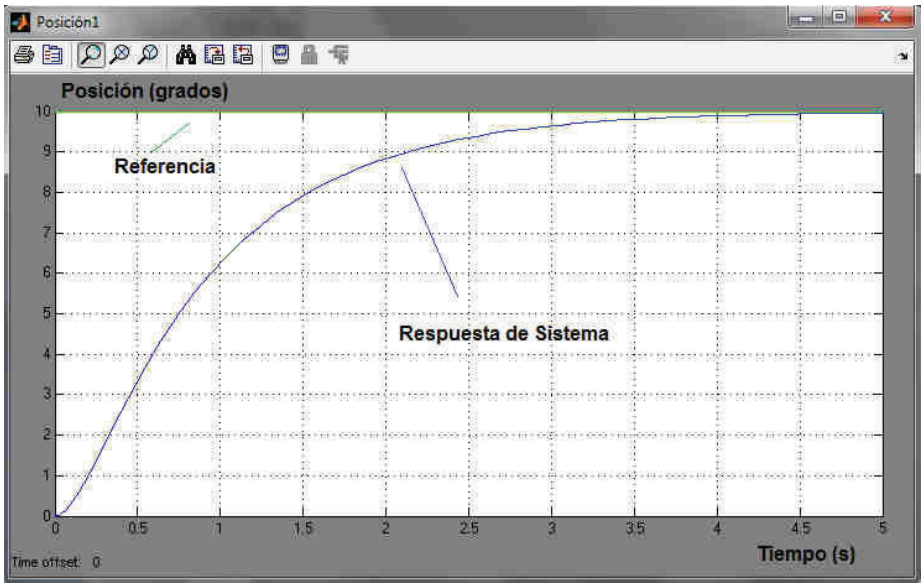


Figura 3.88 Respuesta del sistema para una rotación de 10º

De acuerdo con la Figura 3.88 la respuesta obtenida el tiempo que tardaría el Robotino® en rotar 10º es de 3.5 segundos, además esta respuesta tiene variaciones de la velocidad representada por las curvas en el estado transitorio y no llega a la velocidad máxima del saturador, aumentando el tiempo de reacción.

Al incluir un controlador al sistema de control de la Figura 3.87 se puede disminuir el tiempo que tarda el Robot en girar ya que se mantiene la velocidad constante al valor de saturación.

Para el diseño del controlador se parte de la función de transferencia del controlador PID paralelo:

$$G_{PID} = k_p(1 + t_d s + \frac{1}{t_i s}) \tag{3.46}$$

Este controlador tiene 2 ceros y un polo en el origen que permite eliminar el control de posición de un sistema en lazo cerrado, pero al tener la función de transferencia del Robotino® un polo en el origen se puede eliminar la parte integral del controlador y solo considerar la parte proporcional y derivativa.

$$G_{PID\,\varphi} = k_p(1 + t_d s) \tag{3.47}$$

Un primer ajuste del controlador es eliminación de polos y ceros, agregando también una constante de tiempo de $200ms$ entonces las constantes del controlador son:

$$k_p = 5$$
$$t_d = 0.125$$
$$G_{PID\,\varphi} = 5(0.125s + 1) \tag{3.48}$$

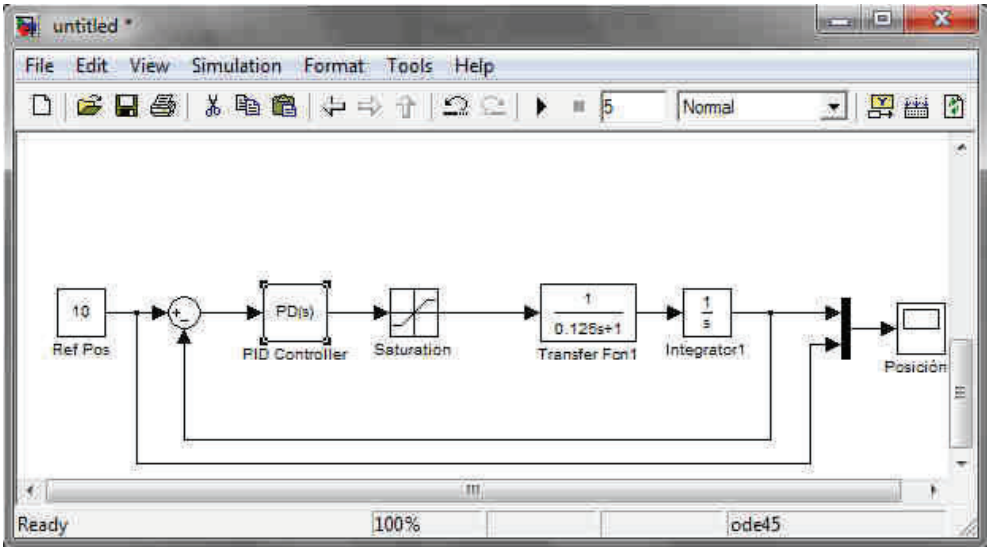


Figura 3.89 Sistema de control para rotación del Robotino®

Integrando el control PID al sistema de control de rotación del robot (Figura 3.89), la respuesta del sistema para una rotación de 10º utilizando Simulink es:

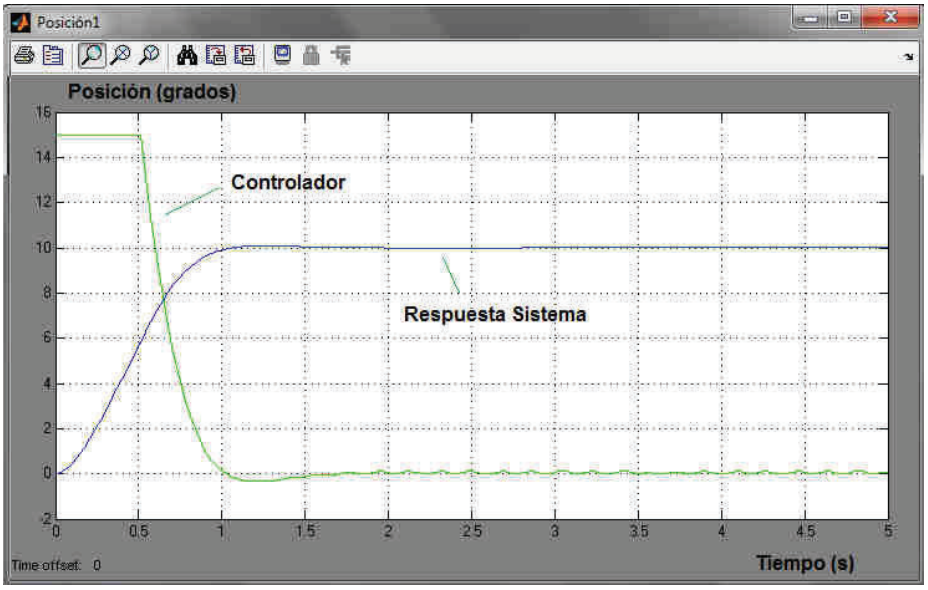


Figura 3.90 Respuesta del sistema con limitación de velocidad

La Figura 3.90 indica que la respuesta del sistema de control tiene un tiempo de establecimiento de $0.85s$, el cual es cuatro veces menor cuando no se utiliza un control, además en la parte transitoria la respuesta es prácticamente lineal y sin oscilaciones, permitiendo que el robot gire prácticamente a una velocidad de rotación constante de $15^o/s$ como indica la señal de control hacia la función de transferencia de orientación del Robot, lo que asegura un buen control en la orientación del robot.

Para incorporar el control PID al programa de control de ruta LabVIEW cuenta con una librería de control a la que se puede incorporar un controlador PID, al que se le puede calibrar las constantes obtenidas, los saturadores y el tiempo de muestreo.

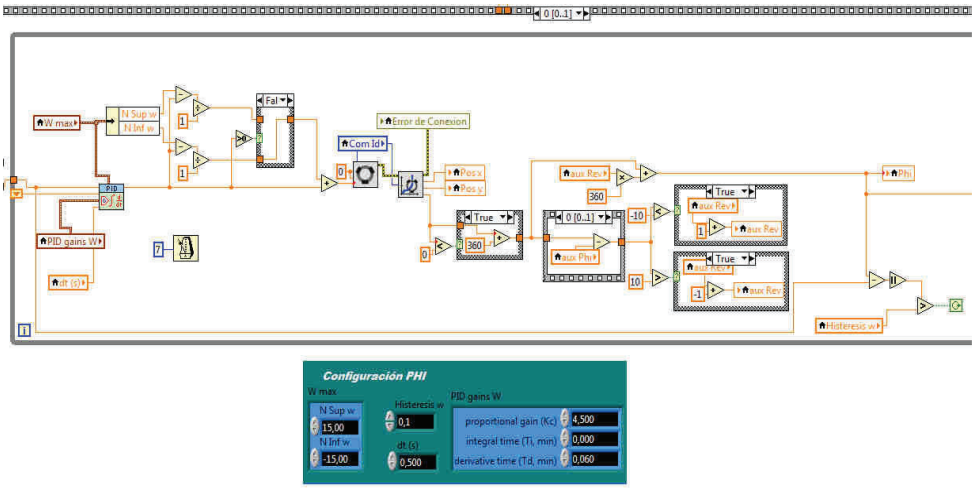


Figura 3.91 Control PID de rotación

El bloque PID se coloca dentro de un lazo WHILE para controlar al Robotino® hasta que llegue al ángulo calculado por la rutina Conversión de Coordenadas (Figura 3.80).

3.7.4 CONTROL DE TRASLACIÓN

El movimiento de avance del Robotino® se da cuando se aplica la velocidad en la componente x del bloque que controla los movimientos del robot (Figura 3.77), pero existen perturbaciones que modifican el movimiento de avance desviando de su ruta, por lo cual se debe controlar también los movimientos laterales (Control movimientos en y), y la rotación del Robotino® ω.

Para tener una mejor respuesta del Robotino®, se incluye el control PID en los movimientos lateral y el de rotación.

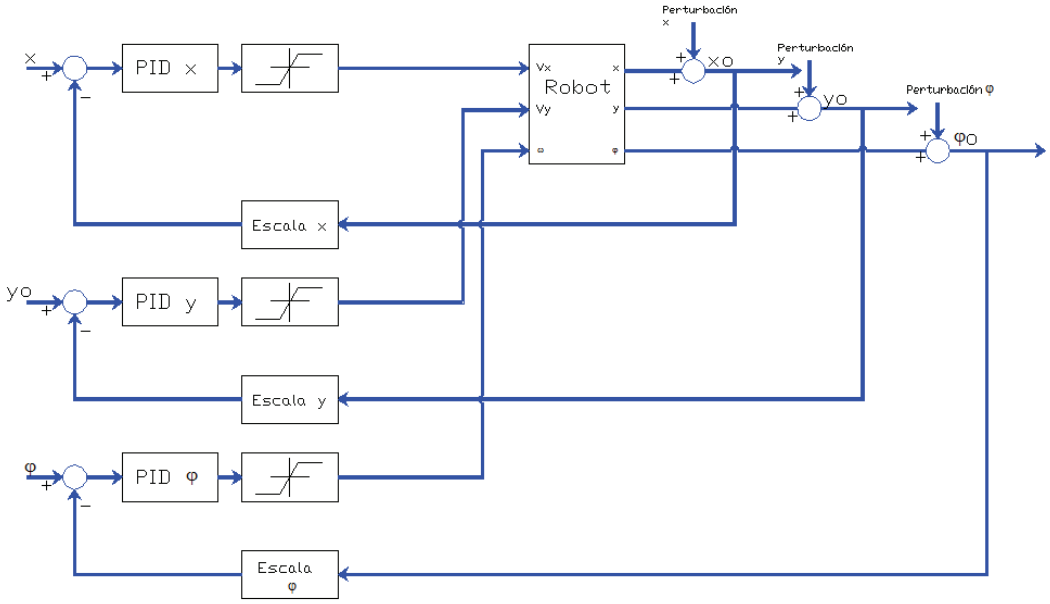


Figura 3.92 Diagrama de bloques de control PID de traslación

Tanto el movimiento en y como en φ, debe mantenerse sin alteración, por ello el diagrama de bloques para el control de traslación incluye el control de todas las variables del bloque de control de movimientos del Robotino® (Figura 3.92).

El controlador PID para la rotación es el mismo que para el cambio de orientación del Robotino®.

Para el diseño del controlador PID de traslación se parte de la función de transferencia del Robotino®. Siguiendo la misma línea del control de rotación los motores responden de la misma forma para producir la traslación, entonces la función de transferencia para la velocidad de avance del robot V_x es:

$$G_{ROB\,vx} = \frac{8}{(s+8)} \tag{3.49}$$

La Respuesta de posición y velocidad del sistema (Figura 3.93) cuando se aplica una velocidad de avance Vx de $150mm/s$ es:

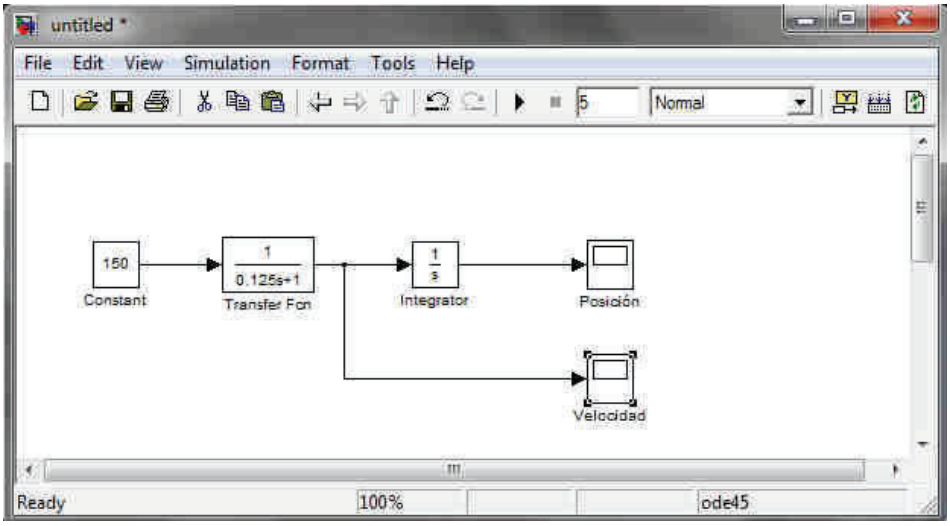


Figura 3.93 Control en lazo abierto de velocidad de lineal y posición del Robotino®

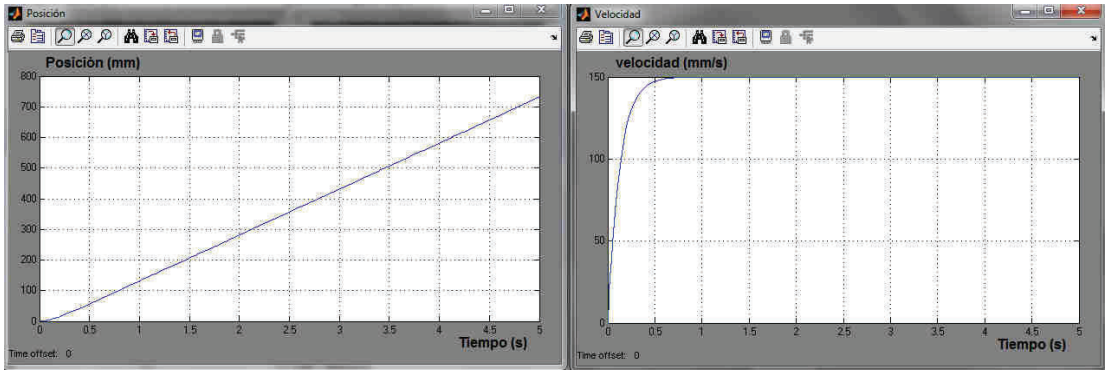


Figura 3.94 Respuesta de posición y velocidad en x del Sistema

Para realizar el control de posición de avance del Robotino®, se coloca un integrador a la función de transferencia de la Figura 3.93, la siguiente ecuación representa el bloque del Robotino® que toma los valores de los encoders.

$$G_{ROB\,X} = \frac{8}{s(s+8)} \tag{3.50}$$

La respuesta de posición del sistema de control en lazo abierto de la Figura 3.94 indica que la velocidad del robot tiene error cero, y la respuesta de la posición es

incremental, para realizar el control de la posición se cierra el lazo de control como muestra la Figura 3.95.

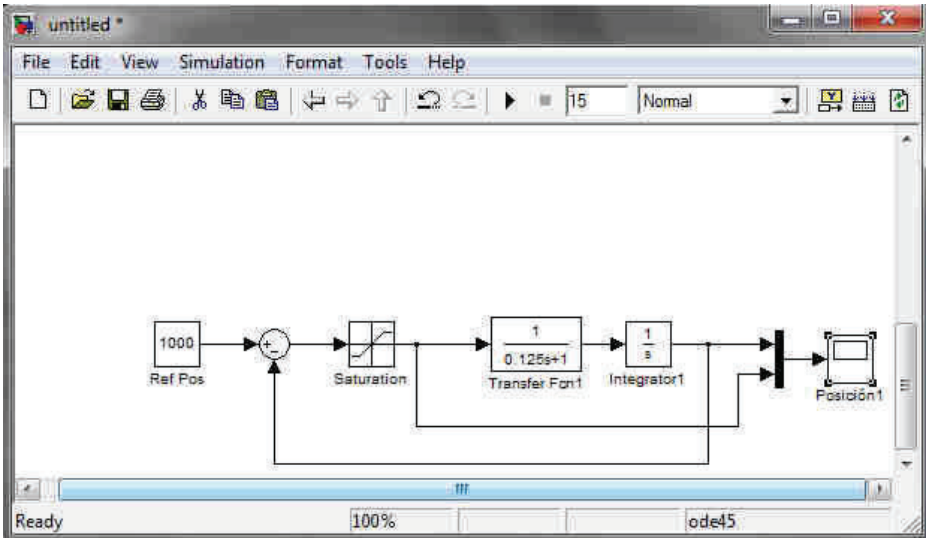


Figura 3.95 Control de posición en lazo cerrado

El sistema de control en lazo cerrado cuenta con un saturador a la velocidad de avance del robot de $150mm/s$. Aplicando un desplazamiento en x, de $1000mm$, la respuesta del sistema es:

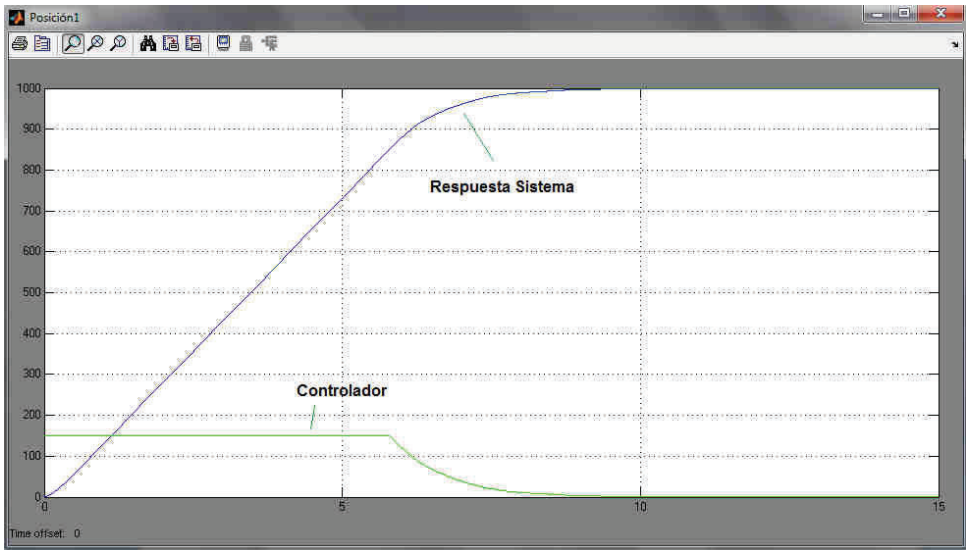


Figura 3.96 Respuesta de posición del Robotino®

En la repuesta de posición del Robotino® de la Figura 3.96 se observa que no tiene error de posición entre el valor del Set Point y el avance del Robotino®.

Cuando el robot se acerca a la posición deseada entre 0.8 y $1m$, reduce su velocidad de la misma forma que en el caso de la rotación, aumentando el tiempo de control de traslación.

Al sistema de control, se incluye un control PID para disminuir el tiempo transitorio en el caso que la distancia de avance sea menor de $0.5m$.

Ya que el sistema tiene un polo en el origen, no es necesario colocar la parte integral en el controlador PID para mantener el error de posición en cero.

Utilizando la técnica de eliminación de polos y ceros, el controlador PID es:

$$G_{PID\,x} = 5(0.125s + 1) \tag{3.51}$$

Con este controlador la función de transferencia del sistema en lazo cerrado sin el saturador es:

$$\frac{Y_T}{R} = \frac{5}{s+5} \tag{3.52}$$

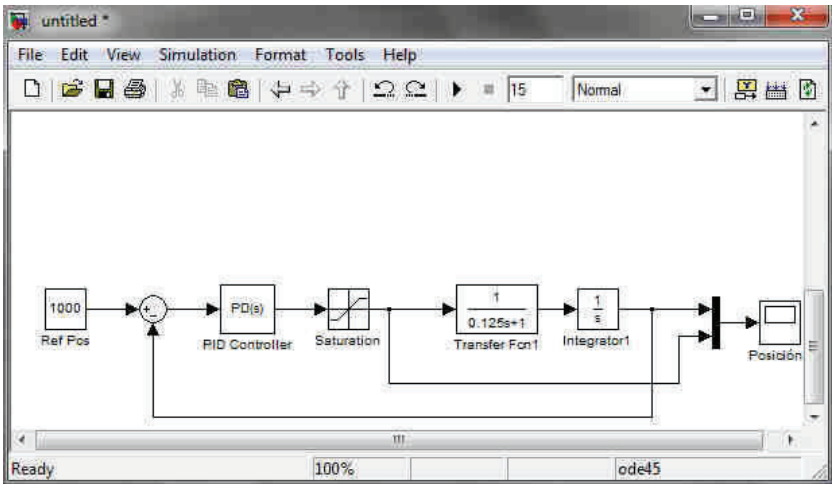


Figura 3.97 Control de posición completo para el Robotino®

Utilizando Simulink se realiza la simulación del sistema de control completo (Figura 3.97), del cual se obtiene los siguientes resultados:

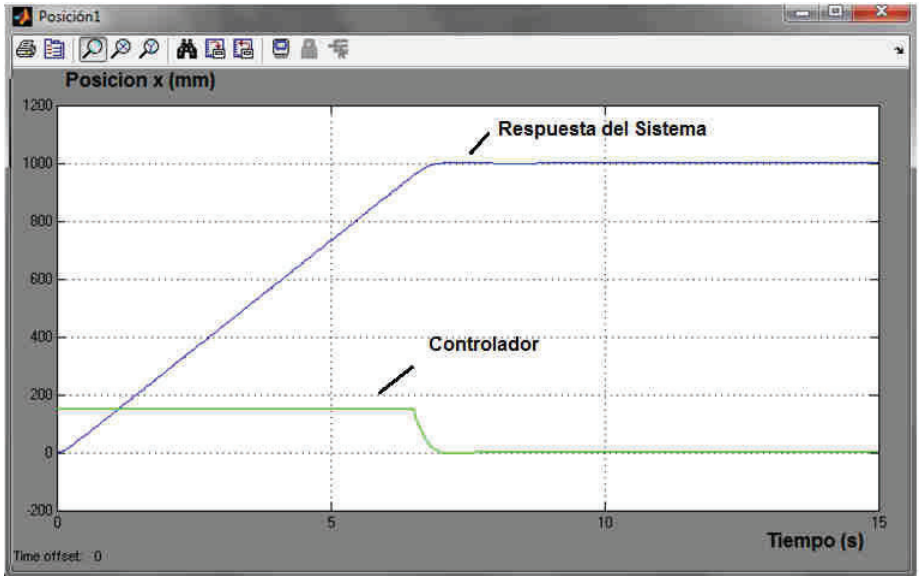


Figura 3.98 Respuesta del Robotino® utilizando el controlador PD

El control PD disminuye el tiempo de control de traslación, manteniendo la velocidad constante hasta un punto que la acción derivativa la disminuye rápidamente sin ocasionar oscilaciones al Robotino® como muestra la Figura 3.98.

En el control de las otras variables se utiliza el mismo controlador y el valor de referencia para la variable y es cero, y para φ es el ángulo calculado por la rutina de Conversión de Coordenadas, permitiendo la estabilidad del sistema para que el robot siga en línea recta.

El sistema de control desarrollado de la Figura 3.92, se integra al programa de control del Robotino® de acuerdo al diagrama de flujo de control de ruta (Figura 3.78).

Las señales de los controladores se envían a los motores del Robotino® como se presenta en la Figura 3.99 y se toma las lecturas de los encoders, las mismas que

son transformadas a valores que el programa los pueda comparar con los del mapa de entorno.

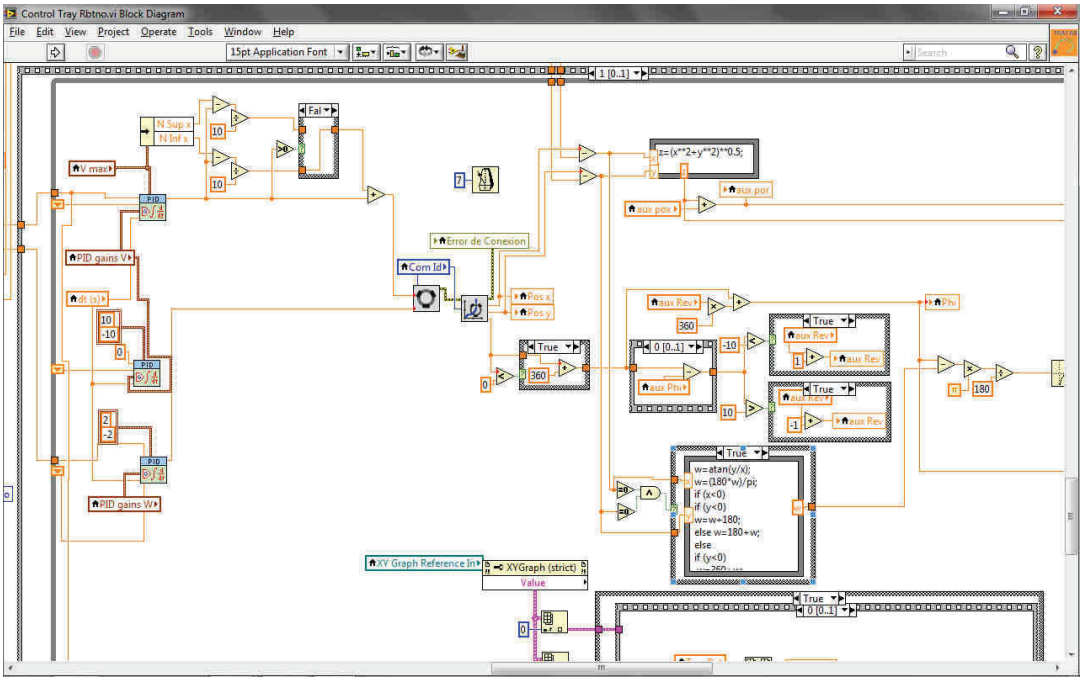


Figura 3.99 Diagrama de bloques para control de Traslación del Robotino® con compensación de desviaciones

El control de rotación y el de traslación se ejecutan secuencialmente y juntos producen el movimiento en línea recta con la distancia y la orientación de la trayectoria. El programa ejecuta todos movimientos y termina deteniendo el Robotino® en la configuración final P_f.

El programa además del controlar al Robotino® monitorea y envía los datos al plano donde los presenta visualmente formando la ruta que sigue el robot y ayuda a comparar con la ruta obtenida por los métodos planificadores.

3.8 IMPLEMENTACIÓN DE CONTROL DE TRAYECTORIA

La última parte del diseño del HMI es la incorporación del control de trayectoria a partir de la ruta obtenida por los métodos planificadores y por el ingreso manual de ruta.

Partiendo de la ecuación matricial de control de trayectoria por métodos numéricos [20]:

$$\boldsymbol{U}_{(k)} = \frac{1}{To}\left(\boldsymbol{X}_{\boldsymbol{T}\,(k+1)} - \boldsymbol{X}_{\boldsymbol{R}\,(k)}\right)\boldsymbol{A}^{-1} \tag{3.53}$$

$$t = kTo \tag{3.54}$$

Donde:

- $\boldsymbol{X}_{\boldsymbol{T}\,(k+1)}$: Configuración de la trayectoria a la que el robot debe llegar.
- $\boldsymbol{X}_{\boldsymbol{R}\,(k)}$: Configuración actual de robot.
- T_o: Periodo de muestreo.
- $\boldsymbol{A}$: Modelo cinemático del robot.
- $\boldsymbol{U}_k$: Acción de control aplicada al robot, y
- $k \in \{0,1,2,3 \ldots .\}$

Dado que el robot utilizando es omnidireccional de tres ruedas, se incluye el modelo cinemático del robot a la ecuación 3.53.

$$\begin{pmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \end{pmatrix} = \frac{1}{To}\begin{pmatrix} x_{\boldsymbol{T}\,(k+1)} - x_{\boldsymbol{R}\,(k)} \\ y_{\boldsymbol{T}\,(k+1)} - y_{\boldsymbol{R}\,(k)} \\ \omega_{\boldsymbol{T}\,(k+1)} - \omega_{\boldsymbol{R}\,(k)} \end{pmatrix}\begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{pmatrix}^{-1} \tag{3.55}$$

Donde:

- $u_{\omega_1},\ u_{\omega_2},\ u_{\omega_3}$: Velocidad de control aplicada a los motores del robot.

- $x_{\boldsymbol{T}\,(k+1)},\ y_{\boldsymbol{T}\,(k+1)},\ \omega_{\boldsymbol{T}\,(k+1)}$: Son las componentes de la posición y orientación del robot a la que se quiere alcanzar.

- $x_{\boldsymbol{R}\,(k)},\ y_{\boldsymbol{R}\,(k)},\ \omega_{\boldsymbol{R}\,(k)}$: Son las componentes de la posición y orientación del robot.

Por otro lado la plataforma de programación LabVIEW cuenta con la rutina de lectura de los encoders del Robotino® (bloque Odómetro) de la Figura 3.76, y con la rutina de control de los movimientos del robot (Figura 3.77), las mismas que en conjunto representan el modelo cinemático del Robotino®, controlado directamente las componentes de velocidad del Robot, y obteniendo la información de la posición y orientación. Con estos bloques de configuración la ecuación 3.55 se reduce a:

$$\begin{pmatrix} u_{\omega_1} \\ u_{\omega_2} \\ u_{\omega_3} \end{pmatrix}\begin{pmatrix} 0 & \frac{R}{\sqrt{3}} & -\frac{R}{\sqrt{3}} \\ \frac{2R}{3} & -\frac{R}{3} & -\frac{R}{3} \\ -\frac{R}{3L} & -\frac{R}{3L} & -\frac{R}{3L} \end{pmatrix} = \frac{1}{To}\begin{pmatrix} x_{\boldsymbol{T}\,(k+1)} - x_{\boldsymbol{R}\,(k)} \\ y_{\boldsymbol{T}\,(k+1)} - x_{\boldsymbol{R}\,(k)} \\ \omega_{\boldsymbol{T}\,(k+1)} - \omega_{\boldsymbol{R}\,(k)} \end{pmatrix}$$

$$\begin{pmatrix} Vx_{C\,(k)} \\ Vy_{C\,(k)} \\ \omega_{C\,(k)} \end{pmatrix} = \frac{1}{To}\begin{pmatrix} x_{\boldsymbol{T}\,(k+1)} - x_{\boldsymbol{R}\,(k)} \\ y_{\boldsymbol{T}\,(k+1)} - x_{\boldsymbol{R}\,(k)} \\ \omega_{\boldsymbol{T}\,(k+1)} - \omega_{\boldsymbol{R}\,(k)} \end{pmatrix} \tag{3.56}$$

Donde:

- $Vx_{C\,(k)}$, $Vy_{C\,(k)}$, $\omega_{C\,(k)}$: Componentes de la velocidad de control aplicada al Robotino®.

Por definición un robot omnidireccional es un sistema que se desplaza en cualquier dirección sin realizar un movimiento previo, por lo que se puede despreciar la componente rotativa del control $\varphi = 0$ manteniendo la misma orientación del robot cuando realice el seguimiento. Simplificando esta componente la ecuación 3.56 se trasforma a una ecuación vectorial que tiene la siguiente forma:

$$\vec{\boldsymbol{V}}_{C\,(k)} = \frac{\vec{\boldsymbol{r}}_{T\,(k+1)} - \vec{\boldsymbol{r}}_{R\,(k)}}{To} \tag{3.57}$$

Donde:

- $\vec{\boldsymbol{V}}_{C\,(k)}$: Velocidad de control aplicada al robot.

- $\vec{\boldsymbol{r}}_{T\,(k+1)}$: Posición de la trayectoria a la que el robot debe llegar.

- $\vec{\boldsymbol{r}}_{R\,(k)}$: Posición actual de Robotino®.

La ecuación 3.57, indica que la velocidad de control depende directamente de la trayectoria construida y de posición del robot, por lo que se requiere construir la trayectoria a partir de la ruta que cumpla con los requerimientos de velocidad del Robotino®.

Agregando a la ecuación 3.57 la posición actual de la trayectoria:

$$\vec{\boldsymbol{V}}_{C\,(k)} = \frac{\vec{\boldsymbol{r}}_{T\,(k+1)} - \vec{\boldsymbol{r}}_{R\,(k)} + \vec{\boldsymbol{r}}_{T\,(k)} - \vec{\boldsymbol{r}}_{T\,(k)}}{To} \tag{3.58}$$

Y conociendo que el periodo de muestreo es:

$$To = t_{(k+1)} - t_{(k)} \tag{3.59}$$

$$\vec{\boldsymbol{V}}_{C\,(k)} = \frac{\vec{\boldsymbol{r}}_{T\,(k+1)} - \vec{\boldsymbol{r}}_{T\,(k)}}{t_{(k+1)} - t_{(k)}} + \frac{\vec{\boldsymbol{r}}_{T\,(k)} - \vec{\boldsymbol{r}}_{R\,(k)}}{t_{(k+1)} - t_{(k)}} \tag{3.60}$$

El desplazamiento de dos posiciones consecutivas de la trayectoria 3.60 en función del periodo de muestro es la velocidad media de la trayectoria $\vec{\boldsymbol{V}}_{\boldsymbol{T}\,(k)}$, y la diferencia entre la posición de la trayectoria $\vec{\boldsymbol{r}}_{T\,(k)}$ y la posición del robot $\vec{\boldsymbol{r}}_{R\,(k)}$ representa el error de posición durante ese periodo de tiempo. La velocidad de control $\vec{\boldsymbol{V}}_{C\,(k)}$ discreta es la suma de la velocidad otorgada por la trayectoria y la velocidad obtenida por el error de posición.

$$\vec{\boldsymbol{V}}_{C\,(k)} = \vec{\boldsymbol{V}}_{T\,(k)} + \vec{\boldsymbol{V}}_{E\,(k)} \tag{3.61}$$

La ecuación 3.61 indica que para calcular la velocidad de control al robot se requiere conocer la trayectoria que sigue el robot y por ende su velocidad, por lo que el siguiente paso del diseño del sistema de control es la construcción de la trayectoria a partir de la ruta considerando los requerimientos de velocidad para el Robotino®.

3.8.1 RUTINA CONSTRUCTOR DE TRAYECTORIA

Desde el punto de vista físico la trayectoria, es un lugar geométrico que describe el movimiento de un cuerpo en el tiempo, por lo que cuenta con la información de posición, velocidad, y aceleración de un cuerpo en cualquier instante.

Con la definición física, matemáticamente la trayectoria se representa por una función vectorial parametrizada en el tiempo:

$$\vec{\boldsymbol{r}}(t) = f_x(t)\vec{\boldsymbol{\imath}} + f_y(t)\vec{\boldsymbol{\jmath}} + f_z(t)\vec{\boldsymbol{k}} \tag{3.62}$$

Para calcular la velocidad a partir de la trayectoria, se deriva la ecuación 3.62 en función del tiempo:

$$\vec{\boldsymbol{v}}(t) = \frac{d}{dt}\vec{\boldsymbol{r}}(t) \tag{3.63}$$

De forma reciproca se puede determinar la trayectoria de un cuerpo si se conoce la función de la velocidad y la posición inicial.

$$\vec{r}(t) = \int_0^t \int \vec{v}(t)dt + \vec{r}_0 \tag{3.64}$$

De acuerdo con la ecuación 3.64 se puede construir una trayectoria a partir de la velocidad que el robot puede ejecutar dentro de límites aceptables.

La rapidez que puede desarrollar el Robotino® de forma segura en cualquier dirección determinada experimentalmente es $V = 0.5m/s$.

De la ecuación (3.60), para mantener la rapidez de la trayectoria $\left|\overrightarrow{\boldsymbol{V_T}}\right|$ constante por debajo del valor máximo a la que el Robotino® puede desarrollar, la distancia generada por el desplazamiento entre la posición actual y la siguiente también debe mantenerse constante siendo el periodo de muestreo la variación del tiempo entre dos posiciones consecutivas (ecuación 3.59).

$$V_{T\,(k)} = \frac{\left|\vec{\boldsymbol{r}}_{T\,(k+1)} - \vec{\boldsymbol{r}}_{T\,(k)}\right|}{To} = Kte \tag{3.65}$$

El requerimiento de para la construcción de la trayectoria es mantener la distancia constante durante el periodo de muestreo, para garantizar que la rapidez también lo sea.

Para la construcción de la trayectoria a partir de la ruta obtenida por los métodos planificadores se utiliza la interpolación Spline utilizando rectas y arcos de circunferencia.

Para calcular las ecuaciones se parte de la expresión de la recta y de circunferencia:

$$y_n = m_n x_n + b_n \qquad \forall \qquad x_{n-1} \leq x \leq x_n \tag{3.66}$$

$$(x_m - x_o)^2 + (y_m - y_o)^2 = R_m^2 \qquad \forall \qquad x_{m-1} \leq x \leq x_m \tag{3.67}$$

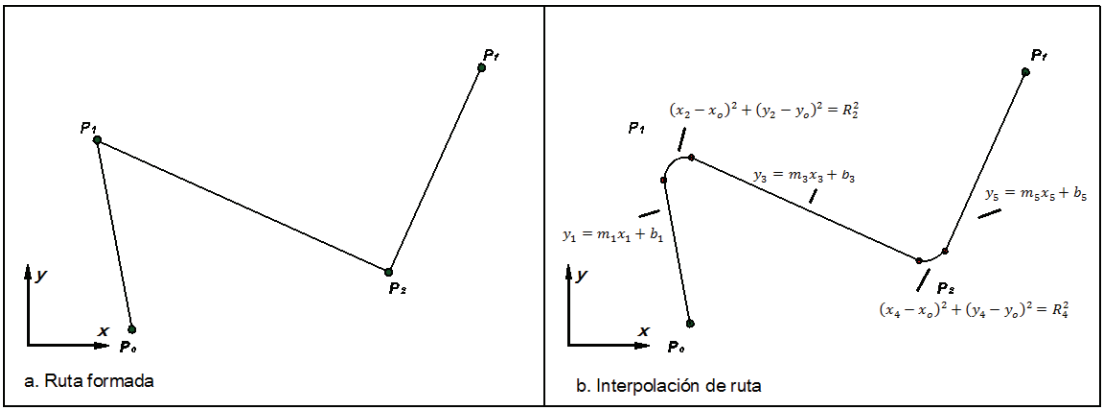


Figura 3.100 Cálculo de rectas para la ruta

Para calcular las rectas se toman las coordenadas de dos nodos consecutivos de la ruta (Figura 3.100), y para eliminar las aristas de la ruta se determina dos puntos de tangencia correspondiente a las rectas formadas que comparten un nodo (Figura 3.101), cada punto de tangencia se encuentra a la misma distancia D del nodo formando dos triángulos rectángulos.

