OPTIMIZACION DE UN SISTEMA DE CONTROL PARA LA NAVEGACION BASADO EN EL METODO DE CAMPO DE FUERZA VIRTUAL PARA UN VEHICULO AUTONOMO EMPLEANDO ALGORITMOS GENETICOS

GERMAN EDUARDO CASTRO BURGOS
GREGORY RICARDO LEGARDA ROSERO

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA ELECTRÓNICA
SAN JUAN DE PASTO
2012

OPTIMIZACION DE UN SISTEMA DE CONTROL PARA LA NAVEGACION BASADO EN EL METODO DE CAMPO DE FUERZA VIRTUAL PARA UN VEHICULO AUTONOMO EMPLEANDO ALGORITMOS GENETICOS

GERMAN EDUARDO CASTRO BURGOS
GREGORY RICARDO LEGARDA ROSERO

Trabajo de grado presentado como requisito para obtener el título de Ingeniero Electrónico

Asesor: MG. JAIME ORLANDO RUIZ PAZOS

UNIVERSIDAD DE NARIÑO
FACULTAD DE INGENIERIA
PROGRAMA DE INGENIERIA ELECTRONICA
SAN JUAN DE PASTO
2012

“LAS IDEAS Y CONCLUSIONES APORTADAS EN ESTE TRABAJO DE GRADO, SON RESPONSABILIDAD EXCLUSIVA DE LOS AUTORES”

ARTÍCULO 1 DEL ACUERDO No. 324 DE OCTUBRE 11 DE 1966, EMANADO DEL HONORABLE CONSEJO DIRECTIVO DE LA UNIVERSIDAD DE NARIÑO

Nota de aceptación:

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

_____

Firma del jurado

_____

Firma del jurado

Pasto, Febrero de 2012

DEDICATORIA

“A mis padres que frente a todas las adversidades han sido el apoyo incondicional que siempre he necesitado, a mis hermanas por su ejemplo intachable e invaluables consejos, a mi compañero de tesis por compartir conmigo la experiencia de este proyecto, a Diana Maria Erazo por permitirme disfrutar de cada instante que comparto a su lado y a mis amigos y compañeros por hacer posible la realización de este proyecto”.

Germán Eduardo Castro Burgos.

“A mis padres que han sido mis mejores maestros, por darme su amor, guía, apoyo y colaboración incondicional a lo largo de toda mi vida; a mi hermano gemelo, quien siempre ha estado a mi lado, aun estando lejos; a mis hermanas por su paciencia y comprensión y amor; a mi compañero de tesis por compartir una experiencia de vida enriquecedora.”

Gregory Ricardo Legarda Rosero

RESUMEN

Los sistemas de navegación autónomos han sido objeto de continuo estudio y desarrollo durante las últimas décadas, debido en gran medida, al desarrollo de componentes embebidos de mayores prestaciones que facilitan la delegación de un mayor número de tareas de navegación a robots móviles de todo tipo. Los sistemas de navegación basados en el método de campo de fuerzas virtuales (*VFF Virtual Forced Field*) surgen entonces como una alternativa práctica y eficaz durante el desarrollo de vehículos autónomos dotados una red sensorial simple y que cuentan con una capacidad limitada de procesamiento tal y como ocurre en prototipos provistos de un sistema micro-controlado destinado a las labores de navegación.

No obstante, el desempeño de este tipo de sistemas se ve supeditado a la elección de una configuración eficiente para el conjunto de variables implicadas en el método, de manera tal que este proceso de búsqueda puede consumir una gran cantidad de recursos a expensas de la cantidad de posibles configuraciones existentes. Los algoritmos genéticos son empleados con frecuencia en procesos de búsqueda de condiciones similares a las expuestas, de manera tal que durante esta investigación, un prototipo de vehículo autónomo será desarrollado empleando como fundamento operativo el método de campo de fuerzas virtuales en un sistema embebido, presentando los resultados obtenidos durante la optimización del sistema a partir de la puesta en marcha de un algoritmo genético destinado para tal fin.

ABSTRACT

The Virtual Force Field method has been frequently used on autonomous navigation systems, due to the fact that permits the detection of unknown obstacles and the robot's move toward the target using inaccurate sensor data. The increase in the production of embedded technology with improved data process capacity, during the last decades, makes possible to implement navigation systems that deal with innumerable tasks. However, the performance of any VFF system depends on the value of a joint of variables used by the navigation method. Looking for the optimal configuration, a genetic algorithm can be used to search the needed values to warrant an expected mobile behavior.

This research was developed with the purpose of build an autonomous vehicle equipped with an embedded navigation system that compiles environment information since a low cost sensorial network. The system optimization was carried out using a genetic algorithm, to search an optimal configuration into the space of possible configurations. The results of this research show the suitability of use this kind of algorithm during the optimization process making evident an increase on the performance of the navigation system after the procedure.

CONTENIDO

Pág.

INTRODUCCION…… 24

1. DESCRIPCION DEL PROBLEMA…… 26

1.1 PLANTEAMIENTO DEL PROBLEMA…… 26

1.2 FORMULACION DEL PROBLEMA…… 27

2. OBJETIVOS…… 28

2.1 OBJETIVO GENERAL…… 28

2.2 OBJETIVOS ESPECIFICOS…… 28

3. JUSTIFICACION…… 29

4. REVISION SOBRE LA NAVEGACION PARA VEHICULOS AUTONOMOS…… 31

4.1 NAVEGACION EN ROBOTS MOVILES…… 31

4.2 ESQUEMAS DE NAVEGACION EN ROBOTS MOVILES…… 33

4.2.1 Percepción del entorno…… 33

4.2.2 Planificación de la ruta…… 33

4.2.3 Generación de la trayectoria…… 33

4.2.4 Seguimiento del camino…… 33

4.2.5 Planificación Global…… 36

4.2.6 Planificación Local…… 36

4.3 El problema de la planificación de ruta…… 38

4.3.1 Formalización teórica…… 39

4.4 Métodos empleados para la planificación…… 42

4.5 Principales enfoques para la planificación de trayectorias…… 43

4.5.1 Planificación basada en grafos de visibilidad…… 43

4.5.2 Planificación empleando diagramas de Voronoi…… 44

4.5.3 Planificación empleando el método probabilístico de carreteras…… 46

4.5.4 Planificación mediante descomposición en celdas……. 47

4.5.5 Planificación basada en el modelo de campos potenciales……. 49

4.5.6 Mejoras implementadas para el comportamiento dinámico de robots móviles que emplean el método *VFF*…… 53

4.5.6.1 Filtro pasa bajos para el control de giro……... 53

4.5.6.2 Amortiguamiento para el control de velocidad……. 54

4.5.6.3 Control de velocidad……. 56

4.5.7 Detección y solución de "trampas" en el entorno……... 56

4.5.7.1 Solución de trampas empleando el método *Wall-Following*……. 57

4.6 Sistemas de localización para robots móviles……... 58

4.6.1 Sistemas de localización empleando estaciones de monitoreo…….. 59

4.6.2 Sistemas de localización basados en ángulos ó Triangulación…….. 59

4.6.3 Sistemas de localización basados en distancias ó Trilateration…….. 60

4.6.4 Sistemas de localización basados en diferencia de distancias (TDOA)…… 61

4.6.5 Sistemas le localización basados en distancias absolutas (TOA)…….. 62

4.6.5.1 TOA Dos Dimensiones……. 63

4.6.5.2 TOA Tres Dimensiones……... 63

4.6.6 Sistemas de localización basados en el modelo cinemático del móvil.. 64

4.7 Estimación odométrica en una plataforma diferencial…….. 64

5. REVISION ACERCA DE LOS ALGORITMOS GENETICOS……. 68

5.1 Algoritmos Genéticos……. 69

5.2 Funcionamiento de un Algoritmo Genético Simple…….. 69

5.3 Codificación……. 71

5.4 Población……. 71

5.4.1 Tamaño de la población……. 71

5.4.2 Población inicial…….. 71

5.5 Función objetivo…….. 72

5.6 Selección…….. 73

5.7 Cruce…… 73

5.8 Mutación…….. 74

5.9 Reducción…… 74

6. PROTOTIPO DE VEHICULO AUTONOMO…… 75

6.1 Descripción General…….. 75

6.2 Estructura física…….. 75

6.3 Modulo Sensorial…… 79

6.4 Modulo de navegación…….. 81

6.5 Modulo para control de movimiento……. 83

6.6 Modulo de transmisión de datos…….. 84

6.7 Resumen de dispositivos empleados…….. 85

7. IMPLEMENTACION DEL SISTEMA DE NAVEGACIÓN EN EL VEHÍCULO AUTÓNOMO…….. 87

7.1 Implementación del sistema de navegación embebido…… 87

7.2 Recepción de comandos enviados desde el PC…….. 87

7.3 Determinación de la orientación y la velocidad para el desplazamiento en un instante de tiempo determinado…… 89

7.4 Localización del vehículo en el espacio de trabajo…… 96

7.5 Envío de datos hacia el PC…… 102

7.6 Orientación del vehículo en la dirección establecida…… 105

7.7 Recopilación de la información de los sensores de distancia…….. 110

7.8 Inclusión de los datos correspondientes a la distancia medida en la matriz VFF…… 113

7.9 Movimiento del servo-motor empleado por el sonar…….... 114

8. INTERFAZ DE COMUNICACIONES ENTRE EL VEHICULO Y EL PC…….. 117

8.1 Módulo para la transmisión y recepción de datos vía puerto USB……. 117

8.2 Interfaz gráfica para la presentación de datos en el PC…….. 120

8.3 Envío de los comandos de control……... 120

8.4 Recepción de los datos enviados por el vehículo……. 124

8.5 Representación gráfica de la información suministrada por el prototipo……... 127

8.6 Interacción entre el prototipo desarrollado y la interfaz de comunicaciones elaborada…… 131

9. OPTIMIZACION DEL SISTEMA DE NAVEGACION PARA EL VEHICULO AUTONOMO……... 133

9.1 Entorno de simulación para la optimización del sistema de navegación…….. 133

9.2 Generación de un entorno de trabajo…….. 134

9.3 Asignación de valores las variables necesarias para la operación del sistema de navegación……... 136

9.4 Simulación del comportamiento del móvil dentro de un entorno establecido….. 138

9.5 Optimización del sistema de navegación a partir de los valores asignados a las variables del sistema…… 147

10. RESULTADOS…… 156

10.1 Implementación del prototipo de vehículo autónomo…… 156

10.2 Resultados para el proceso de optimización del sistema de navegación implementado…… 159

10.2.1 Resultados optimización del sistema de navegación para el entorno1..... 161

10.3 Trayectoria optimizada versus trayectoria aleatoria……... 162

10.4 Comparación entre la trayectoria simulada y la trayectoria real…… 165

10.5 Resultados optimización del sistema de navegación para el entorno 2…….. 166

10.6 Trayectoria optimizada versus trayectoria aleatoria……... 167

10.7 Comparación entre la trayectoria simulada y la trayectoria real……... 169

10.8 Resultados optimización del sistema de navegación para el entorno 3…….. 170

10.9 Trayectoria optimizada versus trayectoria aleatoria……... 172

10.10 Comparación entre la trayectoria simulada y la trayectoria real……. 173

11. CONCLUCIONES……... 176

BIBLIOGRAFIA…… 179

LISTA DE FIGURAS

Pág.

Figura 1. Interrelación entre cada una de las tareas……..... 34

Figura 2. Esquema de navegación implementado por Nelson y Cox……. 35

Figura 3. Esquema propuesto por Goto y Stentz…….. 37

Figura 4. Esquemas básico basado en Subsumption architecture……. 38

Figura 5. Espacio de configuraciones espaciales del robot……. 40

Figura 6. Grafo de visibilidad uniendo las posiciones inicial (q_a) y final (q_f) y los vértices de cada obstáculo……. 44

Figura 7. Entorno delimitado por un polígono de aristas e_1,e_2,e_3,e_4, obstáculo triangular de vértices n_1,n_2, n_3 y aristas a_1,a_2, a_3. …….. 46

Figura 8. Mapa de caminos generado en un espacio bidimensional, compuesto de dos obstáculos (rojo) y veinte configuraciones libre (azul), unida en línea recta con sus k vecinos. La ruta entre las configuraciones iniciales y final (amarillo), se indica con una línea azul oscuro……. 47

Figura 9. Modelo del espacio libre para un entorno bidimensional……. 48

Figura 10. Implementación del método de descomposición en celdas……. 48

Figura 11. Esquema del campo de fuerzas virtuales…….. 51

Figura 12. (a) Trayectoria descrita por el robot, con un significativo número de oscilaciones (b) movimiento gracias al incremento en la magnitud de las fuerzas repulsivas…….. 55

Figura 13. Diagrama de fuerzas empleado durante la implementación de *WFM*…… 58

Figura 14. Localización de un vehículo en alta mar……... 59

Figura 15. Método TOA……. 60

Figura 16. Método TDOA…….. 61

Figura 17. Cálculo de la posición final mediante la intersección de todas las hipérbolas o hiperboloides…….. 62

Figura 18. Localización en el plano cartesiano del robot……. 65

Figura 19. Desplazamiento de un móvil durante un intervalo de tiempo (k-1,k)…….. 66

Figura 20. Configuración modular del móvil…….. 75

Figura 21. Imagen real del vehículo……. 76

Figura 22. Esquema y dimensiones del chasis construido……. 76

Figura 23. Motores asociados a cada una de las ruedas traseras……. 77

Figura 24. Rueda ubicada en la parte delantera de tipo libre……. 77

Figura 25. Rueda trasera asociada a cada motor……. 78

Figura 26. Servomotor marca Futaba de referencia S3003……. 78

Figura 27. Servomotor y sonar……. 79

Figura 28. Ubicación del sonar en el chasis…….. 79

Figura 29. Sensores SRF02 en el modulo sensorial……. 80

Figura 30. Transmisor de radio frecuencia, referencia XBee, estándar de transmisión IEEE 802.15.4 en la banda correspondiente a los 2.4GHz…….. 84

Figura 31. Módulo UART dispuesto en el micro-controlador maestro…….. 88

Figura 32. Diagrama de flujo empleado en la etapa de implementación…….. 92

Figura 33. Diagrama de flujo correspondiente a la función *cálculos*……. 94

Figura 34. Diagrama de flujo correspondiente a la función *posición_vehículo*……. 98

Figura 35. Diagrama de flujo de la función *leer_encoder*……. 99

Figura 36. El diagrama de flujo correspondiente a la función *recorrido*……. 100

Figura 37. Diagrama de flujo de la función *mover_matriz*…….. 103

Figura 38. Diagrama de flujo que indica el proceso llevado a cabo durante cada ejecución de la función *orientar* ……. 107

Figura 39. Diagrama de flujo en el cual se muestra la estructura de la función *I2C_motor*, que se utiliza para controlar el sentido de giro de los motores…….. 108

Figura 40. Diagrama de flujo en el cual se esquematiza la interrupción relacionada a la recepción de mensajes vía I2C…….. 109

Figura 41: Diagrama de flujo que bosqueja pseudocódigo empleando en la función *sensor_i2c* del programa principal……..... 112

Figura 42. Diagrama de flujo en el cual puede observarse el pseudocódigo empleado por la función *matriz_sensor*…….. 115

Figura 43. Esquema representativo, donde puede observase la separación angular de 30° y la sensibilidad con respecto a la distancia de rebote…….. 116

Figura 44. Diagrama de flujo en el cual puede observarse la gestión de recepción de datos provenientes del vehículo, la transmisión de estos hacia el PC…….. 119

Figura 45. Diagrama de flujo del esquema general del programa que coordina las actividades del módulo para la recepción y transmisión de datos vía puerto USB……. 119

Figura 46a. En la cual se muestra una caja de texto y un indicador visual -en este caso en color azul-, el cual significa que no hay conexión……. 121

Figura 46b. En la cual puede verse nuevamente la caja de texto y el indicador visual -en este caso en color verde-, que indica que hay conexión……. 121

Figura 47. Donde se indica el grupo de tres cajas de texto ubicadas sobre el *groupbox,* en las cuales se introducirán las variables seleccionadas X, Y, punto objetivo……. 122

Figura 48. Donde puede observarse el botón de *INICIAR, DETANER y BORRAR* dispuestos en la interfaz gráfica…….. 124

Figura 49. Donde puede observarse el cuadro donde se realiza la reconstrucción del entorno a partir de los datos asignados en la matriz……. 127

Figura 50. Donde se representa un entorno típico mediante la matriz de coeficientes de certeza……. 128

Figura 51. En la cual puede verse en color blanco el cuadro de imagen 2, el cual se emplea para la representación de la trayectoria que describe el móvil durante su desplazamiento……. 129

Figura 52. Donde puede observarse la disposición de todos los indicadores de desplazamiento, componentes, orientación posición y tarea…….. 130

Figura 53a. Ventana desplegada al pulsar el botón “guardar trayectoria…….. 130

Figura 53b. Donde puede observarse el archivo de texto generado…….. 131

Figura 54. Diagrama de bloques que bosqueja el funcionamiento del sistema vehículo-PC……. 132

Figura 55a. En la cual pueden observarse la disposición de los comandos descritos……. 135

Figura 55b. En la cual puede observarse un entorno desarrollado mediante la aplicación…….. 135

Figura 56. Donde se aprecia la ventana que se despliega cuando pulsando el botón guardar…….. 136

Figura 57. En la cual puede apreciarse la disposición del conjunto de controles para la asignación de valores a las variables para la operación del sistema de navegación……. 137

Figura 58. En el cual puede observarse el conjunto de contenedores de texto e indicadores visuales…… 139

Figura 59. Donde puede observarse el mensaje de alerta emitido tras la inclusión de un valor erróneo para una de las componentes del punto objetivo…..... 140

Figura 60. Diagrama de flujo que corresponde a la función *inicio*……... 141

Figura 61. Diagrama de flujo correspondiente al comportamiento efectuado por los sensores en busca de obstáculos presentes en el entorno……... 143

Figura 62. En la cual puede observarse una trayectoria descrita por el vehículo, desde el punto de inicio hasta el punto final u objetivo…….. 145

Figura 63a. En la cual puede observarse el cuadro desplegado cuando se presiona el botón *guardar_trayectoria*……... 147

Figura 63b. Datos del vector de posición empleado para la construcción gráfica de la trayectoria en el archivo de texto en columnas para cada componente…….. 147

Figura 64. En la cual se observa el cromosoma del individuo……... 148

Figura 65. Puede observarse la caja de texto 15, en la cual se introducirá el valor correspondiente a la condición de parada para el proceso de optimización…….. 149

Figura 66. Ejemplo que modela el cruce de 2 padres de la población total para un cruce definido en la posición 21 de la ristra……. 153

Figura 67. Cuadro de diálogo desplegado donde se solicita al usuario definir nombre y ubicación del archivo de texto en el cual está la información de la media del valor de adaptación, representación numérica del individuo mejor adaptado y valor en la función de adaptación…….. 155

Figura 68. Trayectoria constituida por los desplazamientos predeterminados, en un cuadrado de dimensiones 5x5m, el vehículo debe desplazarse en línea recta desde el punto de partida, hasta culminar el recorrido en el punto de inicio nuevamente……. 157

Figura 69a. Primer recorrido, se observa en rojo trayectoria deseada y en azul la trayectoria descrita por el móvil…….. 157

Figura 69b. Segundo recorrido, de igual forma en rojo trayectoria deseada y en azul la trayectoria descrita por el móvil……. 158

Figura 69c. Tercer recorrido, nuevamente en rojo se observa trayectoria deseada y en azul la trayectoria descrita por el móvil……. 158

Figura 70. En la cual puede apreciarse, el espacio físico empleado durante los recorridos reales, la parte en color rojo son los muros……... 160

Figura 71a. Primer entorno, constituido por un solo obstáculo cerca del centro del espacio de trabajo. …….. 160

Figura 71b. Segundo entorno, dos obstáculos rectangulares de diferentes dimensiones, en las posiciones observadas en la grafica……. 160

Figura 71c. Puede observarse el tercer entorno, con un mayor grado de complejidad, constituido por cuatro obstáculos rectangulares……..... 161

Figura 72. Donde puede observarse la curva de la progresión, que representa la evolución de la media del valor de adaptación……. 161

Figura 73. Recorrido que realiza el móvil empleando los valores de las constantes anteriormente, sirviéndose de la aplicación para simulación del comportamiento……. 162

Figura 74. Donde se observa la trayectoria generada por el simulador en una configuración aleatoria. …….. 163

Figura 75a. Comparación grafica entre los valores asociados a los parámetros de evaluación para desplazamientos sobre el entorno 1 antes de la optimización……... 164

Figura 75b. Comparación grafica de los valores asociados a los parámetros de evaluación para desplazamientos sobre el entorno 1 después de la optimización…….. 165

Figura 76. En la cual se presenta las trayectorias que fueron descritas por el móvil en el simulador y el espacio de trabajo real……. 165

Figura 77. Esquema de la progresión resultado de la media de los valores de adaptación de los individuos, generación a generación, con un aumento progresivo para el valor medio……. 166

Figura 78. En la cual puede observarse el recorrido efectuado por el vehículo empleando la optimización…….. 167

Figura 79. Trayectoria generada por el simulador empleando la configuración aleatoria, para el entorno numero 2……..... 168

Figura 80a. En la cual puede observarse el esquema comparativo de los valores obtenidos para la trayectoria optimizada y aleatoria…….. 169

Figura 80b. En la cual puede observarse la distancia recorrida, en la trayectoria optimizada y aleatoria…….. 169

Figura 81. En la cual se observa las trayectorias que fueron descritas tanto por el simulador, como por el prototipo real al emplear la configuración correspondiente al mejor individuo…….. 170

Figura 82. En la cual se modela el comportamiento de la media de los individuos, que componen la población que se emplea en el algoritmo genético durante del proceso……... 171

Figura 83. Se observa la trayectoria producto de la simulación del comportamiento del vehículo, con la configuración para el individuo más apto dentro del entorno……. 172

Figura 84. Donde se observa el comportamiento del vehículo debido a una configuración aleatoria establecida, para poder cuantificar los efectos de la optimización en las variables del sistema sobre la trayectoria…….. 172

Figura 85a. Donde se observa la comparación de los valores de los parámetros de evaluación para la trayectoria optimizada y aleatoria……. 174

Figura 85b. En la cual se hace una comparación de la distancia recorrida en la trayectoria aleatoria tanto como en la optimizada……... 174

Figura 86. En la cual puede observarse gráficamente, la trayectoria real tanto como la simulada para el entorno 3…….. 175

LISTA DE TABLAS

Pág.

Tabla 1. Registros internos del compás electrónico CMPS03…….. 82

Tabla 2. Resumen de los dispositivos empleados durante la implementación del sistema de navegación embebido…….. 86

Tabla 3. Comandos que pueden ser enviados……. 89

Tabla 4. Variables asignadas a la trama enviada constituida de 22 bytes…….. 104

Tabla 5. Relación lógica establecida para el motor derecho (motor 1)…… 108

Tabla 6. Relación lógica establecida para el motor izquierdo (motor 2)…….. 108

Tabla 7. Lógica del movimiento del vehículo, resultado de la interacción de los movimientos independientes de cada motor durante un intervalo de tiempo determinado……..... 110

Tabla 8. Donde puede verse las posiciones y su respectiva orientación relacionada al sonar……..... 114

Tabla 9. En la cual se muestra la asignación que corresponde a cada byte, según la variable que se envía……. 123

Tabla 10. Significado de los diferentes bytes recibidos, relacionado con la posición en la trama de recepción……..... 125

Tabla 11. Donde se muestran los nueve comportamientos para el valor hexadecimal del primer byte de la trama recibida que corresponde a la tarea que se encuentra en ejecución…… 126

Tabla 12. Bits correspondientes a cada variable e intervalo numérico…… 137

Tabla 13. Donde se observa el individuo mejor adaptado, su valor en la función de evaluación, también los valores asociados a cada constante del sistema, obtenidos con la configuración establecida……..... 162

Tabla 14. Donde pueden verse los datos de la configuración optimizada del móvil y una aleatoria, en la que las características se incluyen en la misma..... 163

Tabla 15. Donde puede observarse los datos sobre la función de adaptación y los valores asociados a cada constante del sistema, del individuo mejor adaptado al problema del entorno 2…….. 167

Tabla 16. En la cual pueden observarse los valores, haciendo uso de la configuración aleatoria, que contiene los valores para el sistema……. 168

Tabla 17. En la cual se muestra la representación binaria del individuo con un mayor valor para la función de evaluación……..... 171

Tabla 18. Representación binaria de los valores derivados, asociada al individuo aleatorio, así como los valores generados para los parámetros de evaluación durante el recorrido empleando la optimización……..... 173

LISTA DE ANEXOS

ANEXO A. Código del micro-controlador maestro del sistema de navegación….. 181

ANEXO B. Código del micro-controlador esclavo 1 del sistema de navegación.... 196

ANEXO C. Código del micro-controlador esclavo 2 del sistema de navegación.... 201

ANEXO D. Código del micro-controlador empleado para la adquisición de datos vía USB…….... 206

ANEXO E. Código de la interfaz gráfica para la adquisición de datos……..... 209

ANEXO F. Código del simulador empleado para la optimización del sistema…... 225

GLOSARIO

VFF(*Virtual Force Field Method*): Método de navegación basado en un modelo de campos potenciales que emplea una representación en grilla del entorno haciendo uso de coeficientes de certeza para llevar a cabo la reconstrucción del mismo a partir del establecimiento de potenciales de atracción y repulsión.

WFF(*Wall Following Method*): Método de navegación caracterizado por llevar a cabo desplazamientos de un móvil siguiendo el contorno de obstáculos presentes en el entorno.

NAVEGACION: Método que permite guiar el curso de un robot móvil a través de un entorno con obstáculos. La capacidad de reacción ante situaciones inesperadas debe ser la principal cualidad para desenvolverse, de modo eficaz en entorno no estructurados.

LOCALIZACION: Proceso mediante el cual es posible determinar la ubicación del móvil en el entorno de trabajo respecto a un punto o sistema de referencia.

PLANIFICACION: Proceso que permite encontrar una ruta segura capaz de llevar al móvil desde la posición actual hasta la especificada como destino. El concepto de ruta segura implica el cálculo de un camino al menos continuo en posición que sea libre de obstáculos.

AG: Sigla empleada para identificar a un algoritmo genético, algoritmo inspirado en la evolución biológica. Para su funcionamiento, estos últimos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan durante la evolución biológica, así como también a un proceso de selección de acuerdo con algún criterio.

ODOMETRIA: Técnica de posicionamiento que emplea información de sensores propio receptivos (aquellos que adquieren datos del propio sistema), para obtener una aproximación de la posición real en la que se encuentra un sistema móvil.

DSPIC: DSPic es un nombre genérico empleado para referirse a los controladores digitales de señales (DSC) que ha diseñado Microchip Technology Inc para facilitar a los usuarios la transición al campo de las aplicaciones de los procesos digitales de señales.

I2C: Bus de comunicaciones en serie. Su nombre viene de *Inter Integrated Circuit*. La versión 1.0 data de 1992 y la versión 2.1 del año 2000. La velocidad de transmisión corresponde a 100kbits en el modo estándar.

MDF: *Medium Density Fiberboard*. Correponde a un material que presenta un estructura uniforme y homogénea que permite que sus cantos tengan un acabado perfecto. Se trabaja igual a la madera maciza pudiéndose fresar y tallar.

PWM: Definida como *Modulación por ancho de Pulsos,* de una señal o fuente de energía, es una técnica en la que se modifica el ciclo de trabajo de una señal periódica ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

MIPS: Sigla utilizada para referirse a la cantidad máxima de instrucciones que un procesador puede realizar durante un periodo de un segundo.

INTRODUCCION

El desarrollo de sistemas de navegación para vehículos autónomos se ha constituido durante las últimas décadas en uno de los paradigmas objeto de investigación dentro del área de la robótica. Dichas investigaciones buscan implementar sistemas que permitan planificar la ruta que debe seguir un vehículo para llegar desde un punto a otro, considerando una serie de variables operativas entre las cuales pueden citarse, la presencia de obstáculos en el entorno, el conocimiento previo del mismo, el tiempo transcurrido durante el desplazamiento, la distancia recorrida para alcanzar el objetivo, etc.

Uno de los métodos empleados para determinar el comportamiento de un vehículo autónomo es el conocido como Método de Campo de Fuerzas Virtuales (Vff por sus siglas en inglés *Virtual Force Field Method*), en el cual, haciendo uso del modelo físico correspondiente al campo eléctrico, el vehículo se considera como un punto que interactúa dentro de un campo de fuerzas virtuales cuya sumatoria determina la dirección que debe seguirse para llegar hasta un punto objetivo evitando colisiones con los obstáculos presentes en el entorno.

No obstante la implementación de este método de navegación permite reducir significativamente la complejidad de los algoritmos necesarios para definir el comportamiento de un vehículo autónomo, su funcionamiento intrínseco está determinado por una serie de constantes matemáticas cuyas variaciones en conjunto tienen efectos significativos sobre la orientación y velocidad del móvil en un instante de tiempo determinado. Por lo tanto, resulta de suma importancia determinar qué conjunto de valores asignados a cada constante generan el comportamiento deseado por parte del vehículo, teniendo en cuenta un grupo de parámetros definidos a conveniencia que determinaran el buen o mal funcionamiento del sistema en un entorno dado.

La búsqueda de los parámetros del modelo se convierte entonces en un proceso que puede consumir una gran cantidad de tiempo y de recursos debido al elevado número de posibles combinaciones existentes. Resulta imprescindible aplicar técnicas de optimización que permiten simplificar dicho procedimiento empleando algoritmos de búsqueda sobre el espacio de posibles soluciones evaluando su adaptación al sistema, a partir de una serie de condiciones que determinen la calidad de la solución asociada a cada conjunto de valores.

Los algoritmos bio-inspirados se presentan como técnicas de optimización de grandes prestaciones para este tipo de sistemas dado que permiten realizar la construcción de modelos de cómputo encaminados a encontrar la solución a un determinado problema, empleando los recursos disponibles de la forma más eficiente, teniendo en consideración una serie de parámetros de evaluación tal y como ocurre en la naturaleza, entre ellos pueden citarse las redes neuronales, los algoritmos genéticos, la computación por enjambres, la lógica genética, los algoritmos basados en inteligencia colectiva, etc.

Los algoritmos genéticos se emplean como técnica de optimización para el sistema de navegación del vehículo autónomo basado en el método de campo de fuerzas virtuales, dadas las numerosas semejanzas existentes entre el conjunto de constantes pertenecientes a dicho sistema y el formato de cromosoma empleado por este tipo de procedimientos para llevar a cabo la búsqueda de la solución de mayor calidad.

La evolución de materiales semiconductores para la creación de dispositivos de procesamiento tales como Dspics y DSPs permite la implementación de un sistema de navegación embebido capaz de realizar la totalidad de las tareas necesarias para determinar el comportamiento del vehículo de forma autónoma y confiable.

El presente trabajo de grado muestra los resultados obtenidos a partir de la investigación correspondiente al desarrollo, optimización e implementación de un sistema de navegación para un vehículo autónomo, empleando como fundamento operativo el método de campo de fuerzas virtuales. Los resultados demuestran que es posible desarrollar un sistema embebido encargado de realizar todas las tareas de control necesarias para la navegación de un vehículo autónomo, mostrando la evolución del sistema al ser optimizado empleando algoritmos genéticos.

1. DESCRIPCION DEL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Durante las últimas décadas los numerosos avances en el desarrollo de sistemas autónomos han permitido otorgar un número de tareas cada vez mayor a dispositivos empleados en diversas actividades de la sociedad. Dentro de este ámbito los sistemas de navegación para robots móviles han adquirido importancia puesto que a partir de ellos pueden derivarse un conjunto de actividades de mayor complejidad. Diferentes sistemas de navegación basados en grafos de visibilidad, diagramas de Voronoi, modelamiento de espacio libre, descomposición en celdas, campos potenciales, entre otras técnicas, se han desarrollado con el fin de solventar las necesidades direccionales de un vehículo autónomo, cumpliendo satisfactoriamente dicha labor en su gran mayoría. Sin embargo, dado el costo computacional de los mismos su implementación en sistemas embebidos de bajo costo no es viable en la mayoría de los casos.

El método de campo de fuerzas virtuales *VFF,* al combinar las ventajas de las técnicas basadas en la descomposición en celdas y el modelo de campos potenciales reduce significativamente el tiempo de cómputo así como también la cantidad de memoria requerida para almacenar los datos necesarios para el funcionamiento de un sistema de navegación eficiente, haciendo posible su implementación en vehículos autónomos provistos de un sistema micro-controlado destinado al procesamiento de la información recopilada por una red sensorial con el fin de establecer la dirección y velocidad con las que el móvil debe desplazarse en un instante de tiempo determinado, para alcanzar un punto de destino sin colisionar con los objetos presentes en el entorno.

Sin embargo, para garantizar el correcto funcionamiento del sistema se hace necesario establecer un conjunto de valores asociados a las constantes matemáticas incluidas en el modelo, evaluando la calidad de cada posible solución a partir de ciertos parámetros relacionados con el comportamiento deseado para el móvil en un entorno determinado. Debido al elevado número de posibles configuraciones existentes, la búsqueda de la solución deseada se convierte en un proceso que puede consumir una gran cantidad de tiempo y recursos computacionales puesto que la evaluación debe hacerse sobre el sistema mismo puesto en marcha. Los algoritmos genéticos empleados como técnica de optimización se presentan como un método aplicable al sistema desarrollado

dadas sus ventajas sobre otros cuando se llevan a cabo búsquedas de configuraciones sobre espacios de posibles soluciones.

El sistema de navegación embebido desarrollado durante este proyecto realiza las tareas básicas para el direccionamiento de un móvil de forma autónoma, empleando el método de campo de fuerzas virtuales para tal fin. La optimización del mismo se lleva a cabo haciendo uso de algoritmos genéticos para encontrar la configuración más adecuada con el fin de evadir los obstáculos presentes en el entorno durante su recorrido siguiendo una serie de parámetros establecidos.

1.2 FORMULACION DEL PROBLEMA

¿Es posible implementar un sistema de navegación embebido para un vehículo autónomo tomando como fundamento el método de campo de fuerzas virtuales, llevando a cabo la optimización del mismo empleando algoritmos genéticos?

2. OBJETIVOS

2.1 OBJETIVO GENERAL

Diseñar e implementar un sistema de control de navegación embebido para un vehículo autónomo basado en el método de campo de fuerzas virtuales, optimizado mediante la aplicación de algoritmos genéticos.

2.2 OBJETIVOS ESPECIFICOS

Construir un vehículo terrestre dotado de un sistema de exploración del entorno que permita recopilar información acerca de los obstáculos presentes en el ambiente de trabajo.

Implementar un sistema de control de navegación basado en el método de campo de fuerzas virtuales que permita al vehículo evitar obstáculos mientras se mueve desde un punto de origen hasta un punto objetivo.

Implementar un algoritmo genético destinado a optimizar el sistema de control de navegación del vehículo, determinando los parámetros necesarios para el comportamiento óptimo del mismo.

3. JUSTIFICACION

Los sistemas autónomos día a día adquieren mayor importancia para la sociedad puesto que desarrollan innumerables tareas en diversos campos que abarcan la industria, la minería, la exploración espacial, la milicia, el transporte, las telecomunicaciones, entre otros. Su complejidad ha evolucionado rápidamente gracias al desarrollo de materiales semiconductores para la creación de dispositivos de procesamiento de mejores prestaciones que permiten disminuir el tiempo de cómputo e implementar sistemas que requieren almacenar un gran volumen de datos. Los sistemas de navegación para vehículos autónomos desempeñan un rol fundamental para mecanismos destinados a labores que requieren desplazamientos en ambientes desconocidos o que representan algún tipo de riesgo para el ser humano, siendo por tal motivo, materia de investigación durante las últimas décadas.

Diversas técnicas para el modelamiento del entorno han sido utilizadas en la implementación de sistemas de navegación autónomos; entre ellas pueden citarse, los diagramas de Voronoi, las técnicas de descomposición en celdas y finalmente el método de campo de fuerzas virtuales *VFF.* Aunque las dos primeras han demostrado ser efectivas para determinar el comportamiento de un móvil dentro de un entorno, el costo computacional de su operación es significativamente alto razón por la cual la implementación de las mismas en un sistema embebido de bajo costo es poco viable. La tercera técnica hace uso de las ventajas expuestas por el método de descomposición en celdas incluyendo a su vez la simplificación operativa derivada de los métodos basados en campos de fuerzas, reduciendo significativamente los recursos necesarios para su implementación en un sistema micro-controlado. Sin embargo, los valores correspondientes a la orientación y velocidad del móvil en un instante de tiempo determinado se ven estrechamente ligados a la configuración de una serie de constantes matemáticas incluidas en el modelo del sistema, evidenciando la necesidad de llevar a cabo un proceso de optimización.

Los algoritmos bio-inspirados empleados como técnicas de optimización han sido aplicados de manera satisfactoria en sistemas de control utilizados en diversos campos, generando por esta razón una rápida evolución de los métodos de búsqueda y clasificación de soluciones sobre un espacio de posibles configuraciones. Al considerar las características del sistema de navegación basado en el método de campo de fuerzas virtuales, se encuentran varias

similitudes con el formato empleado por los algoritmos genéticos para la búsqueda de una solución basada en la evaluación de cada posibilidad frente a un conjunto de parámetros establecidos, haciendo que su aplicación sobre el sistema en mención sea viable. Dado que a nivel regional existen un sinnúmero de procesos industriales y agrícolas que pueden ser sometidos a optimización resulta de suma importancia generar nuevos espacios investigativos que permitan establecer las pautas a seguir durante un procedimiento de esta índole.

El desarrollo de esta investigación permitirá que los estudiantes del departamento de Ingeniería Electrónica de la Universidad de Nariño, cuenten con un punto de partida para aplicar algoritmos bio-inspirados como técnicas de optimización en diversos sistemas de control. La optimización de los procesos relacionados a cada sistema permitirá reducir costos de operación y mantenimiento para industrias establecidas en el departamento de Nariño. Finalmente, el sistema de navegación autónomo creado durante este proyecto facilitará la recopilación de información en ambientes hostiles para el ser humano, siendo así una herramienta de mucha utilidad en el estudio de los fenómenos que afectan a la región.

4. REVISION SOBRE LA NAVEGACION PARA VEHICULOS AUTONOMOS

Este capítulo presenta una revisión de los conceptos fundamentales referentes a los sistemas de navegación para vehículos móviles describiendo los elementos que los componen y su interrelación. De igual manera se incluye una introducción al modelamiento dinámico de un vehículo autónomo y se describe el proceso llevado a cabo para determinar la posición de un móvil empleando sistemas odométricos.

4.1 NAVEGACION EN ROBOTS MOVILES

Durante las últimas décadas el desarrollo de robots móviles se ha visto estrechamente ligado al creciente interés de incluir sistemas autónomos en actividades no tradicionales que requieren además de la operación de sistemas manipuladores, desplazamientos e interacción entre dispositivos de toda índole. La exploración minera y espacial, los sistemas de vigilancia, el transporte de materiales en ambientes industriales, los sistemas de cosechado, etc, forman parte de los nuevos ámbitos de investigación en el área de la robótica.

Un vehículo autónomo puede ser definido como una plataforma mecánica dotada de un conjunto de sistemas capaces de realizar las tareas básicas para el guiado del móvil a través de un entorno con presencia de obstáculos y definir los parámetros necesarios para llevar a cabo las tareas de interacción con elementos del entorno u otros dispositivos. Los procesos necesarios para el desempeño de estas labores pueden agruparse en la mayoría de los casos, en cuatro controladores diferenciados; sistema de locomoción, sistema de planificación y supervisión, sistema de percepción y sistema de navegación. El primero determina la capacidad de maniobra del vehículo y hace referencia a sus características mecánicas. Generalmente la tracción del móvil se debe a un conjunto de ruedas que le permiten desplazarse por el entorno, sin embargo, en la actualidad se llevan a cabo innumerables investigaciones que buscan dotar a vehículos de sistemas de patas y articulaciones con el fin de permitir su operación en terrenos de características complejas. El sistema de planificación y supervisión está encargado de realizar el procesamiento de los datos necesarios para determinar el comportamiento del robot según cada labor específica, monitoreando el estado de los actuadores y demás controladores. La percepción del entorno es una tarea llevada a cabo a partir del análisis de la información entregada por una red sensorial que permite crear un modelo del entorno e identificar los obstáculos presentes en el mismo. Finalmente el sistema de navegación está encargado de

planificar los movimientos requeridos para desplazarse hasta el punto de destino evitando los obstáculos presentes en el entorno, empleando para tal fin el modelo del mismo.

Se puede definir entonces al sistema de navegación para un vehículo autónomo como la metodología encargada de determinar la trayectoria que debe seguir un móvil a través de un entorno con obstáculos, con el fin de alcanzar un punto objetivo. Aunque existen diversos esquemas que buscan abordar esta problemática, todos ellos tienen como finalidad incrementar la capacidad de reacción del móvil ante situaciones inesperadas en un entorno no estructurado. El conjunto de tareas que definen un sistema de navegación está compuesto como mínimo por la percepción del entorno a través de sensores, la planificación de una trayectoria de movimiento libre de obstáculos y finalmente el guiado del vehículo a través de la trayectoria definida. De forma similar, la interacción del móvil con elementos del entorno de trabajo define el concepto de operación como la programación de las herramientas de a bordo que le permiten a un sistema autónomo realizar cada tarea especificada.

Dada la importancia de los sistemas de navegación para el buen funcionamiento de diversos robots empleados principalmente en la industria, a partir de la segunda mitad del siglo XX se han desarrollado distintas investigaciones en este campo. En principio las exploraciones desarrolladas hacia finales de los sesenta estaban encaminadas a profundizar los conocimientos acerca de los problemas y posibles soluciones derivadas de los sistemas de inteligencia artificial para el modelo del entorno, dejando de lado la problemática sensorial y de posicionamiento. Los resultados obtenidos fueron significativamente escasos debido en gran medida a la simplicidad de los sistemas sensoriales implementados. Sistemas de navegación con grandes limitaciones aparecieron como resultado de la implementación de planificadores de trayectoria basados en la lógica computacional, que operaban a partir de mapas preconocidos del entorno, sin incluir en su estructura redes sensoriales para la adquisición de información acerca del ambiente de trabajo. Durante la década de los ochenta, el desarrollo de la exploración espacial impulsó una serie de investigaciones que trajeron como resultado sistemas de navegación parcialmente supervisados por operadores humanos, sin embargo, la capacidad de procesamiento de los dispositivos existentes hasta entonces limitaba en gran medida la autonomía de los vehículos. Durante las últimas dos décadas los grandes avances en la ciencia de los semiconductores han hecho posible la construcción de dispositivos de procesamiento de grandes prestaciones y reducido tamaño, fomentando el interés

en el desarrollo de sistemas de navegación autónomos que permitan realizar modelos del entorno en tiempo de ejecución a partir de información adquirida por sensores de varios tipos que se fusionan en una sola red. El desarrollo de sensores de mayor precisión, con un grado menor de susceptibilidad al ruido del ambiente ha hecho posible la implementación de sistemas de navegación basados en varias técnicas para el modelado del entorno, siendo implementados en vehículos multipropósito que se desempeñan en ambientes externos cuya estructura es modificada con suma frecuencia.

Aunque este conjunto de avances muestran un significativo desarrollo en este ámbito durante los últimos años, existen aún deficiencias en los sistemas de posicionamiento y modelado del entorno. Por tanto su optimización se ha convertido en un nuevo paradigma de investigación que presenta una amplia gama de posibles técnicas aplicables para este fin.

4.2 ESQUEMAS DE NAVEGACION EN ROBOTS MOVILES

Para un robot móvil se define una tarea de navegación como el conjunto de desplazamientos que deben realizarse para ir desde un punto de origen hasta un punto objetivo, la ejecución de esta labor está determinada por un grupo de cuatro procedimientos principales:

4.2.1 Percepción del entorno: Esta labor se lleva a cabo mediante el uso de una red sensorial externa que entrega información acerca de las características del entorno del vehículo. El análisis conjunto de estos datos permite generar un mapa o modelo del ambiente donde va a ser realizado cada desplazamiento.

4.2.2 Planificación de la ruta: Se define como el procedimiento encargado de establecer la secuencia de objetivos o sub-metas que deben ser alcanzadas partiendo de la información consignada en el modelo del entorno de acuerdo con los requerimientos de cada tarea de navegación específica, siguiendo los parámetros establecidos por un conjunto de directivas estratégicas.

4.2.3 Generación de la trayectoria: Se define como la creación de una función tal que sea capaz de interpolar los puntos establecidos durante la planificación de la ruta para posteriormente ser discretizada generando un conjunto de movimientos que pueden ser realizados por el vehículo con el fin de alcanzar el punto objetivo.

4.2.4 Seguimiento del camino: Establece la conexión entre el camino generado y el sistema de actuadores del vehículo, definiendo las señales de control

necesarias para cada uno de ellos a fin de realizar todos los desplazamientos descritos por la trayectoria establecida durante la tarea.

En cualquier esquema de navegación, la ejecución de estos procedimientos debe realizarse siguiendo el orden especificado, sin importar la simultaneidad de los cálculos realizados, determinando de esta manera la estructura básica de un control de navegación, según la metodología descrita por Shin y Singh en 1990.

La interrelación entre cada tarea se muestra en la figura 1.

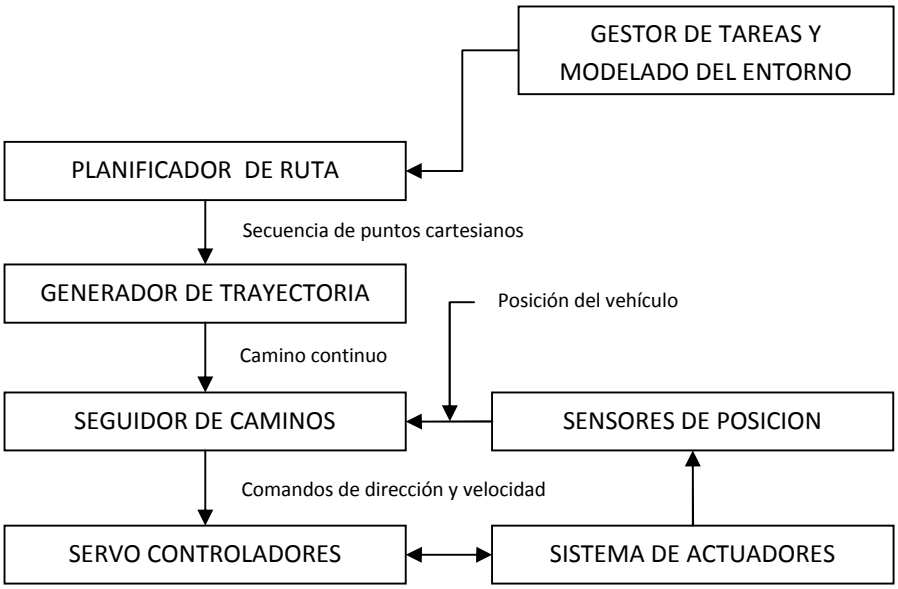


Figura 1. Interrelación entre cada una de las tareas

El funcionamiento del sistema se basa inicialmente en la información consignada en el mapa del entorno y en la tarea a realizar, determinando una trayectoria e ejecutar compuesta por un conjunto de puntos que definen la ruta a seguir, garantizando que esta misma se encuentre libre de obstáculos y permita alcanzar un punto objetivo. El seguidor de caminos se encarga de guiar al vehículo a través de la referencia establecida, empleando para tal fin la información suministrada por un conjunto de sensores que le permiten estimar la posición actual del vehículo y emplearla como parámetro de realimentación.

La fidelidad del sistema se ve afectada en gran medida por la calidad del modelo del entorno generado, puesto que en conjunto con el estimador de posición, constituyen la principal fuente de errores acumulativos que pueden llegar a

deteriorar significativamente los cálculos correspondientes a la orientación inmediata del móvil. El conocimiento del entorno puede variar desde un perfecto conocimiento del mismo hasta poseer un cierto grado de incertidumbre, haciendo necesaria la inclusión de esquemas que permitan mitigar el impacto generado por dichas variaciones. Un esquema de navegación empleado para tal fin, que corresponde al implementado por Nelson y Cox en 1990 se muestra en la figura 2:

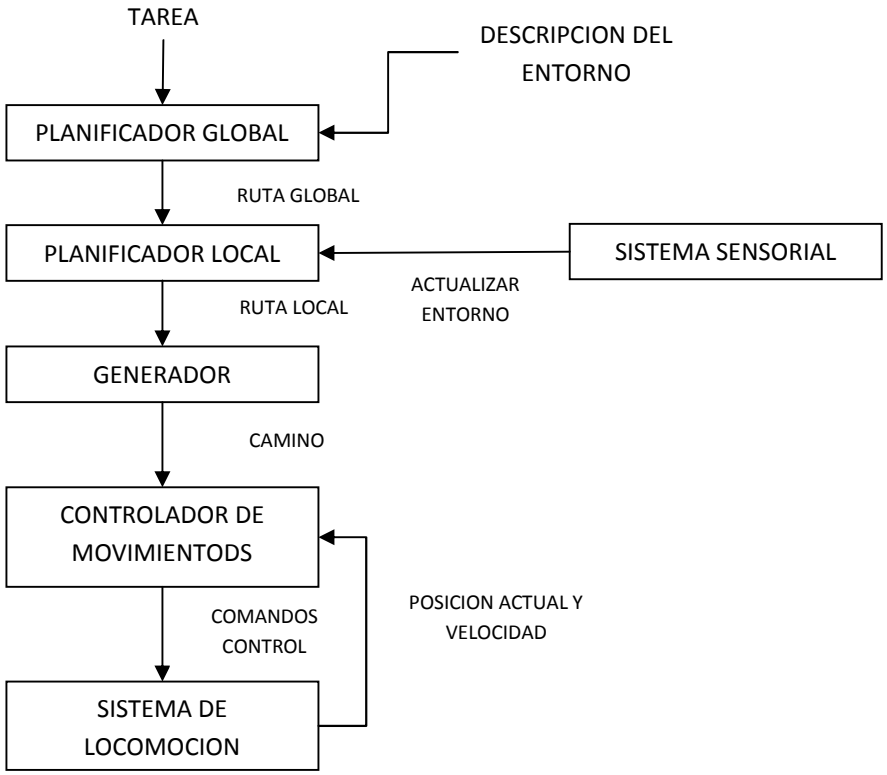


Figura 2. Esquema de navegación implementado por Nelson y Cox.

La principal novedad introducida en este sistema radica en la segmentación de la planificación de la ruta en dos procesos diferentes: planificación local y planificación global. El primero realiza al análisis de la información previa que se tiene acerca del entorno para definir una trayectoria libre de obstáculos que permita alcanzar el punto objetivo. El segundo se encarga de recopilar los datos provenientes del sistema sensorial del vehículo para determinar si han ocurrido cambios significativos dentro del radio de operación del sistema, realizando una actualización del modelo del entorno para determinar si la ruta a seguir debe ser planificada nuevamente.

Los métodos de planificación global y local pueden ser definidos como sigue:

4.2.5 Planificación Global: Proceso que se lleva a cabo con base en las metas geográficas (lugares) definidos durante la planeación de la misión. Estas metas globales dependen de la disponibilidad del entorno, las restricciones de movilidad, el comportamiento del vehículo, etc. Esta trayectoria es una aproximación del camino real sin considerar detalles de los elementos locales (obstáculos).

4.2.6 Planificación Local: La navegación local determina la trayectoria real, percatándose de las obstrucciones inesperadas sobre la trayectoria global. Para tal fin, un modelo del entorno local es construido empleando los datos sensoriales adquiridos[1].

La diferencia fundamental establecida entre los dos métodos radica el tiempo de ejecución de cada uno de ellos. El primero puede ser llevado a cabo antes de que el vehículo inicie la tarea de navegación, mientras que por su parte, el segundo se lleva a cabo en tiempo real, es decir mientras del vehículo se desplaza en busca de alcanzar el punto objetivo.

Cuando el conocimiento a priori del entorno es nulo (ambientes exteriores), el sistema de navegación se hace completamente dependiente de la planificación local de la ruta, de forma tal que el camino a seguir se construye de manera dinámica mientras el robot ejecuta el desplazamiento haciendo uso intensivo del sistema sensorial que a su vez, se encarga de coordinar las demás tareas subsecuentes. Un esquema de este tipo fue propuesto por Goto y Stentz en 1978 y su arquitectura se muestra en la figura 3.

Durante la ejecución de este esquema, una reconstrucción del entorno es llevada a cabo para cada sub-meta empleando la información sensorial del vehículo para a partir de ella definir una trayectoria segura que el seguidor de caminos tendrá como referencia. Tan pronto la sub-meta es alcanzada, es decir se concluye el recorrido sobre cada camino creado, el modelado del entorno vuelve a realizarse y el proceso se repite hasta que el punto objetivo sea alcanzado.

[1] Autonomus mobile robot… levi 1987.

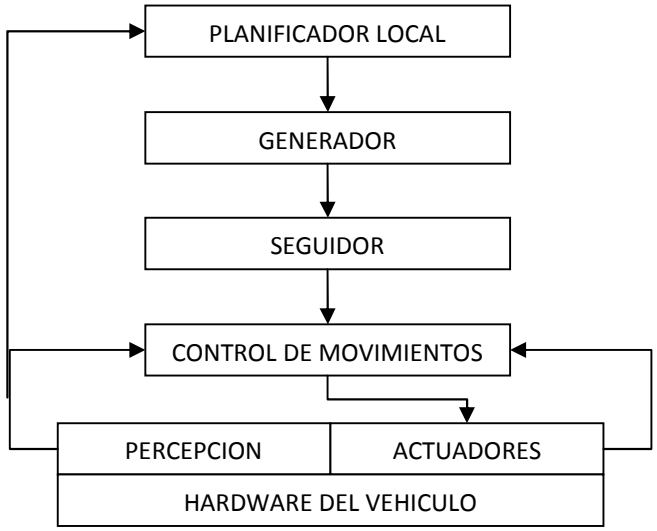


Figura 3. Esquema propuesto por Goto y Stentz.

Los tres esquemas anteriormente mostrados muestran cierta similitud en el orden de operación de cada sub-sistema, permitiendo maximizar los recursos operativos del vehículo a través de una descomposición jerárquica en módulos funcionales que encadenados en forma de ciclo realizan lo que se conoce como navegación estratégica, termino introducido por Brooks en 1986. Las limitaciones para este tipo de sistemas surgen al enfrentarse a entornos dinámicos puesto que en tal caso se hace necesario conocer de antemano el comportamiento dinámico de todos los elementos que interactúan en el ambiente y se requiere un modelo del entorno de alta calidad.

Un nuevo enfoque permite implementar sistemas de navegación altamente efectivos en ambientes dinámicos mediante el uso de sistemas sensoriales de bajo costo que permiten reaccionar rápidamente frente a cambios repentinos en el entorno, eliminando casi en su totalidad la necesidad de sistema de planificación de ruta y seguimiento de caminos. Este tipo de esquemas se basan en lo que se conoce como *Subsumption architecture,* concepto introducido por Brooks, que determina la descomposición del funcionamiento del sistema en una serie de comportamientos encaminados a cumplir con tareas individuales. El esquema básico se presenta en la figura 4.

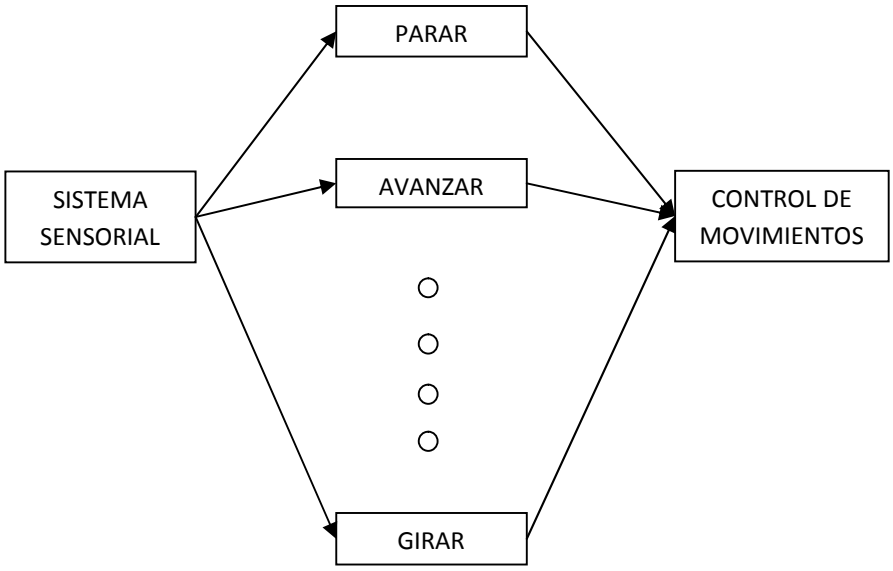


Figura 4. Esquemas básico basado en *Subsumption architecture*

En cada ciclo de trabajo, a partir de la información recopilada por los sensores el sistema habilita uno o varios comportamientos a la vez, determinando el funcionamiento conjunto del sistema de navegación. No obstante, en ausencia de un sistema de planificación de ruta, el vehículo puede desplazarse de forma errante a través del entorno.

La elección del esquema a implementar está estrechamente ligada al grado de adaptabilidad que debe poseer el vehículo, dependiendo de las características del entorno de trabajo, la tarea específica a ser realizada, el conocimiento previo del entorno y las características dinámicas del móvil y de los elementos constituyentes del ambiente.

4.3 EL PROBLEMA DE LA PLANIFICACION DE RUTA

El problema de la planificación de ruta en robótica puede ser definido como la necesidad de establecer una estructura operativa que permita determinar rutas, caminos y trayectorias necesarios para llevar a cabo de manera eficaz una tarea específica. La ruta se define como el conjunto de coordenadas espaciales que deben ser alcanzados por el robot, para desplazarse desde un punto de origen hasta un punto final desarrollando una serie de actividades secundarias. El camino por su parte, corresponde al conjunto de representaciones discretas de dichas coordenadas, que permiten interpolar las configuraciones expresadas por la ruta

mediante una función continua. Finalmente la trayectoria emplea la información discreta entregada por el camino para definir una secuencia de desplazamientos que cuentan con información cinemática propia de cada uno, es decir, que poseen un valor de velocidad específico asociado a cada intervalo de desplazamiento.

El entorno de trabajo del móvil puede ser representado como una serie de configuraciones espaciales que mediante desplazamientos pueden ser alcanzadas durante un tiempo determinado. De igual manera, existen configuraciones ocupadas por obstáculos que no pueden ser ocupadas por el vehículo sin importar la trayectoria ejecutada. La planificación se entiende entonces como el proceso mediante el cual se obtiene una trayectoria o camino cuya ruta está definida por una configuración inicial y otra final.[2]

4.3.1 Formalización teórica

Al considerar a un robot como un objeto rígido que tiene asociado a él un sistema de coordenadas móvil, es posible determinar una configuración *q* constituida por un vector que entrega la información pertinente acerca de la posición y orientación del móvil en un instante de tiempo determinado. Se establece un sistema de coordenadas global denominado F_g, la localización del vehículo se determina por la relación existente entre este sistema y el sistema local propio del robot llamado F_r.

Dado que el vector encargado de proporcionar la información acerca de la localización del móvil en el espacio está constituido por la componente de posición *p* y la de orientación θ, una configuración espacial en el entorno puede definirse como:

$$q = (p, \theta) = (x, y, \theta) \qquad (1)$$

El espacio de configuraciones espaciales *C* del robot *R* está constituido por todas las posiciones *q* que el móvil puede ocupar en el entorno. La porción de *C* que se encuentra ocupada por el robot cuando este se ubica en *q* se denomina *R(q)* y estará conformada por el espacio contenido dentro del circulo de radio ρ cuando el móvil se modela de forma circular como puede observarse en la figura 5.

$$R(q) = \{q_i \in C | \, \|q, q_i\| \leq \rho\} \qquad (2)$$

[2] F. Gómez-Bravo, F. Cuesta y A. Ollero, Planificación de trayectorias en robots móviles basada en técnicas de control de sistemas no holónomos, Depto. Ingeniería de Sistemas y Automática. Escuela Superior de Ingenieros, Universidad de Sevilla.

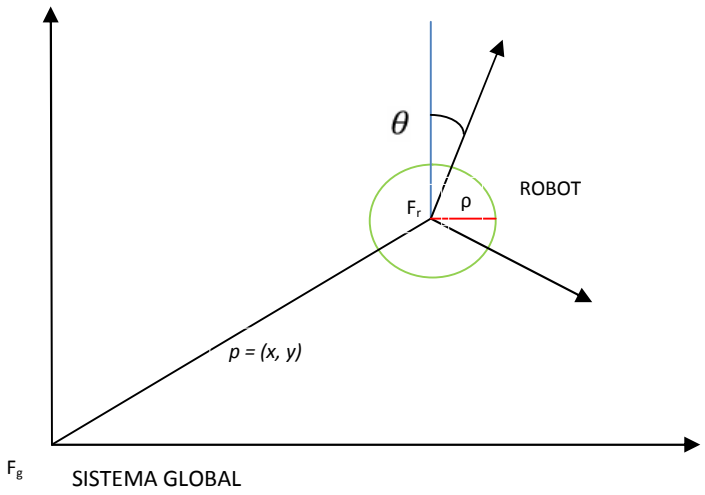


Figura 5. Espacio de configuraciones espaciales del robot.

Si el vehículo se modela como un objeto puntual, el espacio de configuraciones espaciales ocupadas por el mismo se reduce a una única posición *q* dado que el radio ρ es igual a cero.

$$R(q) = \{q\} \qquad (3)$$

La presencia de obstáculos en el entorno de trabajo considerados como un conjunto de objetos rígidos *B* determina una porción de configuraciones de *C* ocupadas por los mismos que forma parte del espacio total.

$$B = \{b_1, b_2, b_3, \ldots . . b_q\} \qquad (4)$$

Siendo $b_i(q)$ cada una de las posiciones del espacio *C* ocupadas por un obstáculo puede establecerse el subconjunto que define al espacio libre como:

$$C_l = \{q \in C \mid R(q) \cap \left(\bigcup_{i=1}^{q} b_i(q)\right) = \emptyset \qquad (5)$$

A partir de lo definido anteriormente puede apreciarse que el problema de la planificación de la trayectoria se reduce a la búsqueda de una sucesión de posiciones *q* pertenecientes al espacio de configuraciones C_l que permitan unir una posición inicial q_a con una posición final q_f. Por tanto, la ruta Q_r generada para tal fin está definida por:

$$Q_r = \{q_a, \ldots . , q_f \mid q_i \in C_l\} \qquad (6)$$

La función ruta τ estará definida de la siguiente manera:

$$\tau: [0,1] \rightarrow R \rightarrow C_l \qquad (7)$$

Tal que:

$$\tau(0) = q_a \qquad (8) \qquad\qquad \tau(1) = q_f \qquad (9)$$

Además de esta última restricción impuesta, la función de ruta deberá ser continua en el intervalo especificado dado que se asume un robot omnidireccional, puesto que está en capacidad de ser orientado en cualquier dirección requerida durante dos intervalos de desplazamiento consecutivos de forma tal que la siguiente expresión deberá siempre satisfacerse:

$$\lim_{s \to s_0} \|\tau(s)\| = 0 \qquad (10)$$

Un sistema holónomo es aquel en el que es posible expresar su comportamiento cinemático a partir de ecuaciones matemáticas de su posición o ecuaciones diferenciales integrales. El sistema no holónomo o mecánico será aquel en el que algunas de sus características cinemáticas no puedan ser representadas empleando ningún tipo de ecuación. La planificación de trayectorias para sistemas no holónomos se ve afectada por el hecho de que no puede usarse cualquier trayectoria para unir un punto de inicio con uno final dadas las restricciones establecidas para su movimiento. En este caso la búsqueda de la ruta debe enfocarse en encontrar una trayectoria que permita alcanzar la posición final sin violar las restricciones impuestas para el movimiento evaluando la admisibilidad de cada camino generado. Para ello se emplean técnicas de planificación basadas en la teoría de control, generando una serie de leyes que aplicadas al modelo del sistema permiten unir los puntos inicial y final por medio de una trayectoria sometida a integración numérica o analítica con el fin de satisfacer las restricciones cinemáticas.

La consideración de las restricciones impuestas por el entorno constituye otro parámetro de análisis en todo controlador para la planificación de ruta. Existen sistemas que evalúan el cumplimiento de este tipo de restricciones antes de planificar la ruta, mientras que por el contrario otro grupo de planificadores trazan la ruta a seguir para después someter a estudio los efectos de los limitantes sobre la trayectoria expuesta, en ambos casos, prevaleciendo como objetivo principal el llevar al móvil desde la posición de origen hacia la posición objetivo evitando colisiones con los obstáculos presentes en el entorno.

4.4 METODOS EMPLEADOS PARA LA PLANIFICACION

Los métodos empleados para realizar la planificación de una trayectoria tienen como objetivo común el llevar a un móvil desde una posición inicial hasta una posición final a través de una serie de recorridos libres de colisiones. Sin embargo, las características del entorno de trabajo y las restricciones cinemáticas en sistemas no holónomos tienen un gran impacto sobre el funcionamiento de cada modelo, principalmente en lo concerniente a tiempo de operación y a las limitaciones generadas en cuanto a la identificación de obstáculos. La implementación de cada modelo se ve sometida entonces a un análisis del entorno de trabajo y de las características propias del vehículo a emplear.

4.4.1 Clasificación de los modelos para la planificación de trayectoria.

Los modelos empleados para la planificación de una trayectoria pueden ser clasificados como dinámicos o estáticos dependiendo de las características de los elementos constituyentes del entorno. Si para un entorno determinado la posición de cada obstáculo permanece constante a través del tiempo, y se conoce con anterioridad, entonces el movimiento del vehículo se planea con anticipación y es posible hablar de un modelo de tipo estático. Si por el contrario, la composición del entorno varia dinámicamente a medida que el robot interactúa, se hace necesaria la adquisición de nueva información acerca del mismo, el modelo empleado se cataloga como de tipo dinámico puesto que la determinación de la trayectoria a seguir es un proceso continuo que debe ser llevado a cabo en tiempo de ejecución.

Si la resolución característica de cada modelo, definida como la unidad mínima de representación del entorno con la que el mismo cuenta, se toma como parámetro de clasificación, se establecen como categorías principales los modelos completos, heurísticos, de resolución completa y los probabilísticamente completos. Un modelo completo ejecuta la búsqueda de una solución y reporta el fallo del sistema de no encontrar ninguna, mediante una serie de algoritmos de alto costo operativo. Los modelos heurísticos se encargan de dar una rápida solución al problema de la trayectoria, sin embargo en un gran número de casos la misma resulta ser producto de mínimos locales que generan una respuesta sub-óptima. Los modelos de resolución completa realizan una división en forma de malla del espacio de configuraciones con el fin de encontrar aquellas posiciones que pueden ser alcanzadas por el vehículo. Finalmente los modelos probabilísticamente completos determinan que la probabilidad de encontrar un camino solución (ruta) tiende a uno cuando el tiempo permitido se incrementa hacia el infinito.

