Multibot

Multibot

Robótica aplicada al rescate urbano

Miguel Ricardo Pérez Pereira
Giovanni Rodrigo Bermúdez Bohórquez
José Danilo Rairán Antolines

ESPACIOS

Agradecimientos

Una vez terminada la laboriosa tarea que es la investigación en Colombia y después de materializar los resultados en un libro, se nos vienen a la cabeza personas que sin su trabajo y dedicación no hubiera sido posible la materialización de esta obra. Es por esto que emprendemos con mucha seriedad la importante tarea de escribir estos agradecimientos, pues olvidar a alguien sería imperdonable.

Esta investigación es producto del trabajo de tres grupos de investigación de la Universidad Distrital Francisco José de Caldas: Robótica Móvil Autónoma (Roma), Grupo de Investigación en Control Electrónico (Gice) y Grupo de Investigación de Sistemas Digitales Inteligentes (Digiti), a los cuales, los autores les agradecemos enormemente, y aunque esta obra es el resultado obtenido de las investigaciones de dos de ellos, las otras personas hicieron contribuciones importantes para la obtención de los resultados que aquí se presentan.

Otras personas importantes para esta investigación fueron los auxiliares de investigación y los profesionales externos que se encuentran en cada grupo, de los cuales nos permitimos mencionarlos individualmente, con el objetivo de hacerles un reconocimiento a su dedicación:

Ingeniera Leidy Yolanda López Osorio

Ingeniero Amed Esteban Vanegas

Ingeniero Jhonathan Cruz

Ingeniero Edwin Beltran

Asimismo, damos gracias a la Universidad Distrital Francisco José de Caldas, por haber aportado recursos económicos, logísticos y administrativos para el desarrollo de nuestro proyecto de investigación.

Finalmente, los autores expresamos gratitud a nuestras familias, por su incondicional apoyo y comprensión.

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

UD Editorial

ESPACIOS

Primera edición, diciembre de 2017
ISBN: 978-958-5434-85-1

Dirección Sección de Publicaciones
Rubén Eliécer Carvajalino C.

Coordinación editorial
María Elvira Mejía Pardo
Miguel Fernando Niño Roa

Corrección de estilo
Irina Florián Ortiz

Diagramación
Margoth de Olivos SAS

Editorial UD
Universidad Distrital Francisco José de Caldas
Carrera 24 No. 34-37
Teléfono: 3239300 ext. 6202
Correo electrónico: publicaciones@udistrital.edu.co

Rairán Antolines, José Danilo, 1971-
Multibot. Robótica aplicada al rescate urbano / José Danilo Rairán Antolines, Giovanni Rodrigo Bermúdez Bohórquez, Miguel Pérez Pereira. -- Bogotá : Universidad Distrital Francisco José de Caldas, 2017.
230 páginas ; 24 cm. -- (Colección espacios)
ISBN 978-958-5434-85-1
1. Robótica 2. Robots (Tecnología) 3. Ingeniería electrónica 4. Urbanismo - Robótica I. Bermúdez Bohórquez, Giovanni Rodrigo, autor II. Pérez Pereira, Miguel, autor III. Tít. IV. Serie.
629.892 cd 22 ed.
A1586313

CEP-Banco de la República-Biblioteca Luis Ángel Arango

Contenido

Agradecimientos 7

Prefacio 23

Búsqueda y rescate urbano 25
Equipos de búsqueda y rescate urbano 25
Técnicas de búsqueda 27
Técnicas de búsqueda en espacios cerrados 27
Técnica de levantamiento de croquis 28
Técnica de rastrillaje 28
Técnica circular externa sin rotación 29
Técnica circular externa con rotación 30
Inclusión de robots a los equipos *USAR* 30

Plataforma robótica DaNI 2.0 33
Especificaciones generales 34
Configuración dirección IP 35
Comunicación inalámbrica 36
Router Linksys E2500 37
Network Streams 38
Desarrollo y validación del modelo cinemático 39

Fusión sensorial 49
Sensor de calor TPA81 49
Sensor para el monitoreo de la humedad relativa y la temperatura SHT71 53
Conexión 54
Pruebas 55
Error absoluto y error relativo 57
Histéresis 59

Sensores para la detección de gases tóxicos o inflamables 59
Sensor de gas TGS3870 59
Sensor de gas MQ135 64
Sensores para la medición de la velocidad angular Giroscopio NGY1044 73
Sensor para determinar la localización. Brújula digital CMPS03 76
Conexión principal entre los sensores y la plataforma 80
Resultados 81
Algoritmos de fusión de sensores 82
Riesgo químico 84
Peligro de los gases a detectar 85
Entradas y salidas del sistema 87
Reglas 92
Pruebas del sistema de fusión sensorial 94
Cigarrillo 94
Madera 95
Sensor MQ135 (óxido nitroso) 96
Sensor MQ135 (amoniaco) 96
Sensor MQ135 (benceno) 97
Sensor TGS3870 (metano y monóxido de carbono) 97
Fotografías 98

Navegación del robot por filtros de Kalman 101
Determinación de parámetros para el filtro de Kalman 101
Modelo cinemático 101
Modelo odométrico 102
Fases del filtro de Kalman 103
Implementación del filtro 104
Algoritmo de evasión de obstáculos 106
Primera prueba 106
Segunda prueba 107
Tercera prueba 108

Visión artificial 111
Detección de obstáculos basado en cámara web 111
Adquisición de imágenes en LabVIEW 112
Algoritmo segmentación de imágenes 114
Detección de marcas naturales con Kinect de Microsoft 122
Principio de funcionamiento 122
Composición del Kinect 123

Kinect for Windows y Matlab 127
Kinect for Windows y LabVIEW 129
Obtención de parámetros para ajuste de coordenadas x, y en la matriz de profundidades 130
Adaptación del Kinect a la plataforma robótica 136
Extracción de marcas naturales 138

Control del robot basado en emociones 143
Selección de un modelo computacional de las emociones 143
Arquitectura del control basado en emociones 145
Definición de los estados emocionales para el controlador basado en emociones 146
Caracterización en la frecuencia del controlador basado en emociones 148
Modelo de la plataforma diferencial 152
Control clásico del robot 154
Modelo de referencia 158
Estrategia de control basado en emociones aplicada al robot 159
Aprendizaje por refuerzo 163
Controlador basado en emociones con aprendizaje por refuerzo 167
Conclusiones y trabajo futuro 170
Anexo 1. Diagrama de conexiones de una red neuronal con aprendizaje en línea, 4,4,2,1 171

Protocolo de comunicación del sistema Multirobot 173
Implementación de la comunicación por *sockets* 174
Java sockets 175
Modelo de comunicaciones con Java 176
Librerías para manejo de tareas de red 176