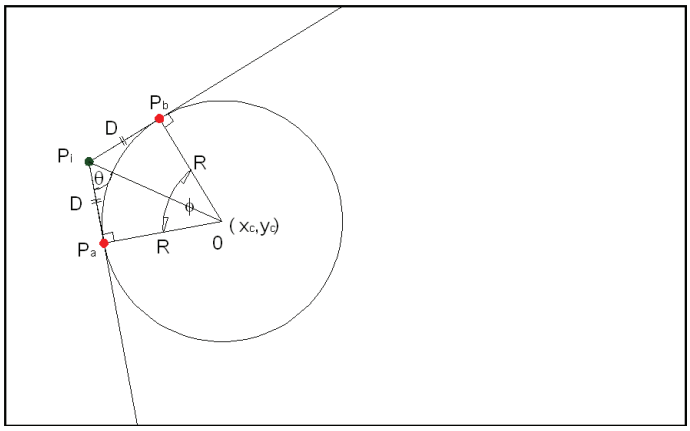


Figura 3.101 Formación de círculos

Para determinar el radio, se resuelve el triángulo rectángulo $\Delta O P_i P_a$, sabiendo que:

$$\theta = \frac{tg^{-1}\left(m_{\overline{P_i P_a}}\right) - tg^{-1}\left(m_{\overline{P_i P_b}}\right)}{2} \tag{3.68}$$

Donde:

- $m_{\overline{P_i P_x}}$: Pendiente de la recta formada por los nodos P_x y P_i,

- D: distancia asignada para el tamaño del círculo.

El siguiente paso para la formación de la trayectoria es la parametrización de las ecuaciones de rectas y arcos de circunferencia en el tiempo, que cumpla con el requerimiento mantener la distancia constante.

Para introducir el parámetro del tiempo, primero se ingresa el parámetro (λ), que es un número entero positivo que representa número de muestras de las funciones de la ruta, discretizando la función de y obtener una tabla con las coordenadas formado un camino para cada valor de (λ).

Tabla 3.3 Almacenamiento de puntos del camino

Dirección de Variable (A)	Posición (x)	Posición (y)	Parámetro λ
0	x_0	y_0	1
1	x_1	y_1	2
2	x_2	y_2	3
	⋮	⋮	4
A_i	x_i	y_i	5
	⋮	⋮	⋮
A_f	x_f	y_f	f

Estas posiciones son calculadas generando un vector partición de la misma forma a la utilizada para la formación de sólidos en la programación del método de Campos Potenciales, utilizando las expresiones 3.5, 3.6, 3.7, 3.8, 3.9, 3.10 y 3.11 y la distancia entre nodos se mantiene constante a $10cm$ de separación Figura 3.102.

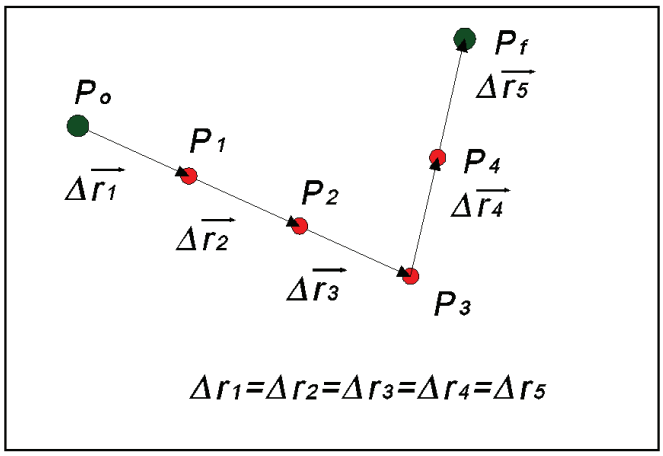


Figura 3.102 Formación de vectores de igual módulo

A cada uno de las posiciones formadas se asigna el una posición en la Tabla 3.3 y se ingresa el parámetro (λ).

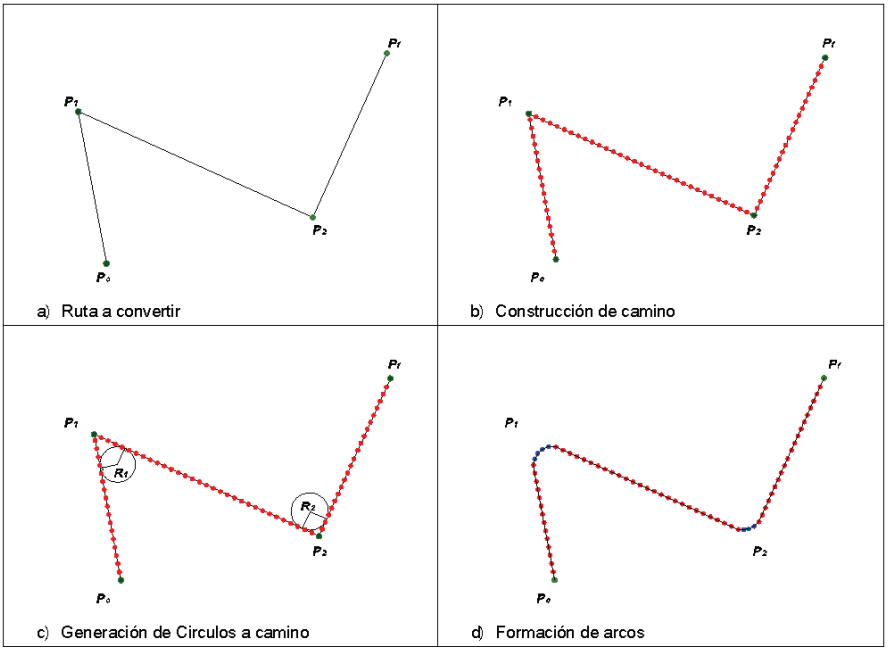


Figura 3.103 Proceso de construcción de caminos

Para ingresar el tiempo para obtener la trayectoria discreta se relaciona con el valor del parámetro (λ), donde el tiempo para alcanzar la configuración final es proporcional con el valor final de λ. Con esto se garantiza que cada intervalo de tiempo entre los posiciones sea constante, manteniendo por ende la rapidez también constante.

Tabla 3.4 Generación de trayectoria discreta

Dirección de Variable (A)	**Posición (x)**	**Posición (y)**	**Parámetro**
0	x_0	y_0	t_0
1	x_1	y_1	t_1
2	x_2	y_2	t_2
	$\vdots$	$\vdots$	$\vdots$
A_i	x_i	y_i	t_i
	$\vdots$	$\vdots$	$\vdots$
A_f	x_f	y_f	t_f

Donde:

- t_f: valor al cual alcanza la configuración final del camino

El diagrama de flujo de la Figura 3.103 representa el procedimiento para formar la trayectoria en el programa de control.

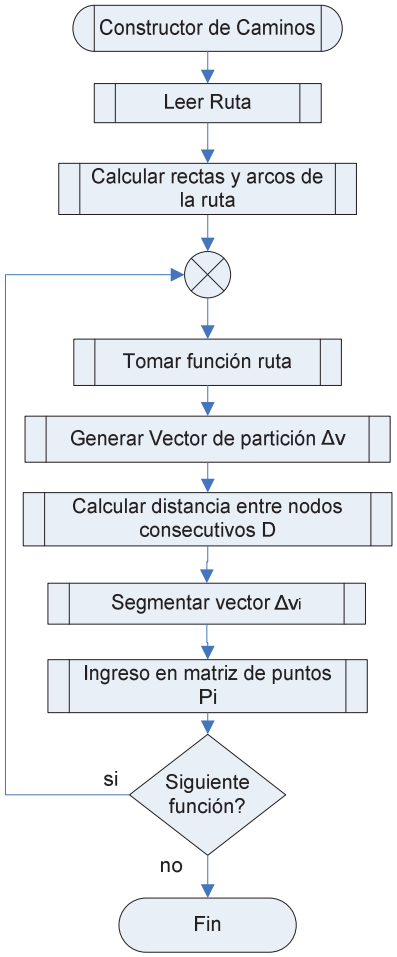


Figura 3.104 Diagrama de flujo constructor de Trayectoria

El diagrama de flujo de la Figura 3.104, indica que la rutina de construcción de trayectoria toma los nodos de la ruta y realizar la interpolación Spline con las funciones de rectas y los arcos de circunferencias. Luego el programa toma las funciones, las segmenta y discretiza en función del parámetro λ, las cuales son

almacenadas en un mapa de datos similar a la Tabla 3.3 que contiene las posiciones a una distancia constante (Figura 3.103). La formalización de la trayectoria se calcula el periodo de muestreo conociendo que la rapidez máxima que el robot puede ejecutar es $0.5m/s$

$$To = \frac{0.1m}{0.5m/s} = 0.2s$$

Con este valor del periodo de muestreo se puede determinar el tiempo mínimo para garantiza que el seguimiento sea correcto para el robot sin alterar la trayectoria seguida.

$$t_{\min Tray} = 0.2\lambda$$

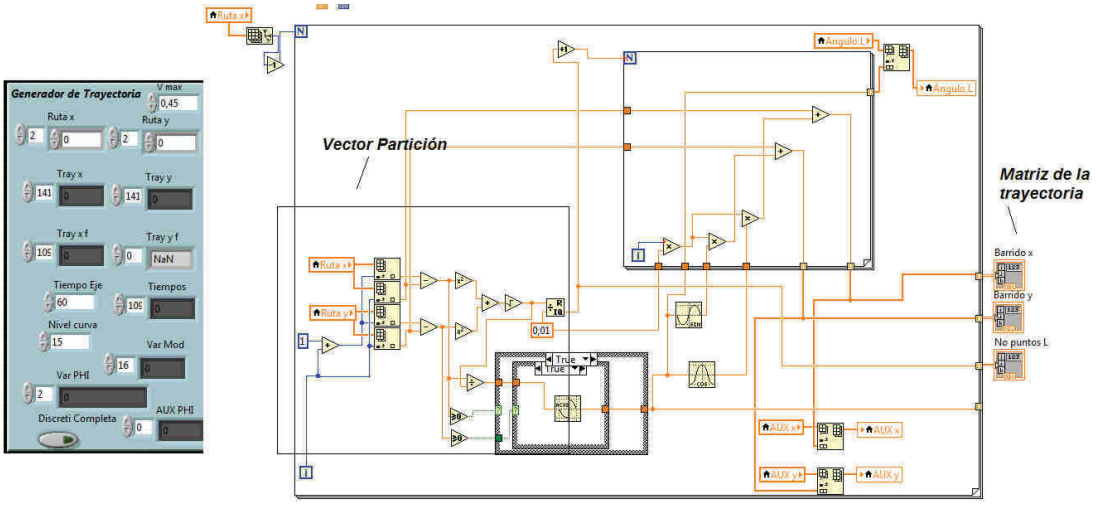


Figura 3.105 Rutina Constructor de Trayectoria

El diagrama presentado del la rutina del constructor de trayectoria (Figura 3.105), obedece a la discretización de las funciones calculadas de la ruta formado el vector de partición mantenido y agregando los puntos faltantes a una distancia constante, estos son guardados en una matriz cuyas direccionamiento es el parámetro λ

3.8.2 SISTEMA DE CONTROL DE TRAYECTORIA

Determinada la ecuación de la ley de control de trayectoria (Ecuación 3.60), y con los cálculos para construcción de la trayectoria con los requerimientos de velocidad del Robotino®, el siguiente paso para la programación del sistema de control de trayectoria es la esquematización del sistema en diagrama de bloques.

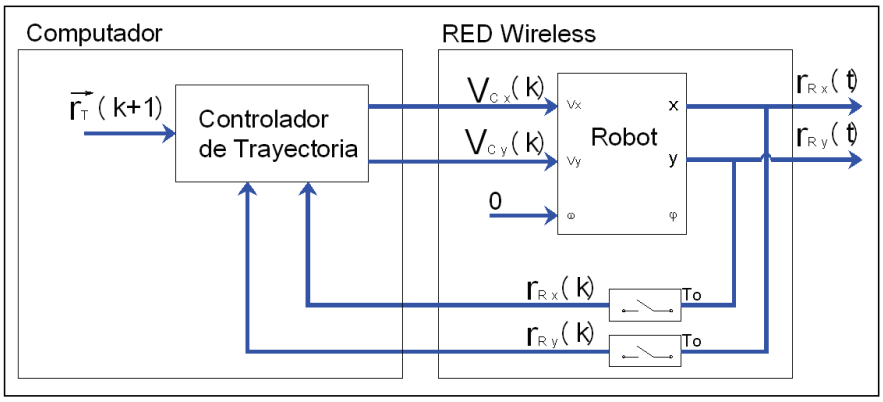


Figura 3.106 Diagrama de Bloques de sistema de control.

El diagrama de bloques de la Figura 3.106 indica que el sistema de control de trayectoria está compuesto por dos elementos, el robot y el HMI. El HMI a través de un computador con comunicación inalámbrica, se encarga del monitoreo de señales, el procesamiento, construcción de trayectoria, y la ejecución del la ley de control.

Del diagrama de bloques de la figura 3.106, y la ecuación de la ley de control (Ecuación 3.60), se conforma el sistema de control completo formado por el siguiente diagrama:

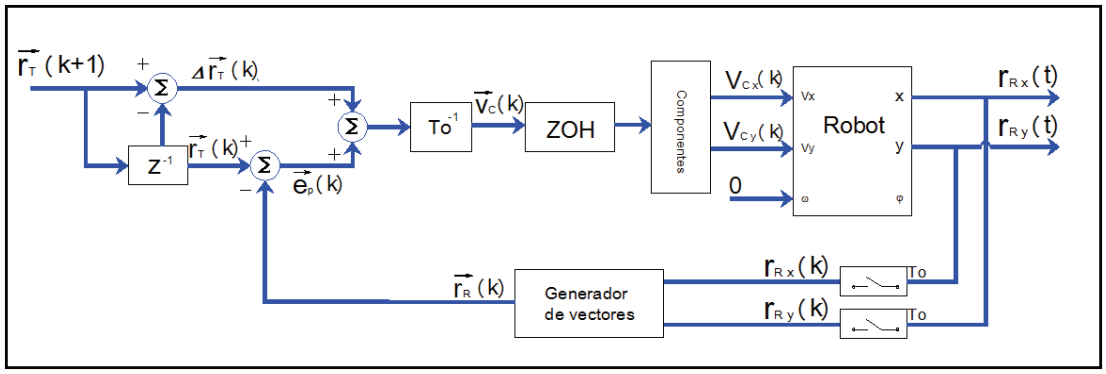


Figura 3.107 Sistema de control de Trayectoria

El diagrama de bloques de la Figura 3.107 presenta el esquema del sistema de control discreto, donde la variable de entrada del sistema es la trayectoria discreta, la cual utilizando el desplazamiento (z^{-1}), de la trayectoria para obtener el valor de la posición actual de la trayectoria $\vec{r}_{T\,(k)}$. Con el valor de la posición actual y de la siguiente se tiene el desplazamiento $\overrightarrow{\Delta r}_{T\,(k)}$. Por otro lado con la lectura de los valores de la posición del Robotino®, y la posición de la trayectoria se calcula el error de posición del robot completando la ley de control (ecuación 3.58) aplicando el periodo de muestreo. Utilizando un retenedor de orden cero se mantiene la velocidad hasta que se actualice el controlador.

El controlador procesa la información vectorialmente, y envía la velocidad de control en componentes rectangulares, que el robot ejecuta.

A partir del diagrama de bloques de la Figura 3.107 y del la construcción de la trayectoria se diseña el siguiente diagrama de flujo que conforma el programa de control que se incluye el HMI.

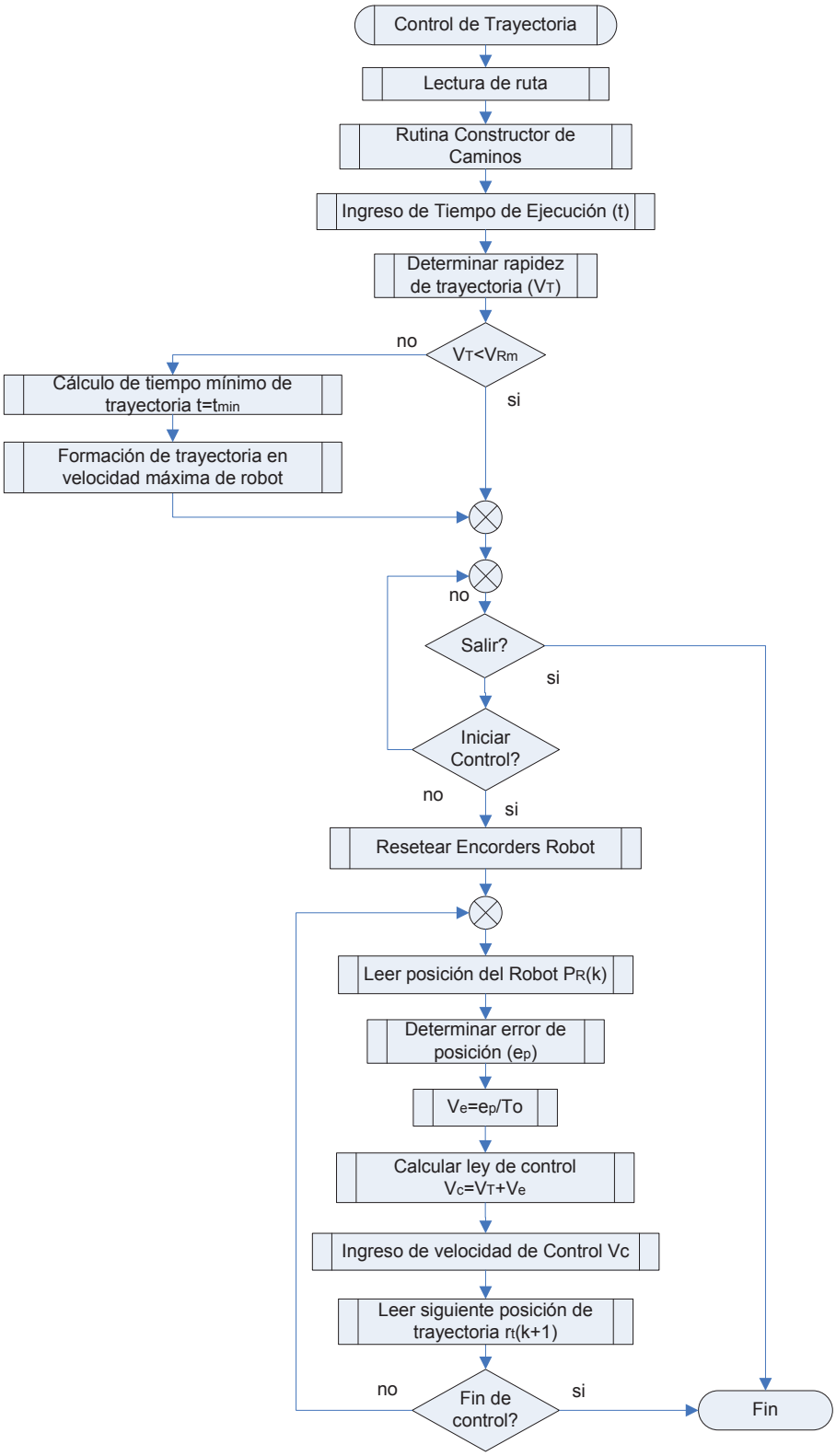


Figura 3.108 Diagrama de Flujo de control de Trayectoria

El diagrama de flujo de la Figura 3.108 conforma el procedimiento para completar el control de trayectoria a partir de la ruta formada. El programa primero toma la ruta y ejecutando la rutina constructor de trayectoria toma el valor del tiempo de ejecución y con la ruta calcula la trayectoria tomando en cuenta la velocidad máxima que el robot puede ejecutar. El programa evalúa si el tiempo ingresado es adecuado para ejecutar la trayectoria, en ese caso la trayectoria se ajusta a ese tiempo, caso contrario el programa ajusta la trayectoria a la velocidad máxima de ejecución del robot $0.5m/s$. Calculada la trayectoria, el programa espera el inicio del control.

Cuando se ejecuta el control de trayectoria en $t = 0s$, el programa lee la posición del robot y la transforma en un vector $\overrightarrow{\Delta \boldsymbol{r}}_{T\,(k)}$, al mismo tiempo del mapa que tiene las configuraciones de la trayectoria (Tabla 3.4), toma el programa la posición de la trayectoria $\overrightarrow{\Delta \boldsymbol{r}}_{T\,(k)}$, y la siguiente posición $\overrightarrow{\Delta \boldsymbol{r}}_{T\,(k+1)}$, para luego calcular la ley de control (ecuación 3.58) obteniendo la velocidad de control $\vec{\boldsymbol{V}}_{C\,(k)}$, la cual es procesada para obtener la componentes y enviarlas al robot.

Enviada la velocidad de control, el programa actualiza las variables de control y vuelve a ejecutar el control en $t = To$.

Para sincronizar el controlador al periodo de muestreo, el programa toma los valores de los tiempos de la trayectoria (Tabla 3.4), y con un contador de tiempo el programa determina los tiempos de actualización de la ley de control.

Con la lógica de control impuesta se ingresa al programa de control. Utilizando una interface para el control de trayectoria. La Figura 3.109 presenta la forma de ingreso de la trayectoria construida utilizando variables de tipo ARRAY, calcula la ley de control y envía a los bloques que controlan el robot. En la parte superior de la Figura 3.109 está un diagrama cuya función es saturar la velocidad de control aplicada al robot.

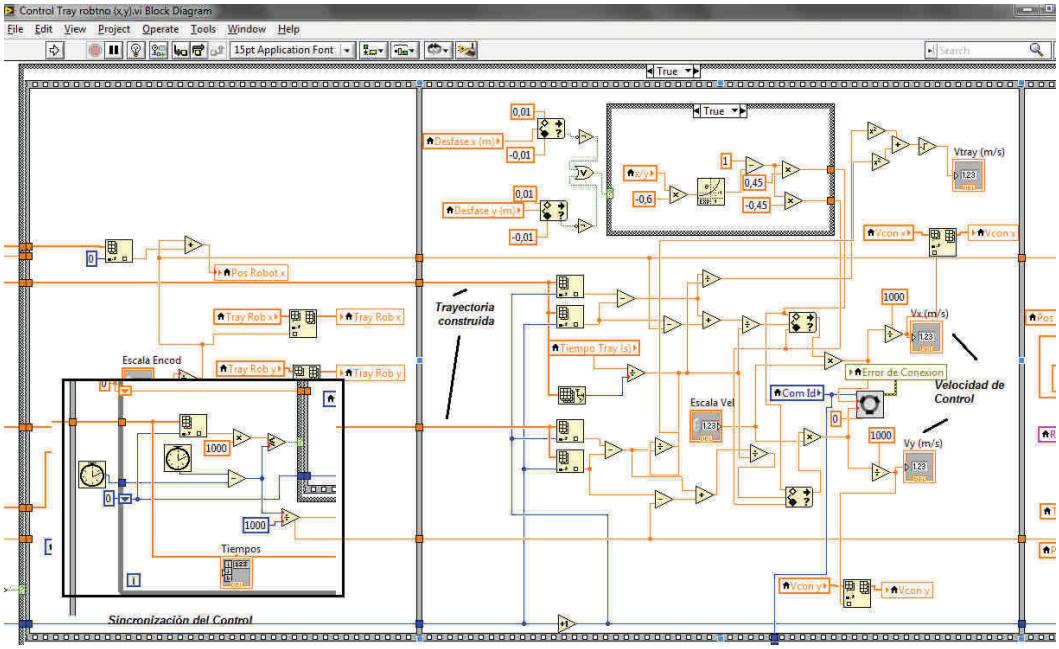


Figura 3.109 Diagrama de bloques de control de trayectoria

En la parte inferior de la Figura 3.109 se encuentra la rutina que sincroniza la ejecución del control, utilizando los tiempos de la trayectoria y con un contador que interrumpe la ejecución del HMI para ejecutar el control y actualizar los gráficos del HMI.

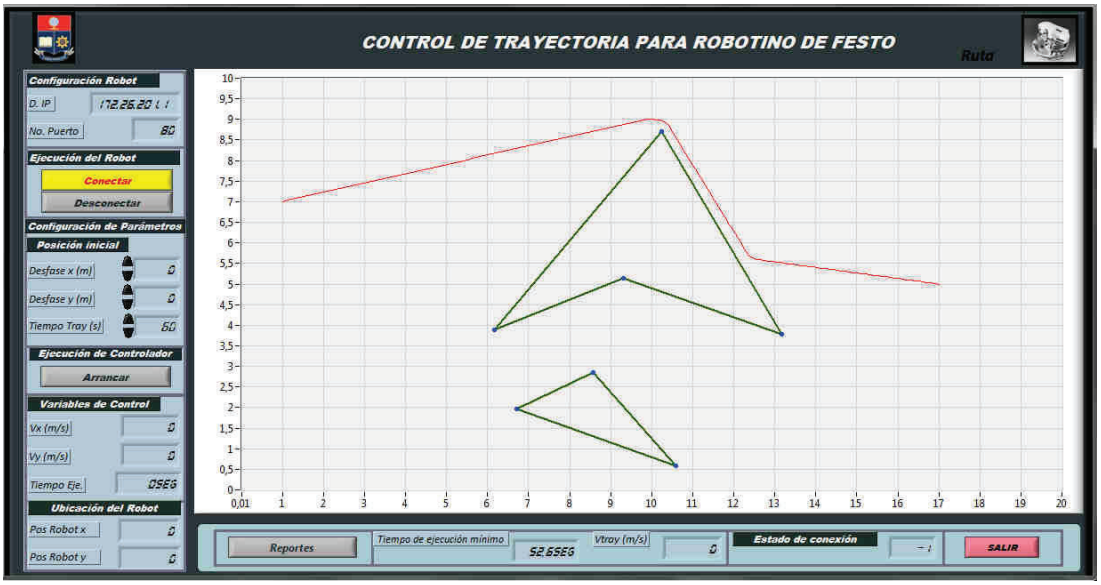


Figura 3.110 Panel Frontal de Control de trayectoria

El HMI de control de trayectoria de la Figura 3.110 cuenta con las funciones necesarias para controlar al Robotino®. Para el ingreso del tiempo de ejecución de trayectoria el programa limita al tiempo mínimo de ejecución siempre y cuando el tiempo ingresado es menor al mínimo garantizando la que velocidad máxima permitida no sea mayor a $0.5m/s$. Los movimientos ejecutados por el robot son presentados en tiempo real en el HMI y monitoreados en función del tiempo transcurrido del programa de control.

El panel frontal también cuenta con un sistema de reportes que indican la información de las variables de control incluyendo la velocidad de control, sus componentes el error de posición, trayectoria trazada. En el siguiente capítulo se realizaran las pruebas de control de trayectoria bajo varias circunstancias que revelan el funcionamiento del sistema de control.

CAPÍTULO 4

PRUEBAS Y RESULTADOS

En el presenta capítulo se comprobará el funcionamiento de los tres métodos planificadores, control de ruta y de trayectoria para el Robotino® descritos en el capítulo anterior bajo diferentes escenarios.

Para cumplir con lo expuesto, los siguientes ítems son los requerimientos a tomarse en cuenta para completar las pruebas.

- Tiempo de ejecución del método planificador para obtener una ruta entre las configuraciones.

- Distancia y quiebres de la trayectoria obtenida.

- Complejidad del mapa de entorno.

- Desempeño respecto al computador utilizado, de acuerdo con el tiempo de ejecución.

- Errores de posición por parte del Robotino® y distancia recorrida real.

- Análisis de la trayectoria ejecutada por el Robotino con sus respectivos controladores y el tiempo de ejecución en el caso de control de ruta.

- Desempeño del simulador ante perturbaciones de velocidad aplicada.

4.1 ANÁLISIS DE LOS MÉTODOS PLANIFICADORES

Analizar el comportamiento de los métodos planificadores, tiene como objetivo determinar cual resulta ser el más eficaz respecto a los otros sobre un mismo

escenario, tanto en rendimiento para encontrar la ruta entre las configuraciones, como en la ejecución de la misma al Robotino® de Festo.

Para analizar los métodos planificadores, se toma tres escenarios definidos considerando los requerimientos anteriormente mencionados, utilizando un computador para todas las pruebas que tiene las siguientes prestaciones:

- Procesador Intel CORE 2 DUO $T5750$ de $2.0GHz$.
- Disco duro de $500GB$.
- Memoria RAM de $3GB$.
- Windows 7 Home Premium 32bits.

4.1.1 PRIMERA PRUEBA, MAPA DE ENTORNO ESTILO LIBRE

Para la primera prueba de simulación de los métodos planificadores se toma el siguiente mapa de entorno graficado en forma libre.

Figura 4.1 Mapa de entorno para primera prueba

Del mapa de entorno (Figura 4.1), todos los obstáculos están distribuidos a lo largo del plano de trabajo del HMI, sin un orden establecido, los vértices y secuencia de los obstáculos se presentan en la Tabla 4.1:

Tabla 4.1 Vértices de obstáculos de mapa de entorno para la primera prueba

	Polígono 1		**Polígono 2**		**Polígono 3**		**Polígono 4**		**Polígono 5**	
No Vértice	**Vértice (x) (m)**	**Vértice (y) (m)**	**Vértice (x) (m)**	**Vértice (y) (m)**	**Vértice (x) (m)**	**Vértice (y) (m)**	**Vértice (x) (m)**	**Vértice (y) (m)**	**Vértice (x) (m)**	**Vértice (y) (m)**
1	1.72	7.15	3.87	3.07	10.83	7.43	12.00	3.00	16.00	7.00
2	6.67	8.27	8.77	1.32	11.48	4.19	14.64	1.04	18.55	8.98
3	6.30	5.75	8.94	2.70	12.93	5.53	17.81	2.51	11.41	9.68
4	5.56	7.45	7.32	2.85	13.93	4.59	15.82	3.13	15.14	8.86
5			7.60	4.41	13.11	7.30				

Para los tres métodos planificadores, se toman las siguientes configuraciones para calcular la ruta:

$$P_0 : (2\,, 3)$$

$$P_f : (15\,, 7)$$

Todas las pruebas a realizarse se toma las mismas dimensiones del entorno de trabajo ($20x10m$).

4.1.1.1 Primera prueba aplicada a Grafo de Visibilidad

Con el mapa de entorno y las configuraciones determinadas, se calcula la ruta con el método Grafo de Visibilidad, pulsando el botón correspondiente en el HMI. El programa calcula la ruta entre las configuraciones, obteniendo los siguientes resultados:

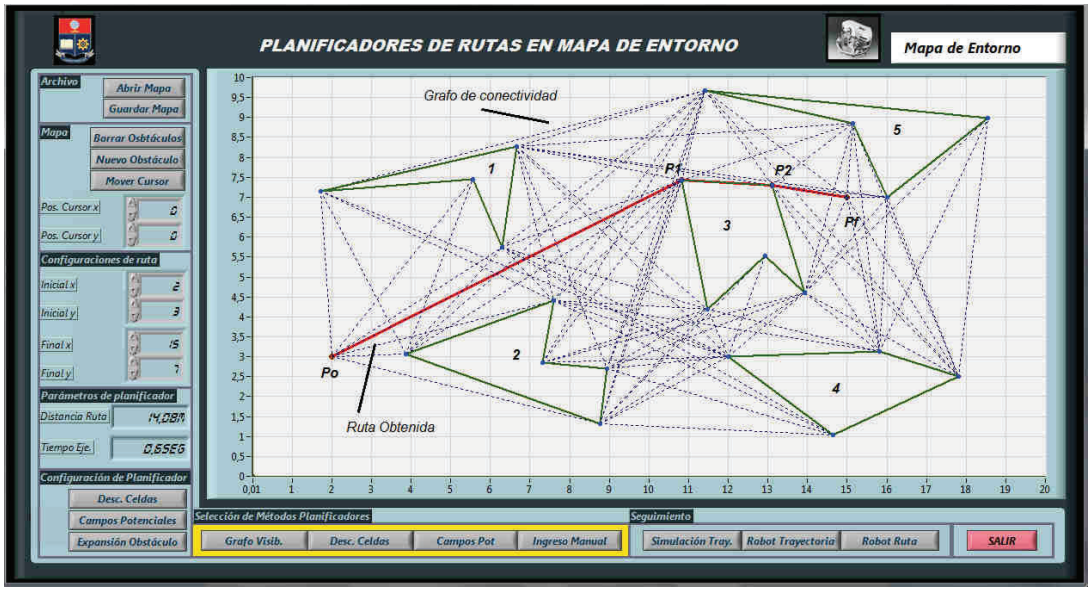


Figura 4.2 Respuesta por método Grafo de Visibilidad

La Figura 4.2 presenta en detalle la construcción del grafo de conectividad, la ruta obtenida por el método Grafo de Visibilidad, los indicadores numéricos presentan la información del tiempo de ejecución y la distancia total entre las posiciones inicial P_0 y final P_f.

Respecto al grafo de conectividad, el algoritmo generó 85 enlaces válidos para completarlo incluyendo los enlaces de las configuraciones inicial P_0 y final P_f.

La ruta obtenida tiene 4 nodos como indica la Tabla 4.2, dos de los cuales son los vértices del tercer obstáculo y los demás son las configuraciones.

Tabla 4.2 Ruta Obtenida por Grafo de Visibilidad para prueba 1

Nodos	P_0	P_1	P_2	P_f
Nodo x (m)	2.00	10.83	13.11	15.00
Nodo y (m)	3.00	7.43	7.30	7.00

La Tabla 4.2, además de presentar los nodos de la ruta, también indica la secuencia de la misma.

Para comprobar la distancia recorrida obtenida en el HMI se calcula las distancias parciales entre los nodos con las siguientes expresiones:

$$D_i = \sqrt{(P_{ix} - P_{(i-1)x})^2 + (P_{iy} - P_{(i-1)y})^2} \quad (4.1)$$

$$D_1 = \sqrt{(10.83 - 2)^2 + (7.43 - 3)^2} = 9.88m$$

$$D_2 = \sqrt{(13.11 - 10.83)^2 + (7.3 - 7.43)^2} = 2.28m$$

$$D_3 = \sqrt{(15 - 13.11)^2 + (7 - 7.3)^2} = 1.91m$$

$$D = \sum_{D=0}^{i} D_i \quad (4.2)$$

Entonces la distancia neta de la ruta es:

$$D = 9.88m + 2.28m + 1.91m$$

$$D = 14.07m$$

En la Figura 4.2 el indicador de distancia presenta $14.07m$, ratificando la distancia calculada automáticamente.

El tiempo de ejecución del método planificador es $0.6s$ por lo que el método planificador es eficaz para este mapa de entorno.

4.1.1.2 Primera prueba aplicada a Descomposición del Celdas

Aplicando el método Descomposición de Celdas sobre el mapa de entorno de la Figura 4.1 con las mismas configuraciones, se obtienen los siguientes resultados:

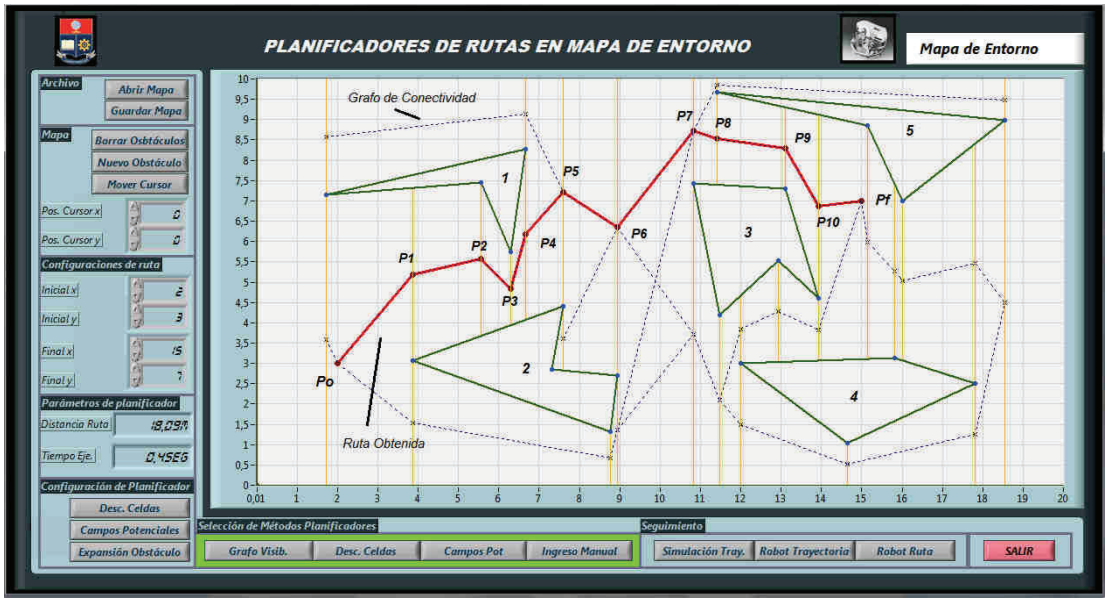


Figura 4.3 Respuesta por método Descomposición de Celdas

El HMI de la Figura 4.3, presenta los pasos para obtener el grafo de conectividad como los nodos formado por las rectas verticales que nacen en los vértices de los obstáculos, los enlaces válidos entre celdas adyacentes, la ruta entre las configuraciones.

El grafo de conectividad generado tiene 34 enlaces válidos que rodea todos los obstáculos del mapa de entorno, y los nodos de la ruta obtenida por el método son:

Tabla 4.3 Ruta obtenida por Descomposición de Celdas para prueba 1

Nodos	Nodo x (m)	Nodo y (m)	Nodos	Nodo x (m)	Nodo y (m)
P_0	2.00	3.00	P_6	8.94	6.35
P_1	3.87	5.19	P_7	10.83	8.71
P_2	5.56	5.56	P_8	11.41	8.54
P_3	6.30	4.85	P_9	13.11	8.30
P_4	6.67	6.17	P_{10}	13.93	6.86
P_5	7.60	7.21	P_f	15.00	7.00

La Tabla 4.3 presenta 12 nodos para formar la ruta incluyendo las configuraciones inicial y final, los cuales son más que del método Grafo de Visibilidad, pero la ruta está alejada de los obstáculos.

La distancia recorrida por la ruta entre las configuraciones, es de $18.09m$ calculada con las fórmulas 4.1 y 4.2 respectivamente, el tiempo de ejecución del método es de $0.4s$, siendo menor al del método de Grafo de Visibilidad.

4.1.1.3 Primera prueba aplicada a Campos Potenciales

Al método por Campos Potenciales se analiza en el mismo mapa de entorno con las mismas configuraciones (Figura 4.1) obteniendo los siguientes resultados:

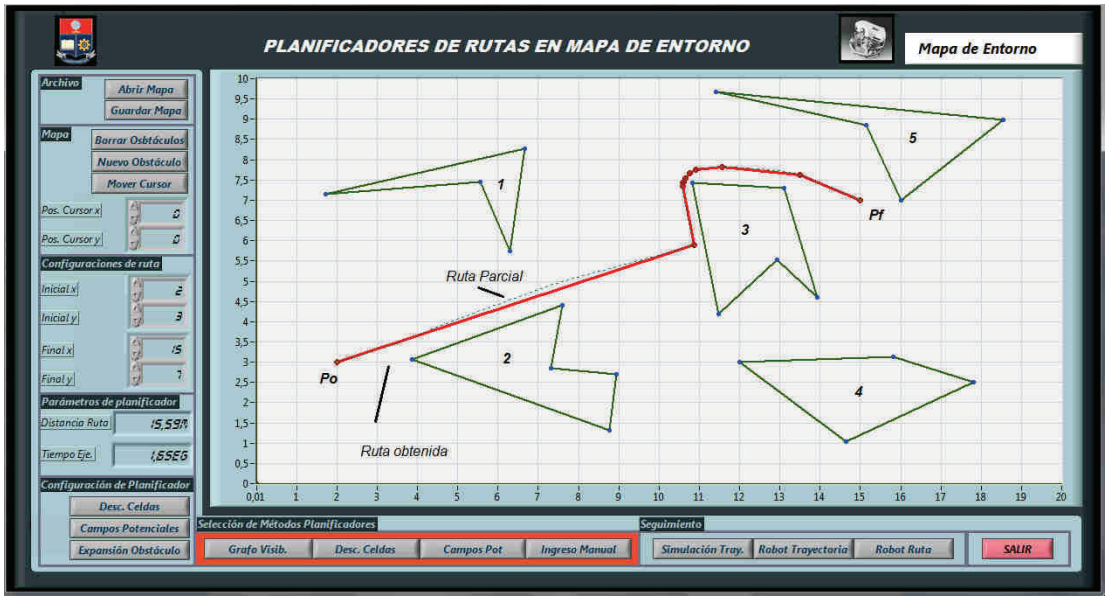


Figura 4.4 Respuesta por método Campos Potenciales

La Figura 4.4 presenta el camino real del método tiene 149 nodos entre las configuraciones, pero estos se reducen a los presentados en la Tabla 4.4 cuando la ruta pasa por el proceso de filtrado.