Si el parámetro empleado para realizar la clasificación es el alcance del modelo, aparecen como categorías principales los modelos globales y los modelos locales. Los primeros emplean toda la información acerca del medio y planifican la ruta desde el punto de inicio hasta el punto objetivo, mientras que los segundos no consideran para su operación la totalidad del espacio y por el contrario entregan soluciones rápidas enfocadas a la planificación de trayectorias de corta duración que son empleadas por los métodos globales para componer la ruta del movimiento total. El progreso de cada avance es representado por un grafo donde los nodos representan cada una de las posiciones alcanzadas y las aristas representan el segmento del movimiento realizado hasta el instante de tiempo actual.

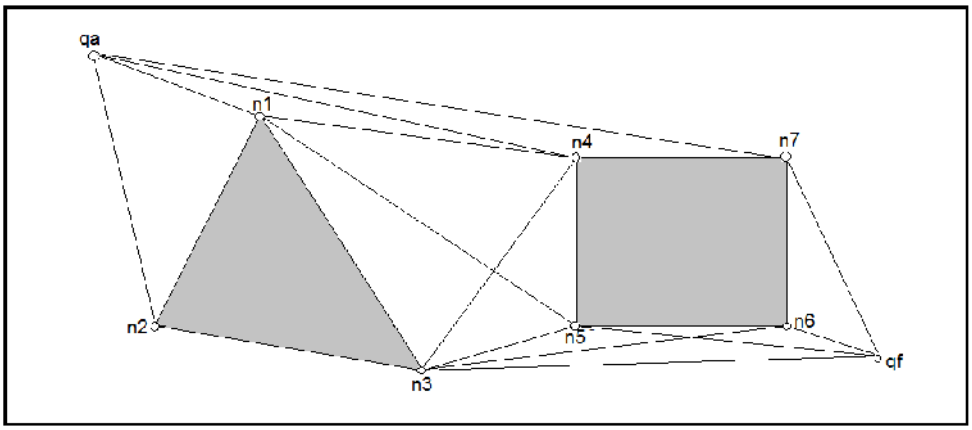
4.5 PRINCIPALES ENFOQUES PARA LA PLANIFICACIÓN DE TRAYECTORIAS.

La planificación de trayectorias en espacios altamente conocidos y estáticos se ha estudiado durante las últimas décadas. Aunque ha sido posible establecer una serie de modelos que permiten planificar rutas de forma efectiva en la actualidad las investigaciones se enfocan en el desarrollo de métodos de navegación que puedan ser implementados y puestos en marcha en entornos dinámicos que se ajustan en mayor medida a las necesidades reales del ámbito robótico.

4.5.1 Planificación basada en grafos de visibilidad.

Los grafos de visibilidad proporcionan un método geométrico de gran utilidad para la determinación de una ruta. Este método se caracteriza principalmente por presentar un tiempo de respuesta significativamente corto. El espacio ocupado por un obstáculo residente en el entorno se modela empleando figuras poligonales que encierran el conjunto de configuraciones correspondiente a dicha ubicación espacial. Para poder establecer la estructura de un grafo, el modelo incluye el concepto de visibilidad según el cual dos puntos del entorno son visibles si y solo si pueden ser unidos mediante una línea libre de obstáculos es decir que dicha trayectoria está constituida en su totalidad por posiciones libres del espacio de configuraciones.

Considerando como nodos del grafo de visibilidad la posición inicial, la posición final y cada uno de los vértices de los obstáculos presentes en el entorno, el grafo estará constituido por la unión mediante arcos de todos los nodos definidos como visibles. La figura 6 muestra el grafo de visibilidad construido uniendo las


qa
n1
n4
n7
n2
n5
n6
qf
n3

$$RT(q): C_l \rightarrow C_v / C_v \square C_l \qquad (11)$$

Así, la existencia de una trayectoria que permita unir una posición inicial q_a con otra final q_b está condicionada a la presencia de una curva continua que vaya desde $RT(q_a)$ hasta $RT(q_b)$.

La ventaja primordial que presenta este método con respecto a la planificación empleando grafos de visibilidad radica en el hecho de que la distancia existente entre la ruta trazada empleando diagramas de Voronoi y los obstáculos presentes en el entorno se maximiza garantizando que la trayectoria descrita por el móvil se encuentre lo más distante posible del contorno de los obstáculos presentes en el mismo evitando colisiones durante el desplazamiento.

Cada diagrama está compuesto por una serie de segmentos rectilíneos o parabólicos cuya elección está relacionada con el tipo de obstáculos enfrentados durante la determinación de la trayectoria. Así, si la trayectoria generada se encuentra a igual distancia de dos aristas pertenecientes a varios obstáculos, el segmento es rectilíneo y si por el contrario la misma se ubica entre una arista y un vértice será de tipo parabólico.

En la figura 7 se muestra el entorno delimitado por un polígono de aristas $e_1,e_2,e_3,e_4,$ y un obstáculo triangular constituido por los vértices n_1,n_2 y n_3 y las aristas a_1,a_2 y a_3. La función de retracción RT está representada por el conjunto de curvas C_v que constituyen el diagrama de Voronoi para esta configuración en particular.

Para determinar la ruta a seguir, el procedimiento generalmente inicia con la construcción del diagrama de Voronoi correspondiente a la situación en estudio para posteriormente determinar las funciones de retracción correspondientes al punto inicial ($RT(q_a)$) y punto final ($RT(q_b)$) mediante la curva de intersección entre cada uno de estos puntos y el vértice o arista más cercano. El paso siguiente corresponde al cálculo de todos los segmentos necesarios para construir la trayectoria entre q_a y q_b de manera tal que la no existencia de la misma también deberá ser considerada como resultado para la puesta en marcha del modelo.

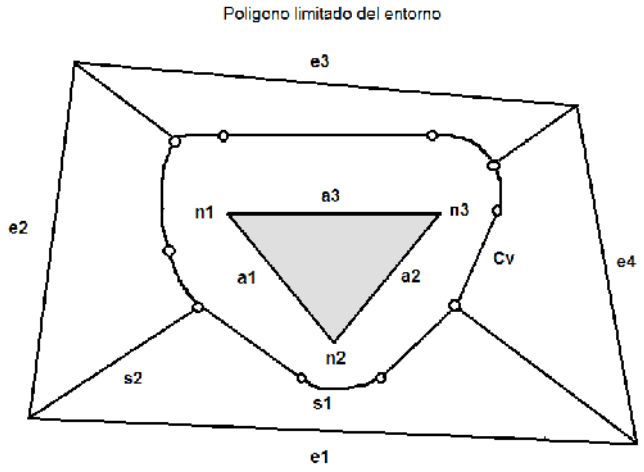
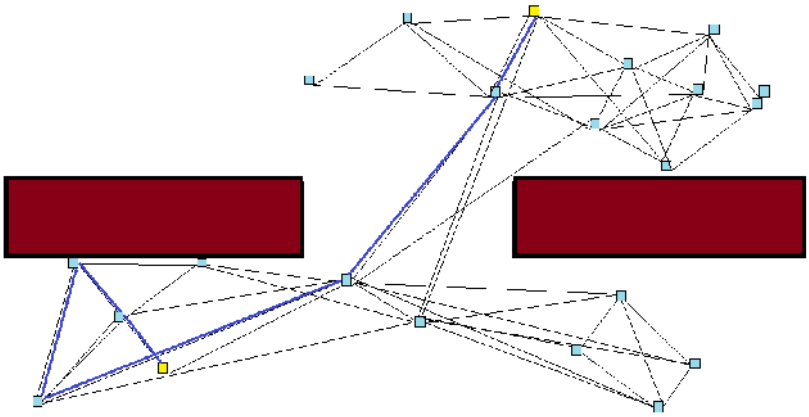


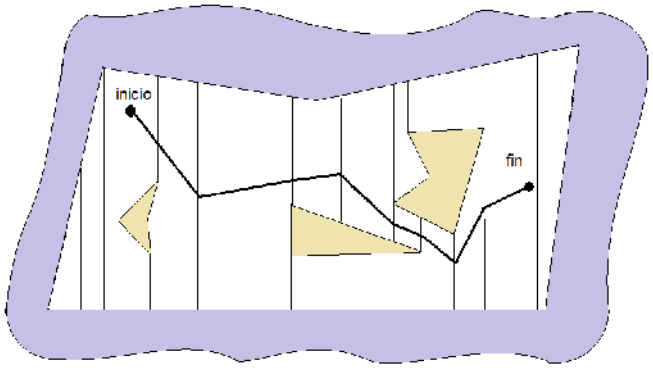
Figura 7. Entorno delimitado por un polígono de aristas $e_1,e_2,e_3,e_4,$ obstáculo triangular de vértices n_1,n_2,n_3 y aristas a_1,a_2,a_3.

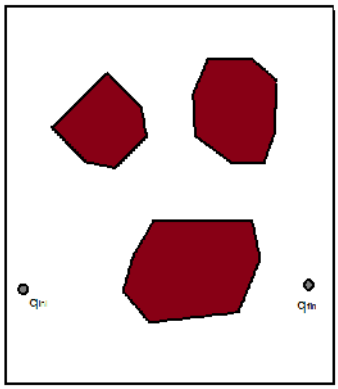
4.5.3 Planificación empleando el método probabilístico de carreteras.

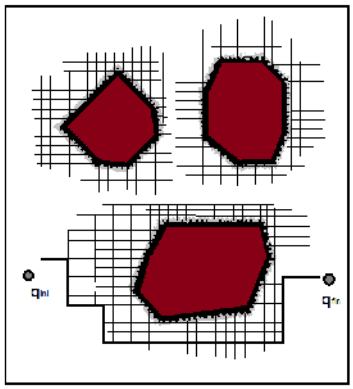
Este método está conformado por dos etapas principales, la primera se denominada *fase de construcción.* En esta fase el objetivo, es la construcción de un mapa de caminos (*Roadmap*) probabilístico que está representado por un grafo cuyos nodos corresponden a configuraciones libres de colisiones, y sus aristas son rutas factibles entre las mismas. La etapa correspondiente a la generación de configuraciones espaciales contenida dentro de la fase de construcción, consiste en generar un número *n* de posiciones libres de colisión de manera aleatoria uniforme en todo el espacio de trabajo. Si *n* es suficientemente grande el espacio de trabajo se cubre totalmente y el mapa del camino o grafo generado está completamente conectado. En caso contrario este estará formado por componentes no conectados indicando que no se ha capturado eficientemente la conectividad de dicho espacio. Posteriormente se tiene la etapa de conexión donde cada uno de los nodos se intenta conectar con sus *k* nodos vecinos más cercanos; estas líneas de conexión definen las aristas en el mapa del camino como se puede observar en la figura 8.

La segunda fase o de consulta intenta conectar la posiciones inicial y final de un robot a dos nodos pertenecientes al conjunto de configuraciones libres. Una vez conectados se busca una ruta que una a estos dos nodos, de forma tal que el método se califica como un éxito si se encuentra la ruta, y un fracaso en caso contrario.




inicio
fin





deseada y si no es posible establecer un camino el reporte corresponderá a un fallo en la trayectoria.

Mediante la puesta en marcha del esquema aproximado es posible encontrar una ruta partiendo de una resolución para la descomposición del espacio bastante burda que con el transcurrir de las iteraciones puede ser refinada con el fin de obtener una mejor solución, teniendo como limitante una resolución mínima para el modelo del espacio. Por lo tanto es posible que en situaciones particulares el método sea incapaz de encontrar la trayectoria buscada aunque esta misma exista.

El esquema exacto por el contrario, permite garantizar que de existir una trayectoria libre de obstáculos que una los puntos en estudio, el algoritmo de búsqueda será siempre capaz de encontrarla, sin embargo, el costo computacional y el tiempo de operación son mucho más elevados en comparación con el esquema aproximado.

4.5.5 Planificación basada en el modelo de campos potenciales.

Los métodos basados en el modelo de campos potenciales poseen un enfoque completamente diferente a los expuestos anteriormente. Se emplean generalmente con el fin de otorgar un comportamiento reactivo a sistemas de planificación de ruta expuestos a situaciones en las que se hace necesario evadir obstáculos en entornos totalmente desconocidos.

En 1989 Johann Borenstein y Yorem Koren plantean un método para la evasión de obstáculos en tiempo real denominado método para el Campo de Fuerzas Virtuales (*VFF, Virtual Force Field Method)*, en el cuál, un robot móvil es considerado como una partícula puntual que interactúa dentro de un campo de potencial artificial cuya intensidad está relacionada directamente con la estructura del entorno circundante. Para ello, el método hace uso de esquemas representados por valores de certeza que determinan la magnitud de las fuerzas de repulsión y atracción que experimenta el móvil a medida que se desplaza por un entorno desconocido. El valor de certeza puede ser definido como la cuantificación de la probabilidad asociada a la existencia de un obstáculo en el entorno de trabajo. La recopilación de la información concerniente al entorno es llevada a cabo mediante una red de sensores externos con un alcance limitado que determina el rango de operación del sistema.

Inicialmente el entorno de trabajo se descompone en celdas rectangulares de tamaño fijo. Durante el movimiento del robot, se realizan mediciones acerca de las características del entorno, proyectando los valores resultantes sobre la grilla de representación del área de trabajo, de forma tal que la respuesta del sistema sensorial corresponde a la distancia encontrada hasta uno o más obstáculos presentes en el ambiente. Para tal fin, se designa una grilla de menor tamaño con el robot ubicado siempre en el centro de la misma. Cada medición de distancia realizada por la red sensorial corresponde a una celda en específico que almacena un valor numérico incrementado en cada coincidencia. Toda celda ocupada por un obstáculo aplica una fuerza de repulsión sobre el robot que lo obliga a alejarse de la misma. La magnitud de esta fuerza es proporcional al valor de certeza contenido en la celda *C(i,j)*, e inversamente proporcional al cuadrado de la distancia comprendida entre la celda y el robot puesto que las celdas ocupadas que se encuentran próximas al vehículo ejercen una gran fuerza de repulsión sobre el mismo, en tanto que las celdas ocupadas distantes, ejercen una fuerza de magnitud menor.

$$F(i,j) = \frac{F_{cr} C(i,j)}{d^2(i,j)} \left[\frac{x_i - x_0}{d(i,j)} \hat{x} + \frac{y_i - y_0}{d(i,j)} \hat{y} \right] \qquad (12)$$

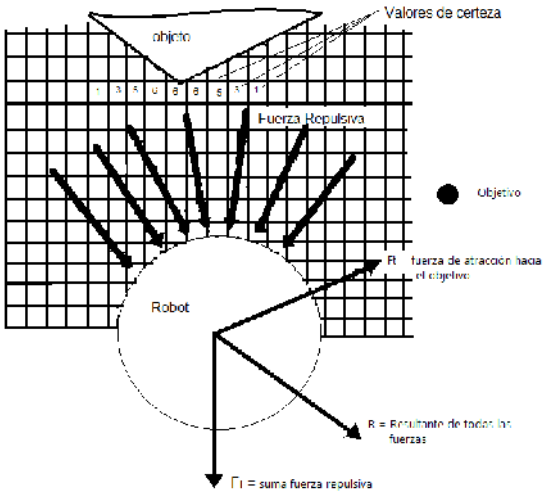
Donde:
F_{cr} = *Constante de fuerza de repulsión*
d(i,j) = Distancia entre la celda (i,j) y el robot
C(i,j) = Valor de certeza de la celda (i,j)
x_0, y_0 = *coordenadas presentes del robot*
x_i, y_i = *coordenadas de la celda (i,j)*

La fuerza de repulsión resultante se define como la suma vectorial de las fuerzas individuales generadas por cada celda.

$$F_r = \sum_{i,j} F(i,j) \qquad (13)$$

Simultáneamente, a medida que el robot se desplaza, una fuerza de magnitud constante F_t empuja al robot hacia el punto objetivo. Esta fuerza se denomina fuerza de atracción y depende de forma directa del valor asociado a la constante de proporcionalidad (F_{ct}) definida para el sistema.

$$F_t = F_{ct} \left[\frac{x_t - x_0}{d(t)} \hat{x} + \frac{y_t - y_0}{d(t)} \hat{y} \right] \qquad (14)$$



$$\boldsymbol{R} \quad \boldsymbol{F_r} \quad \boldsymbol{F_t}$$

$$\delta \quad \boldsymbol{R} \quad \boldsymbol{R}$$

$$\Omega \quad K_s \dot{\theta} \quad \delta$$

diferencia rotacional entre α y β. Por tanto γ estará siempre comprendido en el rango $-180^{o}<\gamma<180^{o}$.[3]

La representación de un entorno constituido por un conjunto de obstáculos mediante el uso de una grilla de certeza muestra los cuerpos de cada objeto como una serie de celdas cercanas que poseen un elevado coeficiente de certeza, mientras que por el contrario las lecturas erróneas entregadas por el sistema sensorial del robot están representadas por celdas aisladas que poseen bajos coeficientes. La orientación del vector fuerza repulsiva está entonces determinada por los cúmulos de celdas ocupadas correspondientes a la posición de los obstáculos mientras que cada celda ocupada como consecuencia de una lectura errónea del sistema sensorial tiene un efecto casi nulo sobre dicho vector resultante.

Frente a otros métodos para la planificación de ruta en entornos desconocidos, el método *VFF* provee una serie de soluciones ventajosas respecto a sus similares. En la mayoría de métodos implementados para la reconstrucción del entorno y la posterior planificación de ruta las lecturas erróneas entregadas por los diferentes tipos de sensores dedicados a la recopilación de información acerca del ambiente de trabajo tienen un efecto significativo sobre el comportamiento del sistema, puesto que en algunos casos pueden ser interpretadas como obstáculos presentes en una determinada posición sin que en la realidad la situación lo amerite. Por el contrario, el método *VFF* al evitar el uso de contornos para establecer la posición de un obstáculo permite minimizar los efectos de las lecturas erróneas. De igual manera, este método no requiere que el robot se detenga para realizar la reconstrucción del entorno o definir cuál es la orientación a seguir, permitiendo que el sorteo de los obstáculos se presente durante su movimiento aumentando la velocidad del desplazamiento general. La capacidad para realizar los procesos de recopilación sensorial y el cálculo de la orientación del móvil empleando la grilla de certezas de forma simultánea constituye una mejora significativa en comparación con los métodos basados en grafos puesto que para estos últimos es indispensable realizar una serie de tareas de navegación siguiendo una jerarquía inamovible generando incrementos sustanciales en el tiempo de ejecución y decrementos en la capacidad de reacción del sistema frente a cambios inesperados. Finalmente, la representación del entorno empleando la descomposición en celdas permite fusionar la información

[3] J. Borenstein, Y. Koren, Real-Time Obstacle Avoidance for Fast Mobile Robots, Technical Report, *IEEE Transactions on Systems Man and Cybernetics. Vol 19* no. 25, September/October 1989.

recopilada por diferentes tipos de sensores de forma rápida y sencilla, sin costos computacionales adicionales.

4.5.6 Mejoras implementadas para el comportamiento dinámico de robots móviles que emplean el método VFF.

Las características dinámicas de cada tipo de robot imponen una serie de restricciones que deben considerarse para realizar desplazamientos durante la navegación empleando el método de campo de fuerzas virtuales. Estas restricciones en su mayoría están asociadas a la velocidad real que puede alcanzar el móvil durante un desplazamiento, el ángulo de giro máximo para un determinado cambio de orientación y la velocidad de respuesta frente a cambios de orientación inesperados que pueden aparecer como resultado de incrementos o decrementos significativos en la magnitud y orientación del vector que representa la fuerza repulsiva del entorno en un instante de tiempo dado. Una serie de controladores adicionales han sido desarrollados con el fin de mitigar el impacto de las condiciones dinámicas del móvil empleado, estableciendo leyes de control para diferentes variables componentes del sistema autónomo. A continuación se presentan tres casos particulares.

4.5.6.1 Filtro pasa bajos para el control de giro.

En busca de obtener un movimiento continuo y sin oscilaciones de gran magnitud, el método *VFF* requiere el cumplimiento de la siguiente condición que relaciona la resolución de la grilla de certeza (Δ*s*) con el tiempo de muestreo (*T*) correspondiente al cálculo de la orientación deseada para el móvil se cumpla.

$$\Delta s > TV_{max} \qquad (17)$$

Donde V_{max} representa la velocidad máxima con la que el vehículo puede desplazarse determinada a partir del comportamiento cinemático del mismo.

Esta exigencia está fundamentada en el hecho de que la fuerza repulsiva total generada por el entorno, esta cuantizada en valores relativos a la resolución de la grilla de certeza y a su vez se ve afectada por la distancia existente entre el móvil y los obstáculos, de forma tal que grandes cambios en el vector resultante ***R*** pueden producirse a medida que el móvil se desplaza de una celda a otra, incluso si la condición establecida en (17) se satisface, generando una serie de oscilaciones a medida que el robot intenta ajustar su dirección a la deseada. Para solucionar este inconveniente, un filtro pasa bajos digital se adiciona al control de giro del sistema mediante la ecuación mostrada en (18).

$$\Omega_i = \frac{T\Omega'_i + (\tau - T)\Omega_{i-1}}{\tau} \qquad (18)$$

Donde:
Ω_i = magnitud del giro del robot (después de aplicar el filtro).
Ω_{i-1} = magnitud de giro anterior.
Ω'_i = magnitud de giro (antes de aplicar el filtro).
T = tiempo de muestreo del sistema
τ = constante de tiempo del filtro pasa bajos.

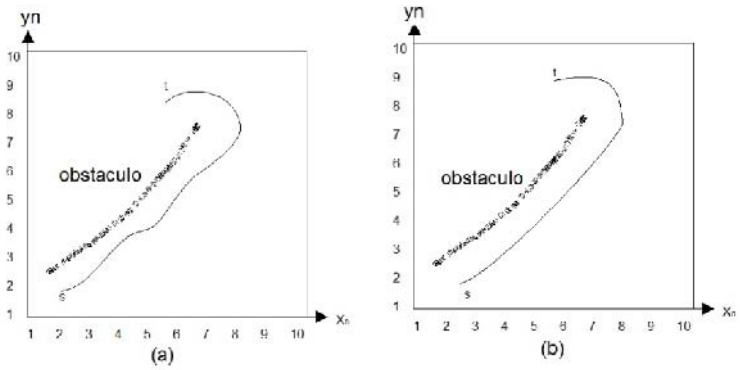
La implementación de este tipo de filtro permite suavizar el movimiento del robot a medida que este se desplaza paralelo a un obstáculo debido a que suprime en gran medida las variaciones instantáneas en la magnitud de giro que debe realizar el móvil para alcanzar la orientación deseada. No obstante el filtro introduce un retardo de valor correspondiente a $\tau\Omega$ en la respuesta del sistema de control de giro.

4.5.6.2 Amortiguamiento para el control de velocidad.

Idealmente cuando el robot encuentra un obstáculo, debería moverse suavemente de forma paralela al mismo hasta que sea posible girar nuevamente en dirección al objetivo. Sin embargo, los efectos combinados de la inercia del sistema y el filtro pasa bajos introducen un retardo considerable en la generación de los controles de giro y orientación. Como resultado, el robot puede aproximarse considerablemente a un obstáculo incluso si el sistema genera grandes fuerzas de repulsión. Cuando el robot finalmente gira para evadir el obstáculo es posible que se aleje más de lo necesario del mismo como consecuencia de este efecto, ocasionando que la trayectoria descrita durante el movimiento este compuesta de un significativo número de oscilaciones tal y como se muestra en la figura 12.

Una alternativa eficiente para reducir las oscilaciones durante el movimiento consiste en incrementar la magnitud de las fuerzas repulsivas cuando el robot se mueve en dirección a un obstáculo y reducirla cuando el movimiento se lleva a cabo de forma paralela a los mismos. Para conseguir este efecto, la magnitud de la fuerza de repulsión debe ser establecida como una función dependiente de la orientación relativa entre el vector correspondiente a la fuerza repulsiva resultante $\boldsymbol{F_r}$ y el vector que representa la velocidad instantánea del movimiento $\boldsymbol{V}$. Matemáticamente, la función puede establecerse multiplicando el vector de fuerza

$$F_r' \quad wF_r \qquad w\ F_r \quad cos\theta$$



$$cos\theta \quad \frac{V_x F_{rx} + V_y F_{ry}}{\boldsymbol{V}\ \boldsymbol{F_r}}$$

obstáculo a medida que se aproxima a este desde una orientación correspondiente a un ángulo pequeño.

4.5.6.3 Control de velocidad.

De forma intuitiva, el control de la velocidad de un móvil que se desplaza empleando el método de campos virtuales consiste en establecer el valor de la misma como una cantidad proporcional a la magnitud de la resultante de todas las fuerzas que interactúan en el sistema $\boldsymbol{R} = \boldsymbol{F_r} + \boldsymbol{F_t}$ de manera tal que cuando el vehículo se desplaza en un ambiente libre de obstáculos el campo de potencial total del sistema está relacionado únicamente con la fuerza de atracción al objetivo y la velocidad del desplazamiento es máxima. La presencia de fuerzas repulsivas asociadas a obstáculos presentes en el entorno disminuye significativamente el valor de la magnitud de la fuerza resultante ***R*** dado que su dirección es opuesta a la que posee la fuerza atractiva, reduciendo a su vez el valor de la velocidad del desplazamiento de manera efectiva. No obstante, el coseno del ángulo direccional existente entre los vectores $\boldsymbol{F_r}$ y ***V*** también puede ser utilizado con el mismo fin estableciendo la siguiente relación:

$$V = \begin{cases} V_{max} & para\ |F_r = 0| \quad ausencia\ de\ obstáculos \\ V_{max}(1 - cos\theta) & para\ |F_r| > 0 \end{cases} \qquad (21)$$

Donde V_{max} corresponde al valor máximo de la velocidad con la que el móvil puede desplazarse durante la ejecución de una tarea.

Al aplicar esta función, el robot aún se mueve a su velocidad máxima cuando el entorno local se presenta libre de obstáculos. En presencia de algún objeto en el entorno, la velocidad del desplazamiento solo se reduce si el robot se desplaza directamente hacia el contorno del obstáculo. Cuando el movimiento se produce en forma paralela al mismo, la magnitud de la velocidad se reduce en un valor mínimo y se mantiene cercana al parámetro máximo del robot.

4.5.7 Detección y solución de “trampas” en el entorno.

Uno de los problemas que ocurren con mayor frecuencia en sistemas de navegación basados en el método de campo de fuerzas virtuales es la ocurrencia esporádica de situaciones en las que el robot puede asumir comportamientos erráticos repetitivos que lo mantienen desplazándose sobre una trayectoria cerrada. Estos escenarios se conocen como trampas y son generados por diversas configuraciones espaciales.

Para determinar en qué momento una situación se convierte en trampa es necesario monitorear la diferencia existente entre el ángulo de atracción al punto objetivo θ_t y la dirección actual de desplazamiento θ_0. Si la diferencia entre estos dos ángulos resulta ser mayor que 90 grados, el vehículo está alejándose de la orientación hacia el objetivo y resulta muy probable que este enfrentando una situación similar a las descritas anteriormente.

$$|\theta_t - \theta_0| > 90° \quad (22)$$

Para evadir la condición de trampa, el controlador de navegación monitorea el resultado de la relación expresada en (22). Si la relación se cumple, el sistema selecciona el algoritmo de recuperación de trampas y lo establece como directriz operativa para la navegación mientras la condición se cumpla. Existen situaciones en las que la condición (22) puede cumplirse sin que exista una trampa en el entorno. Sin embargo, la aplicación del algoritmo de recuperación de manera temporal no afecta el desempeño del sistema en general.

4.5.7.1 Solución de trampas empleando el método Seguimiento de Contorno (WFM, *Wall-Following Method).*

El algoritmo empleado para resolver las situaciones críticas descritas anteriormente se denomina Seguimiento de Contorno (WFM, *Wall Following Method).* Consiste en establecer una fuerza atractiva momentánea que obligue al móvil a desplazarse siguiendo el contorno de un obstáculo ubicado frente a el, siempre y cuando la condición establecida en (22) se cumpla. Cuando el robot está orientado en una dirección que respecto al punto objetivo es menor de 90 grados el control de navegación retoma el método *VFF* como principio operativo y continúa con su normal ejecución de no detectarse una nueva condición de trampa.

La implementación del método se lleva a cabo de la siguiente manera. Inicialmente el sistema calcula la fuerza repulsiva resultante $\boldsymbol{F_r}$ con el fin de establecer cuál es su dirección (θ_r). A continuación, un ángulo α comprendido entre 90 y 180 grados es sumado al ángulo correspondiente a la orientación de la fuerza repulsiva si se quiere seguir el contorno de un obstáculo a la izquierda del vehículo. Por el contrario, si el ángulo establecido se resta a la orientación de $\boldsymbol{F_r}$ el vehículo sigue el contorno del obstáculo ubicado a su derecha. Posteriormente, el algoritmo proyecta una fuerza repulsiva temporal $\boldsymbol{F_t'}$ en la dirección resultante que momentáneamente reemplaza a la fuerza de atracción generada por el punto

objetivo F_0. El nuevo vector resultante ***R*** apunta o en su defecto converge a una orientación paralela al contorno del obstáculo ubicado frente al vehículo. En la figura 13 se presenta el diagrama de fuerzas empleado durante la implementación de *WFM.*[4]

4.6 SISTEMAS DE LOCALIZACION PARA ROBOTS MOVILES

La localización de un robot en el entorno constituye uno de los problemas clásicos en el estudio de la robótica móvil definida como aquel conjunto de técnicas que se encargan de solucionar el problema de determinar la ubicación de un robot en su entorno usando sensores propios. La mayor parte de las tareas que un robot móvil tiene que realizar dependen de su posición, puesto que a partir de ella es posible llevar a cabo la interacción con el entorno y el proceso de navegación.

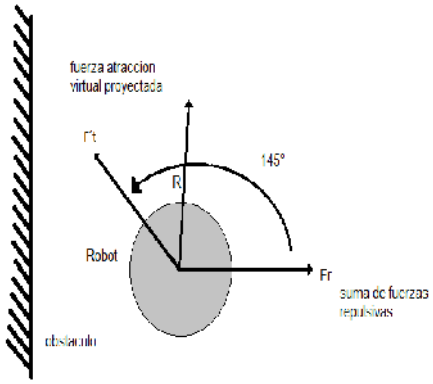
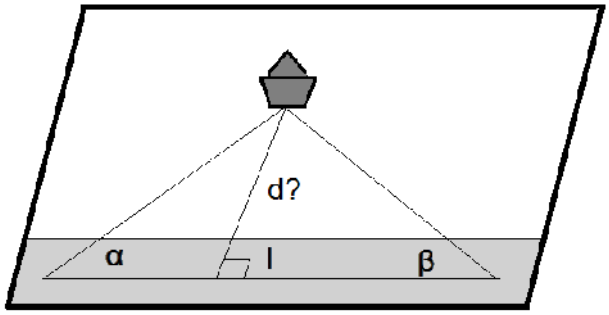


Figura 13. Diagrama de fuerzas empleado durante la implementación de *WFM.*

Existen sistemas de localización que basan su funcionamiento en las características cinemáticas del móvil empleado, utilizando una serie de ecuaciones matemáticas que permiten determinar la magnitud y dirección de un movimiento a partir de la información recopilada por sensores de tipo codificador encargados de establecer el desplazamiento generado en cada rueda del vehículo. Otro tipo de sistemas emplean estaciones de referencia que permiten establecer la posición del móvil mediante la interacción de dichos puestos de control y sensores ubicados en el vehículo, a partir de una serie de relaciones geométricas.

[4] J. Borenstein, Y. Koren, Real-Time Obstacle Avoidance for Fast Mobile Robots, Technical Report, *IEEE Transactions on Systems Man and Cybernetics. Vol 19* no. 25.

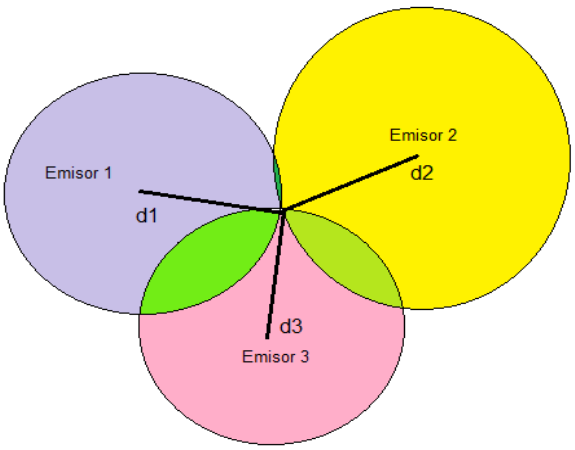


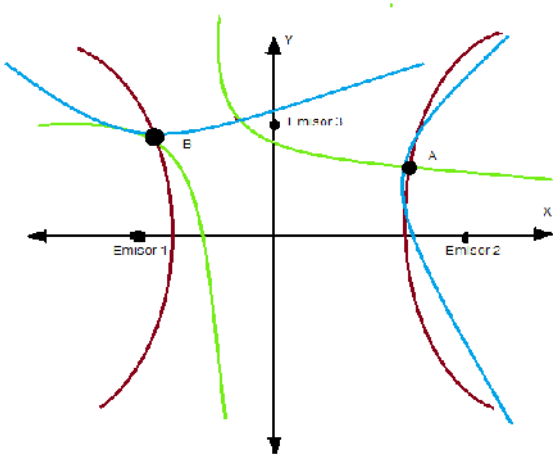
$$d \quad l/(\frac{1}{\tan\alpha}+\frac{1}{\tan\beta}$$

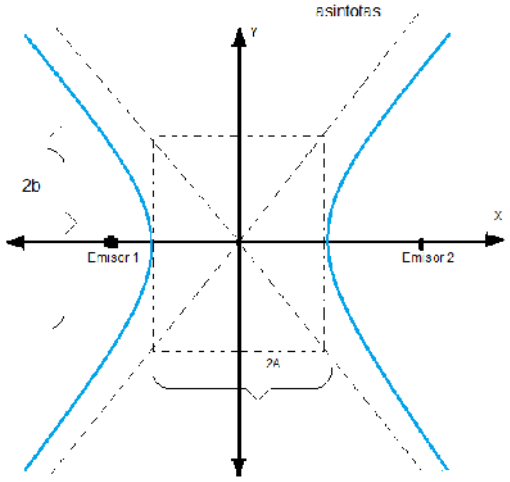
d

α

β




Y
Emisor 3
B
A
X
Emisor 1
Emisor 2



$$\frac{x-h}{a} \qquad \frac{y-h}{b}$$

el cálculo exacto del tiempo de vuelo para cada señal a partir del conocimiento del instante preciso en que se inicio la transmisión. La distancia existente entre el móvil y cada estación se determina mediante la multiplicación del tiempo de vuelo correspondiente y la velocidad de propagación de la onda. Si se asume que la onda se propaga omnidireccionalmente los puntos que se encuentran a la distancia calculada corresponden a una esfera centrada en el emisor para el caso de tres dimensiones, o una circunferencia en dos dimensiones. Para calcular el punto en donde se encuentra el móvil, basta con encontrar el punto en el que las distancias a los emisores concuerdan. Dependiendo del número de dimensiones la estimación del punto de ubicación se lleva a cabo de la siguiente manera[5].

4.6.5.1 TOA Dos Dimensiones:

Considerando a uno de los emisores en las coordenadas (0,0), el segundo emisor en las coordenadas (a, 0) y el tercer emisor en las coordenadas (c, b), entonces, la posición del móvil en el espacio viene dada por:

$$x = \frac{d1^2 - d2^2 + a^2}{2a} \quad (27) \qquad y = \frac{c^2 + b^2 + d1^2 - d3^2}{2b} - \frac{cx}{b} \quad (28)$$

Donde d1, d2 y d3 son las distancias a los emisores 1, 2 y 3 respectivamente.

$$x = \frac{d1^2 - d2^2 + a^2}{2a} \quad (29) \qquad y = \sqrt{d1^2 - x^2} \quad (30)$$

4.6.5.2 TOA Tres Dimensiones:

Si se considera que todos los emisores están a la misma altura, y el primer emisor está en las coordenadas (0, 0, 0), el segundo emisor está en las coordenadas (a, 0, 0) y el tercer emisor está en las coordenadas (c, b, 0), entonces, la posición del móvil es:

$$x = \frac{d1^2 - d2^2 + a^2}{2a} \quad (31)$$

$$y = \frac{c^2 + b^2 + d1^2 - d3^2}{2b} - \frac{cx}{b} \quad (32)$$

[5] Santiago Elvira Díaz, Redes de sensores inalámbricas aplicadas a robótica colaborativa , Universidad Autónoma de Madrid Escuela Politécnica Superior, Septiembre 2009.

$$z = \sqrt{d1^2 - x^2 - y^2} \qquad (33)$$

4.6.6 Sistemas de localización basados en el modelo cinemático del móvil.

El proceso más básico de Navegación de un robot móvil se basa en el modelo cinemático del sistema de propulsión que le permite al robot moverse dentro de un entorno. Entre los sistemas de locomoción con ruedas o cintas de deslizamiento pueden nombrarse el diferencial, síncrono, triciclo, Ackerman, omnidireccionales etc. El más común de entre estos es el denominado de tracción diferencial, debido a su bajo grado de complejidad y alta capacidad de adecuación para la navegación en ambientes de desarrollo típicos de actividades humanas.

Las configuraciones de tracción diferencial, son muy populares y permiten calcular la posición del robot a partir de las ecuaciones geométricas, que surgen de la relación entre los componentes del sistema de propulsión y la información de los codificadores rotativos que usualmente llevan acoplados a sus ruedas, mediante un procedimiento que se conoce como estimación odométrica de la posición. A pesar de que su implementación presenta una serie de ventajas frente a otros métodos de localización, el gran inconveniente asociado a este método está relacionado con el error incremental producido durante cada medida puesto que por pequeña que sea la constante de error en el sensor, el error de localización del robot crecerá sin límites debido al incremento progresivo del mismo a través del tiempo. La acumulación de este tipo de error puede ser mitigada corrigiendo la posición del robot en intervalos regulares, empleando para ello sistemas de apoyo que permitan realimentar el sistema de posicionamiento con datos provenientes de varios tipos de sensores. Sin embargo, estos sistemas tienen altos costos de instalación y de mantenimiento, por esta razón, los sistemas de localización odométrica siguen siendo de interés, puesto que además de producir resultados aceptables para la localización en trayectos cortos, también ayudan a reducir los costos de otros sistemas de localización ya que hacen que la actualización de la posición absoluta del robot sea menos frecuente. Además, los resultados de la navegación basada en la odometría, pueden ser fusionados con mediciones de posición absoluta para producir así mejores resultados en la localización global del vehículo.

4.7 Estimación odométrica en una plataforma diferencial.

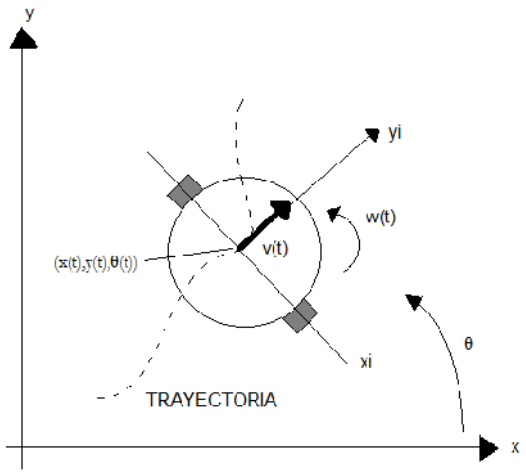
Una plataforma móvil de tracción diferencial cuenta generalmente con dos ruedas de tracción que tienen acoplados dos motores DC, y dos ruedas de estabilización que mantienen el balance del vehículo. La traslación y rotación de este tipo de plataformas diferenciales se determinan por el movimiento independiente de cada

$$\dot{x} \quad v\ t\ \cos\theta\ t$$

$$\dot{y} \quad v\ t\ \operatorname{sen}\theta\ t$$

$$\theta\ t \quad w\ t$$

$\dot{x}$
$\dot{y}$
$v\ t$
$\theta\ t$
$w\ t$
$\theta\ t$



$$x\ t \quad x\ t_0 \quad {}_{,t} \quad (\theta(t))\ dt$$

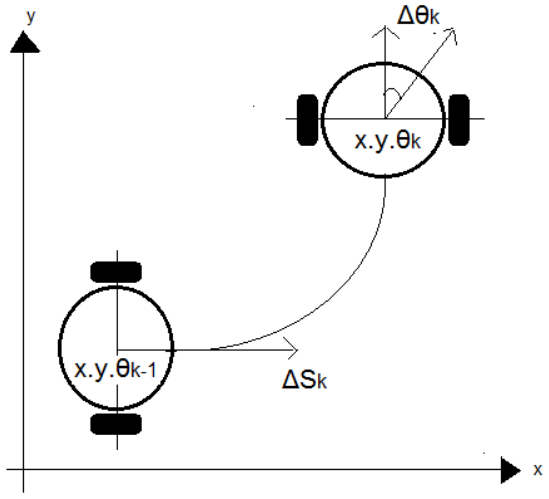
$$y\ t \quad y\ t_0 \quad {}_{,t} \quad (\theta(t))\ dt$$

$$\theta\ t \quad \theta\ t_0 \quad {}_{,t}$$

$$x_k \quad x_{k-1} \quad .x_k$$

$$y_k \quad y_{k-1} \quad .y_k$$

$$\theta_k \quad \theta_{k-1} \quad .\theta_k$$



$$\Delta S_k = \frac{\Delta S_R + \Delta S_L}{2} \qquad (43)$$

$$\Delta\theta_k = \frac{\Delta S_R - \Delta S_L}{b} \qquad (44)$$

ΔS_L y ΔS_R corresponde a los desplazamientos efectuados por la rueda izquierda y la rueda derecha respectivamente en tanto que la distancia *d* corresponde a la separación existente entre las dos ruedas. A partir de la relación cinemática establecida entre cada encoder y las ruedas del vehículo, el desplazamiento producido por cada rueda (ΔS) está determinado por:

$$\Delta S = \frac{2\pi r}{n}\Delta N \qquad (45)$$

En donde ΔN corresponde a la cantidad de pulsos generados por el encoder en el intervalo de muestreo, *n* está determinada por la cantidad de pulsos que se generan durante una rotación completa de la rueda y *r* corresponde a su radio. Los desplazamientos Δx, Δy y $\Delta\theta$ efectuados durante el intervalo establecido pueden ser calculados de la siguiente manera:

$$\Delta X_k = \Delta S_k \cos(\theta_{k-1} + \Delta\theta_k) \qquad (46)$$

$$\Delta Y_k = \Delta S_k \operatorname{sen}(\theta_{k-1} + \Delta\theta_k) \qquad (47)$$

De esta manera es posible determinar la posición y orientación actuales de un móvil conociendo una configuración anterior para estas mismas magnitudes.

5. REVISION ACERCA DE LOS ALGORITMOS GENETICOS

Durante los últimos 50 años, el desarrollo conjunto de la ingeniería biomédica y la bio-ingeniería ha rendido significativos frutos en diferentes áreas. Dichos avances se centran principalmente en la solución de problemas dentro de cualquier campo, aplicando los conocimientos adquiridos sobre el comportamiento de diferentes sistemas de la naturaleza, esto teniendo en cuenta que los seres vivos han demostrado ser realmente eficientes solucionando problemas, de modo tal que la evolución y la selección natural se constituyen como una prueba fehaciente en este sentido. Dentro del reino animal, el comportamiento social de diversos grupos de individuos ha sido analizado con el fin de encontrar patrones de trabajo que permitan formalizar técnicas de optimización de procesos que tiene como fundamento el aprovechamiento de recursos. De esta forma, hoy se conocen algoritmos de optimización de este tipo basados en el comportamiento de abejas, hormigas, aves, entre otros. Estos algoritmos están dentro de lo que se conoce como algoritmos bioinspirados, los cuales se basan en el empleo de analogías con algunos sistemas naturales o sociales para la resolución de una variedad de problemas; estos tienen como fin el diseño de métodos heurísticos no determinísticos de “búsqueda”, ”aprendizaje” y ”comportamiento”. La heurística se define como la capacidad que posen algunos sistemas para la realización de forma inmediata innovaciones positivas para sus fines. Dentro de las capacidades que a menudo suelen poseer los algoritmos bioinspirados, se presenta implícitamente, una estructura paralela o de múltiples agentes, además de ser adaptativos, utilizando la realimentación con el entorno para modificar el modelo y los parámetros. Pueden nombrarse como ejemplos de estos Algoritmos Evolutivos, inspirados en los principios Darwinianos de la evolución natural; la Inteligencia de Enjambres, que tiene como fundamento la inteligencia colectiva emergente de un grupo de agentes simples, algoritmos o mecanismos distribuidos de resolución de problemas inspirados en el comportamiento colectivo de colonias de insectos sociales u otras sociedades de animales y por último las Colonias de Hormigas, basados en la simulación del comportamiento de las colonias de hormigas cuando recogen comida.

Un segundo enfoque centra su atención en el estudio del funcionamiento del sistema orgánico. Así surgen las redes neuronales que están basadas en la simulación del comportamiento del sistema nervioso del cerebro humano y los algoritmos genéticos.

5.1 Algoritmos Genéticos.

Se definen como teorías de búsqueda, que tienen su origen en la teoría de la evolución y se basan en los mecanismos por los cuales la naturaleza procede para la selección, así es que los organismos que presentan una mayor adaptación al medio son los que sobreviven, adecuándose mucho más fácilmente a los cambios que se producen en su entorno; en la actualidad se conoce que estos cambios se producen a nivel celular específicamente en los genes, que son la unidad básica de codificación de cada uno de los atributos que presenta un ser vivo o individuo, y que los atributos mas deseables, siendo aquellos que le permitan a cada individuo adaptase de mejor manera a su entorno, se transmiten a sus descendientes cuando este se reproduce.

Uno de los principales aportes acerca del potencial consignado en los algoritmos genéticos se debe a John Holland, investigador de la universidad de Michigan que enfocado en la importancia de la selección natural desarrollo hacia finales de los años 60 una técnica que se logro incorporar en un programa de computadora, el objetivo de este programa era conseguir que la máquina lograse aprender por ella sola, a esta técnica desarrollada por Holland se la conoció en un principio como "planes reproductivos", pero se hizo mucho más popular con el nombre de "algoritmos genéticos" tras la publicación de su libro en el año de 1975.

Una definición bastante acertada de lo que es un algoritmo genético es la presentada a continuación por Jhon Kosa: "*Es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud"* [6].

5.2 Funcionamiento de un Algoritmo Genético Simple.

A continuación se presenta la operación de un algoritmo genético simple (abstracto), con el siguiente segmento de pseudo-código:

---

[6] KOZA John R., "Genetic Programing", *On the programming of computers by means of natural selection,* Massachusetts: Pac-Man, 1992, p.18.

INICIO;
Generar población inicial;
Evaluar la población inicial;
MIENTRAS CRITERIO DE PARADA == FALSO **HACER**

/*PRODUCIR NUEVA GENERACION*/

PARA (Tamaño de población/2) **HACER**

Seleccionar 2 individuos para ser cruzados (empleando la probabilidad de cruce);
Cruzar los individuos seleccionados para obtener los hijos;
Mutar los individuos obtenidos a partir de cierto criterio de probabilidad;
Evaluar el nivel de adaptación de los individuos mutados o no empleando la función establecida para tal fin;
Insertar los nuevos individuos en la población que será empleada para la siguiente iteración;

END

SI El criterio de convergencia se cumple **ENTONCES**
CRITERIO DE PARADA == VERDADERO;
END
END

Para la operación exitosa del algoritmo, se hace necesario contar con un sistema de codificación que permita representar el problema a tratar de manera adecuada. La presencia de una función de adaptación se hace indispensable puesto que cada población generada debe ser evaluada con el fin de establecer el grado de adaptación de los individuos ante la situación planteada, para posteriormente seleccionar los individuos más aptos que constituirán el conjunto de padres para la siguiente generación. Los padres seleccionados deben cruzarme a fin de originar una nueva población de individuos que están expuestos a sufrir mutaciones esporádicas. Como resultado, estos individuos constituyen un conjunto de posibles soluciones al problema planteado y han de formar parte de la siguiente población puesto que el proceso se repite hasta que algún criterio de parada o convergencia establecido se cumpla.

A continuación se establecen algunas características principales que cumple cada uno de los componentes del algoritmo en general.

5.3 Codificación.

Dado que los individuos que conforman el conjunto de posibles soluciones pertenecientes a un problema en específico, deben estar representados mediante un conjunto de parámetros conocidos como *genes*, cuya agrupación determina la estructura básica del algoritmo genético conocida como *cromosoma* resulta de suma importancia definir un código de representación que permita transmitir las características primordiales de cada individuo de forma adecuada.

Generalmente la codificación de las posibles soluciones se lleva a cabo empleando el alfabeto binario, aunque existen otros sistemas de codificación que han sido puestos en marcha de forma exitosa. Si el sistema de números binarios es empleado, cada posible solución conocida como genotipo está compuesta por una serie de parámetros identificados mediante una etiqueta numérica que en conjunto constituyen el cromosoma como tal. Así, si la solución al problema en estudio está determinada por la búsqueda de un grupo específico de valores correspondientes a las variables bajo estudio, la codificación de un individuo corresponde a una ristra de valores binarios que identifican a cada uno de los parámetros buscados de una manera establecida previamente.

5.4 Población.

5.4.1 Tamaño de la población.

Parece intuitivo que las poblaciones pequeñas corren el riesgo de no cubrir adecuadamente el espacio de búsqueda, mientras que las poblaciones de gran tamaño pueden acarrear problemas relacionados con el excesivo costo computacional. Goldberg efectuó un estudio teórico, que tuvo como conclusión que la aplicabilidad de los Algoritmos Genéticos en problemas reales sería muy limitada, ya que resultan no competitivos con otros métodos de optimización combinatoria. Alander, sugiere que un tamaño de población comprendida entre l y 21 es suficiente para atacar con éxito los problemas por él considerados.

5.4.2 Población inicial.

Habitualmente la población inicial se escoge generando ristras al azar, pudiendo contener cada gen uno de los posibles valores del alfabeto con probabilidad uniforme. En los pocos trabajos que existen sobre este aspecto, se constata que esta inicialización no aleatoria de la población inicial, puede acelerar la convergencia del Algoritmo Genético. Sin embargo en algunos casos la desventaja resulta ser la prematura convergencia del algoritmo, queriendo indicar con esto la convergencia hacia óptimos locales.

5.5 Función objetivo.

Dos aspectos que resultan cruciales en el comportamiento de los Algoritmos Genéticos son la determinación de una adecuada función de adaptación o función objetivo, así como la codificación utilizada. Idealmente interesa construir funciones objetivo con "ciertas regularidades", es decir funciones objetivo que verifiquen que para dos individuos que se encuentren cercanos en el espacio de búsqueda, sus respectivos valores en las funciones objetivo sean similares.

La regla, general para construir una buena función objetivo es que ésta debe reflejar el valor del individuo de una manera "real", pero en muchos problemas de optimización combinatoria, donde existe gran cantidad de restricciones, buena parte de los puntos del espacio de búsqueda representan individuos no válidos.

Para este planteamiento en el que los individuos están sometidos a restricciones, se han propuesto varias soluciones. La primera denominada absolutista, en la que aquellos individuos que no verifican las restricciones, no son considerados como tales, y se siguen efectuando cruces y mutaciones hasta obtener individuos válidos, o bien, a dichos individuos se les asigna una función objetivo igual a cero. La segunda consiste en reconstruir aquellos individuos que no verifican las restricciones. Dicha reconstrucción suele llevarse a cabo por medio de un nuevo operador que se acostumbra a denominar reparador.

Un problema habitual en la ejecución de los Algoritmos Genéticos surge debido a la velocidad con la que el algoritmo converge. En algunos casos la convergencia es muy rápida, lo que suele denominarse convergencia prematura, en la cual el algoritmo converge hacia óptimos locales, mientras que en otros casos el problema es justo el contrario, es decir se produce una convergencia lenta del algoritmo. Una posible solución a estos problemas se basa en transformaciones de la función objetivo. El problema de la convergencia prematura, surge a menudo cuando la selección de individuos se realiza de manera proporcional a su función objetivo. Así, pueden existir individuos con una adaptación al problema muy superior al resto, que a medida que avanza el algoritmo "dominan" a la población. Por medio de una transformación de la función objetivo, en este caso una comprensión del rango de variación de la función objetivo, se pretende que dichos "superindividuos" no lleguen a dominar a la población. El problema de la lenta convergencia del algoritmo, se resuelve de manera análoga, pero en este caso efectuando una expansión del rango de la función objetivo.

5.6 Selección.

La función de selección más utilizada es la denominada función de selección proporcional a la función objetivo, en esta cada individuo tiene una probabilidad de ser seleccionado como padre, que es proporcional al valor de su función objetivo.

Denotando por $P_{j,t}^{prop}$ la probabilidad de que el individuo I_t^j sea seleccionado como padre, se tiene que:

$$P_{j,t}^{prop} = \frac{g\,(I_t^j)}{\sum_{j=1}^{\lambda} g(I_t^j)} \quad (48)$$

Una manera de superar el problema de la rápida convergencia proveniente de los superindividuos, es efectuar la selección proporcional al rango del individuo, con lo cual se produce una repartición más uniforme de la probabilidad de selección.

5.7 Cruce.

El cruce basado en un punto es en el cual los dos individuos seleccionados para jugar el papel de padres, son recombinados por medio de la selección de un punto de corte, para posteriormente intercambiar las secciones que se encuentran a la derecha de dicho punto. Se han investigado otros operadores de cruce, habitualmente teniendo en cuenta más de un punto de cruce, por ejemplo el basado en múltiples puntos, pero se tiene que el cruce basado en dos puntos, representaba una mejora mientras que añadir más puntos de cruce no beneficiaba el comportamiento del algoritmo. La ventaja de tener más de un punto de cruce radica en que el espacio de búsqueda puede ser explorado más fácilmente, siendo la principal desventaja el hecho de aumentar la probabilidad de ruptura de buenos esquemas. En el operador de cruce basado en dos puntos, los cromosomas (individuos) pueden contemplarse como un circuito en el cual se efectúa la selección aleatoria de dos puntos.

En el denominado operador de cruce uniforme cada gen, en la descendencia se crea copiando el correspondiente gen de uno de los dos padres, escogido de acuerdo a una "máscara de cruce" generada aleatoriamente. Cuando existe un 1 en la "máscara de cruce", el gen es copiado del primer padre, mientras que cuando exista un 0 en la "máscara de cruce", el gen se copia del segundo padre. En la literatura el término operador de cruce uniforme se relaciona con la obtención de la "máscara de cruce" uniforme, en el sentido de que cualquiera de los elementos del alfabeto tenga asociada la misma probabilidad.

5.8 Mutación.

La mutación se considera un operador básico, que proporciona un pequeño elemento de aleatoriedad en la vecindad de los individuos de la población. Si bien se admite que el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, también parece desprenderse de los experimentos efectuados por varios investigadores que el operador de mutación va ganando en importancia a medida que la población de individuos va convergiendo. Schaffer y Col. encuentran que el efecto del cruce en la búsqueda es inferior al que previamente se esperaba. Utilizan la denominada evolución primitiva, en la cual, el proceso evolutivo consta tan sólo de selección y mutación. Encuentran que dicha evolución primitiva supera con creces a una evolución basada exclusivamente en la selección y el cruce.

5.9 Reducción.

Una vez que se obtiene los individuos descendientes de una determinada población, el proceso de reducción al tamaño original, consiste en escoger algunos individuos de entre los que forman parte de la población, y los individuos descendientes de los mismos. Hay dos formas fundamentales; la que se denomina reducción simple, que hace a los individuos descendientes, los que forman parte de la población, o se escogen los individuos más adaptados al problema, que se denomina criterio de reducción elitista. El concepto de reducción está ligado con el de tasa de sustitución generacional, que es el porcentaje de hijos generados con respecto del tamaño de la población.

6. PROTOTIPO DE VEHICULO AUTONOMO

Una vez establecidas las características primordiales de los controladores básicos necesarios para la implementación de un sistema de navegación para un vehículo autónomo, en el presente apartado se describe cada uno de los módulos que componen el prototipo desarrollado durante esta investigación, así como también su estructura física y las condiciones operativas de los dispositivos sensoriales y de transmisión empleados.

6.1 Descripción General.

El prototipo construido está conformado por una serie de módulos creados con un propósito específico, con el fin de reducir los tiempos de operación del sistema realizando operaciones de forma simultánea. La figura 20 muestra la configuración modular del móvil. El sistema de tracción del móvil está constituido por un conjunto de dos motores de corriente continua provistos cada uno, de una caja de reducción y un codificador incremental de efecto hall implementados en configuración diferencial.

6.2 Estructura física.

El vehículo elaborado está constituido por un chasis fabricado en *MDF* (*Medium Density Fibreboard*), material que se ha escogido dada la proporción existente entre su alta resistencia al esfuerzo y su bajo peso. Las dimensiones del chasis corresponden a un vehículo de tamaño intermedio, en busca de construir futuros prototipos que puedan ser operados en ambientes exteriores con condiciones topográficas adversas. En la figura 21 se muestra una imagen real del vehículo.

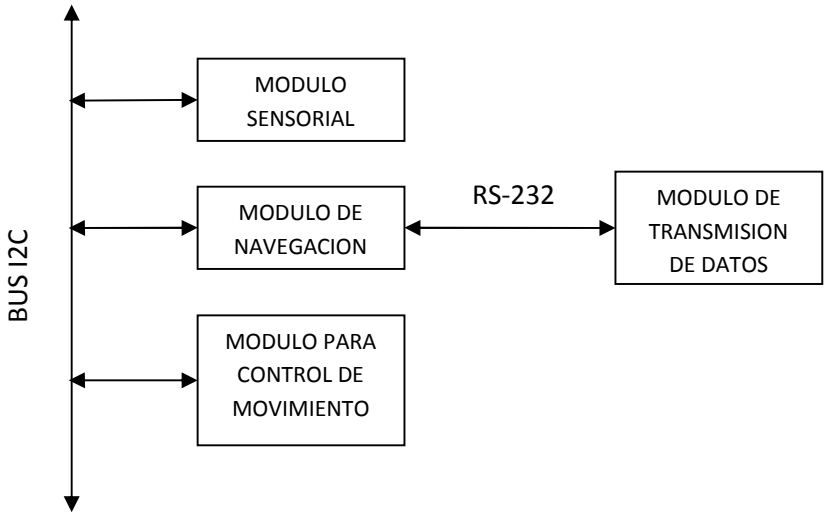
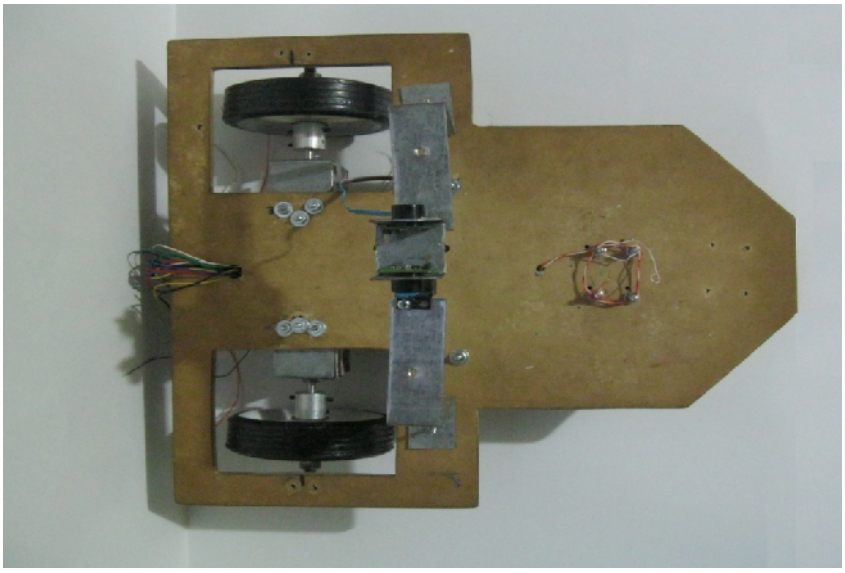
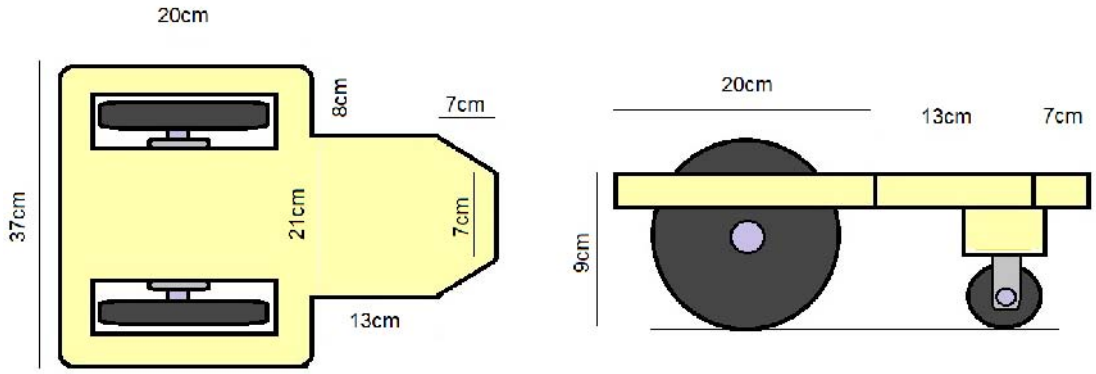
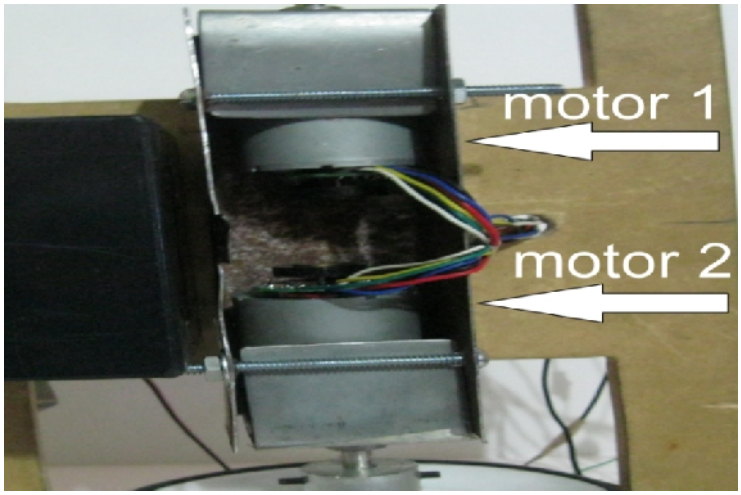


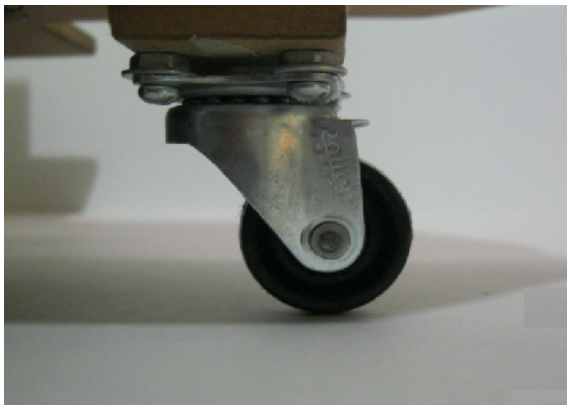
Figura 20. Configuración modular del móvil.