Sistema Multirobot cooperativo 183
Técnica de rastrillaje 184
Comportamientos reactivos básicos implementados a nivel de un robot 184
Capa Path 185
Comportamiento Path 185
Comportamiento Scan 187
Comportamiento Avoid 190
Capa Search 196
Capa Wander 198
Capa Call 205

Comportamientos reactivos básicos implementados a nivel Multirobot 207
Capa Sync 207
Capa Lineup 208
Capa Prey 212

Conclusiones 217

Referencias 221

Lista de figuras

Figura 1.	Técnica de levantamiento de croquis	28
Figura 2.	Técnica de rastrillaje en espacios cerrados	29
Figura 3.	Técnica circular externa sin rotación	29
Figura 4.	Técnica circular externa con rotación	30
Figura 5.	Rescate robotizado	31
Figura 6.	Robot DaNI 2.0 de National Instruments	33
Figura 7.	Measurement & Automation Explorer	35
Figura 8.	Configuración de la dirección IP estática para el robot DaNI 2.0	36
Figura 9.	*Router* Cisco Linksys E2500	37
Figura 10.	Configuración Network Stream (lectura)	38
Figura 11.	Configuración Network Stream (escritura)	39
Figura 12.	Descripción de una plataforma diferencial en el espacio	40
Figura 13.	Representación del modelo obtenido	42
Figura 14.	Comportamiento ideal en negro, comportamiento brindado por el modelo en gris, frente a una trayectoria circular	43
Figura 15.	Comportamiento ideal en negro, comportamiento brindado por el modelo en gris, frente a unas trayectorias circulares	43
Figura 16.	Se observa el comportamiento ideal en negro y la trayectoria estimada que tomó el modelo con un experimento de trayectoria aleatoria	44
Figura 17.	Error de medida y desviación estándar de los datos adquiridos del proceso de validación para el desplazamiento en línea recta	44
Figura 18.	Error de medida y desviación estándar de los datos adquiridos del proceso de validación para movimientos de rotación	45
Figura 19.	Iteraciones de la plataforma robótica respecto a un marco de referencia global (x, y)	46
Figura 20.	Sensor de calor TPA81	50
Figura 21.	Circuito impreso del sensor TPA81	51

Figura 22.	Ángulo de detección del sensor TPA81	52
Figura 23.	Exposición a fuentes de calor	52
Figura 24.	Sensor de humedad relativa y temperatura SHT71	53
Figura 25.	Conexión entre el sensor SHT71 y el microcontrolador PIC16F873a	54
Figura 26.	Circuito impreso para el sensor	55
Figura 27.	Datalogger RHT10	56
Figura 28.	Gráfico entregado por el RHT10	56
Figura 29.	Superior: prueba de humedad. Inferior: prueba de temperatura	57
Figura 30.	Superior: histéresis para la humedad. Inferior: histéresis para la temperatura	59
Figura 31.	Sensor TGS3870	60
Figura 32.	Circuito básico del sensor TGS3870	61
Figura 33.	Circuito impreso del sensor TGS3870	62
Figura 34.	Respuesta del sensor TGS3870 al gas natural domiciliario	63
Figura 35.	Respuesta del sensor TGS3870 al humo de un automóvil	64
Figura 36.	Sensor MQ135	64
Figura 37.	Circuito impreso de sensor MQ135	66
Figura 38.	Respuesta del MQ135 al hidróxido de amonio. (a) R = 900 KΩ (b) R= 500 KΩ (c) R = 200 KΩ (d) R = 40 Ω	66
Figura 39.	Respuesta del MQ135 al óxido nitroso. (a) R = 1 KΩ (b) R = 1.4 KΩ (c) R = 2 KΩ (d) R = 40 Ω	69
Figura 40.	Respuesta del MQ135 al benceno. (a) R = 1KΩ (b) R = 2.5 KΩ (c) R = 40 Ω (d) R = 2 KΩ	71
Figura 41.	Giroscopio NGY1044	73
Figura 42.	Conector RJ45 Lego NXT	74
Figura 43.	Circuito impreso para el sensor NGY1044	75
Figura 44.	Brújula digital CMPS03	77
Figura 45.	Circuito impreso para brújula digital CMPS03	78
Figura 46.	Circuito impreso para la conexión de sensores y la tarjeta SBRIO 9632	80
Figura 47.	Propuesta de fusión sensorial	83
Figura 48a.	Fuzzificación de la humedad relativa	88
Figura 48b.	Fuzzificación de la temperatura	88
Figura 49.	Fuzzificación para el calor radiante	88
Figura 50.	Fuzzificación para el metano	89

Figura 51.	Fuzzificación para el monóxido de carbono	89
Figura 52.	Fuzzificación para el óxido nitroso	90
Figura 53.	Fuzzificación para el amoniaco	90
Figura 54.	Fuzzificación para el benceno	90
Figura 55.	Fuzzificación para incendio	91
Figura 56.	Fuzzificación para intoxicación	91
Figura 57.	Fuzzificación para explosión	92
Figura 58.	Test System de LabVIEW	94
Figura 59.	Componentes del cigarrillo	95
Figura 60.	Respuesta del sensor MQ135 al óxido nitroso contenido en el humo emanado por el carbón	96
Figura 61.	Respuesta del sensor MQ135 al amoniaco contenido en el humo emanado por los cigarrillos	96
Figura 62.	Respuesta sensor MQ135	97
Figura 63.	Respuesta del sensor TGS3870 al humo emanado por los cigarrillos	97
Figura 64.	Plataforma robótica DaNI 2.0	98
Figura 65.	Exposición a cigarrillo y carbón	98
Figura 66.	Exposición a fuentes de calor y cambios de temperatura y humedad relativa	99
Figura 67.	Respuesta del filtro de Kalman	105
Figura 68.	Algoritmo de evasión de obstáculos	106
Figura 69.	Distribución de obstáculos-prueba 1	107
Figura 70.	Trayectoria realizada por la plataforma para la primera prueba; estimación de la posición realizada por el filtro de Kalman	107
Figura 71.	Distribución de los obstáculos-prueba 2	108
Figura 72.	Prueba 2. Izquierda: trayectoria de la plataforma. Derecha: estimación de la posición	108
Figura 73.	Distribución de obstáculos-prueba 3	109
Figura 74.	Prueba 3. Izquierda: trayectoria de la plataforma. Derecha: estimación de la posición	109
Figura 75.	Pruebas al algoritmo de evasión de obstáculos	110
Figura 76.	Logitech QuickCam 9000 pro	112
Figura 77.	Configuración de atributos de adquisición	113
Figura 78.	Algoritmo para la adquisición de imágenes	114
Figura 79.	Diagrama general del sistema implementado	115