Tabla 4.4 Ruta obtenida por Campos Potenciales para prueba 1

Nodos	Nodo x (m)	Nodo y (m)	Nodos	Nodo x (m)	Nodo y (m)
P_0	2.00	3.00	P_5	10.70	7.62
P_1	10.86	5.89	P_6	11.03	7.79
P_2	10.58	7.34	P_7	13.40	7.65
P_3	10.59	7.43	P_f	15.00	7.00
P_4	10.62	7.44			

Los nueve nodos de la ruta definitiva conservan casi toda la información, donde la mayoría estos rodean al obstáculo 3 para alcanzar la configuración final.

La distancia recorrida entre las configuraciones es $15.58m$, y el tiempo de ejecución del método $1.6s$.

4.1.1.4 Análisis comparativo para la primera prueba

Las variables más importantes recogidas de los tres métodos planificadores, se resumen en la Tabla 4.5:

Tabla 4.5 Parámetros relevantes de métodos planificadores para primera prueba

	Grafo de Visibilidad	Descomposición de Celdas	Campos Potenciales
Distancia Recorrida (m)	14.07	18.09	15.58
Tiempo de Procesamiento (s)	0.6	0.4	1.6
Número de quiebres	2	9	7
Nodos de Grafo/Iteraciones	85	34	149

Considerando la ruta entre las configuraciones inicial P_0 y final P_f de la Tabla 4.5, la respuesta dada por Grafo de Visibilidad, presenta el menor número de quiebres y también la menor distancia de la ruta formada, pero por la naturaleza del

método no se aplica al control de rutas y de trayectoria del Robotino® debido a sus dimensiones.

El método Descomposición de Celdas presenta la distancia más larga y el mayor número de quiebres para evitar los obstáculos pero al ser un mapa de entorno que no tiene caminos estrechos, el camino está muy alejado de los obstáculos haciendo ineficiente el método para aplicar robot no Holónomos. Esta condición se debe a los enlaces para formar el grafo de conectividad se realizan entre los nodos vecinos.

El Método de Campos Potenciales, presenta la respuesta intermedia entre las configuraciones. Los primeros tramos del camino, el método tiende a dirigirse hacia la configuración final P_f en línea recta hasta que se encuentra con el obstáculo 3. El algoritmo lo evita utilizando la técnica por seguimiento de pared, evitando al obstáculo, y obteniendo una ruta similar a la del Grafo de Visibilidad, pero no es necesario realizar adaptaciones para ejecutarla trayectoria al robot, ya que se encuentra calculada la distancia entre la trayectoria y obstáculo calculadas con las expresiones matemáticas de las fuerzas de atracción y repulsión.

Desde el punto de vista del tiempo de ejecución, el método de Campos potenciales es el más lento de los tres, sin embargo tomado valores absolutos, el tiempo de procesamiento de $1.6s$, es un valor bajo para obtener una respuesta ante el problema de planificación.

Los tiempos de los otros planificadores son muy similares, aunque la diferencia radica en el número de enlaces del los grafos de conectividad, siempre mayor los enlaces para el método Grafo de Visibilidad.

De acuerdo con lo expresado a nivel de búsqueda de rutas el método que presenta más beneficios es el Grafo de Visibilidad, tanto en tiempo como en ruta obtenida, aunque no sea aplicable para el Robotino®, aplicando el criterio de expansión de obstáculos se puede tener resultados similares, con la trayectoria más simple.

4.1.2 SEGUNDA PRUEBA, COMPLEJIDAD DE OBSTÁCULOS

Para esta prueba se utiliza los mismos criterios de análisis que para la anterior, para lo cual se parte del siguiente mapa de entorno.

Figura 4.5 Mapa de entorno para segunda prueba

En el mapa de entorno de la Figura 4.5 a diferencia del mapa de la Figura 4.1, se aumenta la complejidad de los obstáculos que intervienen y se estrecha los caminos libres de colisión.

Tabla 4.6 Obstáculos de mapa de entorno para la segunda prueba

Polígono 1		Polígono 2		Polígono 3		Polígono 4	
Vértice x (m)	Vértice y (m)	Vértice x (m)	Vértice y (m)	Vértice x (m)	Vértice y (m)	Vértice x (m)	Vértice y (m)
3.44	8.31	11.35	8.60	11.65	4.97	6.84	1.08
5.80	6.54	13.84	7.04	18.50	6.20	14.08	1.51
4.84	4.17	17.61	8.08	19.35	4.17	15.47	0.32
6.47	2.33	15.47	9.31	17.81	2.66	7.25	0.24
8.90	2.92	14.45	7.80	16.40	4.47		
7.34	5.01	13.67	8.83	14.99	2.55		
8.92	7.56			12.35	3.24		
7.21	9.29						
6.04	7.80						

Las configuraciones inicial y final para la determinación de una ruta para los tres métodos planificadores son:

$$P_0 : (2\,,5)$$

$$P_f : (17\,,2)$$

4.1.2.1 Segunda prueba aplicada a Grafo de Visibilidad

Aplicando el método de Grafo de visibilidad al mapa de la Figura 4.5 con las configuraciones correspondientes, se tienen los siguientes resultados:

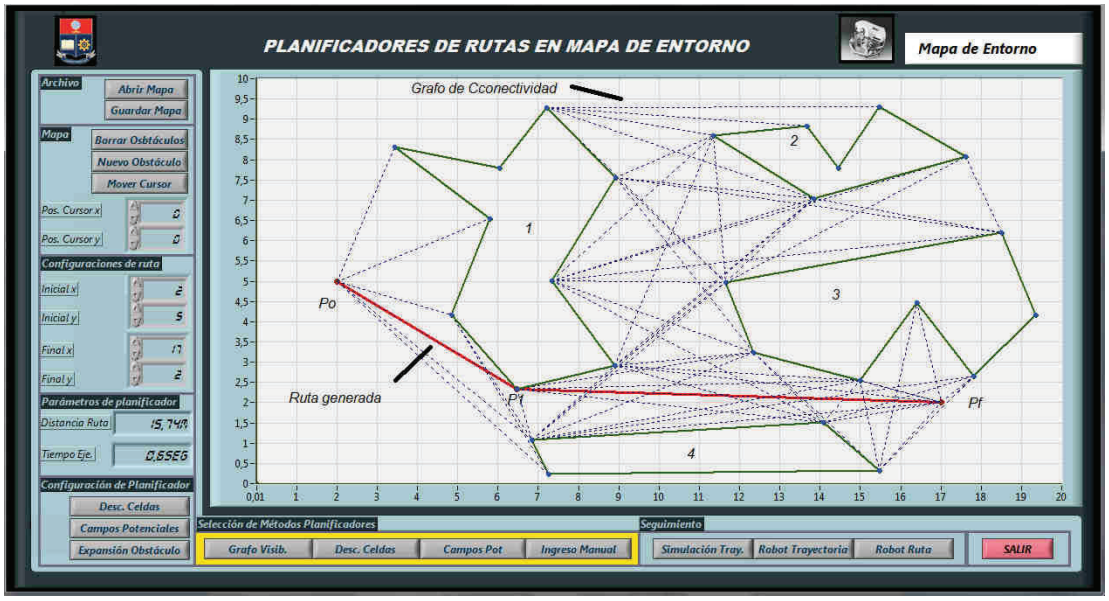


Figura 4.6 Respuesta por método Grafo de Visibilidad

La Figura 4.6 presenta el grafo de conectividad construido por el método planificador, el cual tiene 63 enlaces, y la ruta formada entre las configuraciones, contiene tres nodos incluyendo las configuraciones, generando un quiebre en la ruta cuyas coordenadas se presentan en la Tabla 4.7.

Tabla 4.7 Ruta obtenida por método Grafo de Visibilidad para prueba 2

Nodos	P_0	P_1	P_f
Nodo x (m)	2	6.47	17
Nodo y (m)	5	2.33	2

La Distancia de la ruta entre las configuraciones es $15.74m$, y el tiempo de procesamiento del método planificador es $0.6s$

4.1.2.2 Segunda prueba aplicada a Descomposición de Celdas

Utilizando el mismo mapa de la segunda prueba para el método Descomposición de Celdas, se obtienen los siguientes resultados:

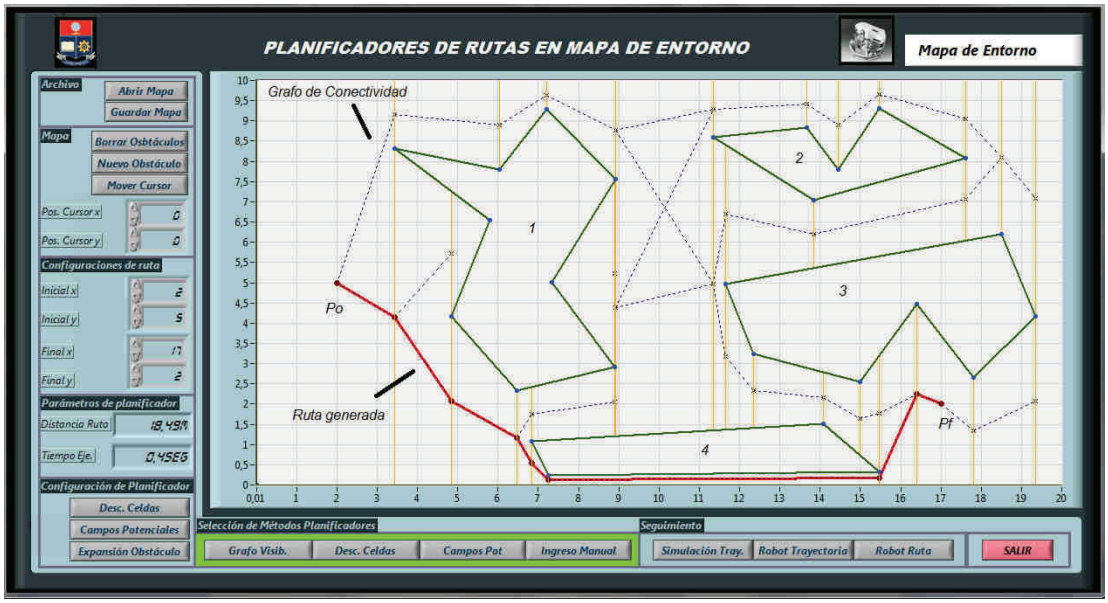


Figura 4.7 Respuesta por método Descomposición de Celdas

El HMI de la Figura 4.7, arroja un grafo de conectividad de 39 enlaces, pero a diferencia del grafo obtenido en la primera prueba, este grafo no encierra totalmente a los obstáculos por lo que la ruta del método es diferente a la

presentada en el método Grafo de Visibilidad, esta ruta tiene nueve nodos incluyendo las configuraciones, los cuales se desglosan en la Tabla 4.8.

Tabla 4.8 Ruta obtenida por Descomposición de celdas para la segunda prueba

Nodos	Nodo x (m)	Nodo y (m)	Nodos	Nodo x (m)	Nodo y (m)
P_0	2.00	5.00	P_5	7.25	0.12
P_1	3.44	4.16	P_6	15.47	0.16
P_2	4.84	2.08	P_7	16.40	2.24
P_3	6.47	1.17	P_f	17.00	2.00
P_4	6.84	0.54			

De los indicadores de distancia de la ruta y tiempo de procesamiento del método, se obtuvieron $18.49m$ y $0.4s$ respectivamente.

4.1.2.3 Segunda prueba aplicada a Campos Potenciales

Aplicando el método Campos Potenciales sobre el mapa de entorno (Figura 4.5), los resultados son los siguientes:

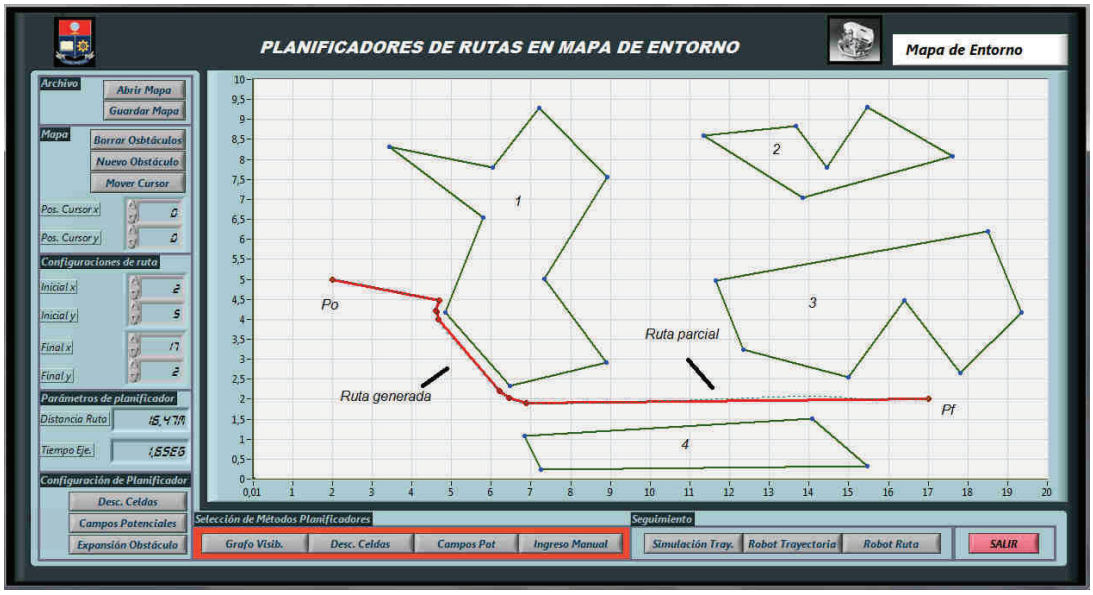


Figura 4.8 Respuesta por método Campos Potenciales

La ruta calculada por el método planificador (línea segmentada), tiene 162 nodos y por ende el mismo número de iteraciones para completarla. Utilizando el filtro de eliminación de nodos, la ruta definitiva reduce sus nodos y quiebres a ocho y seis respectivamente, la Tabla 4.9 presenta las coordenadas de cada punto.

Tabla 4.9 Ruta obtenida por Campos Potenciales para la segunda prueba

Nodos	Nodo x (m)	Nodo y (m)	Nodos	Nodo x (m)	Nodo y (m)
P_0	2.00	5.00	P_4	4.66	3.99
P_1	4.68	4.48	P_5	6.35	2.08
P_2	4.61	4.20	P_6	6.88	1.91
P_3	4.63	4.18	P_f	17	2.00

El tiempo de procesamiento presentado por el HMI (Figura 4.8), es $1.6s$, y la distancia del camino entre configuraciones es $16.47m$.

4.1.2.4 Análisis comparativo para la segunda prueba

La información recogida de los métodos planificadores, se resume en la Tabla 4.10 para el análisis comparativo.

Tabla 4.10 Parámetros de los métodos planificadores para segunda prueba

	Grafo de Visibilidad	Descomposición de Celdas	Campos Potenciales
Distancia Recorrida (m)	15.74	18.49	16.47
Tiempo de Procesamiento (s)	0.6	0.4	1.6
Número de quiebres	1	7	6
Nodos de Grafo/Iteraciones	63	35	162

De igual manera que la primera prueba se obtienen resultados similares entre los tres métodos planificadores. La distancia de la ruta obtenida por el método Descomposición de Celdas es un 18% mayor que el Grafo de visibilidad y 12% respecto al método de Campos Potenciales lo cual no es un incremento

significativo, considerando que para fines prácticos hay que aplicar el criterio del entorno expandido al método de Grafo de Visibilidad para utilizar el Robotino®.

El principal inconveniente del método de Descomposición de Celdas es el número de quiebres aumentando el procesamiento posterior para la construcción de la trayectoria y la ejecución al Robotino®.

Respecto a los tiempos de procesamiento, el método de Campos Potenciales tiene el tiempo más alto, debido al número de iteraciones ejecutadas por el programa para calcular la ruta, pero la respuesta es la más suave de las tres, rodea uniformemente los obstáculos que se encuentran entre las configuraciones y se asemeja a la respuesta de Grafo de Visibilidad.

El método por Grafo de Visibilidad presenta la trayectoria más simple, y además el tiempo de ejecución es igual de bueno (menor a $1s$) a pesar que el grafo de conectividad calculado tiene prácticamente el doble de enlaces, que el obtenido por Descomposición de Celdas.

Tomado en cuenta el tiempo de procesamiento invertido encontrando la ruta entre las configuraciones, respecto a la primera prueba, el incremento de la complejidad de los obstáculos no influye en el rendimiento de los métodos planificadores programados.

4.1.3 TERCERA PRUEBA, ENTORNO REAL

Para la tercera prueba aplicada a los métodos planificadores, utiliza un entorno real de trabajo, con el objetivo de analizar su comportamiento y capacidades ante un sistema con alta complejidad y cantidad de obstáculos. Para lo cual se utiliza como punto de partida el entorno del Laboratorio de Control y Sistemas (Anexo C), que tenga las dimensiones dentro de los límites del entorno de trabajo del HMI.

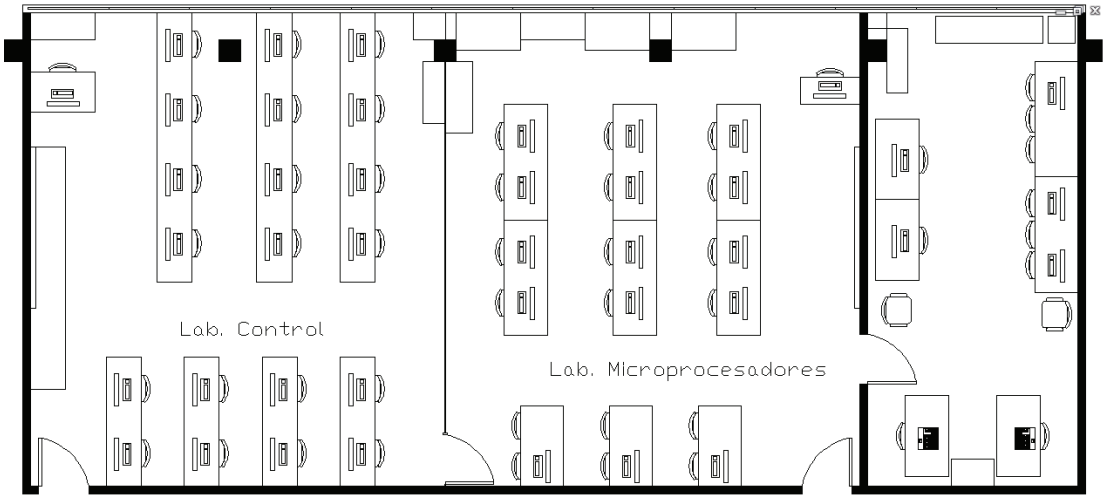


Figura 4.9 Plano de Laboratorio de Control y Sistemas

El plano del Laboratorio (Figura 4.9), consta de 3 cuatros individuales separados por puertas de dimensiones estándar ($90cm$). Cada cuarto tiene escritorios y estantes de dimensiones especificadas, y distribución según indica el plano a escala (Lámina C01 Anexo C).

Para ingresar el plano al espacio de trabajo del HMI, primero se considera a todos los objetos como obstáculos poligonales que tengan las dimensiones y ubicaciones correctas. Las coordenadas de todos los objetos se determinaron tomando las medidas localmente (Lámina C02 Anexo C), y las dimensiones del laboratorio se las tomó de la misma manera.

La Figura 4.10, representa al Laboratorio como un mapa de entorno con obstáculos rectangulares, tomando las coordenadas de los vértices (Tabla C.1).

Las dimensiones del Laboratorio completo coinciden con el entorno de trabajo del HMI, por lo que no existen inconvenientes para aplicar los métodos planificadores.

Figura 4.10 Representación en HMI de Laboratorio

Para la prueba se toman las siguientes coordenadas para que representen a las configuraciones:

$$P_0 : (2\ ,4)$$

$$P_f : (15\ ,1)$$

4.1.3.1 Tercera prueba aplicada a Grafo de Visibilidad

Del mapa de entorno dibujado en el HMI (Figura 4.10), la representación de los obstáculos se reducen a Figuras geométricas simples de 6 lados, para construir los escritorios colocados en las paredes y rectángulos para los que representan los escritorios aislados.

Construido el mapa de entorno y colocadas las configuraciones, se procede a la ejecución del método Grafo de Visibilidad, obteniendo los siguientes resultados:

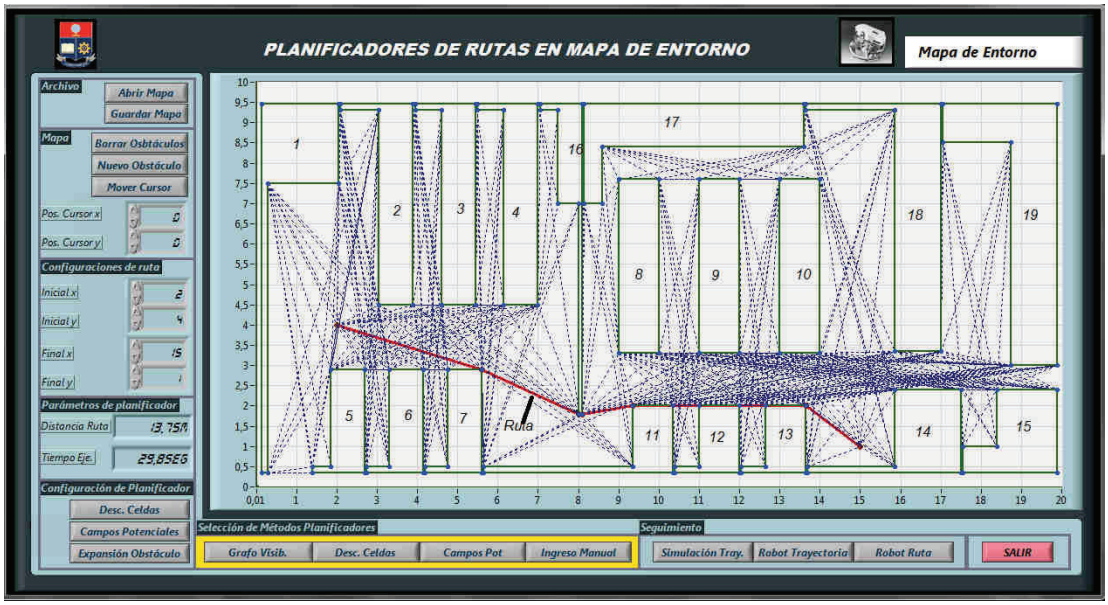


Figura 4.11 Respuesta por método Grafo de Visibilidad

El HMI (Figura 4.11), presenta el grafo de conectividad que tiene 714 enlaces válidos para terminarlo, y la mayor densidad de estos enlaces se encuentran sobre los obstáculos 8, 9, 10, 11, 12, y 13.

Tabla 4.11 Ruta obtenida por Grafo de Visibilidad en entorno real

Nodos	P_0	P_1	P_2	P_3	P_4	P_5	P_f
Nodo x (m)	2	5.6	8	8.1	9.35	13.65	15
Nodo y (m)	4	2.9	1.8	1.8	2	2	1

La ruta obtenida para el método planificador tiene 7 nodos, 5 quiebres (Tabla 4.11) y la distancia total es $13.75m$.

El indicador de tiempo indica que el método planificador tardó $29.8s$ para ejecutar el algoritmo, y encontrar la ruta entre las configuraciones.

4.1.3.2 Tercera prueba aplicada a Descomposición de Celdas

Modificando el mapa de entorno del Laboratorio, agrupando los obstáculos simples en dos más complejos que contiene la parte superior e inferior del entorno (Figura C.2), se calcula la ruta aplicando el método de Descomposición de Celdas obteniendo los siguientes Resultados:

Figura 4.12 Respuesta por método Descomposición de Celdas

La respuesta presentada por el HMI (Figura 4.12), indica que el grafo de conectividad tiene 58 enlaces en total, 12 veces menor al obtenido por el método Grafo de Visibilidad. Considerando que es un mapa de entorno cerrado solamente existe un enlace adyacente entre las celdas sin atajos, y por la distribución de los obstáculos que no están alineados se producen los quiebres bruscos obteniéndose una ruta irregular cuya distancia es $47.82m$.

Tabla 4.12 Ruta obtenida por Descomposición de Celdas en entorno real

Nodo	Nodo x (m)	Nodo y (m)	Nodo	Nodo x (m)	Nodo y (m)
P_0	2	4	P_{14}	8	1
P_1	2.05	5.2	P_{15}	81	1
P_2	2.7	6.1	P_{16}	8,6	3.75
P_3	3.05	2.5	P_{17}	9	1.68
P_4	3.3	3.7	P_{18}	9.35	2.42
P_5	3.9	3.7	P_{19}	10	2.42
P_6	4.15	6.1	P_{20}	10.35	5.2
P_7	4.6	2.5	P_{21}	11	7.78
P_8	4.75	3.7	P_{22}	12	7.78
P_9	5.45	3.7	P_{23}	12.65	5.2
P_{10}	5.6	6.1	P_{24}	13	2.42
P_{11}	6.15	2.5	P_{25}	13.65	2.42
P_{12}	7	2.5	P_{26}	14	1.68
P_{13}	7.5	3.75	P_f	15	1

Al construirse solo enlaces adyacentes no existen atajos para la ruta final el número de nodos es alto, 28 incluyendo las configuraciones inicial y final, y los quiebres son 26 siendo la mayoría inútiles. El tiempo de ejecución del método es $1.2s$.

4.1.3.3 Tercera prueba aplicada a Descomposición de Celdas Modificado

Aplicando el criterio de enlaces entre celdas adyacentes, la ruta obtenida presenta una respuesta ineficiente ante este mapa de entorno aumentando la distancia entre las configuraciones, se aplica el algoritmo que agrega sub grafos de conectividad a las celdas adyacentes para eliminar los quiebres innecesarios y aplicarlos en el mapa de entorno real, obteniendo la siguiente respuesta representada en la Figura 4.13:

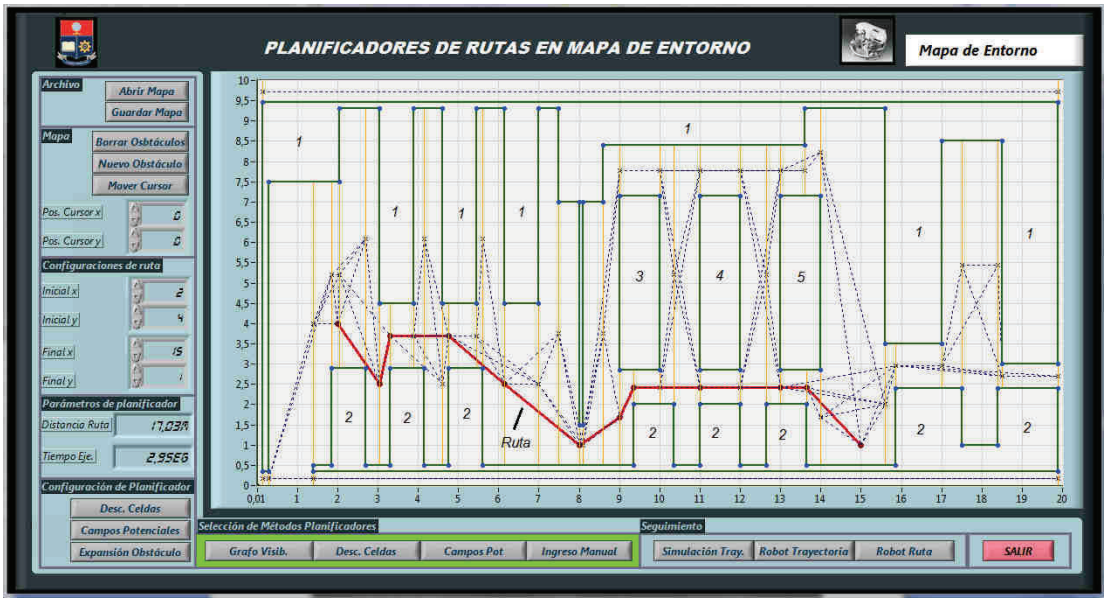


Figura 4.13 Respuesta por método Descomposición de Celdas Modificado

La respuesta del método planificador presenta que el grafo de conectividad forma enlaces adicionales entre nodos vecinos no adyacentes, formando subgrafos locales. Los enlaces totales del grafo de conectividad suman 128, duplicando el número de enlaces respecto al método de Descomposición de Celdas original, pero disminuyendo considerablemente el número quiebres y nodos de la ruta a 8 y 10 respectivamente presentados en la Tabla 4.13.

Tabla 4.13 Ruta obtenida por Descomposición de Celdas Modificado en entorno real

Nodo	Nodo x (m)	Nodo y (m)	Nodo	Nodo x (m)	Nodo y (m)
P_0	2.00	4.00	P_5	8.00	1.00
P_1	3.05	2.50	P_6	9.00	1.90
P_2	3.30	3.70	P_7	11.00	2.65
P_3	4.75	3.70	P_8	13.00	2.65
P_4	6.15	2.50	P_f	15.00	1.00

La distancia de la ruta presentada también se reduce a $17.03m$, y el tiempo de procesamiento es $2.8s$.

4.1.3.4 Tercera prueba aplicada a Campos Potenciales

Para la ejecución del método de Campos Potenciales, el mapa de entorno se vuelve a modificar de tal manera que tenga el menor número posible de obstáculos.

El mapa de entorno dibujado (Figura C.3) tiene cuatro obstáculos, tres de los cuales son rectángulos del cuarto central y un obstáculo cerrado con 74 vértices en secuencia conteniendo las paredes y los escritorios. Para mantener el entorno de acuerdo con el plano las paredes dibujadas en el HMI tienen el mismo grosor para obtener la fuerza de repulsión adecuada en función del entorno.

Con estas consideraciones se ejecuta el método Campos potenciales sobre las configuraciones obteniendo los siguientes resultados presentados en la Figura 4.14:

Figura 4.14 Respuesta de método Campos Potenciales

La respuesta del método planificador (Figura 4.14) indica que la ruta obtenida tiene 168 nodos, de los cuales se reducen a 14 con el proceso de filtrado de ruta.

Visualmente este método es el menos invasivo para apreciar la respuesta. La Tabla 4.14 presenta los valores de las coordenadas de los nodos.

Tabla 4.14 Ruta obtenida por Campos Potenciales en entorno real

Nodo	Nodo x (m)	Nodo y (m)	Nodo	Nodo x (m)	Nodo y (m)
P_0	2.00	4.00	P_8	9.09	1.71
P_1	7.68	2.82	P_9	9.15	2.10
P_2	7.70	2.80	P_{10}	9.58	2.36
P_3	7.75	1.46	P_{11}	12.28	2.31
P_4	7.90	1.26	P_{12}	12.55	2.23
P_5	8.20	1.12	P_{13}	13.74	2.28
P_6	9.08	1.12	P_{14}	14.12	2.05
P_7	9.09	1.41	P_{15}	15.00	1.00

Los indicadores de distancia y tiempo de procesamiento muestran que la distancia del camino es $16.11m$, y el tiempo de ejecución es $6.1s$.

4.1.3.5 Análisis comparativo para la tercera prueba

Con los datos obtenidos de las variables más importantes de los tres métodos planificadores, se resumen en la siguiente tabla:

Tabla 4.15 Parámetros relevantes de métodos planificadores para prueba 3

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13.75	47.82	17.02	16.11
Tiempo de Procesamiento (s)	27.6	1.2	2.8	6.1
Número de quiebres	5	26	8	14
Nodos de Grafo/Iteraciones	714	58	128	168

Desde el punto vista de la ruta obtenida, el método de Descomposición de Celdas es el que presenta la mayor distancia entre las configuraciones, el triple respecto a los otros métodos planificadores, debido a que el entorno presenta escalonamientos por lo que los nodos de las celdas adyacentes se encuentran

muy dispersos formando quiebres innecesarios al formar los enlaces entre nodos vecinos, entonces este método se excluye del análisis siendo ineficiente e inaplicable al control con el Robotino® a pesar que tiene el menor tiempo de procesamiento en general.

Con la modificación del algoritmo de Descomposición de Celdas agregando enlaces a las celdas no adyacentes (Descomposición de Celdas Modificado), la respuesta mejora excepcionalmente siendo competitivo con los otros métodos planificadores.

Tomando en cuenta las pruebas anteriores, el método por Grafo de Visibilidad presenta la trayectoria más simple y con menor distancia de los tres métodos con el 22% menos distancia respecto al método por Descomposición de Celdas Modificado.

La mejor respuesta de los tres métodos planificadores es la presentada por los Campos potenciales, ya que incluye las distancia entre los obstáculos y la ruta generada para que pueda ser ejecutada por el Robotino®, a diferencia del método de Grafo de visibilidad y de Descomposición de Celdas Modificado, ya que se debe considerar el entorno expandido.

Desde el punto de vista del tiempo de ejecución, inmediatamente se concluye que el método por Grafo de Visibilidad es el más lento respecto a los otros dos ratificando que el método se vuelve lento cuando el mapa de entorno se complica, debido a que el número de enlaces se disparó respecto a los otros, y la gran mayoría no tienen nada que ver con el área que se encuentran las configuraciones.

El algoritmo modificado del método Descomposición de Celdas Modificado, presenta el menor tiempo invertido, siendo el método más rápido que resuelve la ruta entre las configuraciones en un mapa de entorno complejo, considerando datos absolutos el tiempo de $2.8s$, versus la ruta obtenida resulta ser la mejor opción para ejecutar este método en casos que se requiera una respuesta rápida

en mapas complejos. Como aspecto adicional el tanto el método de Descomposición de Celdas como el de Campos Potenciales tiene cierta inmunidad a la complejidad del mapa de entorno obteniendo variaciones no considerables en el tiempo de procesamiento.

4.1.4 CUARTA PRUEBA, RENDIMIENTO EN FUNCIÓN DEL PROCESADOR

Las tres pruebas anteriores, están enfocadas en el análisis del comportamiento de los métodos planificadores entre sí, en función del mapa de entorno bajo las mismas condiciones respecto al equipo utilizado.

En esta última prueba se analiza el rendimiento y comportamiento de los planificadores, con equipos de diferentes prestaciones este es el caso del tiempo de procesamiento, para lo cual se mantienen bajo las mismas condiciones el mapa de entorno implementado.

Las pruebas se realizan en tres computadores que tienes las siguientes características:

Computador 1.

- Netbook Acer One de 10.1”
- Procesador Intel Atom de $1.6GHz$.
- RAM $1GB$.
- Disco duro de $160GB$.
- Windows XP Home $32bits$.
- Plataforma de Programación LabVIEW 2010.

Computador 2.

- Notebook Dell Inspirion 1525 15.4”
- Procesador Intel CORE 2 DUO $T5750$ de $2.0GHz$.
- Disco duro de $500GB$.
- RAM $3GB$.
- Sistema Operativo: Windows 7 Profesional de $32bits$.
- Plataforma de Programación LabVIEW 2010.

Computador 3.

- Computador de Escritorio
- Procesador Intel CORE i7-3770 $3.4Ghz$
- RAM $8GB$
- Disco Duro de $1TB$
- Sistema Operativo: Windows 7 Profesional de $32bits$.
- Plataforma de Programación LabVIEW 2010.

Para comprobar el rendimiento de los métodos planificadores, se toma un entorno complejo para el análisis, en este caso se considera el Laboratorio (Figura 4.10).

4.1.4.1 Prueba con Computador 1

Calculando la ruta de los tres planificadores con las configuraciones establecidas en la tercera prueba, los resultados se presentan en la Tabla 4.16:

Tabla 4.16 Parámetros relevantes utilizando computador 1

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13.75	47.24	16.81	16.11
Tiempo de Procesamiento (s)	76.5	3.2	7.4	15.9
Número de quiebres	5	26	8	14
Nodos de Grafo/Iteraciones	714	58	128	168

Cambiando las configuraciones a las coordenadas:

$$P_0 : (7\,,1.5)$$
$$P_f : (18\,,6)$$

Se vuelve a determinar las rutas de los planificadores sobre el mapa de entorno obteniendo:

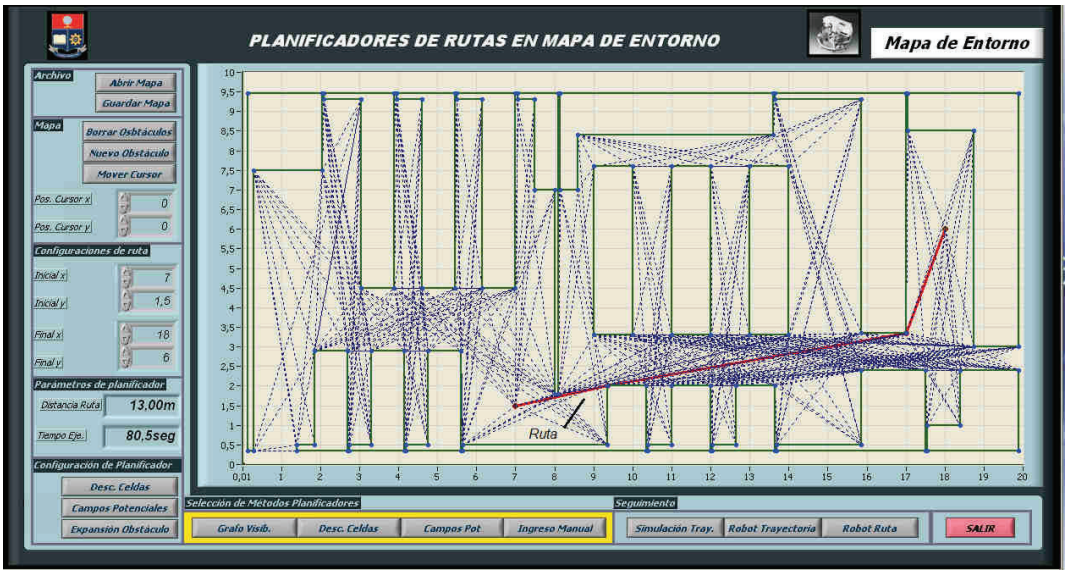


Figura 4.15 Respuesta por Grafo de Visibilidad utilizando Computador 1

El grafo de conectividad desarrollado por Grafo de Visibilidad (Figura 4.15) arroja 718 enlaces válidos y 6 nodos que contienen a la ruta.

Figura 4.16 Respuesta por Descomposición de Celdas utilizando computador 1

Con el método Descomposición de Celdas (Figura 4.16) el grafo de conectividad tiene 25 enlaces entre los nodos, y 19 nodos que forman la ruta entre las configuraciones.

Figura 4.17 Respuesta por Descomposición de Celdas Modificado

Utilizando la modificación del grafo de conectividad (Figura 4.17) del método de Descomposición de Celdas, número de enlaces aumentó a 60, mientras que la ruta contiene 10 nodos entre las configuraciones.

Figura 4.18 Respuesta por Campos Potenciales utilizando Computador 1

La ruta obtenida con el método de Campos potenciales (Figura 4.18) presenta varios quiebres para la evasión de los obstáculos rodeándolos, con 24 nodos para completarla.