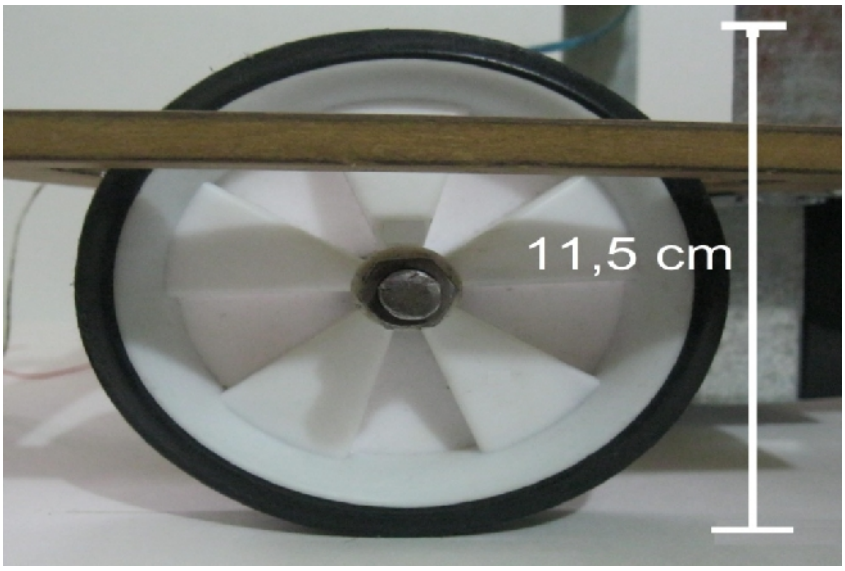


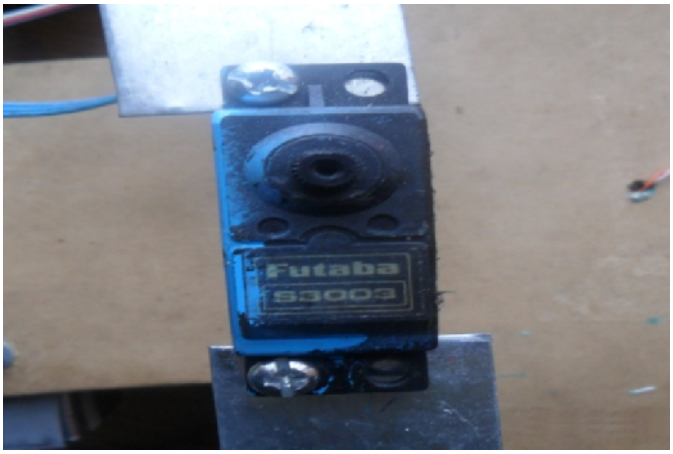
Dimensiones del Vehículo



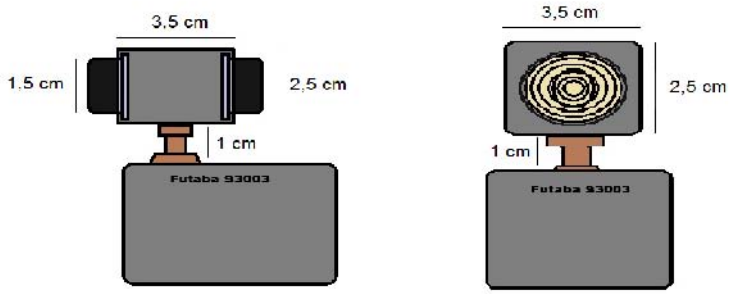

motor 1
motor 2

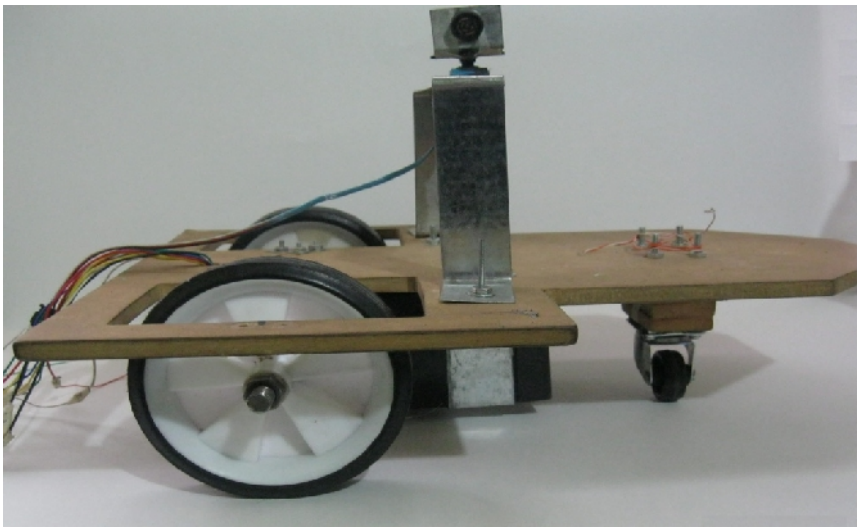


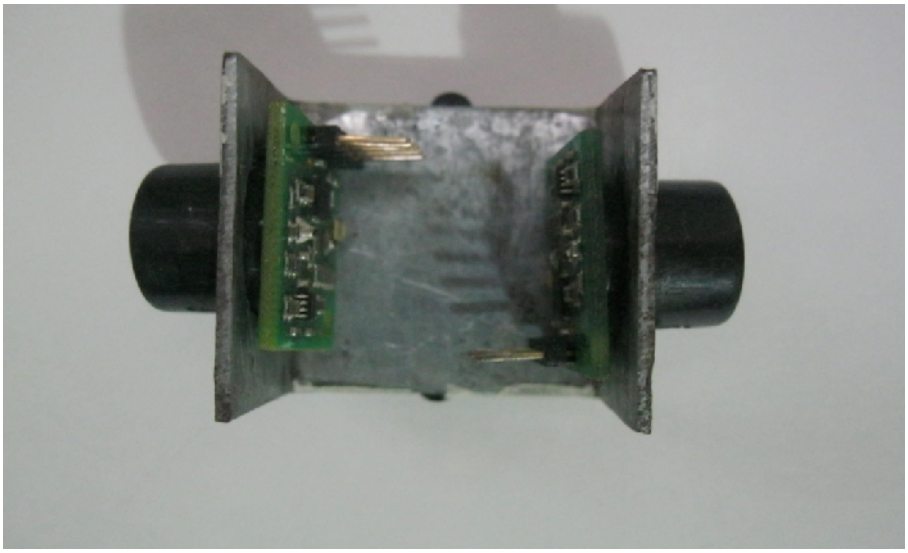

11,5 cm


Futaba
S3003

Sonar







próximo empleando una serie de ecuaciones matemáticas relacionadas con la velocidad de propagación del sonido en el aire.

La brújula magnética empleada para la determinación conjunta de la orientación del móvil en un instante de tiempo determinado corresponde a la referencia CMPS03. Este dispositivo está constituido por un par de sensores de campo magnético KMZ51 desarrollados por PHILIPS capaces de medir con precisión la desviación del campo magnético terrestre, dispuestos de forma tal que entre ellos existe un ángulo de noventa grados, a fin de poder estimar la dirección de la componente horizontal de dicho campo magnético. Este dispositivo opera a un voltaje nominal de 5V y requiere una fuente de alimentación capaz de suministrar una corriente del orden de los 15mA. La comunicación puede ser establecida empleando un tren de pulsos PWM (*Pulse Width Modulated)* o la interfaz i2c de la que dispone. Si se emplea una señal de tipo PWM, el ancho correspondiente a la señal puesta en alto representa el ángulo de orientación medido en grados, de forma tal que dicha magnitud varía entre 1ms para 0º y 36.99ms que representan una orientación de 359.9º. El tiempo que la señal permanece en nivel bajo entre cada pulso corresponde a 65ms de forma tal que su periodo está comprendido entre 66ms y 102ms.

Por su parte, la interfaz i2c está diseñada para operar empleando una frecuencia en el bus de datos comprendida entre 100KHz y 1MHz, considerando que para frecuencias mayores a 160KHz, un retraso de 50us debe ser incluido entre cada medición con el fin de asegurar que los registros internos del dispositivo puedan ser escritos de manera adecuada. El protocolo i2c empleado para comunicarse con este dispositivo está constituido de forma tal que inicialmente se envía un bit de inicio de transmisión seguido por la dirección del módulo en el bus de datos (0xC0 para este caso en particular). Posteriormente se envía un nuevo bit de inicio seguido por la dirección del módulo con el bit de lectura/escritura puesto en alto (0xC1); el módulo se encuentra entonces habilitado para enviar la información consignada en sus registros de tal manera que es posible leer uno o dos bytes según la información requerida. En la tabla 1 se muestra la función de cada registro contenido en la brújula.

6.4 Módulo de Navegación.

Este módulo se ocupa de llevar a cabo todas las tareas necesarias para determinar la orientación que debe asumir el móvil en un instante de tiempo determinado con el fin de desplazarse hacía un punto objetivo en el espacio de trabajo. De igual manera se encarga de estimar la posición del vehículo en el

espacio en un instante de tiempo y la velocidad con la que debe realizarse cada desplazamiento. La generación de la señal PWM encargada de controlar la posición del servo motor empleado en el módulo sensorial es otra de las actividades realizadas por este componente del sistema, así como también la obtención de los datos correspondientes a la reconstrucción del entorno, entregados por los sensores ultrasónicos empleados en el mismo módulo. A su vez, se encarga de gestionar la transmisión de los datos referentes a la posición y orientación del vehículo hacia el PC.

REGRISTRO	FUNCION
0	Numero de revisión del software
1	Medición del Compás de un byte, de 0-255 para un círculo completo
2,3	Medición del Compás de una palabra, de 0-3599 de un círculo completo, lo que representa 0-359.9 grados.
4,5	Sensor 1, señal en diferencia - 16 bits con signo
6,7	Sensor 2, señal en diferencia - 16 bits con signo
8,9	Valor de calibración 1 a 16 bits con signo
10,11	Valor de calibración 2 a 16 bits con signo
12	No utilizado
13	No utilizado - Lee como cero
14	No utilizado - Lee como indefinido
15	Comando de calibración

Tabla 1. Registros internos del compás electrónico CMPS03.

Físicamente, el módulo está constituido por un micro-controlador que oficia como cerebro del sistema general puesto que recopila la información proveniente del sistema sensorial y la traduce en instrucciones que determinan el comportamiento del modulo de control de movimiento. A su vez, oficia como maestro del bus de comunicaciones I2C que lleva a cabo la transferencia de datos entre los módulos del prototipo. El micro-controlador empleado corresponde a la referencia DSPic30f4011desarrollado por MICROCHIP. Este dispositivo cuenta con un núcleo capaz de interpretar instrucciones de 24 bits que constituyen un conjunto de 84 instrucciones básicas decodificables por el dispositivo. Los datos a procesar poseen una estructura de 16 bits de longitud que pueden ser almacenados en un espacio de memoria de 48Kbytes correspondientes a memoria no volátil de tipo flash, 2Kbytes de memoria tipo RAM o en su defecto 1Kbyte de memoria tipo EPROM. Según la configuración de la CPU empleada, el dispositivo puede operar a una frecuencia máxima de 30MIPs gracias a la presencia de un multiplicador tipo PLL de 4x, 8x y 16x que permite incrementar la frecuencia de un oscilador externo en un valor correspondiente al multiplicador especificado.

El sistema de periféricos presentes en el dispositivo está constituido por un módulo SPI de tres líneas, un módulo I2C capaz de soportar configuraciones Multi-Master/Slave con un protocolo de direccionamiento de siete o diez bits, dos módulos UART y un módulo CAN. Dado que los micro-controladores identificados con la referencia DSpic30f hacen parte de la familia de dispositivos destinados al control de motores, cuentan con un módulo dedicado a la generación de señales PWM con ciertas características específicas necesarias para el control de motores tipo DC y AC. El módulo cuenta con seis canales de salida capaces de operar en modo complementario o independiente. Cada par de canales, cuenta con un generador de tiempo de trabajo independiente y un control de tiempo de espera de señal en bajo útil durante la operación del sistema en modo complementario. De igual manera, el control de motores se complementa con la presencia de un módulo de codificación en cuadratura constituido por una señal de fase A, una señal de fase B y una señal de inicio de vuelta que permiten en conjunto establecer la posición del motor en un instante de tiempo determinado empleando un contador de 16bits.

Para su correcta operación, este dispositivo requiere de un voltaje de alimentación comprendido entre los 2.5V y los 5.5V. Debe considerarse que una frecuencia de operación equivalente a 30MIPS requiere de un voltaje de alimentación correspondiente al rango de 4.5V a 5.5V. Cada pin de salida puede entregar una corriente máxima de 25mA, y la corriente total que puede ser entregada por el dispositivo en un instante de tiempo determinado no debe superar los 200mA. Finalmente, la temperatura de operación está comprendida entre el rango de los -40ºC a 85ºC.

6.5 Módulo para control de movimiento.

Está constituido por dos micro-controladores DSpic30f4011 configurados como esclavos en el bus de comunicaciones I2C, encargados de transformar la información entregada por el módulo de navegación en señales de control de tipo PWM con la finalidad de gestionar el movimiento de dos motores DC provistos de una caja de reducción independiente. De igual manera, cada micro-controlador recopila la información proveniente de cada codificador de posición ubicado tanto en la rueda derecha como en la izquierda con el fin de entregar estos datos al módulo de navegación permitiendo estimar la posición del móvil en el espacio de trabajo. Los componentes electrónicos necesarios para realizar el control de los motores también se incluyen dentro de este sub sistema de forma tal que se hace uso de un driver de referencia L298N con el fin de proporcionar la corriente necesaria a cada motor directamente desde la batería, impidiendo sobre cargas en


XBee
PRO

de información está constituido por 1 bit de inicio, 8 bits de información, con el bit menos significativo encabezando la trama y 1 bit de parada. La configuración de los parámetros de transmisión (baud rate, paridad, bit de inicio, bit de parada y número de bits de información) depende de los requerimientos incluidos en el micro-controlador adjunto de forma tal que es posible modificar estos valores en el módulo XBee para que exista compatibilidad. En su configuración de fábrica, cada módulo se encuentra programado para operar en modo transparente, de manera tal que se comporta como una línea de transmisión serial en la cual cada dato que ingresa a través del módulo UART es enviado vía radio-frecuencia, y cada dato que ingresa en el receptor de radio es enviado a través del módulo UART hacia el micro-controlador.

La recepción de los datos en el PC se lleva a cabo empleando un dispositivo XBee similar al presente en el vehículo, que integrado a un micro-controlador de referencia PIC18f2550, permite la presentación de la información requerida en el PC mediante una conexión USB.

6.7 Resumen de dispositivos empleados.

Una vez descritos cada uno de los módulos implementados durante la investigación resulta importante realizar un resumen de los dispositivos empleados en cada módulo, incluyendo una pequeña descripción de la función primordial que desempeñan y el módulo al que pertenece. La información pertinente a estos aspectos puede apreciarse en la tabla 2.

DISPOSITIVO	DESCRIPCION	MODULO EN EL QUE SE EMPLEA	FUNCION	TIPO DE COMUNICACIÓN
DSpic 30f4011.	Micro-controlador desarrollado por Microchip, configurado para trabajar desarrollando 20Mips. Dispone de un módulo I2C, 2 módulos UART, un módulo para codificador de posición, 6 salidas tipo pwm, etc.	Módulos de navegación y para el control de movimiento.	Módulo de navegación: Determinación de velocidad y orientación del movimiento, recopilación de datos provenientes de la red sensorial, control del sonar y transmisión de los datos al PC. Módulo para el control de movimiento: Control del sentido y la velocidad de giro del par de motores empleados para el desplazamiento.	Bus I2C (control de sensores de distancia y comunicación maestro-esclavos). Protocolo serial (Comunicación con el dispositivo XBee para transmisión de datos al PC).
Brújula Electrónica CMPS03.	Sensor de campos magnéticos basado en el cálculo de la dirección de la componente horizontal del campo magnético terrestre para definir la orientación actual.	Módulo sensorial.	Determinación de la orientación actual del móvil a partir del valor del campo magnético terrestre.	Bus I2C.
Sensor SRF02.	Sensor de distancias por ultrasonidos que emplea un único transductor para emitir y recibir la señal. Su rango de medidas se ubica entre 15 y 600cm.	Módulo sensorial.	Determinación de la distancia existente hasta el obstáculo más próximo en una dirección establecida.	Bus I2C.
Módulo XBee de referencia XB24-AWI-001.	Transceptor que opera a una frecuencia de 2.4GHz para realizar transmisión de datos empleando protocolo serial.	Módulo de transmisión de datos.	Enviar y recibir cadenas de datos hacia y desde el PC empleando protocolo serial y comunicación vía RF.	Protocolo serial: (comunicación con el micro-controlador maestro). Comunicación RF: (Transmisión bidireccional de datos con el PC).
Motor reductor DC.	Motor DC con un sistema de reducción de 1:131 asociado que opera a 12V y permite entregar 18kg-cm de torque a un máximo de 80RPM.	Módulo para el control de movimiento.	Orientar al vehículo en la dirección establecida y posteriormente efectuar el desplazamiento del mismo a una determinada velocidad.	Señal PWM para control de giro y velocidad.
Codificador de efecto Hall en cuadratura.	Codificador de cuadratura acoplado al eje del motor reductor DC que emplea el efecto Hall para entregar un total de 8384 pulsos por vuelta del eje final (después de la reducción).	Módulo de Localización.	Establecer el número de pulsos que la rueda se ha desplazado durante cada intervalo de muestreo.	Señal en cuadratura hacia cada dispositivo esclavo.

Tabla 2. Resumen de los dispositivos empleados durante la implementación del sistema de navegación embebido.

7. IMPLEMENTACION DEL SISTEMA DE NAVEGACIÓN EN EL VEHÍCULO AUTÓNOMO

A continuación se presentan los procedimientos llevados a cabos durante la implementación del sistema de navegación en los dispositivos embebidos contenidos en el prototipo del vehículo autónomo descrito anteriormente.

7.1 Implementación del sistema de navegación embebido.

El sistema de navegación embebido implementado al interior del prototipo cumple con una serie de tareas en su mayoría coordinadas por el módulo de navegación descrito en el apartado anterior. Las tareas fundamentales llevadas a cabo por este sistema son la estimación de la posición del móvil en un instante de tiempo determinado, la reconstrucción del entorno de trabajo empleando la información proporcionada por la red sensorial, la determinación de la orientación a seguir así como el cálculo de la velocidad de desplazamiento requerida en un instante de tiempo y el establecimiento de las comunicaciones con el PC. El diagrama de flujo que representa el funcionamiento general del sistema implementado se muestra en la figura 31.

Las comunicaciones existentes entre los diferentes módulos y componentes del prototipo se llevan a cabo empleando un bus de comunicaciones I2C regido por el micro-controlador que constituye el módulo de navegación y que está configurado como maestro del bus. En el caso particular del módulo de transmisión de datos dichas comunicaciones se llevan a cabo empleando protocolo serial que rige la línea de transmisión establecida entre el módulo UART del micro-controlador y su similar presente en el dispositivo XBee.

7.2 Recepción de comandos enviados desde el PC.

El inicio de cada desplazamiento, la detención del proceso y el establecimiento de las coordenadas del punto objetivo, son directrices establecidas por el usuario desde el PC. Cada comando enviado está constituido por una cadena de 4 bytes con el fin de permitir ampliar el número de instrucciones que pueden ser enviadas hacia el vehículo. El módulo UART dispuesto en el micro-controlador maestro está configurado de manera tal que cada transmisión se realiza a una tasa de transferencia de 9600 baudios empleando 8 bits de datos, sin paridad, 1 bit de inicio y 2 bits de parada.

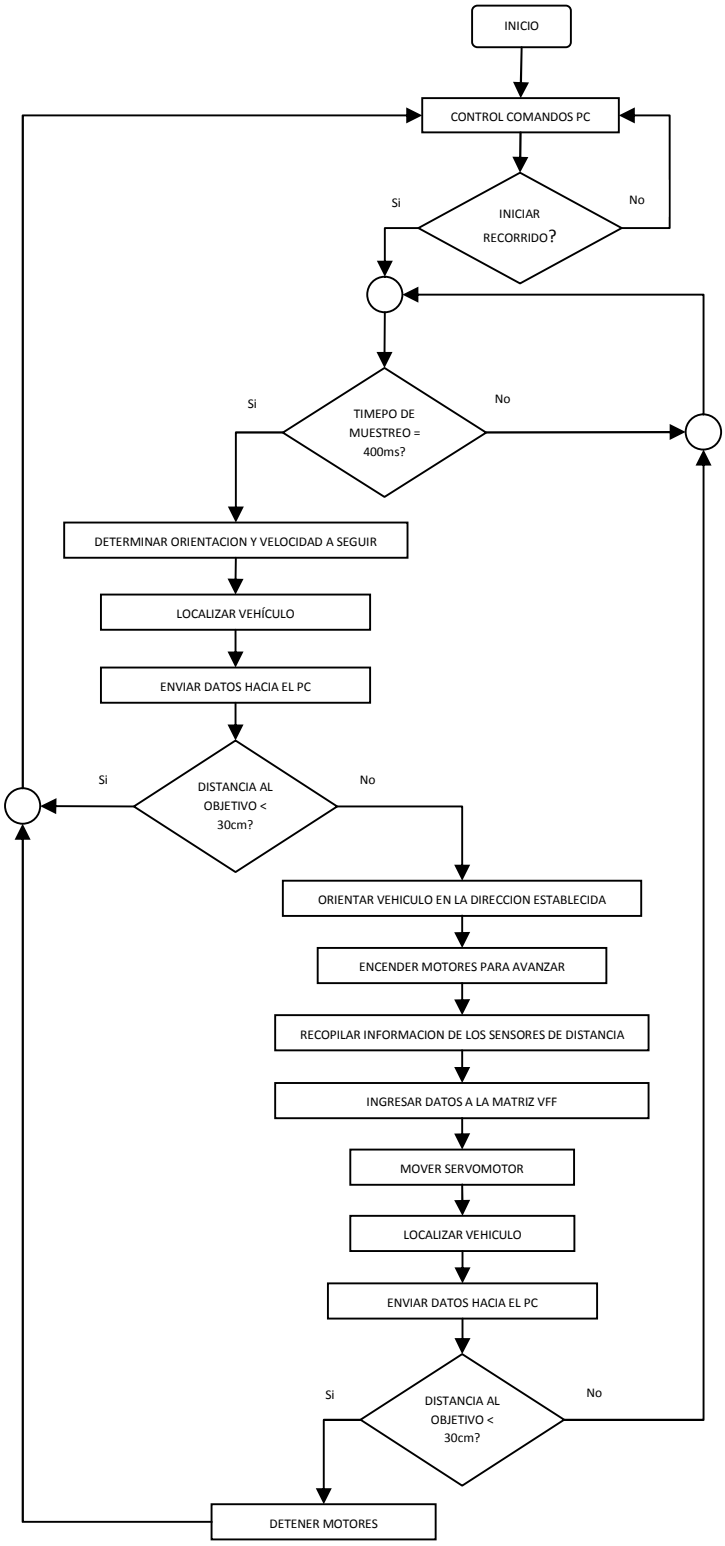


Figura 31. Diagrama de flujo correspondiente al sistema de navegación implementado.

Una interrupción es generada cada vez que se han recibido 4 bytes completos de manera tal que cada comando enviado por el PC es gestionado a través de una interrupción en la cual se determina el comportamiento a seguir. La serie de comandos que pueden ser enviados se presenta en la tabla 3.

REPRESENTACION COMANDO (hexadecimal)	**FUNCION DESARROLLADA (MICRO-CONTROLADOR)**
08,X_1X_2,X_3X_4,2ª	Recepción del valor correspondiente a las constantes de repulsión y atracción, siendo X_1X_2 el valor de la primera y X_3X_4 el valor de la segunda.
X_1X_2,X_3X_4,X_5X_6,2B	Recepción de los valores correspondientes a ω (X_1X_2), z (X_3X_4), τ (X_5X_6).
00,X_1X_2,X_3X_4,CD	Recepción del valor correspondiente a la coordenada x del punto objetivo cuando este mismo posee signo positivo con X_1X_2 representando al byte más significativo y X_3X_4 representando al byte menos significativo.
01,X_1X_2,X_3X_4,CD	Recepción del valor correspondiente a la coordenada x del punto objetivo cuando este mismo posee signo negativo con X_1X_2 representando al byte más significativo y X_3X_4 representando al byte menos significativo.
00,X_1X_2,X_3X_4,DE	Recepción del valor correspondiente a la coordenada y del punto objetivo cuando este mismo posee signo positivo con X_1X_2 representando al byte más significativo y X_3X_4 representando al byte menos significativo.
01,X_1X_2,X_3X_4,DE	Recepción del valor correspondiente a la coordenada x del punto objetivo cuando este mismo posee signo negativo con X_1X_2 representando al byte más significativo y X_3X_4 representando al byte menos significativo.
00,X_1X_2,00,EF	Recepción del valor correspondiente a la velocidad máxima de desplazamiento.
08,09,0A,0B	Recepción del comando correspondiente al inicio del desplazamiento.
08,09,0A,0C	Recepción del comando correspondiente a la detención del desplazamiento.

Tabla 3. Comandos enviados por el PC.

7.3 Determinación de la orientación y la velocidad para el desplazamiento en un instante de tiempo determinado.

Una vez que el comando correspondiente al inicio del desplazamiento ha sido recibido, los procedimientos llevados a cabo para determinar la orientación y velocidad que el móvil debe asumir para alcanzar el punto objetivo se activan hasta que la distancia existente hacia dicho punto alcance un valor establecido. El

proceso se repite cada vez que un temporizador equipara un valor determinado que para este caso particular corresponde a 400ms. El método de navegación empleado durante la implementación del sistema corresponde al método de campo de fuerzas virtuales en el cual el espacio de trabajo es modelado como una grilla compuesta por valores de certeza correspondientes a los datos obtenidos a partir de la red sensorial integrada al sistema.

El esquema original del método presenta una división del espacio de trabajo en celdas de tamaño fijo, que son recorridas por una ventana activa de tamaño inferior con el fin de monitorear la presencia de obstáculos en el entorno inmediato del vehículo. Sin embargo, las limitaciones derivadas del tamaño restringido de la memoria de acceso aleatorio (RAM) con la que cuenta el micro-controlador empleado (2Kbytes), hacen que sea poco viable realizar una segmentación del espacio total de trabajo puesto que únicamente sería posible llevar a cabo este proceso en ambientes con dimensiones menores a los $4m^2$. Como solución se propone entonces la reconstrucción del espacio inmediato del móvil empleando únicamente la ventana activa, en conjunto con un algoritmo de localización que permite desplazar los coeficientes consignados en las celdas de la misma dependiendo de la ubicación del vehículo dentro del espacio de trabajo. La grilla de certeza empleada está constituida por un conjunto de 33x33 celdas cada una de ellas con un tamaño igual a 10x10 cm, con el vehículo ubicado en todo momento en el centro de la misma. Al interior del micro-controlador, la grilla está representada por una matriz de similares dimensiones, cada celda representa un valor de certeza constituido por 8 bits (1byte) de manera tal que en total se emplean 1089bytes (1Kbyte aproximadamente) de la memoria RAM del dispositivo para reconstruir el entorno, en la figura 31, se indica el diagrama de flujo generado para los procesos necesarios para la navegación.

La orientación que el móvil debe adoptar en un instante de tiempo determinado para desplazarse hacia el punto objetivo evitando colisionar con los obstáculos presentes a su alrededor, se calcula empleando la función denominada *cálculos* dentro del programa general del micro-controlador maestro del sistema.

A su vez, está relacionada estrechamente con la magnitud de dos fuerzas virtuales establecidas en el método de navegación empleado. La primera corresponde a la fuerza de atracción generada por el punto objetivo y la segunda se manifiesta como una representación de la repulsión generada por la totalidad de los objetos presentes en el entorno de trabajo. Para determinar la magnitud de la fuerza de atracción es necesario conocer el valor de la distancia comprendida entre la

posición actual del vehículo y las coordenadas del punto objetivo, así como el valor asociado a la constante de atracción del sistema. La expresión relacionada al cálculo del vector correspondiente a la fuerza de atracción se muestra en (49).

$$\boldsymbol{F}_A = C_A\left[\frac{x_t - x_0}{d(t)}\hat{x} + \frac{y_t - y_0}{d(t)}\hat{y}\right] \qquad (49)$$

Donde:
F_A = Fuerza atractiva hacia el punto objetivo.
C_A = Constante atractiva del sistema.
x_t = Coordenada x del punto objetivo.
y_t = Coordenada y del punto objetivo.
x_0 = Coordenada x de la posición actual del robot.
y_0 = Coordenada y de la posición actual del robot.

El cálculo del vector fuerza repulsiva relacionado con las características del entorno requiere el uso de la matriz de certeza empleada para recopilar la información entregada por la red sensorial. La expresión matemática empleada para su determinación se muestra en (50).

$$\boldsymbol{F}_{ij} = \frac{C_R C(i,j)}{d^2(i,j)}\left[\frac{x_t - x_0}{d(i,j)}\hat{x} + \frac{y_t - y_0}{d(i,j)}\hat{y}\right] \qquad (50)$$

Donde:
F_{ij} = Fuerza de repulsión ejercida por la celda *i,j.*
C_R = Constante de repulsión del sistema.
$C(i,j)$ = Valor del coeficiente de certeza consignado en la celda *i,j.*
$d(i,j)$ = Distancia existente entre la celda central de la grilla y la celda *i,j.*
x_t = Coordenada x del punto objetivo.
y_t = Coordenada y del punto objetivo.
x_0 = Coordenada x de la posición actual del robot.
y_0 = Coordenada y de la posición actual del robot.

Dado que la fuerza de repulsión total a la cual se ve sometido el vehículo corresponde a la sumatoria de las fuerzas ejercidas por cada una de las celdas componentes de la matriz de certeza, la resultante ($\boldsymbol{F}_r$) puede ser expresada como se muestra en (51).

$$\boldsymbol{F}_r = \sum_{i,j} \boldsymbol{F}(i,j) \qquad (51)$$

Al interior del micro-controlador, el cálculo correspondiente a la magnitud de cada componente del vector repulsión se lleva a cabo empleando dos ciclos *for* anidados encargados de recorrer las filas y columnas de la matriz incrementando el valor de la resultante para cada celda recorrida. El diagrama de flujo empleado durante esta etapa de la implementación se muestra en la figura 32.

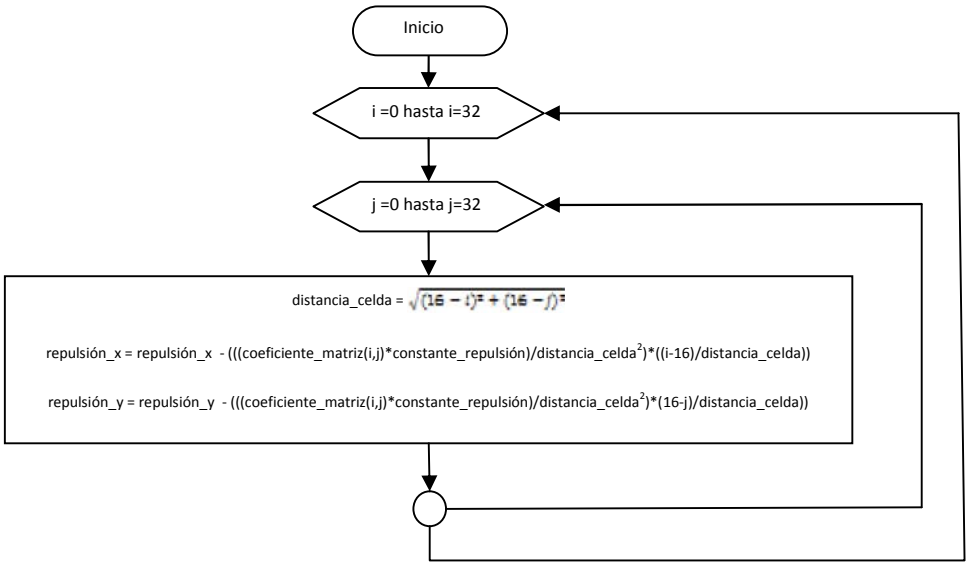


Figura 32. Recorrido de la matriz para el cálculo de la magnitud correspondiente a la fuerza de repulsión.

La resultante generada al sumar los vectores fuerza de atracción y fuerza de repulsión determina la orientación que el móvil debe asumir en el instante de tiempo establecido. La suma vectorial se realiza componente a componente de forma tal que para establecer el ángulo de orientación se hace uso de la función arco tangente tal como se muestra en (52).

$$orientacion = \tan^{-1}\left(\frac{atraccion_y + repulsion_y}{atraccion_x + repulsion_x}\right) \qquad (52)$$

El comando de giro empleado para realizar el movimiento del robot corresponde al mínimo desplazamiento angular necesario para alcanzar la orientación deseada partiendo desde la orientación actual del móvil; por tanto, dicho comando está comprendido entre los 180º y los -180º.

Una vez determinada la orientación que el móvil debe asumir, una serie de controladores se han incluido en el sistema con el fin de disminuir los efectos

negativos presentes en la trayectoria descrita ocasionados por una serie de factores asociados al funcionamiento del método de navegación por campo de fuerzas virtuales. El diagrama de flujo del proceso llevado a cabo durante esta fase de la implementación correspondiente a la función *cálculos* se muestra en la figura 33.

Dado que el vector resultante correspondiente a la suma de las fuerzas repulsivas del sistema en un instante de tiempo determinado puede diferir significativamente de su valor inmediatamente anterior, la trayectoria descrita por el móvil durante su desplazamiento posiblemente presente una serie de oscilaciones relacionadas a la presencia de obstáculos en el entorno inmediato de trabajo. Para mitigar este efecto, es posible monitorear la magnitud del ángulo existente entre los vectores de repulsión y de velocidad instantánea de desplazamiento con el fin de conseguir que el móvil experimente la fuerza de repulsión a su magnitud máxima cuando se aproxima a un obstáculo de manera frontal ($-\cos\theta = 1$). A medida que la dirección de desplazamiento del móvil se aleja de la orientación correspondiente al vector fuerza repulsiva, dicha magnitud se reduce en un factor de amortiguamiento denominado ω. La relación matemática empleada durante esta etapa de la investigación corresponde a la expresión descrita en el capítulo 4 durante la implementación del controlador para amortiguamiento.

Para garantizar que el móvil se desplace a una velocidad acorde a las características del entorno circundante y así disminuir los efectos producidos por cambios abruptos en la orientación a seguir, es necesario establecer una relación matemática determinada por la magnitud del ángulo θ de manera tal, que la velocidad sea máxima para un entorno sin presencia alguna de obstáculos y en caso contrario varíe según la concentración de objetos en el espacio lo exija. El controlador implementado monitorea la magnitud del ángulo θ y establece el comportamiento a seguir según dos reglas preestablecidas; si el móvil se desplaza en dirección frontal hacia un obstáculo (magnitud del $\cos\theta < 0$) la velocidad disminuye en proporción a un factor de reducción empleado en el controlador; en ausencia de obstáculos (magnitud del vector repulsión igual a cero) la velocidad del desplazamiento corresponde a su valor máximo (30cm/s para este caso particular); finalmente si el móvil se desplaza de forma tal que el ángulo θ está comprendido entre los 90° y 180° (el vehículo se aleja de los obstáculos presentes en el entorno) la velocidad se incrementa desde su valor medio hasta el umbral máximo en un factor constante establecido para el sistema. La función resultante para determinar la velocidad de desplazamiento del vehículo se muestra en (53).

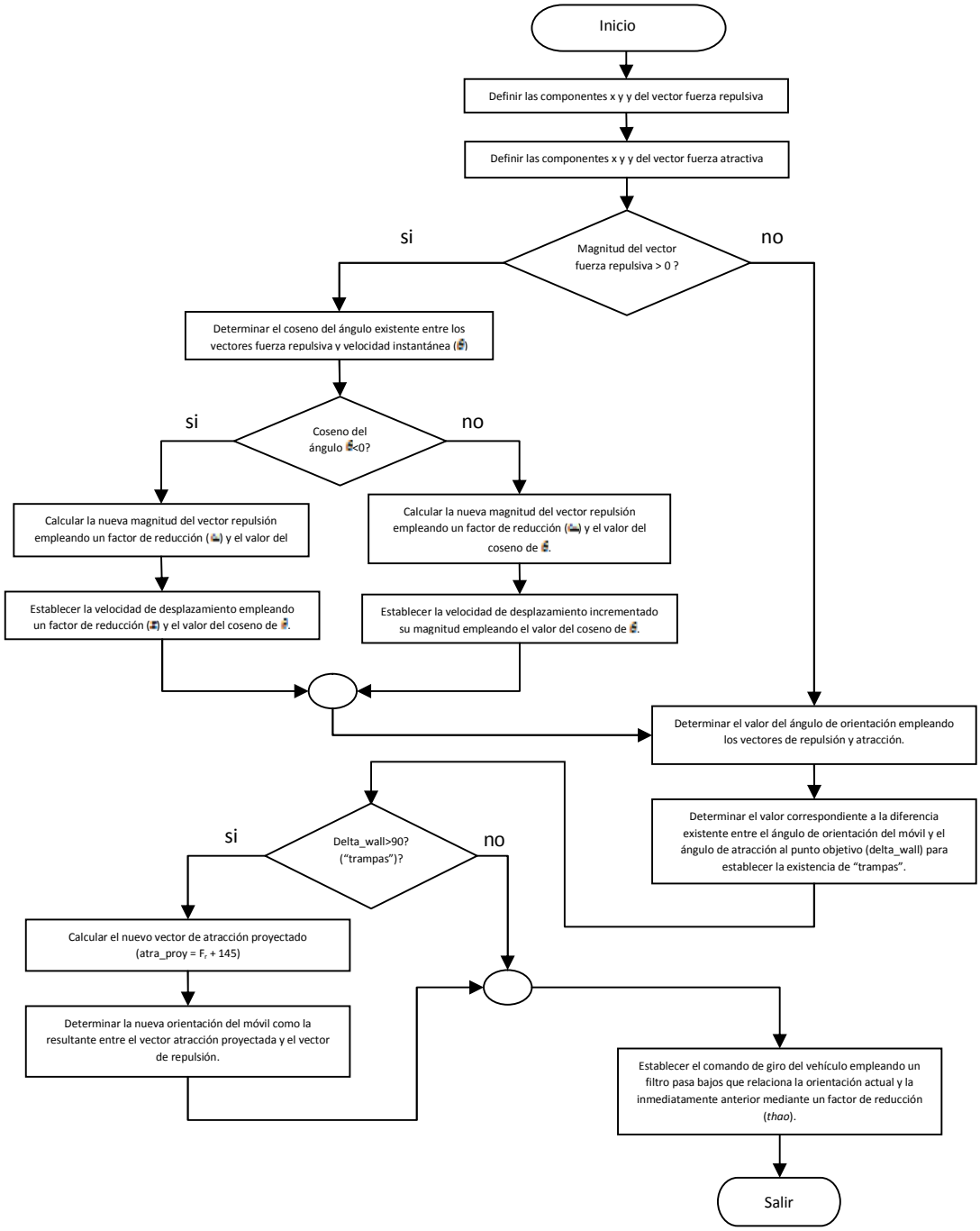


Figura 33. Diagrama de flujo correspondiente a la función *cálculos.*

$$V = \begin{cases} V_{máx} & para\ |F_r| = 0 \\ (zV_{máx}) + \left[(1-z) \left[V_{máx} \left(\frac{\sin\theta}{k_v} \right) \right] \right] & para\ \cos\theta < 0 \\ \left(\frac{V_{máx}}{2} \right) + (0.5 V_{máx} \cos\theta) & para\ \cos\theta > 0 \end{cases} \quad (53)$$

Donde:
$V_{máx}$ = Velocidad máxima que el móvil puede alcanzar durante su desplazamiento.
z = Factor de reducción para el control de velocidad.
θ = Angulo existente entre los vectores fuerza repulsiva y velocidad instantánea.

La constante k_v se emplea para determinar el valor umbral máximo para la velocidad de desplazamiento y está relacionada con el valor asignado al factor de reducción *z* como se muestra en (54).

$$k_v = \frac{(2(1-z))}{(1-(2z))} \quad (54)$$

Una vez determinada la orientación que el móvil debe seguir, existe una condición a evaluar que permite determinar si las características del ambiente suponen la presencia de condiciones físicas denominadas “trampas”. Cada trampa puede definirse como una configuración física del medio cuyas características obligan al móvil a moverse de manera errática, oscilatoria o describiendo trayectorias cerradas continuas que le impiden alcanzar el punto objetivo. La condición a evaluar ($delta_{wall}$) está relacionada con el delta de variación existente entre la orientación que el vehículo debe asumir en un instante de tiempo específico y el ángulo referente al vector de fuerza atractiva durante el mismo instante, de forma tal que al ser superado un valor umbral, se proyecta una nueva fuerza atractiva que obliga al vehículo a moverse siguiendo el contorno de un obstáculo presente en su ambiente inmediato. El valor umbral para este caso particular ha sido establecido en 90° como se aprecia en (55).

$$delta_{wall} = |ángulo_{atraccion} - orientación_{actual}| \quad (55)$$

El vector temporal que representa la fuerza atractiva del sistema se obtiene al sumar un ángulo constante a la magnitud de la orientación correspondiente a la atracción generada por el punto objetivo. Dicho valor constante se ha establecido en 145°. Las expresiones mostradas en (56) hacen referencia a la determinación de la nueva orientación que el móvil debe asumir para desplazarse de acuerdo al método de navegación descrito como *Wall Following.*

$$Atracción_{Proyectada} = \boldsymbol{F}_r + 145°$$

$$Orientación = \tan^{-1}\left(\frac{atracción_{proyectaday} + repulsión_y}{atraccion_{proyectadax} + repulsión_x}\right) \quad (56)$$

Debido a que durante un intervalo correspondiente al tiempo de muestreo pueden presentarse cambios abruptos en la orientación que debe seguir el móvil, un sistema de compensación se introduce en el controlador de navegación con el fin de establecer un tiempo de retraso durante cada medición. Esto se logra mediante la implementación de un filtro pasa bajas regido por la ecuación mostrada en (57).

$$st = \frac{\left((t_{muestreo} \cdot si) + \left((thao - t_{muestreo}) \cdot sta\right)\right)}{thao} \quad (57)$$

st = Comando de giro actual del sistema (entre 180º, y -180º).
$t_{muestreo}$ = Tiempo transcurrido entre cada determinación de la orientación (400ms para este caso en específico).
si = Orientación anterior del móvil.
sta = Comando de giro anterior del móvil.
$thao$ = Constante del filtro pasa bajos.

Las constantes del sistema enunciadas como *thao*, *constante atractiva, constante de repulsión, ω y z,* serán sometidas al proceso de optimización más adelante durante el desarrollo de esta investigación.

7.4 Localización del vehículo en el espacio de trabajo.

La determinación de la posición que el móvil ocupa en el espacio de trabajo es un proceso de gran importancia dentro del sistema de navegación general, puesto que a partir de sus resultados se establecen diversas relaciones que determinan el comportamiento del vehículo.

La estimación de la posición se lleva a cabo empleando el método de estimación odométrica para una plataforma diferencial. Para ello, el DSpic30f4011 empleado en el módulo de navegación constituido como el maestro del sistema general, recopila la información concerniente a la diferencia de pulsos generados por cada codificador de cuadratura ubicado en los motores del sistema de tracción durante un intervalo de tiempo determinado a través de la función denominada *posición_vehículo*. Los datos, son adquiridos en primera instancia por cada uno de

los micro-controladores configurados en modo esclavo, incluidos en el módulo para el control de movimiento haciendo uso del anteriormente mencionado módulo para codificador en cuadratura con el que cada dispositivo cuenta mediante una serie de instrucciones agrupadas en la función de nombre *leer_encoder*. Dicha función tiene como parámetro de entrada la dirección que identifica a cada micro-controlador esclavo en el bus de datos I2C implementado en el prototipo, en el cual el micro-controlador ubicado en el módulo de navegación oficia como maestro y sus similares incluidos en el módulo para el control de movimiento desempeñan el papel de esclavos. La frecuencia de la señal de reloj empleada por el maestro del bus para la trasferencia de los datos corresponde a 100KHz. Una vez obtenido el desplazamiento de cada rueda, como un número de pulsos entregados por cada codificador, la función *recorrido* se encarga de realizar la conversión de estas unidades en centímetros considerando para tal fin las dimensiones de cada rueda, para posteriormente retornar los valores de desplazamiento lineal izquierdo y derecho y a partir de ellos emplear las relaciones odométricas establecidas para un plataforma diferencial y así estimar las coordenadas del punto de giro del vehículo y su orientación inmediata. El diagrama de flujo correspondiente a la función *posición_vehículo* se muestra en la figura 34.

Para comunicarse con cada uno de los micro-controladores configurados como esclavos, el maestro envía inicialmente un bit de inicio sobre el bus de datos I2C, seguido por la dirección del dispositivo a comunicar con el bit de lectura/escritura en nivel bajo. Posteriormente un comando constituido por un byte de información es enviado a fin de preparar al dispositivo esclavo para realizar una operación específica o en su defecto transferir uno o más bytes de información al bus de datos como respuesta.

Si se espera recibir información proveniente del dispositivo esclavo, un bit de reinicio es enviado seguido por la dirección del dispositivo con el bit de lectura/escritura puesto en alto quedando el bus en espera de los datos correspondientes a la respuesta requerida para finalmente enviar un bit de final de transmisión. De lo contrario, existe la posibilidad de enviar más bytes de instrucción o simplemente enviar el bit de parada. El diagrama de flujo correspondiente a la función *leer_encoder* se muestra en la figura 35, en tanto que su similar que representa el funcionamiento básico de la función *recorrido* se muestra en la figura 36.

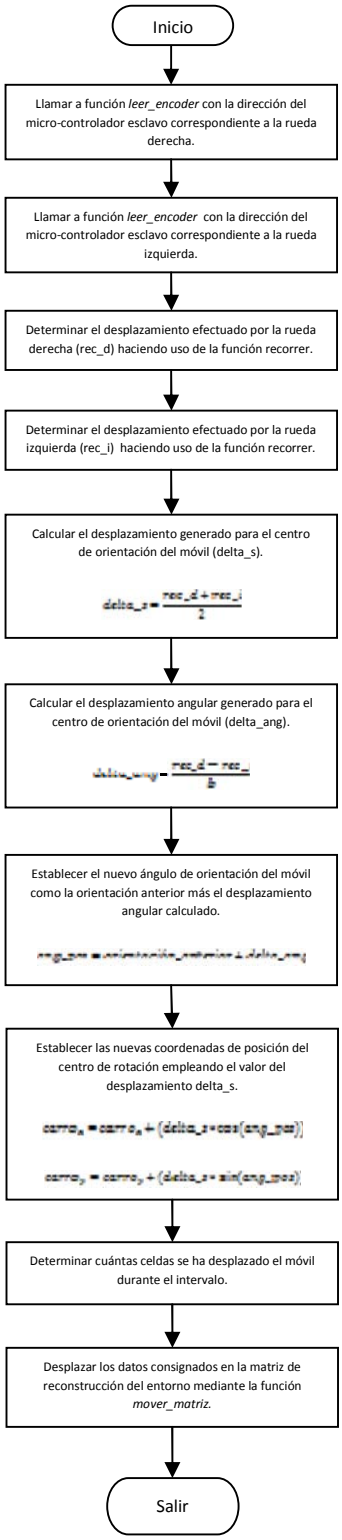


Figura 34. Diagrama de flujo correspondiente a la función *posición_vehículo.*

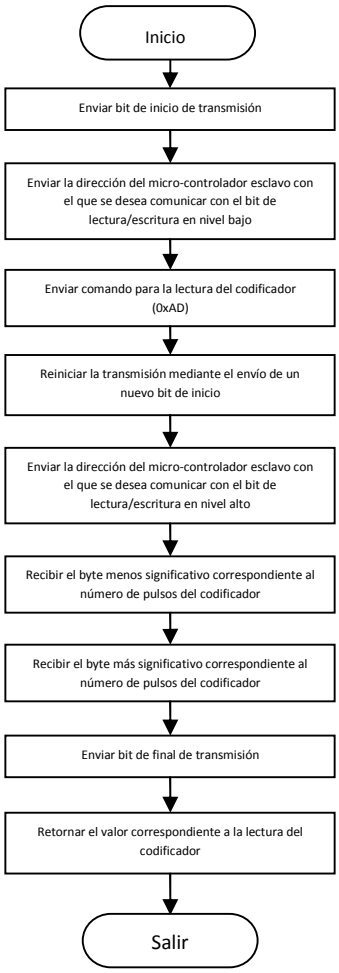


Figura 35. Diagrama de flujo de la función *leer_encoder.*

Las ruedas del prototipo construido poseen un radio r = 5.8cm. El intervalo de tiempo ΔT corresponde al tiempo transcurrido entre cada procedimiento de localización realizado, que para este caso en particular fue establecido en 400ms.

Establecido el desplazamiento generado por cada rueda y haciendo uso de las relaciones odométricas establecidas para una plataforma diferencial se determina el desplazamiento efectuado por el punto de giro del vehículo y se establece ángulo de rotación del mismo durante el intervalo ΔT, tal y como se aprecia en la ecuación (58).

$$delta\_s = \frac{rec\_d + rec\_i}{2}$$
$$delta\_ang = \frac{rec\_d - rec\_i}{b} \quad (58)$$

Siendo:

$delta\_s$ = Desplazamiento realizado por el centro de giro del vehículo durante el intervalo de muestreo.
rec_d = Recorrido efectuado por la rueda derecha.
rec_i = Recorrido efectuado por la rueda izquierda.
$delta\_ang$ = Desplazamiento angular efectuado por el centro de giro del vehículo durante el intervalo de muestreo.
b = Distancia de separación entre las ruedas del móvil.

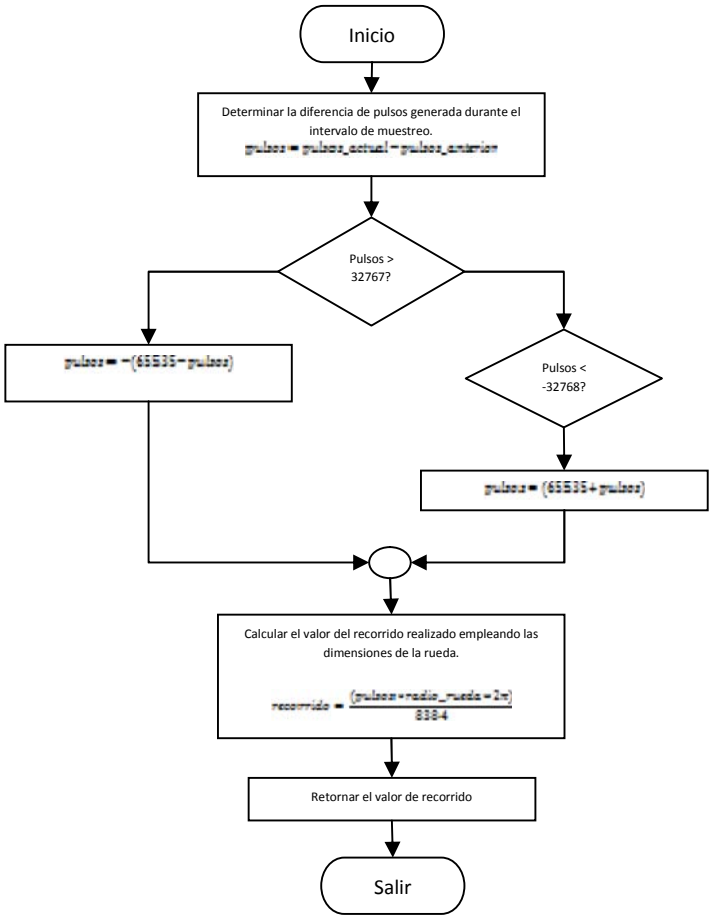


Figura 36. El diagrama de flujo correspondiente a la función *recorrido.*

La nueva orientación del móvil está determinada por el incremento correspondiente a *delta_ang*. A partir de este valor es posible determinar las nuevas coordenadas del centro de giro del vehículo empleando las relaciones establecidas en las ecuaciones (59), (60) y (61).

$$ang\_pos = ang\_actual + delta\_ang \quad (59)$$

$$\Delta x = delta\_s \cdot \cos(ang\_pos) \quad (60)$$

$$\Delta y = delta\_s \cdot sin(ang\_pos) \quad (61)$$

Donde:
ang_pos = Angulo de orientación final del móvil.
ang_actual = Angulo de orientación del móvil antes del intervalo de muestreo.
$delta\_ang$ = Desplazamiento angular efectuado durante el intervalo de muestreo.
Δx = Desplazamiento del punto de giro del vehículo en el eje x durante el intervalo de muestreo.
Δy = Desplazamiento del punto de giro del vehículo en el eje y durante el intervalo de muestreo.
$delta\_s$ = Desplazamiento realizado por el centro de giro del vehículo durante el intervalo de muestreo.

Las nuevas coordenadas del punto de giro del vehículo vendrán determinadas por los desplazamientos componentes del desplazamiento final *delta_s.* La relación final establecida se presenta en las ecuaciones (62) y (63).

$$carro\_x = carro\_x\_anterior + \Delta x \quad (62)$$

$$carro\_y = carro\_y\_anterior + \Delta y \quad (63)$$

Donde:
$carro\_x$ = Coordenada *x* final del vehículo.
$carro\_y$ = Coordenada *y* final del vehículo.
$carro\_x\_anterior$ = Coordenada *x* anterior del vehículo.
$carro\_y\_anterior$ = Coordenada *y* anterior del vehículo.

Para determinar si los datos consignados en la matriz de reconstrucción del entorno deben ser desplazados es necesario conocer la ubicación del móvil respecto a la configuración en celdas del entorno. Para tal fin, el vehículo se posiciona en una celda determinada cada vez que el proceso de localización se lleva a cabo, estableciendo cuantas celdas se ha desplazado desde su posición anterior en cada dirección posible. Para ello, dos variables denominadas *celda_x* y *celda_y* se establecen con el fin de asignar a cada una de ellas la celda ocupada por el vehículo en un instante de tiempo determinado como se muestra en las ecuaciones (64) y (65).

$$celda\_x = \frac{carro\_x}{10} \quad (64)$$

$$celda\_y = \frac{carro\_y}{10} \quad (65)$$

Debido a que la posición del vehículo está definida por dos coordenadas absolutas es posible determinar cuántas celdas se ha desplazado durante el intervalo en estudio tanto en el eje x como en y obteniendo el valor entero de las relaciones presentadas anteriormente. Si las coordenadas de la celda ocupada durante la anterior iteración se conservan en dos variables independientes es posible determinar los desplazamientos realizados en cada eje mediante la diferencia de los valores actual y anterior (66).

$$\begin{aligned} delta\_celda\_x &= celda\_x - celda\_anterior\_x \\ delta\_celda\_y &= celda\_y - celda\_anterior\_y \end{aligned} \quad (66)$$

Donde:
$delta\_celda\_x$ = Celdas desplazadas en la dirección x.
$delta\_celda\_y$ = Celdas desplazadas en la dirección y.
$celda\_x$ = Coordenada x de la celda ocupada por el móvil.
$celda\_y$ = Coordenada y de la celda ocupada por el móvil.
$celda\_anterior\_x$ = Coordenada x de la celda ocupada por el móvil durante el intervalo inmediatamente anterior.
$celda\_anterior\_y$ = Coordenada y de la celda ocupada por el móvil durante el intervalo inmediatamente anterior.

Si existe un desplazamiento diferente de cero en cualquiera de los dos ejes, un algoritmo de corrimiento de matriz llevado a cabo por la función de nombre *mover_matriz* es puesto en marcha, desplazando los valores consignados en la matriz que contiene los datos adquiridos por los sensores de distancia según el movimiento efectuado por el vehículo. El diagrama de flujo correspondiente a la función *mover_matriz* se presenta en la figura 37.

7.5 Envío de datos hacia el PC.

El proceso de comunicación con el PC se lleva a cabo empleando el módulo UART incluido en el micro-controlador que hace parte del módulo de navegación del sistema.

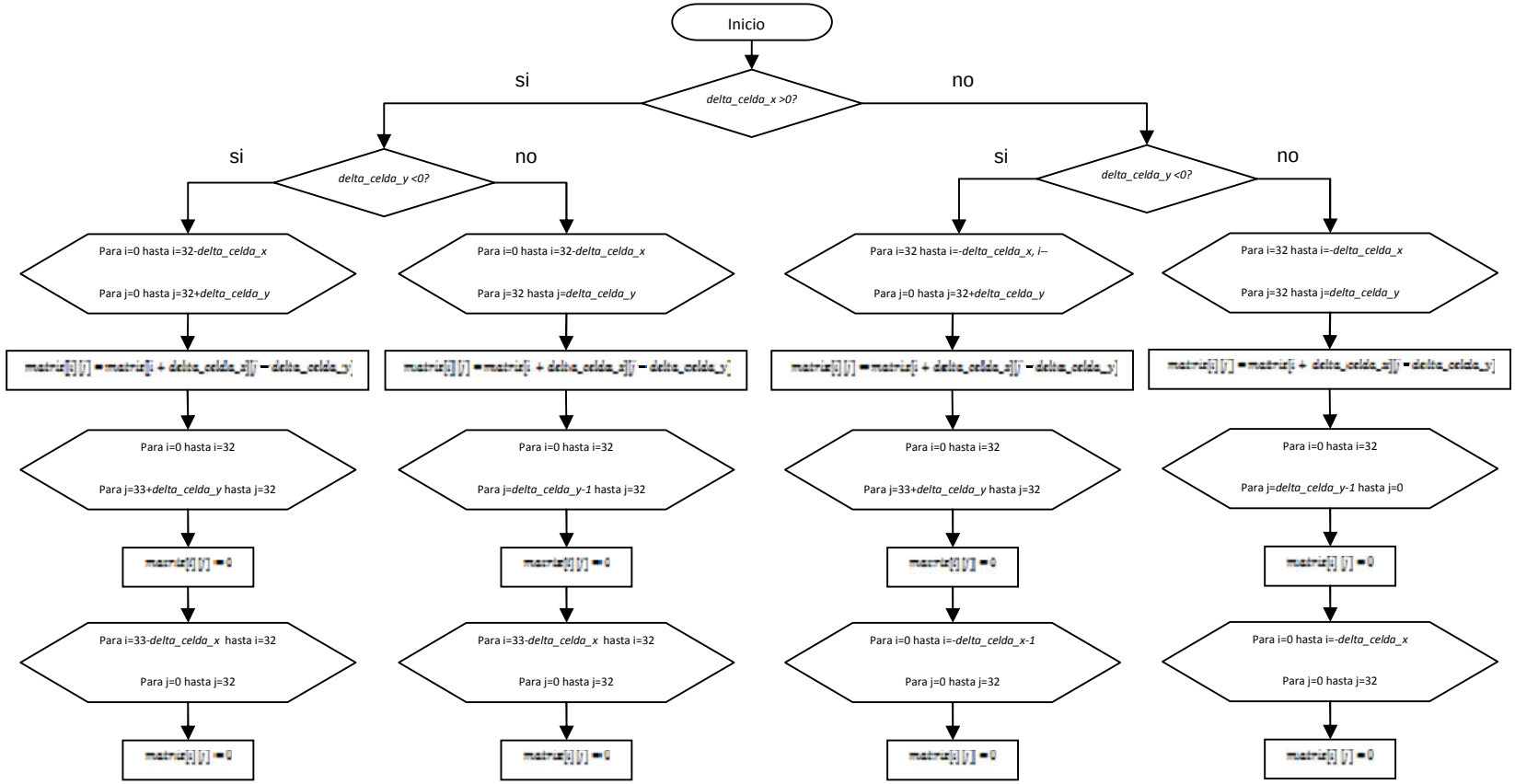


Figura 37. Diagrama de flujo de la función *mover_matriz*.

La tasa de transferencia de datos corresponde a los 9600 baudios, empleando durante cada transmisión 8 bits de datos, 1 bit de inicio y 2 bits de parada. La función encargada de realizar este procedimiento se denomina *envio_serial* y es invocada cada vez que finaliza el proceso de localización del vehículo. La trama enviada está constituida por un conjunto de 22 bytes, cada uno de ellos asignado a una variable específica cuyos valores son necesarios para realizar la presentación de la información al usuario del software que sirve como interfaz entre el vehículo y su operador. Las variables asignadas dentro de este conjunto se muestran en la tabla 4. La cabecera de la trama está constituida por un valor hexadecimal (0xAA) encargado de garantizar la recepción de una nueva serie de variables.

POSICION DEL BYTE EN LA TRAMA	VARIABLE A TRANSMITIR
1	Cabecera de la trama
2	Variable de asignación libre
3	Celda correspondiente a la distancia medida por el sensor 1 (componente x)
4	Celda correspondiente a la distancia medida por el sensor 1 (componente y)
5	Celda correspondiente a la distancia medida por el sensor 2 (componente x)
6	Celda correspondiente a la distancia medida por el sensor 2 (componente y)
7	Sentido de giro
8	Variable de asignación libre
9	Desplazamiento de la matriz en la dirección x
10	Desplazamiento de la matriz en la dirección y
11	Primer byte de la posición del vehículo en la dirección x
12	Segundo byte de la posición del vehículo en la dirección x
13	Tercer byte de la posición del vehículo en la dirección x
14	Cuarto byte de la posición del vehículo en la dirección x
15	Primer byte de la posición del vehículo en la dirección y
16	Segundo byte de la posición del vehículo en la dirección y
17	Tercer byte de la posición del vehículo en la dirección y
18	Cuarto byte de la posición del vehículo en la dirección y
19	Primer byte de la orientación del vehículo
20	Segundo byte de la orientación del vehículo
21	Tercer byte de la orientación del vehículo
22	Cuarto byte de la orientación del vehículo

Tabla 4. Variables asignadas a la trama enviada constituida de 22 bytes.

Las variables correspondientes a la posición de las celdas ocupadas por los obstáculos detectados mediante los sensores 1 y 2 son de tipo entero de 8 bits y están comprendidas por valores entre 1 y 33 tanto para la componente x como para la y. La variable correspondiente al sentido de giro hace referencia a una de cuatro posibilidades que determinan la dirección en la cual el vehículo se está desplazando de forma tal que el valor 4 corresponde a un giro a la izquierda, 3 un giro a la derecha, 5 desplazamiento hacia adelante y 2 motores detenidos. Los desplazamientos de la matriz para la reconstrucción del entorno

están representados por variables enteras de 8 bits, puesto que debido a la velocidad máxima de desplazamiento del móvil durante dos intervalos de muestreo consecutivos, el desplazamiento máximo no supera las 5 celdas en cualquier dirección.

Las variables correspondientes a la posición del vehículo en las direcciones x y y son de tipo flotante de 32 bits bajo la configuración consignada en el estándar IEEE-754.Para realizar la transmisión de las mismas se hace necesario emplear 4 bytes en cada caso de manera tal que los bytes correspondientes a la posición en la dirección x son enviados en las ubicaciones 11, 12,13 y 14 de la trama general y la variable de posición en y es enviada en las casillas 15, 16, 17 y 18, de manera tal que en los dos casos el primer byte enviado corresponde al byte menos significativo de la variable. La variable empleada para transmitir la orientación del vehículo posee una estructura similar a la descrita anteriormente para las variables de posición, siendo el byte 19 de la trama el menos significativo del valor a enviar y los restantes bytes (20, 21 y 22) complementan los 32 bits necesarios para su representación.

7.6 Orientación del vehículo en la dirección establecida.

Una vez establecida la orientación que el móvil debe asumir para desplazarse con rumbo al punto objetivo evitando los obstáculos presentes en el entorno, es necesario alinear el punto de giro del vehículo con dicho vector dirección. Para ello es necesario conocer la posición y dirección actuales del móvil con el fin de establecer la dirección del movimiento que debe ser efectuado para encaminarse hacia la orientación previamente establecida. Dicho procedimiento se lleva a cabo mediante la función *orientar* consignada en el programa principal del micro-controlador maestro del sistema. Inicialmente, debe ser determinada la distancia existente desde la posición actual hasta el punto objetivo con el fin de constatar la condición establecida para la detención del móvil, de manera tal que si dicho valor es menor a un parámetro delimitado para tal fin, el vehículo debe detenerse y en caso contrario debe asumir la dirección entregada después de realizar los cálculos correspondientes al método de campo de fuerzas virtuales. Si la distancia mínima al punto objetivo aún no ha sido alcanzada, la orientación del móvil está determinada por el valor entregado por la brújula magnética y el valor determinado a partir de los codificadores en cuadratura empleados durante la localización del vehículo. Estas dos variables son promediadas con el fin de disminuir el error generado durante el movimiento del vehículo, siendo dicho promedio comparado con el valor de la orientación deseada empleando una resta para definir el sentido de giro que el vehículo debe asumir. Si el valor absoluto del resultado de la resta entre las orientaciones actual y deseada es menor a 5º el vehículo conserva el rumbo y no realiza giro alguno (la variable sentido de giro toma el valor de 5);

por el contrario, si dicha valor resulta mayor a 5º, el resultado de la resta entre las dos variables mencionadas es empleado para determinar el sentido de giro de la siguiente manera: si el valor está comprendido entre 5º y 180º el vehículo debe girar a la izquierda para alcanzar la orientación deseada (la variable sentido de giro toma el valor de 4). Cuando dicho valor es menor a 5º y superior a -180º el vehículo gira hacia la derecha (la variable sentido de giro toma el valor de 3). El sentido de giro de los motores se mantiene durante el tiempo correspondiente al intervalo de muestreo. El proceso llevado a cabo durante cada ejecución de la función *orientar* se muestra en el diagrama de flujo correspondiente a la figura 38.

La variable sentido de giro es enviada a cada micro-controlador esclavo incluido en el módulo para el control de movimiento, empleando el bus de datos I2C y la dirección asignada a cada dispositivo en el bus mencionado. El DSpic encargado de controlar el movimiento del motor izquierdo tiene como dirección 0x30 en tanto que su similar ubicado en la parte derecha está direccionado con el valor 0x38. Internamente, cada micro-controlador esclavo está programado con una serie de rutinas encargas de gestionar el encendido del motor en la dirección requerida, la lectura de los pulsos entregados por el codificador en cuadratura y el establecimiento de las condiciones de las señales pwm empleadas para el control de los motores según lo determine la velocidad calculada para cada movimiento. Para controlar el sentido de giro de los motores, el dispositivo maestro emplea la función *I2C_motor* cuya estructura se muestra en el diagrama de flujo consignado en la figura 39. Dado que cada micro-controlador esclavo debe gestionar tres comandos de instrucción correspondientes a la lectura del codificador, al sentido de giro de los motores y a la velocidad de desplazamiento, internamente se encuentra programado de forma tal que la gestión de la interrupción generada por la recepción de un mensaje vía I2C se lleva a cabo considerando el valor asignado a la variable de control enviada inmediatamente después de la dirección del dispositivo. Cuando el comando recibido corresponde a 0xAB, el módulo I2C espera por el dato correspondiente al sentido de giro que debe ser establecido para cada motor. Una vez recibida tal instrucción, las salidas del módulo pwm son habilitadas para generar la rotación necesaria para hacer que el vehículo gire en la dirección establecida, avance hacia el punto objetivo o se detenga. Si el comando recibido tiene el valor 0xAC, el módulo espera el valor correspondiente a la velocidad de desplazamiento para establecer las características de la señal pwm que controla cada motor. Finalmente si la variable de control posee el valor 0xAD, el dispositivo se prepara para realizar la lectura del registro encargado de almacenar la información correspondiente al codificador en cuadratura para posteriormente enviarla hacia el maestro.

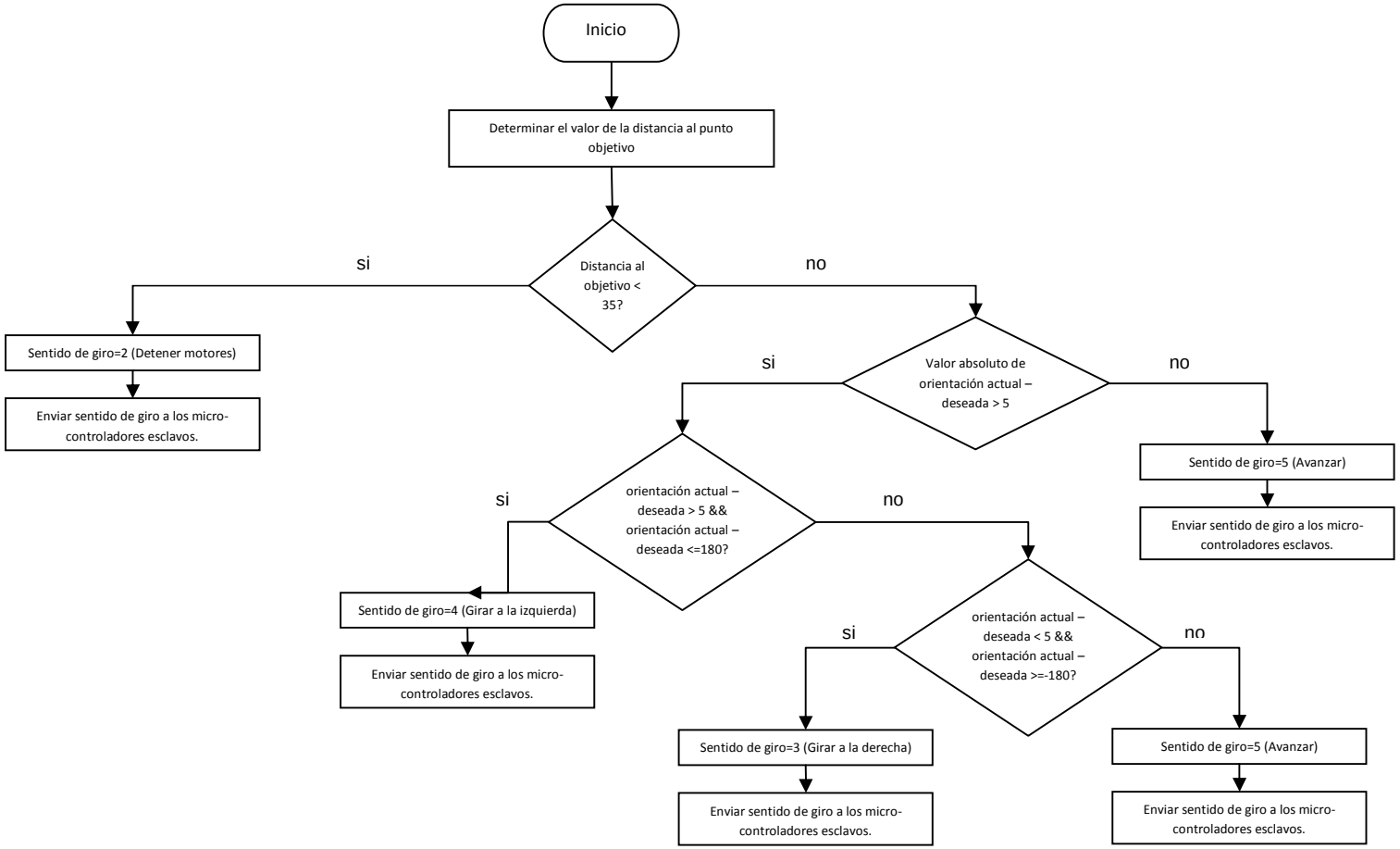


Figura 38. Diagrama de flujo que indica el proceso llevado a cabo durante cada ejecución de la función *orientar*.