Figura 80.	Matlab Script Node	116
Figura 81.	Diagrama de flujo del algoritmo de captura, detección y segmentación de imágenes	118
Figura 82.	Filtro de huecos, imagen tomada de Matworks	119
Figura 83.	(a) Imagen tomada por la cámara, (b) Imagen binarizada, (c) Imagen resultante filtro elementos, (d) Imagen resultante filtro huecos	119
Figura 84.	Segmentación de imágenes	120
Figura 85a.	Imagen a escala de grises	
Figura 85b.	Resultado final del algoritmo	120
Figura 86.	Diagrama de flujo del algoritmo esclavo	121
Figura 87.	*Kinect for Windows de Microsoft*	122
Figura 88.	Patrones de luz estructurada a) Líneas, b) Puntos sensor Kinect	123
Figura 89.	Principio de triangulación aplicado a líneas	123
Figura 90.	Partes principales del Kinect de Microsoft	124
Figura 91.	Motor para ajuste de inclinación del Kinect	125
Figura 92.	Ubicación de los micrófonos del Kinect de Microsoft	126
Figura 93.	Esquema general del Kinect de Microsoft	127
Figura 94.	Panel Frontal Kinect_Test.vi	130
Figura 95.	Imagen referencia para la obtención de parámetros	131
Figura 96.	Imagen referencia binarizada	132
Figura 97.	Imagen referencia binarizada y ampliada	132
Figura 98.	Imagen referencia con mediciones en los extremos de la cuadrícula	133
Figura 99.	Ángulos de visión del Kinect de Microsoft	134
Figura 100.	Imágenes obtenidas por (a) cámara VGA (b) sensor de profundidad (c) gráfica en coordenadas rectangulares	136
Figura 101.	Kinect Sensor TV Mounting Clip	137
Figura 102.	Foto de la plataforma con el Kinect	137
Figura 103.	Discretización de la imagen de profundidad	138
Figura 104.	Gráfica ejemplo de una fila de la matriz de profundidades en coordenadas cartesianas	139
Figura 105.	Imagen de una esquina capturada con el Kinect (izquierda). Detección de marcas naturales (derecha)	141
Figura 106.	Modelo circumplejo de las emociones	144
Figura 107.	Definición del estado emocional α	145

Figura 108. Rendimiento de la arquitectura de control nueva 146
Figura 109. Demostración experimental de estabilidad. 147
Figura 110. Montaje para evaluar el rendimiento del controlador emocional 150
Figura 111. Caracterización en la frecuencia 151
Figura 112. Modelo del robot en Simulink 152
Figura 113. Modelo de los motores del robot 153
Figura 114. Simulación del movimiento del robot 153
Figura 115. Evaluación del radio y ángulo de dirección del robot 154
Figura 116. Configuración de los dos controladores para manipular al robot 155
Figura 117. Control de posición 156
Figura 118. Modelo de referencia 159
Figura 119. Robot y modelo de referencia 160
Figura 120. Definición del estado emocional del sistema dinámico 160
Figura 121. Medición de ángulos 161
Figura 122. Definición del estado emocional del sistema dinámico para el ángulo 162
Figura 123. Comparación entre las dos definiciones de error 162
Figura 124. Red neuronal perceptron multicapa 165
Figura 125. Configuración de la red neuronal 166
Figura 126. Conexión de la neurona *N11*. 167
Figura 127. Control con aprendizaje por refuerzo. 167
Figura 128. Controlador emocional con aprendizaje por refuerzo. 168
Figura 129. Resultado del control basado en emociones con aprendizaje por refuerzo 169
Figura 130. Recorrido de la plataforma con el controlador final 169
Figura 131. Infraestructura tipo cliente-servidor 173
Figura 132. Petición de conexión arquitectura cliente-servidor 175
Figura 133. Comunicación cliente-servidor 175
Figura 134. Modelo de comunicación por *sockets* en Java 176
Figura 135. Librerías implementadas en código fuente 177
Figura 136. Sintaxis de implementación para comunicación TCP/IP 177
Figura 137. Evento de conexión 178
Figura 138. Parámetros de configuración de red de la interfaz ethernet 179
Figura 139. Transmisión de datos (sintaxis) 180
Figura 140. Trama general de comunicaciones 180

Figura 141. Descripción de comportamientos a nivel de un solo robot 185
Figura 142. Descripción I/O para el comportamiento Path 186
Figura 143. Máquina de estados para la descripción del comportamiento Path 187
Figura 144. Resultados del proceso de validación para el comportamiento Path 188
Figura 145. Descripción I/O para el comportamiento SCAN 188
Figura 146. Diagrama de flujo para la implementación del comportamiento SCAN 189
Figura 147. Experimentación y validación del comportamiento SCAN 189
Figura 148. Descripción I/O para el comportamiento Avoid 190
Figura 149. Relación entre la celda de obstáculo y el robot 192
Figura 150. Naturaleza del proceso de selección de dirección en el algoritmo VFH+ 193
Figura 151. Resultados del proceso de validación del comportamiento Avoid 195
Figura 152. Resultados del proceso de validación del comportamiento Avoid 195
Figura 153. Descripción I/O para el comportamiento Search 196
Figura 154. Diagrama de autómatas finitos para el comportamiento Search 197
Figura 155. Resultado del proceso de validación del comportamiento Search 197
Figura 156. Descripción I/O para el comportamiento Sail 198
Figura 157. Máquina de estado para la implementación del comportamiento Sail 199
Figura 158. Ruta realizada por el robot como parte del proceso de experimentación para la validación del comportamiento Sail 200
Figura 159. Descripción I/O para el comportamiento SCAN 201
Figura 160. Escaneo del entorno realizado por el robot cuando se encuentra describiendo una trayectoria en línea recta por un pasillo 201
Figura 161. Escaneo del entorno realizado por el robot ante un obstáculo al frente 201
Figura 162. Escaneo del entorno realizado por el robot cuando se encuentre en una esquina interior 202
Figura 163. Escaneo del entorno realizado por el robot cuando se encuentre en una esquina interior 202
Figura 164. Trayectoria descrita por el robot en el proceso de búsqueda y localización de una fuente de calor 203
Figura 165. Trayectoria descrita por el robot en el proceso de búsqueda y localización de una fuente de calor 204

Figura 166. Path, Scan, Avoid, Sail 205
Figura 167. Descripción I/O para el comportamiento Call 206
Figura 168. Descripción de máquinas de estado para el comportamiento Call 206
Figura 169. Descripción de comportamientos a nivel Multirobot 207
Figura 170. Descripción I/O para el comportamiento Sync 208
Figura 171. Descripción I/O para el comportamiento Train 208
Figura 172. Prueba Path Multirobot 209
Figura 173. Descripción I/O para el comportamiento Sweep 210
Figura 174. Comportamiento Sweep 210
Figura 175. Descripción I/O para el comportamiento Route 211
Figura 176. Prueba Avoid Multirobot 212
Figura 177. Segunda prueba Avoid Multirobot 212
Figura 178. Descripción I/O para el comportamiento Sense 213
Figura 179. Descripción I/O para el comportamiento Target 213
Figura 180. Validación de la capa comportamental Prey 214
Figura 181. Validación de la capa comportamental Prey 215