Los valores de las variables más importantes, se resumen en la Tabla 4.17:

Tabla 4.17 Parámetros relevantes utilizando computador 1

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13	32.08	13.82	16.21
Tiempo de Procesamiento (s)	80.1	3.2	8.2	21.8
Número de quiebres	2	17	7	30
Nodos de Grafo/Iteraciones	708	57	118	211

4.1.4.2 Prueba con Computador 2

Aplicando los métodos planificadores sobre el mismo mapa de entorno y configuraciones utilizando el segundo computador, se obtienen los siguientes resultados presentados en las Tablas 4.18, y 4.19 para sus correspondientes configuraciones.:

Para las configuraciones:

$$P_0 : (2\ , 4)$$
$$P_f : (15\ , 1)$$

Tabla 4.18 Parámetros relevantes utilizando computador 2

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13.75	47.24	17.02	16.11
Tiempo de Procesamiento (s)	28.9	1.2	2.8	6.1
Número de quiebres	5	26	8	14
Nodos de Grafo/Iteraciones	714	58	128	168

Para la siguiente prueba se toma las configuraciones:

$$P_0 : (7, 1.5\ )$$
$$P_f : (18\ , 6)$$

Tabla 4.19 Parámetros relevantes utilizando computador 2

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13	32.08	13.82	16.21
Tiempo de Procesamiento (s)	27.8	1.3	2.9	7.4
Número de quiebres	2	19	9	32
Nodos de Grafo/Iteraciones	708	57	118	211

4.1.4.3 Prueba con Computador 3

Ejecutando los métodos planificadores, con el computador de mejores características con las siguientes configuraciones, se obtienen los resultados correspondientes en las Tablas 4.20, y 4.21:

$$P_0 : (2\,, 4)$$
$$P_f : (15\,, 1)$$

Tabla 4.20 Parámetros relevantes utilizando computador 3

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13.75	30.82	16.81	16.11
Tiempo de Procesamiento (s)	9.5	0.1	1	1.9
Número de quiebres	5	26	8	14
Nodos de Grafo/Iteraciones	714	58	128	168

Y para la siguiente prueba se cambia las configuraciones a:

$$P_0 : (7, 1.5\,)$$
$$P_f : (18\,, 6)$$

Tabla 4.21 Parámetros relevantes utilizando computador 3

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Distancia Recorrida (m)	13	32.08	13.82	16.21
Tiempo de Procesamiento (s)	9.5	0.4	1	2.6
Número de quiebres	2	19	9	32
Nodos de Grafo/Iteraciones	708	57	118	211

4.1.4.4 Análisis comparativo de dependencia de procesador.

De los resultados obtenidos en utilizando los tres computadores aplicado a dos trayectorias, el único parámetro que cambia es el tiempo de procesamiento del método planificador.

Tabla 4.22 Rendimiento de procesador en función del método aplicado, primera ruta

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Computador 1 t (s)	76.5	3.2	7.4	15.9
Computador 2 t (s)	27.6	1.2	2.8	6.3
Computador 3 t (s)	9.5	0.4	1	1.9

Tabla 4.23 Rendimiento de procesador en función del método aplicado, segunda ruta

	Grafo de Visibilidad	Descomposición de Celdas	Descomposición de Celdas Mod.	Campos Potenciales
Computador 1 t (s)	77.1	3.3	7.6	21.4
Computador 2 t (s)	27.8	1.3	2.9	7.4
Computador 3 t (s)	9.5	0.1	1	2.6

Si se relacionan los tiempos de los métodos planificadores individualmente de las Tablas 4.22 y 4.23, en ambos casos el computador 3 es aproximadamente 8 veces más rápido respecto al primero, y 3 veces más rápido que el computador 2. Estos datos demuestran que los métodos planificadores implementados dependen del rendimiento del procesador para disminuir el tiempo de procesamiento, y además también los resultados indican que los tiempos invertidos para cada método planificador son proporcionales sin importar el computador utilizado.

Comparando los tiempos de los métodos basados en grafos, cuando se tiene un mapa de entorno existen pequeñas variaciones en los tiempos de ejecución pero no significativos, cuando se cambian las coordenadas de las configuraciones, haciendo que la ruta obtenida a diferentes configuraciones sean inmunes variaciones de tiempo.

En el caso del método de campos potenciales, para obtener la ruta existe dependencia entre la distancia a la que se encuentren las configuraciones, y la cantidad de obstáculos entre ellas.

4.2 PRUEBA DE FUNCIONAMIENTO DE CONTROL DE RUTAS

Esta sección del capítulo está enfocada en el comportamiento del Robotino® aplicando el control de ruta implementado en el capítulo 3, para lo cual se traza una ruta sin mapa de entorno, utilizando la opción ingreso manual de ruta presentado en la Figura 4.19.

Figura 4.19 Prueba de control para Robotino®

La ruta trazada de la Figura 4.19, tiene las siguientes coordenadas y secuencia presentadas en la Tabla 4.24

Tabla 4.24 Nodos de ruta trazada

Nodos	P_0	P_1	P_2	P_f
Nodo x (m)	1	4	1	4
Nodo y (m)	1	2	2	1

La distancia total calculada de la ruta es $9.32m$.

Al ejecutar el programa de control de ruta, este calcula los ángulos de rotación y las distancias de avance del Robotino®, para alcanzar las configuraciones que componen la ruta.

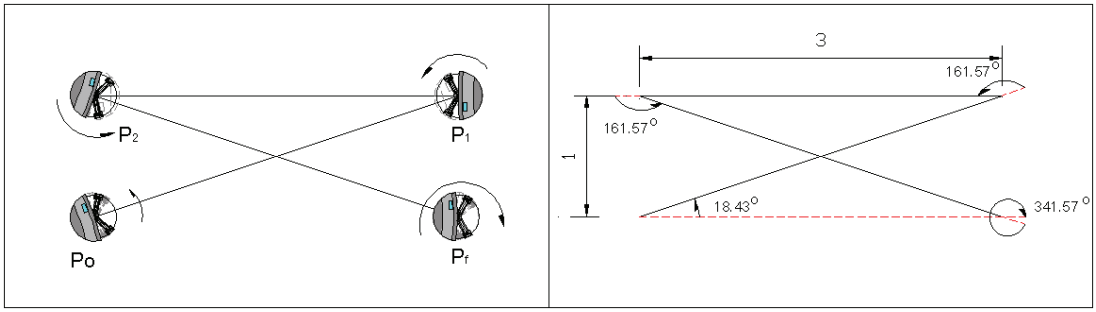


Figura 4.20 Representación geométrica de movimientos del Robot

El seguimiento de la ruta se ejecuta por medio de rotaciones y desplazamientos independientes (Figura 4.20). El robot primero se orienta con el ángulo de rotación correspondiente al nodo donde esté ubicado, para luego desplazarse hacia el siguiente nodo.

Figura 4.21 Ruta ejecutada por el Robotino®

Durante la ejecución del control de ruta, el HMI (Figura 4.21) presenta la información de los movimientos del Robotino® utilizando los encoders, y los reportes presentan toda la información recolectada por los encoders del robot que para este caso particular el reporte indica que la ruta ejecutada tiene 228 muestras de las cuales se presentan 46 en la Tabla 4.25:

Tabla 4.25 Puntos coordenados entregados por el Robotino® para control de ruta

Nodo	Ruta R x (m)	Ruta R y (m)	Teórico y (m)	Error (m)	Nodo	Ruta R x (m)	Ruta R y (m)	Teórico y (m)	Error (m)
P_0	1	1	1,00	0	P_{114}	2.9	1.99	2.00	0.01
P_4	1.16	1.05	1.05	0	P_{119}	2.73	1.99	2.00	0.01
P_9	1.37	1.12	1.12	0	P_{124}	2.5	1.99	2.00	0.01
P_{14}	1.61	1.2	1.20	0	P_{129}	2.27	1.99	2.00	0.01
P_{19}	1.83	1.27	1.28	0.01	P_{134}	2.04	1.99	2.00	0.01
P_{24}	2.07	1.35	1.36	0.01	P_{139}	1.8	1.99	2.00	0.01
P_{29}	2.28	1.42	1.43	0.01	P_{144}	1.56	1.99	2.00	0.01
P_{34}	2.5	1.49	1.50	0.01	P_{149}	1.33	1.99	2.00	0.01
P_{39}	2.71	1.56	1.57	0.01	P_{154}	1.1	1.99	2.00	0.01
P_{44}	2.93	1.63	1.64	0.01	P_{159}	1.07	1.96	1.98	0.02
P_{49}	3.14	1.71	1.71	0	P_{164}	1.29	1.89	1.90	0.01
P_{54}	3.36	1.78	1.79	0.01	P_{169}	1.51	1.82	1.83	0.01
P_{59}	3.57	1.85	1.86	0.01	P_{174}	1.73	1.75	1.76	0.01
P_{64}	3.78	1.92	1.93	0.01	P_{179}	1.95	1,67	1.68	0.01
P_{69}	4	1.99	2.00	0.01	P_{184}	2.17	1.6	1.61	0.01
P_{74}	3.83	1.99	2.00	0.01	P_{189}	2.38	1.53	1.54	0.01
P_{79}	3.6	1.99	2.00	0.01	P_{194}	2.6	1.45	1.47	0.02
P_{84}	3.48	1.99	2.00	0.01	P_{199}	2.83	1.38	1.39	0.01
P_{89}	3.36	1.99	2.00	0.01	P_{204}	3.04	1.31	1.32	0.01
P_{94}	3.25	1.99	2.00	0.01	P_{209}	3.27	1.23	1.24	0.01
P_{99}	3.02	1.99	2.00	0.01	P_{214}	3.48	1.16	1.17	0.01
P_{104}	2.94	1.99	2.00	0.01	P_{219}	3.7	1.09	1.10	0.01
P_{109}	2.92	1.99	2.00	0.01	P_f	3.97	1	1.01	0.01

En la Tabla 4.25, se muestra los valores de las posiciones llegadas por el Robotino® en x, y, en la tercera columna se calculó el valores teóricos de la ruta a partir de las coordenadas de la Tabla 2.24 utilizando rectas, y en la última columna se encuentran la desviación del robot frente a los valores calculados. Por el HMI.

En esta prueba, la desviación no supera los $2cm$, lo cual permite ver que el control de ruta está calibrado para los requerimientos del Robotino® para la velocidad de $150mm/s$ para la traslación.

4.3 PRUEBAS DE CONTROL DE RUTAS DEL ROBOTINO® EN ENTORNO REAL

En esta sección del capítulo se analiza el control de ruta aplicada en un entorno real utilizando los métodos planificadores.

El mapa de entorno para el control de ruta es el laboratorio (Anexo C Lámina C01) y tiene las siguientes configuraciones:

$$P_0 : (15\ ,5.5\ )$$
$$P_f : (8.5\ ,1)$$

Determinado los parámetros de entorno y configuraciones, se procede a ejecutar los métodos planificadores:

4.3.1 PRUEBA DE CONTROL POR GRAFO DE VISIBILIDAD

La limitación que tiene el método Grafo de Visibilidad al llevarlo a la práctica es considerar el robot como un punto en el plano, por lo cual se utiliza la técnica del entorno expandido.

Los obstáculos $8, 9, 10, 11, 12, y\ 13$ de la Figura 4.22 están expandidos $50cm$ por lado; aplicando el método planificador entre las configuraciones se obtienen los siguientes resultados presentados en el HMI.

Los parámetros presentados por el HMI, indican que ruta tiene cuatro nodos en total, la distancia es de $9.68m$ y el tiempo de cálculo del método planificador es $29.3s$.

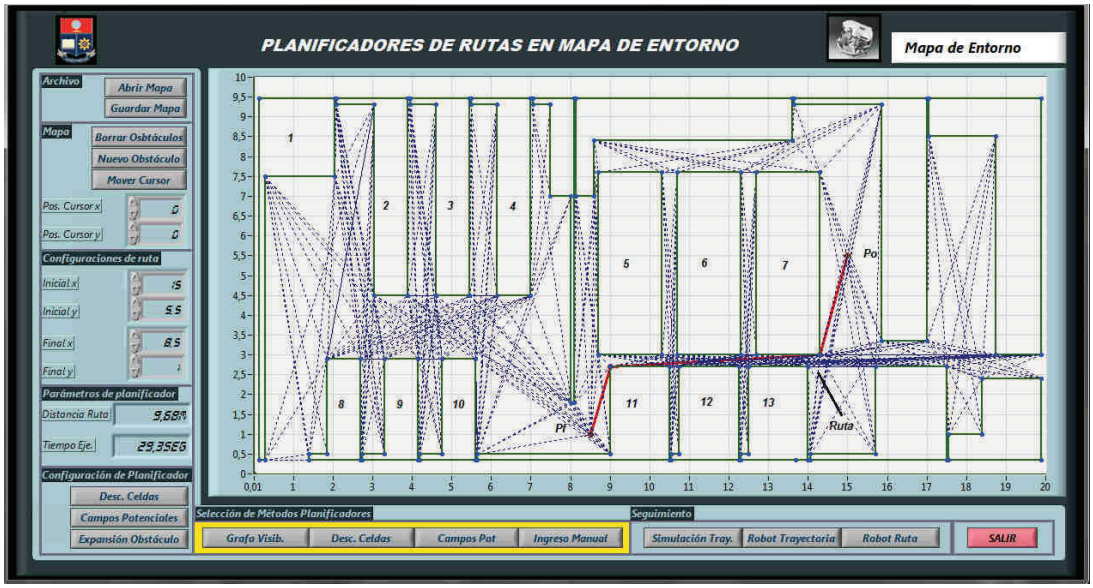


Figura 4.22 Respuesta de por Grafo de Visibilidad

Con la ruta obtenida del método planificador, se procede a ejecutar el control de ruta Robotino®.

Durante la ejecución, el HMI dibuja el camino recorrido por del Robotino® tomada desde los encoders del mismo (Figura 4.23).

Figura 4.23 Camino obtenido por el robot, método Grafo de Visibilidad

La Figura 4.23 muestra la ruta obtenida por el método planificador con el mapa de entorno expandido, la cual difiere respecto al entorno real del Laboratorio permitiendo ejecutar el control de ruta.

La ruta ejecutada por el Robot es prácticamente la del método planificador. La Tabla 4.26 presenta una muestra del 20% de los puntos coordenados obtenidos por las mediciones de los encoders del Robotino®, y sus desviaciones respecto a la ruta calculada por el método planificador.

Tabla 4.26 Puntos coordenados entregados por el Robotino® para control de ruta, Método Grafo de Visibilidad

Nodo	Ruta R x (m)	Ruta R y (m)	Ruta planificad	Error (m)	Nodo	Ruta R x (m)	Ruta R y (m)	Ruta planificad	Error (m)
P_0	15.00	5.50	5.50	0	P_{114}	11.59	2.85	2.85	0
P_4	14.95	5.32	5.32	0	P_{119}	11.36	2.84	2.83	0.01
P_9	14.89	5.10	5.11	0.01	P_{124}	11.12	2.83	2.82	0.01
P_{14}	14.83	4.88	4.89	0.01	P_{129}	10.89	2.82	2.81	0.01
P_{19}	14.75	4.61	4.61	0	P_{134}	10.65	2.80	2.79	0.01
P_{24}	14.69	4.39	4.39	0	P_{139}	10.42	2.79	2.78	0.01
P_{29}	14.63	4.17	4.18	0.01	P_{144}	10.18	2.78	2.77	0.01
P_{34}	14.57	3.95	3.96	0.01	P_{149}	9.95	2.76	2.75	0.01
P_{39}	14.50	3.73	3.71	0.02	P_{154}	9.71	2.75	2.74	0.01
P_{44}	14.44	3.50	3.50	0	P_{159}	9.48	2.74	2.73	0.01
P_{49}	14.38	3.28	3.29	0.01	P_{164}	9.24	2.73	2.71	0.02
P_{54}	14.32	3.06	3.07	0.01	P_{169}	9.01	2.71	2.70	0.01
P_{59}	14.17	2.99	2.99	0	P_{174}	8.94	2.50	2.50	0
P_{64}	13.94	2.98	2.98	0	P_{179}	8.88	2.28	2.29	0.01
P_{69}	13.70	2.97	2.97	0	P_{184}	8.81	2.05	2.05	0
P_{74}	13.47	2.95	2.95	0	P_{189}	8.80	2.01	2.02	0.01
P_{79}	13.24	2.94	2.94	0	P_{194}	8.74	1.82	1.82	0
P_{84}	13.01	2.93	2.93	0	P_{199}	8.67	1.57	1.58	0.01
P_{89}	12.76	2.91	2.91	0	P_{204}	8.60	1.34	1.34	0
P_{94}	12.53	2.90	2.90	0	P_{209}	8.54	1.14	1.14	0
P_{99}	12.29	2.89	2.89	0	P_{214}	8.54	1.12	1.14	0.02
P_{104}	12.06	2.88	2.87	0.01	P_f	8.50	1.00	1.00	0
P_{109}	11.83	2.87	286	0.01					

El error se calcula a partir de la diferencia entre los valores de posición del Robotino® y los nodos del camino construido de la tura obtenida por el método

planificador, la Figura 4.24 presenta la curva del error a lo largo de la ejecución del control de ruta.

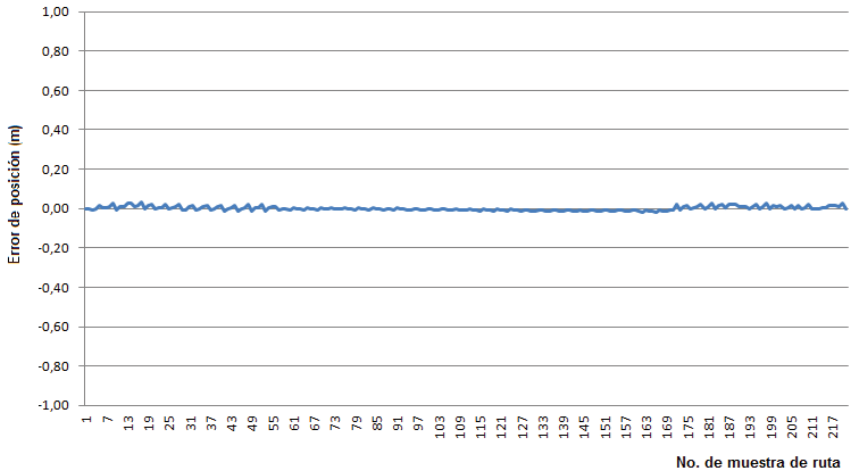


Figura 4.24 Curva de error de posición del Robotino®

El reporte de la Tabla 4.26 y la curva de la Figura 4.24 muestran que el error es bajo, demostrando que el programa de control de ruta desplaza al robot adecuadamente.

Utilizando los puntos coordenados del reporte se calcula la distancia real recorrida por el robot leída por los encoders, $D = 9.68m$

El error de la distancia comparada con el método planificador es: $E_D = 0.41\%$

El tiempo que tardó el robot en ejecutar la ruta es:

$$t_{Rob} = 110.28s \text{ ó } 1'{:}50''$$

Tabla 4.27 Rotaciones del Robotino por Grafo de Visibilidad

Rotación	Grados	Rotación	Grados
θ_1	254.35	θ_3	72.72
θ_2	71.11	θ_4	255.96

La información recolectada representa las variables que el HMI puede controlar para ejecutar el control de ruta lo mejor posible, pero no representa el camino real que robot dibuja. Estos errores que pueden generarse debido a factores externos como las posibles imperfecciones del piso, suciedad, inclinaciones, coeficiente de rozamiento de las ruedas del Robotino® con el piso que determina la tracción, que no pueden ser registradas correctamente por los encoders, entonces para determinar el desempeño real del robot, se realizó mediciones directas de los desplazamientos del robot, y se determinó su verdadero camino y los errores generados, para lo cual en el mapa de entorno se traza una cuadrícula para referenciar el trayectoria del robot presentado en la Figura 4.25 y anexo C.

Figura 4.25 Ejecución de movimientos del robot

La Figura 4.26 presenta la cuadrícula de $50cm$ por lado, en el laboratorio dimensionado a escala.

Para las mediciones y posterior cálculo de errores se toma en cuenta que el error en la configuración inicial es cero, y para cada nodo se toma los valores de las coordenadas de los mismos, y se calcula la desviación de cada nodo. Llevando las mediciones a un gráfico se puede comparar con la trayectoria teórica con la del Robotino® (ver anexo C).

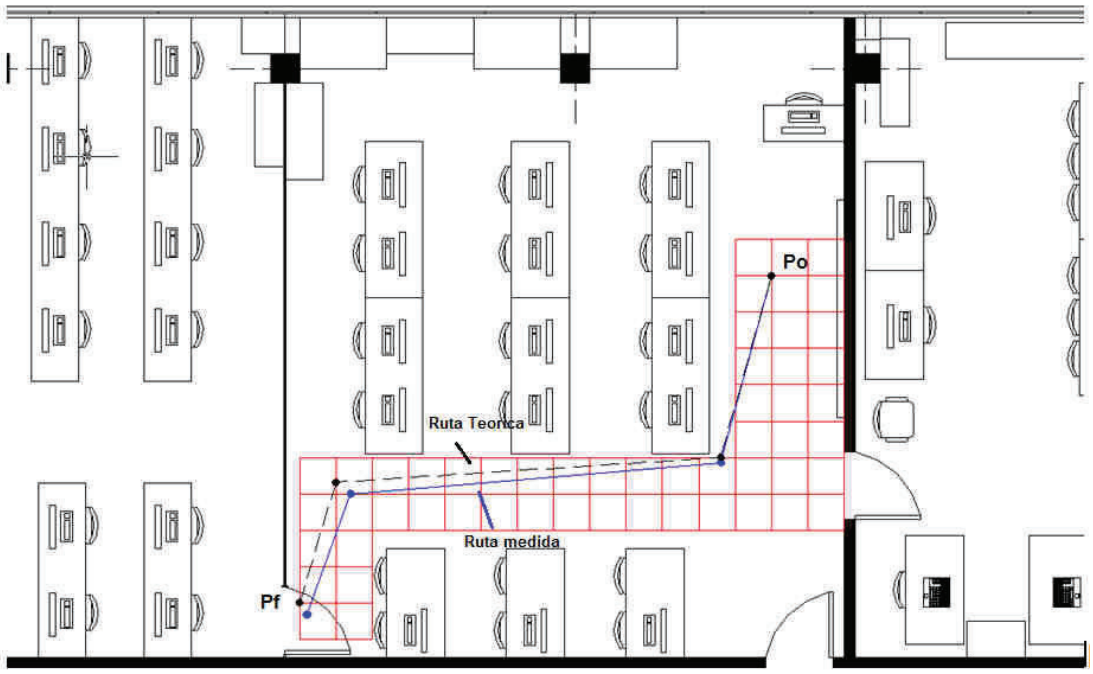


Figura 4.26 Comparación visual entre nodos teóricos y medidos directamente

Al comparar los caminos formados por el método planificador, con la medida del Robotino®, se observa que tiene error de posición que se incrementa conforme el robot ejecuta los movimientos. La Tabla 4.28 presenta las coordenadas de los nodos ejecutados por el Robotino® medidos directamente en el mapa de entorno:

Tabla 4.28 Nodos medidos en el mapa de entorno

Nodos de Ruta				Desplazamientos entre nodos				Distancia entre nodos	
Teórica (m)		Medida (m)		Teórico		medido		Teórico	Medido
$Ruta\ (x)$	$Ruta\ (y)$	$Ruta\ (x)$	$Ruta\ (y)$	Dx (m)	Dy (m)	Dx (m)	Dy (m)	(m)	(m)
15.00	5.50	15	5.5						
14.30	3.00	14.27	2.97	0.70	2.50	0.73	2.53	2.60	2.63
9.00	2.70	9.2	2.5	5.30	0.30	5.07	0.47	5.31	5.09
8.50	1.00	8.6	0.85	0.50	1.70	0.60	1.65	1.77	1.76

Como el desplazamiento del robot es en línea recta, la información a tomar en cuenta son las coordenadas de los nodos, que se producen cuando el robot ejecuta las rotaciones. Con la información de los nodos se pueden determinar los

demás parámetros como las distancias parciales, desviaciones entre datos teóricos y los medidos.

Tabla 4.29 Errores calculados por nodos medidos en el mapa de entorno

Desviación		Error de posición	
$V(x)$	$V(y)$	$e(x)$	$e(y)$
0	0	0.00%	0.00%
0.03	0.03	0.21%	1.01%
0.2	0.2	2.17%	8.00%
0.1	0.15	1.16%	17.65%

La Tabla 4.29 presenta los errores relativos de posición en función del desplazamiento ejecutado por el robot manteniendo la independencia entre los ejes coordenados. Estas desviaciones muestran que existe un incremento del error conforme el robot se acerca a su posición objetivo, pero el resultado no es significativo ya que en términos absolutos las desviaciones obtenidas son 10 y $15cm$ para los ejes x y y respectivamente para una ruta que tiene una distancia de $9.68m$.

Otro parámetro a tomar en cuenta es la distancia recorrida entre los nodos de la trayectoria, la Tabla 4.28 presenta la información de la distancia teórica y la real medida, los cuales son muy cercanos entre sí, ratificando la información visual presentada en la Figura 4.26.

4.3.2 PRUEBA DE CONTROL POR DESCOMPOSICIÓN DE CELDAS

Aplicando el método de Descomposición de Celdas modificado, sobre el mapa de entorno expandido $15cm$ a los obstáculos, con las mismas configuraciones se obtiene los siguientes resultados:

Figura 4.27 Respuesta de por Descomposición de Celdas Modificado

La Figura 4.27 indica que la ruta del método planificador tiene 5 nodos en total, la distancia es $10.87m$, y el tiempo de cálculo para la obtención de la ruta es $3.1s$, este tiempo es 10 veces menor al utilizado por el Grafo de Visibilidad y la distancia de la ruta aumenta en 10%.

El siguiente paso es realizar el control de ruta al Robotino®. El HMI dibuja la ruta ejecutada por el robot, y al finalizar se obtiene:

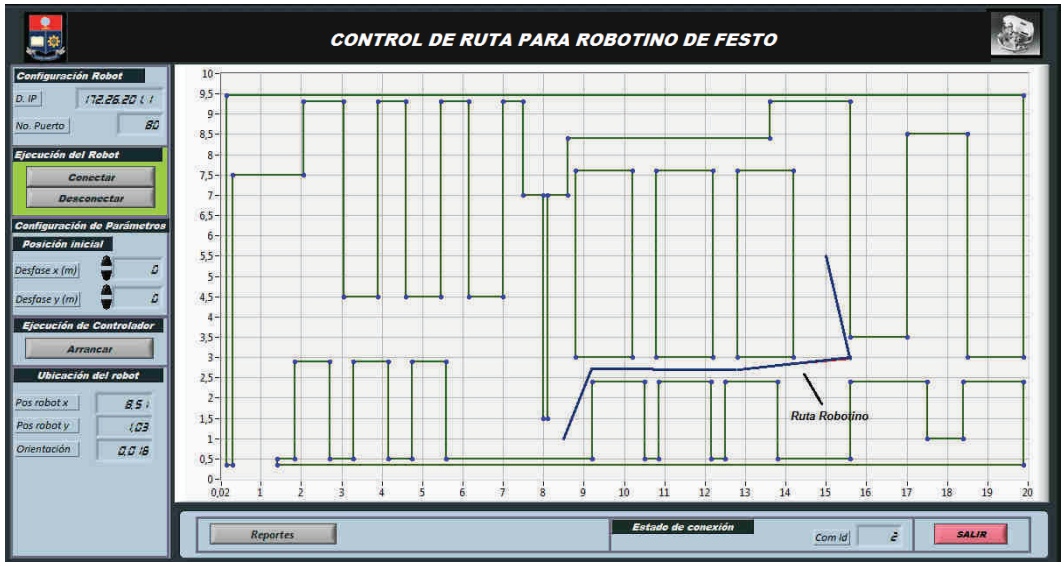


Figura 4.28 Ruta obtenida, método Descomposición de Celdas Modificado

El HMI de la Figura 4.28 presenta el camino seguido por el Robotino®, y en la Tabla 4.30 presenta la tabulación de los datos recolectados por los encoders del robot.

Tabla 4.30 Puntos coordenados entregados por el Robotino® para control de ruta, Método Descomposición de Celdas Modificado

Nodo	Ruta R x (m)	Ruta R y (m)	Teórico	Error (m)	Nodo	Ruta R x (m)	Ruta R y (m)	Teórico	Error (m)
P_0	1500	5.50	5.50	0	P_{114}	12.64	2.80	2.79	0.01
P_4	15.05	5.31	5.29	0.02	P_{119}	12.39	2.79	2.78	0.01
P_9	15.11	5.06	5.03	0.03	P_{124}	12.13	2.78	2.77	0.01
P_{14}	15.17	4.80	4.78	0.02	P_{129}	11.89	2.76	2.75	0.01
P_{19}	15.22	4.58	4.57	0.01	P_{134}	11.63	2.75	2.74	0.01
P_{24}	15.28	4.35	4.31	0.04	P_{139}	11.35	2.73	2.73	0
P_{29}	15.33	4.11	4.10	0.01	P_{144}	11.10	272	2.71	0.01
P_{34}	15.39	3.89	3.84	0.05	P_{149}	10.86	2.71	2.70	0.01
P_{39}	15.45	3.64	3.59	0.05	P_{154}	10.68	2.71	2.70	0.01
P_{44}	15.50	3.41	3.38	0.03	P_{159}	10.43	2.71	2.70	0.01
P_{49}	15.56	3.16	3.12	0.04	P_{164}	10.20	2.71	2.70	0.01
P_{54}	15.61	2.95	2.95	0	P_{169}	9.97	2.71	2.70	0.01
P_{59}	15.39	2.94	2.94	0	P_{174}	9.71	2.71	2.70	0.01
P_{64}	15.16	2.93	2.93	0	P_{179}	9.42	2.71	2.70	0.01
P_{69}	14.90	2.91	2.91	0	P_{184}	9.20	2.71	2.70	0.01
P_{74}	14.67	2.90	2.90	0	P_{189}	9.12	2.51	2.51	0
P_{79}	14.42	2.89	2.89	0	P_{194}	9.02	2.28	2.26	0.02
P_{84}	14.18	2.88	2.88	0	P_{199}	8.93	2.07	2.04	0.03
P_{89}	13.91	2.86	2.86	0	P_{204}	8.83	1.83	1.80	0.03
P_{94}	13.63	2.85	2.85	0	P_{209}	8.75	1.61	1.61	0
P_{99}	13.39	2.84	2.83	0.01	P_{214}	8.64	1.34	1.34	0
P_{104}	13.16	2.83	2.82	0.01	P_{219}	8.53	1.08	1.07	0.01
P_{109}	12.92	2.82	2.81	0.01	P_f	8.51	1.03	1.02	0.01

Los resultados del método planificador, y del obtenido por el Robotino®, es una muestra del 20% presentado en el reporte. El error de la ruta real respecto a la del método planificador se corrige conforme el robot cumple con sus desplazamientos, hasta llegar a la configuración final cuyo error es $1cm$ entre los valores medidos y teóricos.

El tiempo de ejecución de ruta se calcula con las rotaciones y la distancia total recorrida por el Robotino®.

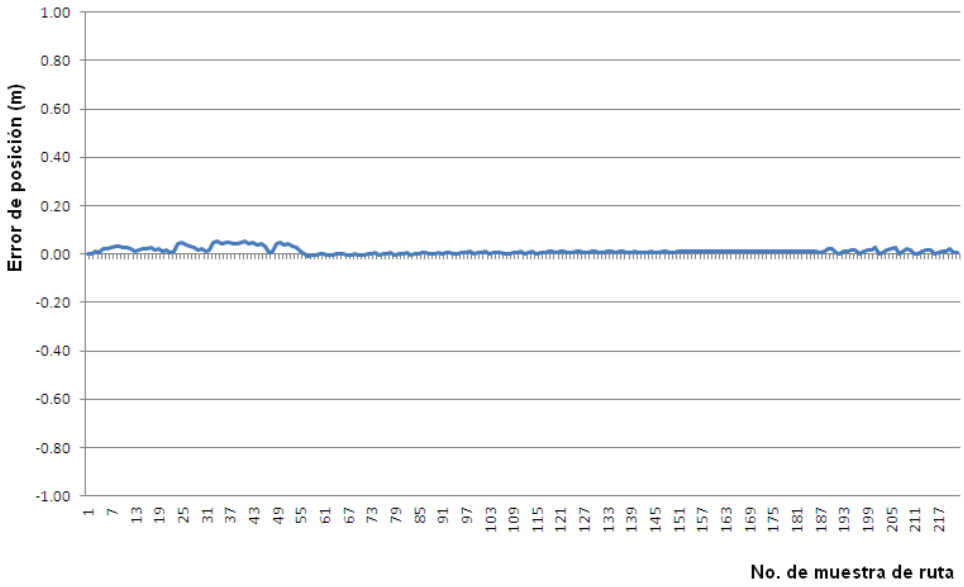


Figura 4.29 Curva de error de posición del Robotino®

La Figura 4.29 presenta la curva del error de posición de la ruta obtenida por el método planificador y la recolectada por los encoders del Robotino® indica que el control de ruta corrige el error generado por la primera parte del trayecto ejecutado por el robot y mantiene el error cercano a 0 ratificando la ruta dibujada por el HMI de la Figura 4.28.

La distancia recorrida se calcula con los puntos obtenidos por los encoders del robot obteniendo $10.91m$

El tiempo que el Robotino® tarda en ejecutar la ruta aplicando el método de Descomposición de Celdas es:

$$t_{Rob} = 119.51s \text{ ó } 1': 59''$$

Las rotaciones que realiza el Robot en los quiebres de la ruta para orientarse y posteriormente realizar los desplazamientos son presentados en la Tabla 2.31.

Tabla 2.31 Rotaciones producidas por el Robotino® en cada no

Rotación	Grados	Rotación	Grados
θ_1	283.24	θ_4	67.62
θ_2	100.22	θ_5	247.62
θ_3	3.02		

Para determinar el recorrido real del Robotino®, se toma las medidas del camino recorrido, construyendo una cuadrícula en el Laboratorio (Figura 4.30) y marcando la ruta que sigue el robot.

Figura 4.30 Ejecución de movimientos del robot

Para comparar la ruta medida ejecutada por el robot y la calculada por el método planificador, se trasladó las dos rutas en un plano obtenido el siguiente resultado presentado en la Figura 4.31.

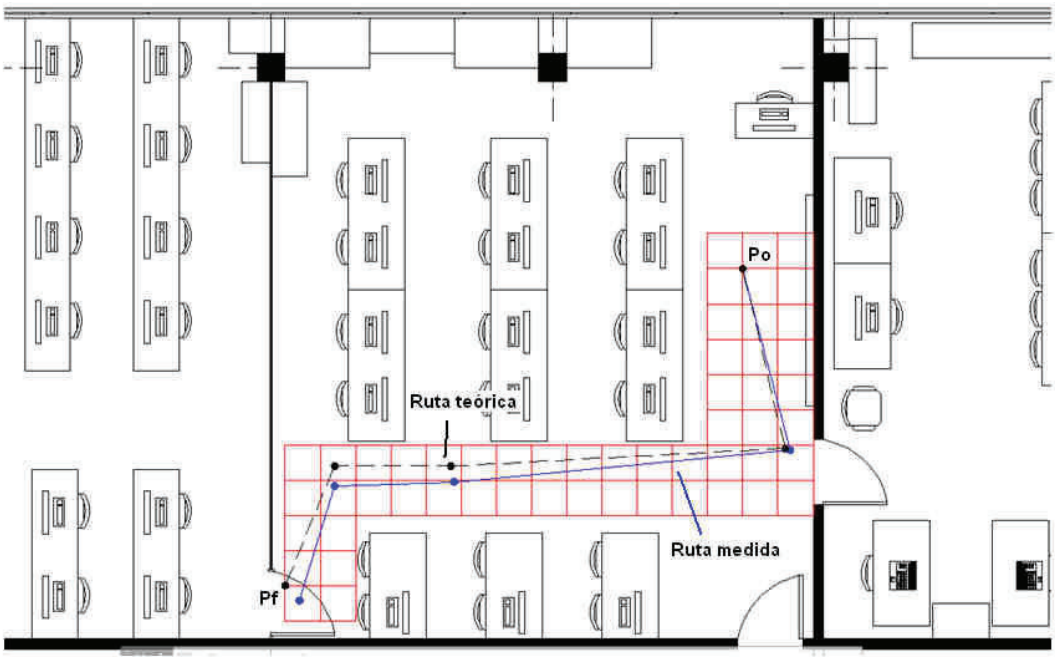


Figura 4.31 Comparación visual entre nodos teóricos y medidos directamente

Partiendo desde la misma configuración inicial (error 0), se sobreponen las dos rutas, en el mismo gráfico. Para determinar los errores, se calculan las coordenadas. La Tabla 4.32 presenta los nodos de ambas rutas presentadas en la Figura 4.31.

Tabla 4.32 Nodos medidos en el mapa de entorno

Nodos de Ruta				Desplazamientos entre nodos				Distancia entre nodos	
Teórica (m)		Medida (m)		Teórico (m)		Medido (m)		Teórico	Medido
$Ruta\ (x)$	$Ruta\ (y)$	$Ruta\ (x)$	$Ruta\ (x)$	Dx (m)	Dy (m)	Dx (m)	Dy (m)	(m)	(m)
15.00	5.50	15	5.5						
15.60	2.95	15.65	2.93	-0.60	2.55	-0.65	2.57	2.62	2.65
10.85	2.70	10.9	2.5	4.75	0.25	4.75	0.43	4.76	4.77
9.20	2.70	9.2	2.45	1.65	0.00	1.70	0.05	1.65	1.70
8.50	1.00	8.7	0.85	0.70	1.70	0.50	1.60	1.84	1.68

Los valores son procesados para determinar las distancias de cada desplazamiento del robot, y tener los parámetros necesarios para determinar las desviaciones que se presentan en la Tabla 4.33

Tabla 4.33 Errores calculados por nodos medidos en el mapa de entorno

Desviación		Error de posición	
$V(x)$	$V(y)$	$e(x)$	$V(x)$
0	0	0.00%	0.00%
0.05	0.02	0.32%	0.68%
0.05	0.2	0.46%	8.00%
0	0.25	0.00%	10.20%
0.2	0.15	2.30%	17.65%

La desviación calculada entre nodos en los dos ejes coordenados se incrementa conforme se llega hacia la configuración final, pero no es mayor a $20cm$ y $15cm$, que representa un error de posición del 3% y 4.41% respectivamente, relacionando con la distancia recorrida por el Robotino®.

4.3.3 PRUEBA DE CONTROL POR CAMPOS POTENCIALES

Manteniendo el entorno expandido de $15cm$ a los obstáculos comprendidos entre las configuraciones, se aplica el método de Campos Potenciales obteniendo la siguiente respuesta:

Figura 4.32 Respuesta de por Campos Potenciales

La ruta obtenida está compuesta por 11 nodos incluyendo las configuraciones, la distancia de la trayectoria es $9.93m$, el tiempo de ejecución es $5s$.

Trasladando la ruta al programa de control de ruta. El Robotino® ejecuta los movimientos controlados por el HMI y construye la ruta indicada por la Figura 4.33.

Figura 4.33 Ruta ejecutada por el robot, método Campos Potenciales

El camino construido tiene aproximadamente 210 muestras y se compara gráficamente con la obtenida por el método planificador presentando gran similitud entre las dos. La siguiente tabla presenta una muestra del 20% comparando ambas rutas.