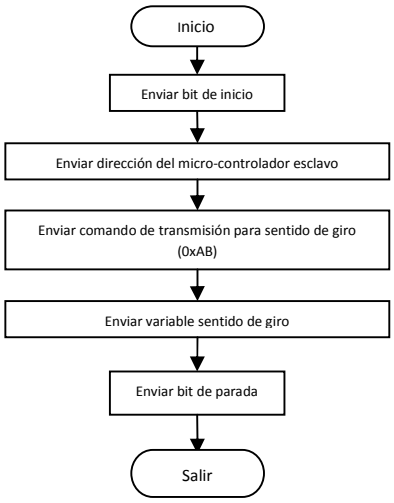


Figura 39. Diagrama de flujo en el cual se muestra la estructura la función *I2C_motor*, que se utiliza para controlar el sentido de giro de los motores.

La gestión de la interrupción relacionada a la recepción de mensajes vía I2C se esquematiza en la figura 40.

Una vez recibido el byte correspondiente al comando de control, el micro-controlador realiza una serie de procedimientos a fin de cumplir con la labor especificada por dicha variable haciendo uso de un parámetro denominado *bandera.* Para establecer el sentido de giro de cada motor (comando de control=0xAB), cada micro-controlador esclavo hace uso de dos salidas del módulo pwm con el fin de polarizar las entradas del controlador de corriente empleado para alimentar las líneas de poder de cada motor DC, mediante un esquema lógico definido a partir de la ubicación de cada motor en el vehículo. Las relaciones lógicas establecidas para los motores derecho (motor 1) e izquierdo (motor 2) se presentan en las tablas 5 y 6.

VALOR COMANDO SENTIDO DE GIRO	**NIVEL LOGICO SALIDA 1**	**NIVEL LOGICO SALIDA 2**	**SENTIDO DE GIRO MOTOR 1 (Motor derecho)**
2 (Detener motores)	BAJO	BAJO	DETENER MOTOR
3 (Girar a la izquierda)	BAJO	ALTO	ADELANTE
4 (Girar a la derecha)	ALTO	BAJO	ATRAS
5 (Avanzar)	BAJO	ALTO	ADELANTE

Tabla 5. Relación lógica establecida para el motor derecho (motor 1).

VALOR COMANDO DE GIRO	NIVEL LOGICO SALIDA 1	NIVEL LOGICO SALIDA 2	SENTIDO DE GIRO MOTOR 2 (Motor izquierdo)
2 (Detener motores)	BAJO	BAJO	DETENER MOTOR
3 (Girar a la izquierda)	BAJO	ALTO	ATRÁS
4 (Girar a la derecha)	ALTO	BAJO	ADELANTE
5 (Avanzar)	ALTO	BAJO	ADELANTE

Tabla 6. Relación lógica establecida para el motor izquierdo (motor 2).

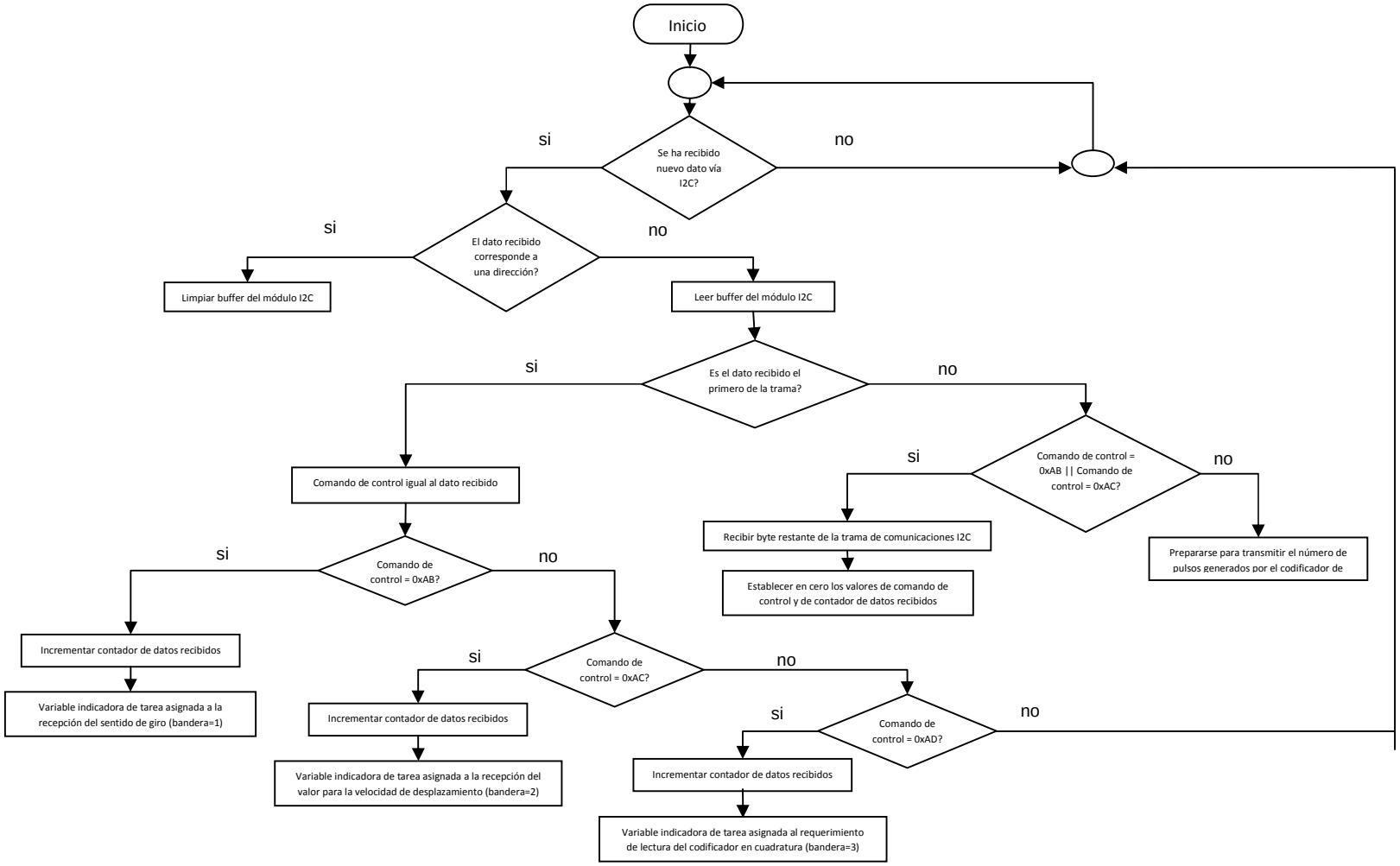


Figura 40. Diagrama de flujo en el cual se esquematiza la interrupción relacionada a la recepción de mensajes vía I2C.

El movimiento del vehículo se presenta entonces como resultado de la interacción de los movimientos independientes de cada motor durante un intervalo de tiempo determinado como resultado de la lógica consignada en la tabla 7.

SENTIDO DE GIRO MOTOR 1	SENTIDO DE GIRO MOTOR 2	MOVIMIENTO DEL VEHICULO
ADELANTE	ADELANTE	HACIA ADELANTE
ATRÁS	ADELANTE	GIRO A LA DERECHA
ADELANTE	ATRAS	GIRO A LA IZQUIERDA

Tabla 7. Lógica del movimiento del vehículo, resultado de la interacción de los movimientos independientes de cada motor durante un intervalo de tiempo determinado.

Cuando el controlador de navegación determina que el vehículo debe desplazarse hacia adelante, es necesario determinar con qué velocidad se llevará a cabo este proceso. Para ello, una vez recibido el comando de control correspondiente a esta labor (0xAC), el dispositivo puede asumir 255 velocidades diferentes determinadas por la variable enviada por el maestro siendo esta de tipo entero de 8 bits.

Sin embargo, dadas las características físicas del vehículo desarrollado, se ha establecido como parámetro mínimo para la velocidad de desplazamiento el valor de 0.15m/s. La variación en la velocidad lineal del móvil está relacionada con el ciclo útil de la señal pwm empleada establecido el periodo de la misma en un valor constante de 20ms, tiempo sobre el cual habrá de establecerse un porcentaje de señal en alto para variar la velocidad del vehículo. Para determinar la magnitud del ciclo útil, el módulo pwm del dispositivo cuenta con un registro de 16 bits de forma tal que un valor de 49998 determina la velocidad máxima del vehículo (46cm/s). Sin embargo cabe recordar que como restricción del sistema, el umbral máximo se ha fijado en 30cm/s. Una vez recibida la variable velocidad, se realiza el cálculo porcentual de este valor sobre la velocidad máxima del vehículo y se procede a generar el valor del registro para el ciclo de trabajo a fin de alcanzar el desplazamiento requerido. Establecido el tiempo de la señal en alto se procede a polarizar las entradas del controlador de los motores con el fin de avanzar en la dirección requerida.

7.7 Recopilación de la información de los sensores de distancia

La reconstrucción del entorno de trabajo se lleva a cabo empleando la información entregada por dos sensores ultrasónicos encargados de establecer cuál es la distancia existente hasta el obstáculo más próximo. Para recopilar dicha

información, se hace uso del bus de comunicaciones I2C, adquiriendo el dato entregado por cada sensor en un instante de tiempo determinado empleando las direcciones asignadas a cada dispositivo, el sensor número 1 asignado como 0xE0 y el número 2 como 0xE2.

La trama empleada durante cada comunicación está constituida por un bit de inicio seguido por la dirección del sensor con el cual va a establecerse el intercambio de datos con el bit de lectura/escritura en nivel bajo. Posteriormente son enviados dos comandos requeridos por el sensor, el primero corresponde al valor hexadecimal 0x00 que prepara al dispositivo para la recepción del comando de instrucción a ejecutar, correspondiente al valor 0x51 para la determinación de la distancia en centímetros. Un bit de parada se hace necesario con el fin de incluir un retardo en la transmisión de 80ms para permitir que el dispositivo realice las operaciones necesarias para el envío y recepción de la señal ultrasónica así como el procesamiento del tiempo de vuelo determinado para a partir de esta magnitud establecer la distancia medida hasta un objeto. Transcurrido este periodo de tiempo, los dos bytes que constituyen la medición de la distancia se encuentran listos a ser transmitidos por el sensor de forma tal que una nueva trama es enviada, conformada por un bit de inicio, la dirección del sensor con el bit de lectura/escritura en nivel bajo, el valor correspondiente al primer registro a leer en el sensor (0x01), un bit de reinicio de transmisión y la dirección del sensor empleando el bit de lectura/escritura en nivel alto. En este punto el micro-controlador maestro está listo para recibir los dos bytes de información requerida siendo el primero el menos significativo y el segundo el byte más significativo de la distancia determinada. Finalmente se envía un bit de parada que pone fin a la transmisión. Este procedimiento es llevado a cabo empleando la función *sensor_i2c* del programa principal y su funcionamiento se bosqueja en el pseudocódigo presente en la figura 41.

Una vez obtenidos los dos bytes correspondientes a la variable distancia medida cuya naturaleza es de tipo entero de 16 bits, es necesario realiza la integración de estos dos datos a fin de construir el valor buscado. Este procedimiento se lleva a cabo empleando la relación matemática expresada en (67).

$$distancia = (byte1 \square 256) + byte2 \qquad (67)$$

Donde:

$byte1$ = Byte más significativo entregado por el sensor.

$byte2$ = Byte menos significativo entregado por el sensor.

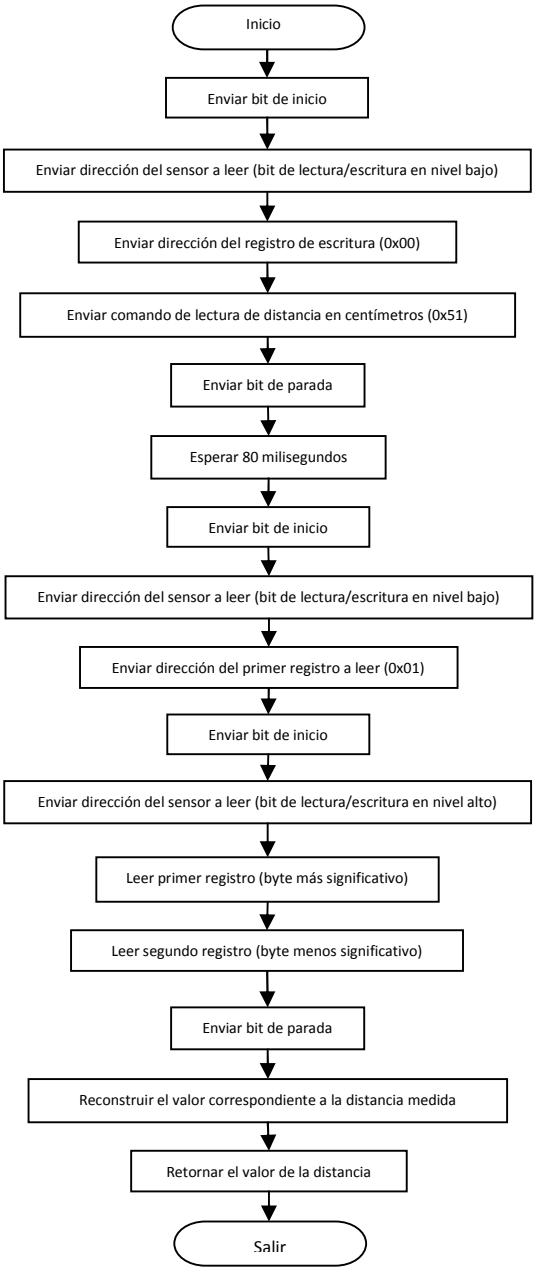


Figura 41. Diagrama correspondiente a la función *sensor_i2c.*

La función sensor_i2c es convocada una vez por cada sensor ultrasónico entregando en cada ocasión, un valor entero de 16bits correspondiente a la distancia establecida hasta el objeto más próximo por el sensor en uso durante su ejecución.

7.8 Inclusión de los datos correspondientes a la distancia medida en la matriz VFF.

La reconstrucción del entorno de trabajo se lleva a cabo a partir de la información consignada en la matriz empleada durante el desarrollo del método de campo de fuerzas virtuales. Los datos consignados en la misma corresponden a las proyecciones de los valores entregados por los sensores ultrasónicos para la determinación de la distancia existente hasta el obstáculo más próximo, empleando para tal cálculo la información concerniente a la ubicación y orientación del móvil durante el instante de medición. Dado que las dimensiones de la matriz para la reconstrucción únicamente permiten consignar en ella valores de distancia inferiores a los 160cm, la función de inclusión de datos denominada *matriz_sensor* es puesta en marcha en el caso tal que esta restricción sea cumplida.

Una vez obtenidos los valores de distancia anteriormente mencionados, tras un procesamiento de los mismos se establece la celda a la que pertenece dicha posición dentro de la segmentación hecha del espacio circundante, empleando las relaciones expuestas en (68).

$$distancia\_x = distancia \cdot \cos(ang\_actual)$$

$$distancia\_y = distancia \cdot \sin(ang\_actual)$$

$$celda\_x = \frac{distancia\_x}{10} \qquad (68)$$

$$celda\_y = \frac{distancia\_y}{10}$$

Siendo:
$distancia\_x$ = Componente x de la distancia obtenida por el sensor.
$distancia\_y$ = Componente y de la distancia obtenida por el sensor.
$celda\_x$ = Coordenada x de la celda correspondiente a la distancia medida.
$celda\_y$ = Coordenada y de la celda correspondiente a la distancia medida.

Las variables *celda_x* y *celda_y* corresponden a los valores enteros de las proyecciones de la distancia sobre cada eje, divididas por las dimensiones de las celdas que constituyen la matriz, en este caso 10cm en cada dirección. El coeficiente correspondiente a la posición de coordenadas *celda_x, celda_y* es incrementado en 1 cada vez que la distancia determinada por los sensores lo

amerita. Durante cada proceso de medición, cada sensor entrega un valor correspondiente a distancia de manera tal que el procedimiento anteriormente descrito se lleva a cabo una vez para cada dispositivo dentro de la misma función, realizando la inclusión de dos datos por ciclo. El pseudocódigo empleado por la función *matriz_sensor* se muestra en la figura 42.

7.9 Movimiento del servo-motor empleado por el sonar.

El sonar empleado para llevar a cabo la exploración del entorno inmediato del robot está constituido por un par de sensores ultrasónicos para la determinación de la distancia hasta el objeto más cercano y un servo-motor que asume una serie de posiciones preestablecidas a fin de apuntar los sensores en un ángulo determinado. La secuencia de ángulos que el motor aplica en su funcionamiento está determinada por las características operativas de los dispositivos sensoriales, de forma tal que entre cada posición existe una diferencia angular correspondiente a los 30º de modo tal que se cumplen las disposiciones indicadas por el fabricante de los dispositivos ultrasónicos en lo que respecta a la sensibilidad del sensor con respecto a la distancia de rebote del rayo emitido. Un esquema representativo de este comportamiento se presenta en la figura 43.

Cada posición está relacionada con una orientación relativa del sonar a fin de realizar un barrido de 360º durante cada recorrido empleando el par de sensores, de manera tal que cada sensor tiene asignado un recorrido angular equivalente a 180° empleando los pasos angulares expresados en la tabla 8 para cada uno de los sensores. La orientación absoluta del sonar empleada para realizar la proyección de los datos recopilados de los sensores de distancia sobre la matriz para la reconstrucción del entorno está determinada entonces por la suma algebraica de su posición relativa y la orientación absoluta del móvil en un instante de tiempo específico.

POSICION	ANGULO SENSOR 1	ANGULO SENSOR 2
0	90	-90
1	60	-120
2	30	-150
3	0	180
4	30	-150
5	60	-120
6	90	-90

Tabla 8. Donde puede verse las posiciones y su respectiva orientación relacionada al sonar.

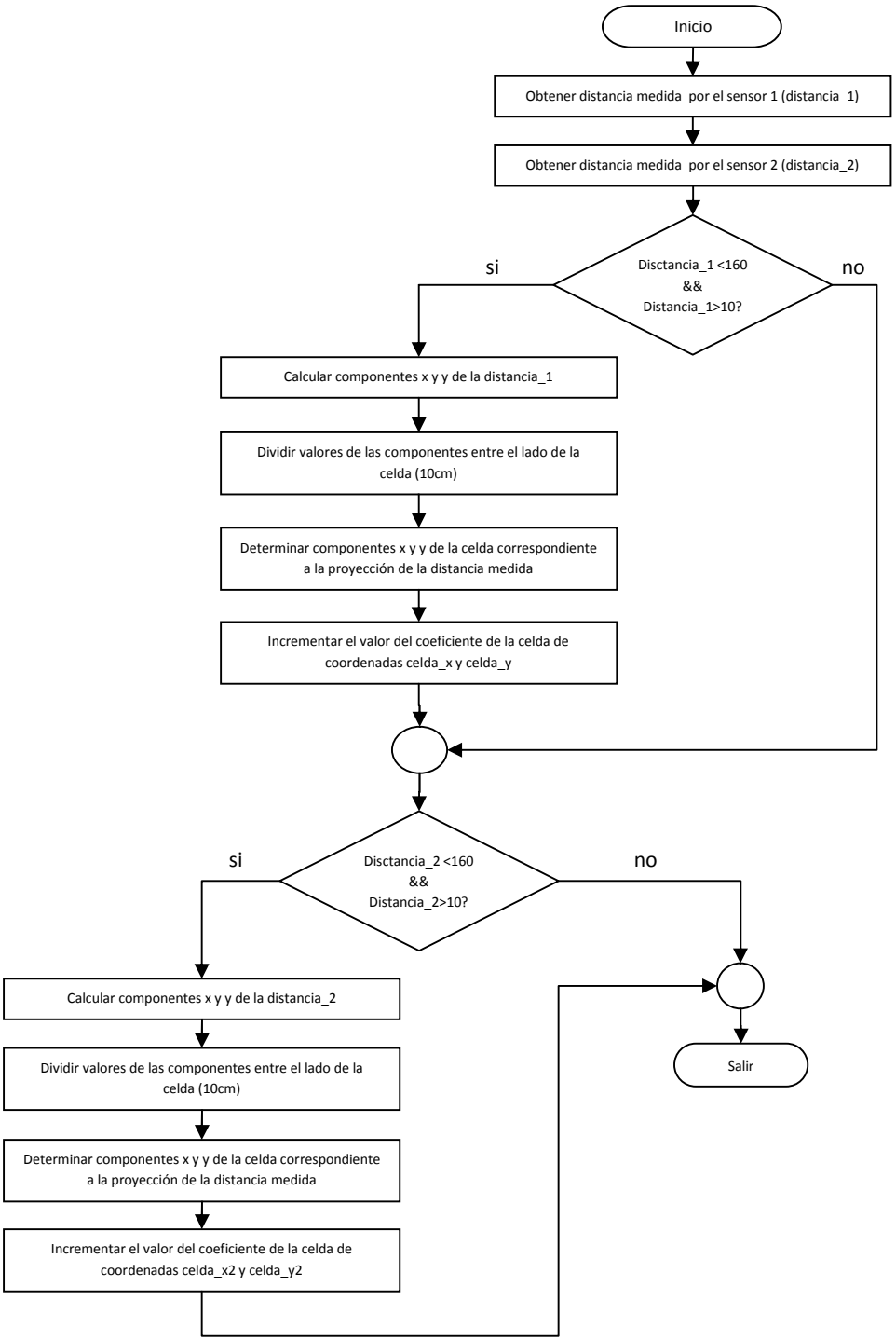
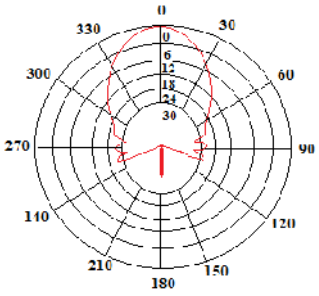
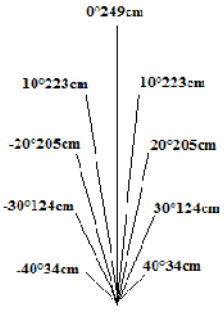


Figura 42. Diagrama de flujo en el cual puede observarse el pseudocódigo empleado por la función. *matriz_sensor.*


0
30
60
90
120
150
180
210
240
270
300
330
0
6
12
18
24
30


0°249cm
10°223cm
10°223cm
-20°205cm
20°205cm
-30°124cm
30°124cm
-40°34cm
40°34cm

8. INTERFAZ DE COMUNICACIONES ENTRE EL VEHICULO Y EL PC

Para recopilar los datos provenientes del vehículo con el fin de permitir al usuario observar el comportamiento del mismo, se hace necesario implementar una interfaz encargada de gestionar la transmisión bidireccional de la información, y el tratamiento de los datos recibidos a fin de presentar de manera gráfica los fenómenos ocurridos durante el desplazamiento del móvil. La interfaz desarrollada está constituida por un módulo físico para la adquisición de datos vía puerto USB, y una aplicación desarrollada bajo el lenguaje de programación Visual C# 2008. A continuación se presentan los datos referentes a la implementación de cada uno de estos componentes haciendo énfasis en la función que desempeñan al interior del interfaz en general.

8.1 Módulo para la transmisión y recepción de datos vía puerto USB.

El circuito encargado de obtener los datos transmitidos por el vehículo mediante el transmisor de radiofrecuencia y de enviarlos al PC empleando el puerto paralelo está constituido por un micro-controlador de referencia PIC18f2550 y un transmisor XBee de referencia y características similares al empleado en el módulo para la transmisión de datos implementado en el prototipo desarrollado. El PIC18f2550 es un dispositivo caracterizado por permitir establecer comunicaciones vía puerto USB empleando la versión 2.0 del protocolo respectivo mediante el uso de un transceptor propio que permite realizar la regulación de los voltajes necesarios para la transmisión mediante un regulador interno. La recepción de los datos provenientes del PC se lleva a cabo a partir de la gestión de varias interrupciones que habilitan diversos tipos de transferencias. De igual manera, este dispositivo está dotado de las características fundamentales de todo micro-controlador de la familia PIC, entre las cuales pueden citarse frecuencia de operación configurable, interrupciones externas, módulos de contador, módulo para la comunicación serial, conversión análoga-digital, set de instrucciones para programación en lenguaje C, entre otras.

Para llevar a cabo la recepción de los datos provenientes del vehículo, el transmisor XBee ha sido configurado de forma tal que una vez recibidos dichos datos, han de ser transmitidos de manera transparente hacia el micro-controlador, es decir que no sufren ningún tipo de transformación. En este nivel, la transmisión se lleva a cabo empleando el modulo UART del dispositivo XBee y su similar dispuesto en el micro-controlador, empleando una protocolo de transmisión compuesta por 1 bit de inicio, 8 bits de datos y 2 bits de parada, a una tasa de transferencia de 9600 baudios. La trama enviada por el vehículo se constituye por

22 bytes de información de forma tal como se presentó en el apartado correspondiente a la transmisión de datos desde el vehículo hacia el PC.

Al interior del micro-controlador, la gestión de los datos provenientes del dispositivo XBee es llevada a cabo empleando una interrupción generada por el módulo UART cada vez que un dato nuevo se encuentra disponible en el buffer de recepción del mismo, mediante una serie de comparaciones destinadas a establecer cuál es el significado del dato recibido. La función ejecutada durante cada convocación de la interrupción se identifica como *recepción_datos* y está constituida por una estructura comparativa tipo interruptor (switch) que permite establecer un orden para la recepción de los datos. Cuando una trama completa ha sido recibida y almacenada en un vector destinado a la transmisión de datos hacia el PC empleando el puerto USB, el micro-controlador envía la información disponible empleando una cadena de datos de longitud igual a 22. La gestión de la recepción de datos provenientes del vehículo y la posterior transmisión de los mismos hacia el PC se bosquejan en la figura 44.

La recepción de los comandos de control enviados por el usuario desde el PC es llevada a cabo empleando el protocolo de comunicaciones USB mediante una interrupción generada al cumplirse una trasmisión de longitud preestablecida. Cada comando enviado desde el PC está constituido por una cadena de datos de longitud 4, en la que cada combinación de bytes está relacionada con una instrucción definida tal y como se expresó en el apartado correspondiente a la recepción de datos provenientes del PC dentro del prototipo desarrollado. Dado que cada transmisión proveniente del PC posee una longitud fija preestablecida, la interrupción encargada de enviar los datos al transceptor XBee se produce cuando la totalidad de los bytes que constituyen un comando de control han sido recibidos de manera tal que en este punto, el micro-controlador realiza la transmisión de datos empleando el módulo UART hacia el transceptor que a su vez envía esta información de forma transparente hacia el vehículo. El esquema general del programa que coordina las actividades del módulo para la recepción y transmisión de datos vía puerto USB se ilustra en la figura 45.

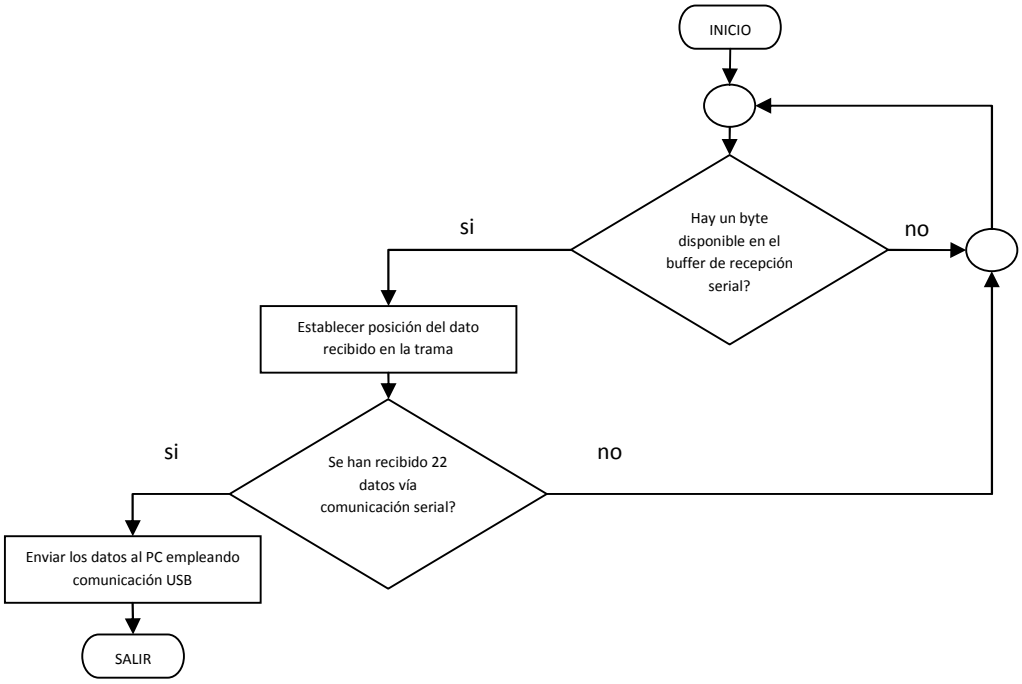


Figura 44. Diagrama de flujo en el cual puede observarse la gestión de recepción de datos provenientes del vehículo, la transmisión de estos hacia el PC.

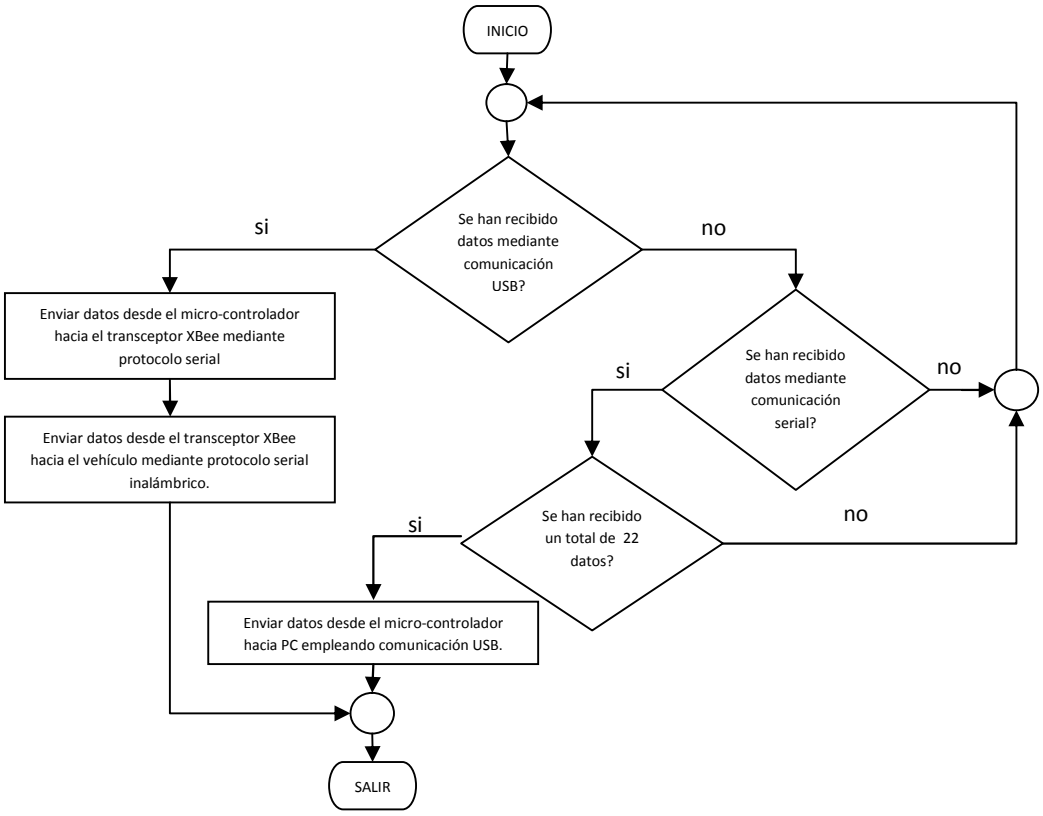


Figura 45. Diagrama de flujo del esquema general del programa que coordina las actividades del módulo para la recepción y transmisión de datos vía puerto USB.

8.2 Interfaz gráfica para la presentación de datos en el PC.

Para poner en marcha el prototipo en busca de alcanzar un punto objetivo dentro del espacio de trabajo, es necesario establecer una interfaz gráfica que permita al usuario enviar la información correspondiente a las coordenadas espaciales de dicho punto, el valor de las constantes empleadas en el método para la navegación, la instrucción de inicio del desplazamiento y la instrucción para la detención del mismo. A su vez es necesario identificar la naturaleza de los datos enviados por el móvil a fin de ser vinculados con el comportamiento del vehículo mientras este se desplaza durante el cumplimiento de una tarea establecida, mediante una representación gráfica del desplazamiento y de la matriz encargada de realizar la reconstrucción del entorno.

8.3 Envío de los comandos de control.

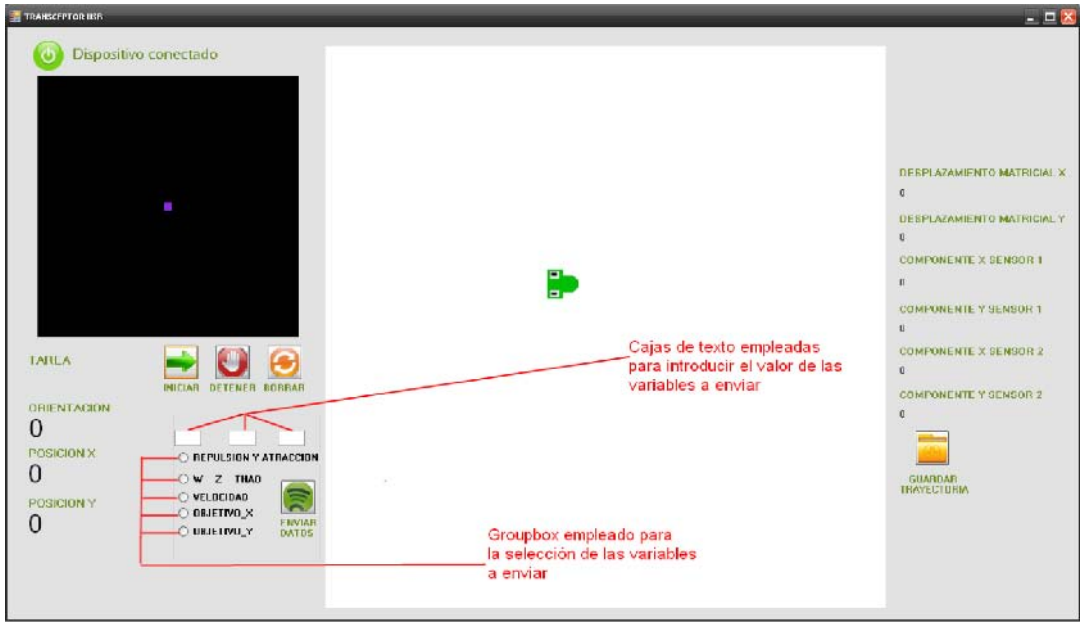
Para establecer la comunicación bidireccional entre el PC y el prototipo, la aplicación desarrollada cuenta con un gestor de comunicaciones conocido como *usbapi* incluido en el programa principal encargado de gestionar la comunicación USB con el PIC18f2550 haciendo uso de la librería entregada por MICROCHIP para tal fin. Esta aplicación secundaria permite identificar la presencia de módulos para la recepción y transmisión de datos vía USB conectados al computador en uso, enviar y recibir paquetes de datos vía puerto USB, entre otras funciones. Durante cada proceso llevado a cabo por el vehículo es necesario que un módulo de recepción y transmisión de datos se encuentre conectado al equipo encargado de presentar la interfaz gráfica al usuario, para garantizar este hecho, se hace uso de la función *identificación* incluida en *usbapi*, mediante la implementación de un temporizador encargado de establecer el tiempo de sondeo de conexión. La función *identificación* ha sido construida de manera tal que lleva a cabo la búsqueda de dispositivos identificados con dos variables preestablecidas (*product_id y vendor_id*) que identifican a cada equipo desarrollado empleando tecnología MICROCHIP. Si al culminar la búsqueda, la función determina que al menos un módulo transceptor está vinculado al equipo, el programa informa al usuario que el dispositivo se encuentra conectado o en caso contrario establece que no hay ningún dispositivo conectado. Esta información se muestra mediante una caja de texto y un indicador visual que cambia de color según el estado de la conexión lo amerite tal y como se indica en las figura 46a y 46b.

Previamente al inicio de cada recorrido, es necesario que el vehículo reciba la información correspondiente a las coordenadas x y y del punto objetivo, con el fin de establecer la dirección de viaje y la distancia comprendida hasta dicho punto, así como también, es necesario establecer el valor de la velocidad máxima que el

ω


Sin conexión
Caja de texto para estado de conexión
Indicador visual para la conexión
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W Z THAU
VELOCIDAD
OBJETIVO_X
OBJETIVO Y
ENVIAR DATOS
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA


Dispositivo conectado
Caja de texto para estado de conexión
Indicador visual para la conexión
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W Z THAU
VELOCIDAD
OBJETIVO_X
OBJETIVO_Y
ENVIAR DATOS
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA

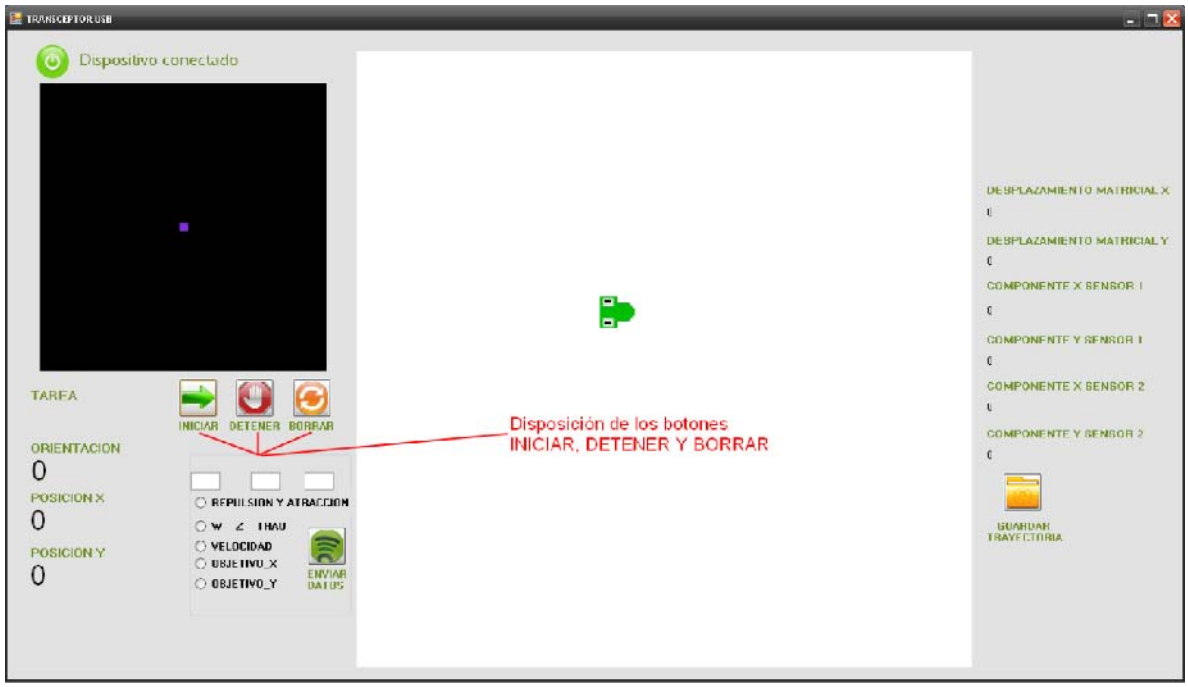

TRANSCEPTOR BSR
Dispositivo conectado
TARLA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W Z THAO
VELOCIDAD
OBJETIVO_X
OBJETIVO_Y
ENVIAR DATOS
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA
Cajas de texto empleadas para introducir el valor de las variables a enviar
Groupbox empleado para la selección de las variables a enviar

comando de control iniciar recorrido (0x08, 0x09, 0x0A, 0x0B) y la función denominada *ENVIOPIC* incluida en la aplicación que controla las comunicaciones USB con el micro-controlador es invocada para realizar la tarea requerida. Esta función a su vez, emplea la subrutina identificada como *SendPacket* para realizar la transferencia, una vez que dentro de la misma se han especificado la longitud de la trama a enviar (4 bytes en este caso) y el retraso definido entre transferencia de trama (1ms durante cada comunicación). Enviado el comando de control, el código activa la operación de un temporizador establecido en 500ms, encargado de gestionar el almacenamiento de los puntos que constituyen la trayectoria efectuada por el vehículo en un vector de dos dimensiones que permite almacenar hasta 2000 datos en cada una de ellas.

VARIABLES A ENVIAR	VALOR BYTE 1	VALOR BYTE 2	VALOR BYTE 3	VALOR BYTE 4	Valor Caja de texto 1	Valor Caja de Texto 2	Valor Caja de Texto 3
Repulsión y Atracción	0x00	Valor entero constante de repulsión	Valor entero constante de atracción	0x2A	Valor constante de repulsión	Valor constante de atracción	XX
ω, *z, thao*	Valor variable ω	Valor variable *z*	Valor variable *thao*	0x2B	Valor variable ω	Valor variable z	Valor variable thao
Velocidad	0x00	Valor entero variable velocidad	0x00	0xEF	Valor variable velocidad	XX	XX
Objetivo X	0x00	Byte más significativo del punto objetivo x	Byte menos significativo del punto objetivo x	0xCD	Valor del punto objetivo x (entero de 16 bits con signo)	XX	XX
Objetivo Y	0x00	Byte más significativo del punto objetivo y	Byte menos significativo del punto objetivo y	0xDE	Valor del punto objetivo y (entero de 16 bits con signo)	XX	XX

Tabla 9. En la cual se muestra la asignación que corresponde a cada byte, según la variable que se envía.

La activación del botón *DETENER* genera el envío de una cadena de datos hexadecimales correspondientes al comando de control encargado de interrumpir la operación del vehículo (0x08, 0x09, 0x0A, 0x0B), empleando para tal fin las funciones *ENVIOPIC* y *SendPacket* descritas anteriormente. No obstante, la


TRANSCEPTOR USB
Dispositivo conectado
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W
VELOCIDAD
OBJETIVO_X
OBJETIVO_Y
ENVIAR DATOS
Disposición de los botones
INICIAR, DETENER Y BORRAR
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA

móvil permanece constante en 22 bytes. Recibidos la totalidad de los datos, la función *LECTURAPIC* se encarga de retornar un vector de longitud igual a la trama definida, en el cual cada posición corresponde a uno de los datos enviados por el vehículo. El significado de los diferentes bytes recibidos está relacionado con su posición en la trama de recepción tal y como se describe en la tabla 10.

POSICION DEL BYTE EN LA TRAMA DE RECEPCION	VARIABLE RECIBIDA
1	Tarea en ejecución
2	Celda correspondiente a la distancia medida por el sensor 1 (componente x)
3	Celda correspondiente a la distancia medida por el sensor 1 (componente y)
4	Celda correspondiente a la distancia medida por el sensor 2 (componente x)
5	Celda correspondiente a la distancia medida por el sensor 2 (componente y)
6	Sentido de giro
7	Variable de monitoreo
8	Desplazamiento de la matriz en la dirección x
9	Desplazamiento de la matriz en la dirección y
10	Primer byte de la posición del vehículo en la dirección x
11	Segundo byte de la posición del vehículo en la dirección x
12	Tercer byte de la posición del vehículo en la dirección x
13	Cuarto byte de la posición del vehículo en la dirección x
14	Primer byte de la posición del vehículo en la dirección y
15	Segundo byte de la posición del vehículo en la dirección y
16	Tercer byte de la posición del vehículo en la dirección y
17	Cuarto byte de la posición del vehículo en la dirección y
18	Primer byte de la orientación del vehículo
19	Segundo byte de la orientación del vehículo
20	Tercer byte de la orientación del vehículo
21	Cuarto byte de la orientación del vehículo
22	Final de la trama

Tabla 10. Significado de los diferentes bytes recibidos, relacionado con la posición en la trama de recepción.

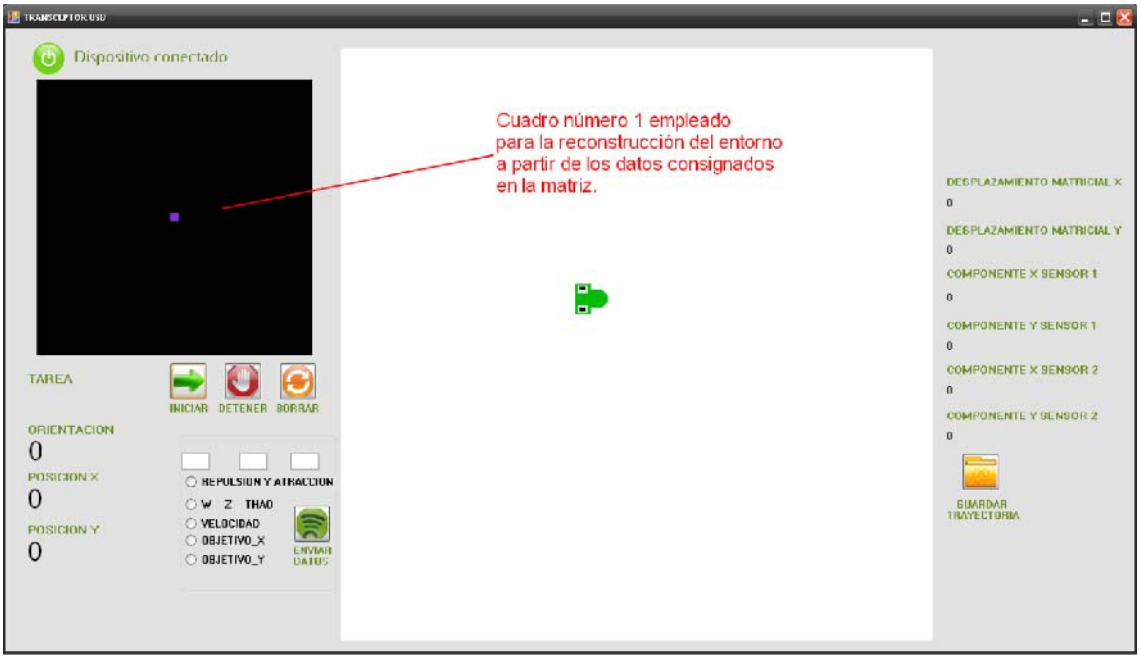
El primer byte de la trama recibida corresponde a la tarea que se encuentra en ejecución en el vehículo durante la transmisión de los datos, de manera tal que su valor hexadecimal representa una serie de nueve comportamientos distintos descritos en la tabla 11. Las posiciones 2, 3, 4 y 5 de la trama son empleadas para incrementar los coeficientes de la matriz para la reconstrucción del entorno elaborada con el fin de representar el comportamiento sensorial del vehículo de manera tal que los bytes 2 y 3 representan las coordenadas x y y de la celda cuyo coeficiente debe ser incrementado como resultado de la medición de distancia hasta el obstáculo más próximo realizada por el sensor 1, en tanto que los bytes 4 y 5 cumplen con una labor similar asociada al comportamiento del sensor 2.

VALOR HEXADECIMAL VARIABLE TAREA	TAREA EN EJECUCION
0x07	Navegación empleando *Virtual Forced Field Method*
0x08	Vehículo en espera de instrucciones
0x09	Navegación empleando *Wall Following Method*
0x01	Recepción del comando para inicio de recorrido
0x02	Recepción de la componente x del punto objetivo
0x03	Recepción de la componente y del punto objetivo
0x04	Recepción del valor para la velocidad máxima de desplazamiento
0x05	Recepción de los valores correspondientes a las constantes de repulsión y atracción
0x06	Recepción de los valores correspondientes a las constantes ω, *z, thao*

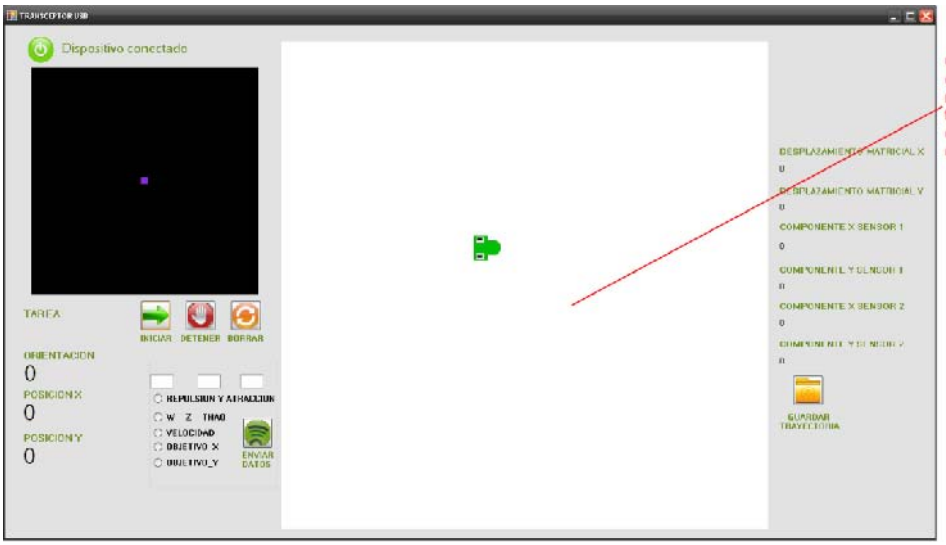
Tabla 11. Donde se muestran los nueve comportamientos para el valor hexadecimal del primer byte de la trama recibida que corresponde a la tarea que se encuentra en ejecución.

La variable asociada a la posición número 6 de la trama corresponde al sentido de giro actual del vehículo en tanto que el byte número 7 fue empleado como variable de monitoreo durante la programación del prototipo y el acople del mismo a la interfaz con el PC. Los desplazamientos matriciales representados por los bytes 8 y 9, corresponden a la cantidad de celdas que el vehículo ha recorrido desde la última transmisión recibida, en las direcciones x (byte 8) y y (byte 9). Las variables de desplazamiento matricial son de tipo entero de 8 bits con signo y son empleadas de manera tal que cuando su valor es diferente de cero, una rutina de corrimiento de matriz denominada *mover_matriz,* similar a la implementada en el módulo de navegación del prototipo es puesta en marcha con el fin de garantizar la similitud entre el funcionamiento de este último y el funcionamiento de la aplicación desarrollada.

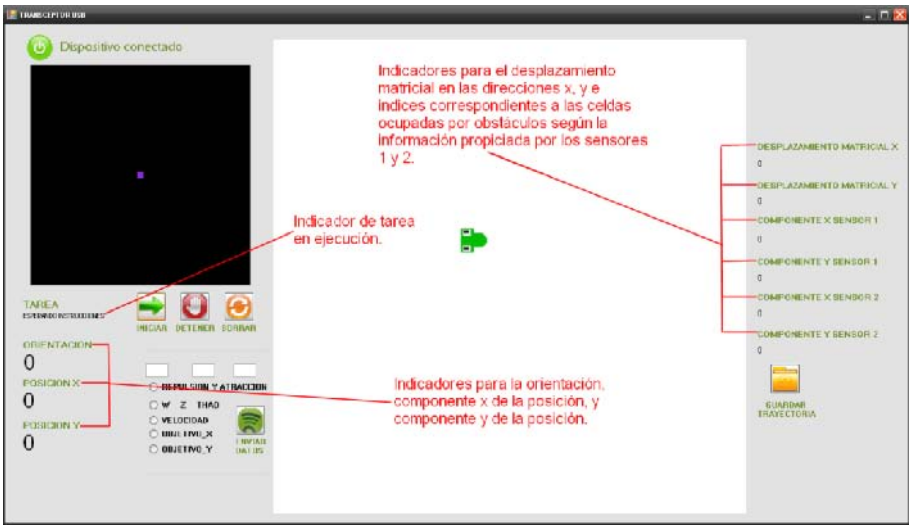
Las variables que representan la posición del móvil en cada componente del espacio de trabajo (x, y) son de tipo flotante de 32 bits y su valor resulta de la reconstrucción de un conjunto de 4 bytes recibidos por el PC para cada componente empleando el estándar IEEE-754. La componente x de la posición está representada por los bytes 10, 11, 12 y 13 de la trama de recepción, en tanto que la componente y debe ser reconstruida a partir de los valores consignados en las posiciones 14, 15, 16 y 17. Los bytes correspondientes a las posiciones de la trama 18, 19, 20 y 21 almacenan la información correspondiente a la variable de tipo flotante de 32 bits cuyo valor representa la orientación actual del móvil durante cada transmisión y deben ser empleados para reconstruir su valor de manera similar a lo expresado para la posición del vehículo. Finalmente, el byte número 22 de la trama es empleado para validar la veracidad de la información consignada en el buffer de recepción de manera tal que la totalidad de los datos

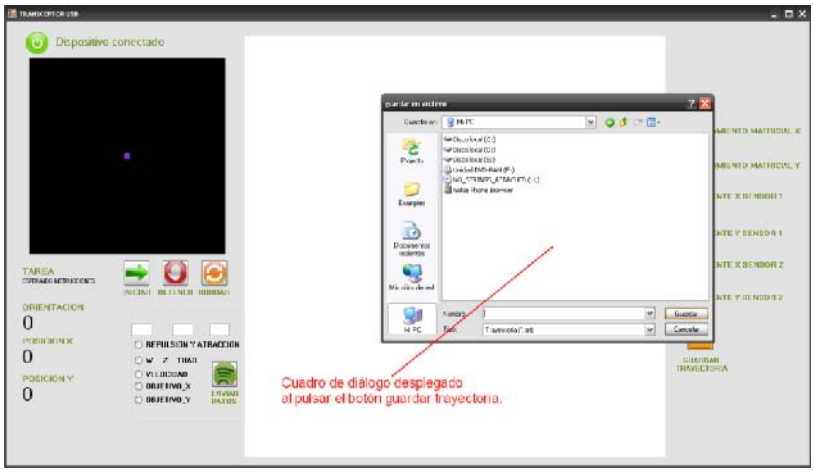

Dispositivo conectado
Cuadro número 1 empleado
para la reconstrucción del entorno
a partir de los datos consignados
en la matriz.
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W Z THAO
VELOCIDAD
OBJETIVO_X
OBJETIVO_Y
ENVIAR DATOS
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA

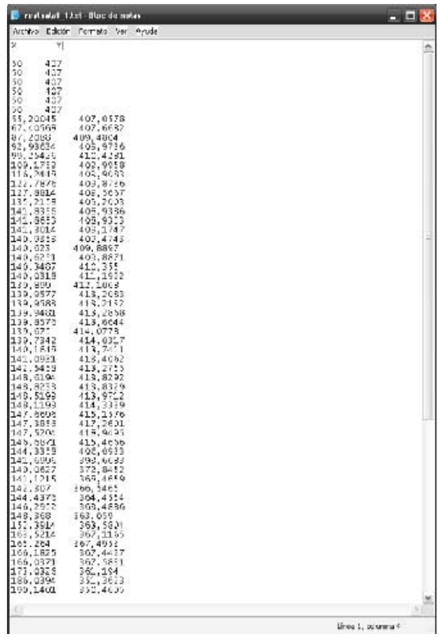

TRANSCEPTOR USB
Dispositivo conectado
Representación típica de un entorno a partir de los datos consignados en la matriz de coeficientes de certeza.
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
W Z THAO
VELOCIDAD
OBJETIVO_X
OBJETIVO_Y
ENVIAR DATOS
DESPLAZAMIENTO MATRICIAL X
0
DESPLAZAMIENTO MATRICIAL Y
0
COMPONENTE X SENSOR 1
0
COMPONENTE Y SENSOR 1
0
COMPONENTE X SENSOR 2
0
COMPONENTE Y SENSOR 2
0
GUARDAR TRAYECTORIA



Cuadro de imagen 2 empleado para la representación de la trayectoria descrita por el móvil durante su desplazamiento.


Dispositivo conectado
Indicadores para el desplazamiento matricial en las direcciones x, y e índices correspondientes a las celdas ocupadas por obstáculos según la información propiciada por los sensores 1 y 2.
Indicador de tarea en ejecución.
Indicadores para la orientación, componente x de la posición, y componente y de la posición.
TAREA
INICIAR
DETENER
BORRAR
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
VELOCIDAD
DESPLAZAMIENTO MATRICIAL X
DESPLAZAMIENTO MATRICIAL Y
COMPONENTE X SENSOR 1
COMPONENTE Y SENSOR 1
COMPONENTE X SENSOR 2
COMPONENTE Y SENSOR 2
GUARDAR TRAYECTORIA


Dispositivo conectado
TAREA
ORIENTACION
0
POSICION X
0
POSICION Y
0
REPULSION Y ATRACCION
GUARDAR TRAYECTORIA
Cuadro de diálogo desplegado al pulsar el botón guardar trayectoria.



resultado1_1.txt - Bloc de notas

Archivo Edición Formato Ver Ayuda

X	Y
50	407
50	407
50	407
50	407
50	407
50	407
55,20045	407,0578
67,60569	407,6682
87,2088	409,4804
92,93634	405,9716
99,25426	411,4281
109,1752	402,9958
116,2449	405,9083
122,7876	403,8716
127,8814	408,5637
135,2158	405,2003
141,8355	405,9336
141,8653	408,9323
141,3014	405,1747
140,9858	402,4743
140,623	409,8897
140,6251	403,8871
140,3407	412,355
140,0318	411,1922
139,899	412,1068
139,9577	413,2083
139,9588	413,2132
139,9481	413,2858
139,8573	413,6644
139,672	414,0778
139,7342	414,0317
140,1649	413,7411
141,0981	413,4062
142,5419	413,2755
148,6194	413,8292
148,8233	413,8329
148,5198	413,9712
148,1198	414,3319
147,8608	415,1376
147,3853	417,2601
147,5208	418,9425
145,6671	415,4656
144,3319	408,0513
141,6906	398,6683
140,0627	372,8452
141,1215	368,4659
142,307	366,3465
144,4373	364,4314
146,2912	363,4886
148,368	363,059
152,3914	363,5801
163,5214	367,1165
165,284	367,4953
166,1825	367,4437
166,0371	367,5851
173,0325	362,194
186,0394	352,3623
190,1401	352,4635

Línea 1, columna 4

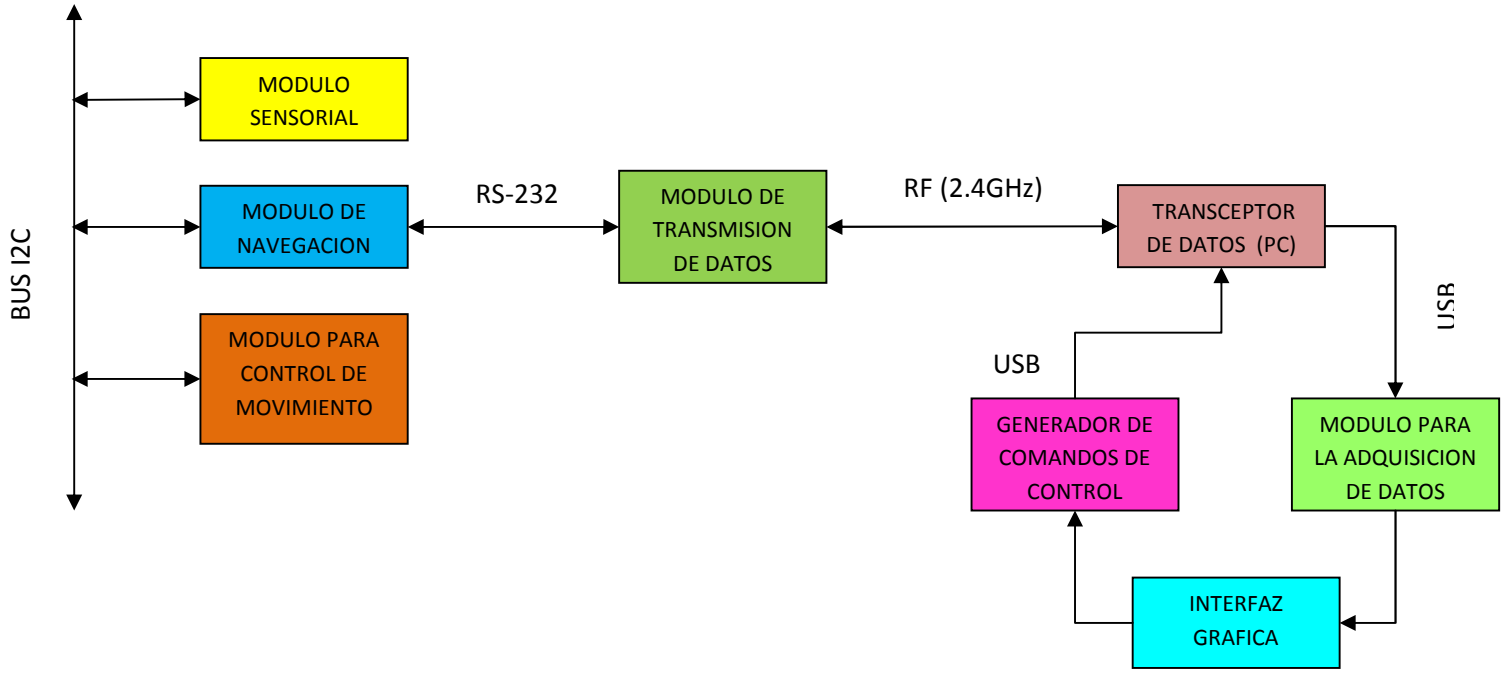


Figura 54. Diagrama de bloques que bosqueja el funcionamiento del sistema vehículo-PC.

9. OPTIMIZACION DEL SISTEMA DE NAVEGACION PARA EL VEHICULO AUTONOMO

El proceso de optimización del sistema de navegación implementado está constituido en su mayoría por la búsqueda de la configuración de las variables incluidas en el modelo matemático del mismo, que permita al vehículo desempeñar una labor determinada bajo una serie de parámetros establecidos previamente. Dado que el espacio de búsqueda está comprendido por todo el conjunto de combinaciones posibles que pueden asumir las variables del sistema, su tamaño representa un conjunto de diversas posibilidades de gran envergadura. Por tanto la búsqueda de la configuración ideal es un proceso que puede consumir una gran cantidad de recursos computacionales y operativos, incrementando la magnitud del problema a resolver.

Para lidiar con esta situación, los algoritmos genéticos han sido empleado durante esta investigación, dadas sus características favorables durante procesos de optimización relacionados con la búsqueda de configuraciones en espacios de posibles soluciones delimitados. A partir de la descripción del método de navegación empleado durante la implementación del sistema, pueden apreciarse una serie de variables que determinan el comportamiento del vehículo cuando se desplaza desde un punto origen hasta un punto objetivo en un entorno constituido por una serie de obstáculos. Este conjunto de variables está conformado por la constante de repulsión, la constante de atracción, el factor de amortiguamiento (ω), el factor de reducción para el control de velocidad (z) y la constante de tiempo empleada por el filtro pasa bajos incluido en el sistema (*thao*). El valor que cada uno de estos componentes puede asumir está contenido en un intervalo preestablecido para cada caso en particular.

La búsqueda del conjunto de variables solución al problema de la optimización del sistema de navegación requiere la creación de un entorno de simulación computarizada a fin de disminuir los requerimientos operativos durante la puesta en marcha del prototipo para cada conjunto de posibles valores que serán asignados a las variables del sistema. A continuación se lleva a cabo la descripción de la aplicación desarrollada para tal fin.

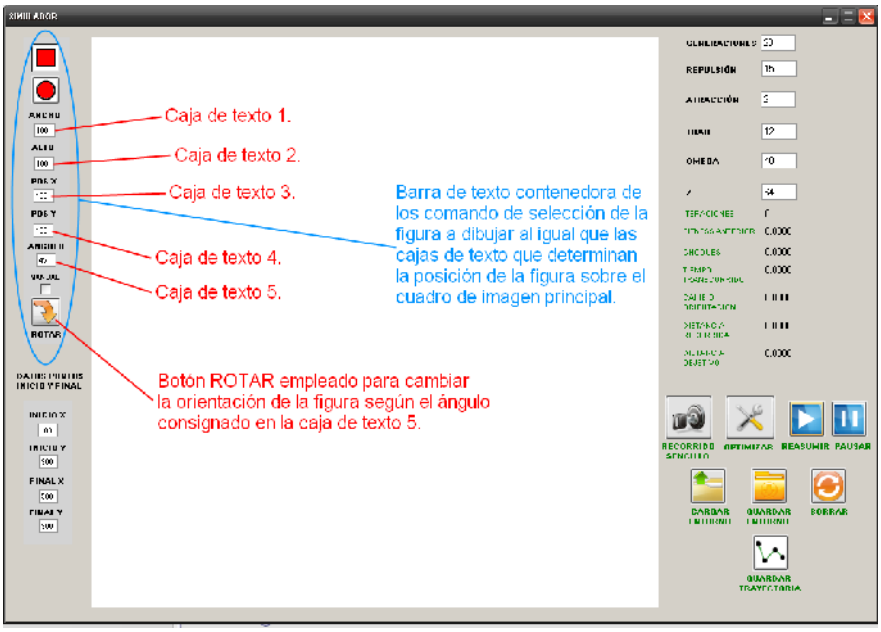
9.1 Entorno de simulación para la optimización del sistema de navegación.