Lista de tablas

Tabla 1. Especificaciones generales del robot móvil 34
Tabla 2. Especificaciones del *router* E2500 37
Tabla 3. Características sensor TPA81 50
Tabla 4. Características sensor SHT71 53
Tabla 5. Meq10didas del sensor SHT71 en un punto de la escala 58
Tabla 6. Características sensor TGS3870 61
Tabla 7. Características Sensor MQ135 65
Tabla 8. Prueba al Gyro de NXT 75
Tabla 9. Medidas del sensor SHT71 en un punto de la escala 76
Tabla 10. Características CMPS03 77
Tabla 11. Prueba a la brújula magnética CMPS03 79
Tabla 12. Medidas del sensor CMPS03 para 150° 79
Tabla 13. Resultados 81
Tabla 14. Composición del humo del cigarrillo 95
Tabla 15. Tipos de datos del Matlab Script Node 116
Tabla 16. Mediciones realizadas en los extremos de la cuadrícula dadas en milímetros 133
Tabla 18. Código Matlab para simular el robot, parte a. 156
Tabla 17. Definición de los estados emocionales 148
Tabla 19. Código Matlab para simular el robot, parte b. 157
Tabla 20. Código Matlab para simular el robot, parte c. 157
Tabla 21. Configuración de las principales funciones que se utilizan para realizar una comunicación cliente-servidor 178
Tabla 22. Formato de trama 181

Prefacio

La operación de rescates en desastres es un suceso de importancia social, en el cual agentes con diferentes características interactúan con un fin específico: salvar vidas. Las condiciones a las cuales se enfrentan estos agentes son desconocidas y, en muchos casos, son ambientes devastados, que disminuyen los factores de seguridad y aumentan el riesgo; ponen en peligro su integridad y disminuyen la posibilidad de éxito en los rescates.

Así es como el Grupo de Investigación en Robótica Móvil Autónoma (Roma), el Grupo de Investigación en Control Electrónico (Gice) y el Grupo de Investigación de Sistemas Digitales Inteligentes (Digiti) presentan como solución el desarrollo de un sistema multiagente mediante la implementación de algoritmos cooperativos, aprendizaje por refuerzo basado en emociones humanas y sistemas bioinspirados, que permita obtener un equipo de rescate urbano que facilite las operaciones en la atención de emergencias.

El objetivo final del proyecto es presentar la prueba de alternativas de robótica cooperativa en el rescate de víctimas en zonas colapsadas simuladas. Esto con el fin de ofrecer más oportunidades de vida en este tipo de siniestros. Además, se presenta el estudio de sistemas de control autónomo, para lo cual se toma como base algoritmos existentes de control bioinspirado e implementando uno de ellos a una plataforma robótica comercial.

Este libro está conformado por 8 capítulos que estructuran la información que fue relevante para el desarrollo del proyecto de investigación. En el capítulo 1 se expone una revisión bibliográfica relacionada con los equipos de búsqueda y rescate urbano; técnicas de búsqueda y la inclusión de la robótica en las tareas de rescate. En el capítulo 2 se presenta una descripción detallada de las plataformas robóticas que se implementaron, las llamadas Labview Robotics Sbrio Starter Kit 2.0 (DaNI) de la empresa National Instruments. En el capítulo 3 se da a conocer el estudio de todos los sensores que se implementaron en la plataforma, así como los algoritmos de función sensorial que se efectuaron. En el capítulo 4 se documenta sobre el algoritmo sobre los algoritmos que se probaron para la navegación de las plataformas y se hace énfasis en el algoritmo de calma como principal sistema para la corrección de errores de desplazamientos. En el capítulo 5 se muestran varias técnicas de visión artificial empleadas para la orientación de la plataforma y la evasión de obstáculos. En

el capítulo 6 se expone el control de la plataforma robótica basado en emociones. En el capítulo 7 se hace un planteamiento del protocolo de comunicaciones que se implementó en el equipo Multirobot y, finalmente, en el capítulo 8 se muestra el sistema Multirobot cooperativo.

Búsqueda y rescate urbano

Salvar vidas humanas durante una catástrofe (terremotos, explosiones, colapsos estructurales, etc.) resulta ser la tarea primordial de los equipos de búsqueda y rescate urbano (*USAR*, por sus siglas en inglés [Urban Search and Rescue Robotics]). Para este fin, cuentan con un recurso humano y animal especializado en labores de búsqueda, localización y rescate de víctimas.

Cuando las condiciones a las que se enfrentan exigen una mayor precisión y rapidez, es necesario contar con un recurso adicional, ya que no se sabe a qué se pueda enfrentar el equipo de rescate. En este punto, la robótica se presenta como una herramienta de ayuda y soporte para mejorar las condiciones del equipo *USAR*.

En este capítulo se presenta una recopilación de la revisión bibliográfica relacionada con los equipos de búsqueda y rescate urbano; técnicas de búsqueda y la inclusión de la robótica en las tareas de rescate.

Equipos de búsqueda y rescate urbano

El término *USAR* involucra todo lo relacionado con el rescate de una o más víctimas de una zona colapsada, que por lo general es una estructura artificial creada por el hombre con materiales como acero, hierro, concreto, vidrio, etc. Dentro de los factores que caracterizan a una zona colapsada están que no existe un gran rango de visibilidad, hay poca presencia de luz, altos índices de polvo y gases condensados y fuga de elementos gaseosos que se constituyen como altamente riesgosos para el ser humano.

Los equipos de búsqueda y rescate son agrupaciones de personas que se encargan de localizar, retirar y prestar asistencia a personas que estén en peligro en la escena de un evento adverso. Estos tienen un jefe de búsqueda y un jefe de rescate, ya que cada una de estas actividades requieren de diferentes habilidades y, para ello, siempre se debe designar al más capacitado en cada una de las especialidades [1].

Cada equipo *USAR* está compuesto de 2 grupos, cada uno de 31 personas, 4 canes y equipo especializado. Las tareas que se ejecutan en un equipo de esta clase se dividen en 4 especialidades:

- *Búsqueda.* El objetivo de esta tarea es buscar y localizar la víctima que está atrapada en la zona colapsada.

- *Rescate.* En esta, el equipo se encarga de sacar a la víctima del lugar en riesgo lo más rápido posible.
- *Técnico*. Las labores que se ejecutan en esta especialidad radican básicamente en el estudio de las zonas, los caminos óptimos y menos riesgosos y la elaboración de las estrategias de búsqueda y rescate.
- *Médico*. Las tareas que se ejecutan están enfocadas en la prestación de apoyo médico tanto a las víctimas como al equipo *USAR*, prestando primeros auxilios y la atención en primer grado para la estabilización y el posterior tratamiento especializado.