Tabla 4.34 Puntos coordenados entregados por el Robotino® para control de ruta, Método Campos Potenciales

Nodo	Ruta R x (m)	Ruta R y (m)	Teórico (m)	Error (m)	Nodo	Ruta R x (m)	Ruta R y (m)	Teórico (m)	Error (m)
P_0	15.00	5.50	5.50	0	P_{109}	12.34	2.61	2.62	0.01
P_4	14.84	5.36	5.37	0.1	P_{114}	12.09	2.61	2.62	0.01
P_9	14.66	5.22	5.22	0	P_{119}	11.83	2.61	2.62	0.01

P_{14}	14.51	5.08	5.06	0.02	P_{124}	11.59	2.61	2.62	0.01
P_{19}	14.46	4.89	4.87	0.02	P_{129}	11.36	2.61	2.62	0.01
P_{24}	14.46	4.69	4.65	0.04	P_{134}	11.13	2.61	2.62	0.01
P_{29}	14.46	4.43	4.39	0.04	P_{139}	10.90	2.61	2.62	0.01
P_{34}	14.47	4.18	4.16	0.02	P_{144}	10.67	2.61	2.62	0.01
P_{39}	14.47	3.95	3.90	0.05	P_{149}	10.43	2.61	2.62	0.01
P_{44}	14.46	3.70	3.67	0.03	P_{154}	10.12	2.61	2.62	0.01
P_{49}	14.47	3.47	3.44	0.03	P_{159}	9.86	2.61	2.62	0.01
P_{54}	14.44	3.26	3.20	0.06	P_{164}	9.63	2.61	2.62	0.01
P_{59}	14.41	3.03	2.98	0.05	P_{169}	9.40	2.61	2.62	0.01
P_{64}	14.32	2.87	2.87	0	P_{174}	9.15	2.59	2.59	0
P_{69}	14.18	2.72	2.73	0.01	P_{179}	9.00	2.48	2.47	0.01
P_{74}	13.95	2.60	2.61	0.01	P_{184}	8.86	2.20	2.18	0.02
P_{79}	13.75	2.60	2.62	0.02	P_{189}	8.75	1.97	1.96	0.01
P_{84}	13.52	2.60	2.62	0.02	P_{194}	8.65	1.76	1.75	0.01
P_{89}	13.29	2.60	2.62	0.02	P_{199}	8.60	1.57	1.52	0.05
P_{94}	13.06	2.60	2.62	0.02	P_{204}	8.56	1.34	1.31	0.03
P_{99}	12.83	2.60	2.62	0.02	P_{209}	8.51	1.11	1.05	0.06
P_{104}	12.57	2.60	2.62	0.02	P_f	8.49	1.02	0.95	0.07

La mayoría de los errores presentados en la Tabla 4.34, no superan la desviación de $2cm$, por lo que el control de ruta cumple con el objetivo de llevar al Robotino® a la configuración final.

Trazando la curva de los errores de la Tabla 4.34 se obtiene:

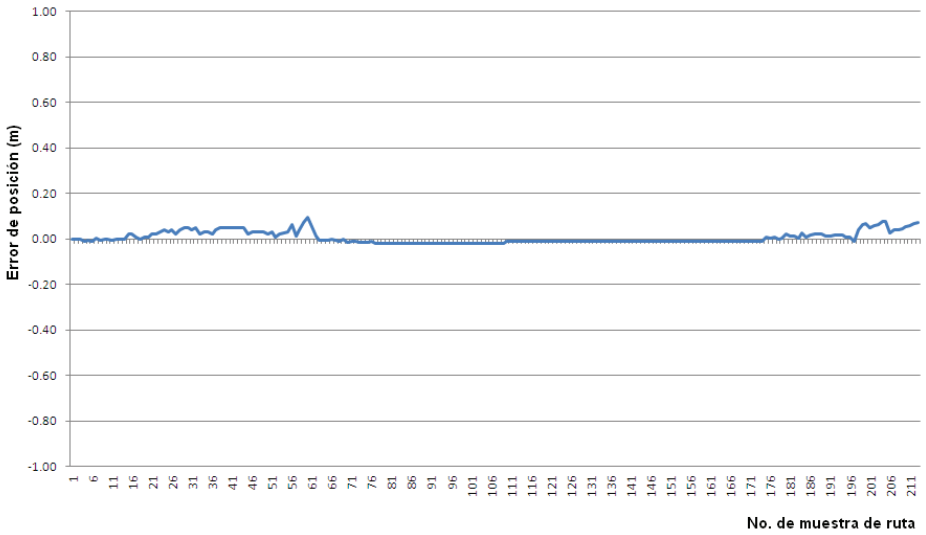


Figura 4.34 Curva de error de posición del Robotino®

La Figura 4.34 indica que se forma un error en la primera parte del seguimiento de la ruta por parte del robot, pero se corrige conforme el robot se traslada por la parte con menor número de nodos de la ruta y mantiene en valores bajos considerando la distancia que se realiza el control

Con las coordenadas de los nodos de la ruta seguida por el robot (Tabla 4.34), se calcula la distancia recorrida por el Robotino® según los encoders y su valor es:

$$D = 9.97m$$

El tiempo de ejecución para completar el control de ruta aplicando este método planificador es:

$$t_{Rob} = 112.96s \text{ ó } 1'{:}53''$$

Las rotaciones que realiza el Robot en los quiebres de la ruta para orientarse y posteriormente realizar los desplazamientos son presentados en la Tabla 4.31.

Tabla 4.35 Rotaciones producidas por el Robotino® producidas por nodos

Rotación	Grados	Rotación	Grados
θ_1	218.95	θ_7	23.03
θ_2	32.61	θ_8	27.53
θ_3	18.44	θ_9	33.02
θ_4	0	θ_{10}	31.13
θ_5	7.66	θ_{11}	15.11
θ_6	31.9	θ_{12}	259.14

Utilizando una cuadrícula en el Laboratorio se tomó medidas de los desplazamientos del robot, y se trazó una ruta directamente en el entorno real presentado en la Figura 4.35.

Figura 4.35 Ejecución de movimientos del robot

Con las medidas de la ruta se compara con la ruta calculada por el método planificador, las cuales son trasladadas al plano (Figura 4.36).

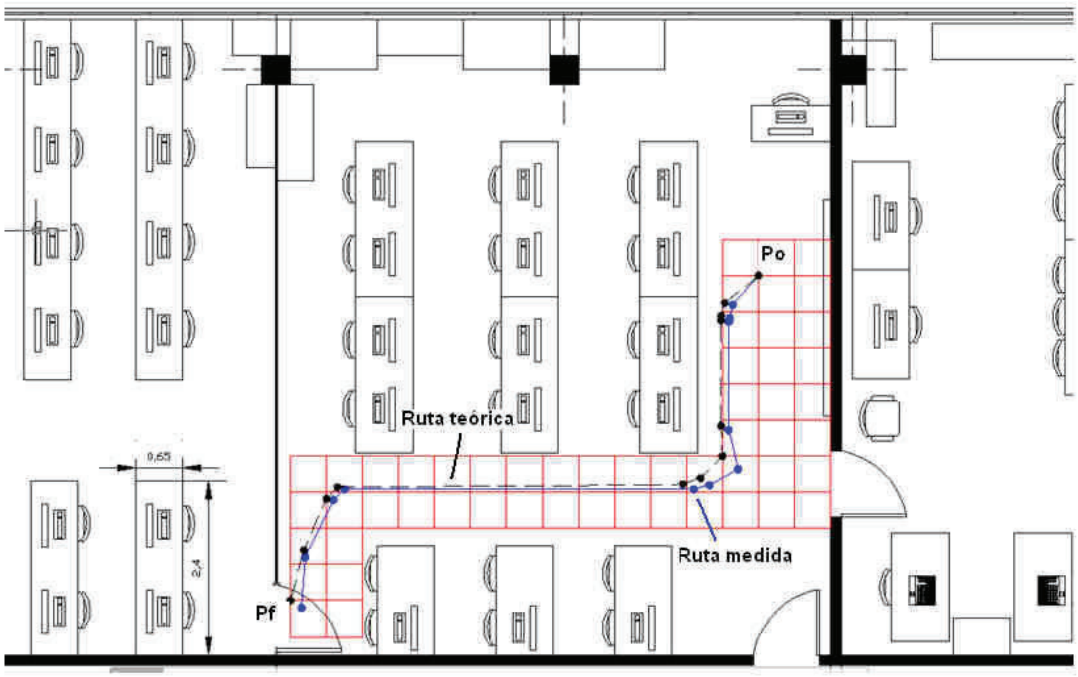


Figura 4.36 Comparación visual entre nodos teóricos y medidos directamente

Visualmente las curvas presentan gran similitud entre ellas, y de estas, se toman todos los nodos, se determinan numéricamente las coordenadas de cada uno y se presenta en la Tabla 4.35:

Tabla 4.35 Nodos medidos en el mapa de entorno

Nodos de Ruta				Desplazamientos entre nodos				Distancia entre nodos	
Teórica (m)		Medida (m)		Teórico (m)		Medido (m)		Teórico (m)	Medido (m)
$Ruta\ (x)$	$Ruta\ (y)$	$Ruta\ (x)$	$Ruta\ (y)$	Dx (m)	Dy (m)	Dx (m)	Dy (m)	(m)	(m)
15.00	5.50	15	5.5						
14.53	5.12	14.67	5.09	0.47	0.38	0.33	0.41	0.60	0.53
14.47	4.94	14.6	4.92	0.06	0.18	0.07	0.17	0.19	0.18
14.47	4.88	14.58	4.85	0.00	0.06	0.02	0.07	0.06	0.07
14.47	3.42	14.6	3.36	0.00	1.46	-0.02	1.49	1.46	1.49
14.41	2.98	14.8	2.82	0.06	0.44	-0.20	0.54	0.44	0.58
14.22	2.75	14.32	2.63	0.19	0.23	0.48	0.19	0.30	0.52
13.95	2.61	14.1	2.55	0.27	0.14	0.22	0.08	0.30	0.23
9.19	2.62	9.25	2.55	4.76	-0.01	4.85	0.00	4.76	4.85
9.02	2.51	9.1	2.41	0.17	0.11	0.15	0.14	0.20	0.21
8.64	1.73	8.7	1.62	0.38	0.78	0.40	0.79	0.87	0.89
8.50	1.00	8.62	0.9	0.14	0.73	0.08	0.72	0.74	0.72

La recolección de los nodos se realizó cuando el robot ejecuta los movimientos rotacionales, y con la ayuda de la cuadrícula se determina fácilmente las coordenadas de los nodos.

Con cada nodo tomado, se calcula los desplazamientos necesarios para llegar al siguiente nodo para completar el camino, estos valores se procesan para determinar la desviación entre los nodos y el error del recorrido total producido por el Robotino®.

Tabla 4.36 Errores calculados por nodos medidos en el mapa de entorno

Desviación		Error de posición	
$V(x)$	$V(y)$	$e(x)$	$e(y)$
0	0	0.00%	0.00%
0.14	0.03	0.95%	0.59%
0.13	0.02	0.89%	0.41%
0.11	0.03	0.75%	0.62%

0.13	0.06	0.89%	1.79%
0.39	0.16	2.64%	5.67%
0.1	0.12	0.70%	4.56%
0.15	0.06	1.06%	2.35%
0.06	0.07	0.65%	2.75%
0.08	0.1	0.88%	4.15%
0.06	0.11	0.69%	6.79%
0.12	0.1	1.39%	11.11%

Las desviaciones obtenidas entre nodos, se observa que no supera $15cm$, a excepción de un punto, pero el Robotino® compensa para terminar con una desviación de $12cm$ y $10cm$ para los ejes x y y respectivamente.

Los errores de posición al final de la ruta se incrementan debido a que son referenciados con la posición final de la ruta, pero en términos absolutos, estos son pequeños presentando una respuesta muy cercana a la calculada por el método planificador y la recolectada por los encoders.

4.3.4 ANÁLISIS DE PRUEBAS DE CONTROL DE RUTAS

De igual manera que las pruebas aplicadas para ver el rendimiento de los métodos planificadores en el entorno real, en esta prueba se tiene resultados similares en función del tiempo de procesamiento, entonces se puede concluir que para los métodos basados en grafos el tiempo de procesamiento es prácticamente el mismo sin importar las posiciones de las configuraciones, pero en el método de Campos Potenciales, sí depende de la ruta a ser calculada.

El método Grafo de Visibilidad aplicado al entorno expandido del laboratorio, presentó aumento de la distancia de la ruta acercándose a la obtenida por los otros métodos planificadores, pero manteniendo la simplicidad en función del número de quiebres.

Desde el punto de vista del tiempo de ejecución del Robotino®. Las tres pruebas presentan tiempos similares en la ejecución siendo el tiempo más alto producido por el método de Campos potenciales (un 9% adicional), concluyendo que las

rutas obtenidas por los métodos planificadores son aplicables al Robotino® y su respuesta es similar en las tres respuestas.

Los errores calculados entre rutas de los métodos planificadores y la obtenida por los encoders del Robotino®, son muy pequeños menores al 1%, pero existe un incremento del error conforme el robot se acerca la configuración final, esta condición se debe a que al realizar el control de posición se presenta la acumulación del error que se presenta a un sistema que tiene memoria ya que la función de transferencia tiene un polo en el origen (Capítulo 3)

Las desviaciones producidas por el Robotino®, en función de la ruta teórica muestra que el robot se mueve de forma diferente a la que sus sensores presentan y que el programa procesa, pero no son significativos. Estos errores se producen debido a las imperfecciones del piso que los encoders no registran. Pero son aplicables a los tres métodos planificadores.

4.4 PRUEBAS DE CONTROL DE TRAYECTORIA DEL ROBOTINO® EN ENTORNO REAL

Las pruebas de control de trayectoria se realizan para los tres métodos planificadores desarrollados de la misma forma que se realizaron las pruebas de control de ruta, para lo cual se parte de las siguientes configuraciones:

$$P_0 : (8.5\ , 1.25\ )$$
$$P_f : (15\ , 5\ )$$

4.4.1 PRUEBA DE CONTROL DE TRAYECTORIA POR GRAFO DE VISIBILIDAD

Para realizar la prueba primero se calcula la ruta entre las configuraciones seleccionadas con el método Grafo de Visibilidad presentada en la Figura 4.37,

obteniendo una ruta de 4 nodos incluyendo las configuraciones, distancia de la ruta es $8.96m$, y el tiempo de ejecución del método planificador es $28.6s$.

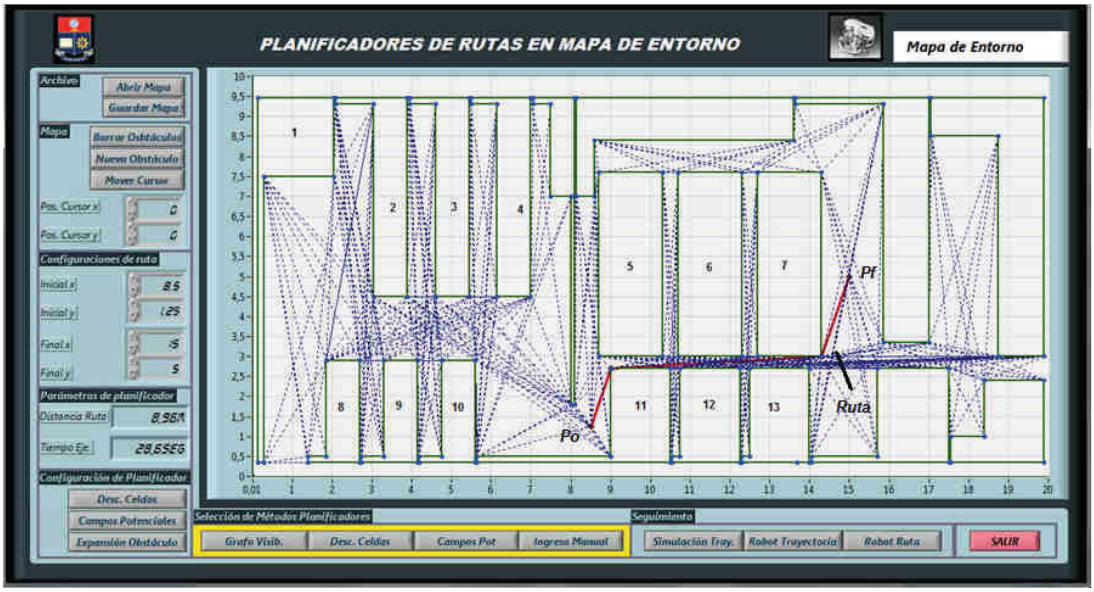


Figura 4.37 Ruta obtenida por Grafo de Visibilidad

Determinada la ruta, se procede a la ejecución del control de trayectoria para lo cual se conecta al robot con la dirección IP y el número de puerto (Figura 4.38). Adicionalmente se ingresa el tiempo de ejecución para completar el control que es $40s$.

Cuando se ingresa el tiempo de ejecución el programa construye la trayectoria a partir de ruta. Utilizando la interpolación y discretizando la trayectoria el HMI asigna automáticamente el tiempo al que pertenece cada nodo antes de empezar con el control para el robot y calcula las velocidades referenciales al control de trayectoria.

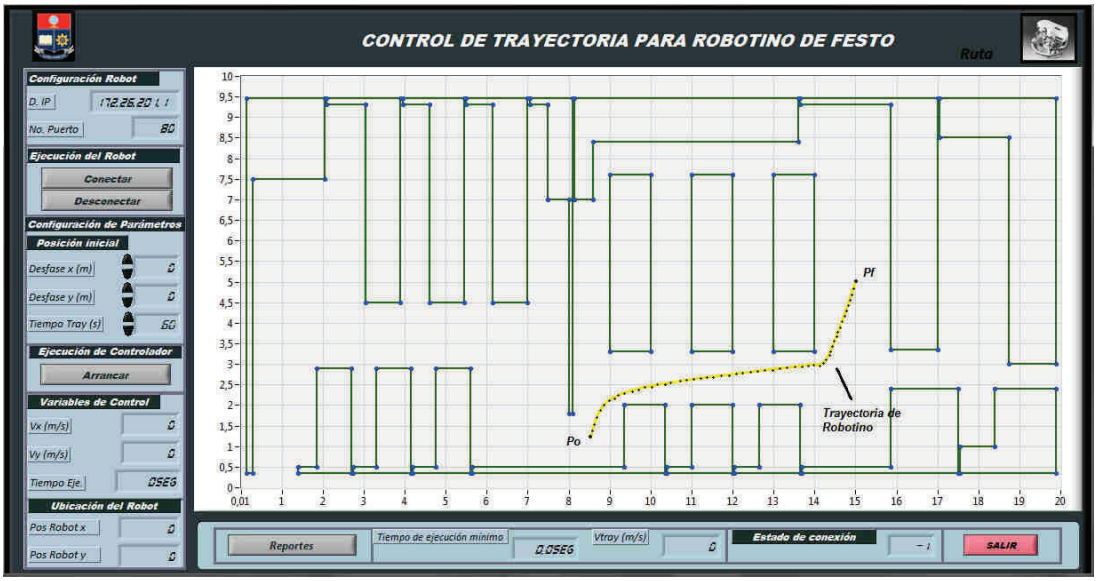


Figura 4.38 Respuesta obtenida por control de trayectoria para tiempo de $40s$

Al iniciar el control de trayectoria el HMI envía la velocidad de control hacia el robot y al mismo tiempo recibe la información de la posición del robot para actualizar la velocidad en función del periodo de muestreo T_0 del control. Terminado el control de trayectoria para la ruta obtenida por grafo de Visibilidad se toma la información del reporte del HMI y se presenta en la Tabla 4.37.

Tabla 4.37 Coordenadas obtenidas para el control de trayectoria por Robotino®, por Grafo de visibilidad.

	Trayectoria Teórica		Trayectoria Robotino		Tiempo Tray	Velocidad de Control		Error de Posición		
	Tray $x\,(m)$	**Tray** $y\,(m)$	**Tray R** $x(m)$	**Tray R** $y(m)$	t_s	Vc_x (m/s)	Vc_y (m/s)	**Error** $e_x\,(m)$	**Error** $e_y\,(m)$	**Error** $\lvert e\rvert\,(m)$
P_0	8.5	1.25	8.51	1.27	0	0.06	0.17	-0.01	-0.02	0.022
P_4	8.63	1.63	8.63	1.63	1.82	0.07	0.22	0	0	0.000
P_8	8.76	2.01	8.77	2	3.64	0.06	0.22	-0.01	0.01	0.014
P_{12}	8.89	2.38	8.9	2.38	5.45	0.07	0.22	-0.01	0	0.010
P_{16}	9.14	2.68	9.13	2.68	7.27	0.23	0.07	0.01	0	0.010
P_{20}	9.52	2.73	9.51	2.72	9.09	0.24	0.02	0.01	0.01	0.014
P_{24}	9.92	2.75	9.91	2.75	10.91	0.24	0.01	0.01	0	0.010
P_{28}	10.32	2.77	10.31	2.78	12.73	0.24	0	0.01	-0.01	0.014
P_{32}	10.72	2.8	10.71	2.8	14.55	0.24	0	0.01	0	0.010
P_{36}	11.12	2.82	11.11	2.83	16.36	0.23	0	0.01	-0.01	0.014

P_{40}	11.52	2.84	11.51	2.85	18.18	0.23	0	0.01	-0.01	0.014
P_{44}	11.92	2.87	11.91	2.87	20	0.24	0	0.01	0	0.010
P_{48}	12.31	2.89	12.32	2.9	21.82	0.21	0	-0.01	-0.01	0.014
P_{52}	12.71	2.91	12.71	2.91	23.64	0.23	0.01	0	0	0.000
P_{56}	13.11	2.93	13.11	2.94	25.45	0.24	0	0	-0.01	0.010
P_{60}	13.51	2.96	13.51	2.96	27.27	0.23	0	0	0	0.000
P_{64}	13.91	2.98	13.91	2.99	29.09	0.22	-0.01	0	-0.01	0.010
P_{68}	14.28	3.1	14.27	3.1	30.91	0.16	0.17	0.01	0	0.010
P_{72}	14.46	3.44	14.45	3.44	32.73	0.09	0.22	0.01	0	0.010
P_{76}	14.59	3.82	14.58	3.82	34.55	0.09	0.22	0.01	0	0.010
P_{80}	14.72	4.2	14.7	4.17	36.36	0.11	0.27	0.02	0.03	0.036
P_{84}	14.85	4.58	14.85	4.57	38.18	0.08	0.22	0	0.01	0.010
P_f	15	4.99	14.99	4.97	40	-0.5	-0.55	0.01	0.02	0.022

La Tabla 4.37 tiene la cuarta parte de los resultados en el reporte generado por el control de trayectoria. Cada muestra tiene la información acerca de la configuración de la trayectoria calculada, la medida que envía el robot, el tiempo a la que se alcanza la posición la velocidad de control y el error de posición entre la trayectoria calculada y la medida tanto en componentes como en la distancia.

El error de posición máximo producido por el control es $3.6cm$ en la muestra 80, la cual se corrige cuando se toma la siguiente muestra reduciendo su valor a $1cm$, y este se mantiene dentro de una rango de $2cm$ a lo largo de toda la trayectoria.

La trayectoria ejecutada por el robot se completa en el tiempo de ejecución asignado alcanzando la configuración final cuando el tiempo de ejecución es $40s$, lo cual se comprueba cuando el robot alcanza las parciales en el tiempo asignado.

Los datos obtenidos por el control de trayectoria son representados en curvas (Figura 4.39) que indican en forma visual el comportamiento del sistema de control.

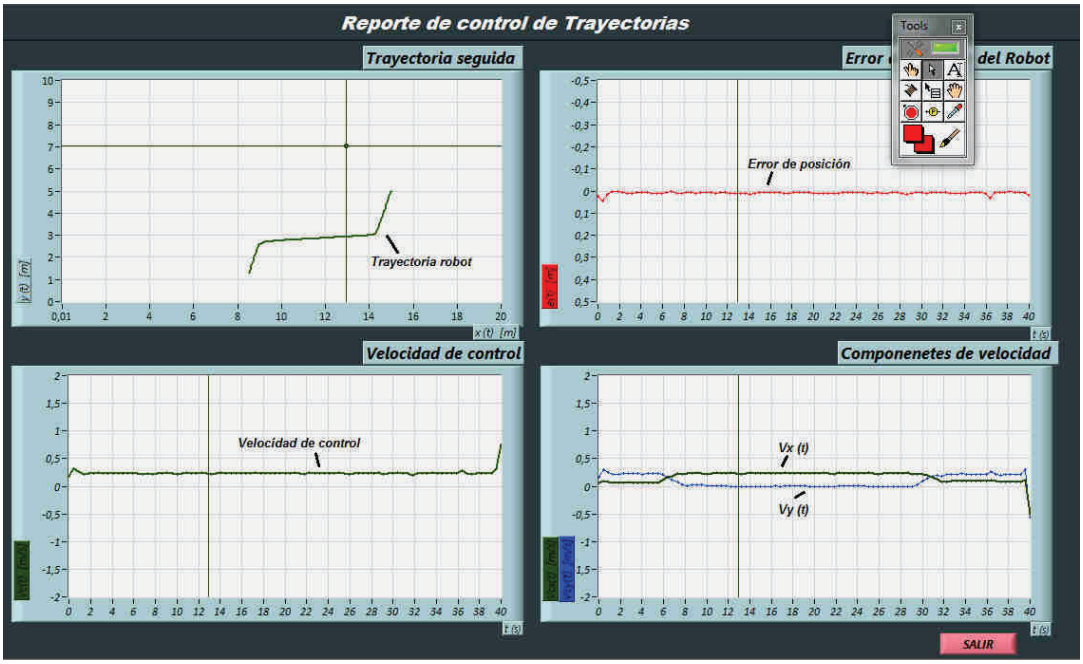


Figura 4.39 Curvas de variables de control de Trayectoria

Estas curvas presenta el comportamiento del error de posición $|\boldsymbol{e}|$, la trayectoria que sigue el robot, la velocidad de control $|\boldsymbol{Vc}|$ y sus componentes. La curva del error de la Figura 4.39 confirma los datos en el reporte de la Tabla 4.37 manteniendo su valor cercano a 0 y manteniéndose prácticamente constante a lo largo de la trayectoria.

Las componentes de la velocidad de control Vcx y Vcy, varían de acuerdo a los quiebres de la ruta obtenida por el método Grafo de Visibilidad, el comportamiento de la velocidad Vcx indica que existe poco desplazamiento del robot en esta coordenada en dos partes de la trayectoria, lo cual confirma la trayectoria, y entre los $6s$ y $30s$ existe un marcado desplazamiento en el eje x, caso contrario ocurre con la velocidad Vcy donde es representativa en el tramo comprendido entre $0s$ y $6s$, y entre los $30s$ y $40s$. Este comportamiento de las componentes de la velocidad evidencia que la velocidad de control $|\boldsymbol{Vc}|$ es constante la cual ratifica la curva la correspondiente curva de la Figura 4.39 con un sobre pico al iniciar el control que se estabiliza en $1s$ de iniciar el control.

4.4.2 PRUEBA DE CONTROL DE TRAYECTORIA POR DESCOMPOSICIÓN DE CELDAS

Tomando en cuenta las mismas configuraciones y mapa de entorno utilizado en la prueba anterior, se ejecuta el método de Descomposición de Celdas modificado obteniendo la ruta presentada en la Figura 4.40.

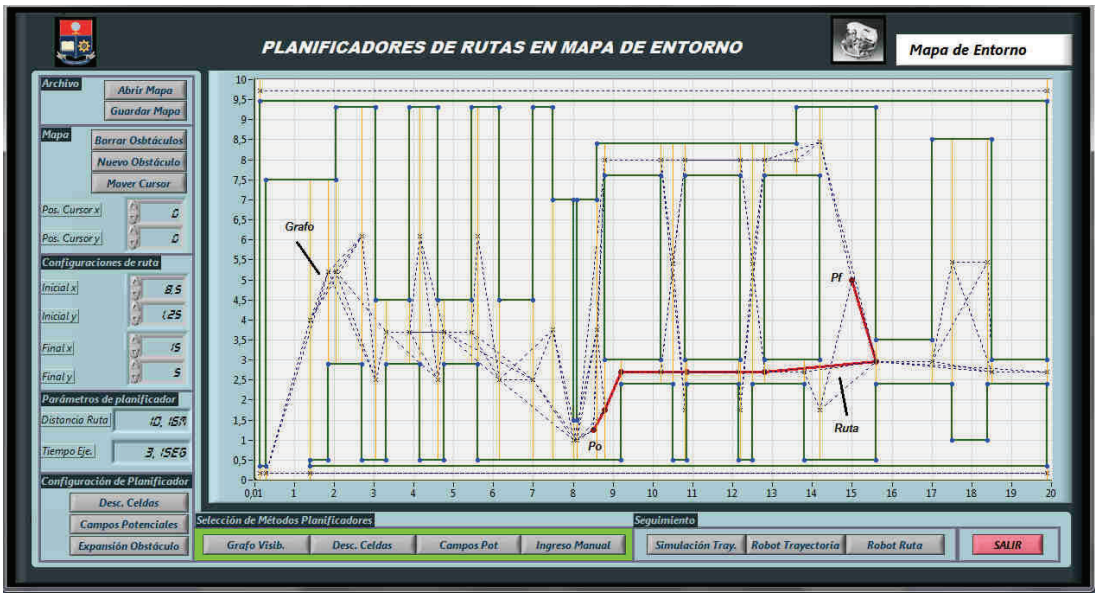


Figura 4.40 Respuesta de por Descomposición de Celdas Modificado

La ruta generada por el método planificador tiene 5 nodos en total. La distancia es $10.16m$, y el tiempo de procesamientos $3s$.

Ejecutando la ruta dentro de la rutina control de trayectoria, se construye la trayectoria configurada para $40s$, y el HMI controla al robot para ejecutar la trayectoria, obteniendo la siguiente respuesta del Robotino®, representado en la Figura 4.41.

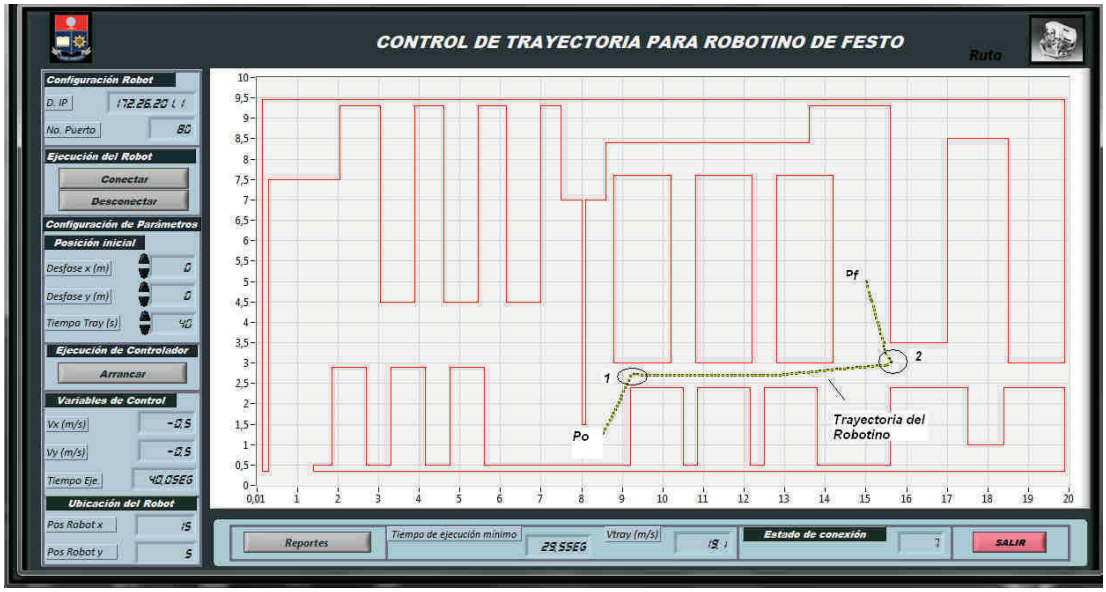


Figura 4.41 Respuesta obtenida por control de trayectoria para tiempo de $40s$

La curva punteada de la Figura 4.41, es la trayectoria ejecutada por el robot en $40s$, y la información de las variables del sistema de control se presentan en la Tabla 4.38.

Tabla 4.38 Resultados de control de ruta para un tiempo de ejecución de $40s$

	Trayectoria Teórica		Trayectoria Robotino		Tiempo Tray	Velocidad de Control		Error de Posición		
	Tray $x\,(m)$	**Tray** $y\,(m)$	**Tray R** $x(m)$	**Tray R** $y(m)$	t_s	Vc_x (m/s)	Vc_y (m/s)	**Error** $e_x\,(m)$	**Error** $e_y\,(m)$	**Error** $\lvert e\rvert\,(m)$
P_0	8.5	1.25	8.5	1.26	0	0.12	0.18	0	-0.01	0.010
P_5	8.76	1.68	8.75	1.68	1.98	0.13	0.22	0.01	0	0.010
P_{10}	8.96	2.13	8.96	2.12	3.96	0.11	0.25	0	0.01	0.010
P_{15}	9.15	2.59	9.15	2.58	5.94	0.13	0.24	0	0.01	0.010
P_{20}	9.59	2.7	9.58	2.69	7.92	0.27	0.02	0.01	0.01	0.014
P_{25}	10.09	2.7	10.08	2.7	9.9	0.28	0	0.01	0	0.010
P_{30}	10.59	2.7	10.58	2.7	11.88	0.28	0.01	0.01	0	0.010
P_{35}	11.09	2.7	11.08	2.7	13.86	0.28	-0.01	0.01	0	0.010
P_{40}	11.59	2.7	11.58	2.7	15.84	0.27	0	0.01	0	0.010
P_{45}	12.09	2.7	12.08	2.7	17.82	0.27	0	001	0	0.010
P_{50}	12.59	2.7	12.59	2.7	19.8	0.27	0	0	0	0.000
P_{55}	13.08	2.72	13.07	2.72	21.78	0.27	0.02	0.01	0	0.010
P_{60}	13.58	2.77	13.57	2.78	23.76	0.26	0.01	0.01	-0.01	0.014

P_{65}	14.07	2.81	14.07	2.82	25.74	0.28	0	0	-0.01	0.010
P_{70}	14.57	2.86	14.57	2.87	27.72	0.27	0	0	-0.01	0.010
P_{75}	15.07	2.9	15.07	2.91	29.7	0.27	0	0	-0.01	0.010
P_{80}	15.57	2.96	15.53	2.96	31.68	0.12	0.23	0.04	0	0.040
P_{85}	15.46	3.43	15.45	3.4	33.66	-0.05	0.32	0.01	0.03	0.032
P_{90}	15.32	3.91	15.32	3.91	35.64	-0.06	0.25	0	0	0.000
P_{95}	15.18	4.39	15.18	4.39	37.62	-0.07	0.26	0	0	0.000
P_{100}	15.04	4.87	15.03	4.89	39.6	-0.08	0.26	0.01	-0.02	0.022
P_f	15	4.99	15	5	40	-0.5	-0.5	0	-0.01	0.010

El error generado entre la trayectoria construida a partir de la ruta del método planificador, y la leída por los encoders del robot tiene resultados similares al obtenido por la prueba anterior donde el error máximo es $4cm$, cuando el robot ejecuta el cambio de dirección en el segundo quiebre significativo presentado en la Figura 4.41, que es corregido posteriormente por el sistema de control.

La trayectoria que sigue el robot se completa en el tiempo establecido aunque las velocidades de control son superiores a las calculadas en la prueba anterior, este comportamiento es debido a que la distancia de recorrido del robot es mayor.

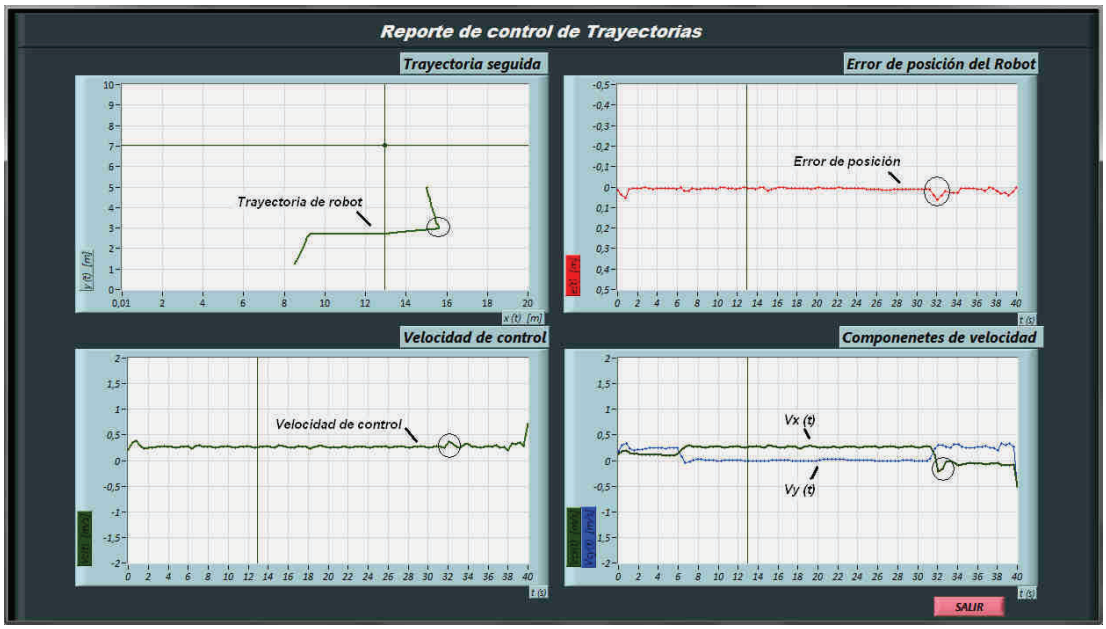


Figura 4.42 Curvas de las variables de control de Trayectoria

La curva del error de posición de la Figura 4.42, indica que el error se mantiene cercano a 0, pero con un pico a los $32s$, generado por el segundo quiebre de la ruta y que el controlador corrige disminuyendo la velocidad Vcx. La curva de la velocidad empieza con un sobre pico que se estabiliza, hasta que llega al quiebre donde corrige este valor disminuyendo su valor para reducir el error de posición.

4.4.3 PRUEBA DE CONTROL DE TRAYECTORIA POR CAMPOS POTENCIALES

Ejecutando el método Campos potenciales sobre el mismo mapa de entorno y configuraciones el HMI presenta la siguiente ruta calculada.

Figura 4.43 Respuesta de por Campos Potenciales

La ruta generada por el método planificador de la Figura 4.43 tiene 11 nodos incluyendo las configuraciones inicial y final, el tiempo de ejecución es de $5.1s$ y la distancia de la ruta es de $9.27m$.

Ejecutando el control de trayectoria, el HMI construye la trayectoria a partir de la ruta calculada por el método planificador con el tiempo ingresado de $40s$, para

luego realizar el control de trayectoria sobre el Robotino®, obtenido la siguiente respuesta:

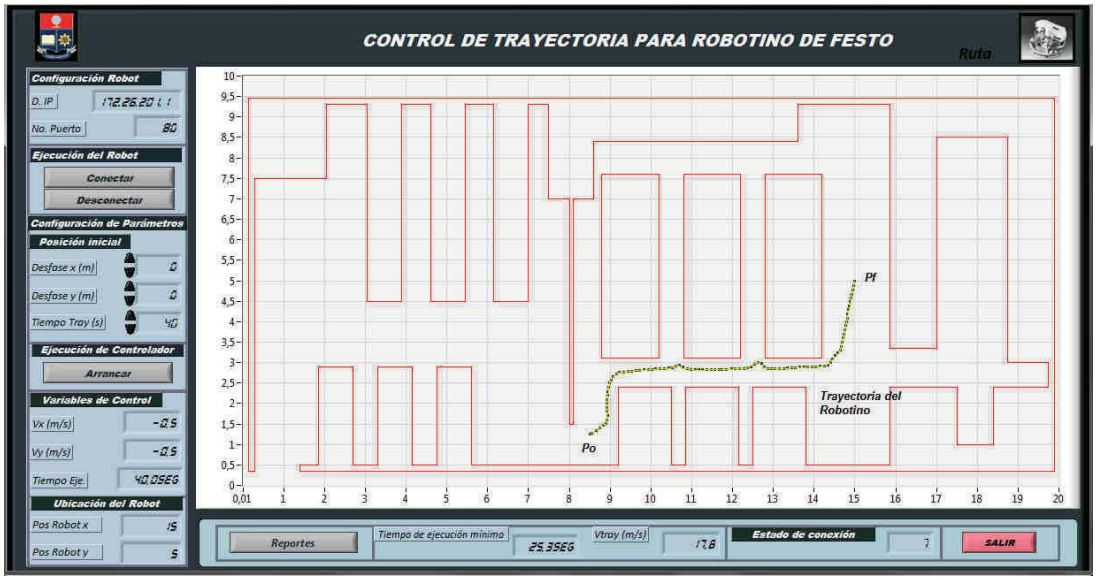


Figura 4.44 Respuesta obtenida por control de trayectoria para tiempo de $40s$

La línea punteada es la trayectoria leída y presentada en el HMI (Figura 4.44) cuando el robot culminó con el seguimiento de la trayectoria, la tabla 4.39 presenta la información de las variables de control del sistema.