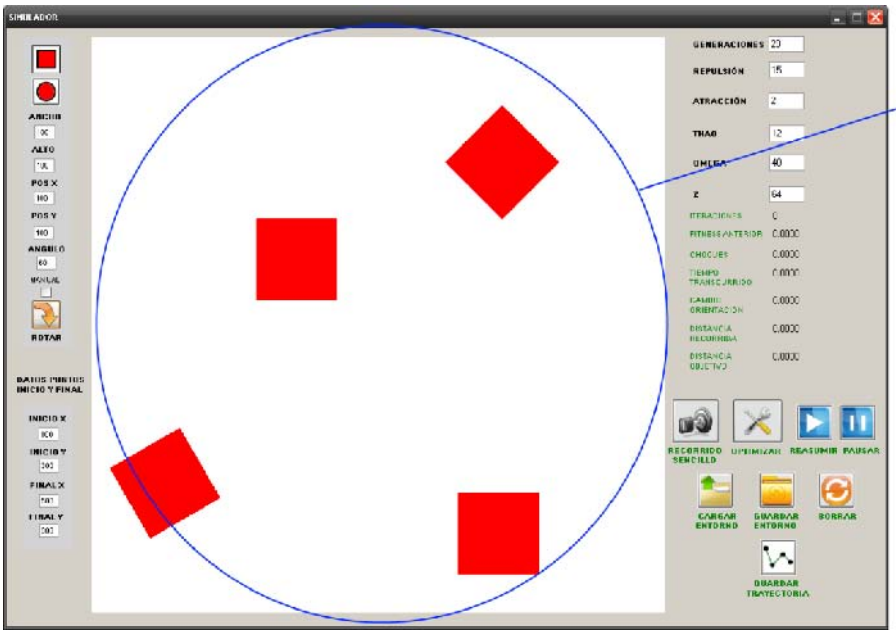
La recopilación de los datos correspondientes al comportamiento del móvil cuando se desplaza hacia un punto objetivo empleado la configuración asociada al conjunto de variables que determinan el accionar del sistema de navegación, constituye un proceso cuyos costos operativos son elevados dado que puesto en

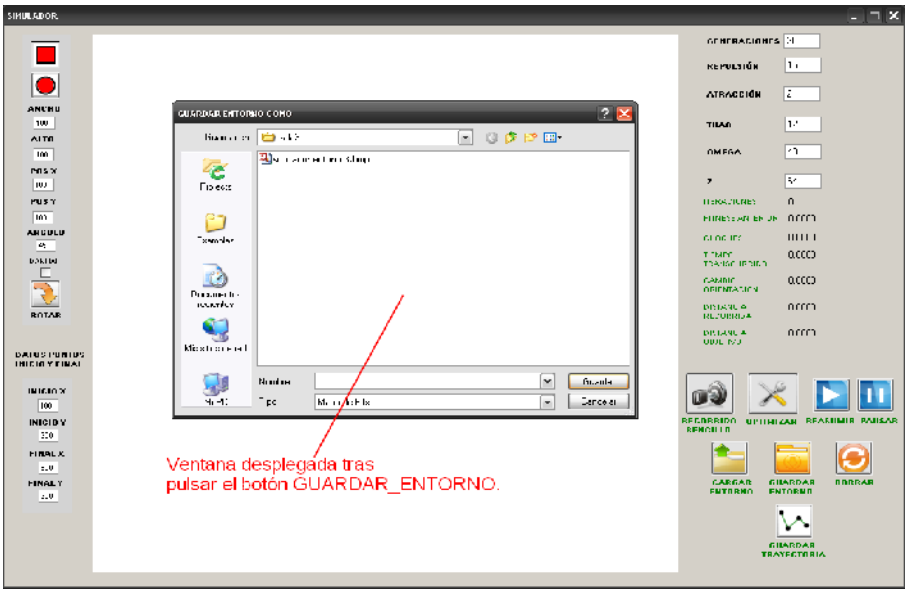
marcha en tiempo real, sería necesario ejecutar un desplazamiento por cada configuración empleada. Con ello, recursos como la batería del sistema se agotarían rápidamente incrementando el tiempo total de ejecución para el proceso de búsqueda de la configuración más adecuada para el prototipo. Para mitigar los efectos de la situación descrita, se ha desarrollado una aplicación encargada de realizar la simulación del comportamiento del sistema en general, cuando interactúa con un entorno cuyas características pueden ser esquematizadas en la misma aplicación. El lenguaje de programación empleado corresponde a C# 2008 al igual que ocurre con la aplicación destinada a la interfaz creada entre el prototipo y el usuario final mediante el PC.

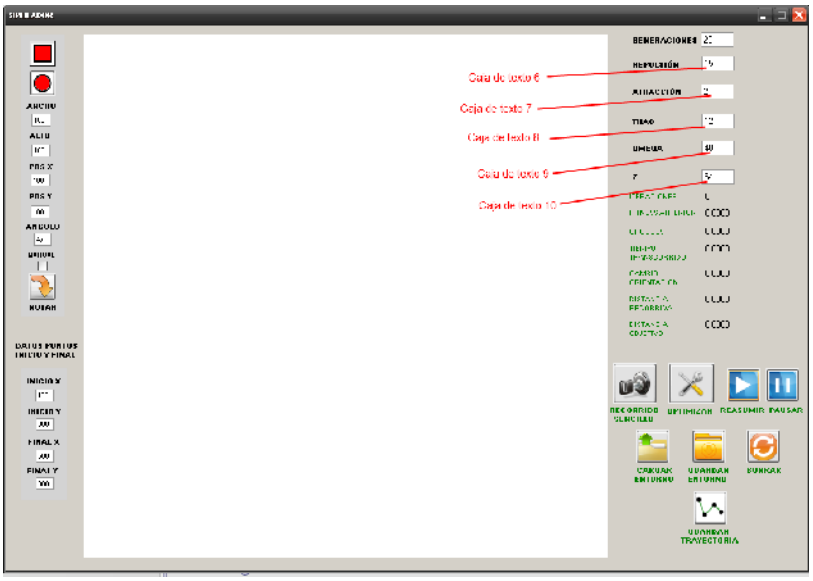
9.2 Generación de un entorno de trabajo.

Para determinar las características del entorno de trabajo, la aplicación dispone de un cuadro de imagen con dimensiones correspondientes a los 707x707 pixeles, con cada pixel representando un centímetro del entorno real de trabajo. En su interior, es posible representar obstáculos empleando los elementos incluidos en la barra de dibujo designada para tal fin. Dadas las características sensoriales del móvil ha sido posible establecer que los obstáculos presentes en un determinado ambiente pueden ser esquematizados con claridad empleando formas circulares y rectangulares únicamente. Posiblemente, sistemas sensoriales con mayor sensibilidad, requieran representaciones espaciales elaboradas con formas geométricas de mayor complejidad. La barra de dibujo elaborada permite por tanto elaborar rectángulos y elipses de dimensiones específicas que son seleccionados empleando los botones incluidos en la barra de dibujo que poseen la imagen de cada figura, siendo su posición en la cuadro de imagen, determinada mediante dos controles independientes. El primero se encuentra consignado en la barra de dibujo y está conformado por las cajas de texto 1 y 2 que determinan las componentes x y y respectivamente de la posición de la figura a elaborar. El segundo comando para establecer la posición de la figura está determinado por la posición del mouse sobre el cuadro de imagen de manera tal que al realizar un click izquierdo sobre el mismo, la figura seleccionada será dibujada en la posición establecida con las dimensiones consignadas en las cajas de texto 3 y 4. La selección del comando que determina la posición se realiza a partir de la caja de selección 1. Finalmente, la orientación de la figura a dibujar puede ser definida mediante el controlador constituido por la caja de texto 5, en la cual el valor del ángulo de orientación debe ser consignado para posteriormente pulsar el botón denominado *ROTAR* y así realizar el giro especificado. Si el usuario desea eliminar las figuras presentes en la caja de imagen, cuenta para ello con el botón *BORRAR*, que tras ser accionado elimina cualquier figura o imagen presente en el


Caja de texto 1.
Caja de texto 2.
Caja de texto 3.
Caja de texto 4.
Caja de texto 5.
Barra de texto contenedora de los comando de selección de la figura a dibujar al igual que las cajas de texto que determinan la posición de la figura sobre el cuadro de imagen principal.
Boton ROTAR empleado para cambiar la orientación de la figura según el ángulo consignado en la caja de texto 5.


Entorno creado mediante la aplicación.


SIMULADOR
GUARDAR ENTORNO COMO
Ventana desplegada tras
pulsar el botón GUARDAR_ENTORNO.



VARIABLE	NUMERO DE BITS EMPLEADOS PARA SU REPRESENTACION	INTERVALO DE VALORES QUE PUEDE ASUMIR CADA VARIABLE	INTERVALO DE VALORES PARA LA REPRESENTACION BINARIA
Constante de repulsión	8	1-256	0-255
Constante de atracción	8	1-256	0-255
Constante de amortiguamiento (ω)	6	0-1	0-63
Factor de reducción de la velocidad (z)	6	0-0.5	0-63
Constante de tiempo para el filtro pasa bajos (*thao*)	5	0.4-1	0-31

Las relaciones establecidas para la conversión de los datos ingresados en datos reales para la operación del sistema de navegación se muestran en (69) para cada variable en específico.

$$C_R = (valor\ caja\ de\ texto\ 6) + 1$$

$$C_A = (valor\ caja\ de\ texto\ 7) + 1$$

$$thao = 0.4 + (valor\ caja\ de\ texto\ 8) \square 0.01875 \quad (69)$$

$$\omega = (valor\ caja\ de\ texto\ 9) \square 0.015625$$

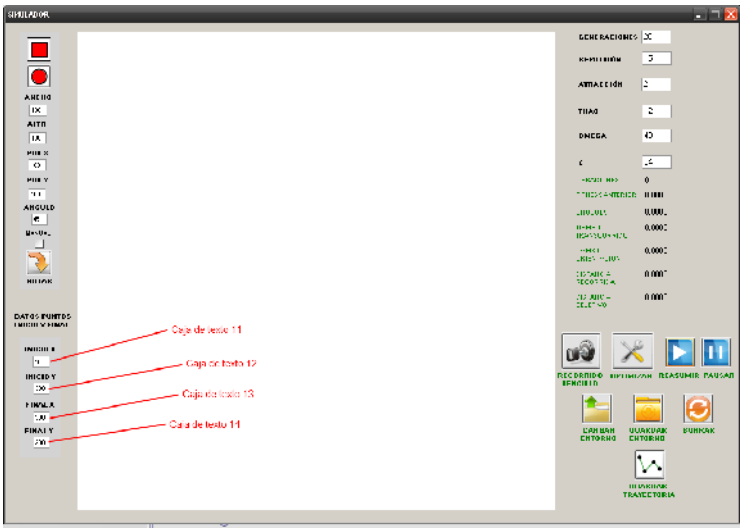
$$z = (valor\ caja\ de\ texto\ 10) \square 0.0078125$$

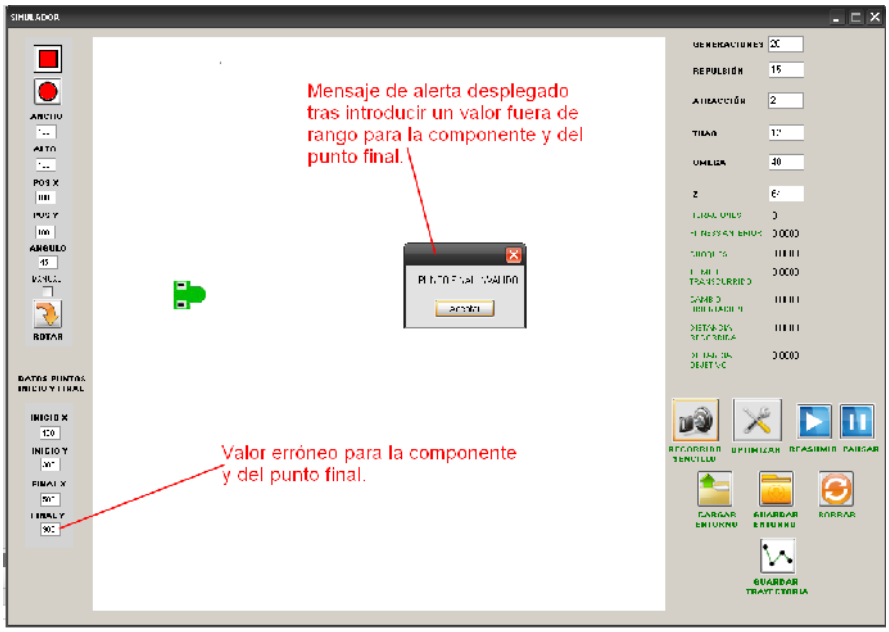
Donde:
C_R = Constante de repulsión del sistema.
C_A = Constante de atracción del sistema.
$thao$ = Constante de tiempo para el filtro pasa bajos.
ω = Constante de amortiguamiento del sistema.
z = Factor de reducción para la velocidad del desplazamiento.

Por defecto, la aplicación establece un conjunto de valores para permitir el funcionamiento del sistema de no introducirse nuevos elementos. Este conjunto se presenta en las cajas de texto correspondientes a las variables mencionadas al iniciar la aplicación.

9.4 Simulación del comportamiento del móvil dentro de un entorno establecido.

Antes de iniciar un recorrido es necesario establecer los puntos de inicio y objetivo del desplazamiento sobre el espacio de trabajo. Para ello existen dos cajas de texto enumeradas 11, 12, 13 y 14 en las que es posible consignar los datos referentes a la coordenada x del punto de inicio, la coordenada y del mismo punto y las coordenadas x y y del punto final respectivamente. Un conjunto de valores por defecto es asignado a las coordenadas de los dos puntos y se muestran sobre las cajas de texto correspondientes. Gráficamente, es posible apreciar la ubicación de los dos puntos sobre el cuadro de imagen empleado para la simulación del entorno. El punto inicio está identificado por una imagen a escala 1:1 del prototipo desarrollado siendo cada pixel de la imagen correspondiente a 1 centímetro de sus dimensiones reales. Por su parte el punto objetivo está representado por una


SIMULADOR
DATOS PUNTOS
INICIO Y FINAL
INICIO X
INICIO Y
FINAL X
FINAL Y
Caja de texto 11
Caja de texto 12
Caja de texto 13
Caja de texto 14
GENERACIONES
ANIMACIÓN
OMEGA
RESUMIR
PAUSAR


Mensaje de alerta desplegado tras introducir un valor fuera de rango para la componente y del punto final.
Valor erróneo para la componente y del punto final.

dirección determinada por la secuencia de movimientos del sonar y la orientación inmediata del móvil. Para garantizar que el primer pixel identificado como perteneciente a un obstáculo sea el más cercano al vehículo, el recorrido de reconocimiento se realiza de manera paralela al frente del móvil, iniciando el proceso sobre el vértice más cercano al mismo.

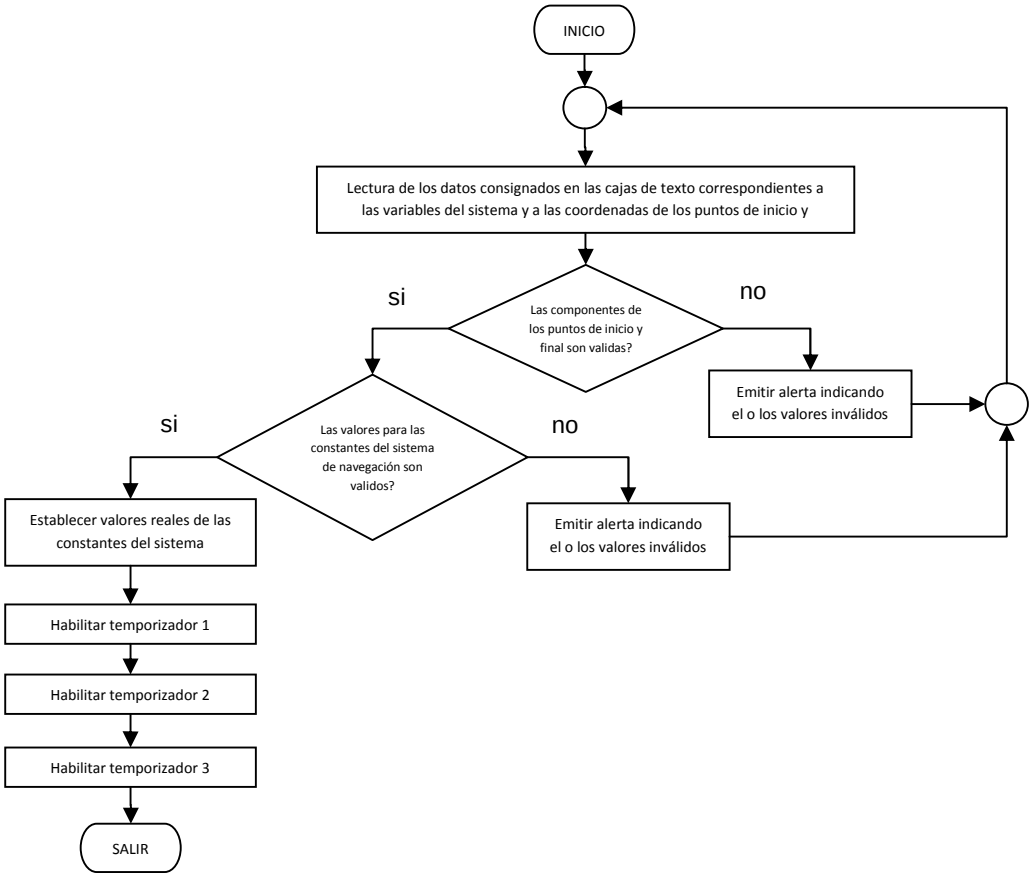


Figura 60. Diagrama de flujo que corresponde a la función *inicio*.

Cuando un obstáculo es identificado, la función almacena el valor de la distancia existente hasta el mismo para posteriormente incluir dicha información en la matriz para la reconstrucción del entorno necesaria para la operación del sistema de navegación VFF. El grupo de instrucciones contenido en *sensor* es ejecutado una vez por cada sensor implementado tanto en el dispositivo real como en el simulado, un total de 2 veces por cada búsqueda del sonar. Cuando la distancia obtenida tras la ejecución de la función tiene un valor diferente de cero, el cálculo de los índices de la celda correspondiente a dicho valor es ejecutado de manera tal que el coeficiente almacenado en la matriz para la reconstrucción del entorno ubicado en la celda establecida es incrementado en 1. El diagrama de flujo

correspondiente al proceso anteriormente descrito se encuentra contenido en la figura 61.

Las funciones necesarias para la simulación del proceso de navegación del vehículo y la presentación gráfica del desplazamiento son llevadas a cabo por el temporizador 2. En busca de minimizar las diferencias existentes entre el comportamiento real del móvil y su simulación, el tiempo de operación se ha establecido en 400ms, independizando de esta manera los procesos sensoriales y de navegación, sin afectar el funcionamiento general del sistema simulado. Al activarse su operación durante la función *inicio,* el temporizador ejecuta las instrucciones contenidas en la función *cálculos*, destinada a la determinación de la orientación y la velocidad que deben guiar el desplazamiento del vehículo con el fin de alcanzar el punto objetivo evadiendo los obstáculos presentes en el entorno. Para tal fin, el código implementado es muy similar al puesto en marcha en el prototipo real, basando su operación en el método de navegación conocido como VFF (*Virtual Force Field Method*). En primera instancia, se ejecuta la determinación de las componentes x y y de la fuerza repulsiva total del sistema, a partir de la información consignada en la matriz para la reconstrucción del entorno cuyas dimensiones corresponden a 33x33 celdas, cada una de ellas representando una sección del espacio de 10x10cm del área, es decir que en cada celda está modelado el espacio correspondiente a 10x10 pixeles de la imagen contenida en el cuadro de imagen de la aplicación.

Para determinar estos valores se emplea la relación establecida en (71) tal y como se llevó a cabo durante la implementación del sistema de navegación en el prototipo real.

$$\boldsymbol{F}_{ij} = \frac{C_R C(i,j)}{d^2(i,j)} \left[\frac{x_t - x_0}{d(i,j)} \hat{x} + \frac{y_t - y_0}{d(i,j)} \hat{y} \right] \qquad (71)$$

Donde:

F_{ij} = Fuerza de repulsión ejercida por la celda *i,j.*
C_R = Constante de repulsión del sistema.
$C(i,j)$ = Valor del coeficiente de certeza consignado en la celda *i,j.*
$d(i,j)$ = Distancia existente entre la celda central de la grilla y la celda *i,j.*
x_t = Coordenada x del punto objetivo.
y_t = Coordenada y del punto objetivo.
x_0 = Coordenada x de la posición actual del robot.
y_0 = Coordenada y de la posición actual del robot.

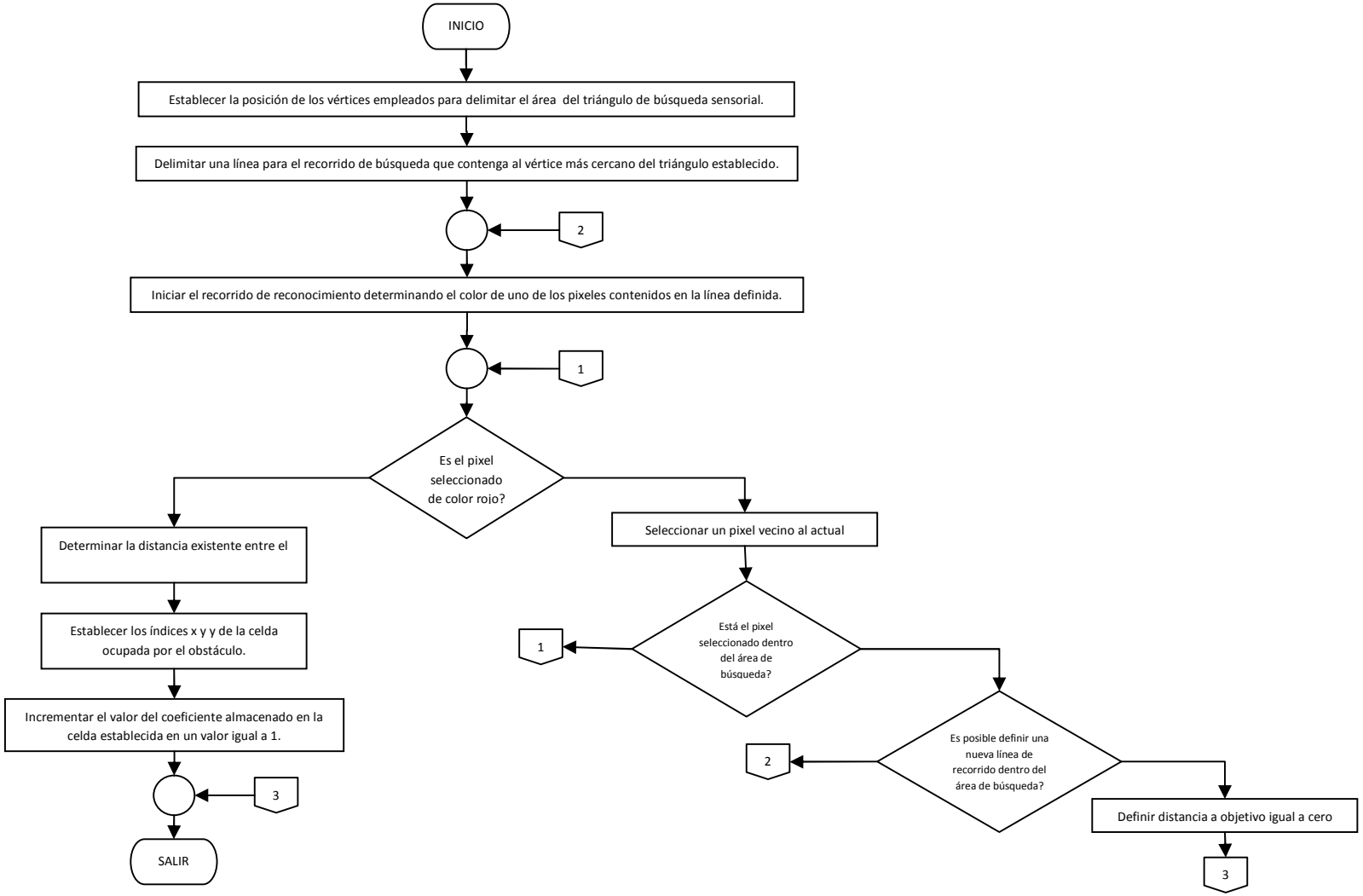


Figura 61. Diagrama de flujo correspondiente al comportamiento efectuado por los sensores en busca de obstáculos presentes en el entorno.

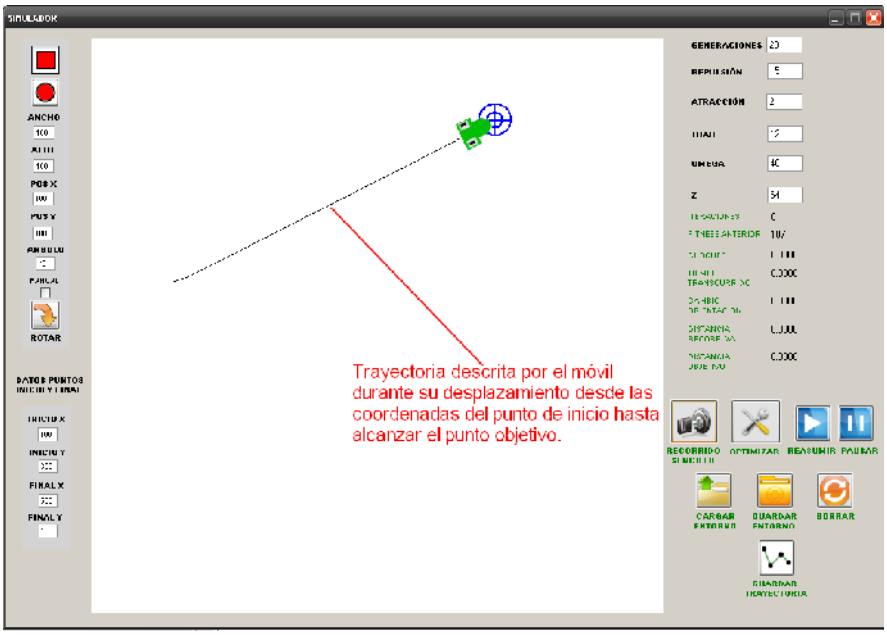
De manera similar, el código se encarga de determinar las componentes del vector fuerza atractiva total del sistema empleando las relaciones establecidas durante la implementación del prototipo real a partir de la relación consignada en (72).

$$F_t = F_{CT}\left[\frac{x_t - x_0}{d(t)}\hat{x} + \frac{y_t - y_0}{d(t)}\hat{y}\right] \qquad (72)$$

Donde:

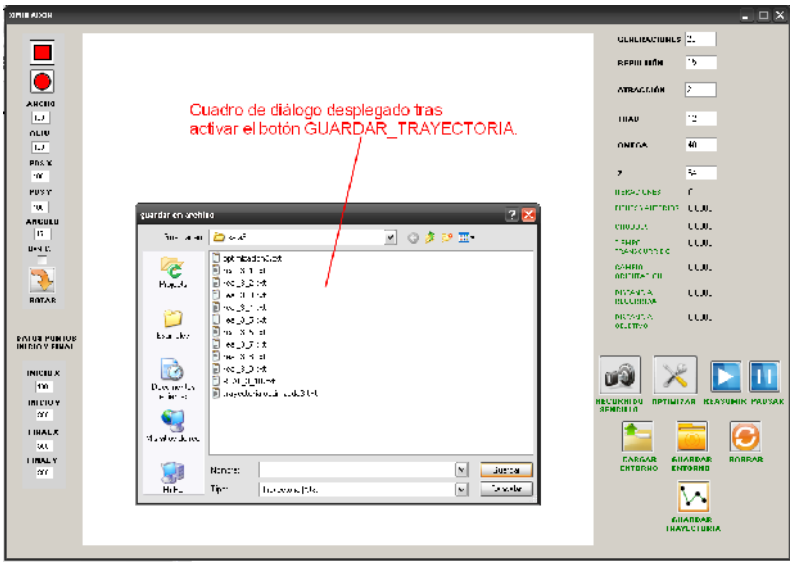
F_{ct} = Constante de fuerza de atracción al objetivo.
x_t, y_t = Coordenadas del punto objetivo.
$d(t)$ = Distancia existente entre el robot y el objetivo en un instante de tiempo t.

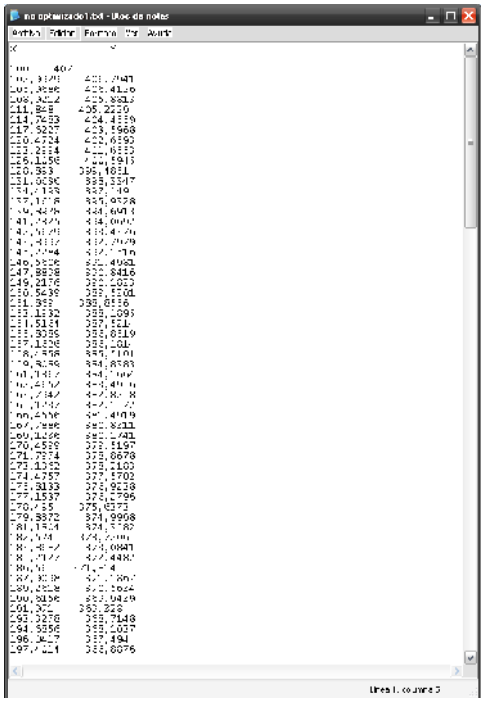
Determinados los valores de estas componentes, es necesario aplicar los controladores definidos en el sistema para establecer el grado de amortiguamiento de la fuerza repulsiva, la velocidad determinada según el controlador implementado y el retraso en el procesamiento de la orientación acorde a las características del filtro pasa bajos puesto en marcha para tal fin en el prototipo real. Las relaciones empleadas para este desarrollo coinciden en su totalidad con las mencionadas durante la implementación del sistema de navegación en el prototipo real y han sido omitidas por dicha razón. De manera similar, el código se encarga de monitorear el comportamiento del sistema de navegación en busca de situaciones determinadas como trampas para efectuar un cambio de método de navegación llevando a cabo el proceso mediante las relaciones establecidas por el método conocido como WFF (*Wall Fallowing Method*) sorteando los obstáculos presentes en el entorno evitando recorridos cíclicos dentro del mismo. La condición de evaluación empleada durante esta etapa del procedimiento establece la diferencia entre las orientaciones correspondientes al vector fuerza repulsiva y fuerza atractiva; cuando dicho valor supera los 90°, es controlador de navegación cambia al modo WFM, de lo contrario su operación continua de manera normal bajo los parámetros del método VFF. La fuerza repulsiva temporal creada para modificar la orientación del móvil mediante el método WFM resulta de sumar a la orientación del vector fuerza de repulsión un ángulo preestablecido de 145°, tal y como ocurre en el sistema de navegación empleado en el prototipo real. Como resultado de esta serie de procesos, la orientación que debe asumir el móvil en un instante de tiempo determinado es definida, al igual que la velocidad de desplazamiento para el móvil. Se hace necesaria entonces una representación gráfica del comportamiento anteriormente descrito. Para ello, la aplicación emplea las herramientas gráficas proporcionadas por el lenguaje empleado para la programación, haciendo uso de una imagen del prototipo real en escala 1:1


SIMULADOR
ANCHO
ROTAR
DATOS PUNTOS INICIAL Y FINAL
INICIAL X
INICIAL Y
FINAL X
FINAL Y
GENERACIONES
REPULSIÓN
ATRACCIÓN
Trayectoria descrita por el móvil durante su desplazamiento desde las coordenadas del punto de inicio hasta alcanzar el punto objetivo.
OPTIMIZAR
REANUDAR
PAUSAR
CARGAR ENTORNO
GUARDAR ENTORNO
BORRAR
GUARDAR TRAYECTORIA

Este procedimiento es llevado a cabo mediante las funciones *movimiento* y *mover_matriz* incluidas en el código principal. La primera emplea la información concerniente a las posiciones anterior y actual del móvil a fin de establecer el recorrido realizado durante el intervalo en consideración estableciendo la celda de la matriz en la que el centro del móvil se ubica. Cuando la celda de posición actual difiere de la celda ocupada en el intervalo de muestreo inmediatamente anterior, el código determina la cantidad de celdas que el vehículo ha recorrido en cada componente y convoca a la función *mover_matriz* con el fin de realizar el desplazamiento de los datos consignados en la matriz la cantidad de celdas definida en cada dirección.

La evaluación de la distancia existente entre la posición actual del móvil y el punto objetivo del desplazamiento permite determinar si el recorrido ha finalizado o no. Cuando el valor de dicha distancia es inferior a 35cm, la operación de los temporizadores es suspendida y los procesos de navegación y representación gráfica del desplazamiento se detienen. El procedimiento es llevado a cabo al inicio de la función *movimiento* y se ejecuta una vez por cada intervalo de muestreo. Al finalizar el recorrido, la aplicación habilita el botón denominado *GUARDAR_TRAYECTORIA*, destinado a almacenar los datos correspondientes a las componentes de la posición del móvil durante cada intervalo de muestreo en un archivo de texto con extensión .txt. Tras ser accionado, el código genera un cuadro de diálogo en el que deben definirse el nombre del archivo y la ubicación en la que debe ser almacenado. Los datos del vector de posición empleado para la construcción gráfica de la trayectoria son transferidos al archivo de texto en columnas separadas para cada componente. La ventana desplegada y un archivo de texto generado se muestran en las figuras 63a y 63b.


Cuadro de diálogo desplegado tras
activar el botón GUARDAR_TRAYECTORIA.
guardar en archivo
Nombre:
Tipo:
Guardar
Cancelar
ANCHO
ALTO
ANGULO
ROTAR
INICIO X
INICIO Y
FINAL X
FINAL Y
CALIBRACIONES
REPULSIÓN
ATRACCIÓN
OMEGA
RECORRIDO SENCILLO
OPTIMIZAR
REASUMIR
PAUSAR
CARGAR ENTORNO
GUARDAR ENTORNO
BORRAR
GUARDAR TRAYECTORIA


Archivo Edición Formato Ver Ayuda
Línea 1, columna 3

recorrido efectuado por el vehículo cumpla con una serie de parámetros establecidos para la evaluación del desempeño del móvil durante un desplazamiento efectuado bajo un conjunto de valores determinado. La aplicación de algoritmos genéticos como modelos de búsqueda de soluciones grupales sobre espacios de valores definidos, minimiza significativamente los costos operativos durante el proceso de optimización. Durante esta investigación, el proceso de optimización hace referencia entonces a la búsqueda de los valores correspondientes a las variables constante de repulsión, constante de atracción, constante de amortiguamiento (*ω*), factor de reducción para la velocidad (*z*) y constante de tiempo para el filtro pasa bajos (*thao*) que le permitan al móvil desplazarse hacia un punto objetivo evadiendo los obstáculos presentes en el entorno de manera que su comportamiento se apegue a una serie de parámetros establecidos previamente. A fin de llevar a cabo el proceso de búsqueda, es necesario establecer el esquema de representación de cada solución posible, o de cada individuo dentro de los términos relativos al algoritmo genético. La representación binaria surge como la posibilidad de mayor atractivo dado que simplifica la asignación de la longitud de cada rango de valores y facilita los procesos de cruce y mutación. La estructura del individuo empleada, conocida también como cromosoma, se presenta en la figura 64.

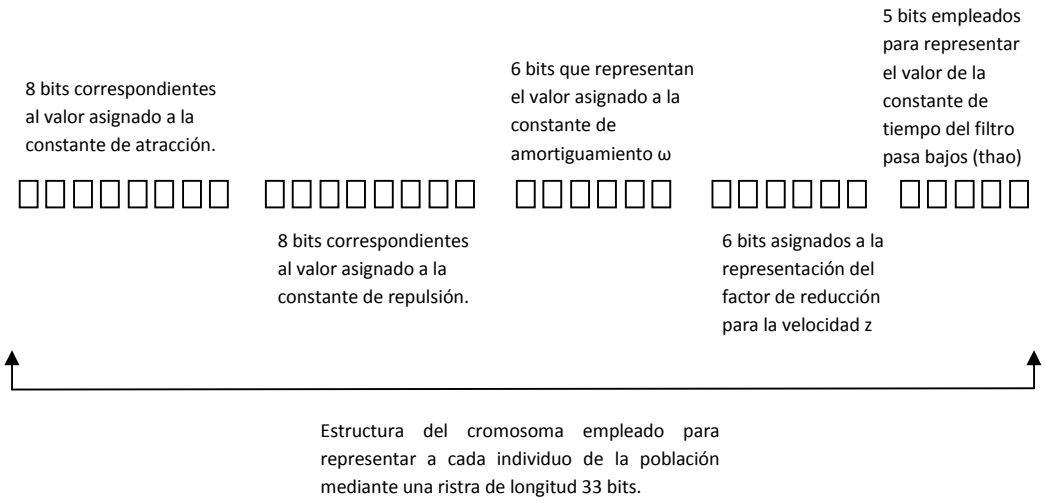
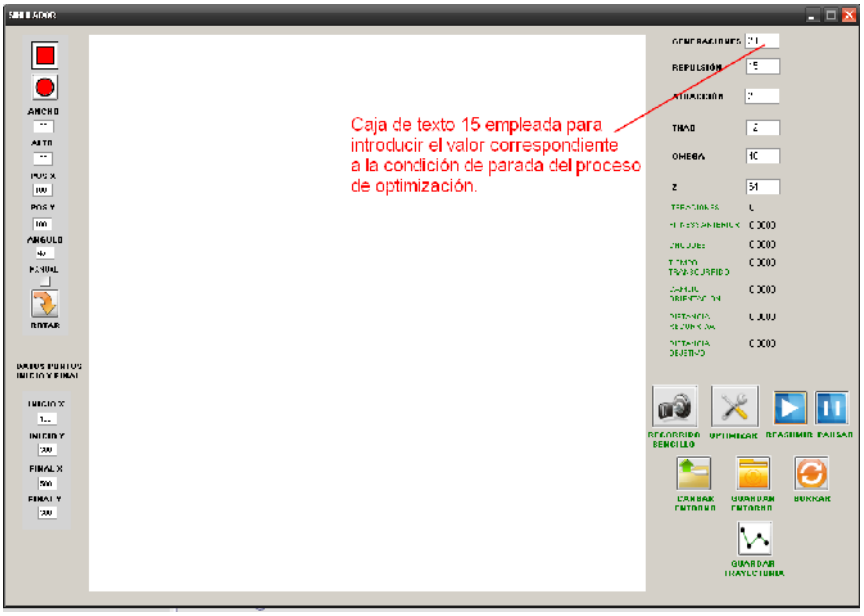


Figura 64. En la cual se observa el cromosoma del individuo.

La longitud de la cadena de bits empleada corresponde a 33 dando lugar a la generación de individuos con valores numéricos contenidos en el rango desde 0 hasta 2^{33}-1. Para almacenar los datos correspondientes a cada individuo se hace necesario emplear variables de tipo entero de 64 bits correspondientes al tipo *long* dentro del lenguaje de programación empleado. El tamaño de la población inicial empleada para la ejecución del algoritmo se ha determinado en 44 individuos dado que en la literatura consultada se afirma que una


Caja de texto 15 empleada para introducir el valor correspondiente a la condición de parada del proceso de optimización.
REPULSIÓN
THAO
OMEGA
Z
ANCHO
ALTO
POS X
POS Y
ANGULO
ROTAR
OPTIMIZAR
REANUDAR
PAUSAR
BORRAR

a fin de habilitar el recorrido del vehículo bajo los valores asignados, en un proceso similar al empleado durante la ejecución de un recorrido sencillo. Para evaluar la adaptación de cada individuo al sistema general es necesario definir el conjunto de parámetros operativos agrupado bajo la función de evaluación conocida como *Fitness* encargada de asignar a cada solución posible un valor operativo que representa el desempeño del vehículo durante el procedimiento. Los parámetros empleados corresponden al número de choques durante el recorrido, el tiempo empleado durante el desplazamiento, el valor promedio para el cambio de orientación del móvil durante dos intervalos de muestreo consecutivos, la distancia total recorrida y la distancia final hasta el punto objetivo. Los parámetros de evaluación son elegidos tras un análisis de las características óptimas para un desplazamiento en específico, considerando el comportamiento del prototipo implementado durante una tarea propuesta. Para construir la función de adaptación (*Fitness*), los parámetros considerados para la evaluación del desempeño son empleados de forma tal que su relación con el comportamiento deseado establece su posición dentro de la función; así, un incremento en el valor de un determinado parámetro que determine un valor de adaptación alto para la función de evaluación tendrá relación directa con el valor de la función en tanto que en caso contrario el valor representará una relación inversa. La relación final se presenta en (73).

$$fitness = \frac{100000000}{(100Ch + 10t_R + 10\Delta Or + D_R + D_O)^2} \qquad (73)$$

Donde:

$fitness$ = Valor que representa la adaptación de una posible solución al sistema.

Ch = Número total de choques ocurridos durante el recorrido.

t_R = Tiempo transcurrido durante el recorrido.

ΔOr = Cambio promedio en la orientación del móvil durante dos intervalos de muestreo consecutivos.

D_R = Distancia total recorrida durante el desplazamiento.

D_O = Distancia comprendida entre la posición final del móvil y el punto objetivo.

Dentro de la función, existen parámetros de mayor relevancia sobre el nivel de adaptación deseado siendo su valor multiplicado por una constante que les otorga un peso mayor sobre el valor total de la función. Finalmente, al elevar al cuadrado la suma de los parámetros establecidos se genera un efecto de representación más significativa entre una solución deseada y una que no se adapte de manera adecuada al problema propuesto. La recopilación de los valores asociados a los parámetros necesarios para la operación de la función

de adaptación se lleva a cabo al finalizar los procedimientos de navegación y representación gráfica del móvil, contenidos en las funciones *cálculos* y *movimiento.*

El número de choques presentados durante la tarea ejecutada corresponde a la cantidad de pixeles de color rojo que han sido ocupados por la imagen representativa del móvil durante cada intervalo de muestreo. Para establecer su número, la función denominada *choques* es implementada, siendo ejecutada unas vez durante cada intervalo al final de la función *cálculos*. La determinación de la cantidad de pixeles ocupados por la imagen se lleva a cabo empleando un algoritmo de búsqueda encargado de recorrer un rectángulo de dimensiones similares a la imagen de representación, superpuesto sobre la misma, de manera tal que el efecto conseguido corresponde a la simulación del espacio ocupado por el móvil real. El valor determinado durante cada intervalo se adiciona al valor correspondiente al intervalo anterior, consolidando un número de choques total apreciable al final de cada recorrido.

El tiempo transcurrido durante la ejecución de cada recorrido total, es obtenido a partir de la información de fecha y hora proporcionada por el sistema operativo del PC, mediante la diferencia de los datos generados al inicio y al final del desplazamiento. Puesto que para determinadas soluciones posibles, el recorrido realizado a fin de alcanzar el punto objetivo contiene trayectorias repetitivas que implican el consumo de grandes cantidades de tiempo y en algunos casos el incumplimiento de la condición de parada (distancia máxima al punto objetivo) tras largos intervalos de tiempo, un temporizador es empleado como límite máximo para la ejecución de la tarea, correspondiendo el valor establecido para el mismo a los 60s. El temporizador 3 está encargado de velar porque dicho límite no sea superado. Su activación se lleva a cabo al inicio de cada recorrido y se suspende una vez la condición de parada ha sido alcanzada. Cuando el tiempo de ejecución es igual al programado en el temporizador, el código detiene las labores de navegación y representación gráfica e inmediatamente se obtienen los datos correspondientes a un nuevo individuo para dar inicio a un nuevo recorrido.

La determinación del valor promedio para la variación de la orientación durante dos intervalos de muestreo consecutivos se lleva a cabo una vez los procedimientos necesarios para la navegación del móvil han sido ejecutados y se dispone de un nuevo valor para la orientación deseada del móvil. A partir de este nuevo valor y del almacenado que corresponde a la orientación del móvil durante el intervalo anterior, es posible establecer un valor promedio empleado por la función de adaptación. La distancia total recorrida por el móvil y la

distancia final al punto objetivo son calculadas empleando la información correspondiente a la ubicación de la imagen en los intervalos actual y anterior, una vez las tareas de navegación han sido ejecutadas y la representación gráfica del móvil ha generado un cambio en la posición de la imagen. La distancia total corresponde a la sumatoria de las distancias recorridas durante cada intervalo y la distancia final al punto objetivo se determina una vez la condición de parada ha sido alcanzada o el tiempo máximo de duración de la tarea ha sido superado, a partir de la posición del móvil en dicho instante y de las coordenadas del punto objetivo previamente establecidas.

El código de optimización evalúa cada individuo que constituye la población inicial almacenando el valor obtenido para la función de adaptación en un vector de tipo flotante conformado por 44 posiciones, empleando el procedimiento descrito para un recorrido sencillo. Una vez asignado un valor de adaptación para cada individuo de la población inicial, la determinación de la probabilidad de selección que cada individuo posee es llevada a cabo empleando los valores consignados en el vector *fitness* que contiene los datos correspondientes a los diversos valores de adaptación. Para ello, inicialmente se determina el valor de la sumatoria de todos los datos consignados en el vector a fin de asignar a cada individuo un valor probabilístico de selección comprendido entre cero y el valor total de la sumatoria. La relación empleada para el cálculo del vector probabilidad de 44 posiciones se muestra en (74).

$$probabilidad[i] = \frac{fitness[i]}{\sum_{i=0}^{i=43} fitness[i]} \qquad (74)$$

Donde:
$probabilidad[i]$ = Probabilidad de selección del individuo i.
$fitness[i]$ = Valor de adaptación del individuo i.

Es posible entonces establecer una serie de intervalos de probabilidad de selección para cada individuo comprendidos entre cero y uno correspondiendo el valor cero a una probabilidad de selección igual a 0% y uno a una probabilidad del 100%. El proceso de selección de los mejores individuos para ser reproducidos es puesto en marcha mediante la función denominada *selección*, encargada de recorrer el vector *probabilidad* en busca de 44 individuos que llevarán a cabo el proceso de cruce a fin de dar origen a una nueva población. Con el fin de garantizar que el individuo mejor adaptado perteneciente a una generación establecida constituida por el conjunto de soluciones abarcadas por padres e hijos sea conservado dentro de la nueva población, la función *elitismo* es empleada con el fin de realizar la búsqueda de dicha solución dentro de los vectores *fitness* correspondiente a los padres de la

generación y *fitness1* empleado de manera similar para el caso de los hijos. La selección de individuos se reduce entonces a 43 iteraciones realizadas generando el mismo número de valores aleatorios de tipo flotante en un rango comprendido entre 0 y 1, de manera tal que es posible establecer a que intervalo corresponde cada valor y así poder seleccionar al individuo contenido dentro de dicho intervalo. El conjunto de posibles soluciones conformado durante este proceso es almacenado en el vector de nombre *padres* que cuenta con 44 posiciones. Al efectuar el proceso de cruce, cada par de padres constituido por dos individuos ubicados de forma consecutiva en el vector que lleva su nombre, dará origen a un par de nuevos individuos que pasarán a formar parte de la población total de una nueva generación. El método empleado para este procedimiento corresponde al cruce basado en un punto, en el cuál la recombinación de los nuevos individuos se realiza a partir de una posición definida dentro de la representación binaria de uno de ellos, de manera tal que tras definir el punto de cruce se forma un individuo constituido por los bits que conforman el cromosoma de uno de los padres hasta el punto de cruce de la cadena, partiendo desde uno de sus extremos, y los restantes serán tomados del padre número 2, desde el punto de corte hasta el extremo opuesto al empleado inicialmente para completar la ristra deseada. El segundo individuo producto del procedimiento estará formado por los bits restantes de la cadena de representación del padre 1 y los bits restantes de la cadena del padre 2. Un ejemplo que modela la situación descrita durante el cruce de 2 padres seleccionados de la población total para una posición de cruce definido en la posición 21 de la ristra se presenta en la figura 66. Completada la operación de cruce se tendrá entonces un vector denominado *hijos* que contiene 44 nuevos individuos que pasarán a formar parte de la población total empleada por el algoritmo durante la búsqueda del individuo más apto mediante el proceso de elitismo.

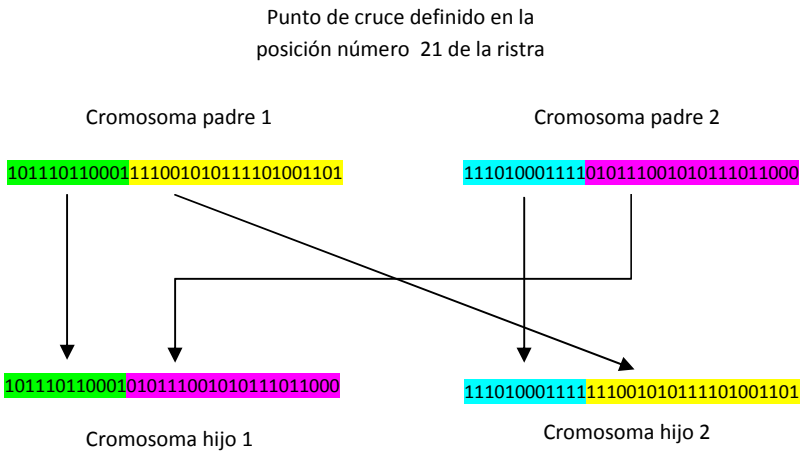
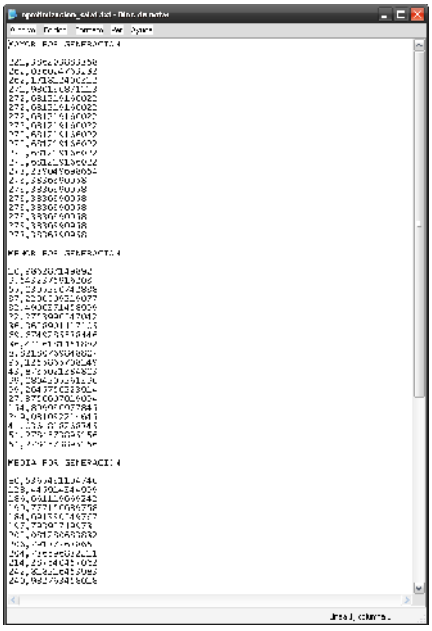


Figura 66. Ejemplo que modela el cruce de 2 padres de la población total para un cruce definido en la posición 21 de la ristra.

Para garantizar que el espacio de búsqueda sea recorrido en su totalidad durante el proceso de optimización, el operador de mutación es introducido en el algoritmo a fin de establecer un cierto margen de aleatoriedad sobre la generación de individuos tras la recombinación producto de la operación de cruce. El operador como tal, genera el cambio en el estado que representa uno de los bits constitutivos de la cadena del cromosoma que define a cada individuo, de manera tal que si el bit posee un valor correspondiente a 1 será cambiado a 0 y viceversa.

No obstante, la mutación es un proceso que debe estar sujeto a una función probabilística que determine su ocurrencia, para el caso particular de esta investigación la función está constituida por un generador de valores aleatorios comprendidos entre 0 y 1, y un valor umbral en el mismo rango que de ser superado da lugar a la aplicación del operador sobre el individuo en estudio. Para determinar la posición dentro de la ristra que será objeto del cambio, se genera un nuevo valor aleatorio esta vez de tipo entero comprendido entre 0 y 32 con cada posible solución representando a una de las posiciones que conforman el cromosoma de cada individuo de longitud 33. Una vez constituida la nueva población el proceso de evaluación de los individuos se repita hasta que la condición de parada relacionada con el número de generaciones realizadas se alcance. Cada vez que la evaluación de una generación concluye, el código realiza el cálculo de la media de los valores de adaptación obtenidos para la totalidad de los individuos y conserva la representación numérica del individuo mejor adaptado así como el valor obtenido para su función de adaptación con el fin de permitir al usuario llevar a cabo un seguimiento de la progresión de los valores obtenidos durante el proceso. Durante el proceso de optimización es posible pausar el desarrollo de la actividad en curso mediante el botón denominado *pausa.* De igual manera es posible reasumir la operación normal del proceso tras pulsar el botón *reasumir*. Una vez finalizado el proceso, un cuadro de diálogo es desplegado a fin de solicitar al usuario definir el nombre y la ubicación del archivo de texto que ha de contener la información concerniente a la media del valor de adaptación de cada generación, la representación numérica del individuo mejor adaptado y su valor en la función de adaptación. Un archivo de optimización generado se muestra en la figura 67.



10. RESULTADOS

Durante los capítulos anteriores se ha realizado un recuento sobre los desarrollos efectuados durante esta investigación, en lo correspondiente al vehículo prototipo, a la interfaz de comunicaciones con el PC y al entorno de simulación creado a fin de llevar a cabo la optimización del sistema de navegación propuesto. A continuación se presentarán los resultados obtenidos durante cada etapa del proceso investigativo, considerando los procedimientos llevados a cabo para la consecución de la información pertinente para la evaluación de los objetivos propuesto al inicio del estudio.

10.1 Implementación del prototipo de vehículo autónomo.

Durante la construcción del prototipo, una serie de circunstancias que comprometen el buen funcionamiento del mismo se hicieron evidentes. En su mayoría relacionadas a las características físicas de la estructura del vehículo y a la precisión de los sensores empleados durante la implementación. La situación más atenuante hace referencia a la determinación de la posición actual del móvil a partir de los desplazamientos generados en cada rueda dados los desperfectos físicos de los materiales empleados (ruedas de hule de diferente diámetro, espacio entre ejes incierto, etc) y la resolución límite de los codificadores empleados para tal fin. Para establecer el nivel de incertidumbre que debe ser considerado al emplear los datos de las componentes x y y que representan la posición del vehículo, un procedimiento para la determinación del error generado durante el desplazamiento fue concebido. Inicialmente, una serie de recorridos predeterminados son programados en el vehículo con el fin de establecer que capacidad posee el mismo para llevar a cabo una labor específica con el mínimo grado de error. La trayectoria constituida por los desplazamientos predeterminados, describe un cuadrado de dimensiones 5x5m de manera tal que el vehículo debe desplazarse en línea recta desde el punto de partida y recorrer una distancia igual a 5 metros para posteriormente realizar un giro de 90°, recorrer 5 metros nuevamente y repetir el proceso hasta culminar el recorrido en el punto de inicio. La figura 68 esquematiza el procedimiento llevado a cabo.

La diferencia existente entre la trayectoria deseada y la descrita por el prototipo, permite establecer el grado de confiabilidad generado por el vehículo durante el desarrollo de un desplazamiento particular. Se realizaron tres recorridos correspondientes a la trayectoria descrita obteniéndose los resultados presentados en las figuras 69a, 69b y 69c.

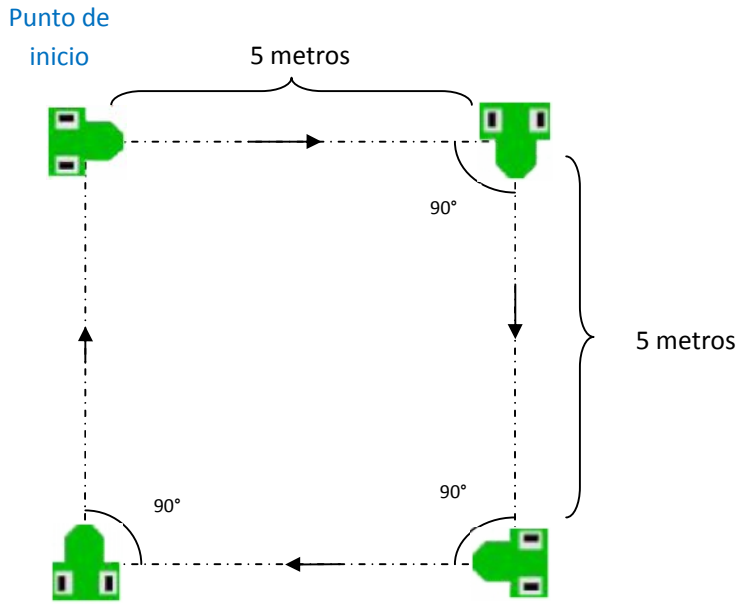


Figura 68. Trayectoria constituida por los desplazamientos predeterminados, en un cuadrado de dimensiones 5x5m, el vehículo debe desplazarse en línea recta desde el punto de partida, hasta culminar el recorrido en el punto de inicio nuevamente.

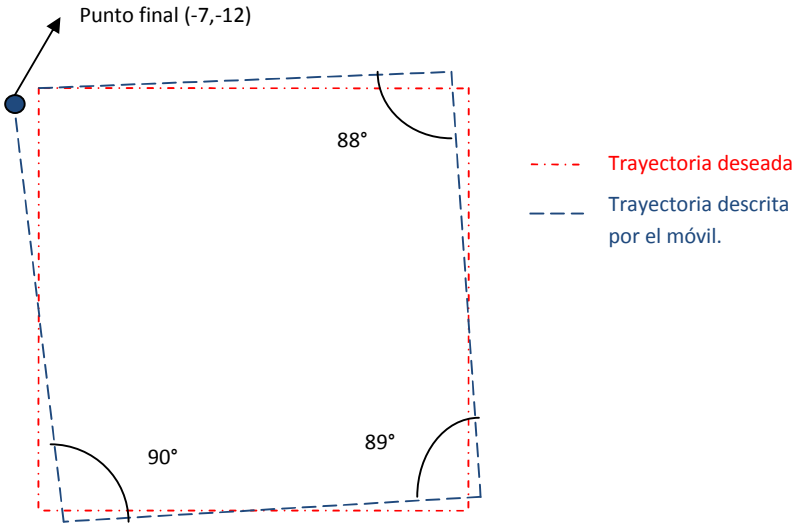


Figura 69a. Primer recorrido, se observa en rojo trayectoria deseada y en azul la trayectoria descrita por el móvil.

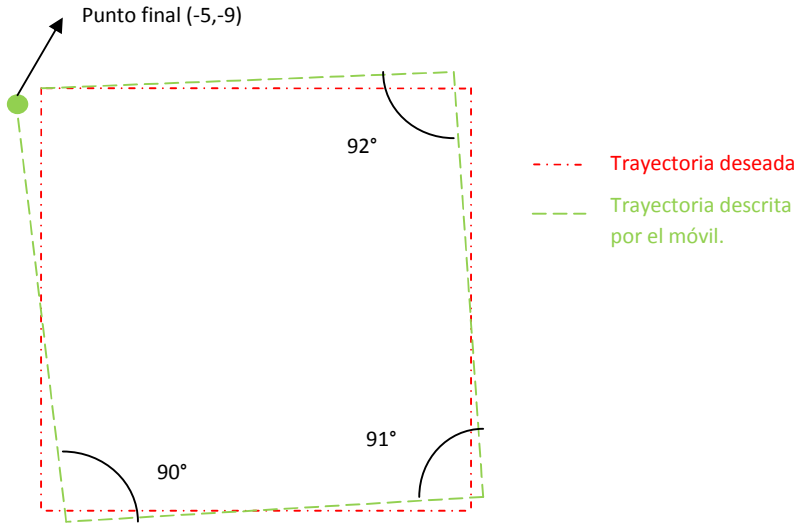


Figura 69b. Segundo recorrido, de igual forma en rojo trayectoria deseada y en verde la trayectoria descrita por el móvil.

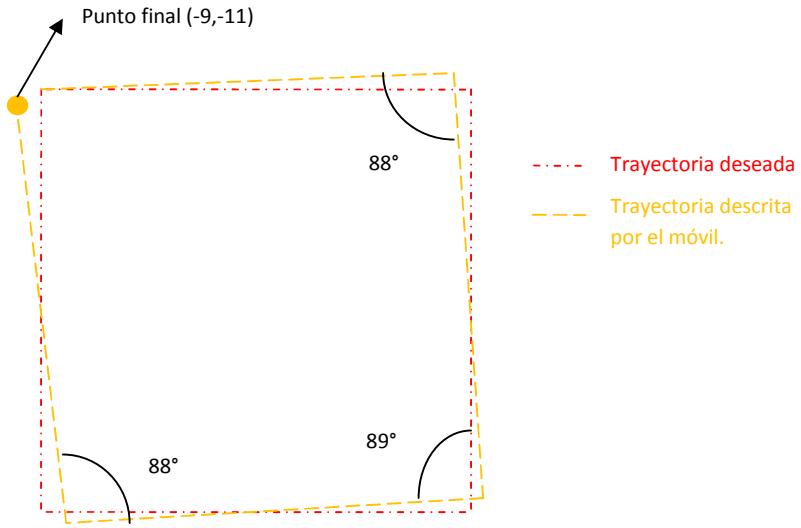


Figura 69c. Tercer recorrido, nuevamente en rojo se observa trayectoria deseada y en amarillo la trayectoria descrita por el móvil.

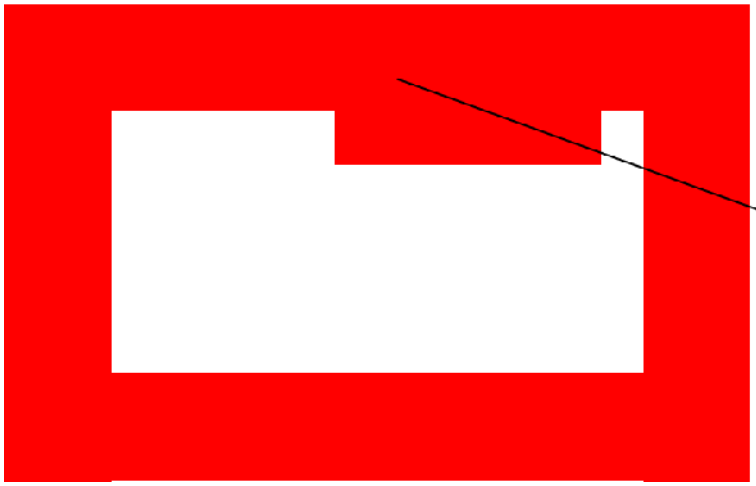
El análisis de los resultados obtenidos permite apreciar que al finalizar los tres recorridos, el vehículo alcanza una posición ubicada entre los -5cm y -9cm en el eje x con respecto al punto de inicio del desplazamiento y los -9cm y -12cm en la componente y medidos con respecto al mismo punto. Durante el recorrido, el móvil experimenta una desviación hacia su izquierda debida a diferencias en el diámetro de las dos ruedas. De manera similar durante cada giro en ángulo recto, el vehículo efectúa rotaciones comprendidas entre los 88° y 92° a causa de las diferencias en las dimensiones de cada rueda y de la incertidumbre generada a causa del punto de contacto de cada rueda con el

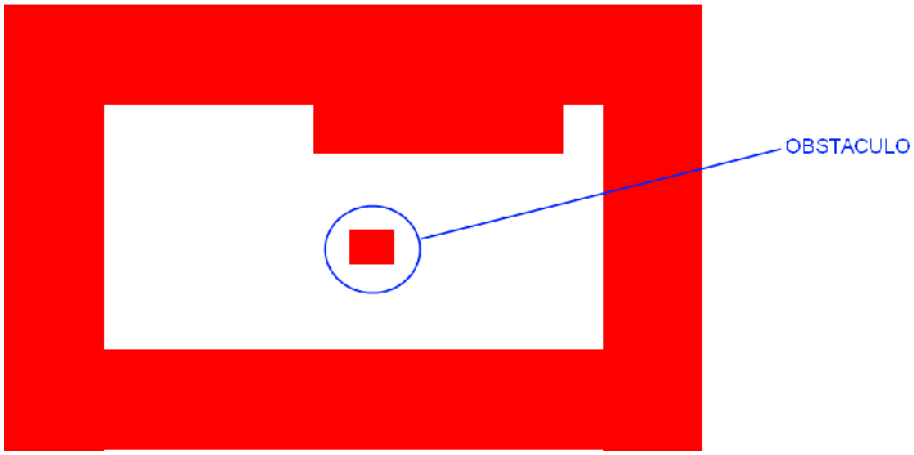
suelo que determina la distancia entre los ejes empleada para establecer el ángulo del giro realizado por el móvil. Aunque porcentualmente las diferencias encontradas entre el desplazamiento ideal y el efectuado por el prototipo no superan el 5% del desplazamiento total (desplazamiento total correspondiente a la distancia recorrida o al ángulo de giro efectuado), el error establecido es de carácter incremental y ha de continuar su progresión a medida que la longitud y complejidad del recorrido a efectuar aumentan ocasionando diferencias de mayor relevancia entre el comportamiento informado por el móvil y las características reales de su desplazamiento. Por tanto es necesario efectuar estudios más específicos sobre sistemas de posicionamiento para robots móviles a fin de minimizar el impacto ocasionado por dichos efectos.

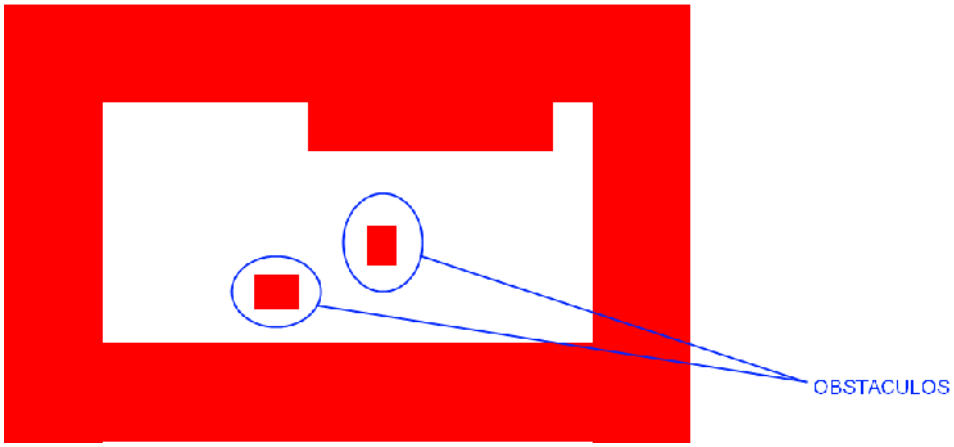
10.2 Resultados para el proceso de optimización del sistema de navegación implementado.