Según la Federal Emergency Management Agency (Fema), los integrantes de un equipo *USAR* deben [1]:

- Estar atentos ante cualquier emergencia y, una vez informados de la existencia de alguna, acudir de manera inmediata como el primer equipo especializado que se presenta en la zona.
- Una vez situados en la zona de desastre, obtener la asesoría de un experto sobre las condiciones estructurales actuales de la zona, esto con el fin de evitar entradas imprudentes y que puedan poner en riesgo la vida del rescatista.
- Llevar a cabo las tareas de búsqueda con la ayuda de un equipo electrónico que sirva de soporte para mejorar las habilidades de detección de víctimas.
- Si la víctima se encuentra rodeada de estructuras arruinadas, el rescatista está en la capacidad de adecuar la zona para empezar con el proceso de rescate.
- Si hace parte del equipo médico, el integrante debe ser una persona creativa y serena con capacidad de actuar tan rápido como la situación lo amerite, sin llegar a caer en errores de precisión en el procedimiento médico, ya que de ser así la víctima no podría sobrevivir al accidente.
- El integrante de un equipo de rescate nunca debe poner en riesgo su vida por tratar de rescatar a una víctima, ya que esto lo único que generaría sería más confusión y una posible víctima más.
- Antes de empezar las labores de búsqueda y rescate, el rescatista debe asegurarse de tener todos y cada uno de los elementos para efectuar su tarea, esto debe hacerse tan rápido y preciso como se presente la situación.

La agencia Fema se constituyó como la primera organización mundial que reglamentó administrativa y técnicamente las labores de búsqueda y rescate urbano, por lo que cada una de las actividades que se realizan en una situación de desastre quedan documentadas con precisión en cada uno de los formatos que se manejan para esta labor, en ellos se encuentran guías de elaboración de estrategias, estructuración de informes, reportes de situación, asignación de equipos y materiales, procedimientos ante fallos catastróficos, protocolos de rescate adecuados, plan de comunicaciones y

determinación de tratamientos médicos. En los Estados Unidos, el sistema nacional de búsqueda y rescate agremia gran cantidad de participantes que se dividen en 3 categorías:

- Fema. Es la encargada de establecer políticas y coordinar las tareas de búsqueda y rescate.
- Fuerzas especiales o fuerzas operativas de tarea. En los Estados Unidos, existen alrededor de veintiocho equipos de esta naturaleza. Cada uno se compone por 130 personas altamente entrenadas, de las cuales se destacan bomberos, ingenieros, médicos, personal de manejo canino, entre otros.
- Equipos de soporte técnico. Como su nombre lo indica, son profesionales altamente entrenados que están encargados de prestar ayuda técnica a las fuerzas operativas de tarea por medio de la integración de tecnologías y ayudas audiovisuales como rastreadores, cámaras, robots, etc.

Técnicas de búsqueda

Las técnicas de búsqueda que implementan un equipo *USAR* varían de acuerdo con la situación y se dividen en dos grandes categorías, dependiendo de las características del medio ambiente donde se van a implementar: i) en espacio abierto; y ii) en espacio cerrado.

En la etapa estratégica, la determinación de las zonas de búsqueda depende de la situación actual de la emergencia. En este punto, existen cinco métodos principales para determinar las zonas óptimas de búsqueda [2]:

- Método teórico. Consiste en calcular cuánto se puede desplazar un equipo bajo determinadas condiciones geográficas, luego se traza un radio de acción y se proyecta una circunferencia sobre este.
- Método estadístico. Surge de los estudios que se hayan realizado en la zona, esto permite utilizar herramientas estadísticas para determinar cuál es la zona con mayor probabilidad de alojar víctimas.

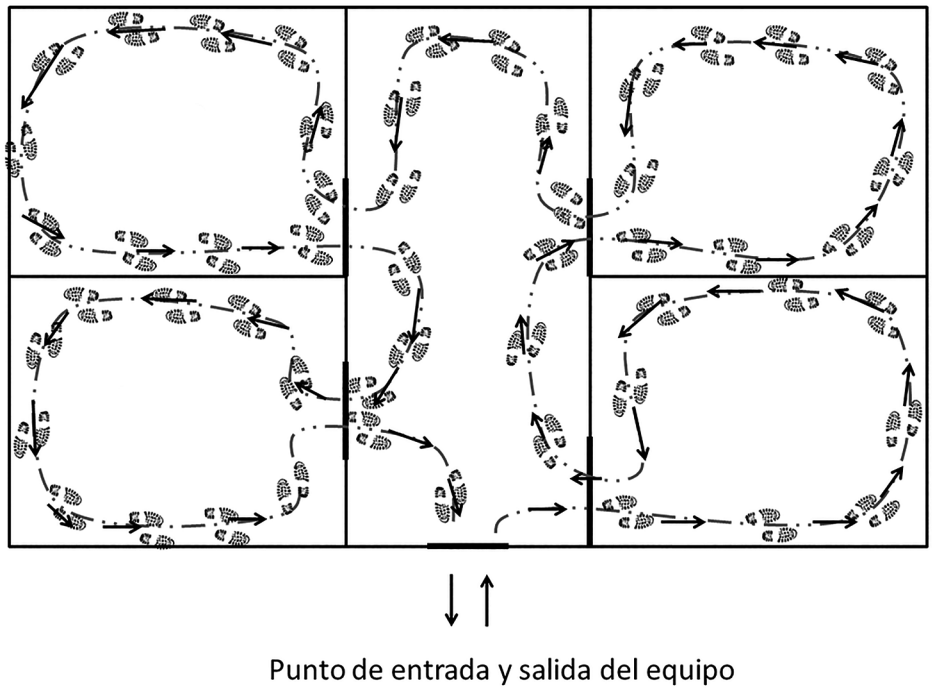
Técnicas de búsqueda en espacios cerrados

Se presentan cuando una estructura creada por el hombre sufre daños debido a un evento natural como un terremoto, tornado, huracán, entre otros, o en su defecto, cuando es causado por el mismo hombre. En estas situaciones, la integridad de los rescatistas se ve altamente afectada, ya que, debido a factores como la inestabilidad del terreno, los fallos en los conductos de gas, la ruptura de vidrios y estructuras metálicas cada procedimiento es altamente peligroso. En estas zonas, por lo general, los equipos no cuentan con croquis de sitio, por esto se envían dos integrantes del equipo para su generación. Para este, los expertos han clasificado cuatro técnicas principales para la búsqueda en espacios cerrados [2].

Técnica de levantamiento de croquis

El método consiste en entrar a la zona por la derecha y seguir por las paredes hasta recorrer la totalidad de la estructura, aunque la principal tarea del equipo es realizar un croquis lo más exacto que se pueda de la zona colapsada, también podrá rescatar víctimas que se encuentren en su camino. Una vez obtenido el croquis y establecidas las zonas de búsqueda (figura 1), el equipo se divide en grupos que deben tener un líder de búsqueda, que es el encargado de poner en ejecución las técnicas de búsqueda y rescate.