Tabla 4.39 Resultados de control de ruta para un tiempo de ejecución de $40s$

	Trayectoria Teórica		Trayectoria Robotino		Tiempo Tray	Velocidad de Control		Error de Posición		
	Tray $x\,(m)$	**Tray** $y\,(m)$	**Tray R** $x(m)$	**Tray R** $y(m)$	t_s	Vc_x (m/s)	Vc_y (m/s)	**Error** $e_x\,(m)$	**Error** $e_y\,(m)$	**Error** $\lvert e\rvert\,(m)$
P_0	8.5	1.25	8.5	1.25	0	0.19	0.11	0	0	0.000
P_4	8.84	1.46	8.84	1.46	1.7	0.13	0.13	0	0	0.000
P_8	8.93	1.81	8.93	1.8	3.4	0	0.25	0	0.01	0.010
P_{12}	8.94	2.2	8.94	2.19	5.11	0.04	0.26	0	0.01	0.010
P_{16}	9.03	2.58	9.02	2.58	6.81	0.15	0.18	0.01	0	0.010
P_{20}	9.34	2.76	9.34	2.77	8.51	0.24	0.01	0	-0.01	0.010
P_{24}	9.73	2.81	9.72	2.81	10.21	0.25	0	0.01	0	0.010
P_{28}	10.13	2.84	10.12	2.84	11.91	0.26	0	0.01	0	0.010
P_{32}	10.52	2.88	10.52	2.88	13.62	0.23	0.07	0	0	0.000
P_{36}	10.88	2.84	10.88	2.85	15.32	0.23	-0.05	0	-0.01	0.010
P_{40}	11.27	2.82	11.27	2.82	17.02	0.25	-0.01	0	0	0.000

P_{44}	11.67	2.83	11.68	2.83	18.72	0.22	0	-0.01	0	0.010
P_{48}	12.07	2.84	12.07	2.84	20.43	0.24	0	0	0	0.000
P_{52}	12.46	2.91	12.45	2.9	22.13	0.23	0.13	0.01	0.01	0.014
P_{56}	12.78	2.87	12.79	2.89	23.83	0.18	-0.07	-0.01	-0.02	0.022
P_{60}	13.15	2.85	13.15	2.85	25.53	0.26	0	0	0	0.000
P_{64}	13.55	2.87	13.55	2.88	27.23	0.26	0	0	-0.01	0.010
P_{68}	13.95	2.89	13.94	2.9	28.94	0.26	0	001	-0.01	0.014
P_{72}	14.34	2.95	14.34	2.94	30.64	0.14	0.16	0	0.01	0.010
P_{76}	14.6	3.24	14.58	3.24	32.34	0.16	0.16	0.02	0	0.020
P_{80}	14.71	3.6	14.7	3.6	34.04	0.08	0.24	0.01	0	0.010
P_{84}	14.79	4	14.79	3.99	35.74	0.06	0.24	0	0.01	0.010
P_{88}	14.87	4.39	14.87	4.39	37.45	0.05	0.23	0	0	0.000
P_{92}	14.95	4.78	14.95	4.77	39.15	0.06	0.25	0	0.01	0.010
P_f	15	5	15	4.99	40	-0.5	-0.5	0	0.01	0.010

El error de posición de la trayectoria presentado en la Tabla 4.39 está dentro del rango de los $1cm$ con un pico de $2.2cm$ en el nodo 56, corregido posteriormente por el sistema de control, indicando que el robot sigue la trayectoria de las muestras recolectadas.

Con la información recolectada en los reportes, se procede a trazar las curvas de las variables del sistema de control presentadas en la Figura 4.45.

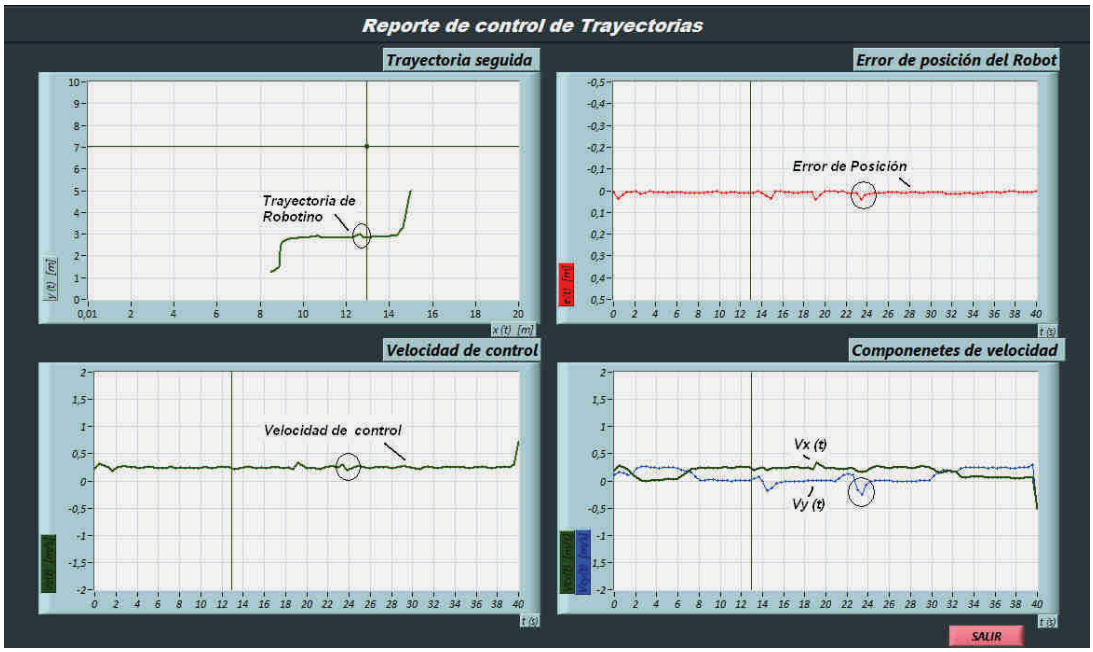


Figura 4.45 Curvas de las variables de control de Trayectoria

La curva del error de posición indica que se mantiene constante y prácticamente 0 pero tiene picos de $2cm$ que son corregidos inmediatamente. Los errores son generados por los cambios de dirección de la trayectoria en $19s$ y $23s$ donde se observa cambios de la componente y de la velocidad de control y cuantificada en la Tabla 4.39.

La velocidad de control también se mantiene constante y estable a excepción del error donde el controlador compensa la velocidad y disminuye el error de posición.

Al completar el control se observa que el robot llega a la configuración final en el tiempo establecido.

4.4.4 PRUEBA DE CONTROL DE TRAYECTORIA CON CORRECCIÓN DE ERROR DE POSICIÓN INICIAL

La última parte de control de trayectoria se analiza el seguimiento de la trayectoria del robot y el comportamiento del controlador para el seguimiento de una trayectoria cuando el robot parte desde una configuración diferente a la configuración inicial.

La ruta utilizada es la calculada por el método Grafo de Visibilidad (Figura 4.37), donde la variable a cambiar es la posición real en la que está el robot.

El HMI de control de trayectoria tiene la opción de configurar el error de posición en componentes. La configuración inicial real del Robotino® es:

$$P_0 : (8.14\ , 1.25\ )$$

El error de posición inicial es:

$$e_{x0} = 36cm$$
$$e_{y0} = 0cm$$

Con el ingreso del error al HMI se procede a ejecutar el control de trayectoria para un tiempo de ejecución de $40s$ obteniendo la siguiente curva representada en la Figura 4.46.

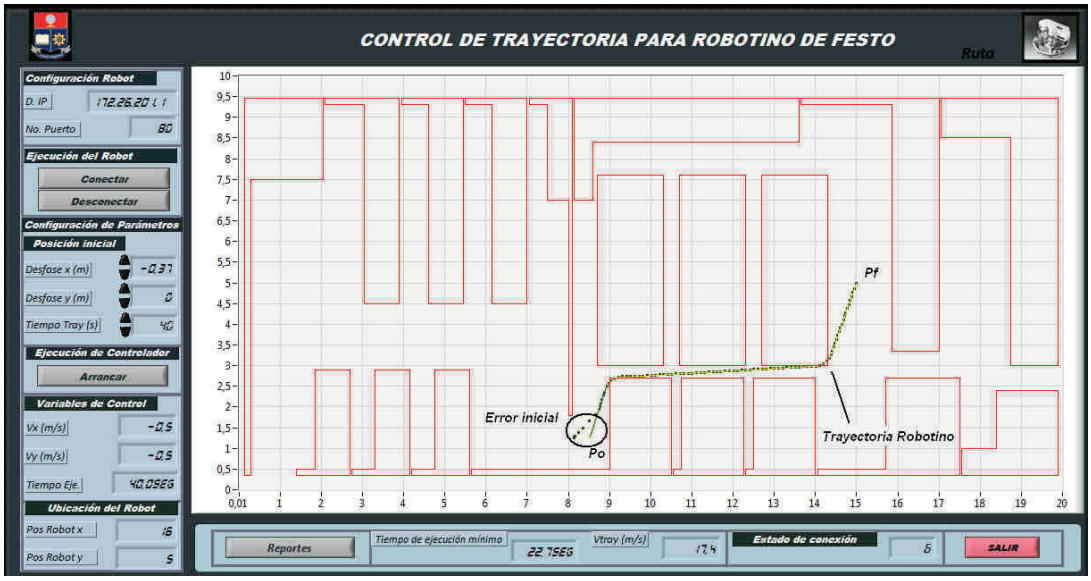


Figura 4.46 Respuesta obtenida por control de trayectoria para tiempo de $40s$

La Figura 4.46 presenta el error inicial formado por la trayectoria formada por el robot (línea punteada) y la calculada por el método planificador. Este error disminuye conforme se ejecuta el control, el robot alcanza rápidamente la trayectoria en un espacio menor de $1m$ calculada, y realiza el seguimiento normal hasta concluirla.

Con el control completo se recoge la información del sistema de control y se presenta en la Tabla 4.40.

Tabla 4.40 Coordenadas obtenidas para el control de trayectoria con error de posición

	Trayectoria Teórica		Trayectoria Robotino		Tiempo Tray	Velocidad de Control		Error de Posición		
	Tray $x\,(m)$	**Tray** $y\,(m)$	**Tray R** $x(m)$	**Tray R** $y(m)$	t_s	Vc_x (m/s)	Vc_y (m/s)	**Error** $e_x\,(m)$	**Error** $e_y\,(m)$	**Error** $\lvert e\rvert\,(m)$
P_0	8.5	1.25	8.14	1.27	0	0	0	0.36	-0.02	0.361
P_1	8.53	1.34	8.13	1.25	0.45	0.11	0.11	0.4	0.09	0.410
P_2	8.57	1.44	8.16	1.28	0.91	0.19	0.19	0.41	0.16	0.440
P_3	8.6	1.53	8.23	1.35	1.36	0.25	0.25	0.37	0.18	0.411
P_4	8.63	1.63	8.33	1.46	1.82	0.3	0.3	0.3	0.17	0.345
P_5	8.66	1.72	8.45	1.59	2.27	0.33	0.33	0.21	0.13	0.247
P_6	8.7	1.82	8.58	1.73	2.73	0.33	0.36	0.12	0.09	0.150
P_7	8.73	1.91	8.73	1.91	3.18	0.07	0.22	0	0	0.000
P_{12}	8.89	2.38	8.89	2.38	5.45	0.09	0.21	0	0	0.000
P_{17}	9.23	2.71	9.22	2.72	7.73	0.22	0	0.01	-0.01	0.014
P_{22}	9.72	2.74	9.71	2.74	10	0.23	0	0.01	0	0.010
P_{27}	10.22	2.77	10.22	2.78	12.27	0.23	0	0	-0.01	0.010
P_{32}	10.72	2.8	10.72	2.81	14.55	0.21	-0.01	0	-0.01	0.010
P_{37}	11.22	2.83	11.21	2.83	16.82	0.23	0	0.01	0	0.010
P_{42}	11.72	2.85	11.71	2.86	19.09	0.24	-0.01	0.01	-0.01	0.014
P_{47}	12.21	2.88	12.21	2.89	21.36	0.23	0	0	-0.01	0.010
P_{52}	12.71	2.91	12.71	2.92	23.64	0.23	-0.01	0	-0.01	0.010
P_{57}	13.21	2.94	13.21	2.94	25.91	0,23	0	0	0	0.000
P_{62}	13.71	2.97	13.71	2.97	28.18	0.24	0	0	0	0.000
P_{67}	14.2	3.04	14.19	3.04	30.45	0.2	0.14	0.01	0	0.010
P_{72}	14.46	3.44	14.44	3.44	32.73	0.1	0.22	0.02	0	0.020
P_{77}	14.62	3.92	14.61	3.91	35	0.09	0.22	0.01	0.01	0.014
P_{82}	14.79	4.39	14.77	4.36	37.27	0.1	0.28	0.02	0.03	0.036
P_{87}	14.95	4.86	14.94	4.83	39.55	0.12	0.37	0.01	0.03	0.032
P_f	15	4.99	14.99	4.97	40	-0.45	-0.45	0.01	0.02	0.022

La Tabla 4.40 tiene las 8 primeras muestras de las variables para ver la compensación del sistema de control, donde indica que error de posición inicial es $36cm$, y esta aumenta a los $41cm$ para la siguiente muestra conforme el controlador acelera al robot suavemente hasta alcanzar una velocidad de $0.45m/s$ a los $3.18s$ donde el error llega a ser 0 en la séptima muestra de la tabla.

Cuando el sistema de control llega a estabilizarse el error de posición llega a ser $3.6cm$, pero fluctúa alrededor de $1cm$. El robot también alcanza la configuración final cuando se cumple el tiempo de ejecución de trayectoria.

Con los valores de las variables de control se procede trazar sus correspondientes curvas representadas en la Figura 4.47.

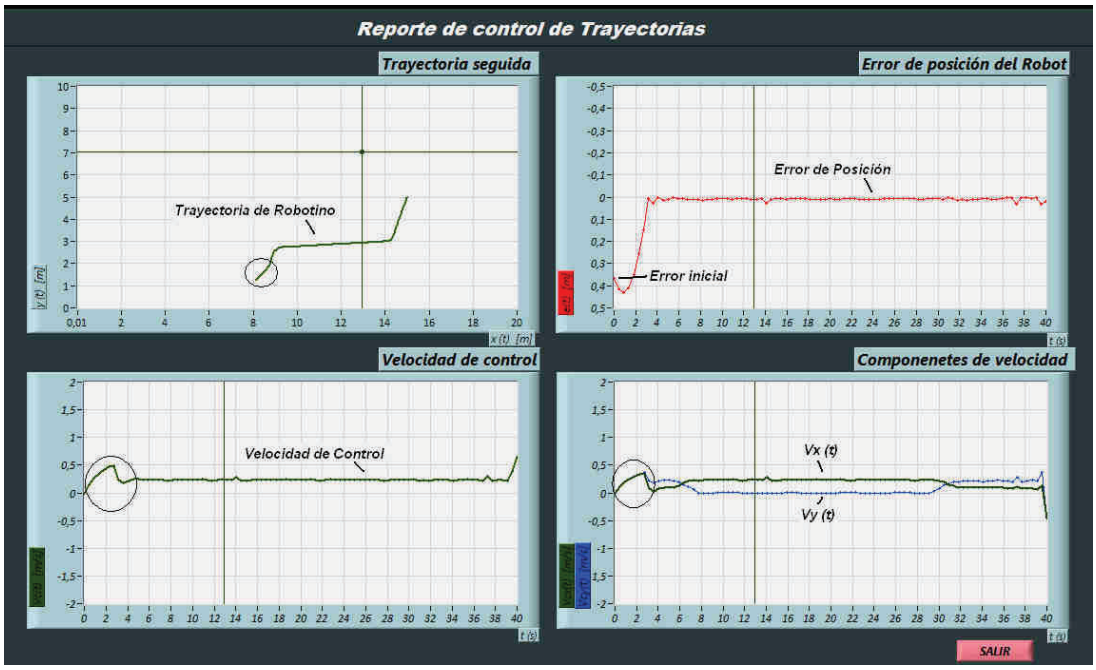


Figura 4.47 Curvas de las variables de control de Trayectoria

La curva del error de posición de la Figura 4.47 tiene un pico del error que rápidamente se reduce, para luego mantenerse estable y cercano a 0. Al incorporarse una función que suavice la velocidad conjuntamente con la saturación a un valor máximo permitido, la curva de velocidad indica que el robot es acelerado durante el mismo intervalo de tiempo donde existe el error de posición la cual se disminuye y se estabiliza cuando el robot se encuentra sobre la trayectoria y se mantiene prácticamente constante durante el resto de la trayectoria.

CAPÍTULO 5

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La navegación de los robots móviles constituye ser el objetivo para alcanzar la autonomía de estos, para lo cual hay que tener en cuenta las características y restricciones que tiene el robot utilizado para la navegación. El presente trabajo plantea diferentes conceptos de planificación de rutas, además de presentar dos alternativas para el seguimiento de la misma considerando un robot omnidireccional.

- Los robots omnidireccionales tienen prestaciones que benefician la circulación en entornos estrechos versus los robots con restricción no holonómicas, ya que ellos pueden desplazarse en varias direcciones sin cambiar su orientación en detrimento de capacidad de carga e irregularidades del piso.

- El modelo matemático constituye una herramienta que determina el diseño del controlador utilizado, en el caso particular de esta tesis el modelo cinemático aplicada a un robot omnidireccional determina que el controlador por métodos numéricos se simplifica a una ecuación vectorial.

- Los métodos planificadores basados en la construcción grafos dependen de la complejidad del mapa de entorno tal como se analizó en las pruebas de determinación de rutas que los tiempos de ejecución pueden llegar a ser muy elevados dependiendo del computador utilizado, pero en virtud de este aspecto se puede obtener rutas que sean más simples en el sentido de tener un menor número de quiebres que es beneficioso cuando se realiza el seguimiento con un robot que tenga restricciones no holonómicas

- Los algoritmos de búsqueda de rutas (Dijkstra y Algoritmo A*), tiene un papel importante para calcular la ruta, en los métodos planificadores basados en la construcción de grafos de conectividad. Para la selección de estos buscadores de rutas dependería de las necesidades requerimientos del sistema, como construir una ruta con el menor número de quiebres, la distancia más corta entre las configuraciones, ó tiempo de procesamiento para encontrar la ruta.

- Una de las características del método por Campos Potenciales cuando se aplica de forma iterativa es la obtención de una ruta que tiene gran cantidad de nodos, que pueden ser innecesarios en el sentido que estos forman parte de una recta, por lo que requieren de un post procesamiento de la ruta para eliminar nodos y quiebres que no brindan información relevante a la ruta, y así disminuir tiempo de procesamiento cuando se realice el seguimiento con un robot.

- Al desarrollar la programación, la mejor opción es la implementación del programa de forma estructurada partiendo el programa en subrutinas que manejan aspectos en común de los métodos planificadores como en el caso de la identificación de obstáculos que permite construir los grafos de conectividad, que constituye la base de los métodos de Grafo de Visibilidad y Descomposición de Celdas

- En la implementación de los algoritmos basados en grafos, el diagrama de flujo diseñado presentó pequeñas modificaciones para obtener resultados totalmente diferentes, y sus variaciones radican en la determinación de los nodos que constituyen el grafo de conectividad.

- Aplicando el concepto físico de distribución continua de carga, y utilizando expresiones exponenciales para calcular la fuerza de atracción y repulsión el método planificador de Campos Potenciales adquiere la propiedad de mantener constante la fuerza de atracción como indica la Figura 3.52, generando una ruta con nodos uniformemente separados, y evitando los

obstáculos con curvas suaves cuando actúa la fuerza repulsiva sin ocasionar desplazamientos irregulares.

- Otro beneficio que brinda la fuerza de repulsión conjuntamente con la rutina de construcción de Sólidos es no considerar las dimensiones y geometría de los obstáculos ya que estos forman un campo de repulsión radial alrededor de sí mismo (Figura 3.59) que se pierde rápidamente cuando se aleja del obstáculo. La característica radial de la fuerza de repulsión premie utilizar la técnica de seguimiento de pared sin la incorporación de obstáculos virtuales para realizar el seguimiento ó salir de mínimos locales.

- El sistema de control de ruta aplicado al Robotino®, segmenta el seguimiento de la ruta en desplazar al robot paso a paso por medio de los nodos que compone la ruta utilizando el control de posición y rotando al robot cuando de quiera llegar a la siguiente manteniendo la parte frontal del Robotino® hacia adelante, por ende este control no tiene límite de tiempo para completar la ruta.

- El control de trayectoria utilizando el método numérico requiere tener la velocidad de la trayectoria y el error de posición entre el robot y la trayectoria en cada periodo de muestreo, lo que implica construir la trayectoria a partir de la ruta con requerimientos de velocidades y aceleraciones dentro de rangos permitidos para controlar al robot.

- La interpolación Spline y la posterior discretización manteniendo la distancia contante entre las muestras en función del periodo de muestreo permite mantener la rapidez de la trayectoria constante a lo largo de toda la trayectoria discreta formada y por debajo del límite máximo que un robot omnidireccional puede seguir minimizando el tiempo de ejecución el control para que el robot alcance la configuración final.

- Durante la ejecución de la primera y segunda prueba de los métodos planificadores, se observó que el método de grafo de Visibilidad presentó los mejores resultados y con un tiempo de procesamiento muy bajo (menor a $1s$), ratificando que el método resulta eficaz para mapas de entorno que tienen pocos obstáculos, a diferencia del método de Campos Potenciales que calcula una ruta con curvas suaves rodeando los obstáculos pero con tiempo invertido mayor

- En un mapa de entorno que tiene obstáculos escalonados el método por Descomposición de Celdas resulta poco eficiente ya que se crean gran cantidad de quiebres debido a que el método trata de evitar los obstáculos trazando las rectas verticales que nacen en los vértices de los mismo obstáculos, y la construcción de grafo de conectividad se basa en el enlazamiento de los nodos adyacentes no se forman más rutas que impidan estos quiebres.

- Realizando una variante del método Descomposición de Celdas agregando enlaces entre nodos vecinos no adyacentes, se forman pequeños grafos locales que eliminan los quiebres por completo provocados por mapas escalonados, llegando a ser el método con una ruta simple y un tiempo de ejecución muy bajo en mapas de entorno, por lo que al realizar variantes utilizadas en otros métodos mejora el rendimiento del método planificador invirtiendo tiempo de procesamiento mucho menor que el método de grafo de visibilidad.

- El método de Campos Potenciales, es el más estable en tiempo de procesamiento de los tres en función de la complejidad del mapa de entorno y adicionalmente el plus que presenta es que no se necesita considere un entorno expandido para ser aplicado al control de trayectoria, ya que en el mismo dimensionamiento de las expresiones de fuerza ya se incluye el rango seguro para que un robot circule considerando sus dimensiones.

- Considerando el tipo de robot utilizado para el seguimiento de trayectoria, el método de campos potenciales al proporcionar una ruta que tiene curvas suaves resulta el más adecuado cuando el robot utilizado no es un robot omnidireccional.

- La ventaja que presenta el método de Grafo de Visibilidad es el cálculo de rutas que tiene la menor distancia en un mapa de entorno agregando que es eficaz cuando el mapa de entorno no es muy complejo y con el concepto de entorno expandido, resulta ser el método idóneo para cálculo de rutas cuando existen obstáculos dispersos.

5.2 RECOMENDACIONES

- Para la utilización del programa, lo más importante es tener presente las condiciones en que se encuentre el Robotino®, como la configuración de la red y cobertura, además de aspectos físicos, como verificar el estado de las ruedas, calibraciones para los encoders, estado de las baterías y otros aspectos que determinan el correcto funcionamiento del Robotino®, y por ende el rendimiento del programa.

- A Nivel de programación, se recomienda cuando se incorpore un algoritmo que tenga una función crítica en el desarrollo del programa, se lo haga utilizando subrutinas y que las variables de estas subrutinas sean estandarizadas, esto ayuda a que realizando unos pequeños cambios al programa se obtengan resultados significativos al rendimiento del programa terminado.

- Al implementar un algoritmo o proceso, en un programa computacional, se recomienda tomar en cuenta otros aspectos adicionales que brinden al usuario del programa la libertad de realizar calibraciones a los sistemas, además de presentar la información necesaria que permita sacar conclusiones del funcionamiento para mejorarlo y optimizarlo.

- Es recomendable en el diseño de este tipo de sistemas dar opciones se simulación del sistema implementado en caso que no se cuente con el robot y que se requiera analizar el rendimiento del controlador en diferentes escenarios.

REFERENCIAS BIBLIOGRÁFICAS

[1] S. Martínez y R. Sisto, Control y Comportamiento de Robots Omnidireccionales, Montevideo: Facultad de Ingeniería - Universidad de la República, 2009.

[2] Robotnik, «XT - Terabot,» 2011. http://www.robotnik.es/ilsupload/TX-TERABOT.pdf.

[3] O. Diegel, A. Badve, G. Bright, J. Potgiter y S. Tlale, Improved Mecanum Wheel Design for Omni-directional Robots, Auckland: Mechatronics and robotics Researh Group, Massey University, November 2002.

[4] Wikipedia, «Mecanum Wheel,» enero 2014. http:/en.wikipedia.org/wiki/Mecanum wheel.

[5] Personal de Robótica y Educativa, «Robot de Televigilancia y Telepresencia,» Febrero 2009. http:/www.robotica-personal.es/2009/02/rovio-es-un-robot-wi-fi-con-una-webcam.html.

[6] ROS.org, «Robots Using ROS: Robotino,» Marzo 2013. http:/www.ros.org/news/2010/06/robots-using-ros-robotino.html.

[7] J.-B. Song y K.-S. Byun, Design and control of an omnidireccional mobile robot with steerable omnidirectional wheel, Germany/ARS, Austria, December 2006.

[8] E. Bricks, «Sistemas Holonómicos,» Julio 2010. http:/blog.electricbricks.com/2010/07/holonomic-robot/.

[9] V. Muñoz, G. Gil-Gómez y A. García, Modelado Cinemático y Dinámico de un robot Móvil Omnidireccional, España: Universidad de Málaga, 2003.

[10] Festo, *Manual Robotino®,* 2007.

[11] C. Mora, M. Armesto y J. Tornero, Sistema de Navegación de Robots Móviles en Entornos Industriales, España: Universidad Politécnica de Valencia , Septiembre 2004.

[12] D. González, Generación Automática de Mapas en Espacios Cerrados Mediante Robots Móviles, España: Universidad Autónoma de Madrid, Septiembre 2011.

[13] A. Ollero, Planificación de Trayectorias para Robots Móviles, España: Universidad de Málaga, 1995.

[14] D. López, Nuevas Aportaciones en Algoritmos de Planificación para la Ejecución de Maniobras en Robots Autónomos no Holónomos, España: Universidad de Huelva, Junio 2011.

[15] G. Ostos, E. Peña, D. Cardozo y G. Aristizabal, Análisis y Diseño de Algoritmos, Universidad Piloto de Colombia; Bogotá DC, Noviembre 2010.

[16] J. Moreno y S. Ordóñez, «Diagramas Voronoi de Alcance Limitado,» Junio 2009.
http://www-ma2.upc.edu/~geoc/DVALon/MemoriaDVALon.pdf.

[17] C. Códova, Electromagnetismo Teoría y Trabajo, Quito: Escuela Politécnica Nacional, 2013.

[18] Wikipedia, «Algoritmo Dijkstra,» Julio 2014.
http://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra.

[19] Wikipedia, «A* Search Algorithm,» July 2014.
http://en.wikipedia.org/wiki/A*_search_algorithm.

[20] A. Rosales, G. Scaglia, V. Mut, F. Di Sciascio, Control Dinámico Mediante Métodos Numéricos Para Robots Móviles Tipo Uniciclo, San Juan.

[21] A. Cruz, Planificación de Trayectorias en Sistemas Multirrobot, Universidad de Málaga, 2004.

[22] E. Ramos, Control Punto a Punto Para Seguimiento de Trayectorias de un Robot Movil de ruedas Tipo Diferencial, Instituto Técnico Nacional, Mexico 2011, 2011.

[23] J. Mathews, Métodos Numéricos con Matlab, Prentice Hall, 2000.

[24] Wikipedia, «Intepolación Poligonal,» 10 2014.
http://es.wikipedia.org/wiki/Interpolaci%C3%B3n_polin%C3%B3mica.

ANEXO A

MANUAL DE FUNCIONAMIENTO

A.1 REQUERIMIENTOS DE SISTEMA

- El HMI funciona bajo la plataforma de programación LabVIEW 2010, compatible con Windows XP, vista, 7 y 8.

- Instalación de Librería Robotics Module, y drives de control de Robotino® compatible con LabVIEW 2010.

- Resolución recomendada $1280x800$ o superior para programa ***Control VZ***.

- Conexión Wireless, o LAN con un dispositivo de red inalámbrica.

- De preferencia tener instalado Microsoft Office.

Nota: La presentación y ejecución del HMI es óptima con sistema operativo vista o posterior.

Nota: En caso que se utilice el programa como simulador, se puede obviar dispositivos de conexión LAN o WLAN.

A.2 FUNCIONAMIENTO DEL PROGRAMA

El programa ***Control VZ*** se encuentra compactado en una carpeta creada por LabVIEW 2010 (***control VZ.LLB***) Figura A.1.

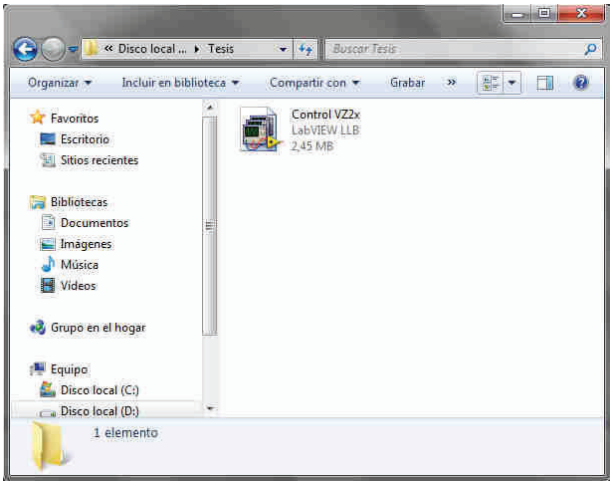


Figura A.1 Ícono del programa de planificadores de rutas

Para ejecutar el programa se hace doble clic sobre el ícono ***Control VZ***.

> **Nota:** Este programa puede ejecutarse desde cualquier lugar donde se lo haya colocado.

En caso que no se ejecute el programa directamente, se abre una ventana (Figura A.2) donde se encuentra los ***.vi*** que forman todo el programa.

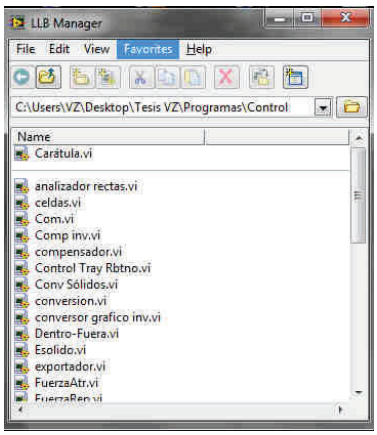


Figura A.2 Rutinas del programa de planificación de rutas

Luego se selecciona el ícono ***Carátula.vi*** y el programa se ejecuta.

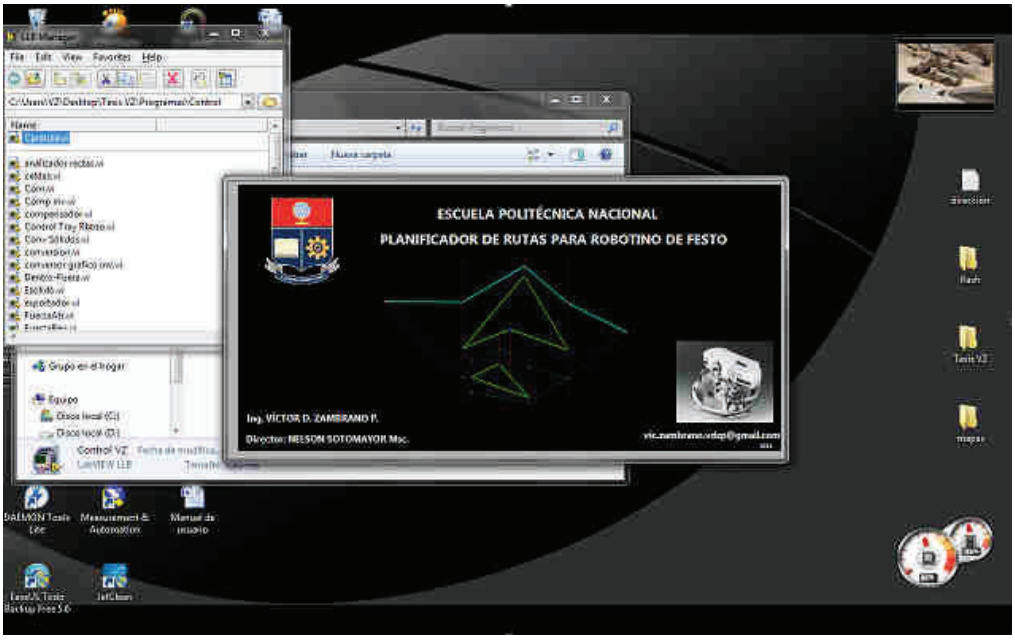


Figura A.3 Presentación del HMI

El programa comienza presentando una carátula (Figura A.3) que dura un tiempo aproximado de $5s$ para luego llevar al HMI de los planificadores de ruta (Figura A.4).

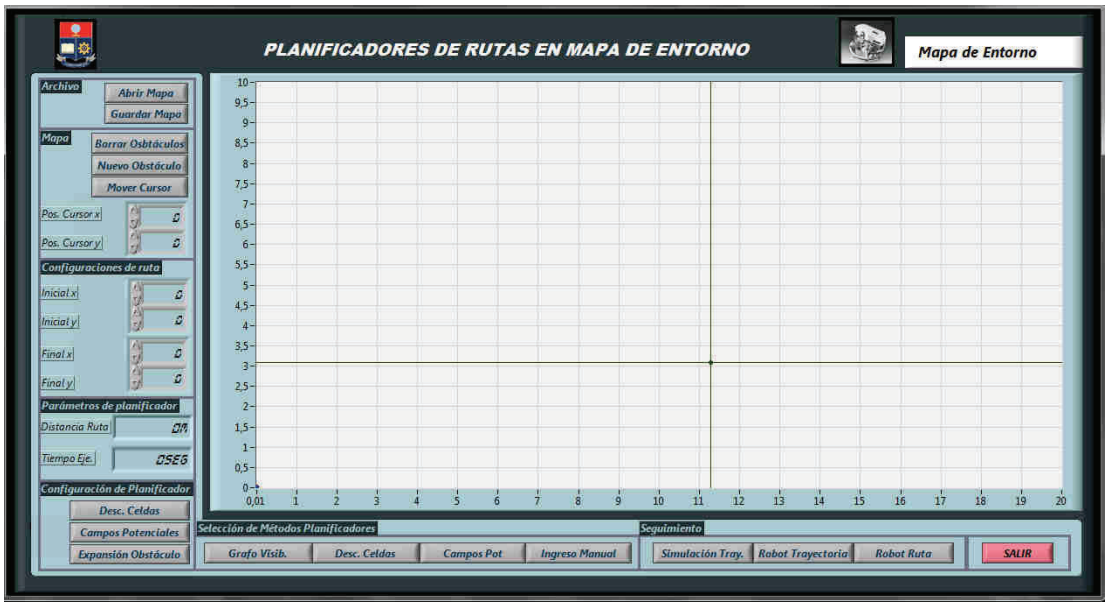


Figura A.4 HMI Planificadores de Rutas en Mapa de Entorno

A.2.1 INGRESO DEL MAPA DE ENTORNO

El ingreso del mapa de entorno se lo hace por medio de polígonos, el HMI tiene la capacidad de ingresar cualquier cantidad de polígonos de cualquier dimensión y número de vértices que estén dentro del rango del plano que es $20x10m$ escalado.

El cursor que se encuentra en el plano y es el instrumento que dibuja el mapa de entorno, como se aprecia en la Figura A.5.

Para dibujar un polígono, primero se selecciona el cursor, e inmediatamente el programa reconoce al punto como el primer vértice el polígono.

Para el segundo vértice se arrastra el cursor hacia cualquier punto del plano y se vuelve a seleccionar el cursor dibujando el primer lado del polígono.

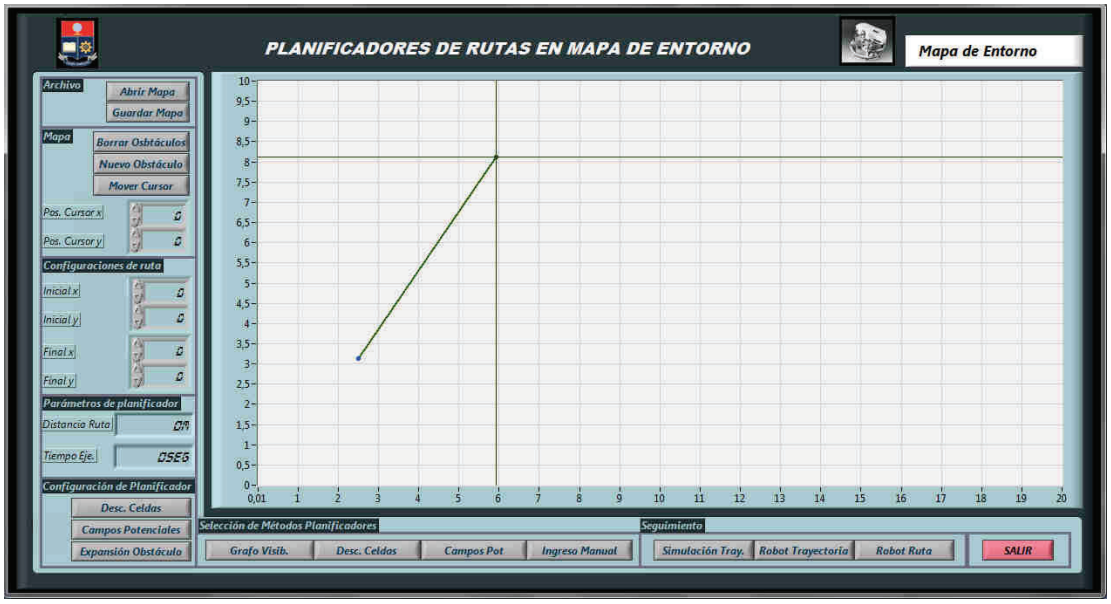


Figura A.5 Ingreso de polígono

Se repite el mismo proceso para dibujar el resto de lados del polígono.

Cuando termine de dibujar el polígono presione el botón ***Nuevo Obstáculo*** y el programa termina el polígono cerrándolo para luego comenzar con el siguiente (Figura A.6).

Figura A.6 Terminación de polígono

> **Nota:** Antes de terminar un polígono dejar el cursor en la posición del primer vértice del siguiente.

En el caso que se requieran figuras con vértices exactos y precisos, se puede posicionar al cursor donde se lo requiera, utilizando el botón ***Mover Cursor*** junto con los controles de posicionamiento (Figura A.7).

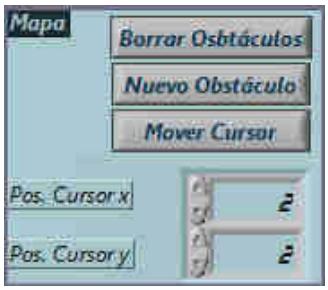


Figura A.7 Controles para mover cursor

Este control permite realizar figuras exactas y/o complejas de igual manera que ingresando los vértices como coordenadas (Figura A.8).