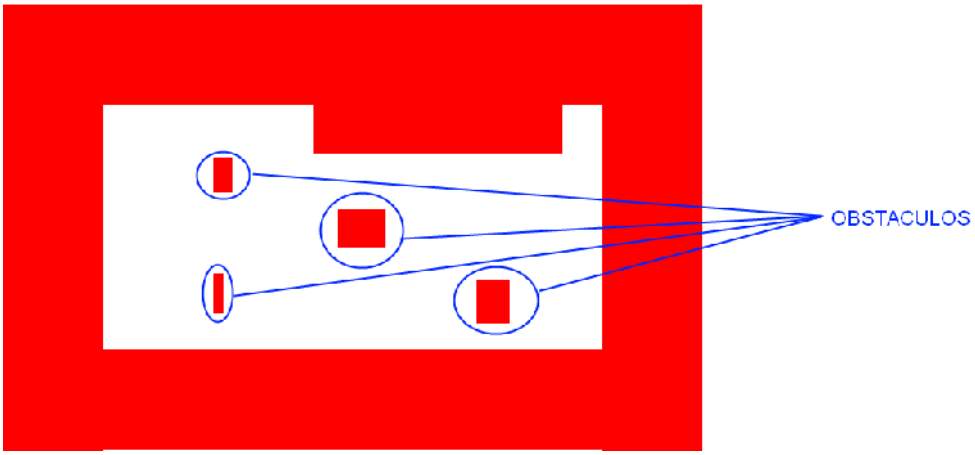
La puesta en marcha del proceso de optimización para el sistema de navegación implementado requiere la construcción de una serie de entornos empleados para simular el comportamiento del prototipo desarrollado durante desplazamientos en entorno constituidos por una serie de obstáculos de diversas características. Tras implementar el sistema de navegación basado en el método del campo de fuerzas virtuales fue posible apreciar que el comportamiento del vehículo está relacionado estrechamente con la cantidad de obstáculos presentes en el entorno de trabajo. De manera similar, debido a que los parámetros empleados para la construcción de la función de adaptación, base del funcionamiento del algoritmo genético implementado, incluyen la distancia total recorrida y el tiempo transcurrido durante la tarea especificada, posiblemente configuraciones significativamente diferentes presenten resultados óptimos en los diferentes entornos construidos. Por tal motivo tres entornos diferentes fueron diseñados considerando la cantidad de obstáculos presentes y las características del espacio físico empleado para la realización de las pruebas reales del sistema. El espacio físico empleado durante los recorridos reales está delimitado por cuatro muros densos tal y como se aprecia en la figura 70.

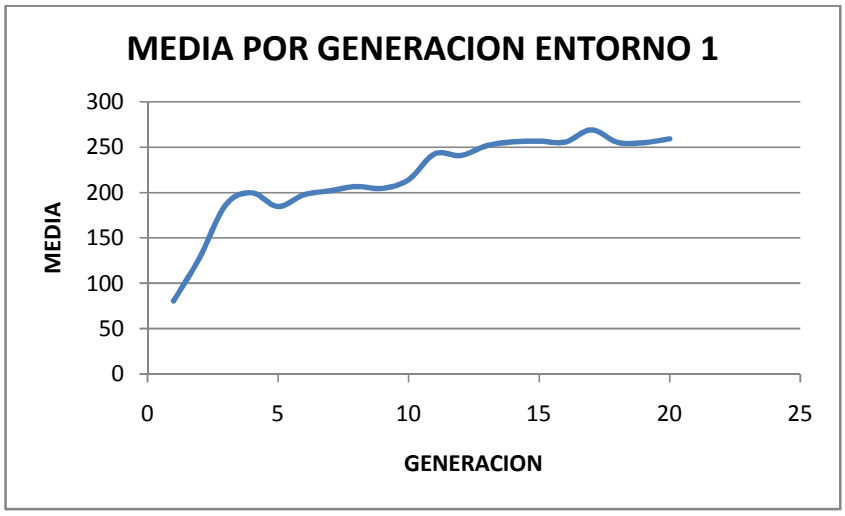
El primer entorno está constituido por un solo obstáculo posicionado cerca del centro del espacio general de trabajo como se muestra en la figura 71a. El segundo está constituido por dos obstáculos rectangulares de diferentes dimensiones, posicionados según la representación expuesta en la figura 71b. Finalmente, el entorno tres posee un grado de complejidad mayor al estar constituido por un conjunto de cuatro obstáculos rectangulares dispuestos como se muestra en la figura 71c.


Contorno del espacio fisico de trabajo delimitado por cuatro muros densos de la forma especificada.


OBSTACULO


OBSTACULOS

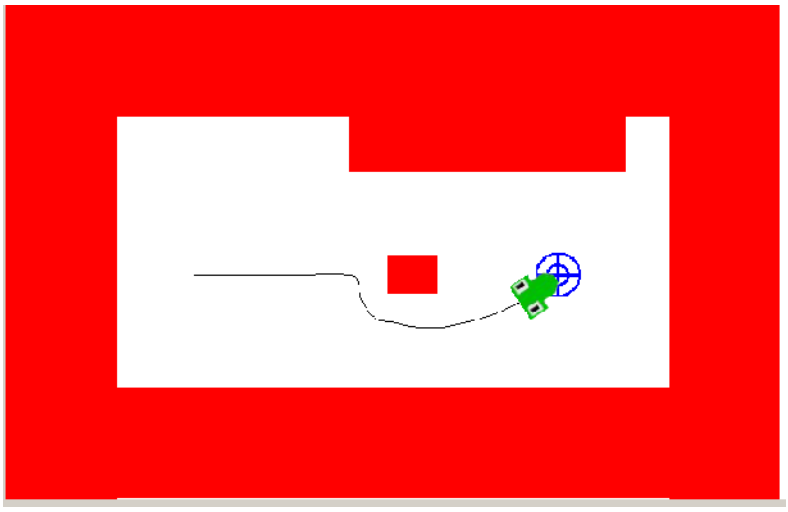

OBSTACULOS

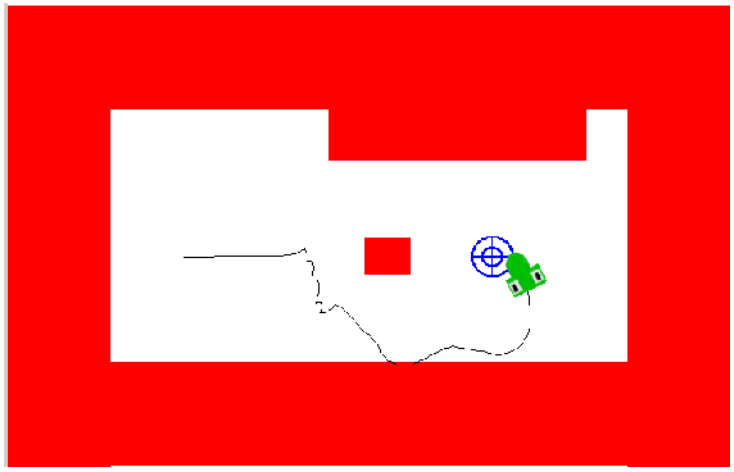

MEDIA POR GENERACION ENTORNO 1
300
250
200
150
100
50
0
MEDIA
0
5
10
15
20
25
GENERACION

REPRESENTACIÓN BINARIA	REPRESENTACION HEXADECIMAL	REPRESENTACION DECIMAL
10100100010101011101101010010101011	0x148ABB52B	5514179883

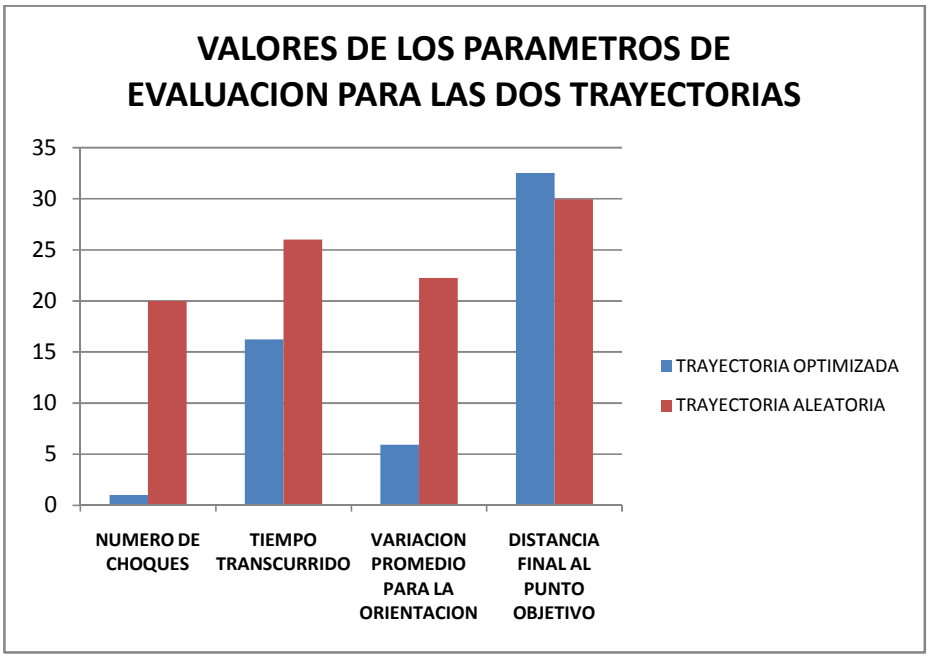
VALOR DE ADAPTACION INDIVIDUO MAYOR
273,3836390958

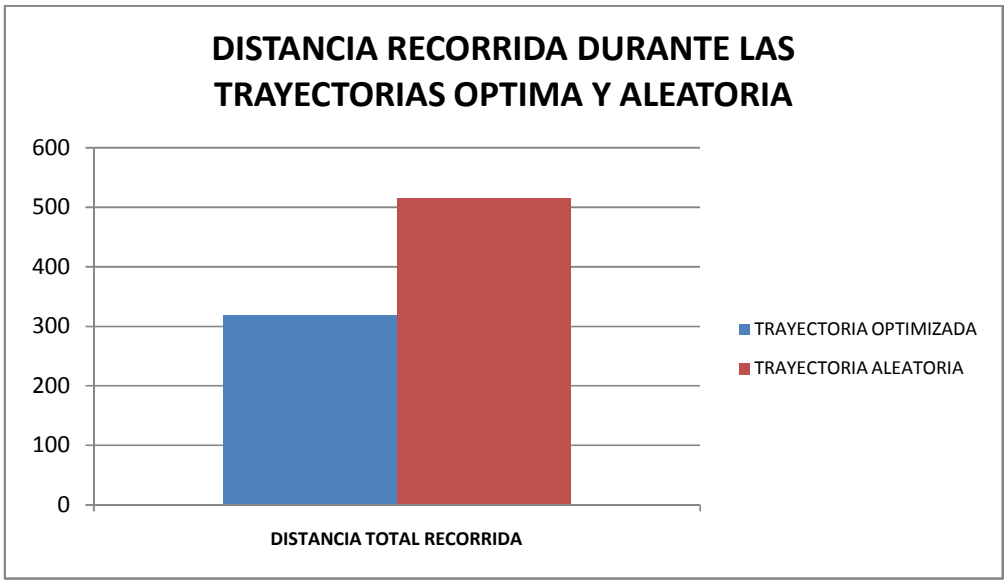
	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (*ω*)	Factor de reducción para la velocidad (*z*)	Constante de tiempo para el filtro pasa bajos (*thao*)
Representación binaria	10100100	01010101	110110	101001	01011
Representación decimal	164	85	54	41	11

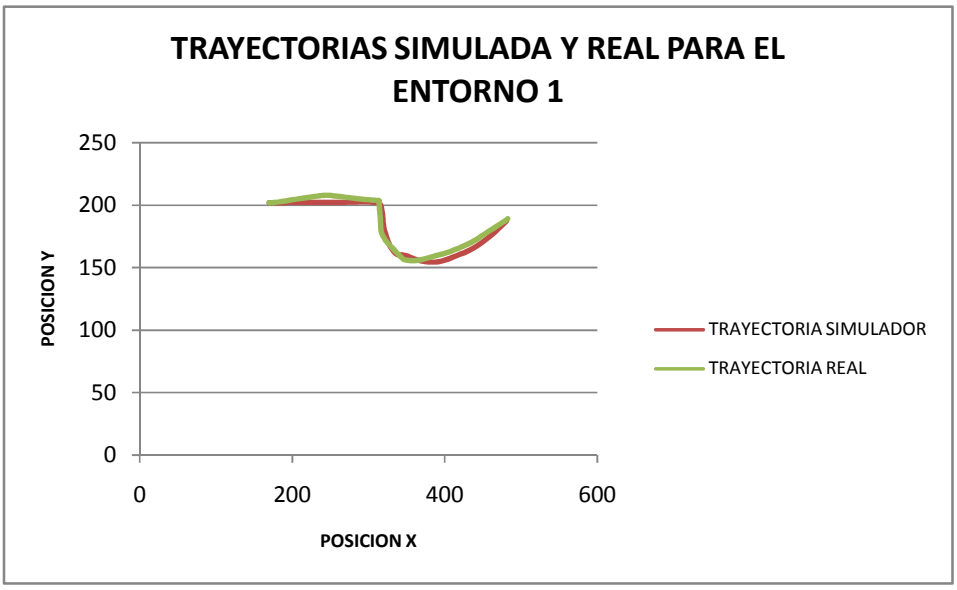


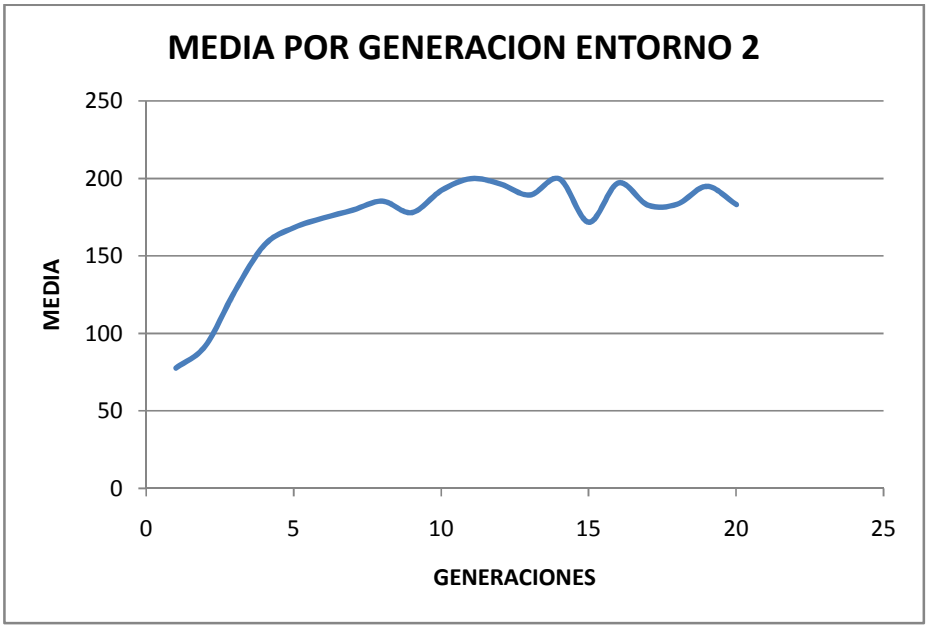


	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (*ω*)	Factor de reducción para la velocidad (*z*)	Constante de tiempo para el filtro pasa bajos (*thao*)
Configuración optimizada	164	85	54	41	11
Configuración aleatoria	30	2	40	55	12
VALORES OBTENIDOS PARA LOS PARAMETROS DE EVALUACION					
	Número de choques	**Tiempo transcurrido**	**Variación promedio de la orientación**	**Distancia total recorrida**	**Distancia final al punto objetivo**
Configuración optimizada	1	16.25	5.96	318.97	32.52
Configuración aleatoria	20	26.01	22.25	514.73	29.96


VALORES DE LOS PARAMETROS DE EVALUACION PARA LAS DOS TRAYECTORIAS
35
30
25
20
15
10
5
0
NUMERO DE CHOQUES
TIEMPO TRANSCURRIDO
VARIACION PROMEDIO PARA LA ORIENTACION
DISTANCIA FINAL AL PUNTO OBJETIVO
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA

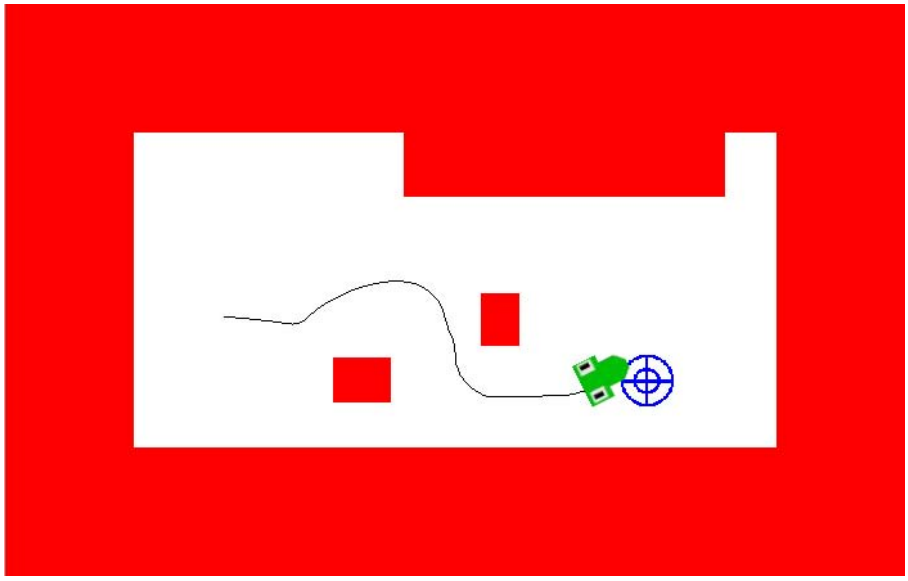

DISTANCIA RECORRIDA DURANTE LAS TRAYECTORIAS OPTIMA Y ALEATORIA
600
500
400
300
200
100
0
DISTANCIA TOTAL RECORRIDA
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA

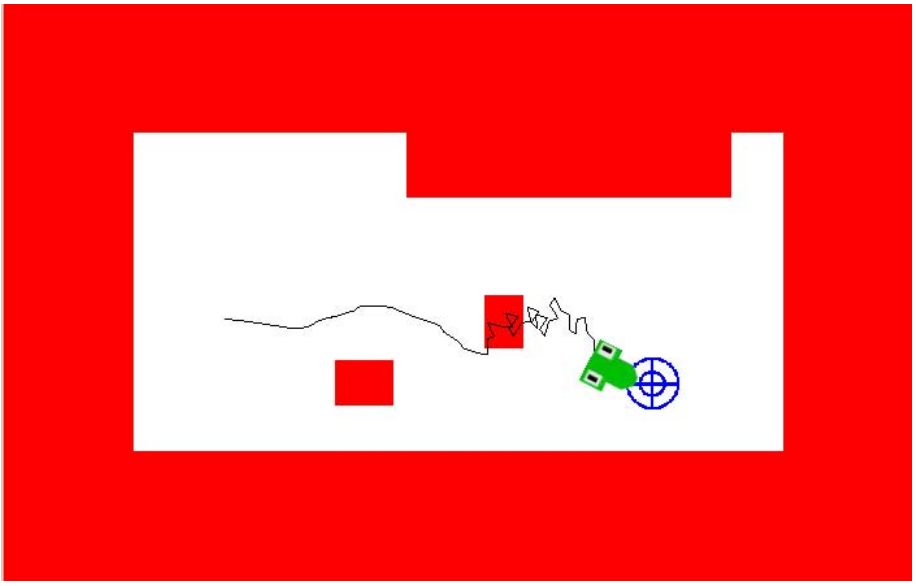

TRAYECTORIAS SIMULADA Y REAL PARA EL ENTORNO 1
250
200
150
100
50
0
POSICION Y
0
200
400
600
POSICION X
TRAYECTORIA SIMULADOR
TRAYECTORIA REAL


MEDIA POR GENERACION ENTORNO 2
250
200
150
100
50
0
MEDIA
0
5
10
15
20
25
GENERACIONES

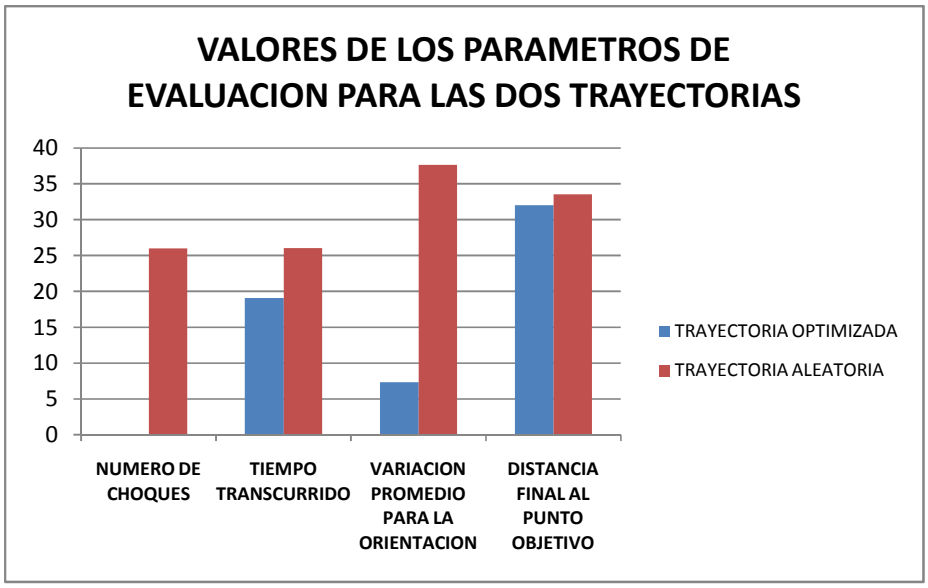
REPRESENTACIÓN BINARIA	REPRESENTACION HEXADECIMAL	REPRESENTACION DECIMAL
10111110000111111001000000 01011111	0x17C3F205F	6379479135
VALOR DE ADAPTACION INDIVIDUO MAYOR		
233,248848		

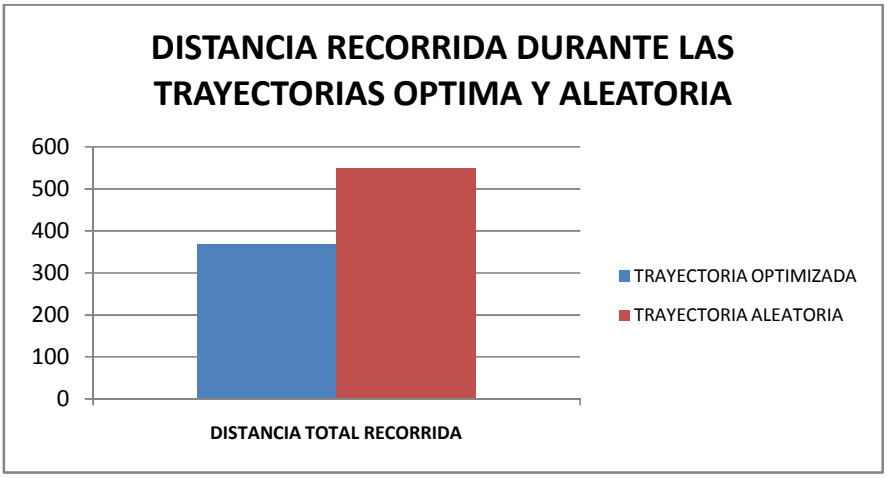
	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (*ω*)	Factor de reducción para la velocidad (*z*)	Constante de tiempo para el filtro pasa bajos (*thao*)
Representación binaria	10111110	00011111	100100	000010	11111
Representación decimal	190	31	36	2	31

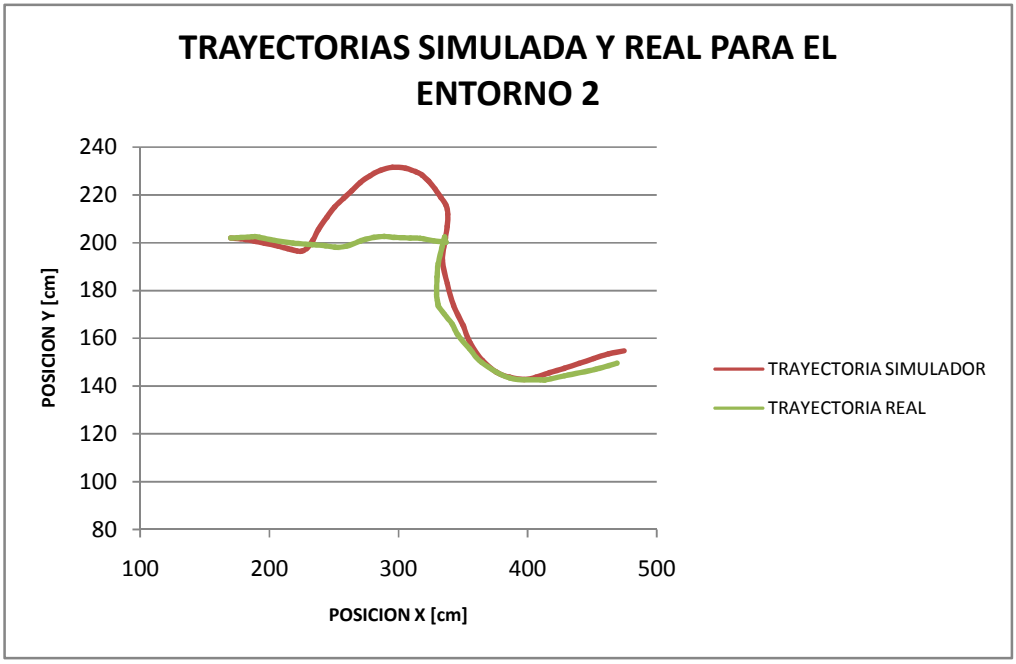


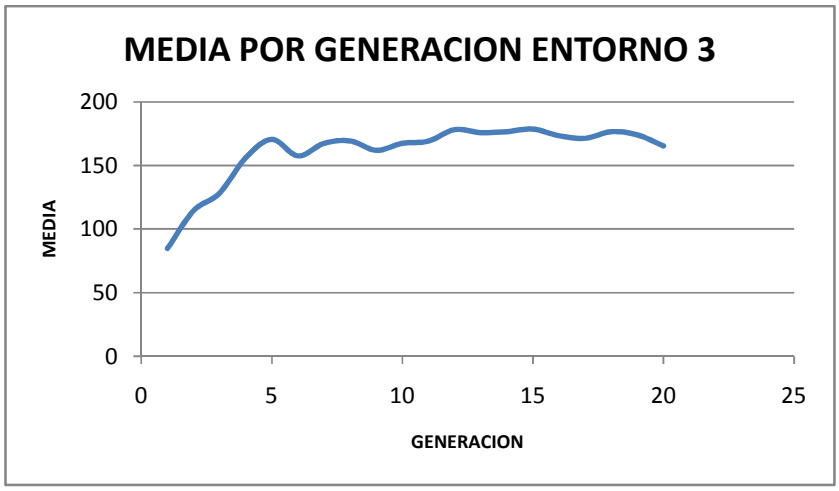


	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (ω)	Factor de reducción para la velocidad (z)	Constante de tiempo para el filtro pasa bajos ($thao$)
Configuración optimizada	190	31	36	2	31
Configuración aleatoria	150	39	11	20	54
VALORES OBTENIDOS PARA LOS PARAMETROS DE EVALUACION					
	Número de choques	**Tiempo transcurrido**	**Variación promedio de la orientación**	**Distancia total recorrida**	**Distancia final al punto objetivo**
Configuración optimizada	0	19.10	7.33	369.04	32.01
Configuración aleatoria	26	26.01	49.51	548.99	33.52


VALORES DE LOS PARAMETROS DE EVALUACION PARA LAS DOS TRAYECTORIAS
40
35
30
25
20
15
10
5
0
NUMERO DE CHOQUES
TIEMPO TRANSCURRIDO
VARIACION PROMEDIO PARA LA ORIENTACION
DISTANCIA FINAL AL PUNTO OBJETIVO
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA

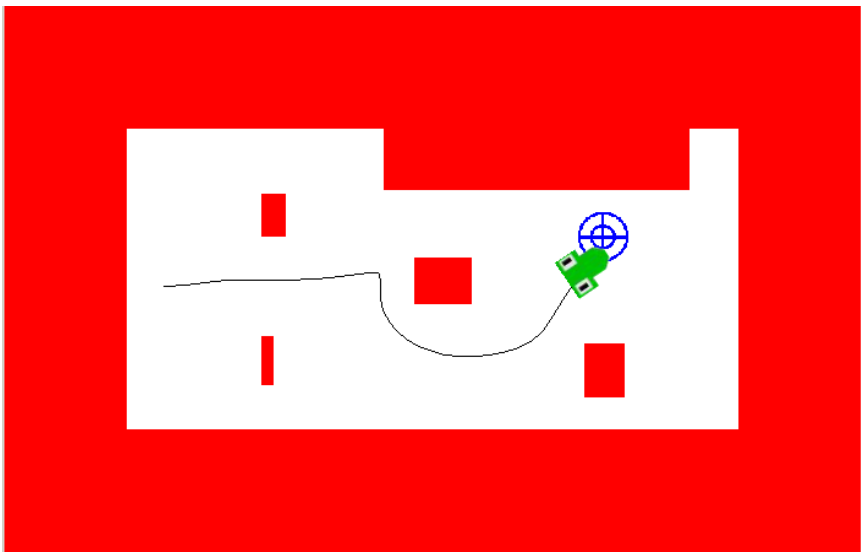

DISTANCIA RECORRIDA DURANTE LAS TRAYECTORIAS OPTIMA Y ALEATORIA
600
500
400
300
200
100
0
DISTANCIA TOTAL RECORRIDA
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA

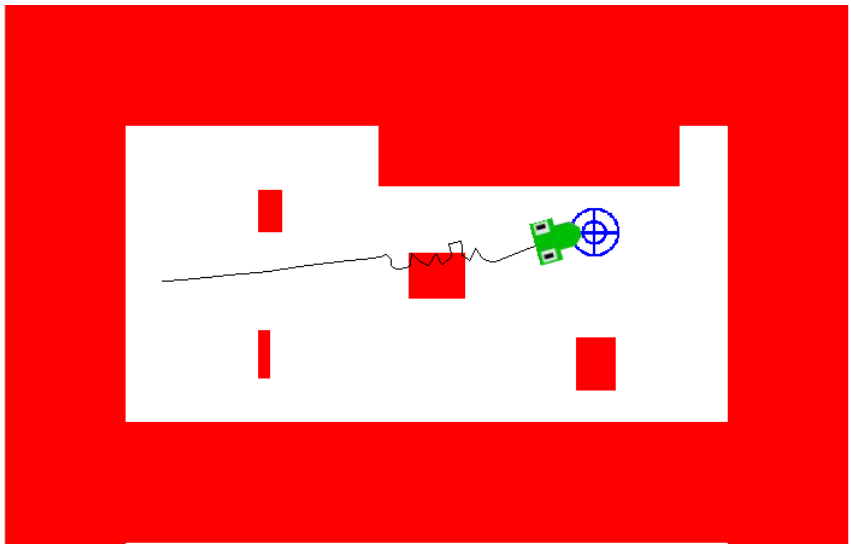

TRAYECTORIAS SIMULADA Y REAL PARA EL ENTORNO 2
240
220
200
180
160
140
120
100
80
POSICION Y [cm]
100
200
300
400
500
POSICION X [cm]
TRAYECTORIA SIMULADOR
TRAYECTORIA REAL



REPRESENTACIÓN BINARIA	REPRESENTACION HEXADECIMAL	REPRESENTACION DECIMAL
11110111010011111111111100110010111	0x1EE9FF997	8298428823
VALOR DE ADAPTACION INDIVIDUO MAYOR		
199,121356318304		

	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (*ω*)	Factor de reducción para la velocidad (*z*)	Constante de tiempo para el filtro pasa bajos (*thao*)
Representación binaria	11110111	01001111	111111	001100	10111
Representación decimal	247	79	63	12	23





configuración optimizada y tras la puesta en marcha del sistema de navegación bajo la configuración aleatoria.

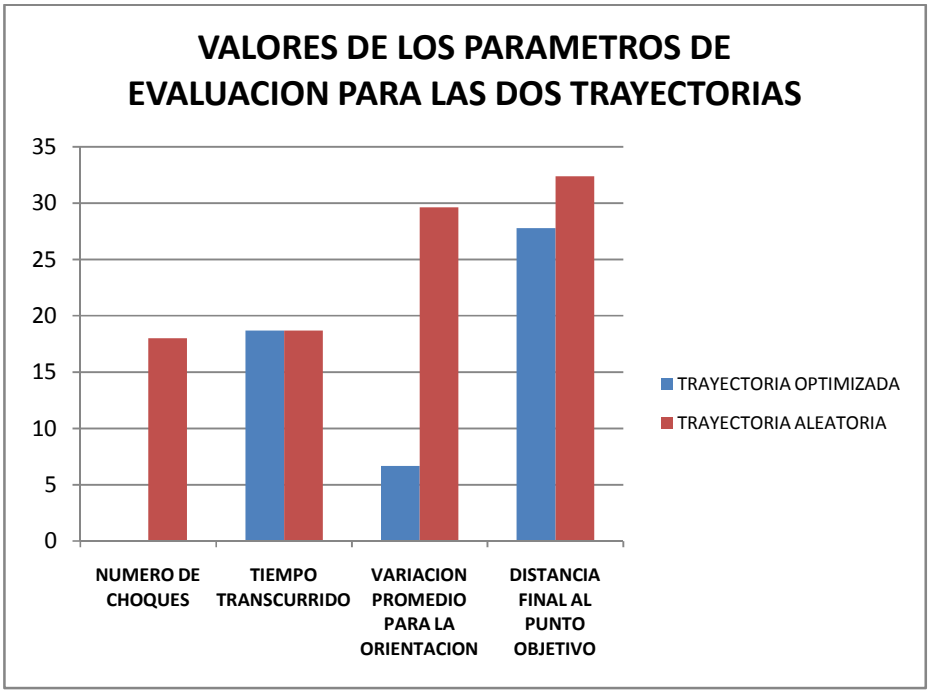
	Constante de Repulsión	Constate de Atracción	Constante de amortiguamiento (ω)	Factor de reducción para la velocidad (z)	Constante de tiempo para el filtro pasa bajos ($thao$)
Configuración optimizada	247	79	63	12	23
Configuración aleatoria	184	212	21	5	49
VALORES OBTENIDOS PARA LOS PARAMETROS DE EVALUACION					
	Número de choques	**Tiempo transcurrido**	**Variación promedio de la orientación**	**Distancia total recorrida**	**Distancia final al punto objetivo**
Configuración optimizada	0	18.70	6.67	427.29	27.78
Configuración aleatoria	18	18.70	29.63	416.09	32.38

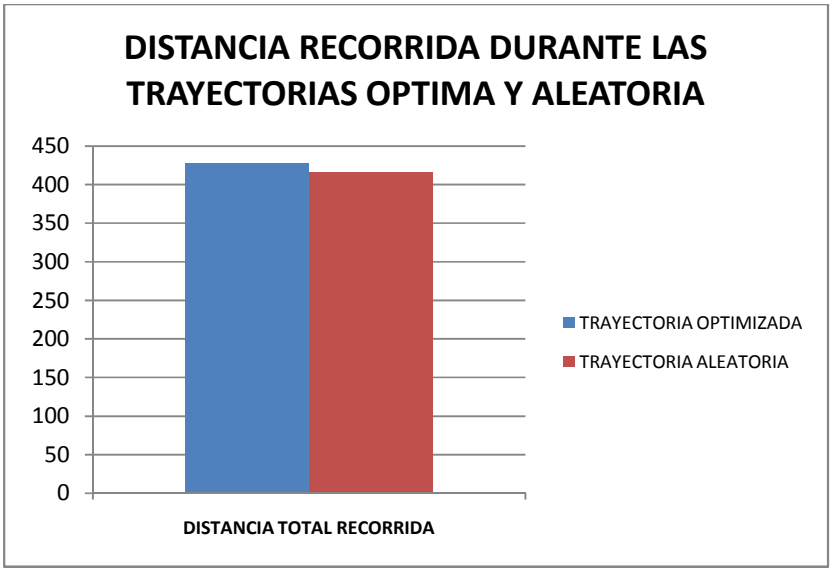
Tabla 18. Representación binaria de los valores derivados, asociada al individuo aleatorio, así como los valores generados para los parámetros de evaluación durante el recorrido empleando la optimización.

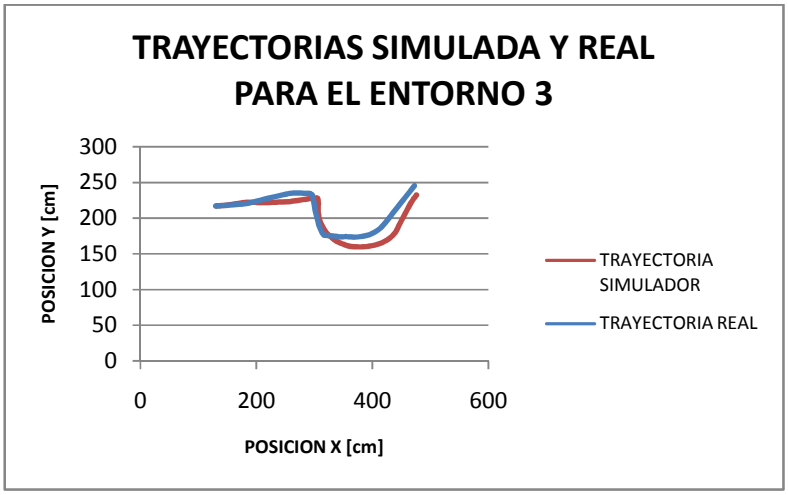
Mediante la comparación gráfica de los resultados generados, bosquejada en las figuras 85a y 85b, es posible determinar un efecto de reducción sobre los valores de los parámetros de evaluación correspondientes al número de choques, variación promedio para la orientación y la distancia final al punto objetivo, como resultado del proceso de optimización. Aunque durante el desplazamiento efectuado empleando la configuración optimizada, el móvil recorre una distancia mayor, la diferencia existente entre este valor y el alcanzado al efectuar el desplazamiento bajo los efectos de la configuración aleatoria, no supera el 3% del valor total para la distancia recorrida, en tanto que los efectos de reducción de los valores asociados a los demás parámetros de evaluación son significativamente favorables para el comportamiento deseado del vehículo.

10.10 Comparación entre la trayectoria simulada y la trayectoria real.

Tras culminar el proceso de optimización, los datos derivados de la configuración asociada al individuo mejor adaptado son transferidos al prototipo a fin de evaluar el comportamiento del mismo bajo la configuración establecida. Como resultado, el vehículo describe la trayectoria mostrada en la figura 86. Puede apreciarse que los defectos estructurales del vehículo mencionados anteriormente generan algunas discrepancias entre la trayectoria generada por el simulador y la efectuada en la realidad por el móvil. No obstante la tendencia que predomina durante el par de desplazamientos en estudio es similar y el


VALORES DE LOS PARAMETROS DE EVALUACION PARA LAS DOS TRAYECTORIAS
35
30
25
20
15
10
5
0
NUMERO DE CHOQUES
TIEMPO TRANSCURRIDO
VARIACION PROMEDIO PARA LA ORIENTACION
DISTANCIA FINAL AL PUNTO OBJETIVO
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA


DISTANCIA RECORRIDA DURANTE LAS TRAYECTORIAS OPTIMA Y ALEATORIA
450
400
350
300
250
200
150
100
50
0
DISTANCIA TOTAL RECORRIDA
TRAYECTORIA OPTIMIZADA
TRAYECTORIA ALEATORIA


TRAYECTORIAS SIMULADA Y REAL
PARA EL ENTORNO 3
300
250
200
150
100
50
0
POSICION Y [cm]
0
200
400
600
POSICION X [cm]
TRAYECTORIA SIMULADOR
TRAYECTORIA REAL

11. CONCLUSIONES

La construcción de un vehículo autónomo es un proceso que requiere considerar diversos aspectos referentes a la labor específica que ha de desempeñar el mismo. Durante esta investigación se comprobó que es posible implementar un prototipo dotado de un sistema sensorial capaz de recopilar información acerca de su entorno, que puede operar efectivamente en ambientes controlados que no impliquen grandes retos para efectuar desplazamientos. Entornos con características geológicas más complejas requerirán el desarrollo de estructuras mecánicas de mayor envergadura que permitan salvaguardar los componentes asociados al funcionamiento del vehículo y sistemas de tracción de mayor capacidad a fin de mitigar los efectos generados por falsos desplazamientos y condiciones de difícil tránsito para el móvil. La red sensorial empleada por el prototipo desarrollado permite recopilar información referente al entorno de trabajo, importante para procesos de reconstrucción del ambiente, no obstante las características operativas de los sensores empleados para su constitución determinan la operatividad del sistema en ambientes con alto grado de ruido industrial. El vehículo cuanta con una interfaz de comunicaciones que facilita su operación mediante el uso de un PC sin grandes requerimientos externos, entablando comunicación bidireccional que permite dar inicio a cada labor específica y durante la misma recopilar la información generada por el vehículo para traducirla en información relevante para el usuario final.

Las características estructurales del prototipo desarrollado delimitan su confiabilidad puesto que ciertos materiales empleados durante su elaboración, introducen al modelo estructural ciertos rangos de incertidumbre que dificultan la definición exacta de las dimensiones del vehículo, empleadas por el sistema de posicionamiento del mismo, generando errores de tipo incremental sobre la determinación de la posición en el entorno de trabajo, que se tornan significativos en desplazamientos de gran longitud.

La implementación de un sistema embebido de navegación autónomo, basado en el método de campo de fuerzas virtuales (VFF) es un proceso de gran viabilidad en la búsqueda de reducir costos operativos para sistemas de navegación, dada la adaptabilidad del método empleado a las características funcionales de los dispositivos en uso. No obstante, los dispositivos empleados para el procesamiento de la información pertinente al entorno deben contar con una elevada capacidad operativa, que permita realizar las tareas necesarias para la navegación del móvil de manera pertinente a fin llevar a cabo las labores asociadas a la reconstrucción del entorno a la mayor frecuencia posible.

El dispositivo DSPic30f4011 empleado para el procesamiento de datos durante esta investigación, proporciona una serie de características favorables al desempeño del sistema de navegación embebido, derivadas de su estructura modular que permite el control de diferentes sistemas de manera rápida y sencilla, disminuyendo la complejidad durante el proceso de implementación. La navegación del móvil empleando el método de campo de fuerzas virtuales constituye un proceso de gran adaptabilidad en ambientes de características estables puesto que proporciona al vehículo cierto grado de comportamiento reactivo que le permite sortear obstáculos presentes en el entorno mediante recorridos suaves acordes a las características del prototipo desarrollado. En general, el sistema de navegación implementado cumple con las características propuestas durante la formulación de los objetivos de la investigación, adaptándose a una serie de entornos elaborados con el fin de establecer el grado de aplicabilidad del sistema, a partir de los valores empleados para el conjunto de variables empleadas por el método durante las tareas necesarias para la navegación.

El proceso de optimización del sistema de navegación implementado, definido como la búsqueda del conjunto de valores asociados a las variables empleadas por el método de campo de fuerzas virtuales se hace necesario a fin de garantizar la adaptabilidad del sistema a diferentes entornos, puesto que pequeñas variaciones en dichos valores ocasionan cambios significativos en el comportamiento del vehículo. Debido al gran número de combinaciones posibles existentes para este conjunto, el proceso de búsqueda puede llegar a consumir una enorme cantidad de recursos asociados a la duración del proceso, implicando la puesta en marcha de soluciones de búsqueda que minimicen los efectos provocados por este consumo. Culminada la investigación, es posible afirmar que los algoritmos genéticos pueden ser aplicados de manera exitosa a fin de llevar a cabo la optimización del sistema desarrollado, dado que permiten establecer configuraciones de excelentes prestaciones para el funcionamiento general del prototipo en diversos ambientes, mediante procedimientos de búsqueda autónomos que consumen un porcentaje de tiempo significativamente menor al empleado por otros sistemas destinados a la misma labor. La selección de la configuración deseada se ve estrechamente ligada a la definición de los parámetros de evaluación empleados para la construcción de una función que permita evaluar la adaptación de cada solución posible al problema propuesto. En el caso del sistema de navegación implementado, el proceso de optimización arrojó como resultado una disminución significativa en los valores de los parámetros de evaluación seleccionados durante la operación del móvil en cada entorno construido, evidenciando el buen funcionamiento del procedimiento

desarrollado. Por tanto, el éxito del proceso de optimización se ve estrechamente ligado a la correcta formulación del problema a solucionar, mediante la construcción de una función para la evaluación del nivel de adaptación de cada posible solución al sistema propuesto, que traduzca de manera adecuada las necesidades expresadas durante la formulación del problema a un modelo numérico para cuantificar los efectos favorables o contrarios que una solución específica genera al ser aplicada.

La puesta en marcha del proceso de optimización del sistema de navegación implementado empleando algoritmos genéticos hace necesario el desarrollo de una aplicación para la simulación del comportamiento del prototipo a fin de recopilar los datos necesarios para la búsqueda de la solución ideal con un mínimo consumo de recursos operativos y temporales. Durante la investigación, una aplicación tipo software fue desarrollada para este fin de manera exitosa, considerando diversos factores que influyen sobre el comportamiento real del vehículo construido a fin de concebir procedimientos de simulación apegados al comportamiento real del prototipo. Sin embargo, las incertidumbre estructurales debidas a los materiales empleados durante la construcción generan cierto rango de discrepancia entre los resultados entregados por el simulador y el comportamiento descrito por el móvil tras el proceso de optimización, haciendo necesario minimizar el impacto que dicho rango de incertidumbre tiene sobre el comportamiento deseado y el llevado a cabo en la realidad por el prototipo implementado, mediante construcciones de mayor precisión y la implementación de sistemas de fusión sensorial para la determinación de la posición del móvil en el entorno de trabajo.

BIBLIOGRAFIA

BORENSTEIN, Johann. Measurement and Correction of systematic Odometry Errors in Mobile Robots. USA: IEEE Transactions on Robotics and Automation. December , 1996; Vol 12, No 6.

BORENSTEIN, Johann y KOREN, Yorem. Fast Obstacle Avoidance For Mobile Robots. USA: IEEE Journal of Robotics and Automation. June 1991. Vol 7, No 3; pp. 278-288.

BORENSTEIN, Johann y KOREN, Yorem. Histogramic In-Motion Mapping For Mobile Robot Obstacle Avoidance. USA: IEEE Journal of Robotics and Automation. 1991. Vol 7, No 4; pp. 535-539.

BORENSTEIN, Johann y KOREN, Yorem. Obstacle Avoidance With Ultrasonic sensors. USA: IEEE Journal of Robotics and Automation. April 1988. Vol 4, no 5.

BORENSTEIN, Johann, KOREN, Yorem and LEVINE, Simon. Obstacle – Avoiding Navigation System. USA: United stated Patent; Patent Number 5,006,988. April 9, 1991.

BORENSTEIN, Johann y KOREN, Yorem. Real – Time Obstacle Avoidance For Fast Mobile Robots. USA: IEEE Transaction on Sistems, Man, and Cibernetics. September/October 1989. Vol 19, no 5.

BORENSTEIN, Johann and KOREN, Yorem. The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. USA: IEEE Journal of Robotics and Automation, 1991; Vol 7, No 3.

CALVO, Oscar, RODRIGUEZ, Gonzalo and PICO, Rodrigo. Real Time Navigation of Autonomous Robot. España: June, 2001.

GALLARDO, G.; COLOMINA, O.; FLORES, F.; ARQUES, O.; COMPANY, P. y RISO, R. Control Local de Robots Móviles basado en Métodos Estadísticos y Algoritmos Genéticos. Conferencia de la Asociación Española para la Inteligencia Artificial (CAEPIA'97); España, Torremolinos, Málaga: Generación de Trayectorias Robustas Mediante Algoritmos Genéticos. Noviembre 1997.

MUÑOZ, V.; OLLERO, A.; PRADO, M.; SIMON, A. Mobile Robot Trajectory Planning with Dynamic and Kinematic Constraints. España: IEEE International Conference on Robotics and Automation, 1994.

OLLERO, Anibal. Planificación de Trayectorias Para Robots Móviles. Tesis Doctoral: Universidad de Málaga. España. Julio 5 1995.

RÍOS, Luis y BUENO, Maximiliano. Modelo Matemático para un Robot Móvil. Colombia. Universidad Tecnológica de Pereira, Abril 2008.

ANEXO A Código del micro-controlador maestro del sistema de navegación.

```
/****************** MICRO CONTROLADOR MAESTRO SISTEMA DE NAVEGACION*****************************

Este programa se encarga de llevar a cabo las tareas necesarias para la navegación del vehículo autónomo, mediante
la recopilación de información sensorial, la localización del vehículo, el control del sonar, etc.  Está implementado en un
dispositivo de referencia DSPic 30f4011.*/

/**LIBRERIAS*/

#include <30f4011.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <float.h>


/**CONFIGURACION DEL MICROCONTROLADOR*/

#fuses PR,XT_PLL8,NOPROTECT,NOCKSFSM,NOWDT,DEBUG,NOWRT,PUT64,COE,NOMCLR
#define Device_SDA PIN_F2
#define Device_SCL  PIN_F3
#use delay(clock=80000000)
#use rs232(uart2,baud=9600, xmit=PIN_F5, rcv=PIN_F4)
#use i2c(master, slow, sda=device_SDA, scl=device_SCL,force_hw)

/**ASIGNACION DE REGISTROS INTERNOS DEL DISPOSITIVO A VARIABLES GLOBALES*/

#WORD INTCON1 = 0x0080
#WORD PTCON = 0x01C0
#WORD PWMCON1 = 0x01C8
#WORD PWMCON2 = 0x01CA
#WORD OVDCON = 0x01D4
#WORD PDC1 = 0x01D6
#WORD PDC2 = 0x01D8
#WORD PDC3 = 0x01DA
#WORD PTPER = 0x01C4
#WORD IFS0=0x0084
#WORD IFS1=0x0086
#WORD IEC0=0x008C
#WORD IEC1=0x008E
#WORD U2STA=0x0218
#WORD T2CON=0x0110
#WORD T4CON=0x011E
#WORD PR2=0x010C
#WORD PR4=0x011A
#WORD IPC1=0x0096
#WORD IPC5=0x009E
#WORD IPC6=0x00A0
#WORD TMR2=0x0106
#WORD TMR4=0x0114
#WORD U2RXREG=0x021C

/**DEFINICION DE VARIABLES EMPLEADAS*/

int dato,dato1,dato2,dato3,dummie,inicio,contador,giro;
int posicion,direccion,opcion,brujula1,brujula2,brujula_dummie;
int iq;
int8 coeficiente,coeficiente1,sentido_g;
int8 matriz[33][33]={};
int8 bandera,bandera1,tarea;
float ic,jc,uc,vc,delta_d,delta_i,diana,vl,delta_angulo,deltad,f1,f2,f3,f4,reci,recd;
int8 ie,je,ue,ve,vb;
signed int cuenta_matriz;
int8 variable[4]={};
int8 variable_2[4]={};
int8 variable_3[4]={};
int8 address_1,address_2,cont_giro;
float
angulo,repulsion_x,repulsion_y,distancia_celda,repulsion,atraccion,distancia_objetivo,atraccion_x,atraccion_y,pos_bruj
ula,kv,w,z,sta,thao,orientacion_1,orientacion_2,delta_s,delta_ang,rec_d,rec_i,brujula_anterior,brujula_actual;
```

```
float
posicion_x,posicion_y,fuerza_x,fuerza_y,orientacion,recorrido_d,recorrido_ant_d,recorrido_i,recorrido_ant_i,referencia,
carro_x,carro_y,carro_xant,carro_yant,celda_x1,celda_y1,ang_actual,orientacion_anterior,b,orientacion_inicial,orientaci
on_brujula;

float actuali,actuald,anteriori,anteriord,velmax;
unsigned int8 constante_repulsion,constante_atraccion;
signed int16 objetivo_x,objetivo_y;
int dato_rcv[4]={};
//int8 comando[5]={};
//int8 entrada[3]={};
int16 distancia,distancia1,sentido_anterior;
signed int mx,my;
int vl1;
int16 serial1,serial2,serial3,serial4;


/**INTERRUPCION PARA EL TEMPORIZADOR EMPLEADO PARA DETERMINACION DE ORIENTACION Y
LOCALIZACION DEL VEHICULO*/
#INT_TIMER2
void timer_calculos()
{

bit_clear(IFS0,6);
bandera=1;
bit_clear(T2CON,15);

}

/**INTERRUPCION PARA LA RECEPCION DE DATOS SERIALES*/
#INT_RDA2
void serial()
{
bit_clear(IFS1,8);

serial1=U2RXREG;
serial2=U2RXREG;
serial3=U2RXREG;
serial4=U2RXREG;

inicio=serial4;
bandera=2;
}

/**FUNCION ENCARGADA DE ENVIAR LOS DATOS A LOS DISPOSITIVOS ESCLAVOS PARA EL CONTROL DE
SENTIDO DE
GIRO DE LOS MOTORES MEDIANTE COMUNICACION I2C*/

void i2c_motor(int sentido,int address)
{
i2c_start();
i2c_write(address);
i2c_write(0xAB);
i2c_write(sentido);
i2c_stop();
}

/**FUNCION ENCARGADA DE ENVIAR LOS DATOS A LOS DISPOSITIVOS ESCLAVOS PARA EL CONTROL DE LA
VELOCIDAD DE
GIRO DE LOS MOTORES MEDIANTE COMUNICACION I2C*/

void i2c_motor_velocidad(int address, int velocidad)
{
i2c_start();
i2c_write(address);
i2c_write(0xAC);
i2c_write(velocidad);
i2c_stop();
}

/**FUNCION EMPLEADA PARA RECOPILAR LOS DATOS ENTREGADOR POR LA BRUJULA MAGNETICA
MEDIANTE COMUNICACION I2C*/
```

```
void brujula()
{
i2c_start();
i2c_write(0xC0);
i2c_write(0x01);

delay_us(100);

i2c_start();
i2c_write(0xC1);
brujula_dummie=i2c_read(1);
brujula1=i2c_read(1);
brujula2=i2c_read(0);
i2c_stop();

pos_brujula=(brujula1*256)+brujula2;
pos_brujula=pos_brujula/10;
}

/**FUNCION EMPLEADA PARA ENVIAR LOS DATOS AL PC VIA PROTOCOLO SERIAL*/

void envio(int sentido,int xa,int ya,int xb, int yb)
{
putc(0xAA);
putc(sentido);
putc(ie);
putc(je);
putc(ue);
putc(vb);
putc(sentido_g);
putc(coeficiente1);
putc(mx);
putc(my);
putc(variable[0]);
putc(variable[1]);
putc(variable[2]);
putc(variable[3]);
putc(variable_2[0]);
putc(variable_2[1]);
putc(variable_2[2]);
putc(variable_2[3]);
putc(variable_3[0]);
putc(variable_3[1]);
putc(variable_3[2]);
putc(variable_3[3]);
}

/**FUNCION EMPLEADA PARA OBTENER LA LECTURA DE LOS CODIFICADORES UBICADOS EN CADA
DISPOSITIVO ESCLAVO VIA I2C*/

int16 leer_encoder(int address)
{
int16 encoder;

i2c_start();
i2c_write(address);
i2c_write(0xAD);

i2c_start();
i2c_write(address+1);
dato_rcv[0]=i2c_read(1);
dato_rcv[1]=i2c_read(0);
i2c_stop();

encoder=((dato_rcv[1]*256)+dato_rcv[0]);
return encoder;
}

/**FUNCION EMPLEADA PARA DETERMINAR EL DESPLAZAMIENTO LINEAL DE CADA RUEDA*/

float recorrer(float rec1,float rec2,int lado)
{
float recorrido1;
```

```
recorrido1=rec1-rec2;

if(recorrido1>32767) recorrido1=-(65536-recorrido1);
if(recorrido1<-32768) recorrido1=(65536+recorrido1);

if(lado==1)recorrido1=(recorrido1*34.5)/8184;
else recorrido1=(recorrido1*36.1)/8384;

return recorrido1;
}

/**FUNCION EMPLEADA PARA LLEVAR A CABO EL DESPLAZAMIENTO MATRICIAL DE LOS DATOS EMPLEADOS PARA LA RECONSTRUCCION DEL ENTORNO*/

void mover_matriz(signed int movx,signed int movy)
{
signed int r=0,s=0;

if (movx >= 0)
{
if (movy <= 0)
{
for (r = 0; r <= 32 - movx; r++)
{
for (s = 0; s <= 32+movy; s++)
{
matriz[r][s] = matriz[r + movx][s-movy];

}
}

for (r = 0; r <= 32; r++)
{
for (s =33+movy; s <=32; s++)
{
matriz[r][s] = 0;
}
}
for (r = 33 - movx; r <= 32; r++)
{
for (s = 0; s <= 32; s++)
{
matriz[r][s] = 0;
}
}
}
else
{
for (r = 0; r <= 32 - movx; r++)
{
for (s = 32; s >= movy; s--)
{
matriz[r][s] = matriz[r + movx][s - movy];
}
}

for (r = 0; r <= 32; r++)
{
for (s =movy-1; s >= 32; s--)
{
matriz[r][s] = 0;

}
}
for (r = 33 - movx; r <= 32; r++)
{
for (s = 0; s <= 32; s++)
{
matriz[r][s] = 0;
}
}
```

```
}

}
else
{
if (movy <= 0)
{
for (r = 32; r >= -movx; r--)
{
for (s = 0; s <= 32+movy; s++)
{
matriz[r][s] = matriz[r + movx][s - movy];
}
}

for (r = 0; r <= 32; r++)
{
for (s = 33+movy; s <=32; s++)
{
matriz[r][s] = 0;
}
}
for (r = 0; r <= -movx - 1; r++)
{
for (s = 0; s <= 32; s++)
{
matriz[r][s] = 0;
}
}
}
else
{
for (r = 32; r >= - movx; r--)
{
for (s = 32; s >= movy; s--)
{
matriz[r][s] = matriz[r + movx][s - movy];
}
}

for (r = 0; r <= 32; r++)
{
for (s =movy-1; s >= 0; s--)
{
matriz[r][s] = 0;
}
}
for (r =0; r <= -movx; r++)
{
for (s = 0; s <= 32; s++)
{
matriz[r][s] = 0;
}
}
}
}
}

/**FUNCION EMPLEADA PARA ESTIMAR LA POSICION DEL VEHICULO A PARTIR DEL DESPLAZAMIENTO LINEAL DE CADA RUEDA*/

void posicion_vehiculo(float reci,float recd)
{
float compx=0,compy=0,bruj_p=0;
float delta_celda_x=0,delta_celda_y=0,ang_pos=0;
mx=0;
my=0;
int8 *p;
int8 l=0;
int8 *q;
int8 t=0;
int8 *q1;
int8 t1=0;
```

```
delta_s=(recd+reci)/2;
delta_ang=(recd-reci)/b;



ang_pos=ang_actual+(delta_ang/2);
ang_actual=ang_actual+delta_ang;
bruj_p=ang_actual*180/PI;

compx=delta_s*(cos(ang_pos));
compy=delta_s*(sin(ang_pos));

carro_x=carro_x+compx;
carro_y=carro_y+compy;

distancia_objetivo=sqrt(((carro_x-objetivo_x)*(carro_x-objetivo_x))+((carro_y-objetivo_y)*(carro_y-objetivo_y)));

celda_x1=carro_x/10;
celda_y1=carro_y/10;

celda_x1=(int)(celda_x1);
celda_y1=(int)(celda_y1);

delta_celda_x=celda_x1-carro_xant;
delta_celda_y=celda_y1-carro_yant;

p=&carro_x;
for(l=0;l<=3;l++) variable[l]=*(p+l);

q=&carro_y;
for(t=0;t<=3;t++) variable_2[t]=*(q+t);

q1=&ang_actual;
for(t1=0;t1<=3;t1++) variable_3[t1]=*(q1+t1);



if((delta_celda_x>0 && delta_celda_x<=16)||(delta_celda_x<0 && delta_celda_x>=-16) || (delta_celda_y>0 &&
delta_celda_y<=16)||(delta_celda_y<0 && delta_celda_y>=-16))
{
mx=(signed int)delta_celda_x;
my=(signed int)delta_celda_y;
mover_matriz(mx,my);
}

envio(tarea,ie,je,ue,vb);
}

/**FUNCION EMPLEADA PARA ESTABLECER EL SENTIDO DE GIRO DEL VEHICULO PARA ALCANZAR LA
ORIENTACION DESEADA*/

void orientar()
{
carro_xant=(int)(carro_x/10);
carro_yant=(int)(carro_y/10);


recorrido_d=(float)leer_encoder(address_1);
recorrido_i=(float)leer_encoder(address_2);
rec_d=recorrer(recorrido_d,recorrido_ant_d,1);
recorrido_ant_d=recorrido_d;
rec_i=recorrer(recorrido_i,recorrido_ant_i,2);
recorrido_ant_i=recorrido_i;

posicion_vehiculo(rec_i,rec_d);

distancia_objetivo=sqrt(((carro_x-objetivo_x)*(carro_x-objetivo_x))+((carro_y-objetivo_y)*(carro_y-objetivo_y)));

if(distancia_objetivo<=35)
{
bit_clear(T2CON,15);
sentido_g=2;
```

```
inicio=0x0C;
i2c_motor(2,address_1);
i2c_motor(2,address_2);
delay_ms(20);
tarea=7;

}

else
{

pos_brujula=ang_actual*180/PI;
if(pos_brujula>360)pos_brujula=pos_brujula-360;
if(pos_brujula<0)pos_brujula=360+pos_brujula;

if((ABS(pos_brujula-orientacion_brujula)>5))
{

if((orientacion_brujula-pos_brujula>0 && orientacion_brujula-pos_brujula<180) || orientacion_brujula-pos_brujula<-180)
sentido_g=4;
else sentido_g=3;

if(sentido_g!=sentido_anterior)
{
i2c_motor(sentido_g,address_1);
i2c_motor(sentido_g,address_2);
}
sentido_anterior=sentido_g;
}
else
{
sentido_g=5;
i2c_motor_velocidad(address_1,vl1);
i2c_motor_velocidad(address_2,vl1);
i2c_motor(5,address_1);
i2c_motor(5,address_2);
sentido_anterior=5;
}
}
}

/**FUNCION EMPLEADA PARA DETERMINAR LA ORIENTACION QUE EL VEHICULO DEBE SEGUIR*/

void calculos()
{

int8 r=0;
int8 s=0;
vl=velmax;

float cos,sen,avr,ang_repulsion,si,st,vx,vy,ang_atraccion,delta_wall,atraccion_proy=0,atrp_x=0,atrp_y=0;

vx = (float)vl* Cos(ang_actual);
vy = (float)vl *Sin(ang_actual);

repulsion_x=0;
repulsion_y=0;

for(r=0;r<=32;r++)
{
for(s=0;s<=32;s++)
{
cont_giro++;
distancia_celda=sqrt(((16-r)*(16-r))+((16-s)*(16-s)));
if(cont_giro==100)
{
cont_giro=0;
}

if (distancia_celda != 0 && matriz[r][s]!=0)
{
repulsion_x = repulsion_x + ((-1)*(((float)matriz[r][s] * constante_repulsion / (distancia_celda * distancia_celda)) * ((r-16)
/ distancia_celda)));
```

```
repulsion_y = repulsion_y + ((-1)*(((float)matriz[r][s] * constante_repulsion / (distancia_celda * distancia_celda)) * ((16-s)
/ distancia_celda)));
}
}
}
repulsion=sqrt(((repulsion_x)*(repulsion_x))+((repulsion_y)*(repulsion_y)));
ang_repulsion=atan2(repulsion_y,repulsion_x)*180/PI;

if (repulsion != 0)
{
cos = ((vx * repulsion_x) + (vy * repulsion_y)) / (vl * repulsion);
avr = Acos(cos) * 180 / PI;
sen = Sin(avr * PI / 180);
if (cos < 0)
{
repulsion = (w * repulsion) + ((1 - w) * (repulsion * -cos));
kv = (1 - (2 * z)) / (2 * (1 - z));
kv = 1 / kv;
vl = ((z * vl) + ((1 - z) * (vl * (sen/kv))));

}
else
{
repulsion = (w * repulsion) + ((1 - w) * (repulsion * -cos));

vl =(vl/2)+(0.5 * vl * cos);

}

repulsion_x = repulsion*Cos(ang_repulsion * PI/180);
repulsion_y = repulsion*Sin(ang_repulsion * PI / 180);

}

ang_repulsion = Atan2(repulsion_y, repulsion_x) * 180 /PI;
distancia_objetivo=sqrt(((carro_x-objetivo_x)*(carro_x-objetivo_x))+((carro_y-objetivo_y)*(carro_y-objetivo_y)));
atraccion_x=constante_atraccion*((objetivo_x-carro_x)/distancia_objetivo);
atraccion_y=constante_atraccion*((objetivo_y-carro_y)/distancia_objetivo);
atraccion=sqrt((atraccion_x*atraccion_x)+(atraccion_y*atraccion_y));

ang_atraccion=Atan2(atraccion_y,atraccion_x)*180/PI;
orientacion_1=Atan2((atraccion_y + repulsion_y), (atraccion_x + repulsion_x))*180/PI;

f2=ang_repulsion;

delta_wall=ang_atraccion-orientacion_1;
if (delta_wall >= 360) delta_wall = delta_wall - 360;

if(ABS(delta_wall)>90)
{
atraccion_proy = ang_repulsion + 145;
atraccion_proy=atraccion_proy*PI/180;
atrp_x=repulsion*Cos(atraccion_proy);
atrp_y=repulsion*Sin(atraccion_proy);
orientacion_1 = Atan2((atrp_y + repulsion_y), (atrp_x + repulsion_x))*180/PI;
tarea=9;
}
else
{
tarea=8;
}
if (orientacion_1 > 180) orientacion_1 = -(360 - orientacion_1);

orientacion_2=ang_actual;
si = orientacion_1;

st = ((400 * 0.001 * si) + ((thao - (400 * 0.001)) * sta)) / thao;

orientacion_2 =st;
sta = st;

orientacion_brujula=st;
```

```
if(orientacion_brujula<0)orientacion_brujula=360+orientacion_brujula;

orientacion=(int)(orientacion*10);
orientacion=orientacion/10;

vl1=(int)(vl*100/46);
if(vl1<20)vl1=20;
}

/**CONFIGURACION DEL MODULO PWM EMPLEADO PARA EL MOVIMIENTO DEL SONAR DEL SISTMEA*/

void configuracion_pwm()
{
bit_clear(PTCON,0);
bit_clear(PTCON,1);
bit_clear(PTCON,2);
bit_set(PTCON,3);
bit_clear(PTCON,4);
bit_clear(PTCON,5);
bit_clear(PTCON,6);
bit_clear(PTCON,7);

bit_clear(PWMCON1,0);
bit_clear(PWMCON1,1);
bit_clear(PWMCON1,2);
bit_clear(PWMCON1,3);
bit_set(PWMCON1,4);
bit_clear(PWMCON1,5);
bit_clear(PWMCON1,6);
bit_clear(PWMCON1,7);
bit_set(PWMCON1,8);
bit_set(PWMCON1,9);

bit_set(PWMCON2,2);
bit_clear(PWMCON2,0);
bit_set(OVDCON,8);
bit_set(OVDCON,9);

PTPER=0x61A7;
PDC1=1150;

bit_set(PTCON,15);
}

/**FUNCION ENCARGADA DE CALCULAR EL DUTY CYCLE DE LA SEÑAL PWM PARA CADA POSICION DEL
SONAR*/

void duty_cicle()
{
if(direccion==0)
{
switch(posicion)
{
case 0:
PDC1=1833;
posicion=1;
angulo=-60;
opcion=0;
break;
case 1:
PDC1=2516;
posicion=2;
angulo=-30;
opcion=1;
break;
case 2:
PDC1=3200;
posicion=3;
angulo=0;
opcion=2;
break;
case 3:
PDC1=3883;
```

```
posicion=4;
angulo=30;
opcion=3;
break;
case 4:
PDC1=4566;
posicion=5;
angulo=60;
opcion=4;
break;
case 5:
PDC1=5250;
posicion=6;
angulo=90;
opcion=5;
break;
case 6:
PDC1=4566;
posicion=5;
direccion=1;
angulo=60;
opcion=6;
break;
}
}
else
{
switch(posicion)
{
case 1:
PDC1=1150;
posicion=0;
angulo=-90;
opcion=1;
direccion=0;
break;
case 2:
PDC1=1833;
posicion=1;
angulo=-60;
opcion=2;
break;
case 3:
PDC1=2516;
posicion=2;
angulo=-30;
opcion=0;
break;
case 4:
PDC1=3200;
posicion=3;
angulo=0;
opcion=4;
break;
case 5:
PDC1=3883;
posicion=4;
angulo=30;
opcion=5;
break;
}
}
}

/**FUNCION EMPLEADA PARA INTRODUCIR LOS DATOS DE CADA SENSOR EN LA MATRIZ PARA LA
RECONSTRUCCION DEL ENTORNO DETERMINANDO
LA CELDA CORRESPONDIENTE DE LA MISMA*/

void matriz_sensor()
{

float distanciax,distanciay,distanciax_1,distanciay_1,ang_sensor,ang_sensor1;
```

```
ang_sensor=((angulo)*PI/180)+ang_actual;
distanciax=0;
distanciay=0;

if(distancia<=160 && distancia>10)
{
distanciax=((float)distancia*cos(ang_sensor));
distanciay=((float)distancia*sin(ang_sensor));
ic=distanciax/10;
jc=distanciay/10;
ic=16+ic;
jc=16-jc;

ie=(int)ic;
je=(int)jc;

matriz[ie][je]=matriz[ie][je]+1;
coeficiente=matriz[ie][je];
}
else
{
ie=34;
je=34;
}

distanciax=0;
distanciay=0;

ang_sensor1=((angulo+180)*PI/180)+ang_actual;

if(distancia1<=160 && distancia1>10)
{
distanciax_1=((float)distancia1*cos(ang_sensor1));
distanciay_1=((float)distancia1*sin(ang_sensor1));
uc=distanciax_1/10;
vc=distanciay_1/10;

uc=16+uc;
vc=16-vc;

ue=(int)uc;
vb=(int)vc;

matriz[ue][vb]=matriz[ue][vb]+1;
coeficiente1=matriz[ue][vb];
}
else
{
ue=34;
vb=34;
}
}

/**FUNCION EMPLEADA PARA RECOPILAR LA INFORMACION ENTREGADA POR LOS SENSORES ULTRASONICOS MEDIANTE PROTOCOLO I2C*/

int16 sensor_i2c(int address)
{
int16 dis;

i2c_start();
i2c_write(address);
i2c_write(0x00);
i2c_write(0x51);
i2c_stop();

delay_ms(80);

i2c_start();
i2c_write(address);
i2c_write(0x01);
i2c_start();
i2c_write(address+1);
```

```
dummie=i2c_read(1);
dato=i2c_read(1);
dato1=i2c_read(0);
i2c_stop();


dis=(256*dato)+dato1;

return dis;
}

/**FUNCION ENCARGADA DE GESTIONAR LOS COMANDOS DE CONTROL PROVENIENTES DEL PC*/

void entrada_serial()
{

if(inicio==0x0B)
{
orientacion_anterior=referencia;
orientacion_inicial=0;

inicio=0xAB;
tarea=1;

for(iq=1;iq<=18;iq++)
{
distancia=sensor_i2c(0xE0);
distancia1=sensor_i2c(0xE2);
matriz_sensor();

duty_cicle();
delay_ms(100);
}

bit_set(T4CON,5);
bit_set(T4CON,4);
bit_set(T2CON,15);
bit_set(PWMCON1,4);
}
if(inicio==0x0C)
{
i2c_motor(2,address_1);
i2c_motor(2,address_2);
bit_clear(T4CON,15);
bit_clear(T2CON,15);
}
if(inicio==0xCD)
{
objetivo_x=(signed int16)(serial3*256+serial2);
if(serial1==1)objetivo_x=(signed int16)(objetivo_x*(-1));

tarea=2;
}
if(inicio==0xDE)
{
objetivo_y=(signed int16)(serial3*256+serial2);
if(serial1==1)objetivo_y=(signed int16)(objetivo_y*(-1));
tarea=3;
}
if(inicio==0xEF)
{
velmax=(float)serial2;
tarea=4;
}
if(inicio==0x2A)
{
constante_repulsion=(int)serial2;
constante_atraccion=(int)serial3;
tarea=5;
}
if(inicio==0x2B)
{
w =(float)0.015625*serial2 ;
```

```
z = (float)0.0078125*serial3;
thao =(float)( 0.4 + (float)0.01875 * serial1);
tarea=6;
}
bandera=0;
}

/**FUNCION PRINCIPAL EN LA QUE SE ASIGNAN LOS VALORES INICIALES A LAS VARIABLES DEL SISTEMA*/

void main()
{
bit_set(IPC6,0);
bit_set(IPC6,1);
bit_set(IPC6,2);

bit_set(IPC5,4);
bit_set(IPC5,5);
bit_clear(IPC5,6);

bit_set(IPC1,8);
bit_set(IPC1,9);
bit_clear(IPC1,10);

bit_set(U2STA,7);
bit_set(U2STA,6);



bit_set(T2CON,5);
bit_set(T2CON,4);
bit_clear(T2CON,1);

bit_set(T4CON,5);
bit_set(T4CON,4);
bit_clear(T4CON,1);

bit_clear(INTCON1,15);

PR2=31248;
PR4=23438;

address_1=0x70;
address_2=0x60;
dato2=0;
dato3=0;
contador=0;
brujula1=0;
brujula2=0;
constante_repulsion=12;
coeficiente=0;
direccion=0;
posicion=0;
opcion=0;
distancia_celda=0;
repulsion_x=0;
repulsion_y=0;
repulsion=0;
posicion_x=0;
posicion_y=0;
objetivo_x=100;
objetivo_y=0;
constante_atraccion=3;
fuerza_x=0;
fuerza_y=0;
orientacion=0;
orientacion_1=0;
orientacion_2=0;
orientacion_brujula=0;
angulo=-90;
inicio=0;
w=0.5;
z=0.2;
thao=0.8;
```

```
//z=0.2;
kv=0;
sta=0;
referencia=0;
carro_x=0;
carro_y=0;
carro_xant=0;
carro_yant=0;
celda_x1=0;
celda_y1=0;
bandera=0;
recorrido_i=0;
recorrido_d=0;
recorrido_ant_d=0;
recorrido_ant_i=0;
rec_d=0;
rec_i=0;
carro_x=0;
carro_y=0;
celda_x1=0;
celda_y1=0;
carro_xant=0;
carro_yant=0;
distancia_objetivo=0;
giro=0;
ang_actual=0;
delta_ang=0;
delta_angulo=0;
delta_s=0;
b=25;
dato=0;
bandera=0;
bandera1=0;
ie=0;
je=0;
ic=0;
jc=0;
uc=0;
vc=0;
ue=0;
ve=0;
vb=0;
distancia=0;
distancia1=0;
actuali=0;
actuald=0;
anteriori=0;
anteriord=0;
deltad=0;
delta_d=0;
delta_i=0;
sentido_anterior=0;
cuenta_matriz=0;
brujula_anterior=0;
brujula_actual=0;
tarea=7;
mx=0;
my=0;
diana=0;
vl=20;
vl1=50;
cont_giro=0;
pos_brujula=0;
sentido_g=0;
coeficiente=0;
coeficiente1=0;
serial1=0;
serial2=0;
serial3=0;
serial4=0;
f1=0;
velmax=20;
```

```
configuracion_pwm();

bit_clear(IEC1,9);
bit_set(IEC1,8);
bit_set(IEC1,5);
bit_set(IEC0,6);

output_high(PIN_B0);
delay_ms(1000);
output_low(PIN_B0);
delay_ms(1000);

/**CICLO DE EJECUCION CONTINUA DEL PROGRAMA*/

while(true)
{
entrada_serial();
while(inicio==0xAB)
{
switch(bandera)
{
case 1:
bandera=0;

calculos();
output_toggle(PIN_B0);
orientar();

distancia=sensor_i2c(0xE0);
distancia1=sensor_i2c(0xE2);
matriz_sensor();

duty_cicle();

TMR2=0x0000;
bit_set(T2CON,5);
bit_clear(T2CON,4);
bit_set(T2CON,15);
break;
case 2:
entrada_serial();
bandera=0;
break;
case 3:
bandera=0;
break;
}

orientar();
}

anteriord=actuald;
anteriori=actuali;

carro_xant=(int)(carro_x/10);
carro_yant=(int)(carro_y/10);

recorrido_d=(float)leer_encoder(address_1);
recorrido_i=(float)leer_encoder(address_2);
rec_d=recorrer(recorrido_d,recorrido_ant_d,1);
recorrido_ant_d=recorrido_d;
rec_i=recorrer(recorrido_i,recorrido_ant_i,2);
recorrido_ant_i=recorrido_i;

posicion_vehiculo(rec_i,rec_d);
envio(tarea,serial1,serial2,serial3,serial4);
}
}
```