Figura 1. Técnica de levantamiento de croquis

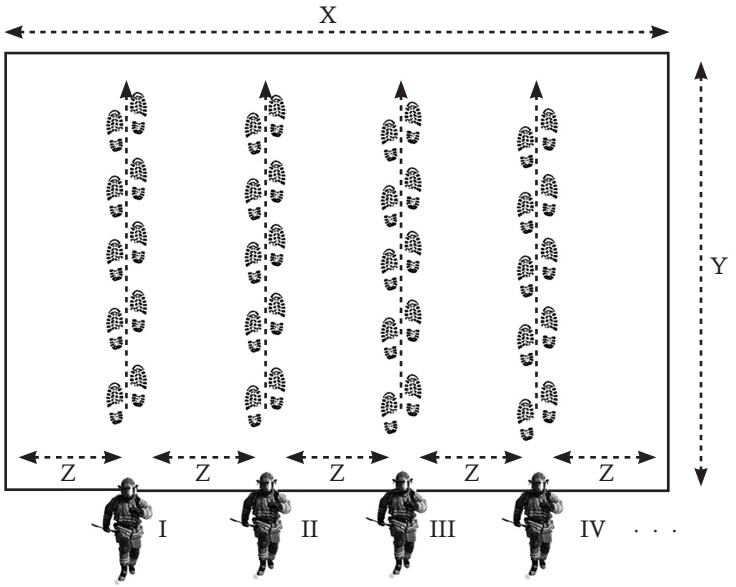


Fuente: [2].

Técnica de rastrillaje

En esta técnica, los desplazamientos del equipo son simultáneos y discreteados, es decir, a cada cierta distancia se realizan paradas para determinar el estado de los rescatistas y de las posibles víctimas; se debe contar con un numeroso grupo de trabajo, cada persona debe tener un equipo de comunicaciones, cartografía y silbato. Se parte de una línea imaginaria que brinde la suficiente cobertura de la zona de búsqueda en la que se considera que se hallan las víctimas. Luego, se asigna un punto de búsqueda para cada rescatista (figura 2).

Figura 2. Técnica de rastrillaje en espacios cerrados

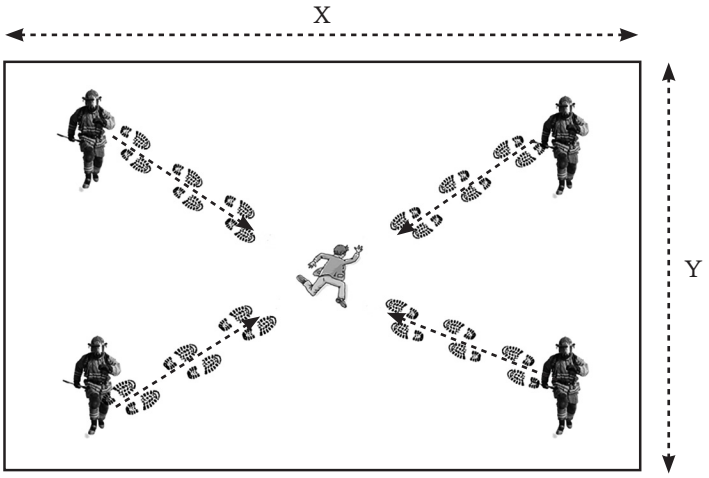


Fuente: [2].

Técnica circular externa sin rotación

En esta técnica de búsqueda, los cuatro rescatistas se ubican en los puntos de escucha seleccionados y con su respectiva copia del croquis (figura 3), el líder solicita hacer silencio absoluto y, en voz alta, se dirige a las posibles víctimas atrapadas indicándoles que griten o que emitan algún sonido. También, puede golpear fuertemente uno de los componentes metálicos de la edificación con algún objeto sólido (al menos 3-5 veces).

Figura 3. Técnica circular externa sin rotación

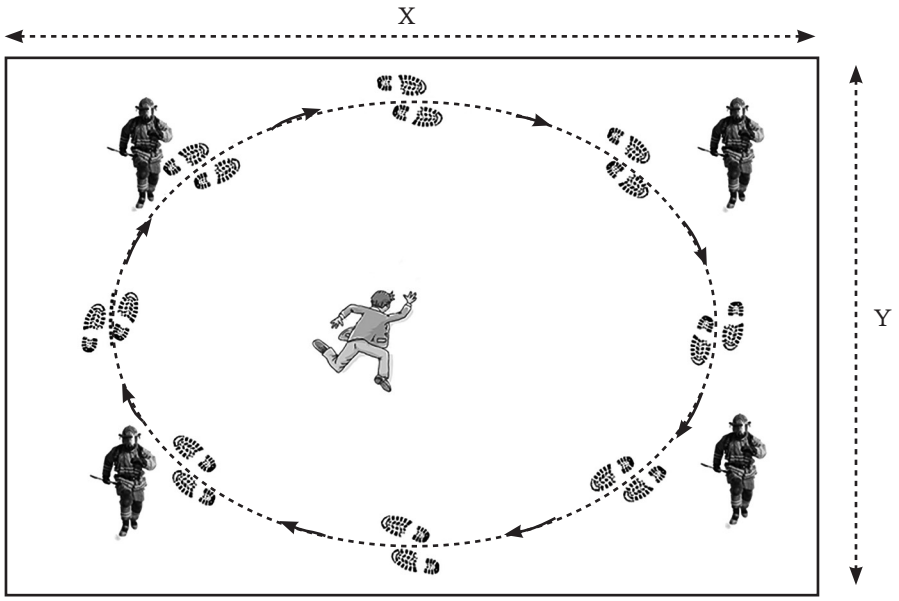


Fuente: [2].

Técnica circular externa con rotación

Funciona de manera análoga a la técnica anterior, la diferencia radica en que si al primer llamado no hay respuesta, los rescatistas avanzan hasta la próxima posición en dirección de las manecillas del reloj (figura 4).

Figura 4. Técnica circular externa con rotación



Fuente: [2].

Inclusión de robots a los equipos *USAR*

La estructura organizacional de los grandes equipos de búsqueda y rescate urbano en el mundo hace pensar que aún existen muchos retos por afrontar; uno de ellos es la integración de los equipos robóticos a los equipos *USAR*.

Aunque existe desconfianza en dejarle a un equipo robótico las tareas de búsqueda y rescate, una verdadera misión de socorro requiere operar por 10 días consecutivos en el peor de los casos, autoabasteciéndose, lo que resulta ser bastante agotador física y mentalmente. Cada miembro que opera en zona roja debe remover pilas de escombros, operar en sitios comandos, saltar y evadir estructuras arruinadas y operar en alturas si el caso lo requiere. En la mayoría de los casos, un perro detecta la presencia de una víctima; una vez determinada la zona de acceso, el equipo rodea la zona buscando el acceso más adecuado, para ello los rescatistas usan cámaras y equipos de video que pueden llegar a acceder visualmente hasta 12 pies de profundidad según el hueco, pero cuando el escenario imposibilita el acceso visual, aparece la primera de las necesidades para pensar en la aplicación de un equipo robótico.