Figura A.8 Ingreso por coordenadas utilizando cursor

Una vez posicionado el cursor se hace clic de igual forma que el método de dibujo libre y programa dibuja el lado del polígono.

A.2.2 GUARDAR MAPA

Al seleccionar el botón ***Guardar Mapa***, el programa abre una ventana (Figura A.9) para seleccionar el nombre y el destino del mapa a guardarse.

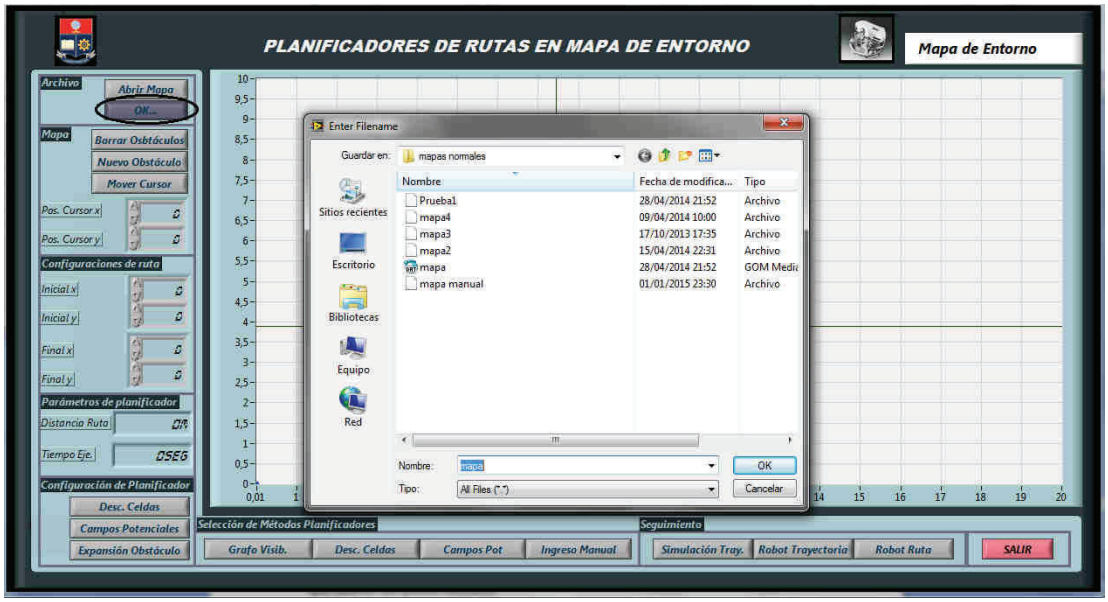


Figura A.9 Función guardar mapa

Por defecto el programa selecciona el escritorio como destino y el nombre ***mapa***, que puede ser personalizable.

Esta opción también tiene la capacidad de guardar el método planificador ejecutado. Se utiliza en caso que el método planificador tarde mucho tiempo en encontrar una ruta ó cuando se requiera utilizar la misma ruta varias veces.

> **Nota:** El tamaño del archivo generado varía en función del número de obstáculos o sí se incluye el método planificador, y está en el orden de las decenas de KB.

A.2.3 ABRIR MAPA

Como su nombre lo indica esta opción abre un mapa previamente guardado en el computador (Figura A.10).

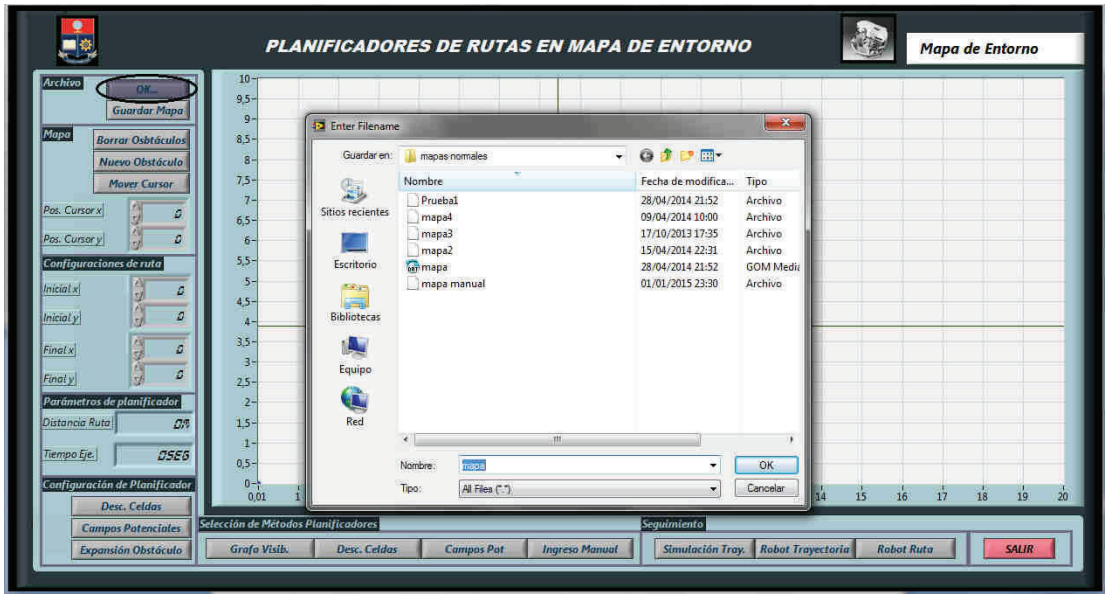


Figura A.10 Función abrir mapa

> **Nota:** Si el archivo incluye el método planificador el plano presenta toda esa información.

Si se abre un archivo con el método planificador terminado puede ser utilizado directamente para ejecutar los movimientos del Robotino®.

A.2.4 EDICIÓN DE MAPA PREVIAMENTE GUARDADO

En el caso que se quiera incluir más obstáculos a un mapa, simplemente hay que deslizar el cursor sobre la zona libre de obstáculos y se dibuja las formas descritas en **A.2.1**.

> **Nota:** El programa automáticamente borra el método planificador dejando únicamente el mapa de entorno (Figura A.11).

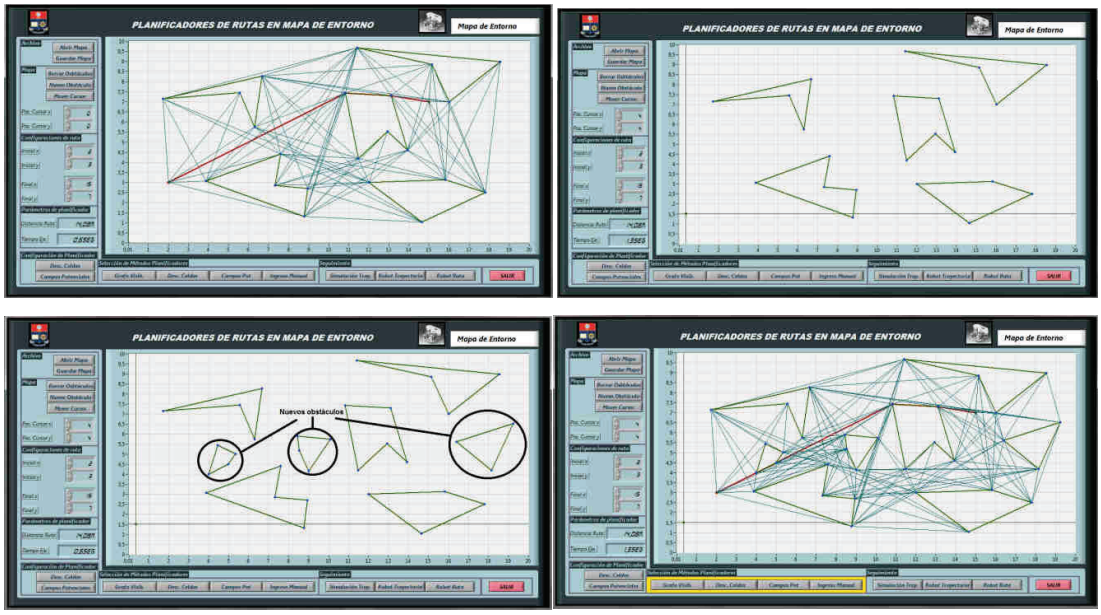


Figura A.11 Proceso de ingreso de obstáculos adicionales

A.2.5 INGRESO DE CONFIGURACIONES

El ingreso de las configuraciones se realiza con los controles del HMI (Figura A.12).

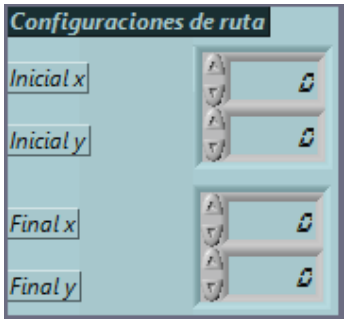


Figura A.12 Ingreso de configuraciones

Se puede utilizar los cursores de los controles, o se puede colocar directamente el número sobre el control y escribir los parámetros.

Nota: Estos valores deben estar dentro del rango del plano, y de la zona libre de obstáculos, ó el programa no ejecutará el método planificador.

A.2.6 EJECUCIÓN DE MÉTODOS PLANIFICADORES

El programa ejecuta el método planificador según la selección. Los métodos planificadores se pueden ejecutar hasta en el caso que no exista un mapa dibujado pero obligatoriamente debe tener las configuraciones para su correcto funcionamiento.

La ejecución del método planificador se produce en tiempo real presentado todo el proceso de construcción del grafo de conectividad en el caso del método Grafo de Visibilidad y el de Descomposición de Celdas, y la construcción de la ruta del método Campos Potenciales. La Figura A.13 muestra la ejecución del método Grafo de Visibilidad.

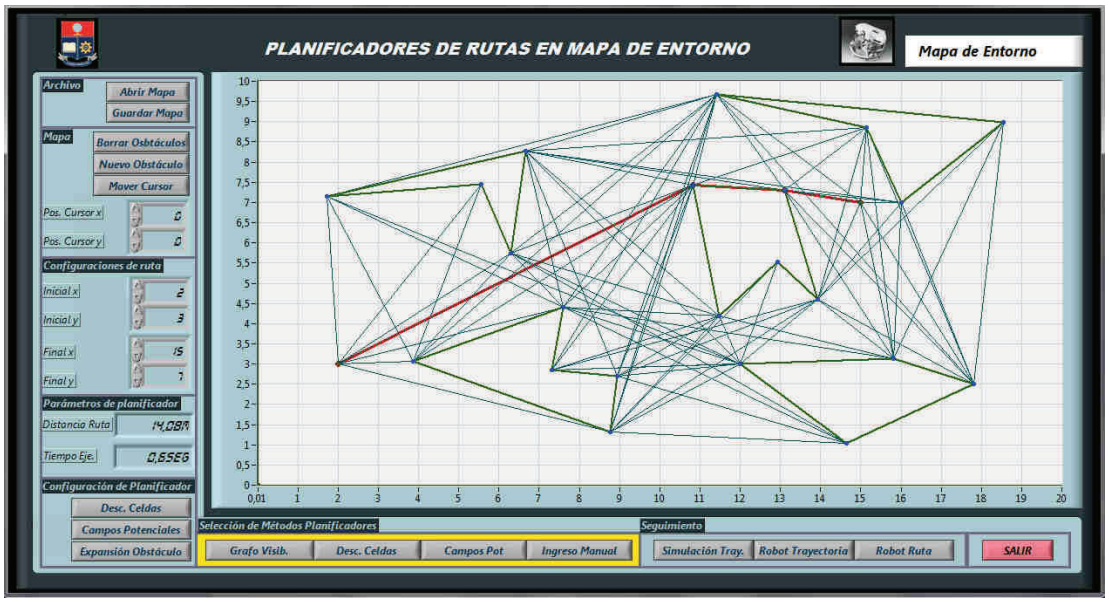


Figura A.13 Ejecución de método Grafo de Visibilidad

El seleccionar el método planificador el programa indica el método planificador utilizando código de colores (Figura A.14).

- Amarillo para el método Grafo de Visibilidad.

- Verde para el método Descomposición de Celdas.

- Rojo para el método de Campos Potenciales.

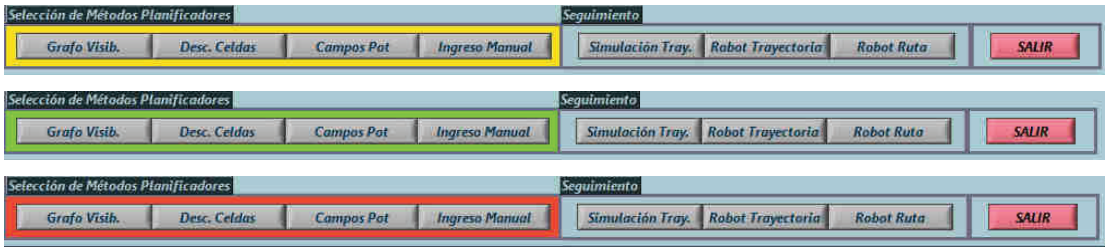


Figura A.14 Indicadores de ejecución de método

El indicador ***Tiempo Eje*** presentado en la Figura A.15***.*** Es un cronómetro que determina el tiempo procesamiento del método de planificador seleccionado y la progresión del mismo para determinar la ruta de un mapa de entorno dibujado.

Al terminar el cálculo de la ruta obtenida, el indicador **Distancia Tray.** Presenta la distancia total de la ruta entre las configuraciones escalada de acuerdo a la trayectoria que después será ejecutará el Robotino®, ó el simulador.

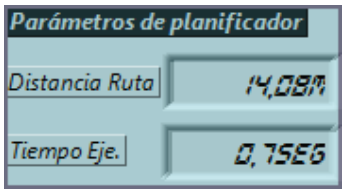


Figura A.15 Indicadores adicionales de ejecución de ruta

Nota: Una vez ejecutado un método planificador, el programa no procesa los cambios por el usuario del mapa de entorno hasta cuando obtenga la ruta determinada.

Nota: El tiempo de ejecución de método planificador varía en función de la complejidad del mapa dibujado y del computador utilizado.

A.2.7 INGRESO MANUAL DE RUTA

Otra opción que brinda el HMI es dibujar la ruta directamente seleccionando el botón ***Ingreso Manual.***

Para dibujar la ruta primero de pulsa este botón, y con el cursor se ubica los vértices del camino para evadir los obstáculos siguiendo la ruta a partir de la configuración inicial, como se presenta en la Figura A.16.

Figura A.16 Ingreso de Ruta manualmente

Para agregar más vértices simplemente se lleva al cursor a la nueva coordenada y se selecciona, el HMI dibujará la ruta hasta ese punto (Figura A.17).

Nota: El programa no detecta los obstáculos en esta opción.

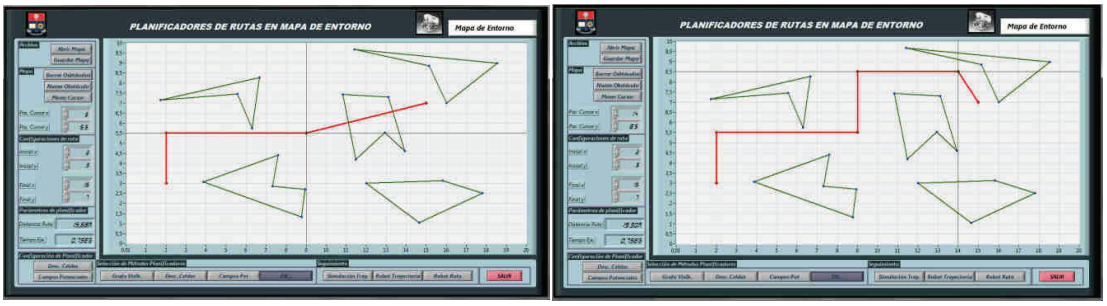


Figura A.17 Proceso de ingreso manual de ruta

En el caso que se requiera borrar la ruta, simplemente se oprime el botón ***Borrar Obstáculo***.

Adicionalmente el HMI presenta la distancia entre las configuraciones en el indicador ***Distancia Tray.***

A.3 PROGRAMA DE CONTROL DE RUTA PARA ROBOTINO®

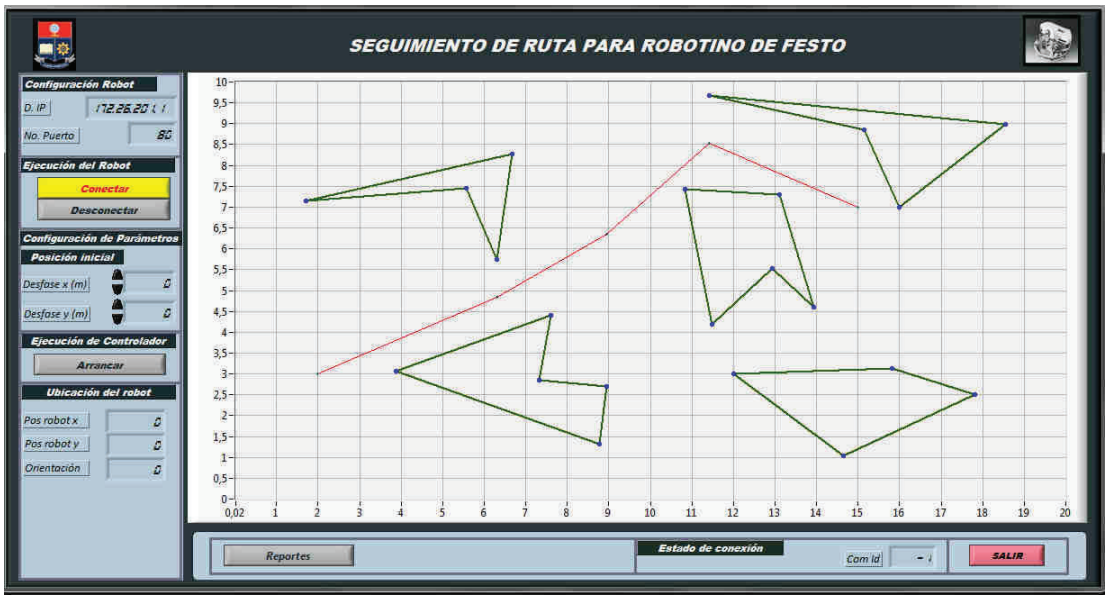


Figura A.18 HMI para control de ruta para Robotino®

La ventana de la Figura A.18 cuenta con todas las opciones del control de Robotino® para que ejecute los movimientos de la ruta obtenida o ingresada.

El HMI presenta el mapa de entorno y la ruta final, sin los demás parámetros de los métodos planificadores.

> **Nota:** Antes de Ejecutar el programa primero se debe tener establecida la conexión con el Robotino® (Figura A.19).

Figura A.19 Controles de funcionamiento

Primero se ingresa la dirección IP y el número de puerto del Robotino®.

Para establecer la conexión con el Robotino® presione el botón ***Conectar***, y para terminar la conexión presione Desconectar.

> **Nota:** Si el botón ***Conectar*** está intermitente el robot se encuentra desconectado.

Cuando se establezca la conexión el HMI informará cambiando de color del fondo del botón arrancar y colocando un valor diferente de ***-1*** en el indicador de estado de conexión (Figura A.20).

Figura A.20 Presentación de estado de conexión

Nota: El HMI solo se ejecutará cuando el Robotino® se haya conectado exitosamente.

A.3.1 EJECUCIÓN DE MOVIMIENTOS

Para iniciar el control al Robotino® se presiona el botón **Arrancar**

Figura A.21 Presentación de movimientos del Robotino®

Cuando el Robotino® ejecuta la ruta, el HMI dibuja en el plano la ruta que el robot sigue procesando la información proveniente de los encoders (Figura A.21).

> **Nota:** El HMI no puede detenerse mientras el Robotino® ejecute la trayectoria.

A.4 MANEJO DE REPORTES Y ALMACENAMIENTO DE INFORMACIÓN

Esta opción permite obtener y guardar toda la información del mapa de entorno, ruta obtenida y la ejecutada por el Robotino® pulsando el botón ***Reportes,*** (Figura A.22).

Figura A.22 Presentación de reportes

Al abrir la ventana se despliega la información de los obstáculos, enlaces del grafo formado, la ruta terminada y la ejecutada por el Robotino®.

El botón abrir Tabla presenta todos los valores de coordenadas (Figura A.23).

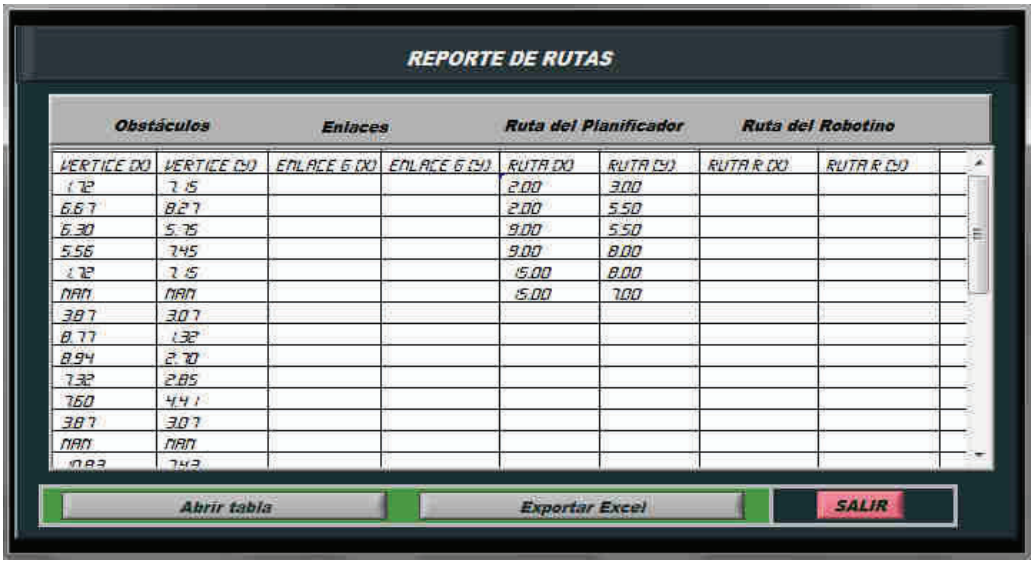


Figura A.23 Presentación de información de mapa de entorno

Adicionalmente se puede exportar a Excel para el uso posterior de la información, para ello se presiona el botón ***Exportar Excel.***

El HMI abre el Excel y presenta en el mismo formato la información de todos los parámetros que interviene en la construcción y reproducción de la ruta (Figura A.24).

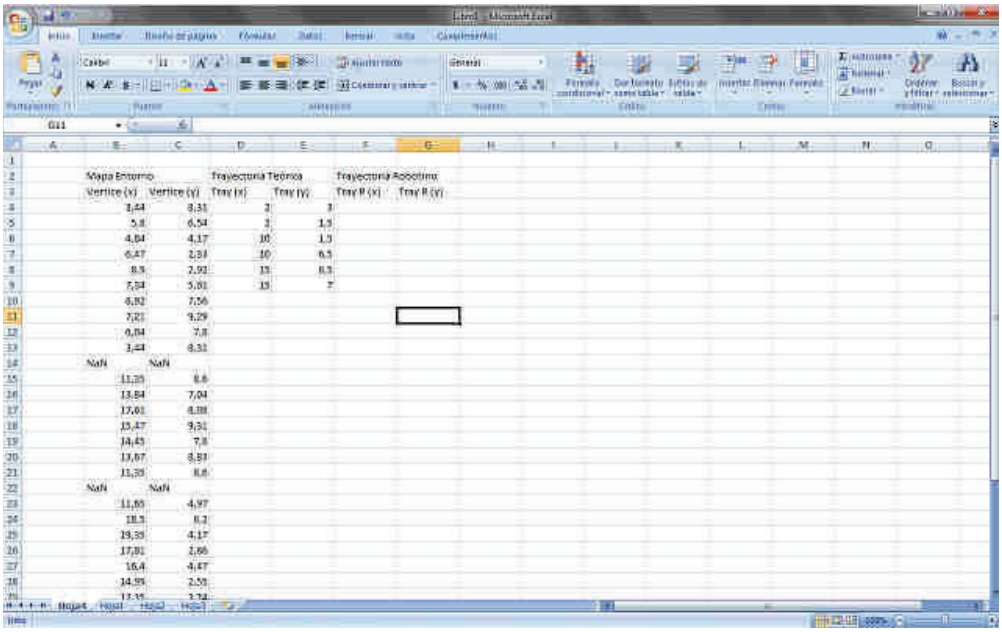


Figura A.24 Exportación de datos a Excel

A.5 PROGRAMA DE CONTROL DE TRAYECTORIA PARA ROBOTINO®

Ingresando a través del HMI principal (***Robot Trayectoria***), este programa a diferencia del control de ruta al Robotino® realiza el procesamiento de la ruta trazada por el método planificador convirtiéndola en una función matemática, a la cual se puede realizar el control de velocidad para cumplir con todo el recorrido en un tiempo especificado (Figura A.26)

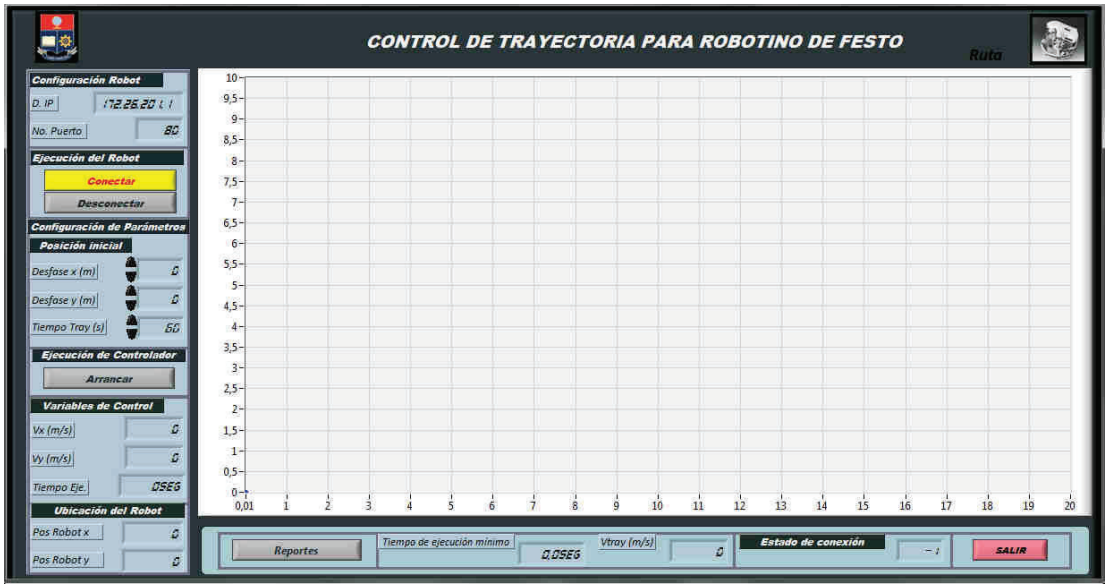


Figura A.26 HMI para control de Trayectoria

El HMI cuenta con las herramientas necesarias para realizar el control de trayectoria, para lo cual primero se debe configurar la dirección IP del Robotino®, y posteriormente conectarlo para que esté en línea con el programa de control de igual forma que el programa de control de ruta utilizando los controles presentados en Figura A.27.

Figura A.27 Configuración de Robotino®

Realizada la configuración del Robotino®, se debe configurar el tiempo de ejecución de trayectoria, por defecto este valor es de 60 segundos.

> **Nota:** El programa tiene la capacidad de calcular el tiempo mínimo para cumplir con la trayectoria satisfactoriamente por lo que sí se configura un tiempo menor el programa automáticamente reprograma la trayectoria en base a este tiempo mínimo.

Los controles ***Desfase***, son controles que permiten desfasar una distancia en metros de la configuración inicial, permitiendo que el controlador compense el error de posición generado cuando el robot se encuentre en una posición diferente a la inicial.

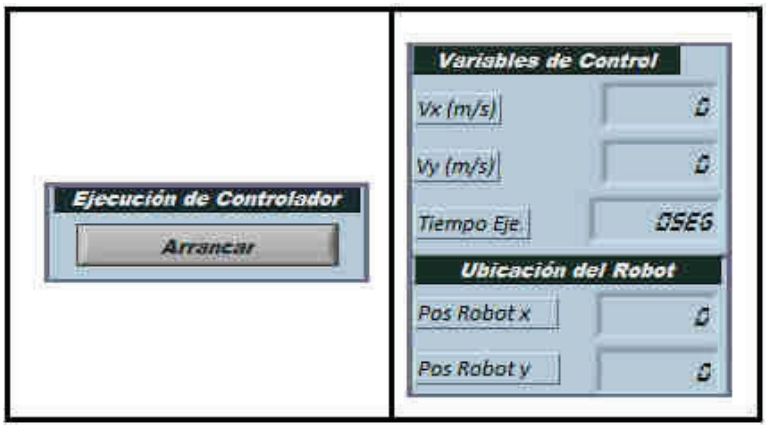


Figura A.28 Indicadores de velocidad y posición

Pulsando el botón ***Arrancar*** de la Figura A.28, el programa realiza el control del robot y presenta los resultados en el plano presentado en el HMI, realizando la comparación con la ruta teórica (Figura A.29). Adicionalmente se tiene un indicador que contabiliza el tiempo transcurrido de la ejecución de la trayectoria y también presenta las velocidades a las que se le envía al robot para ejecutar los movimientos, y la posición actual de robot.

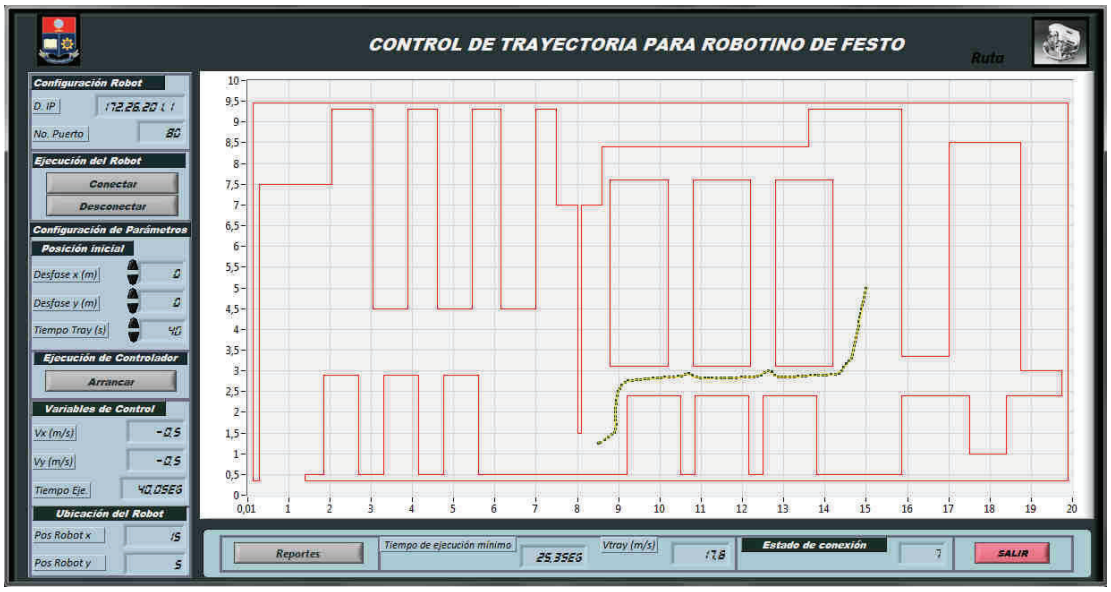


Figura A.29 Ejecución de movimientos del Robotino®

El programa también cuenta con sistema de reportes que presenta la información acerca de la trayectoria del método planificador, la trayectoria ejecutada por el Robotino®, velocidades aplicadas y errores entre la trayectoria generada y la obtenida por el robot, las cuales también pueden presentadas gráficamente y exportadas al Excel (Figura A.30).

Figura A.30 HMI para reportes de control de Trayectoria

Para ingresar al reporte gráfico primero se debe cargarlos datos obtenidos del control de trayectoria (***Abrir Tabla***), y luego con el botón ***reporte gráfico*** el HMI presenta los valores gráficamente (Figura A.31).

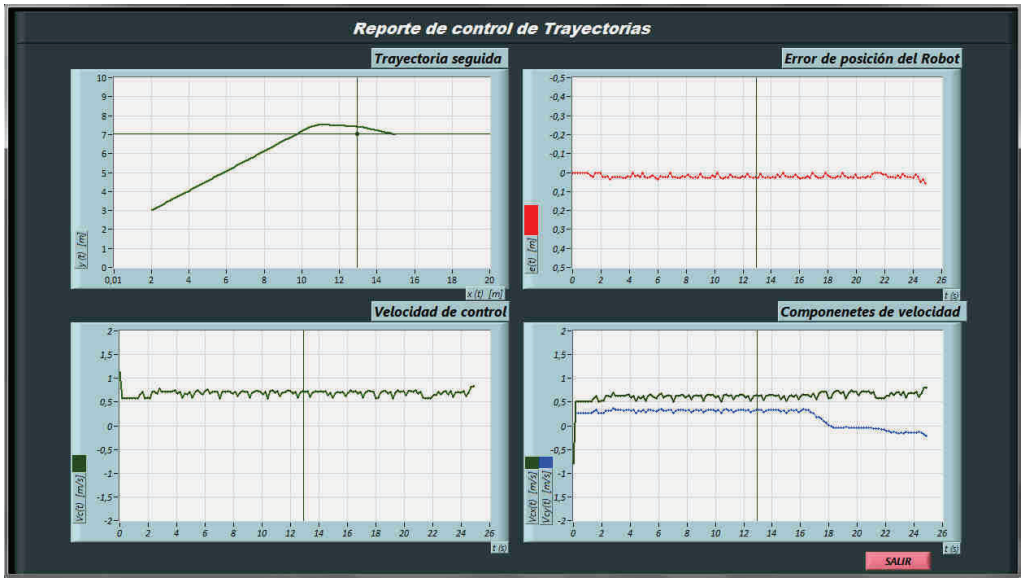


Figura A.31 Reporte gráfico

Los reportes gráficos (Figura A.31) presenta la trayectoria seguida por el robot, el error de posición frente a la trayectoria real en función del tiempo y las componentes de la velocidad de control al robot.

A.6 SIMULADOR DE CONTROL DE TRAYECTORIA

La ventana cuenta con las opciones necesarias para realizar la simulación del control de trayectoria en caso que no se cuente con un Robot (Figura A.32), para lo cual el programa (***ControlVZ.llb***), tiene una rutina que simula los movimientos que ejecuta el Robotino®, enviando las velocidades por parte del controlador.

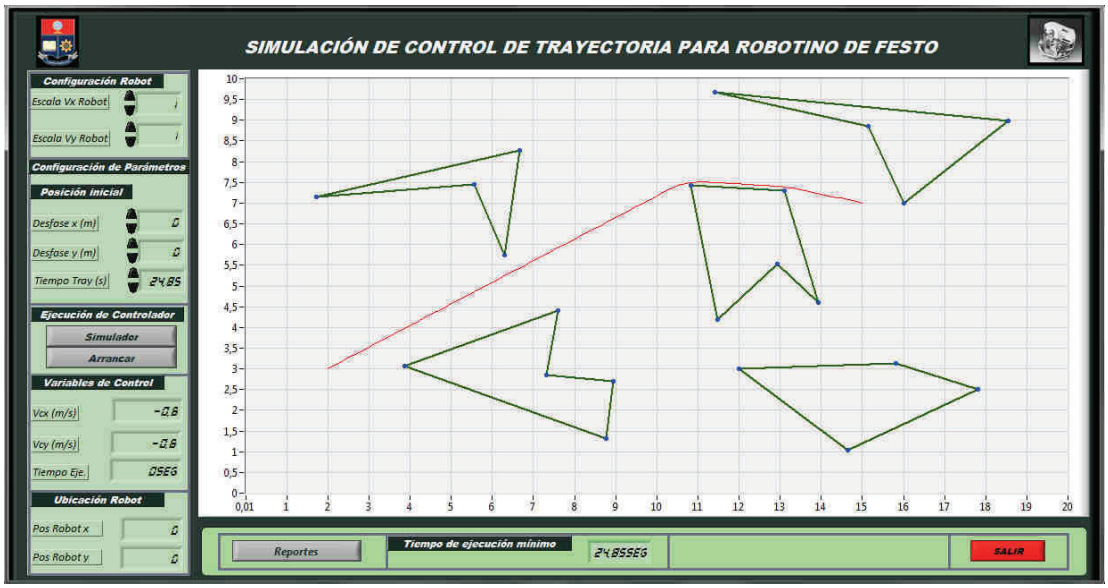


Figura A.32 HMI para simulación de control de Trayectoria

Para realizar la simulación, primero se debe arrancar el simulador para lo cual se presiona el botón ***Simulador***, e inmediatamente se abre una ventana que realiza las funciones del robot (Figura A.33).

> **Nota:** En el caso que no se arranque el simulador el control no funcionará correctamente.

Figura A.33 Simulador de Robotino®

Al realizar la simulación del control de trayectoria el HMI cuanta con las mismas funciones que el control con el robot real, con la presentación de reportes, y calibraciones de los controles.

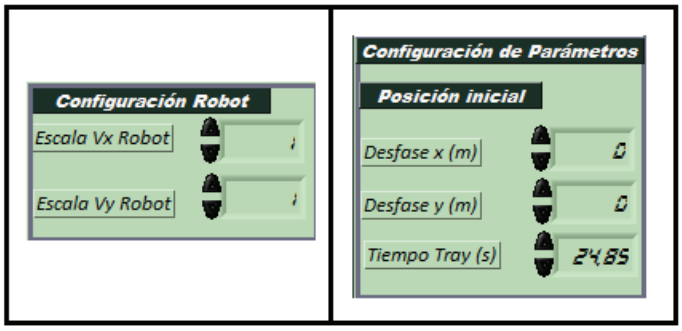


Figura A.34 Parámetros para ejecución de trayectoria

Los controles (***Escala Vx robot***), y (***Escala Vy robot***), son parámetros que sirven para escalar la velocidad del simulador, cuyo objetivo es simular posibles errores del parte del robot al cual se le da una velocidad pero no concuerda con la que verdaderamente ejecuta (Figura A.34). El valor por defecto es 1 que indica que la velocidad del control es la que el robot ejecuta.

Adicionalmente el programa presenta la información acerca de la posición instantánea del robot con el respectivo tiempo que ha transcurrido en los indicadores del HMI (Figura A.35), y la velocidad a la que se desplaza el Robotino.

Figura A.35 Información presentada acerca del control de trayectoria

A.7 CONFIGURACIONES AVANZADAS

A.7.1 MODIFICACIÓN DE CONSTANTES PARA CAMPOS POTENCIALES

Esta opción que presenta el HMI, permite cambiar las constantes de la fuerza de Repulsión neta y la de Atracción del método:

$$F_{atr} = k_a(1 - e^{-6.7D_{PiPf}}) \tag{A.1}$$

$$F_{Rep} = -k_r e^{-6.7D_{PiPj}} \tag{A.2}$$

Los valores de las constantes por defecto están configurados de acuerdo a las dimensiones del plano.