ANEXO B Código del micro-controlador esclavo 1 del sistema de navegación.

```
/*****************************CODIGO DEL MICRO-CONTROLADOR ESCLAVO 1************************************
Este código es empleado para llevar a cabo la recopilación de los datos concernientes al codificador  asociado a la
rueda derecha del vehículo y el control de giro del motor correspondiente a la misma. El dispositivo empleado
corresponde a la referencia DSPic 30f4011 y se encuentra configurado en modo esclavo del bus i2c.*/


/**LIBRERIAS**/

#include <30f4011.h>
#include <math.h>
#include <stdlib.h>


/**CONFIGURACION DEL DISPOSITIVO**/

#fuses PR,XT_PLL8,NOPROTECT,NOCKSFSM,NOWDT,DEBUG,NOWRT,COE,PUT16,PWMPIN
#define Device_SDA PIN_F2
#define Device_SCL  PIN_F3
#use delay(clock=80000000)
#use i2c(slave, slow, sda=device_SDA, scl=device_SCL,force_hw)

/**ASIGNACION DE LOS REGISTROS DEL DISPOSITIVO A VARIBLES EMPLEADAS POR EL CODIGO**/

#BYTE I2CRCV=0x0200
#WORD I2CTRN=0x0202
#WORD I2CCON=0x0206
#WORD I2CSTAT=0x0208
#BYTE I2CADD=0x020A
#WORD IFS0=0x0084
#BYTE I2CBRG=0x0204
#WORD QEICON=0x0122
#WORD DFLTCON=0x0124
#WORD POSCNT=0x0126
#WORD MAXCNT=0x0128
#WORD PTCON = 0x01C0
#WORD PWMCON1 = 0x01C8
#WORD PWMCON2 = 0x01CA
#WORD OVDCON = 0x01D4
#WORD PDC1 = 0x01D6
#WORD PDC2 = 0x01D8
#WORD PDC3 = 0x01DA
#WORD PTPER = 0x01C4

/**DETERMINACION DE LAS VARIABLES A EMPLEAR**/

int cont,dato,sentido,bandera_int;
int incoming,vel;
float velocidad,cuenta,vl,vl1;
int16 pasos;

int8 variable[4]={};
int8 encoder[2]={};

/**OPERADOR EMPLEADO PARA LA RECONSTRUCCION DE VARIABLES FLOTANTES DE 32 BITS**/
union joiner
{
int8 b[4];
float fp;
}combine;

/**INTERRUPCION GENERADA POR EL MODULO I2C**/

#INT_SI2C
void interrupcion_i2c()
{
bit_clear(IFS0,13);
bandera_int=1;
}
```

```
/**FUNCION QUE GESTIONA LOS COMANDOS RECIBIDOS VIA I2C**/

void i2c()
{
int8 *p;
int8 *e;
int8 m=0;
int8 l=0;
int tx[4]={};
int cont1,bandera;

bit_clear(I2CCON,12);

if(bit_test(I2CSTAT,2)==0)
{

if(bit_test(I2CSTAT,5)==0 && bit_test(I2CSTAT,1)==1)
{
dato=I2CRCV;
bit_clear(I2CSTAT,6);
cont1=0;
}

else if(bit_test(I2CSTAT,1)==1 && bit_test(I2CSTAT,5)==1)
{
incoming=I2CRCV;

switch(cont)
{
case 0:
incoming=I2CRCV;
if(incoming==0xAB)
{

bandera=1;
cont++;
}
else
{
if(incoming==0xAC) bandera=2,cont++;
else
{
bandera=3;
cont=0;
cont1=1;
}
}
break;
case 1:

if(bandera==1) cont=0,cont1=1,sentido=incoming;
else
{
if (bandera==2) cont=0,cont1=1,vel=incoming;
else cont++,tx[0]=incoming;
}
break;
case 2:

cont++;
tx[1]=incoming;
break;
case 3:
cont++;
tx[2]=incoming;
break;
case 4:
cont=0;
tx[3]=incoming;
cont1=1;
break;
}
```

```
if (cont1==1)
{
if(bandera==1)
{
output_toggle(PIN_B0);
switch(sentido)
{
case 2:
bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
break;
case 3:
PDC1=9000;
bit_clear(PWMCON1,0);
bit_set(PWMCON1,4);
break;
case 4:
PDC1=9000;
bit_clear(PWMCON1,4);
bit_set(PWMCON1,0);
break;
case 5:
bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
bit_set(PWMCON1,0);
bit_clear(PWMCON1,4);
break;
}
cont1=0;
}
else
{
if(bandera==2)
{
if(vel<5) vel=5;
vl1=((float)vel*49998/100);
PDC1=(int16)vl1;
cont1=0;
}
else
{
pasos=POSCNT;
e=&pasos;
for(m=0;m<=1;m++) encoder[m]=*(e+m);

cont1=0;
}
}
}
}
}
if(bit_test(I2CSTAT,2)==1)
{
i2c_write(encoder[0]);
while(bit_test(I2CSTAT,0)==1)
{
}
i2c_write(encoder[1]);
while(bit_test(I2CSTAT,0)==1)
{
}
i2c_write(variable[3]);
while(bit_test(I2CSTAT,0)==1)
{
}
}
bandera_int=0;
bit_set(I2CCON,12);
}
```

```
/**CONFIGURACION DEL MODULO PWM ENCARGADO DEL CONTROL DE LOS MOTORES**/

void configuracion_pwm()
{
bit_clear(PTCON,0);
bit_clear(PTCON,1);
bit_clear(PTCON,2);
bit_set(PTCON,3);
bit_clear(PTCON,4);
bit_clear(PTCON,5);
bit_clear(PTCON,6);
bit_clear(PTCON,7);


bit_clear(PWMCON1,0);
bit_clear(PWMCON1,1);
bit_clear(PWMCON1,2);
bit_clear(PWMCON1,3);
bit_set(PWMCON1,4);
bit_clear(PWMCON1,5);
bit_clear(PWMCON1,6);
bit_clear(PWMCON1,7);
bit_set(PWMCON1,8);
//bit_set(PWMCON1,9);

bit_set(PWMCON2,2);
bit_clear(PWMCON2,0);
bit_set(OVDCON,8);
bit_set(OVDCON,9);

PTPER=0x61A7;
PDC1=4500;
PDC2=50;

bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
bit_set(PTCON,15);
}

/**CONFIGURACION DEL CODIFICADOR EN CUADRATURA**/

void configuracion_QEI()
{
bit_set(QEICON,0);
bit_clear(QEICON,1);
bit_clear(QEICON,3);
bit_clear(QEICON,4);
bit_clear(QEICON,5);
bit_clear(QEICON,6);
bit_clear(QEICON,7);
bit_set(QEICON,8);
bit_set(QEICON,9);
bit_set(QEICON,10);
bit_set(DFLTCON,8);
bit_set(DFLTCON,7);
bit_set(DFLTCON,4);
bit_clear(DFLTCON,5);
bit_clear(DFLTCON,6);
POSCNT=0;
MAXCNT=0xFFFF;
}

/**FUNCION PRINCIPAL DONDE SE ASIGNAN LOS VALORES INICIALES A LAS VARIABLES EMPLEADAS**/

void main()
{
set_tris_F(0x0C);
output_e(0x00);
I2CADD=0x38;
cont=0;
incoming=0;
dato=0;
velocidad=0;
```

```
cuenta=0;
pasos=0;
vel=0;
vl=0;
vl1=0;

configuracion_QEI();
configuracion_pwm();
delay_ms(200);
output_high(PIN_B0);
delay_ms(500);
output_low(PIN_B0);

bit_set(I2CCON,6);
bit_clear(I2CCON,7);

enable_interrupts(INT_SI2C);

/**CICLO PRINCIPAL DE EJECUCION**/

while(true)
{
if(bandera_int==1)
{
i2c();
}
}
}
```

ANEXO C. Código del micro-controlador esclavo 2 del sistema de navegación.

```
/*********************CODIGO DEL MICRO-CONTROLADOR ESCLAVO 2*************************************************
Este código es empleado para llevar a cabo la recopilación de los datos concernientes al codificador asociado a la rueda izquierda del vehículo y el control de giro del motor correspondiente a la misma. El dispositivo empleado corresponde a la referencia DSPic 30f4011 y se encuentra configurado en modo esclavo.
del bus i2c.*/


/**LIBRERIAS**/

#include <30f4011.h>
#include <math.h>
#include <stdlib.h>

/**CONFIGURACION DEL DISPOSITIVO**/

#fuses PR,XT_PLL8,NOPROTECT,NOCKSFSM,NOWDT,DEBUG,NOWRT,COE,PWMPIN
#define Device_SDA PIN_F2
#define Device_SCL  PIN_F3
#use delay(clock=80000000)
#use i2c(slave, slow, sda=device_SDA, scl=device_SCL,force_hw)

/**ASIGNACION DE LOS REGISTROS DEL DISPOSITIVO A VARIBLES EMPLEADAS POR EL CODIGO**/

#BYTE I2CRCV=0x0200
#WORD I2CTRN=0x0202
#WORD I2CCON=0x0206
#WORD I2CSTAT=0x0208
#BYTE I2CADD=0x020A
#WORD IFS0=0x0084
#BYTE I2CBRG=0x0204
#WORD QEICON=0x0122
#WORD DFLTCON=0x0124
#WORD POSCNT=0x0126
#WORD MAXCNT=0x0128
#WORD PTCON = 0x01C0
#WORD PWMCON1 = 0x01C8
#WORD PWMCON2 = 0x01CA
#WORD OVDCON = 0x01D4
#WORD PDC1 = 0x01D6
#WORD PDC2 = 0x01D8
#WORD PDC3 = 0x01DA
#WORD PTPER = 0x01C4

/**DETERMINACION DE LAS VARIABLES A EMPLEAR**/

int cont,dato,sentido,bandera_int;
int incoming,vel;
float velocidad,cuenta,vl,vl1;
int16 pasos;

int8 variable[4]={};
int8 encoder[2]={};

/**OPERADOR EMPLEADO PARA LA RECONSTRUCCION DE VARIABLES FLOTANTES DE 32 BITS**/

union joiner
{
int8 b[4];
float fp;
}combine;

/**INTERRUPCION GENERADA POR EL MODULO I2C**/

#INT_SI2C
void interrupcion_i2c()
{
bit_clear(IFS0,13);
bandera_int=1;
}
```

```
/**FUNCION QUE GESTIONA LOS COMANDOS RECIBIDOS VIA I2C**/

void i2c()
{
int8 *p;
int8 *e;
int8 m=0;
int8 l=0;
int tx[4]={};
int cont1,bandera;

bit_clear(I2CCON,12);

if(bit_test(I2CSTAT,2)==0)
{

if(bit_test(I2CSTAT,5)==0 && bit_test(I2CSTAT,1)==1)
{


dato=I2CRCV;
bit_clear(I2CSTAT,6);
cont1=0;
}

else if(bit_test(I2CSTAT,1)==1 && bit_test(I2CSTAT,5)==1)
{
incoming=I2CRCV;

switch(cont)
{
case 0:
incoming=I2CRCV;
if(incoming==0xAB)
{

bandera=1;
cont++;
}
else
{
if(incoming==0xAC) bandera=2,cont++;
else
{
bandera=3;
cont=0;
cont1=1;
}
}
break;
case 1:

if(bandera==1) cont=0,cont1=1,sentido=incoming;
else
{
if (bandera==2) cont=0,cont1=1,vel=incoming;
else cont++,tx[0]=incoming;
}


break;
case 2:

cont++;
tx[1]=incoming;
break;
case 3:
cont++;
tx[2]=incoming;
break;
case 4:
cont=0;
tx[3]=incoming;
```

```
cont1=1;
break;
}

if (cont1==1)
{
if(bandera==1)
{
output_toggle(PIN_B0);
switch(sentido)
{
case 2:
bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
break;
case 3:
PDC1=7000;
bit_set(PWMCON1,0);
bit_clear(PWMCON1,4);
break;
case 4:
PDC1=7000;
bit_set(PWMCON1,4);
bit_clear(PWMCON1,0);
break;
case 5:
bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
bit_set(PWMCON1,0);
bit_clear(PWMCON1,4);
break;
}

cont1=0;
}
else
{
if(bandera==2)
{
if(vel<5) vel=5;
vl1=((float)vel*49998/100);
PDC1=(int16)vl1;
cont1=0;
}
else
{
pasos=POSCNT;

e=&pasos;
for(m=0;m<=1;m++) encoder[m]=*(e+m);

cont1=0;
}
}

}

}
}
if(bit_test(I2CSTAT,2)==1)
{
i2c_write(encoder[0]);
while(bit_test(I2CSTAT,0)==1)
{
}
i2c_write(encoder[1]);
while(bit_test(I2CSTAT,0)==1)
{
}
}

bandera_int=0;
bit_set(I2CCON,12);
```

```
}

/**CONFIGURACION DEL MODULO PWM ENCARGADO DEL CONTROL DE LOS MOTORES**/

void configuracion_pwm()
{
bit_clear(PTCON,0);
bit_clear(PTCON,1);
bit_clear(PTCON,2);
bit_set(PTCON,3);
bit_clear(PTCON,4);
bit_clear(PTCON,5);
bit_clear(PTCON,6);
bit_clear(PTCON,7);

bit_clear(PWMCON1,0);
bit_clear(PWMCON1,1);
bit_clear(PWMCON1,2);
bit_clear(PWMCON1,3);
bit_set(PWMCON1,4);
bit_clear(PWMCON1,5);
bit_clear(PWMCON1,6);
bit_clear(PWMCON1,7);
bit_set(PWMCON1,8);

bit_set(PWMCON2,2);
bit_clear(PWMCON2,0);
bit_set(OVDCON,8);
bit_set(OVDCON,9);


PTPER=0x61A7;

PDC1=4500;
PDC2=50;

bit_clear(PWMCON1,0);
bit_clear(PWMCON1,4);
bit_set(PTCON,15);

}

/**CONFIGURACION DEL CODIFICADOR EN CUADRATURA**/

void configuracion_QEI()
{
bit_set(QEICON,0);
bit_clear(QEICON,1);
bit_clear(QEICON,3);
bit_clear(QEICON,4);
bit_clear(QEICON,5);
bit_clear(QEICON,6);
bit_clear(QEICON,7);
bit_set(QEICON,8);
bit_set(QEICON,9);
bit_set(QEICON,10);
bit_set(DFLTCON,8);
bit_set(DFLTCON,7);
bit_clear(DFLTCON,4);
bit_set(DFLTCON,5);
bit_set(DFLTCON,6);
POSCNT=0;
MAXCNT=0xFFFF;
}

/**FUNCION PRINCIPAL DONDE SE ASIGNAN LOS VALORES INICIALES A LAS VARIABLES EMPLEADAS**/

void main()
{
set_tris_F(0x0C);
output_e(0x00);
I2CADD=0x30;
cont=0;
```

```
incoming=0;
dato=0;
velocidad=0;
cuenta=0;
pasos=0;
vel=0;
vl=0;
vl1=0;

configuracion_QEI();
configuracion_pwm();
delay_ms(500);
output_high(PIN_B0);
delay_ms(500);
output_low(PIN_B0);
bit_set(I2CCON,6);
bit_clear(I2CCON,7);

enable_interrupts(INT_SI2C);

/**CICLO PRINCIPAL DE EJECUCION**/

while(true)
{
if(bandera_int==1)
{
i2c();
}
}
}
```

ANEXO D Código del micro-controlador empleado para la adquisición de datos vía USB.

**********RECEPTOR_USB**********
Este código ha sido desarrollado con el fin de permitir enviar y recibir datos desde y hacia el PC mediante comunicación USB empleando un micro-controlador de referencia PIC 18f2550. La recepción de los datos se hace empleando un transmisor de radio frecuencia dotado de un módulo UART capaz de comunicarse con el micro-controlador empleado. El envío de los comandos de control es llevado a cabo de manera similar a la descrita anteriormente.**/

```
/**LIBRERIAS**/

#include <18F2550.h>
#include <stdlib.h>
#include <string.h>

/**CONFIGURACION DEL DISPOSITIVO**/

#fuses HSPLL,NOWDT,NOPROTECT,NOLVP,DEBUG,USBDIV,PLL5,CPUDIV1,VREGEN,NOPUT
#use delay(clock=48000000)
#use rs232(uart,baud=9600, xmit=PIN_C6, rcv=PIN_C7,PARITY=N,BITS=8)
//#use i2c(slave, slow, sda=PIN_B0, scl=PIN_B1,force_HW)

/**DESCRIPCION DEL MODULO USB**/

#define USB_HID_DEVICE     FALSE            //uso de las directivas HID deshabilitado
#define USB_EP1_TX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_RX_ENABLE  USB_ENABLE_BULK
#define USB_EP1_TX_SIZE    22               //tamaño de almacenamiento para cadenas a transmitir
#define USB_EP1_RX_SIZE    4                //tamaño de almacenamiento para cadenas a recibir

/**CONFIGURACION DEL MODULO USB**/

#include <pic18_usb.h>
#include <PicUSB.h>
#include <usb.c>
#include <input.c>

/**ASIGNACION DE LAS DIRECCIONES CORRESPONDIENTES A LOS REGISTROS DEL DISPOSITIVO A LAS VARIABLES EMPLEADAS POR EL CODIGO**/

#byte PIE1=0x0F9D
#byte PIR1=0x0F9E
#byte RCSTA=0x0FBA
#byte RCREG=0x0FAE
#byte TRISC=0x0F94


/**ASIGNACION DE LAS VARIABLES A EMPLEAR**/

int8 dato,cont,cont1,bandera;
int tx[22]={};
int rx[4]={};

/**INTERRUPCION GENERADA POR EL MODULO UART**/

#INT_RDA
void recepcion_datos()
{
bit_clear(PIR1,5);
dato=getc();

switch(cont)
{
case 0:
if(dato==0xAA)
{
cont=1;
cont1=0;
}
break;
```

```
case 1:
tx[0]=dato;
cont=2;
break;
case 2:
tx[1]=dato;
cont=3;
break;
case 3:
tx[2]=dato;
cont=4;
break;
case 4:
tx[3]=dato;
cont=5;
break;
case 5:
tx[4]=dato;
cont=6;
break;
case 6:
tx[5]=dato;
cont=7;
break;
case 7:
tx[6]=dato;
cont=8;
break;
case 8:
tx[7]=dato;
cont=9;
break;
case 9:
tx[8]=dato;
cont=10;
break;
case 10:
tx[9]=dato;
cont=11;
break;
case 11:
tx[10]=dato;
cont=12;
break;
case 12:
tx[11]=dato;
cont=13;
break;
case 13:
tx[12]=dato;
cont=14;
break;
case 14:
tx[13]=dato;
cont=15;
break;
case 15:
tx[14]=dato;
cont=16;
break;
case 16:
tx[15]=dato;
cont=17;
break;
case 17:
tx[16]=dato;
cont=18;
break;
case 18:
tx[17]=dato;
cont=19;
break;
case 19:
```

```
tx[18]=dato;
cont=20;
break;
case 20:
tx[19]=dato;
cont=21;
break;
case 21:
tx[20]=dato;
tx[21]=0xCC;
cont=0;
cont1=1;
break;
}
}

/**FUNCION PRINCIPAL DEL PROGRAMA**/

void main()
{

bit_set(TRISC,7);
bit_clear(TRISC,6);
bit_clear(PIE1,4);
cont=0;
dato=0;
bandera=1;
enable_interrupts(global);
enable_interrupts(INT_RDA);

usb_init();
usb_task();
usb_wait_for_enumeration();

/**CICLO PRINCIPAL DE OPERACION**/

while(true)
{
if(usb_kbhit(1))
{
output_high(PIN_B2);
usb_get_packet(1, rx, 4);
putc(rx[0]);
putc(rx[1]);
putc(rx[2]);
putc(rx[3]);
output_low(PIN_B2);
}
if(cont1==1)
{
usb_put_packet(1,tx,22,USB_DTS_TOGGLE);
cont1=0;
}
}
}
```

ANEXO E Código de la interfaz gráfica para la adquisición de datos.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using DWORD = System.UInt32;
using PVOID = System.IntPtr;

namespace usb
{
   unsafe public partial class Form1 : Form
  {
      uint contador,cont,cont1,tarea;
      uint resultado;
      uint posicion;
      int contador_celdas = 0;
      public int valor_max, valor_min;
      System.Drawing.Size t = new Size();
      PicUSBAPI usbapi = new PicUSBAPI();
      public uint contador2,x,y,coeficiente,u,v,coeficiente1,sentido;
      public Point pos_ant = new Point();
      public int datox,datoy,datou,datov;
      public string color_celda;
      public string filename = null;
      public uint[,] matriz = new uint[33, 33];
      Bitmap carro = new Bitmap(usb.Properties.Resources.carro1_copia);
      public float posicion_x, posicion_y,carro_x,carro_y,angulo_actual;

      float p_x, p_y;
      public bool validacion;

      public float[] trayectoria_x = new float[5000];
      public float[] trayectoria_y = new float[5000];
      int cont_tray;

      public string outputfilename;

      public Form1()
      {
         InitializeComponent();
         t.Width = 10;
         t.Height = 10;

       }
      public void grilla_funcion()
      {
         Point punto_inicio = new Point();
         Point punto_final = new Point();
         Graphics formGraphics = pictureBox1.CreateGraphics();
         Pen grilla = new Pen(System.Drawing.Color.White);
         grilla.Width = 1;
         Pen cono = new Pen(System.Drawing.Color.Blue);
         cono.Width = 1;
         Pen centro = new Pen(System.Drawing.Color.Red);
         punto_inicio.X = 0;
         punto_inicio.Y = 0;
         punto_final.X = 0;
         punto_final.Y = 400;


         for (int i = 0; i <= 32; i++)
         {
            punto_inicio.X = punto_inicio.X + 10;
            punto_final.X = punto_inicio.X;
            formGraphics.DrawLine(grilla, punto_inicio.X, punto_inicio.Y, punto_final.X, punto_final.Y);
```

```
    }

    punto_inicio.X = 0;
    punto_inicio.Y = 0;
    punto_final.X = 460;
    punto_final.Y = 0;

    for (int i = 0; i <= 32; i++)
    {
        punto_inicio.Y = punto_inicio.Y + 10;
        punto_final.Y = punto_inicio.Y;
        formGraphics.DrawLine(grilla, punto_inicio.X, punto_inicio.Y, punto_final.X, punto_final.Y);

    }
    grilla.Dispose();

    formGraphics.DrawLine(cono, 0, 209, 330, 121);
    formGraphics.DrawLine(cono, 0, 330, 330, 0);
    formGraphics.DrawLine(cono, 121, 330, 209, 0);
    formGraphics.DrawLine(centro, 165, 330, 165, 0);
    formGraphics.DrawLine(cono, 0, 121, 330, 209);
    formGraphics.DrawLine(cono, 0, 0, 330, 330);
    formGraphics.DrawLine(cono, 121, 0, 209, 330);
    cono.Dispose();

}
unsafe private void Send_Click(object sender, EventArgs e)
{
    pictureBox1.Refresh();
    byte* send = stackalloc byte[1];
    uint[] datos = new uint[4];
    contador = usbapi.identificacion();

    if (contador < 1)
    {
        MessageBox.Show("No hay ningun dispositivo conectado");
    }
    else
    {
        send[0] = 0x09;
        send[1] = 0x0A;
        send[2] = 0x0B;
        usbapi.SumaPIC(send[0],send[1],send[2]);
        label1.Text = Convert.ToString(send[0]);

     }
    posicion_x = 50;
    posicion_y = 300;

    cont_tray = 0;
    timer3.Start();
    timer2.Start();
}

private void Form1_Load(object sender, EventArgs e)
{


grilla_funcion();
posicion_x = 300;
posicion_y = 300;

}

private void timer1_Tick(object sender, EventArgs e)
{


    contador = usbapi.identificacion();

    if (contador > 0)
    {
```

```
        label2.ForeColor = Color.OliveDrab;
        ORIENTACION.ForeColor = Color.OliveDrab;
        POSICIONX.ForeColor = Color.OliveDrab;
        POSICIONY.ForeColor = Color.OliveDrab;
        TAREA.ForeColor = Color.OliveDrab;
        DEZPLAZAMIENTO_X.ForeColor = Color.OliveDrab;
        DESPLAZAMIENTO_Y.ForeColor = Color.OliveDrab;
        label7.ForeColor = Color.OliveDrab;
        label9.ForeColor = Color.OliveDrab;
        label12.ForeColor = Color.OliveDrab;
        label15.ForeColor = Color.OliveDrab;
        BOTON_INICIO.ForeColor = Color.OliveDrab;
        label17.ForeColor = Color.OliveDrab;
        label18.ForeColor = Color.OliveDrab;
        label19.ForeColor = Color.OliveDrab;
        label20.ForeColor = Color.OliveDrab;
        label2.Text = ("Dispositivo conectado");
        pictureBox3.Image = usb.Properties.Resources.boton_encendido;
    }
    else
    {
        ORIENTACION.ForeColor = Color.RoyalBlue;
        POSICIONX.ForeColor = Color.RoyalBlue;
        POSICIONY.ForeColor = Color.RoyalBlue;
        TAREA.ForeColor = Color.RoyalBlue;
        DEZPLAZAMIENTO_X.ForeColor = Color.RoyalBlue;
        DESPLAZAMIENTO_Y.ForeColor = Color.RoyalBlue;
        label7.ForeColor = Color.RoyalBlue;
        label9.ForeColor = Color.RoyalBlue;
        label12.ForeColor = Color.RoyalBlue;
        label15.ForeColor = Color.RoyalBlue;
        BOTON_INICIO.ForeColor = Color.RoyalBlue;
        label17.ForeColor = Color.RoyalBlue;
        label18.ForeColor = Color.RoyalBlue;
        label19.ForeColor = Color.RoyalBlue;
        label20.ForeColor = Color.RoyalBlue;
        label2.ForeColor = System.Drawing.Color.RoyalBlue;
        label2.Text = ("Sin conexión");
        pictureBox3.Image = usb.Properties.Resources.boton_apagado;
    }


}


void mover_matriz(int movx, int movy)
{


    if (movx >= 0)
    {
        if (movy <= 0)
        {
            for (int r = 0; r <= 32 - movx; r++)
            {
                for (int s = 0; s <= 32 + movy; s++)
                {
                    matriz[r, s] = matriz[r + movx, s - movy];

                }
            }

            for (int r = 0; r <= 32; r++)
            {
                for (int s = 33 + movy; s <= 32; s++)
                {
                    matriz[r, s] = 0;

                }
            }
            for (int r = 33 - movx; r <= 32; r++)
            {
                for (int s = 0; s <= 32; s++)
```

```
            {
                matriz[r, s] = 0;

            }
        }
    }
    else
    {
        for (int r = 0; r <= 32 - movx; r++)
        {
            for (int s = 32; s >= movy; s--)
            {
                matriz[r, s] = matriz[r + movx, s - movy];

            }
        }

        for (int r = 0; r <= 32; r++)
        {
            for (int s = movy - 1; s >= 0; s--)
            {
                matriz[r, s] = 0;

            }
        }
        for (int r = 33 - movx; r <= 32; r++)
        {
            for (int s = 0; s <= 32; s++)
            {
                matriz[r, s] = 0;

            }
        }

    }

}
else
{
    if (movy <= 0)
    {
        for (int r = 32; r >= -movx; r--)
        {
            for (int s = 0; s <= 32 + movy; s++)
            {
                matriz[r, s] = matriz[r + movx, s - movy];

            }
        }

        for (int r = 0; r <= 32; r++)
        {
            for (int s = 33 + movy; s <= 32; s++)
            {
                matriz[r, s] = 0;

            }
        }
        for (int r = 0; r <= -movx - 1; r++)
        {
            for (int s = 0; s <= 32; s++)
            {
                matriz[r, s] = 0;

            }
        }
    }
    else
    {
        for (int r = 32; r >= -movx; r--)
        {
            for (int s = 32; s >= movy; s--)
            {
```

```
                    matriz[r, s] = matriz[r + movx, s - movy];

                }
            }

            for (int r = 0; r <= 32; r++)
            {
                for (int s = movy - 1; s >= 0; s--)
                {
                    matriz[r, s] = 0;

                }
            }
            for (int r = 0; r <= -movx; r++)
            {
                for (int s = 0; s <= 32; s++)
                {
                    matriz[r, s] = 0;

                }
            }
        }
    }

}
 public void recibir()
 {
     uint[] trama = new uint[22];
     uint[] trama1 = new uint[22];
     byte[] bytes = new byte[4];
     byte[] bytes1 = new byte[4];
     byte[] bytes2 = new byte[4];
     double brujula,recorrido;
     uint distancia;
     sbyte movx,movy;

     movx = 0;
     movy = 0;

    trama = usbapi.ResultadoPIC();

    if (trama[21] == 0xCC)
    {

        resultado = ((trama[0] * 256) + trama[1]);
        tarea = trama[0];
    }
    else
    {
        tarea = 0;
    }


    if (tarea != 0  && tarea!=2 && tarea!=3 && tarea!=4 && tarea!=5 && tarea!=6 && tarea!=7)
    {
        x = trama[3];
        y = trama[4];
        u = trama[1];
        v = trama[2];
        if (x < 33 && y < 33)
        {
            matriz[x, y] = matriz[x, y] + 1;

        }
        if (u < 33 && v < 33)
        {
            matriz[u, v] = matriz[u, v] + 1;
        }
        movx = (sbyte)trama[7];
        movy = (sbyte)trama[8];
    }
```

```
    bytes[0] = (byte)trama[9];
    bytes[1] = (byte)trama[10];
    bytes[2] = (byte)trama[11];
    bytes[3] = (byte)trama[12];

    bytes1[0] = (byte)trama[13];
    bytes1[1] = (byte)trama[14];
    bytes1[2] = (byte)trama[15];
    bytes1[3] = (byte)trama[16];

    bytes2[0] = (byte)trama[17];
    bytes2[1] = (byte)trama[18];
    bytes2[2] = (byte)trama[19];
    bytes2[3] = (byte)trama[20];

    brujula = ((double)(trama[7] * 256 + (double)trama[8])/10);
    distancia = trama[1] * 256 + trama[2];
    carro_x = BitConverter.ToSingle(bytes,0);
    carro_y = BitConverter.ToSingle(bytes1, 0);
    angulo_actual = BitConverter.ToSingle(bytes2, 0);
    angulo_actual =angulo_actual * 180 / (float)Math.PI;
    label4.Text =carro_x.ToString();
    recorrido = (256 * trama[4]) + trama[3];
    sentido = trama[5];
  label10.Text = Convert.ToString(angulo_actual);
    label13.Text =carro_y.ToString();
    label6.Text = movy.ToString();
    label3.Text = movx.ToString();
    label8.Text = x.ToString();
    label11.Text = y.ToString();
    label14.Text = u.ToString();
    label16.Text = v.ToString();

    switch (tarea)
    {
        case 0:
            label5.Text = "";
            break;
        case 1:
            label5.Text = "INICIAR RECORRIDO";
            break;
        case 2:
            label5.Text = "RECEPCION OBJETIVO X";
            break;
        case 3:
            label5.Text = "RECEPCION OBJETIVO Y";
            break;
        case 4:
            label5.Text = "RECEPCION VELOCIDAD MAXIMA";
            break;
        case 5:
            label5.Text = "RECEPCION CONSTANTES \nDE REPULSION Y ATRACCION";
            break;
        case 6:
            label5.Text = "RECEPCION W, Z, THAO";
            break;
        case 7:
            label5.Text = "ESPERANDO INSTRUCCIONES";
            break;
        case 8:
            label5.Text = "NAVEGACION VFF";
            break;
        case 9:
            label5.Text = "NAVEGACION WFM";
            break;
    }



    if (movx < 10 && movx > -10 && movy < 10 && movy > -10)
    {
        if(movx!=0 || movy!=0)
```

```
            {
                mover_matriz((int)movx, (int)movy);
            }
        }
}

void celdas()
{
    datox = ((int)((x+1)*10) - 10);
    datoy = ((int)((y+1) * 10) - 10);
    datou = ((int)((u+1)*10) - 10);
    datov = ((int)((v+1) * 10) - 10);


}

private void timer2_Tick(object sender, EventArgs e)
{


    Point centro_celda = new Point();

    Rectangle celda = new Rectangle(centro_celda, t);
    Rectangle anterior = new Rectangle(pos_ant, t);
    Graphics formGraphics = pictureBox1.CreateGraphics();
    Graphics dibujar = pictureBox2.CreateGraphics();
    //SolidBrush color1 = new SolidBrush(Color.Red);
    Pen cuadrado = new Pen(System.Drawing.Color.Cyan);
    Pen lapiz = new Pen(System.Drawing.Color.Black);
    cuadrado.Width = 3;



    SolidBrush black = new SolidBrush(Color.Black);
    SolidBrush centro = new SolidBrush(Color.BlueViolet);


    recibir();

    if (tarea != 0)
    {
        pictureBox2.Refresh();

        for (int i = 0; i <= 32; i++)
        {
            for (int j = 0; j <= 32; j++)
            {
                celda.X = ((int)((i + 1) * 10) - 10);
                celda.Y = ((int)((j + 1) * 10) - 10);
                formGraphics.FillRectangle(black, celda);
            }

        }


        for (int i = 0; i <= 32; i++)
        {
            for (int j = 0; j <= 32; j++)
            {
                celda.X = ((int)((i + 1) * 10) - 10);
                celda.Y = ((int)((j + 1) * 10) - 10);
                coeficiente = matriz[i, j];
                if (coeficiente != 0)
                {
                    if (coeficiente <= 3)
                    {
                        SolidBrush verde = new SolidBrush(Color.Green);
                        formGraphics.FillRectangle(verde, celda);
                    }
                    else
                    {
                        if (coeficiente > 3 && coeficiente <= 6)
                        {
```

```
                    SolidBrush amarillo = new SolidBrush(Color.Yellow);
                    formGraphics.FillRectangle(amarillo, celda);
                }
                else
                {
                    SolidBrush rojo = new SolidBrush(Color.Red);
                    formGraphics.FillRectangle(rojo, celda);
                }
            }

        }

        pos_ant.X = celda.X;
        pos_ant.Y = celda.Y;
        p_x = posicion_x + carro_x;
        p_y = posicion_y - carro_y;
        if (p_x < 650 && p_y < 650 && p_x > 0 && p_y > 0 && angulo_actual < 720  && angulo_actual>-720)
        {
            dibujar.TranslateTransform(p_x, p_y);
            dibujar.RotateTransform((float)-(angulo_actual));
            dibujar.DrawImage(carro, -20, -18);
            dibujar.ResetTransform();


            for (int ic = 1; ic <= cont_tray-1; ic++)
            {
                dibujar.DrawLine(lapiz, trayectoria_x[ic - 1], trayectoria_y[ic - 1], trayectoria_x[ic], trayectoria_y[ic]);
            }

        }
      }
    }
  }
}

private void Calibrar_Click(object sender, EventArgs e)
{
  byte* send = stackalloc byte[1];
  uint[] datos = new uint[4];
  contador = usbapi.identificacion();
  for (int i = 0; i <= 32; i++)
  {
    for(int j=0;j<=32;j++)
    {
      matriz[i, j] = 0;
    }
  }

    if (contador < 1)
    {
      MessageBox.Show("No hay ningun dispositivo conectado");
    }
    else
    {
      send[0] = 0x09;
      send[1] = 0x0A;
      send[2] = 0x0C;
      usbapi.SumaPIC(send[0], send[1], send[2]);
      label1.Text = Convert.ToString(send[0]);

    }
}

private void Cargar_Entorno_Click(object sender, EventArgs e)
{
  OpenFileDialog ld = new OpenFileDialog();
  ld.ShowDialog();

  if (ld.FileName != "")
  {
    pictureBox2.Image = Image.FromFile(ld.FileName);

  }
```

```
}

private void BORRAR_Click(object sender, EventArgs e)
{
    for (int i = 0; i <= 32; i++)
    {
        for (int j = 0; j <= 32; j++)
        {
            matriz[i, j] = 0;
        }
    }
    for (int i = 0; i < 5000; i++)
    {
        trayectoria_x[i] = 0;
    }
    for (int i = 0; i < 5000; i++)
    {
        trayectoria_y[i] = 0;
    }

    pictureBox1.Refresh();
    pictureBox2.Refresh();

}


public bool validar_datos(int caja_texto, string opcion)
{
    int dato_envio, iv;
    bool num = false;
    bool vacio = false;
    validacion = false;
    dato_envio = 0;
    string variable_ing = "";

    switch (opcion)
    {
        case "REPULSION":
            valor_max = 255;
            valor_min = 0;
            variable_ing = "REPULSION";
            break;
        case "ATRACCION":
            valor_max = 255;
            valor_min = 0;
            variable_ing = "ATRACCION";
            break;
        case "W":
            valor_max = 63;
            valor_min = 0;
            variable_ing = "W";
            break;
        case "Z":
            valor_max = 63;
            valor_min = 0;
            variable_ing = "Z";
            break;
        case "THAO":
            valor_max = 31;
            valor_min = 0;
            variable_ing = "THAO";
            break;
        case "VELOCIDAD":
            valor_max = 45;
            valor_min = 0;
            variable_ing = "VELOCIDAD";
            break;
        case "OBJETIVOX":
            valor_max = 65535;
            valor_min = -65535;
            variable_ing = "OBJETIVO X";
            break;
        case "OBJETIVOY":
```

```
            valor_max = 65535;
            valor_min = -65535;
            variable_ing = "OBJETIVO Y";
            break;
    }

    switch (caja_texto)
    {
        case 1:
            if (VARIABLE1.Text != "")
            {
                dato_envio = Convert.ToInt16(VARIABLE1.Text);
                num = int.TryParse(VARIABLE1.Text, out iv);
            }
            else
            {
                MessageBox.Show("INTRODUZCA EL VALOR CORRESPONDIENTE A " + variable_ing);
                vacio = true;
            }
            break;
        case 2:
            if (VARIABLE2.Text != "")
            {
                dato_envio = Convert.ToInt16(VARIABLE2.Text);
                num = int.TryParse(VARIABLE2.Text, out iv);
            }
            else
            {
                MessageBox.Show("INTRODUZCA EL VALOR CORRESPONDIENTE A " + variable_ing);
                vacio = true;
            }
            break;
        case 3:
            if (VARIABLE3.Text != "")
            {
                dato_envio = Convert.ToInt16(VARIABLE3.Text);
                num = int.TryParse(VARIABLE3.Text, out iv);
            }
            else
            {
                MessageBox.Show("INTRODUZCA EL VALOR CORRESPONDIENTE A " + variable_ing);
                vacio = true;
            }
            break;
    }

    if (num == true)
    {


        if (dato_envio < valor_max && dato_envio > valor_min)
        {
            validacion = true;

        }
        else
        {
            MessageBox.Show(variable_ing + " FUERA DE RANGO");
        }
    }
    else
    {

        if (vacio == false)
        {
            MessageBox.Show("VALOR ERRONEO");
        }
    }

    return validacion;

}
private void button1_Click(object sender, EventArgs e)
```

```
    {
        byte* send_buf = stackalloc byte[4];
        int distancia;
        int distancia2;
        bool validar1, validar2, validar3 = false;

        if (contador < 1)
        {
            MessageBox.Show("No hay ningun dispositivo conectado");
        }

        else
        {

            if (REPULSION_B.Checked == true || W_B.Checked == true || OBJETIVOX_B.Checked == true || OBJETIVOY_B.Checked == true || VELOCIDAD_B.Checked == true)
            {
                if (REPULSION_B.Checked == true)
                {
                    validar1 = validar_datos(1, "REPULSION");
                    validar2 = validar_datos(2, "ATRACCION");

                    if (validar1 == true && validar2 == true)
                    {

                        send_buf[0] = 0x08;
                        send_buf[1] = Convert.ToByte(VARIABLE1.Text);
                        send_buf[2] = Convert.ToByte(VARIABLE2.Text); ;
                        send_buf[3] = 0x2A;

                        usbapi.SendPacket(send_buf, 4);
                    }
                }

                if (W_B.Checked == true)
                {
                    validar1 = validar_datos(1, "W");
                    validar2 = validar_datos(2, "Z");
                    validar3 = validar_datos(3, "THAO");

                    if (validar1 == true && validar2 == true && validar3 == true)
                    {
                        send_buf[0] = Convert.ToByte(VARIABLE3.Text);
                        send_buf[1] = Convert.ToByte(VARIABLE1.Text);
                        send_buf[2] = Convert.ToByte(VARIABLE2.Text); ;
                        send_buf[3] = 0x2B;

                        usbapi.SendPacket(send_buf, 4);
                    }
                }


                if (OBJETIVOX_B.Checked == true)
                {
                    validar1 = validar_datos(1, "OBJETIVOX");

                    if (validar1 == true)
                    {
                        distancia = (Int16)Convert.ToInt16(VARIABLE1.Text) / 256;
                        distancia2 = Convert.ToInt16(VARIABLE1.Text) - 256 * distancia;

                        if (Convert.ToInt16(VARIABLE1.Text) < 0)
                        {
                            send_buf[0] = 0x01;
                            distancia2 = distancia2 * (-1);
                        }
                        else
                        {
                            send_buf[0] = 0x00;
                        }


                        send_buf[0] = 0x00;
```

```
                send_buf[1] = Convert.ToByte(distancia2);
                send_buf[2] = Convert.ToByte(distancia);
                send_buf[3] = 0xCD;

                usbapi.SendPacket(send_buf, 4);
            }
        }

        if (OBJETIVOY_B.Checked == true)
        {

            validar1 = validar_datos(1, "OBJETIVOY");
            if (validar1 == true)
            {
                distancia = (Int16)Convert.ToInt16(VARIABLE1.Text) / 256;
                distancia2 = Convert.ToInt16(VARIABLE1.Text) - 256 * distancia;


                if (Convert.ToInt16(VARIABLE1.Text) < 0)
                {
                    send_buf[0] = 0x01;
                    distancia2 = distancia2 * (-1);
                }
                else
                {
                    send_buf[0] = 0x00;
                }

                send_buf[1] = Convert.ToByte(distancia2);
                send_buf[2] = Convert.ToByte(distancia);
                send_buf[3] = 0xDE;

                usbapi.SendPacket(send_buf, 4);
            }
        }

        if (VELOCIDAD_B.Checked == true)
        {
            validar1 = validar_datos(1, "VELOCIDAD");
            if (validar1 == true)
            {
                send_buf[0] = 0x00;
                send_buf[1] = Convert.ToByte(VARIABLE1.Text); ;
                send_buf[2] = 0x00;
                send_buf[3] = 0xEF;

                usbapi.SendPacket(send_buf, 4);
            }
        }
    }
    else
    {
        MessageBox.Show("SELECCIONE LA VARIABLE A PROGRAMAR");
    }

  }
}


private void REPULSION_B_CheckedChanged(object sender, EventArgs e)
{
  VAR1_N.Text = "REPULSION";
  VAR3_N.Text = "";
  VAR2_N.Text = "ATRACCION";

  VARIABLE1.Enabled = true;
  VARIABLE2.Enabled = true;
  VARIABLE3.Enabled = false;

  VARIABLE1.Visible = true;
  VARIABLE2.Visible = true;
  VARIABLE3.Visible = false;
}
```

```
private void W_B_CheckedChanged(object sender, EventArgs e)
{
   VAR1_N.Text = "W";
   VAR2_N.Text = "Z";
   VAR3_N.Text = "THAO";

   VARIABLE1.Enabled = true;
   VARIABLE2.Enabled = true;
   VARIABLE3.Enabled = true;

   VARIABLE1.Visible = true;
   VARIABLE2.Visible = true;
   VARIABLE3.Visible = true;
}

private void VELOCIDAD_B_CheckedChanged(object sender, EventArgs e)
{
   VAR1_N.Text = "VELOCIDAD";
   VAR2_N.Text = "";
   VAR3_N.Text = "";

   VARIABLE1.Enabled = true;
   VARIABLE2.Enabled = false;
   VARIABLE3.Enabled = false;

   VARIABLE1.Visible = true;
   VARIABLE2.Visible = false;
   VARIABLE3.Visible = false;
}

private void OBJETIVOX_B_CheckedChanged(object sender, EventArgs e)
{
   VAR1_N.Text = "OBJETIVO X";
   VAR2_N.Text = "";
   VAR3_N.Text = "";

   VARIABLE1.Enabled = true;
   VARIABLE2.Enabled = false;
   VARIABLE3.Enabled = false;

   VARIABLE1.Visible = true;
   VARIABLE2.Visible = false;
   VARIABLE3.Visible = false;
}

private void OBJETIVOY_B_CheckedChanged(object sender, EventArgs e)
{
   VAR1_N.Text = "OBJETIVO Y";
   VAR2_N.Text = "";
   VAR3_N.Text = "";

   VARIABLE1.Enabled = true;
   VARIABLE2.Enabled = false;
   VARIABLE3.Enabled = false;

   VARIABLE1.Visible = true;
   VARIABLE2.Visible = false;
   VARIABLE3.Visible = false;
}

private void button2_Click(object sender, EventArgs e)
{
   System.IO.StreamWriter writer1;
   SaveFileDialog op = new SaveFileDialog();
   op.Title = "guardar en archivo";
   op.Filter = "Trayectoria (*.txt)|*.txt";
   op.FileName = outputfilename;
   op.ShowDialog();

   timer3.Stop();

   if (op.ShowDialog() == DialogResult.OK)
```

```
        {
            filename = op.FileName;
            writer1 = System.IO.File.CreateText(filename);

            for (int i = 0; i < cont_tray; i++)
            {

                writer1.WriteLine(trayectoria_x[i] + "    " + (707 - trayectoria_y[i]));

            }

            writer1.Close();
        }
    }

    private void timer3_Tick(object sender, EventArgs e)
    {

        if (p_x < 10000 && p_y < 10000 && p_x > -10000 && p_y > -10000 && sentido != 2 && sentido != 0 && sentido
!= 8)
        {
            trayectoria_x[cont_tray] = p_x;
            trayectoria_y[cont_tray] = p_y;

            cont_tray++;
        }
    }

    private void label11_Click(object sender, EventArgs e)
    {

    }

    private void programa_v_Enter(object sender, EventArgs e)
    {

    }


  }

}

//CODIGO DE USBAPI

using System;
using System.Collections.Generic;
using System.Windows.Forms;
using System.Runtime.InteropServices; // Clase para importar DLL

using PVOID = System.IntPtr;
using DWORD = System.UInt32;

namespace usb
{
  unsafe public class PicUSBAPI
  {
    #region Definición de los Strings: EndPoint y VID_PID
    string vid_pid_norm = "vid_04d8&pid_0011";

    string out_pipe = "\\MCHP_EP1";
    string in_pipe = "\\MCHP_EP1";
    #endregion

    #region Funciones importadas de la DLL: mpusbapi.dll
    [DllImport("mpusbapi.dll")]
    private static extern DWORD _MPUSBGetDLLVersion();
    [DllImport("mpusbapi.dll")]
    private static extern DWORD _MPUSBGetDeviceCount(string pVID_PID);
    [DllImport("mpusbapi.dll")]
    private static extern void* _MPUSBOpen(DWORD instance, string pVID_PID, string pEP, DWORD dwDir, DWORD
dwReserved);
    [DllImport("mpusbapi.dll")]
```

```
    private static extern DWORD _MPUSBRead(void* handle, void* pData, DWORD dwLen, DWORD* pLength,
DWORD dwMilliseconds);
    [DllImport("mpusbapi.dll")]
    private static extern DWORD _MPUSBWrite(void* handle, void* pData, DWORD dwLen, DWORD* pLength,
DWORD dwMilliseconds);
    [DllImport("mpusbapi.dll")]
    private static extern DWORD _MPUSBReadInt(void* handle, DWORD* pData, DWORD dwLen, DWORD*
pLength, DWORD dwMilliseconds);
    [DllImport("mpusbapi.dll")]
    private static extern bool _MPUSBClose(void* handle);
    #endregion

    void* myOutPipe;
    void* myInPipe;
    public uint contador2;


    public void OpenPipes()
    {
       DWORD selection = 0;

       myOutPipe = _MPUSBOpen(selection, vid_pid_norm, out_pipe, 0, 0);
       myInPipe = _MPUSBOpen(selection, vid_pid_norm, in_pipe, 1, 0);
    }

    public void ClosePipes()
    {
       _MPUSBClose(myOutPipe);
       _MPUSBClose(myInPipe);
    }

    public void SendPacket(byte* SendData, DWORD SendLength)
    {
       uint SendDelay = 1000;

       DWORD SentDataLength;

       OpenPipes();
       _MPUSBWrite(myOutPipe, (void*)SendData, SendLength, &SentDataLength, SendDelay);
       ClosePipes();
    }

    public void ReceivePacket(byte* ReceiveData, DWORD *ReceiveLength)
                    {
                            uint ReceiveDelay=1000;

                            DWORD ExpectedReceiveLength = *ReceiveLength;

                            OpenPipes();
       _MPUSBRead(myInPipe, (void*)ReceiveData, ExpectedReceiveLength, ReceiveLength, ReceiveDelay);
       ClosePipes();
    }

    public void SumaPIC(uint sumando1, uint sumando2, uint sumando3)
    {
       byte* send_buf = stackalloc byte[4];

       send_buf[0] = 0x08
       send_buf[1] = (byte)sumando1;
       send_buf[2] = (byte)sumando2;
       send_buf[3] = (byte)sumando3;
       SendPacket(send_buf, 4);
    }

    public uint[] ResultadoPIC()
    {

       //uint total = 0;
       byte* receive_buf = stackalloc byte[22];
       uint[] trama = new uint[22];

       DWORD RecvLength = 22;
```

```
            ReceivePacket(receive_buf, &RecvLength);

            /*result = receive_buf[0];
            result1 = receive_buf[1];
            result2 = receive_buf[2];
            result3 = receive_buf[3];

            if (result3 == 0x2C)
            {

                total = ((result * 256) + result1);
            }*/
            trama[0] = receive_buf[0];

            trama[1] = receive_buf[1];
            trama[2] = receive_buf[2];
            trama[3] = receive_buf[3];
            trama[4] = receive_buf[4];
            trama[5] = receive_buf[5];
            trama[6] = receive_buf[6];
            trama[7] = receive_buf[7];
            trama[8] = receive_buf[8];
            trama[9] = receive_buf[9];
            trama[10] = receive_buf[10];
            trama[11] = receive_buf[11];
            trama[12] = receive_buf[12];
            trama[13] = receive_buf[13];
            trama[14] = receive_buf[14];
            trama[15] = receive_buf[15];
            trama[16] = receive_buf[16];
            trama[17] = receive_buf[17];
            trama[18] = receive_buf[18];
            trama[19] = receive_buf[19];
            trama[20] = receive_buf[20];
            trama[21] = receive_buf[21];

            return trama;
        }

         public uint identificacion()
        {
            string identificar;
            uint contador;
            identificar = "vid_04d8&pid_0011";
            contador=_MPUSBGetDeviceCount(identificar);
            return contador;