Un robot se puede considerar como un conjunto de sistemas embebidos que se combinan entre sí para interactuar en entornos reales de manera dinámica. Sin embargo, está compuesto por sensores y actuadores que pueden fallar o, con el tiempo, degradar. Y para poder interactuar, deben existir varios niveles de autonomía y cognición. Las anteriores dimensiones son consideradas como verdaderos retos para los diseñadores de equipos robóticos.

Algunos estudios [3, 4, 5, 6] han revelado algunas de las más relevantes características de este tipo de interacción que se ha evaluado con robots autónomos y con robots teleoperados (figura 5). Sin embargo, el verdadero problema no se encuentra en la situación descrita anteriormente, radica en el mismo ser humano, que en muchos ca-sos subestima y desconfía de las capacidades de un robot, por lo que crea una barrera subjetiva que no le permite interactuar o interpretar de buena manera la información que le suministra el robot.

Figura 5. Rescate robotizado

Fuente: [6].

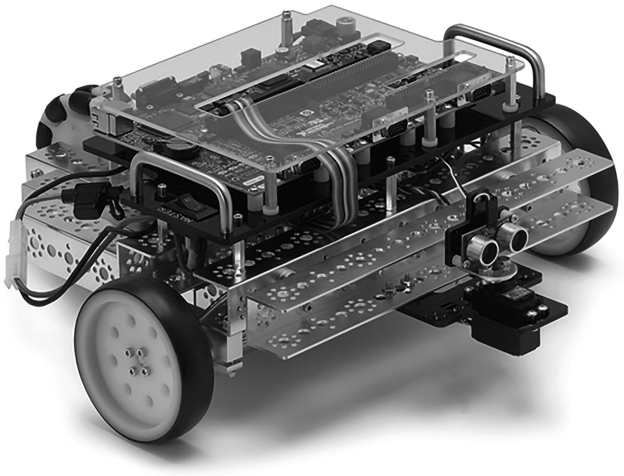
A pesar de que esta situación hoy en día es menos frecuente gracias a algunos estudios y prácticas [6], el verdadero estado de los equipos de búsqueda y rescate que utilizan robots es bastante limitado y aun no se han definido políticas mundiales para la profesionalización de esta útil herramienta.

Plataforma robótica DaNI 2.0

Las plataformas robóticas móviles implementadas en el proyecto, comercialmente llamadas LabviewRoboticsSbrio Starter Kit 2.0 (DaNI) de la empresa National Instruments, fueron previamente adquiridas por el grupo de investigación Roma de la Universidad Distrital Francisco José de Caldas, de la Facultad Tecnológica. Dicha plataforma tiene una configuración diferencial y se basa en un controlador sbRIO-9632 que es programado utilizando el *software* LabVIEW y gracias a las capacidades de los módulos FPGA, Real Time y Robotics (empleado para aplicaciones robóticas), se consigue una rápida familiarización con el manejo y funcionamiento de todo el sistema [8]. Las principales características del prototipo son:

- Totalmente ensamblado sobre una base de robot móvil.
- La toma de decisiones es en tiempo real y el procesamiento de las entradas y salidas es realizado por un dispositivo FPGA.
- Fácil conexión a una variedad de sensores y actuadores de uso robótico.

Figura 6. Robot DaNI 2.0 de National Instruments



Fuente: [8].

Especificaciones generales

En la tabla 1 se describen y se muestran todos los elementos incluidos en el robot móvil LabVIEWRobotics Starter Kit.

Tabla 1. Especificaciones generales del robot móvil

Especificaciones generales del robot LabVIEW RoboticssbRIO Starter Kit		
Características Físicas	**Descripción**	**Figura**
Chasis	Plataforma robótica TETRIX de Pitsco Tamaño: 405 mm x 368 mm x 150 mm (15.9'' x 14.5'' x 5.9'')	
Motores	2 motores de 12 VDCTETRIX de Pitsco Velocidad angular: 152 rpm Torque: 300 oz-in.	
Sensores	1 sensor ultrasónico PING))) de Parallax. Mide distancias entre 3 cm a 3 m.	
	2 encodersópticos de cuadratura 400 pulsos por revolución.	
Servomotor	Servomotor Estándar de Parallax Sirve de soporte de montaje para el sensor PING))). Realizaun barrido de 180 gradosdelentorno.	
Tarjeta de Control	Tarjeta Single-Board RIO (sbRIO-9631), de National Instruments. Versión 1.0. Tarjeta Single-Board RIO (sbRIO-9632), de National Instruments. Versión 2.0.	
Fuente de energía	Paquete de Baterías Recargables NiMH de12 VDC.	
Cargador de baterías	Cargador de baterías NiMH. Autodetecta el voltaje de la batería NiMH, configura el voltaje adecuado de carga, y corta la corriente eléctrica automáticamente cuando la batería está completamente cargada.	
Peso total	3.6 kg (7.9 lb)	_
Tipo de configuración	Diferencial.	_

Especificaciones generales del robot LabVIEW RoboticssbRIO Starter Kit		
Características Físicas	**Descripción**	**Figura**
Entorno de programación	LabVIEW, específicamente los módulos LabVIEWFPGA, LabVIEWRealTime y LabVIEWRobotics.	–

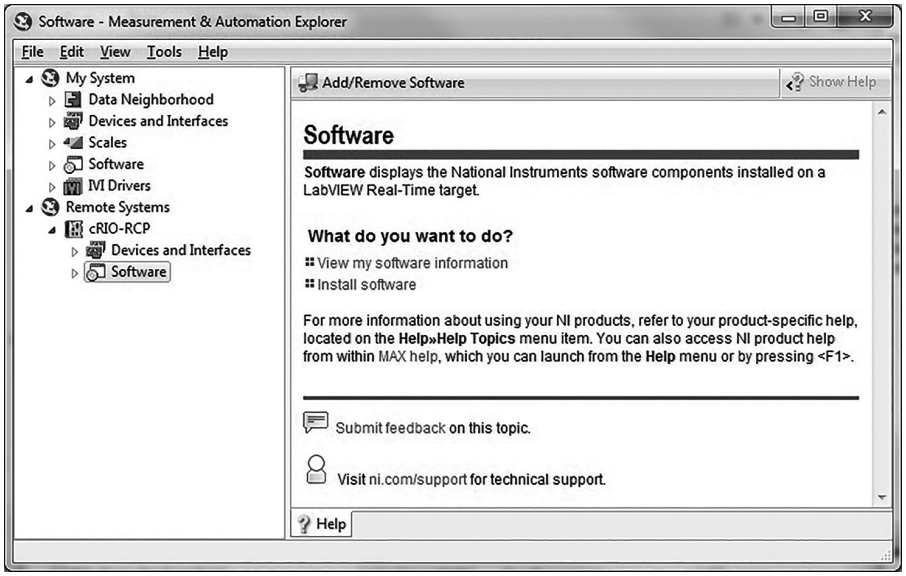
Fuente: [9].