Para abrir esta ventana se pulsa el botón ***Campos Potenciales*** del HMI para calcular los métodos planificadores (Figura A.36)

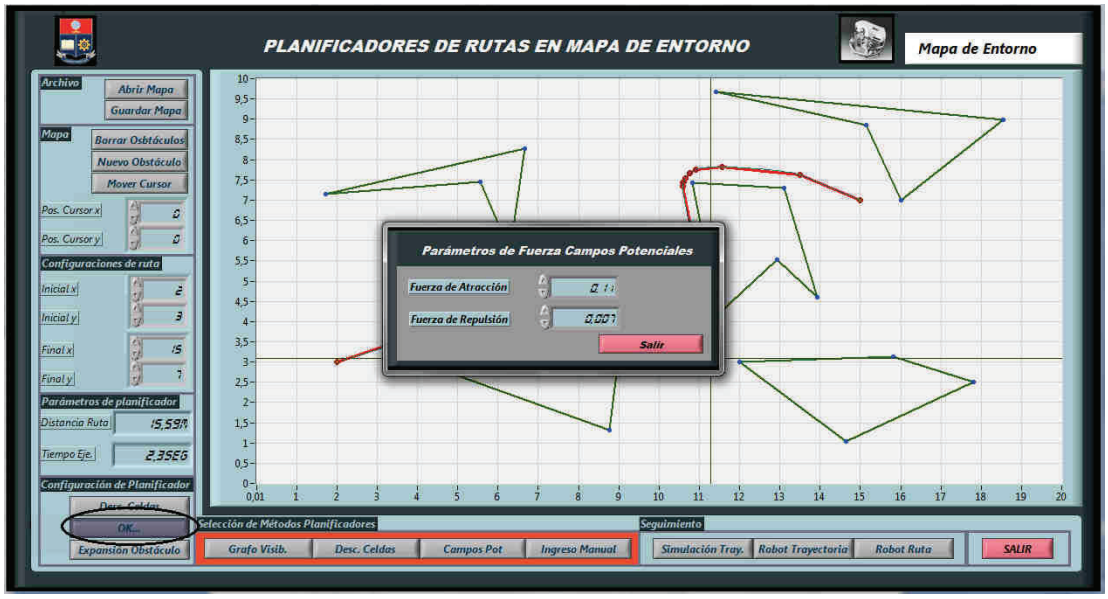


Figura A.36 Calibración de Constantes para Campos Potenciales

Cuando se cambia los valores de las constantes todas las rutas ejecutadas posteriores se resolverán con las nuevas constantes.

> **Nota:** Cuando se cierra el Programa los valores de las constantes retornan a los valores por defecto.

> **Nota:** Los valores de las constantes deben ser cercanos a los valores por defecto para obtener una respuesta adecuada.

A.7.2 MODIFICACIÓN EL MÉTODO DESCOMPOSICIÓN DE CELDAS

Esta opción permite modificar el método descomposición de celdas en la forma de generación de enlaces entre nodos, agregando enlaces adicionales por línea de vista cuya función es aumentar el grafo de conectividad y disminuir los quiebres entre las configuraciones (Figura A.37).

Figura A.37 Ingreso de número de enlaces adicionales

Para agregar enlaces se utiliza el control ***número de enlaces***, estos se incluyen en el grafo de conectividad y con el botón se selecciona el algoritmo tradicional y el modificado.

Nota: El control máximo puede realizar 5 enlaces adicionales por nodo.

Al realizar el accionamiento del método modificado el tiempo de ejecución aumenta en función de los enlaces adicionales realizados la Figura 3.38 presenta la diferencia entre el método tradicional y el modificado.

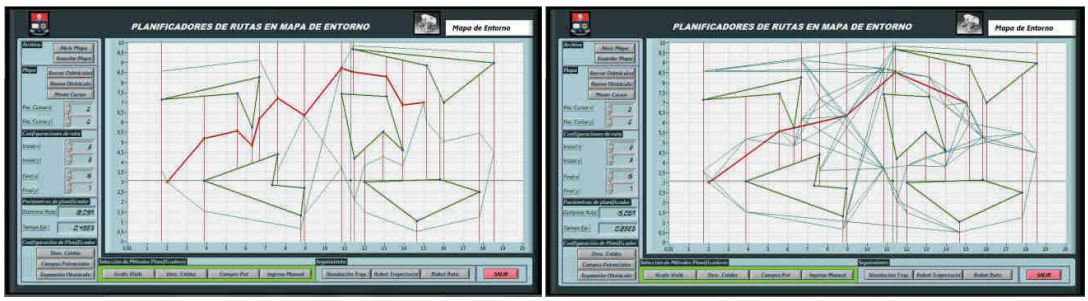


Figura A.38 Algoritmo tradicional vs modificado con 2 enlaces adicionales

A.7.3 EXPANSIÓN DE OBSTÁCULOS

La expansión de obstáculos es una función exclusiva al método Grafo de Visibilidad, el cual expande todos los obstáculos del mapa de entorno en función de la distancia perpendicular de los lados de cada obstáculo, para ingresar a esta opción de oprime el botón ***Expansión Obstáculo*** y el HMI abre una ventana para la expansión (Figura A.39).

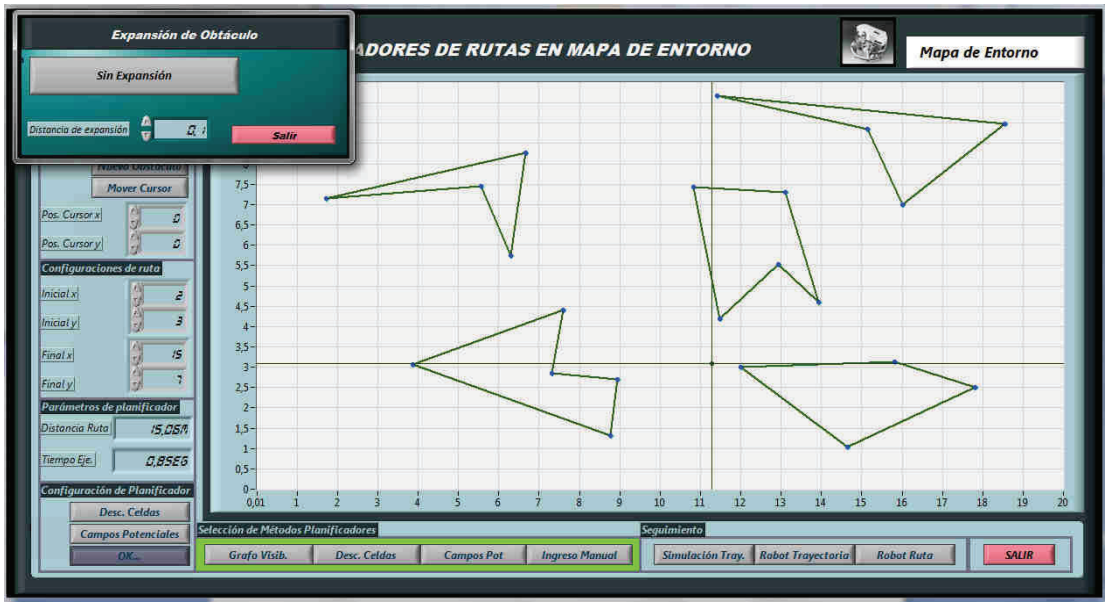


Figura A.39 Expansión de obstáculos

La distancia de expansión de obstáculo se gradúa con el control ***distancia de expansión***, y el HMI realiza la expansión presentando los cambios realizados al momento de la graduación y presentando en el HMI (Figura A.40).

> **Nota:** Cuando se ejecuta el método planificador el mapa expandido no se presenta en el HMI.

> **Nota:** Para la ejecución del control de ruta y trayectoria, el HMI presenta el entorno no expandido sino el original.

Figura A.40 Presentación de obstáculos expandidos

ANEXO B

ALGORITMOS DE BÚSQUEDA DE RUTAS

B.1 ALGORITMO DIJKSTRA [18]

Desarrollado por Edsger Dijkstra en 1959, este método consiste en determinar dentro de un grafo de conectividad el camino de menor costo desde un nodo seleccionado hasta todos los demás sin importar la cantidad de nodos intermedios para llegar a su destino.

El costo es una ponderación para trasladarse desde un nodo hacia otro formando un enlace. El criterio más común para determinar el costo en la construcción de rutas el campo de la robótica es la distancia euclídea entre los nodos involucrados.

Al ser un método de búsqueda iterativo, este se basa en una secuencia de pasos a seguir para completarlo.

B.1.1 PROCEDIMIENTO DE CÁLCULO

A continuación se ilustra la ejecución del método con el siguiente ejemplo (Figura B.1):

Se tiene un grafo dirigido que consta de siete nodos representados con letras mayúsculas. Primero se selecciona el nodo de origen de ruta, para el caso de ejemplo se toma el nodo **A.**

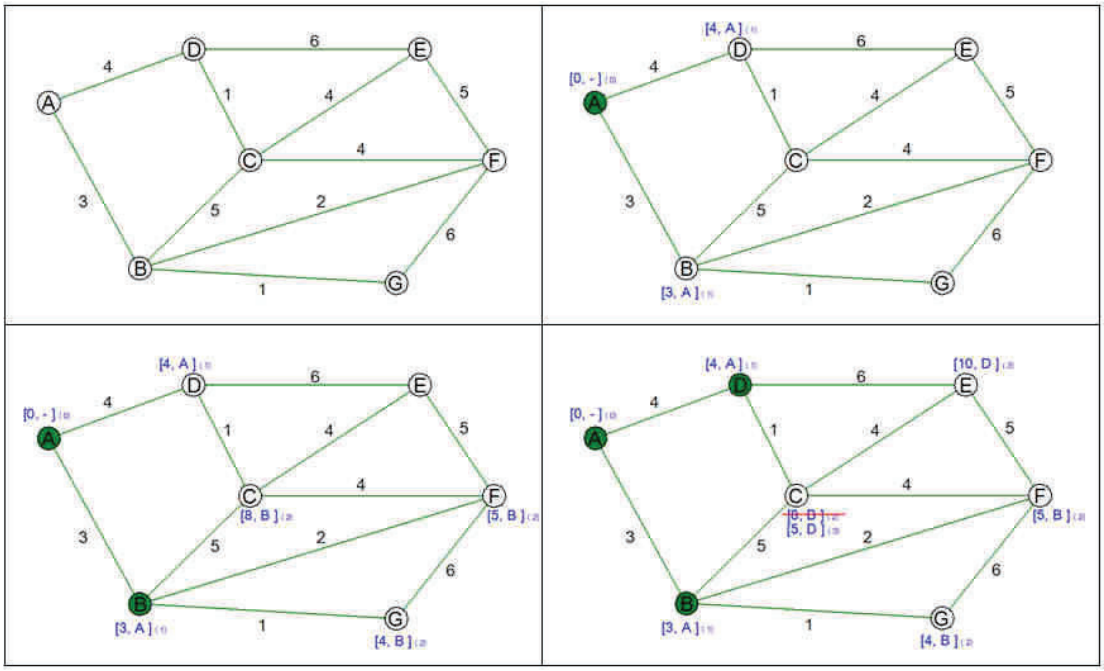


Figura B.1 Procedimiento del algoritmo (paso 1 y 2)

Con el primer nodo seleccionado se analiza todos los nodos adyacentes a él y se toman todos los valores de los costos guardándolos en una matriz, para el caso del ejemplo se escribe la matriz en el nodo del grafo con los valores del costo, la procedencia y la iteración actual.

El la Figura B.1, la ruta para llegar a los nodos **B** y **D** desde **A** son:

$$\mathbf{B}\text{: } [3, \mathbf{A}]_{(1)}$$

$$\mathbf{D}\text{: } [4, \mathbf{A}]_{(1)}$$

Al terminar el análisis se selecciona al nodo **A** como nodo permanente ó nodo ya tomado en cuenta.

El siguiente paso es seleccionar otro nodo, para lo cual se toma el nodo con el menor costo para ser alcanzado por **A**, en el caso del ejemplo sería el nodo **B** ya que tiene el costo de 3.

El nodo **B** tiene tres nodos adyacentes (**C**, **F** y **G**), analizado se tiene los siguientes valores para guardar en la matriz del algoritmo:

$$\mathbf{C}: [8, \mathbf{B}]_{(2)}$$
$$\mathbf{F}: [5, \mathbf{B}]_{(2)}$$
$$\mathbf{G}: [4, \mathbf{B}]_{(2)}$$

En el caso de **C**, la ruta más corta en este instante es 8 para alcanzar el nodo **C** por parte de **A**, y tiene que pasar por el nodo **B**. lo mismo ocurre para los nodos **F** y **G**. y al final se marca al nodo **B** como permanente.

El siguiente paso es tomar el nodo con el menor costo acumulado. Dado que existe dos rutas con el mismo costo mínimo se escoge aleatoriamente el nodo. Para el caso del ejemplo se escoge el nodo **D** ya que tiene el costo de 4.

Se analiza los nodos adyacentes a él menos el nodo **A** ya que fue considerado y está marcado como permanente, analizando se tiene los siguientes valores para guardar en la matriz:

$$\mathbf{C}: [5, \mathbf{D}]_{(3)}$$
$$\mathbf{E}: [10, \mathbf{D}]_{(3)}$$

De acuerdo con la Figura B.1 para alcanzar al nodo **C** se tiene dos rutas, entones se escoge la que tiene menor costo, entonces la nueva ruta es:

$$\mathbf{C}: [5, \mathbf{D}]_{(3)}$$

Descartando la anterior. Para finalizar la iteración se marca al nodo **D** como permanente.

El siguiente paso es tomar el nodo **G** ya que tiene el menor costo acumulado para ser alcanzado por **A**.

Se analizan los nodos adyacentes a él menos los nodos permanentes, entonces sería el nodo **F.** Siendo sus valores los siguientes:

$$\mathbf{F}\colon [10, \mathbf{G}]_{(4)}$$

Al tener dos rutas posibles al ser alcanzado por **A**, se toma la ruta con menor costo, entonces se desecha la ruta que viene por **G** manteniendo la ruta que viene por **C**. al finalizar se marca el nodo **G** como permanente (Figura B.2).

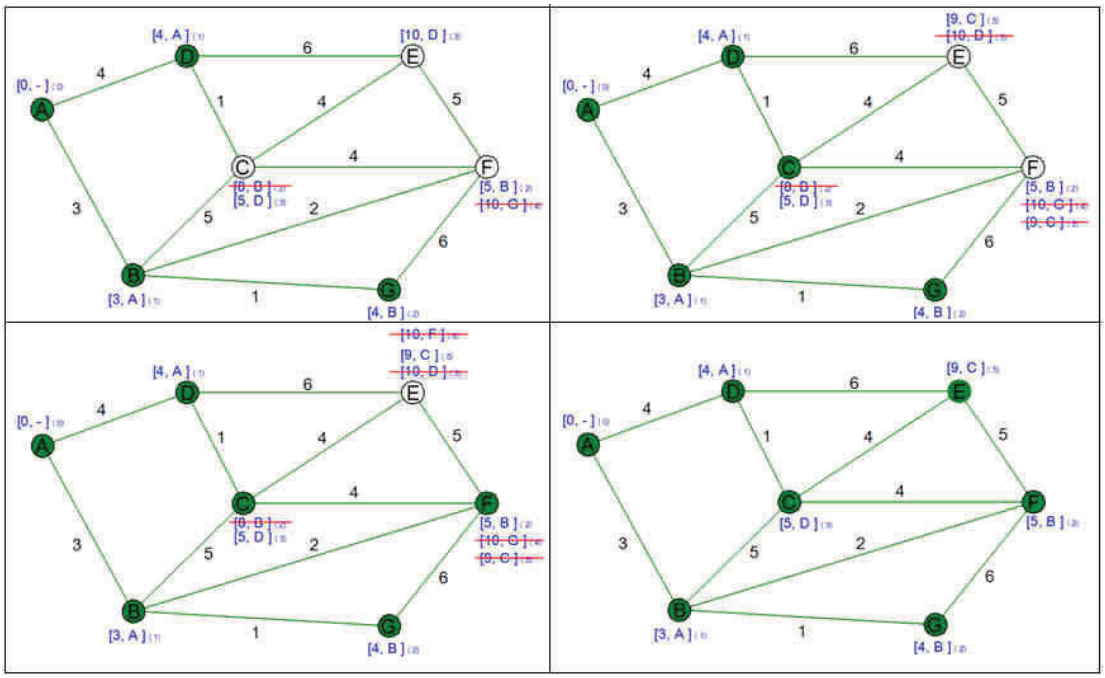


Figura B.2 Procedimiento del algoritmo (paso 3 y 6)

Para el siguiente paso se toma el nodo **C**, ya que tiene el menor costo con un valor de 5.

Se analiza los nodos adyacentes a **C**, menos los marcados como permanentes, entonces sería el nodo **E** y **F**, y los valoras acumulados de costos para ser alcanzados por **A** por esta ruta son:

$$\mathbf{E}\text{: } [9, \mathbf{C}]_{(5)}$$

$$\mathbf{F}\text{: } [9, \mathbf{C}]_{(5)}$$

Estas rutas se comparan con las anteriormente obtenidas, y se toma en cuenta la de menor costo, para el caso de **E** se cambia la ruta con la que viene de **D**, y para alcanzar el nodo **F** se mantiene la ruta que viene por **B**. Para finalizar se marca el nodo **C** como permanente.

El siguiente paso es tomar el nodo **F**, ya que tiene el menor costo acumulado para ser alcanzado por **A**.

Se analiza los nodos adyacentes a **F**, en este caso solo sería **E**, y se determina los valores en la matriz con los valores acumulados de costo ruta e iteración:

$$\mathbf{E}\text{: } [10, \mathbf{F}]_{(6)}$$

Ya que se tiene dos rutas para alcanzar a **E**, se escoge la que tiene el menor costo, entonces se conserva la ruta que viene por **C**. para finalizar se marca el nodo como permanente.

El siguiente paso es tomar el último nodo pero como ya no tiene nodos adyacentes el algoritmo termina.

Una vez terminado el algoritmo la matriz formada tiene toda la información para determinar la ruta desde el nodo **A** hasta cualquier nodo del grafo presentando la ruta que sigue y el costo para alcanzarlo.

De la Figura B.3 presenta la ruta más corta para llegar al nodo **F**, y al nodo **G**

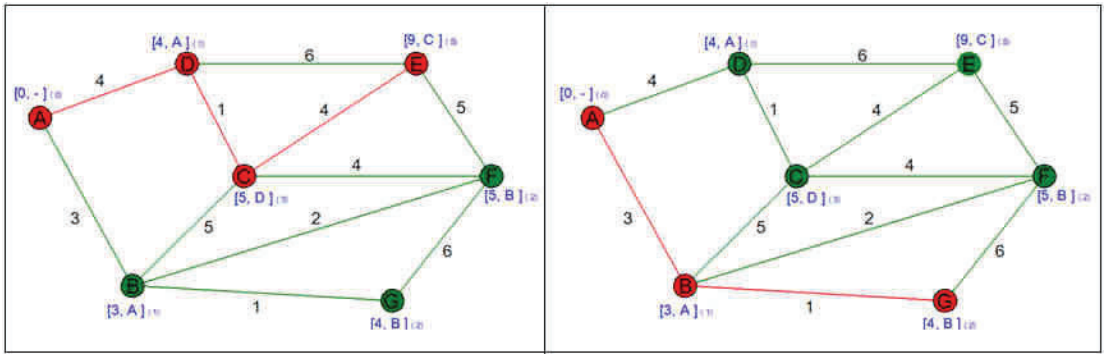


Figura B.3 Ruta para llegar a nodo destino

Para el caso del nodo **E**, especificada por matrices la ruta para llegar a **E** es:

$$\mathbf{E}\text{: } [9, \mathbf{C}]_{(5)}$$

Entonces para alcanzar a **E**, la ruta debe provenir por **C**, y la ruta para llegar a **C** es:

$$\mathbf{C}\text{: } [5, \mathbf{D}]_{(3)}$$

Para alcanzar a **C**, la ruta debe provenir por **D**, y la ruta para llegar a **D** es:

$$\mathbf{D}\text{: } [4, \mathbf{A}]_{(1)}$$

Y **D** es un nodo adyacente **A**, entonces para llegar a **E** la ruta completa es:

A-D-C-E

Y el costo es 9.

Generalizando el algoritmo se describe a continuación.

B.1.2 PASOS DEL ALGORITMO

Teniendo un grafo dirigido de **N** nodos sea:

- **X**: Nodo de Origen.

- **A**: Nodo actual tomado en cuenta para el análisis del algoritmo.

- **D**: Un vector o matriz de tamaño **N** que contiene los costos entre **X** al resto de nodos, y la ruta hasta el nodo **X**, entonces:

  a. Se inicializa los valores de los costos con un valor infinito excepto el costo de **X** el cual se debe colocar en 0.

  $$\boldsymbol{X} = [0, -]_{(0)}$$

  b. Sea **A=X** (Se toma como nodo actual).

  c. Se evalúa todos los nodos adyacentes de **A**, excepto los nodos permanentes, a estos se los llama nodos temporales.

  d. Si el costo entre **X** hasta $\mathbf{v_i}$ (nodos temporales) guardada en **D** es mayor que la distancia desde **X** hasta **A**, sumada desde **A** hasta $\mathbf{v_i}$; ésta es sustituida. Expresada matemáticamente es:

  Si $\mathbf{D_x} > \mathbf{D_a} + d(a, v_i)$ entonces $\mathbf{D_x} = \mathbf{D_a} + d(a, v_i)$

  e. Se marca al nodo actual como nodo permanente.

Se toma el próximo nodo que tenga el menor costo acumulado para la siguiente evaluación, y se repite

B.2 ALGORITMO A* [19]

Desarrollado en 1968 y posteriormente optimizado Peter Hart, NilsNilsson y Bertram Raphael. El algoritmo A* es un replanteamiento del algoritmo Dijkstra que disminuye el tiempo de procesamiento mediante el uso de la heurística.

El algoritmo A* utiliza una función compuesta entre el costo acumulado de la ruta desde la configuración inicial P_0, más una función heurística para determinar el próximo nodo que se incluya en la trayectoria.

$$f(x) = g(x) + h'(x) \tag{B.1}$$

$g(x)$: Es una función de la ruta que representa el costo o la distancia acumulada desde la configuración inicial P_0, hasta la configuración actual evaluada P_i.

$h'(x)$: Es una función de estimación heurística admisible desde la configuración actual evaluada P_i hasta la configuración de destino P_f.

La función heurística es un valor estimado desde la configuración actual evaluada P_i hasta la configuración de destino P_f. Para aplicaciones prácticas la función heurística ($h'(x)$) se puede ser representada como la distancia euclídea, entre la configuración actual y la configuración final, ya que es la distancia más corta entre dos nodos.

Sí la heurística satisface esta condición adicional:

$$h(x) \leq d(x, y) + h(y) \tag{B.2}$$

Para cada nodo del Grafo donde d es la longitud hacia el nodo actual evaluado. En tal caso el algoritmo A* se puede implementar de manera más eficiente, y es equivalente a ejecutar el algoritmo Dijkstra con el costo reducido.

B.2.1 PROPIEDADES [19]

- El algoritmo A* completo encontrará una ruta para todos los casos siempre y cuando estas existan.

- Sí la función heurística $h´(x)$, es admisible (sin sobre estimaciones de costo a la configuración final), entonces el algoritmo A* también lo es, en caso que se utilice un conjunto cerrado, en caso que se use un conjunto cerrado, $h´(x)$ debe ser monotónica para que $h´(x)$ sea óptimo.

- En caso que $g(x) = 0$, se trata de una búsqueda voraz

- En caso que $h´(x) = 0$, el algoritmo A* realiza una búsqueda de costo uniforme no informada (método Dijkstra).

- En caso que las dos componentes de la función sean 0, la búsqueda es aleatoria.

B.2.2 PROCEDIMIENTO [19]

El algoritmo A*, busca las ruta que parecen ser más cortas para llegar a la configuración final. Además de tener un motor de búsqueda definido, el algoritmo almacena la información de la distancia ya recorrida hasta un nodo evaluado.

Al comenzar el algoritmo, la cola de prioridades de los nodos se la conoce como un conjunto abierto. Para cada iteración el algoritmo, toma el nodo con el valor más bajo de la función heurística $f(x)$, y elimina de la cola de prioridades, los valores $f(x)$ y de la distancia real acumulada $g(x)$, de los nodos vecinos se actualizan y estos se añaden a la cola de prioridades (similar al algoritmo Dijkstra). El algoritmo continúa hasta encontrar el nodo del menor valor de $f(x)$, ó cuando la cola de prioridades esté vacía.

Una vez terminado el la ejecución del algoritmo, se obtiene la longitud de la ruta más corta y la secuencia de los nodos que determinan la ruta.

ANEXO C

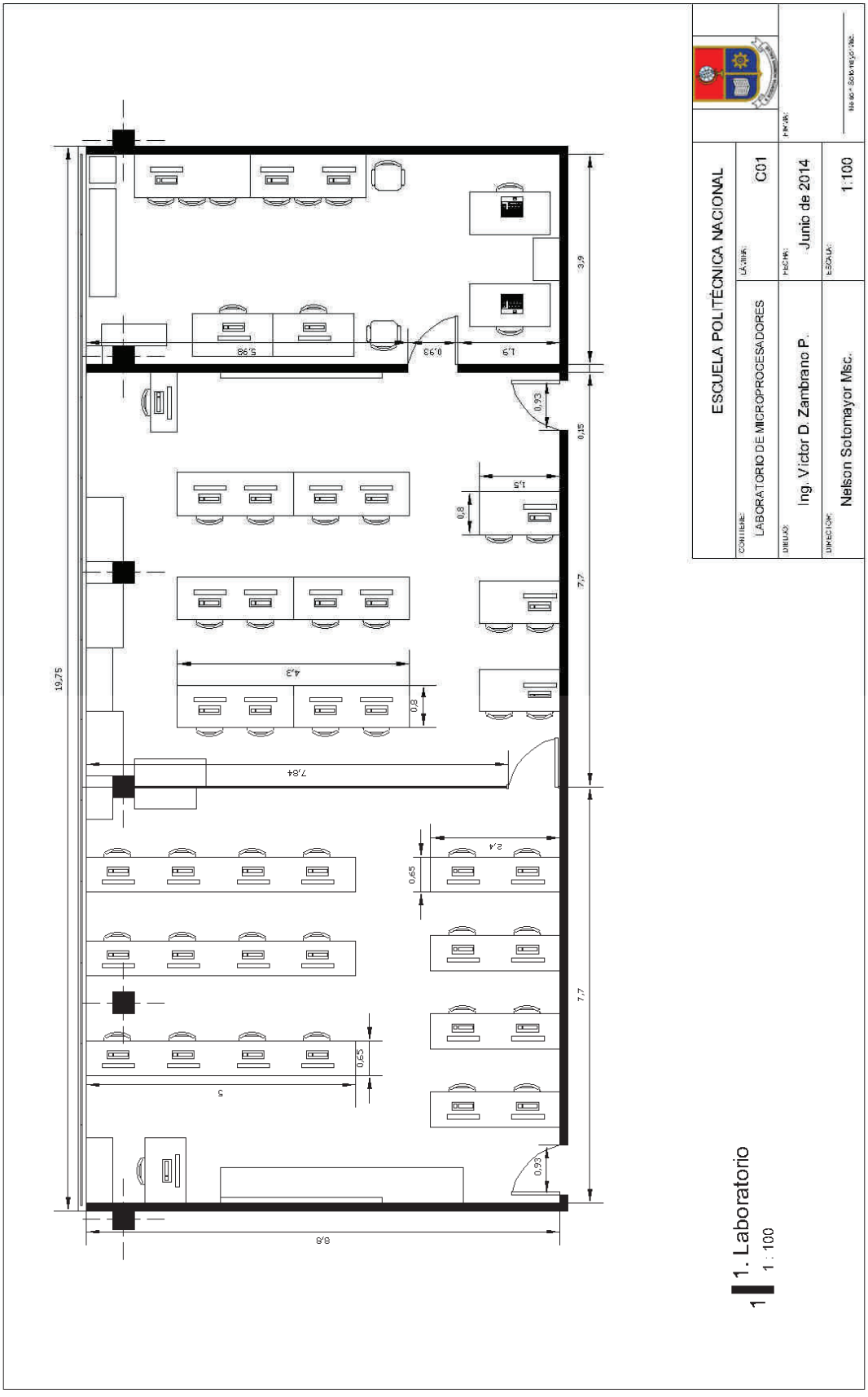
ENTORNO DE LABORATORIO

C.1 ENTORNO DE LABORATORIO

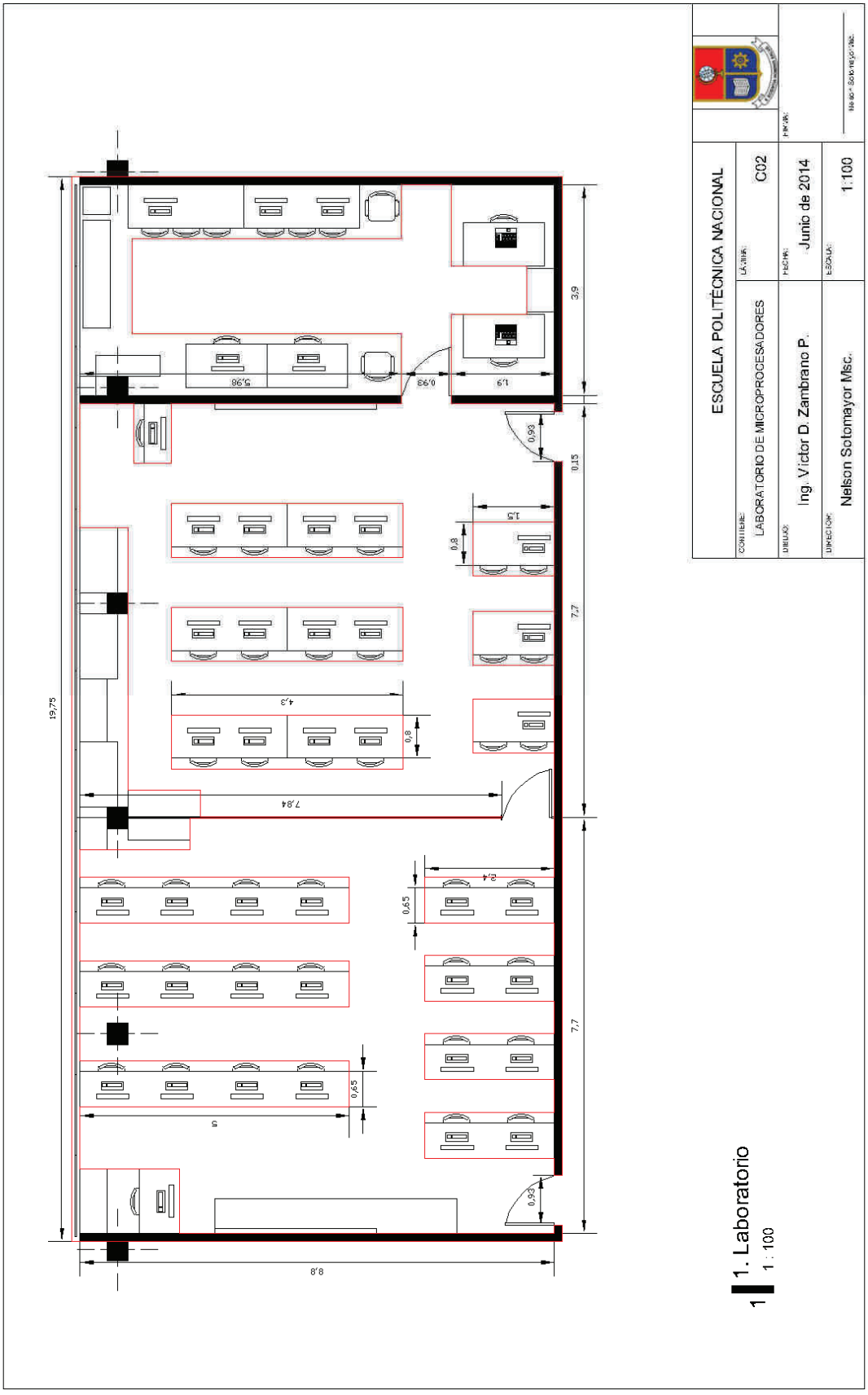
Para aplicaciones prácticas de los métodos planificadores y el control de trayectoria el Robotino® de Festo, se considera un mapa de entorno que venga desde un espacio físico real para ver el comportamiento de los métodos planificadores desarrollados y llevar a un robot que ejecute estos movimientos. Se toma el entorno del Laboratorio de Control y Sistemas donde se realizan este tipo de pruebas, este cuenta con un área que encaja perfectamente en el área de trabajo del HMI.

Dibujar este espacio en el HMI implica que se debe tener un plano con las dimensiones reales del laboratorio. Como no se cuenta con un plano, se tomaron las medidas del mismo y con todos los objetos que cuenta. Con la ayuda de AutoCAD se construyó el plano del laboratorio con las medidas tomadas directamente.

La Lamina C01, tiene todos los muebles y otros objetos. Para poder ingresarlos al HMI se dibuja sobre estos unos rectángulos que representan los obstáculos al HMI, la Lámina C02 tiene el los rectángulos que equivalen a los objetos del laboratorio.



1. Laboratorio
1 : 100


ESCUELA POLITÉCNICA NACIONAL
CONTIENE: LABORATORIO DE MICROPROCESADORES
LÁMINA: C02
DIBUJO: Ing. Victor D. Zambrano P.
FECHA: Junio de 2014
DIRECTOR: Nelson Sotomayor Msc.
ESCALA: 1:100
1. Laboratorio
1 : 100
19,75
8,8
7,7
7,7
3,9
0,15
0,99
0,93
1,9
5,98
1,5
0,8
4,3
0,8
7,84
2,4
0,65
5
0,65
0,93

C.2 INGRESO DE MAPA DE ENTORNO REAL

Con el laboratorio llevado a un plano a escala (Lámina C01), primero se determina las coordenadas de los vértices de los obstáculos representados por los objetos del laboratorio. Con la ayuda del AutoCAD se calculan los vértices de los obstáculos y de las paredes.

Para ingresar el entorno del laboratorio dentro del HMI primero al mapa se divide en diecinueve polígonos para completarlo los cuales se representan en la siguiente tabla.

Tabla C.1 Coordenadas de laboratorio ingresado en el HMI

Polígono 1	**vértice (x)**	1.4	1.4	1.85	1.85	2.7	2.7		
	vértice (y)	0.35	0.5	0.5	2.9	2.9	0.35		
Polígono 2	**vértice (x)**	2.75	2.75	3.3	3.3	4.15	4.15		
	vértice (y)	0.35	0.5	0.5	2.9	2.9	0.35		
Polígono 3	**vértice (x)**	4.2	4.2	4.75	4.75	5.6	5.6		
	vértice (y)	0.35	0.5	0.5	2.9	2.9	0.35		
Polígono 4	**vértice (x)**	5.65	6.65	9.35	9.35	10.35	10.35		
	vértice (y)	0.35	0.5	0.5	2	2	0.35		
Polígono 5	**vértice (x)**	10.4	10.4	11	11	12	12		
	vértice (y)	0.35	0.5	0.5	2	2	0.35		
Polígono 6	**vértice (x)**	12.05	12.05	12.65	12.65	13.65	13.65		
	vértice (y)	0.35	0.5	0.5	2	2	0.35		
Polígono 7	**vértice (x)**	13.7	13.7	15.85	15.85	17.5	17.5		
	vértice (y)	0.35	0.5	0.5	2.4	2.4	0.35		
Polígono 8	**vértice (x)**	17.55	17.55	18.4	18.4	19.9	19.9		
	vértice (y)	0.35	1	1	2.4	2.4	0.35		
Polígono 9	**vértice (x)**	0.15	0.15	2.05	2.05	0.3	0.3		
	vértice (y)	0.35	9.45	9.45	7.5	7.5	0.35		
Polígono 10	**vértice (x)**	3.9	2.1	2.1	3.05	3.05	3.9		
	vértice (y)	9.45	9.45	9.3	9.3	4.5	4.5		
Polígono 11	**vértice (x)**	3.95	3.95	4.6	4.6	5.45	5.45		
	vértice (y)	9.45	9.3	9.3	4.5	9.45	9.45		

Polígono 12	**vértice (x)**	5.5	5.5	6.15	6.15	7	7		
	vértice (y)	9.45	9.3	9.3	4.5	4.5	9.45		
Polígono 13	**vértice (x)**	7.05	7.05	7.5	7.5	8	8	8.1	8.1
	vértice (y)	9.45	9.3	9.3	7	7	1.8	1.8	9.45
Polígono 14	**vértice (x)**	8.15	8.15	8.6	8.6	13.6	13.6		
	vértice (y)	9.45	7	7	8.4	8.4	9.45		
Polígono 15	**vértice (x)**	13.65	13.65	15.85	15.85	17	17		
	vértice (y)	9.45	9.3	9.3	3.35	3.35	9.45		
Polígono 16	**vértice (x)**	17.05	17.05	18.75	18.75	19.9	19.9		
	vértice (y)	9.45	8.5	8.5	3	3	9.45		
Polígono 17	**vértice (x)**	9	10	10	9				
	vértice (y)	3.3	3.3	7.6	7.6				
Polígono 18	**vértice (x)**	11	12	12	11				
	vértice (y)	7.6	7.6	3.3	3.3				
Polígono 19	**vértice (x)**	13	14	14	13				
	vértice (y)	3.3	3.3	7.6	7.6				

La representación de estos polígonos en el plano es la siguiente:

Figura C.1 Representación del laboratorio en el HMI

En la Figura C.1 se aprecia los obstáculos que conforman el plano del laboratorio y el polígonos más grande tiene 8 vértices, pero la gran mayoría tiene 6. Este

plano es el ideal para realizar las pruebas para el método de Grafo de Visibilidad ya que se divide el mapa de entorno en polígonos pequeños fáciles de procesar.

Figura C.2 Representación del laboratorio en el HMI para Descomposición de Celdas

Para el método de descomposición de Celdas, el entorno de laboratorio se cambia, reduciendo todos los obstáculos a 5, tres de los cuales son rectangulares, y los otros dos conforman el entorno las paredes y los escritorios del laboratorio que divide al mismo en dos obstáculos de 33 y 37 vértices.

El objetivo de esta modificación es aumentar el rendimiento del método planificador, disminuyendo la repetición de cálculos para la obtención de los nodos que forman el grafo de conectividad.

Figura C.3 Representación del laboratorio en el HMI para Campos Potenciales

El mapa de entorno dibujado aplicado para el método por Campos Potenciales, tiene 4 obstáculos, del cual se construye uno que contiene todo el entorno del laboratorio que incluye el ancho de las paredes para mantener lo más real posible la características del entorno para aplicar el método planificador.

Se funcionan todos los obstáculos para disminuir el tiempo que tarda el método en construir los obstáculos agregando los puntos que generan la fuerza de repulsión neta.

C.3 FOTOGRAFÍAS Y RESPUESTAS EN ENTORNO REAL

Figura C.4 Trayectoria recorrida por Robotino®, aplicando Grafo de Visibilidad

Figura C.5 Trayectoria recorrida por Robotino®, aplicando Descomposición de Celdas

Figura C.6 Trayectoria recorrida por Robotino®, aplicando Campos Potenciales

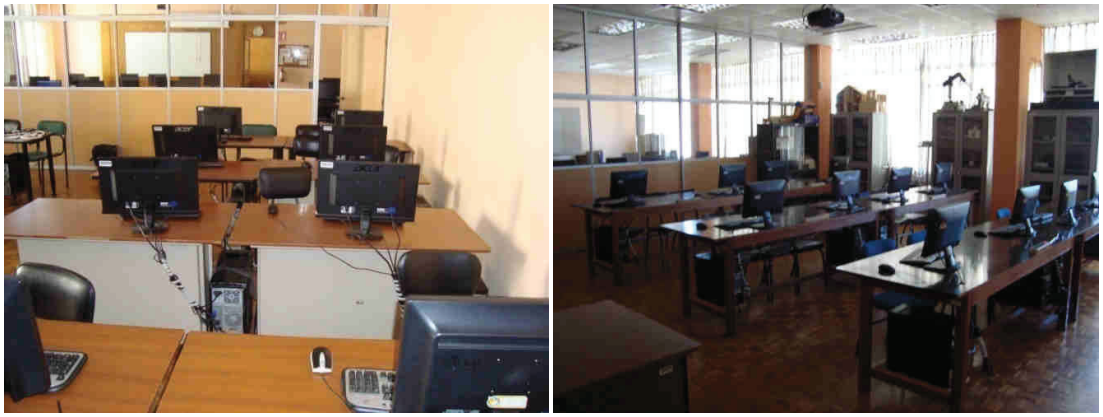


Figura C.8 Laboratorio de Microprocesadores

Figura C.9 Robotino® utilizado para realización de pruebas