        }

    }
}
```

ANEXO F Código del simulador empleado para la optimización del sistema.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Drawing.Imaging;
using System.Drawing.Drawing2D;
using System.Drawing.Design;
using System.Collections;
using System.Resources;


namespace simulador
{


   public partial class Form1 : Form
   {

      Bitmap target = new Bitmap(simulador10.Properties.Resources.target);
      Bitmap carro = new Bitmap(simulador10.Properties.Resources.carro);
      public Bitmap imagen2 = new Bitmap(708, 708);
      int i, j, u, v, posicion, constante_repulsion, constante_atraccion, i3, j3, i1, j1, u1, v1,cuenta,  mov_x, mov_y, figura, rotar1, cuenta_x, cuenta_y, n, ico, ico1, generacion, tiempo_maximo, cont6, cont7,cont9,cont8,gen,i_g,generaciones,bandera_rec,cont_tray;
      double angulo, distancia, angulo_sensor, angulo_sensor1, repulsion_x, repulsion_y, distancia_celda, repulsion, atraccion_x, atraccion_y, atraccion, orientacion, orientacion_anterior, distancia_menor, distancia_objetivo, distancia_x, distancia_y, distancia_recorrida, delta_ort,delta_ort1, fitness, num_choques, choques_total, sta, ort_ant, or_a, orientacion1, vl, dc, w, z, thao, sumatoria, sumatoria1, parada, media, ganador, cromosoma_ganador, mayor_g, menor_g, carro_x, carro_y, pos_actual_x, pos_actual_y, pos_ant_x, pos_ant_y,posicion_anterior_x, posicion_anterior_y,rotar_imagen,orientacion_inicial;
      double tiempo_acelerado;
      int[,] matriz = new int[33, 33];
      int[,] matriz_giro = new int[33, 33];
      long[] individuos = new long[44];
      public double[] vector_fitness = new double[44];
      public double[] probabilidad = new double[44];
      public double[] vector_fitness1 = new double[44];
      public double[] vector_padres = new double[44];
      public double[] intervalos = new double[44];
      public double[] fitness_total = new double[88];
      public double[] medias = new double[100];
      public double[] mayores = new double[100];
      public double[] menores = new double[100];
      public float[] trayectoria_x = new float[2000];
      public float[] trayectoria_y = new float[2000];
      public long[] padres = new long[44];
      public long[] hijos = new long[44];
      public long[] maximos = new long[100];
      Point objetivo=new Point();
      Point posicion_carro = new Point();
      DateTime tiempo1 = new DateTime();
      DateTime tiempo4 = new DateTime();
      BitmapData choquesm = new BitmapData();
      public string filename=null;
      public string outputfilename;
      Random r = new Random();
      SaveFileDialog guardar_op = new SaveFileDialog();
      public float imagen_x, imagen_y;


      public Form1()
      {
         InitializeComponent();

      }
```

```
 private void pictureBox1_Click(object sender, EventArgs e)
{


    Point centro = new Point();
    Point a = new Point();
    Point b = new Point();
    Point c = new Point();


    Point punto_mouse = PointToClient(Form1.MousePosition);
    Point pb = pictureBox1.Location;
    Graphics lienzo = Graphics.FromImage(imagen2);



    System.Drawing.Size t = new Size();

    t.Height = Convert.ToInt16(textBox2.Text);
    t.Width = Convert.ToInt16(textBox1.Text);
    Rectangle rectangulo = new Rectangle(centro, t);


    Graphics dibujar = pictureBox1.CreateGraphics();
    SolidBrush cuadrado = new SolidBrush(System.Drawing.Color.Red);
    Pen negro = new Pen(System.Drawing.Color.Black);
    SolidBrush verde = new SolidBrush(System.Drawing.Color.Green);



    angulo = Convert.ToDouble(textBox3.Text);
    a.X = 354;
    a.Y = 354;
    b.X = 708;
    b.Y = 558;
    c.X = 451;
    c.Y = 380;

    rectangulo.X = punto_mouse.X - pb.X;
    rectangulo.Y = punto_mouse.Y - pb.Y;

    if (checkBox1.Checked)
    {
        rectangulo.X = Convert.ToInt32(pos_x.Text);
        rectangulo.Y = 707-Convert.ToInt32(pos_y.Text);
    }


    if (rotar1==1)
    {
        if (figura == 0)
        {
            lienzo.TranslateTransform((float)rectangulo.X, (float)rectangulo.Y);
            lienzo.RotateTransform((float)angulo);
            label3.Text = Convert.ToString(rectangulo.X);

            rectangulo.X = 0;
            rectangulo.Y = 0;
            lienzo.FillRectangle(cuadrado, rectangulo);
            lienzo.ResetTransform();
        }
        else
        {
            lienzo.FillEllipse(cuadrado, rectangulo.X, rectangulo.Y, t.Width, t.Height);
        }
        rotar1 = 0;
    }
    else
    {
        if (figura == 1)
        {
```

```
                lienzo.FillEllipse(cuadrado, rectangulo.X, rectangulo.Y, t.Width, t.Height);

            }
            else
            {
                lienzo.FillRectangle(cuadrado, rectangulo);
            }
        }
        pictureBox1.Image = imagen2;

    }

    private void button1_Click(object sender, EventArgs e)
    {

        inicio();

    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (radioButton1.Checked == true)
        {
            timer1.Interval = timer1.Interval / 400;
            timer2.Interval = timer2.Interval / 10;
            timer3.Interval = timer3.Interval / 400;
            tiempo_acelerado = 1;
        }
        else
        {
            timer1.Interval = 500;
            timer2.Interval = 60000;
            timer3.Interval = 400;
        }
        generacion = 0;
        algoritmo_genetico();


    }
    void inicio()
     {
    if (INICIO_X.TextLength == 0 || INICIO_Y.TextLength == 0 || FINAL_X.TextLength == 0 || FINAL_Y.TextLength ==
0)
    {
        MessageBox.Show("DEBE ESPECIFICAR LOS PUNTOS DE INICIO Y FINAL");
    }
    else
    {
        if (Convert.ToInt32(INICIO_X.Text) < 0 || Convert.ToInt32(INICIO_X.Text) > 708)
        {
            MessageBox.Show("PUNTO DE INICIO INVALIDO");
        }
        else
        {
            if (Convert.ToInt32(INICIO_Y.Text) < 0 || Convert.ToInt32(INICIO_Y.Text) > 708)
            {
                MessageBox.Show("PUNTO DE INICIO INVALIDO");
            }
            else
            {
                if (Convert.ToInt32(FINAL_X.Text) < 0 || Convert.ToInt32(FINAL_X.Text) > 708)
                {
                    MessageBox.Show("PUNTO FINAL INVALIDO");
                }
                else
                {
                    if (Convert.ToInt32(FINAL_Y.Text) < 0 || Convert.ToInt32(FINAL_Y.Text) > 708)
                    {
                        MessageBox.Show("PUNTO FINAL INVALIDO");
                    }
                    else
                    {
                        for (int i = 0; i <= 32; i++)
```

```
{
   for (int j = 0; j <= 32; j++)
   {
      matriz[i, j] = 0;
   }
}



posicion = 0;
posicion_carro.X = Convert.ToInt32(INICIO_X.Text);
posicion_carro.Y = Convert.ToInt32(INICIO_Y.Text);
objetivo.X = Convert.ToInt32(FINAL_X.Text);
objetivo.Y = Convert.ToInt32(FINAL_Y.Text);
posicion_anterior_x = posicion_carro.X;
posicion_anterior_y = posicion_carro.Y;
if (bandera_rec == 1)
{
   generaciones = 0;


}
else
{
   generaciones = Convert.ToInt32(generaciones_t.Text);
}
carro_x = posicion_carro.X;
carro_y = posicion_carro.Y;
imagen_x = posicion_carro.X;
imagen_y = posicion_carro.Y;
orientacion = 0;
orientacion_anterior = 0;
orientacion1 = 0;

cuenta_x = 0;
cuenta_y = 0;
mov_x = 0;
mov_y = 0;
rotar1 = 0;
rotar_imagen = 0;
figura = 0;
choques_total = 0;

num_choques = 0;
n = 0;
delta_ort = 0;
delta_ort1 = 0;
fitness = 0;

atraccion_var.Text = Convert.ToString(constante_atraccion);
repulsion_var.Text = Convert.ToString(constante_repulsion);
omega_var.Text = Convert.ToString(w);
thao_var.Text = Convert.ToString(thao);
z_var.Text = Convert.ToString(z);


w = 0.015625 * w;
z = 0.0078125 * z;
thao = 0.4 + 0.01875 * thao;
constante_atraccion = 1 + constante_atraccion;
constante_repulsion = 1 + constante_repulsion;
sta = 0;
ort_ant = 0;
or_a = 0;
dc = 0;
tiempo_maximo = 0;
cont_tray = 0;


pos_ant_x = (int)posicion_carro.X / 10;
pos_ant_y = (int)posicion_carro.Y / 10;

distancia_x = posicion_carro.X - objetivo.X;
```

```
                    distancia_y = posicion_carro.Y - objetivo.Y;
                    distancia_objetivo = Math.Sqrt((distancia_x * distancia_x) + (distancia_y * distancia_y));
                    distancia_recorrida = 0;
                    orientacion_inicial = 0;
                    orientacion = orientacion_inicial;
                    sta = orientacion_inicial;


                    timer1.Enabled = true;
                    timer2.Enabled = true;
                    timer3.Enabled = true;

                    DateTime tiempo = DateTime.Now;
                    tiempo1 = tiempo;
                    tiempo4 = tiempo;

                }
            }

        }


    }
  }
}

    void sensor(int ang)
    {

        int k, l, contador,bandera,salir;
        double                                                                                                  angulo1,
orientacion1,orientacion2,orientacion3,orientacion4,i_max,i_min,j_max,j_min,limite,limite_y,distancia_min,ang1;
        double y, y1;
        Point a = new Point();
        Point b = new Point();
        Point c = new Point();

        Graphics dibujar = pictureBox1.CreateGraphics();

        if (posicion == 1)
        {


        }

        byte a2;
        byte r2;
        byte g2 = 0;
        byte b2 = 0;
        orientacion1 = ang;

        BitmapData imagen3 = imagen2.LockBits(new Rectangle(0, 0, imagen2.Width, imagen2.Height),
ImageLockMode.ReadWrite, imagen2.PixelFormat);
        IntPtr ptr = imagen3.Scan0;

        Pen negro = new Pen(System.Drawing.Color.Black);
        Pen azul = new Pen(System.Drawing.Color.Aqua);
        SolidBrush verde = new SolidBrush(System.Drawing.Color.Green);
        int bytes = imagen3.Stride * imagen3.Height;
        byte[] rgbvalores = new byte[bytes];
        System.Runtime.InteropServices.Marshal.Copy(ptr, rgbvalores, 0, bytes);
        imagen2.UnlockBits(imagen3);


            u = 1;
            v = 1;
            i = 0;
            j = 0;
            k = 0;
            l = 1;
            g2 = 0;
            b2 = 0;
            contador = 0;
```

```
bandera = 0;
limite = 0;
limite_y = 1;
orientacion4 = 0;
distancia_min = 0;
y1 = 0;
salir = 0;
i3 = 0;
j3 = 0;
if (orientacion1 < 0)
{
   orientacion1 = 360 + orientacion1;
}

orientacion2 = 75 + orientacion1;
orientacion3=105+orientacion1;
if (orientacion2 < 0)
{
   orientacion2 = 360 + orientacion2;
}

if (orientacion3 > 360)
{
   orientacion3 = orientacion3 - 360;
}
if (orientacion2 > 360)
{
   orientacion2 = orientacion2 - 360;
}
if (orientacion3 < 75 && orientacion3 >= 0)
{
   orientacion3 = orientacion3 + 360;
   if (orientacion2 > 0 && orientacion2<75)
   {
      orientacion2 = orientacion2 + 360;
   }
   l = 4;
}
else
{
   if (orientacion3 > 180)
   {
      l = 3;
   }

}



i_min = 165.7 * Math.Cos(orientacion3 * Math.PI / 180);
i_max = 165.7 * Math.Cos(orientacion2 * Math.PI / 180);
j_min = 165.7 * Math.Sin(orientacion3 * Math.PI / 180);
j_max = 165.7 * Math.Sin(orientacion2 * Math.PI / 180);

if (j_max > 0 && j_min > 0 && i_max>0 && i_min>0)
{
   u = (int)i_max;
   v = 0;
   contador = 0;
  limite = 0;
  limite_y = -1;
  orientacion4 = orientacion2;

}
if (j_max > 0 && j_min > 0 && i_max > 0 && i_min<0)
{
   if (Math.Abs(j_max) > Math.Abs(j_min))
   {

      v = (int)(j_max+0.5);
      bandera = 3;
      limite = (int)i_min*2;
      orientacion4 = orientacion2;
```

```
    }
    else
    {
        v = (int)(j_min+0.5);
        limite = (int)i_max*2;
        bandera = 4;
        orientacion4 = orientacion3;
    }
    k = 1;
    contador = 1;

    limite_y = 0;
}
if (j_max > 0 && j_min > 0 && i_max < 0 && i_min < 0)
{
    u = (int)(i_min-0.5);
    v = 0;
    contador = 2;
    limite = 0;
    limite_y = -1;
}
if (j_min < 0 && j_max > 0 && i_max<0 && i_min<0)
{
    if (i_max < i_min)
    {
        u = (int)(i_max-0.5);
        v = (int)(j_max-0.5);
        bandera = 3;
    }
    else
    {
        u = (int)(i_min-0.5);
        v = (int)(j_min-0.5);
        bandera = 4;
    }
    limite = 0;
    limite_y = i_max * 2;
    contador = 3;
    y1 = 2;
}
if (j_max < 0 && j_min < 0 && i_max<0 && i_min<0)
{

    u = (int)(i_max-0.5);
    v = 0;
    limite = 0;
    limite_y = 1;
    contador = 4;
    orientacion4 = orientacion2;

}
if (j_max < 0 && j_min < 0 && i_max < 0 && i_min > 0)
{

    if (Math.Abs(j_max) > Math.Abs(j_min))
    {

        v = (int)(j_max-0.5);
        bandera = 3;
        limite = (int)i_min*2;
        orientacion4 = orientacion2;
    }
    else
    {
        v = (int)(j_min-0.5);
        limite = (int)i_max*2;
        bandera = 4;
        orientacion4 = orientacion3;
    }
    k = 1;
    limite_y =0;
    contador = 5;
}
```

```
if (j_max < 0 && j_min < 0 && i_max > 0 && i_min > 0)
{
   u = (int)(i_min+0.5);
   v = 0;
   contador = 6;
   limite = 0;
}
if (j_max <= 0 && j_min >= 0 && i_max > 0 && i_min > 0)
{

   if (i_max > i_min)
   {
      u = (int)(i_max+0.5);
      v = (int)j_max;
      bandera = 3;
   }
   else
   {
      u = (int)(i_min+0.5);
      v = (int)j_min;
      bandera = 4;
   }

   contador = 7;
   limite = 0;
   limite_y = j_min * 2;
   y1 = 2;
}

if (k == 1)
{
   u = (int)(v / Math.Tan(orientacion4 * Math.PI / 180));

}
distancia_menor = 1000;

while (u !=(int)limite && v!=(int)limite_y && v >-(707-posicion_carro.Y) && v<posicion_carro.Y)
{

   if (k == 1)
   {
      u = (int)((v / Math.Tan(orientacion4 * Math.PI / 180)));
      k = 0;
   }
   i = posicion_carro.X + u;
   j = posicion_carro.Y - v;

   if (y1 == 2)
   {
      y = (v - j_max + (i_max * Math.Tan(orientacion1 * Math.PI / 180))) / Math.Tan(orientacion1 * Math.PI /
180);

      distancia_min = Math.Sqrt((y * y) + (v * v));
   }
   else
   {
      y = (u * Math.Tan(orientacion1 * Math.PI / 180)) + (j_max - (Math.Tan(orientacion1 * Math.PI / 180) *
i_max));

      distancia_min = Math.Sqrt((y * y) + (u * u));
   }


   angulo1 = Math.Atan2(v, u) * 180 / Math.PI;

   if (l == 3 && angulo1<=0)
   {
      angulo1 = angulo1 + 360;
   }
   if (l == 4)
   {
      angulo1 = angulo1 + 360;
   }
   distancia = Math.Sqrt((u * u) + (v * v));
```

```
if (angulo1 <= orientacion3 && angulo1 >= orientacion2 && distancia<=distancia_min)
{
    a2 = rgbvalores[(i * 4) + (j * imagen3.Stride)];
    r2 = rgbvalores[((i * 4) + 1) + (j * imagen3.Stride)];
    g2 = rgbvalores[((i * 4) + 2) + (j * imagen3.Stride)];
    b2 = rgbvalores[((i * 4) + 3) + (j * imagen3.Stride)];

    if (g2 == 255 && b2 == 255)
    {
        salir++;
        if (distancia <= distancia_menor)
        {
            distancia_menor = distancia;
            i3 = i;
            j3 = j;
            u1 = u;
            v1 = v;
        }

    }

    switch (contador)
    {
        case 0:
            v++;
            break;
        case 1:
            if (bandera == 3)
            {
                u--;
            }
            else
            {
                u++;
            }
            break;
        case 2:
            v++;
            break;
        case 3:
            if (bandera == 3)
            {
                v--;
            }
            else
            {
                v++;
            }
            break;
        case 4:
            v--;
            break;
        case 5:
            if (bandera == 3)
            {
                u++;
            }
            else
            {
                u--;
            }
            break;
        case 6:
            v--;
            break;
        case 7:
            if (bandera == 3)
            {
                v++;
            }
            else
            {
                v--;
```

```
            }
            break;

    }


}
else
{

    if (angulo1 > orientacion3)
    {
        switch (contador)
        {
            case 0:
                u--;
                v = 0;
                break;
            case 1:
                if (bandera == 3)
                {
                    v--;
                    u =(int)( v/Math.Tan(orientacion2*Math.PI/180)+0.5);
                }

                else
                {
                    u++;
                }
                break;
            case 2:
                v++;
                break;
            case 3:
                if (bandera == 3)
                {
                    u++;
                    v = (int)j_max;


                }

                else
                {
                    v++;
                }
                break;
            case 4:
                u++;
                v = 0;
                break;
            case 5:
                if (bandera == 3)
                {
                    v++;
                    u = (int)i_max;
                }

                else
                {
                    u--;
                }
                break;
            case 6:
                v--;
                break;
            case 7:
                if (bandera == 3)
                {
                    u--;
                    v = (int)j_max;
                }
```

```
            else
            {
                v--;
            }
            break;

    }
}
else
{
    if (angulo1 < orientacion2)
    {
        switch (contador)
        {
            case 0:
                v++;
                break;
            case 1:
                if (bandera == 3)
                {
                    u--;

                }
                else
                {
                    v--;
                    u =(int)( v/Math.Tan(orientacion3*Math.PI/180)-0.5);

                }

                break;
            case 2:
                u++;
                v = 0;
                break;
            case 3:
                if (bandera == 3)
                {
                    v--;

                }
                else
                {
                    u++;
                    v = (int)j_min;
                }
                break;
            case 4:
                v--;
                break;
            case 5:
                if (bandera == 3)
                {
                    u++;

                }
                else
                {
                    v++;
                    u = (int)i_min;
                }
                break;
            case 6:
                u--;
                v=0;
                break;
            case 7:
                if (bandera == 3)
                {
                    v++;

                }
                else
```

```
            {
                u--;
                v = (int)j_min;
            }
            break;
    }
}
else
{

    switch (contador)
    {
        case 0:
            u--;
            v = 0;
            break;
        case 1:
            if (bandera == 3)
            {
                v--;
                u = (int)(v / Math.Tan(orientacion2 * Math.PI / 180) + 0.5);
            }
            else
            {
                v--;
                u = (int)(v / Math.Tan(orientacion3 * Math.PI / 180) - 0.5);
            }

            break;
        case 2:
            u++;
            v = 0;
            break;
        case 3:
            if (bandera == 3)
            {
                u++;
                v = (int)j_max;
            }
            else
            {
                u++;
                v = (int)j_min;
            }
            break;
        case 4:
            u++;
            v = 0;
            break;
        case 5:
            if (bandera == 3)
            {
                v++;
                u = (int)i_max;
            }
            else
            {
                v++;
                u = (int)i_min;
            }
            break;
        case 6:
            u--;
            v = 0;
            break;
        case 7:
            if (bandera == 3)
            {
                u--;
                v = (int)j_max;
            }
            else
            {
```

```
            u--;
            v = (int)j_min;
        }
        break;

}

if (u != 0)
{
    switch (contador)
    {
        case 0:
            v = 0;
            break;
        case 1:

            break;
        case 2:
            v = 0;
            break;
        case 3:
            if (bandera == 3)
            {
                v = (int)j_max;
            }
            else
            {
                v = (int)j_min;
            }
            break;
        case 4:
            v=0;
            break;
        case 5:

            break;
        case 6:
            v = 0;
            break;
        case 7:
            if (bandera == 3)
            {
                v = (int)j_max;
            }
            else
            {
                v = (int)j_min;
            }

            break;

    }
}
else

{
    switch (contador)
    {
        case 0:
            v=0;
            break;
        case 1:

            break;
        case 2:
            v = 1;
            break;
        case 3:
            if (bandera == 3)
            {
                v = (int)j_max;
            }
            else
```

```
                                {
                                    v = (int)j_min;
                                }
                                break;
                            case 4:
                                v=-1;
                                break;
                            case 5:

                                break;
                            case 6:
                                v = 1;
                                break;
                            case 7:
                                if (bandera == 3)
                                {
                                    v = (int)j_max;
                                }
                                else
                                {
                                    v = (int)j_min;
                                }
                                break;

                        }

                    }
                }
            }

        }



    }

    if (salir == 0)
    {
        distancia_menor = 0;
    }

    if (distancia_menor > 0)
    {
        ang1 = ang - orientacion;

        i1 =(int)(distancia_menor * (Math.Cos((ang+90) * Math.PI / 180)));
        j1 = (int)(distancia_menor * (Math.Sin((ang+90) * Math.PI / 180)));

        i1 =(int)(i1 / 10);
        j1 = (int)(j1 / 10);

        i1 = 16 + i1;
        j1 = 16 - j1;

        matriz[i1, j1] = matriz[i1, j1] + 1;
    }


}
void calculos()
{
    Pen lapiz = new Pen(System.Drawing.Color.Black);
    double                                                                                                                  st,
si,vx,vy,cos,mag_repulsion,avr,mag_ort,sen,kv,atraccion_proy,atrp_x,atrp_y,mag_atraccion,repulsion_ant,delta_wall,ort
_imagen;
    distancia_x = posicion_carro.X-objetivo.X;
    distancia_y = posicion_carro.Y - objetivo.Y;
    st = 0;
    sen = 0;
    si = 0;
    ort_imagen = 0;

    orientacion_anterior = orientacion;
```

```
        repulsion_ant = repulsion;
        vl = 30;
        vx = vl * Math.Cos(orientacion * Math.PI / 180);
        vy = vl * Math.Sin(orientacion * Math.PI / 180);
        or_a = orientacion1;


      repulsion_x = 0;
      repulsion_y = 0;
      cos = 0;



        for (int ix = 0; ix <= 32;ix++)
        {
           for (int jx = 0; jx <= 32;jx++)
           {
              distancia_celda = Math.Sqrt(((16-ix) * (16-ix)) + ((16-jx) * (16-jx)));
              if (distancia_celda != 0)
              {
                 repulsion_x = repulsion_x - ((matriz[ix, jx] * constante_repulsion / (distancia_celda * distancia_celda)) *
((ix-16) / distancia_celda));
                 repulsion_y = repulsion_y - ((matriz[ix, jx] * constante_repulsion / (distancia_celda * distancia_celda)) *
((16-jx) / distancia_celda));

              }
           }

        }
        repulsion = Math.Atan2(repulsion_y, repulsion_x)*180/Math.PI;
        mag_repulsion = Math.Sqrt((repulsion_x * repulsion_x) + (repulsion_y * repulsion_y));


        if (mag_repulsion != 0)
        {
           cos = ((vx * repulsion_x) + (vy * repulsion_y)) / (vl * mag_repulsion);
           avr = Math.Acos(cos) * 180 / Math.PI;
           sen = Math.Sin(avr * Math.PI / 180);
           if (cos < 0)
           {

              kv = (1 - (2 * z)) / (2 * (1 - z));
              kv = 1 / kv;
              vl = ((z * vl) + ((1 - z) * (vl * (sen/kv))));
           }
           else
           {

              vl = (vl / 2) + (0.5 * vl * cos);

           }
           mag_repulsion = (w * mag_repulsion) + ((1 - w) * (mag_repulsion * -cos));
           repulsion_x = mag_repulsion*Math.Cos(repulsion * Math.PI/180);
           repulsion_y = mag_repulsion * Math.Sin(repulsion * Math.PI / 180);
           repulsion = Math.Atan2(repulsion_y, repulsion_x) * 180 / Math.PI;

        }



        avr = Math.Acos(cos)*180/Math.PI;
        distancia_objetivo = Math.Sqrt((distancia_x*distancia_x) + (distancia_y*distancia_y ));
        if (distancia_objetivo == 0) distancia_objetivo = 1;

        atraccion_x = constante_atraccion * ((objetivo.X-posicion_carro.X) / distancia_objetivo);

        atraccion_y = constante_atraccion * ((-(objetivo.Y-posicion_carro.Y)) / distancia_objetivo);

        mag_atraccion=Math.Sqrt((atraccion_x*atraccion_x)+(atraccion_y*atraccion_y));
        atraccion = Math.Atan2(atraccion_y, atraccion_x)*180/Math.PI;
        orientacion1 = Math.Atan2((atraccion_y + repulsion_y), (atraccion_x + repulsion_x))*180/Math.PI;
```

```
            delta_wall = atraccion - orientacion1;

            if (delta_wall >= 360) delta_wall = delta_wall - 360;



            if (Math.Abs(delta_wall) > 90)
            {

                atraccion_proy = repulsion + 145;

                atrp_x = mag_repulsion * Math.Cos(atraccion_proy * Math.PI / 180);
                atrp_y = mag_repulsion * Math.Sin(atraccion_proy * Math.PI / 180);

                orientacion1 = Math.Atan2((atrp_y + repulsion_y), (atrp_x + repulsion_x)) * 180 / Math.PI;

                label6.Text = "wallfollowing";
            }
            else
            {
                label6.Text = "vff";
            }


            si = orientacion1;

            st = ((timer3.Interval * 0.001 * si) + ((thao - (timer3.Interval * 0.001)) * sta)) / thao;


            orientacion = st;
            sta = st;
            mag_ort = Math.Sqrt(((atraccion_y + repulsion_y) * (atraccion_y + repulsion_y)) + ((atraccion_x + repulsion_x) *
(atraccion_x + repulsion_x)));

            if (radioButton1.Checked == true)
            {
                vl = vl * 400 * 0.001;
                vl = vl;

            }
            else
            {
                vl = vl * 400 * 0.001;

            }
            delta_ort = delta_ort + (Math.Abs(orientacion - orientacion_anterior));

            n++;

            rotar_imagen = (int)(st / 4.1634);
            if (rotar_imagen != 0)
            {
                if (rotar_imagen > 0)
                {
                    ort_imagen = orientacion + 4.1634;
                    rotar_imagen--;
                }
                if (rotar_imagen < 0)
                {
                    ort_imagen = orientacion - 4.1634;
                    rotar_imagen++;
                }
            }
            else
            {
                ort_imagen = orientacion;
            }

            Graphics dibujar = pictureBox1.CreateGraphics();
            pictureBox1.Refresh();

            pictureBox1.Refresh();
```

```
        trayectoria_x[cont_tray]=imagen_x;
        trayectoria_y[cont_tray]=imagen_y;

        for (int i = 1; i <= cont_tray; i++)
        {
            dibujar.DrawLine(lapiz, trayectoria_x[i - 1], trayectoria_y[i - 1], trayectoria_x[i], trayectoria_y[i]);
        }

        cont_tray++;


        dibujar.DrawImage(target, objetivo.X-simulador10.Properties.Resources.target.Width/2, objetivo.Y -
simulador10.Properties.Resources.target.Height / 2);
        dibujar.TranslateTransform(imagen_x, imagen_y);

        dibujar.RotateTransform((float)-(orientacion));
        dibujar.DrawImage(carro, -20, -18);
        dibujar.ResetTransform();

        movimiento(vl);

        choques();
        choques_total = choques_total + num_choques;
        num_choques = 0;
        cont8++;
        label5.Text = orientacion.ToString();
        label4.Text = st.ToString();

    }

    void movimiento(double cm)
    {
        double                                                                                    dirx,
diry,tiempo_transcurrido,distancia_recx,distancia_recy,celda_x1,celda_y1,delta_celda_y,delta_celda_x,tiempo_transcur
rido1,distancia_recorrido_sencillo,distancia_objetivo_sencillo;
        int mx,my;
        posicion_anterior_x = carro_x;
        posicion_anterior_y =carro_y;
        dirx = 0;
        diry = 0;
        celda_x1 = 0;
        celda_y1 = 0;


        dirx = (cm * Math.Cos(orientacion * Math.PI / 180));
        diry = (cm * Math.Sin(orientacion * Math.PI / 180));

        carro_x = carro_x + dirx;
        carro_y = carro_y + diry;

        celda_x1 = carro_x/10;
        celda_y1 = carro_y/10;

        if (celda_x1 < 0)
        {
            celda_x1 = celda_x1 - 0.5;
        }
        else
        {
            celda_x1 = celda_x1 + 0.5;
        }

        if (celda_y1 < 0) celda_y1 = celda_y1 - 0.5;
        else celda_y1 = celda_y1 + 0.5;

        celda_x1 = (int)celda_x1;
        celda_y1 = (int)celda_y1;

       delta_celda_x = celda_x1 - pos_ant_x;
       delta_celda_y = celda_y1 - pos_ant_y;

       imagen_x = imagen_x + (float)dirx;
       imagen_y = imagen_y - (float)diry;
```

```
        posicion_carro.X = (int)imagen_x;
        posicion_carro.Y = (int)imagen_y;


         if (delta_celda_x!=0 || delta_celda_y!=0)
         {

            mx = (int)delta_celda_x;
            my = (int)delta_celda_y;
            mover_matriz(mx, my);

         }

         pos_ant_x = celda_x1;
         pos_ant_y = celda_y1;


         distancia_recx = carro_x - posicion_anterior_x;
         distancia_recy = carro_y - posicion_anterior_y;

         distancia_recorrida  =distancia_recorrida+(Math.Sqrt((distancia_recx * distancia_recx) + (distancia_recy * distancia_recy)));
         distancia_recorrido_sencillo = distancia_recorrida;
         distancia_objetivo_sencillo = distancia_objetivo;

         if (distancia_objetivo <= 35 || tiempo_maximo == 1)
           {
            timer1.Enabled = false;
            timer2.Enabled = false;
            timer3.Enabled = false;

            DateTime tiempo3 = DateTime.Now;

            TimeSpan total1 = new TimeSpan(tiempo3.Ticks - tiempo4.Ticks);
            tiempo_transcurrido1 = (total1.Minutes * 60) + total1.Seconds + (total1.Milliseconds * 0.001);
            delta_ort1 = delta_ort / n;

            label5.Text = imagen_y.ToString();
            label8.Text = posicion_carro.X.ToString();
            label7.Text = posicion_carro.Y.ToString();
            label20.Text = choques_total.ToString();
            label29.Text = tiempo_transcurrido1.ToString();
            label30.Text = delta_ort1.ToString();
            label31.Text = distancia_recorrido_sencillo.ToString();
            label32.Text = distancia_objetivo_sencillo.ToString();


            if (bandera_rec == 2)
            {
               DateTime tiempo2 = DateTime.Now;

               TimeSpan total = new TimeSpan(tiempo2.Ticks - tiempo1.Ticks);
               tiempo_transcurrido = (total.Minutes * 60) + total.Seconds + (total.Milliseconds * 0.001);
               delta_ort = delta_ort / n;

               fitness = (10000 / ((100 * choques_total + 10 * tiempo_transcurrido + 10 * delta_ort + distancia_recorrida + distancia_objetivo))) * (10000 / ((100 * choques_total + 10 * tiempo_transcurrido + 10 * delta_ort + distancia_recorrida + distancia_objetivo)));
               label7.Text = fitness.ToString();

               label9.Text = constante_repulsion.ToString() + "\n" + constante_atraccion.ToString() + "\n" + w.ToString() + "\n" + z.ToString() + "\n" + thao.ToString();
               generacion++;
               label19.Text = Convert.ToString(generacion);

               if (ico <= 42)
               {
                  vector_fitness[ico] = fitness;
                  ico++;
                  constante_repulsion = codificacion("repulsion", individuos[ico]);
                  constante_atraccion = codificacion("atraccion", individuos[ico]);
                  w = codificacion("w", individuos[ico]);
```

```
    z = codificacion("z", individuos[ico]);
    thao = codificacion("thao", individuos[ico]);


    cuenta = ico;



    inicio();


}
else
{
    if (ico == 43)
    {
        vector_fitness[ico] = fitness;
        cuenta = ico;

        ico++;

    }
    if (ico1 == 43)
    {
        vector_fitness1[ico1] = fitness;
        cuenta = ico;

        ico1++;

    }


    if (ico == 45 && ico1 == 44)
    {
        elitismo();

        for (int i = 0; i <= 43; i++)
        {
            individuos[i] = hijos[i];
            vector_fitness[i] = vector_fitness1[i];
        }

        for (int i = 0; i <= 43; i++)
        {
            sumatoria = vector_fitness[i] + sumatoria;
        }

        for (int i = 0; i <= 43; i++)
        {
            probabilidad[i] = vector_fitness[i] / sumatoria;
        }
        media = sumatoria / 44;
        medias[gen] = media;
        mayor_menor();

        gen++;
        criterio_parada();
        if (parada == 0)
        {
            label2.Text = media.ToString();
            sumatoria = 0;
            sumatoria1 = 0;
            seleccion();
            cruce();
            mutacion();

            ico1 = 0;

            constante_repulsion = codificacion("repulsion", hijos[ico1]);
            constante_atraccion = codificacion("atraccion", hijos[ico1]);
            w = codificacion("w", hijos[ico1]);
            z = codificacion("z", hijos[ico1]);
            thao = codificacion("thao", hijos[ico1]);
```

```
            inicio();
        }
        else
        {
            ganador = vector_fitness[0];
            i_g = 0;

            for (int i = 1; i <= 43; i++)
            {
                if (vector_fitness[i] > ganador)
                {
                    ganador = vector_fitness[i];
                    i_g = i;
                }

            }
            label8.Text = ganador.ToString();
            cromosoma_ganador = individuos[i_g];

            if (MessageBox.Show("Optimiacion terminada", "Mensaje", MessageBoxButtons.OK) ==
DialogResult.OK)
            {
                System.IO.StreamWriter writer;

                guardar_op.Title = "Select output file:";
                guardar_op.Filter = "Optimizacion (*.txt)|*.txt";
                guardar_op.FileName = outputfilename;


                if (guardar_op.ShowDialog() == DialogResult.OK)
                {
                    filename = guardar_op.FileName;
                    writer = System.IO.File.CreateText(filename);
                    writer.WriteLine("MAYOR POR GENERACION");
                    writer.WriteLine("");

                    for (int i = 0; i <= generaciones - 1; i++)
                    {
                        writer.WriteLine(mayores[i]);
                    }
                    writer.WriteLine("");
                    writer.WriteLine("MENOR POR GENERACION");
                    writer.WriteLine("");

                    for (int i = 0; i <= generaciones - 1; i++)
                    {
                        writer.WriteLine(menores[i]);
                    }
                    writer.WriteLine("");
                    writer.WriteLine("MEDIA POR GENERACION");
                    writer.WriteLine("");

                    for (int i = 0; i <= generaciones - 1; i++)
                    {
                        writer.WriteLine(medias[i]);
                    }

                    writer.WriteLine("");
                    writer.WriteLine("INDIVIDUO MAYOR POR GENERACION");
                    writer.WriteLine("");

                    for (int i = 0; i <= generaciones - 1; i++)
                    {
                        writer.WriteLine(maximos[i]);
                    }

                    writer.Close();

                }
            }
            else
            {
```

```
                    label1.Text = "";

                }
            }
        }
        else
        {

            if (ico == 44)
            {

                for (int i = 0; i <= 43; i++)
                {
                    sumatoria = vector_fitness[i] + sumatoria;
                }

                for (int i = 0; i <= 43; i++)
                {
                    probabilidad[i] = vector_fitness[i] / sumatoria;
                }
                media = sumatoria / 44;
                label2.Text = media.ToString();
                sumatoria = 0;
                sumatoria1 = 0;
                seleccion();
                cruce();
                mutacion();
                ico++;
                ico1 = 0;

                constante_repulsion = codificacion("repulsion", hijos[ico1]);
                constante_atraccion = codificacion("atraccion", hijos[ico1]);
                w = codificacion("w", hijos[ico1]);
                z = codificacion("z", hijos[ico1]);
                thao = codificacion("thao", hijos[ico1]);
                cuenta = ico1;
                inicio();


            }

            else
            {

                if (ico1 <= 42)
                {
                    vector_fitness1[ico1] = fitness;
                    ico1++;
                    constante_repulsion = codificacion("repulsion", hijos[ico1]);
                    constante_atraccion = codificacion("atraccion", hijos[ico1]);
                    w = codificacion("w", hijos[ico1]);
                    z = codificacion("z", hijos[ico1]);
                    thao = codificacion("thao", hijos[ico1]);

                    cuenta = ico1;

                    inicio();
                }

            }

        }
    }
}


private void timer1_Tick(object sender, EventArgs e)
{
```

```
    switch (posicion)
    {
        case 0:
            posicion = 1;
            angulo_sensor = 0+orientacion;
            angulo_sensor1 = 180 + orientacion;

            break;
        case 1:
            posicion = 2;
            angulo_sensor = -30 + orientacion;
            angulo_sensor1 = -210 + orientacion;
            break;

        case 2:
            posicion = 3;
            angulo_sensor = -60+orientacion;
            angulo_sensor1 =-240+orientacion;
            break;
        case 3:
            posicion = 4;
            angulo_sensor = -90+orientacion;
            angulo_sensor1 = 90+orientacion;
            break;
        case 4:
            posicion = 5;
            angulo_sensor = -120+orientacion;
            angulo_sensor1 =-300+orientacion;
            break;
        case 5:
            posicion = 6;
            angulo_sensor = -150+orientacion;
            angulo_sensor1 = -330+orientacion;

            break;
        case 6:
            posicion = 7;
            angulo_sensor = 180+orientacion;
            angulo_sensor1 = 0+orientacion;
            break;
        case 7:
            posicion = 8;
            angulo_sensor = -150 + orientacion;
            angulo_sensor1 = -330 + orientacion;
            break;
        case 8:
            posicion = 9;
            angulo_sensor = -120 + orientacion;
            angulo_sensor1 = -300 + orientacion;
            break;
        case 9:
            posicion = 10;
            angulo_sensor = -90 + orientacion;
            angulo_sensor1 = 90 + orientacion;
            break;
        case 10:
            posicion = 11;
            angulo_sensor = -60 + orientacion;
            angulo_sensor1 = -240 + orientacion;
            break;
        case 11:
            posicion = 0;
            angulo_sensor = -30 + orientacion;
            angulo_sensor1 = -210 + orientacion;
            break;

    }

    sensor((int)angulo_sensor);
    sensor((int)angulo_sensor1);

}
```

```
private void timer2_Tick(object sender, EventArgs e)
{
   tiempo_maximo = 1;
}

void mover_matriz(int movx,int movy)
{


   if (movx >= 0)
   {
      if (movy <= 0)
      {
         for (int r = 0; r <= 32 - movx; r++)
         {
            for (int s = 0; s <= 32+movy; s++)
            {
               matriz[r, s] = matriz[r + movx, s - movy];

            }
         }

         for (int r = 0; r <= 32; r++)
         {
            for (int s =33+movy; s <= 32; s++)
            {
               matriz[r, s] = 0;

            }
         }
         for (int r = 33 - movx; r <= 32; r++)
         {
            for (int s = 0; s <= 32; s++)
            {
               matriz[r, s] = 0;

            }
         }
      }
      else
      {
         for (int r = 0; r <= 32 - movx; r++)
         {
            for (int s = 32; s >= movy; s--)
            {
               matriz[r, s] = matriz[r + movx, s - movy];

            }
         }

         for (int r = 0; r <= 32; r++)
         {
            for (int s =movy-1; s >= 0; s--)
            {
               matriz[r, s] = 0;

            }
         }
         for (int r = 33 - movx; r <= 32; r++)
         {
            for (int s = 0; s <= 32; s++)
            {
               matriz[r, s] = 0;

            }
         }

      }

   }
   else
   {
      if (movy <= 0)
```

```
            {
                for (int r = 32; r >= -movx; r--)
                {
                    for (int s = 0; s <= 32+movy; s++)
                    {
                        matriz[r, s] = matriz[r + movx, s - movy];

                    }
                }

                for (int r = 0; r <= 32; r++)
                {
                    for (int s = 33+movy; s <= 32; s++)
                    {
                        matriz[r, s] = 0;

                    }
                }
                for (int r = 0; r <= -movx - 1; r++)
                {
                    for (int s = 0; s <= 32; s++)
                    {
                        matriz[r, s] = 0;

                    }
                }
            }
            else
            {
                for (int r = 32; r >= - movx; r--)
                {
                    for (int s = 32; s >= movy; s--)
                    {
                        matriz[r, s] = matriz[r + movx, s - movy];

                    }
                }

                for (int r = 0; r <= 32; r++)
                {
                    for (int s =movy-1; s >= 0; s--)
                    {
                        matriz[r, s] = 0;

                    }
                }
                for (int r =0; r <= -movx; r++)
                {
                    for (int s = 0; s <= 32; s++)
                    {
                        matriz[r, s] = 0;

                    }
                }
            }
        }

        if (movy > 0)
        {
            cont6++;
        }
       if(movy<0)
       {
        cont7++;
       }
      }
     void choques()
     {
        int ic, jc;
        double ort, ort1, xc, yc, dis, x4,x5, xc1,xc2, yc1,yc2,yc3,ort2;
        choquesm = imagen2.LockBits(new Rectangle(0, 0, imagen2.Width, imagen2.Height),
ImageLockMode.ReadWrite, imagen2.PixelFormat);
        IntPtr ptr1 = choquesm.Scan0;
```

```
int bytesm = choquesm.Stride * choquesm.Height;
byte[] rgbvaloresm = new byte[bytesm];
System.Runtime.InteropServices.Marshal.Copy(ptr1, rgbvaloresm, 0, bytesm);
imagen2.UnlockBits(choquesm);

Pen azul = new Pen(System.Drawing.Color.Aqua);
Graphics dibujar = pictureBox1.CreateGraphics();


byte a3;
byte r3;
byte g3 = 0;
byte b3 = 0;

ic = 0;
jc = 0;
x4 = 0;
x5 = 0;
dis = Math.Sqrt((18 * 18) + (20 * 20));
ort2 = orientacion;

if (ort2 > 90)
{
   ort2= -(360 - (ort2 + 180));
}
else
{
   if (ort2 < -90)
   {
      ort2 = (360 + (ort2 - 180));
   }
}

if (ort2 <= 0)
{
   ort = Math.Atan2(18, -20) * 180 / Math.PI;
   ort1 = Math.Atan2(18, 20) * 180 / Math.PI;
   xc = dis * (Math.Cos((ort + ort2) * Math.PI / 180));
   yc = dis * (Math.Sin((ort + ort2) * Math.PI / 180));
   xc1 = dis * (Math.Cos((ort1 + ort2) * Math.PI / 180));
   yc1 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   ort1 = Math.Atan2(-18, -20) * 180 / Math.PI;
   yc2 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   xc2 = dis * (Math.Cos((ort1 + ort2) * Math.PI / 180));
   ort1 = Math.Atan2(-18, 20) * 180 / Math.PI;
   yc3 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   ort = 90 + ort2;
   ort1 = ort2;
}
else
{
   ort = Math.Atan2(18, 20) * 180 / Math.PI;
   ort1 = Math.Atan2(-18, 20) * 180 / Math.PI;
   xc = dis * (Math.Cos((ort + ort2) * Math.PI / 180));
   yc = dis * (Math.Sin((ort + ort2) * Math.PI / 180));
   xc1 = dis * (Math.Cos((ort1 + ort2) * Math.PI / 180));
   yc1 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   ort1 = Math.Atan2(18, -20) * 180 / Math.PI;
   yc2 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   xc2 = dis * (Math.Cos((ort1 + ort2) * Math.PI / 180));
   ort1 = Math.Atan2(-18, -20) * 180 / Math.PI;
   yc3 = dis * (Math.Sin((ort1 + ort2) * Math.PI / 180));
   ort = 90 + ort2;
   ort1 = ort2;
}


if (ort2 != 0  && ort2!=90 && ort2!=180 && ort2!=-90)
{

   if (ort2 < 0)
   {
      if ((int)yc2 >= (int)yc1)
```

```
{
   for (int yx = (int)yc; yx >= (int)yc2; yx--)
   {
      x4 = ((yx - yc + (Math.Tan((90 - (-ort2)) * Math.PI / 180) * xc)) / Math.Tan((90 - (-ort2)) * Math.PI /
180));
      x5 = ((yx - yc + (Math.Tan(ort2 * Math.PI / 180) * xc)) / Math.Tan(ort2 * Math.PI / 180));

      if (x4 <= xc2)
      {
         x4 = xc2;
      }

      for (int ix = (int)x4; ix <= (int)x5; ix++)
      {
         ic = posicion_carro.X + ix;
         jc = posicion_carro.Y + yx;

         if (ic <= 707 && jc <= 707)
         {
            a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
            r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
            g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
            b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
         }
         if (g3 == 255 && b3 == 255)
         {
            num_choques++;

         }
      }
     }

   for (int yx = (int)yc2; yx >= (int)yc1; yx--)
   {

      x4 = ((yx - yc2 + (Math.Tan(ort2 * Math.PI / 180) * xc2)) / Math.Tan(ort2 * Math.PI / 180));
      x5 = ((yx - yc1 + (Math.Tan((ort2) * Math.PI / 180) * xc1)) / Math.Tan((ort2) * Math.PI / 180));

      for (int ix = (int)x4; ix <= (int)x5; ix++)
      {
         ic = posicion_carro.X + ix;
         jc = posicion_carro.Y + yx;

         if (ic <= 707 && jc <= 707)
         {
            a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
            r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
            g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
            b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
         }
         if (g3 == 255 && b3 == 255)
         {
            num_choques++;

         }
      }

   }

   for (int yx = (int)yc1; yx >= (int)yc3; yx--)
   {

      x4 = ((yx - yc2 + (Math.Tan(((ort2)) * Math.PI / 180) * xc2)) / Math.Tan(((ort2)) * Math.PI / 180));
      x5 = ((yx - yc1 + (Math.Tan((90 + ort2) * Math.PI / 180) * xc1)) / Math.Tan((90 + ort2) * Math.PI / 180));

      if (x5 > xc1)
      {
         x5 = xc1;
      }

      for (int ix = (int)x4; ix <= (int)x5; ix++)
      {
         ic = posicion_carro.X + ix;
```

```
                jc = posicion_carro.Y + yx;


                if (ic <= 707 && jc <= 707)
                {
                    a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                    r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                    g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                    b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
                }
                if (g3 == 255 && b3 == 255)
                {
                    num_choques++;

                }
            }

        }



    }
    else
    {
        for (int yx = (int)yc; yx >= (int)yc1; yx--)
        {

            x4 = ((yx - yc + (Math.Tan((90 - (-ort2)) * Math.PI / 180) * xc)) / Math.Tan((90 - (-ort2)) * Math.PI /
180));
            x5 = ((yx - yc + (Math.Tan(ort2 * Math.PI / 180) * xc)) / Math.Tan(ort2 * Math.PI / 180));

            if (x5 > xc1)
            {
                x5 = xc1;
            }
            for (int ix = (int)x4; ix <= (int)x5; ix++)
            {
                ic = posicion_carro.X + ix;
                jc = posicion_carro.Y + yx;

                if (jc <= 707 && ic <= 707)
                {
                    a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                    r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                    g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                    b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
                }
                if (g3 == 255 && b3 == 255)
                {
                    num_choques++;

                }
            }

        }
        for (int yx = (int)yc1; yx >= (int)yc2; yx--)
        {

            x4 = ((yx - yc + (Math.Tan((90 - (-ort2)) * Math.PI / 180) * xc)) / Math.Tan((90 - (-ort2)) * Math.PI /
180));
            x5 = ((yx - yc1 + (Math.Tan((90 + ort2) * Math.PI / 180) * xc1)) / Math.Tan((90 + ort2) * Math.PI / 180));


            for (int ix = (int)x4; ix <= (int)x5; ix++)
            {
                ic = posicion_carro.X + ix;
                jc = posicion_carro.Y + yx;

                if (ic <= 707 && jc <= 707)
                {
                    a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                    r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                    g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
```

```
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
                num_choques++;

            }
        }

    }
    for (int yx = (int)yc2; yx >= (int)yc3; yx--)
    {

        x4 = ((yx - yc2 + (Math.Tan(((ort2)) * Math.PI / 180) * xc2)) / Math.Tan((ort2) * Math.PI / 180));
        x5 = ((yx - yc1 + (Math.Tan((90 + ort2) * Math.PI / 180) * xc1)) / Math.Tan((90 + ort2) * Math.PI / 180));

        if (x4 < xc2)
        {
            x4 = xc2;
        }


        for (int ix = (int)x4; ix <= (int)x5; ix++)
        {
            ic = posicion_carro.X + ix;
            jc = posicion_carro.Y + yx;

            if (ic <= 707 && jc <= 707)
            {
                a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
                num_choques++;

            }
        }

    }
  }
}
else
{
  if ((int)yc2 >= (int)yc1)
  {
    for (int yx = (int)yc; yx >= (int)yc2; yx--)
    {

        x4 = ((yx - yc + (Math.Tan(ort2 * Math.PI / 180) * xc)) / Math.Tan(ort2 * Math.PI / 180));
        x5 = ((yx - yc + (Math.Tan(ort * Math.PI / 180) * xc)) / Math.Tan(ort * Math.PI / 180));

        if (x4 <= xc2)
        {
            x4 = xc2;
        }

        for (int ix = (int)x4; ix <= (int)x5; ix++)
        {
            ic = posicion_carro.X + ix;
            jc = posicion_carro.Y + yx;

            if (ic <= 707 && jc <= 707)
            {
                a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
```

```
                num_choques++;

            }
        }

    }

    for (int yx = (int)yc2; yx >= (int)yc1; yx--)
    {

        x4 = ((yx - yc2 + (Math.Tan(ort * Math.PI / 180) * xc2)) / Math.Tan(ort * Math.PI / 180));
        x5 = ((yx - yc1 + (Math.Tan(ort * Math.PI / 180) * xc1)) / Math.Tan(ort * Math.PI / 180));


        for (int ix = (int)x4; ix <= (int)x5; ix++)
        {
            ic = posicion_carro.X + ix;
            jc = posicion_carro.Y + yx;

            if (ic <= 707 && jc <= 707)
            {
                a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
                num_choques++;

            }
        }

    }

    for (int yx = (int)yc1; yx >= (int)yc3; yx--)
    {

        x4 = ((yx - yc2 + (Math.Tan(ort * Math.PI / 180) * xc2)) / Math.Tan(ort * Math.PI / 180));
        x5 = ((yx - yc1 + (Math.Tan(ort1 * Math.PI / 180) * xc1)) / Math.Tan(ort1 * Math.PI / 180));

        if (x5 > xc1)
        {
            x5 = xc1;
        }

        for (int ix = (int)x4; ix <= (int)x5; ix++)
        {
            ic = posicion_carro.X + ix;
            jc = posicion_carro.Y + yx;

            if (ic <= 707 && jc <= 707)
            {
                a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
                num_choques++;

            }
        }

    }

}
else
{
    for (int yx = (int)yc; yx >= (int)yc1; yx--)
    {
```

```
    x4 = ((yx - yc + (Math.Tan(ort2 * Math.PI / 180) * xc)) / Math.Tan(ort2 * Math.PI / 180));
    x5 = ((yx - yc + (Math.Tan((90+ort2) * Math.PI / 180) * xc)) / Math.Tan((90+ort2) * Math.PI / 180));

    if (x5 > xc1)
    {
        x5 = xc1;
    }

    for (int ix = (int)x4; ix <= (int)x5; ix++)
    {
        ic = posicion_carro.X + ix;
        jc = posicion_carro.Y + yx;

        if (ic <= 707 && jc <= 707)
        {
            a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
            r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
            g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
            b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
        }
        if (g3 == 255 && b3 == 255)
        {
            num_choques++;
        }

    }

}
for (int yx = (int)yc1; yx >= (int)yc2; yx--)
{

    x4 = ((yx - yc + (Math.Tan((ort2) * Math.PI / 180) * xc)) / Math.Tan((ort2) * Math.PI / 180));
    x5 = ((yx - yc1 + (Math.Tan((ort2) * Math.PI / 180) * xc1)) / Math.Tan((ort2) * Math.PI / 180));



    for (int ix = (int)x4; ix <= (int)x5; ix++)
    {
        ic = posicion_carro.X + ix;
        jc = posicion_carro.Y + yx;

        if (ic <= 707 && jc <= 707)
        {
            a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
            r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
            g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
            b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
        }
        if (g3 == 255 && b3 == 255)
        {
            num_choques++;

        }
    }

}
for (int yx = (int)yc2; yx >= (int)yc3; yx--)
{

    x4 = ((yx - yc2 + (Math.Tan(((90+ort2)) * Math.PI / 180) * xc2)) / Math.Tan((90+ort2) * Math.PI / 180));
    x5 = ((yx - yc1 + (Math.Tan((ort2) * Math.PI / 180) * xc1)) / Math.Tan((ort2) * Math.PI / 180));

    if (x4 < xc2)
    {
        x4 = xc2;
    }

    for (int ix = (int)x4; ix <= (int)x5; ix++)
    {
        ic = posicion_carro.X + ix;
        jc = posicion_carro.Y + yx;

        if (ic <= 707 && jc <= 707)
```

```
                {
                    a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                    r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                    g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                    b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
                }
                if (g3 == 255 && b3 == 255)
                {
                    num_choques++;

                }
            }

        }
    }
}
else
{
    if(ort2==0 || ort2==180)
    {
        x4 = -20;
        x5 = 20;
        yc = 16;
        yc1 = -16;

    }

    else
    {
        if(ort2==90 || ort2==-90)
        {
            x4=-16;
            x5 = 16;
            yc = 20;
            yc1 = -20;
        }
    }
    for (int yx = (int)yc; yx >= (int)yc1; yx--)
    {


        for (int ix = (int)x4; ix <= (int)x5; ix++)
        {
            ic = posicion_carro.X + ix;
            jc = posicion_carro.Y + yx;

            if (ic <= 707 && jc <= 707)
            {
                a3 = rgbvaloresm[(ic * 4) + (jc * choquesm.Stride)];
                r3 = rgbvaloresm[((ic * 4) + 1) + (jc * choquesm.Stride)];
                g3 = rgbvaloresm[((ic * 4) + 2) + (jc * choquesm.Stride)];
                b3 = rgbvaloresm[((ic * 4) + 3) + (jc * choquesm.Stride)];
            }
            if (g3 == 255 && b3 == 255)
            {
                num_choques++;

            }
        }

    }
}
if (num_choques > 0) { num_choques = 1; }
else { num_choques = 0; }


cont9++;

}


private void timer3_Tick(object sender, EventArgs e)
```

```
{
    calculos();
}

private void button3_Click(object sender, EventArgs e)
{
    button3.Enabled = false;
    button4.Enabled = true;

    button3.Image = simulador10.Properties.Resources.bot_rec_peq;
    button4.Image = simulador10.Properties.Resources.circulo_gnd;
    figura = 0;
}

private void button4_Click(object sender, EventArgs e)
{
    button3.Enabled = true;
    button4.Enabled = false;
    button4.Image = simulador10.Properties.Resources.circulo_pqn;
    button3.Image = simulador10.Properties.Resources.bot_rec;
    figura = 1;
}


private void rotar_Click(object sender, EventArgs e)
{
    rotar1 = 1;

}

private void button5_Click(object sender, EventArgs e)
{
    Image entorno = pictureBox1.Image;
    pictureBox1.Image = entorno;


    ImageFormat bmp =System.Drawing.Imaging.ImageFormat.Bmp;
    SaveFileDialog sv = new SaveFileDialog();
    sv.Title = "GUARDAR ENTORNO COMO";
    sv.Filter = "Mapa de Bits|*.bmp";

    sv.ShowDialog();
    if (sv.FileName!="")
    {
        entorno.Save(sv.FileName);
    }

}

private void button6_Click(object sender, EventArgs e)
{

    OpenFileDialog ld = new OpenFileDialog();
    ld.ShowDialog();

    if (ld.FileName != "")
    {
        filename = ld.FileName;
        pictureBox1.Image = abrir(filename);
        imagen2 = new Bitmap(pictureBox1.Image);
    }



}

public static Image abrir(string filename)
{
    Image result;

    #region Save file to byte array

    long size = (new System.IO.FileInfo(filename)).Length;
```

```
        System.IO.FileStream fs = new System.IO.FileStream(filename, System.IO.FileMode.Open,
System.IO.FileAccess.Read);
        byte[] data = new byte[size];
        try
        {
            fs.Read(data, 0, (int)size);
        }
        finally
        {
            fs.Close();
            fs.Dispose();
        }

        #endregion

        #region Convert bytes to image

        System.IO.MemoryStream ms = new System.IO.MemoryStream();
        ms.Write(data, 0, (int)size);
        result = new Bitmap(ms);
        ms.Close();

        #endregion
        return result;

    }
    public static void guardar(Image img, string fn, ImageFormat format)
    {

        if (System.IO.File.Exists(fn))
            System.IO.File.Delete(fn);

        try
        {

            #region Convert image to destination format

            System.IO.MemoryStream ms = new System.IO.MemoryStream();
            Image img2 = (Image)img.Clone();
            img2.Save(ms, format);
            img2.Dispose();
            byte[] byteArray = ms.ToArray();
            ms.Close();
            ms.Dispose();

            #endregion

            #region Save byte array to file

            System.IO.FileStream fs = new System.IO.FileStream(fn, System.IO.FileMode.CreateNew,
System.IO.FileAccess.Write);
            try
            {
                fs.Write(byteArray, 0, byteArray.Length);
            }
            finally
            {
                fs.Close();
                fs.Dispose();
            }

            #endregion

        }
        catch
        {

            if (System.IO.File.Exists(fn))
                System.IO.File.Delete(fn);


            throw;
        }
```

```
}
int codificacion(string variable,long entrada)
{
   int o,cont1,n,q,n1,numero;
   long rep,r;

   BitArray cromosoma = new BitArray(33,false);
   n = 0;
   q = 0;
   n1 = 0;

   switch (variable)
   {
      case "repulsion":
         n1 = 25;
         q = 8;
            break;
      case "atraccion":
            n1 = 17;
            q = 8;
            break;
      case "w":
            n1 = 11;
            q = 6;
            break;
      case "z":
            n1 = 5;
            q = 6;
            break;
      case "thao":
            n1 = 0;
            q = 5;
            break;

   }

   rep = entrada;


   cont1 = 0;
   numero = 0;

   for (long i =(long)Math.Pow((double)2,(double)32); i >= 1; i = i / 2)
   {
      cont1++;

      r = (long)(rep % i);
      if (r >= 0 && rep/i>0)
      {
         rep = (long)r;
         cromosoma[33 - cont1] = true;
         if (r == 0) i = 0;
      }
      else
      {
         cromosoma[33 - cont1] = false;
      }

   }
   cont1 = 0;
   for (int i = n1; i <= n1+(q-1); i++)
   {
      if (cromosoma[i] == false) n =0;
      else n = 1;

      numero =(int)(numero+n * Math.Pow(2, cont1));
      cont1++;
   }


  for (int i =32; i >= 0; i--)
  {
     if (cromosoma[i] == false)
```

```
            {
                o = 0;
            }
            else
            {
                o = 1;
            }

        }
        return (numero);

    }
    void algoritmo_genetico()
    {


        bandera_rec = 2;


        for (int i = 0; i <= 43; i++)
        {

            individuos[i] = random((long)Math.Pow(2, 33), 0);
        }


            constante_repulsion = codificacion("repulsion", individuos[0]);
            constante_atraccion = codificacion("atraccion", individuos[0]);
            w = codificacion("w", individuos[0]);
            z = codificacion("z", individuos[0]);
            thao = codificacion("thao", individuos[0]);
            label9.Text = constante_repulsion.ToString() + "\n" + constante_atraccion.ToString() + "\n" + w.ToString() +
"\n" + z.ToString() + "\n" + thao.ToString();
            ico = 0;
            inicio();


    }

    long random(long max, long min)
    {
        long  resultado;


        double rn = (max * 1.0 - min * 1.0) * r.NextDouble() + min * 1.0;

        resultado = Convert.ToInt64(rn);

        return (resultado);
    }

    private void button7_Click(object sender, EventArgs e)
    {

        timer1.Enabled = false;
        timer3.Enabled = false;


        System.IO.StreamWriter writer1;
        SaveFileDialog op = new SaveFileDialog();
        op.Title = "guardar en archivo";
        op.Filter = "Optimizacion (*.txt)|*.txt";
        op.FileName = outputfilename;
        op.ShowDialog();


        if (op.ShowDialog()==DialogResult.OK)
        {
            filename = op.FileName;
            writer1 = System.IO.File.CreateText(filename);

            for (int i = 0; i <= 44; i++)
            {
```

```
            writer1.WriteLine(i + " " + individuos[i]);

        }
        writer1.WriteLine(cuenta);
        writer1.Close();
    }

}
void seleccion()
{

    int marca1 = 0, marca2 = 0;
    double[] decision= new double[2];
    Random r2 = new Random();

    intervalos[0] = probabilidad[0];
    for (int i = 1; i <= 43; i++)
    {
        intervalos[i] = intervalos[i - 1] + probabilidad[i];

    }

    for (int j = 0; j <= 21; j++)
    {
        marca1 = 0;
        marca2 = 0;
        decision[0] = r2.NextDouble();
        decision[1] = r2.NextDouble();

        for (int rz = 0; rz <= 43; rz++)
        {

            if (intervalos[rz] - decision[0] <= 0)
            {
                marca1++;
            }
        }

        for (int rx = 0; rx <= 43; rx++)
        {

            if (intervalos[rx] - decision[1] <= 0)
            {
                marca2++;
            }
        }

            padres[2 * j] = individuos[marca1];
            padres[(2 * j) + 1] = individuos[marca2];
            vector_padres[2 * j] = vector_fitness[marca1];
            vector_padres[(2 * j) + 1] = vector_fitness[marca2];


     }



}
void cruce()
{
    double pc = 0.5;
    double rpc = 0;
    int cr = 0, cont1 = 0,nc=0;
    long rep = 0, numero = 0, numero1 = 0,rc=0;
    BitArray cromosoma = new BitArray(33, false);
    BitArray cromosoma1 = new BitArray(33, false);
    BitArray hijo = new BitArray(33, false);
    BitArray hijo1 = new BitArray(33, false);

    for (int j = 0; j <= 21; j++)
    {
        rpc = r.NextDouble();
```

```
if (rpc < pc)
{
    cr = r.Next(32);
    cont1 = 0;
    rep = padres[2 * j];
    for (long i = (long)Math.Pow((double)2, (double)32); i >= 1; i = i / 2)
    {
        cont1++;

        rc = (long)(rep % i);
        if (rc >= 0 && rep / i > 0)
        {
            rep = (long)rc;
            cromosoma[33 - cont1] = true;
            if (rc == 0) i = 0;
        }
        else
        {
            cromosoma[33 - cont1] = false;
        }

    }

    cont1 = 0;
    rep = padres[(2 * j) + 1];
    for (long i = (long)Math.Pow((double)2, (double)32); i >= 1; i = i / 2)
    {
        cont1++;

        rc = (long)(rep % i);
        if (rc >= 0 && rep / i > 0)
        {
            rep = (long)rc;
            cromosoma1[33 - cont1] = true;
            if (rc == 0) i = 0;
        }
        else
        {
            cromosoma1[33 - cont1] = false;
        }

    }
    for (int i = 32; i >= cr; i--)
    {
        hijo[i] = cromosoma[i];
    }
    for (int i = cr - 1; i >= 0; i--)
    {
        hijo[i] = cromosoma1[i];
    }
    for (int i = 32; i >= cr; i--)
    {
        hijo1[i] = cromosoma1[i];
    }
    for (int i = cr - 1; i >= 0; i--)
    {
        hijo1[i] = cromosoma[i];
    }
    numero = 0;
    cont1=0;
    for (int icr = 0; icr <= 32; icr++)
    {
        if (hijo[icr] == false) nc = 0;
        else nc = 1;

        numero = (long)(numero + nc * Math.Pow(2, icr));
        cont1=icr;

    }
    numero1 = 0;
    for (int icr = 0; icr <= 32; icr++)
    {
        if (hijo1[icr] == false) nc = 0;
```

```
            else nc = 1;

            numero1 = (long)(numero1 + nc * Math.Pow(2, icr));

          }

          hijos[2 * j] = numero;
          hijos[(2 * j) + 1] = numero1;

        }
        else
        {
          hijos[2 * j] = padres[2*j];
          hijos[(2 * j) + 1] = padres[(2*j)+1];

        }


    }
}
 void mutacion()
 {
    double pm = 0.1;
    double rpm = 0;
    long rc = 0,rep=0,numero1=0;
    Random bits = new Random();
    int bit = 0,cont1=0,nc;

    BitArray cromosoma_hijo = new BitArray(33, false);

    for (int j = 0; j <= 43; j++)
    {
       rpm = r.NextDouble();
       if (rpm < pm)
       {
          bit = bits.Next(33);
          cont1 = 0;
          rep = hijos[j];
          for (long i = (long)Math.Pow((double)2, (double)32); i >= 1; i = i / 2)
          {
             cont1++;

             rc = (long)(rep % i);
             if (rc >= 0 && rep / i > 0)
             {
                rep = (long)rc;
                cromosoma_hijo[33 - cont1] = true;
                if (rc == 0) i = 0;
             }
             else
             {
                cromosoma_hijo[33 - cont1] = false;
             }

          }
          if (cromosoma_hijo[bit] == false)
          {
             cromosoma_hijo[bit] = true;
          }
          else
          {
             cromosoma_hijo[bit] = false;
          }
          numero1 = 0;
          for (int icr = 0; icr <= 32; icr++)
          {
             if (cromosoma_hijo[icr] == false) nc = 0;
             else nc = 1;

             numero1 = (long)(numero1 + nc * Math.Pow(2, icr));

          }
          hijos[j] = numero1;
```

```
        }


    }

}
void elitismo()
{
    int max=0,min=0;

    for (int i = 0; i <= 43; i++)
    {
        fitness_total[i] = vector_fitness[i];
    }
    for (int i = 44; i <= 87; i++)
    {
        fitness_total[i] = vector_fitness1[i - 44];
    }

    for (int i = 0; i <= 87; i++)
    {
        if (fitness_total[max] < fitness_total[i])
        {
            max = i;
        }
    }

    for (int i = 0; i <= 43; i++)
    {
        if (vector_fitness1[min] > vector_fitness1[i])
        {
            min = i;
        }
    }

    if (max <= 43)
    {
        hijos[min] = individuos[max];
        maximos[gen] = individuos[max];
        maximos[gen] = individuos[max];
    }
    else
    {
        hijos[min] = hijos[max - 44];
        maximos[gen] = individuos[max-44];
        maximos[gen] = individuos[max - 44];
    }

    vector_fitness1[min] = fitness_total[max];



}
void criterio_parada()
{
    if (gen >=generaciones)
    {
        parada = 1;
    }
}
void mayor_menor()
{
    mayor_g = vector_fitness[0];
    for (int i = 1; i <= 43; i++)
    {
        if (vector_fitness[i] > mayor_g)
        {
            mayor_g = vector_fitness[i];
        }

    }
    menor_g = vector_fitness[0];
    for (int i = 1; i <= 43; i++)
```

```
    {
        if (vector_fitness[i] < menor_g)
        {
            menor_g = vector_fitness[i];
        }

    }
    mayores[gen] = mayor_g;
    menores[gen] = menor_g;


}

private void Form1_Activated(object sender, EventArgs e)
{
    this.Refresh();


}

private void RECORRIDO_SENCILLO_Click(object sender, EventArgs e)
{


    bandera_rec = 1;
    constante_atraccion = Convert.ToInt16(atraccion_var.Text);
    constante_repulsion = Convert.ToInt16(repulsion_var.Text);
    thao = Convert.ToInt16(thao_var.Text);
    w = Convert.ToInt16(omega_var.Text);
    z = Convert.ToInt16(z_var.Text);

    inicio();

    timer2.Enabled = false;


}

private void REFRESH_Click(object sender, EventArgs e)
{
    Graphics lienzo = Graphics.FromImage(imagen2);
    lienzo.Clear(Color.Empty);
    pictureBox1.Refresh();
}


private void button8_Click(object sender, EventArgs e)
{
    System.IO.StreamWriter writer1;
    SaveFileDialog op = new SaveFileDialog();
    op.Title = "guardar en archivo";
    op.Filter = "Trayectoria (*.txt)|*.txt";
    op.FileName = outputfilename;
    op.ShowDialog();


    if (op.ShowDialog() == DialogResult.OK)
    {
        filename = op.FileName;
        writer1 = System.IO.File.CreateText(filename);

        for (int i = 0; i < cont_tray; i++)
        {

            writer1.WriteLine(trayectoria_x[i]+"    "+(707-trayectoria_y[i]));

        }

        writer1.Close();
    }
}

private void FINAL_X_TextChanged(object sender, EventArgs e)
```

```
        {
            objetivo.X = Convert.ToInt32(FINAL_X.Text);
            objetivo.Y = Convert.ToInt32(FINAL_Y.Text);
            posicion_carro.X = Convert.ToInt32(INICIO_X.Text);
            posicion_carro.Y = Convert.ToInt32(INICIO_Y.Text);

            pictureBox1.Refresh();
            Graphics dibujar = pictureBox1.CreateGraphics();
            dibujar.DrawImage(target, objetivo.X, objetivo.Y);
            dibujar.DrawImage(carro, posicion_carro.X, posicion_carro.Y);
        }

        private void FINAL_Y_TextChanged(object sender, EventArgs e)
        {
            objetivo.X = Convert.ToInt32(FINAL_X.Text);
            objetivo.Y = Convert.ToInt32(FINAL_Y.Text);
            posicion_carro.X = Convert.ToInt32(INICIO_X.Text);
            posicion_carro.Y = Convert.ToInt32(INICIO_Y.Text);

            pictureBox1.Refresh();
            Graphics dibujar = pictureBox1.CreateGraphics();
            dibujar.DrawImage(target, objetivo.X, objetivo.Y);
            dibujar.DrawImage(carro, posicion_carro.X, posicion_carro.Y);
        }

        private void INICIO_X_TextChanged(object sender, EventArgs e)
        {
            objetivo.X = Convert.ToInt32(FINAL_X.Text);
            objetivo.Y = Convert.ToInt32(FINAL_Y.Text);
            posicion_carro.X = Convert.ToInt32(INICIO_X.Text);
            posicion_carro.Y = Convert.ToInt32(INICIO_Y.Text);

            pictureBox1.Refresh();
            Graphics dibujar = pictureBox1.CreateGraphics();
            dibujar.DrawImage(target, objetivo.X, objetivo.Y);
            dibujar.DrawImage(carro, posicion_carro.X, posicion_carro.Y);
        }

        private void INICIO_Y_TextChanged(object sender, EventArgs e)
        {
            objetivo.X = Convert.ToInt32(FINAL_X.Text);
            objetivo.Y = Convert.ToInt32(FINAL_Y.Text);
            posicion_carro.X = Convert.ToInt32(INICIO_X.Text);
            posicion_carro.Y = Convert.ToInt32(INICIO_Y.Text);

            pictureBox1.Refresh();
            Graphics dibujar = pictureBox1.CreateGraphics();
            dibujar.DrawImage(target, objetivo.X, objetivo.Y);
            dibujar.DrawImage(carro, posicion_carro.X, posicion_carro.Y);
        }

    }

}
```