Configuración dirección IP

Para poder acceder a la plataforma robótica DaNI 2.0 se debe configurar el *software* o controlador, configurar puertos o descargar el programa desarrollado en LabVIEWRobotics; es necesario configurar una dirección IP para el dispositivo, ya que la comunicación entre la plataforma y el computador se realiza mediante una conexión cableada tipo LAN.

Para configurar dicha dirección IP, es necesario conectar la plataforma robótica y el computador por medio de sus puertos RJ-45, respectivamente. Cuando se establezca comunicación, se procede a ejecutar el programa de National Instruments llamado Measurement & Automation Explorer (NI MAX), el cual permite visualizar los dispositivos compatibles con LabVIEW conectados al computador.

Figura 7. Measurement & Automation Explorer



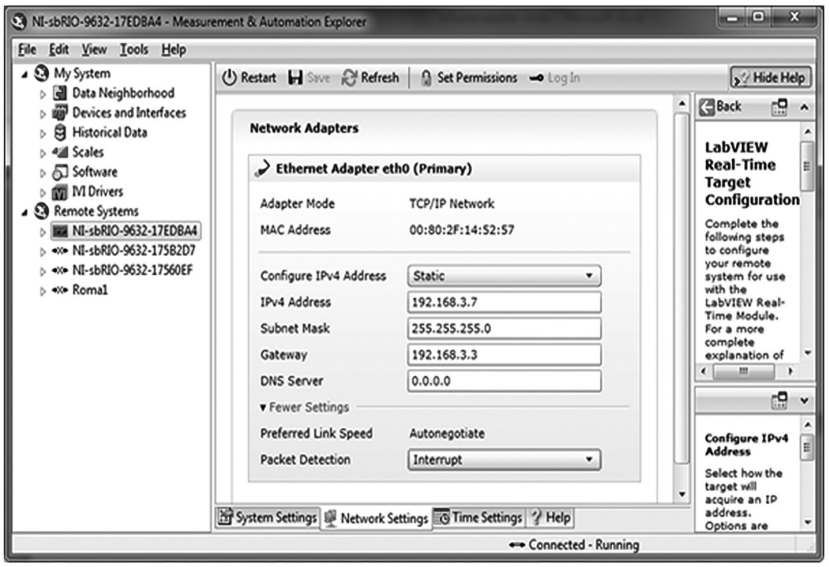
Fuente: [10].

Después, se debe desplegar la pestaña RemoteSystems, y allí aparecerá la tarjeta Single-Board RIO asociada a la plataforma conectada al computador; una vez seleccionada, en la parte central de la ventana aparecen las configuraciones del sistema, la configuración de red y la configuración de hora para la tarjeta.

Para configurar una dirección IP estática en el dispositivo, se debe seleccionar la pestaña Network Settings y en la opción Configure IPv4Address se selecciona 'Static', y luego se completan los datos adicionales como la IP por utilizar, la máscara de subred, la puerta de enlace y un servidor DNS. Luego de completar los campos requeridos, se debe pulsar el botón Save, ubicado en la parte superior central de la ventana, de esta manera el robot se reiniciará y almacenará las configuraciones deseadas por el usuario.

En el caso específico de las plataformas robóticas DaNI 2.0, existentes en el grupo de investigación ROMA, se asignaron direcciones IP número 192.168.3.X con su respectiva máscara de red número 255.255.255.0, correspondiente a una red clase C; esto con el objetivo de configurar una red doméstica que cumpliera con los requerimientos en cuanto a cantidad de dispositivos conectados y baja complejidad de conexión entre ellos.

Figura 8. Configuración de la dirección IP estática para el robot DaNI 2.0



Una vez es configurado y reiniciado el robot, en la parte inferior de la ventana del NI MAX debe aparecer su estado actual, como "Connected – Running", que indica que el dispositivo está listo para usarse.

Comunicación inalámbrica

La tarjeta Single-Board RIO (sbRIO-9632) de National Instruments posee un puerto ethernet, el cual permite una comunicación cableada tipo LAN entre el computador y la plataforma robótica; sin embargo, este tipo de comunicación no es una buena solución para las aplicaciones que requieren el libre movimiento del robot, por tal motivo, se debe implementar una comunicación inalámbrica que, en nuestro caso, estuvo basada en tecnología wifi, adaptando un *router* LinksysE2500 de Cisco a la plataforma robótica DaNI 2.0.

Router Linksys E2500

El Cisco LinksysE2500 es un *router* de gama media dirigido a aquellos usuarios que no le dan un uso muy intensivo a internet ni a la red local. Las principales características de este equipo es que tiene 4 puertos ethernet a 100 Mbps y wifi N a 300 Mbps con doble banda simultánea, adicionalmente integra compatibilidad IPv6 con el firmware de fábrica (última versión de firmware) [8].

Figura 9. *Router* Cisco Linksys E2500

Adicionalmente, las especificaciones proporcionadas por la página oficial de Cisco se describen en la tabla 2.

Tabla 2. Especificaciones del *router* E2500

Modelo:	**LinksysE2500**
Tecnología:	Inalámbrica N
Bandas:	2,4 GHz y 5 GHz simultáneas
Transmisión/recepción:	2 x 2
Antenas:	4 (internas)
Puertos Ethernet x velocidad:	4 x 10/100
Puerto de almacenamiento USB:	Sin puerto USB
Configuración del *software*:	CD de instalación
Software Cisco Connect:	Sí
CPU	Broadcom BCM5358UB0KFBG
Frecuencia de reloj	533MHzMIPS32
Controlador de radio banda 5GHZ	Broadcom BCM43236KMLG
Memoria RAM	EtronTechEM6AB160TSC-5G, 64 Mb
Memoria flash	Winbond25Q64BVS1G, 8Mb
Alimentación	12V, 1A

Fuente: [9].

Network Streams

La comunicación inalámbrica implementada para el intercambio de datos entre la plataforma robótica DaNI 2.0 y el computador está basada en el protocolo TCP/IP; sin embargo, este se basa directamente en el envío y la recepción de datos, sin tener en cuenta posibles fallas en el momento de tratarse de una comunicación tipo wifi. Ante esta necesidad, LabVIEW cuenta con bloques que permiten realizar la configuración de un protocolo llamado Network Streams, en el cual se deben especificar las direcciones IP de la plataforma robótica y del computador, respectivamente, para obtener una comunicación bidireccional.

Network Streams es un método de comunicación dinámica, estrechamente integrada y fácil de configurar para transferir datos de una aplicación a otra con características de rendimiento y latencia que son comparables con TCP. Sin embargo, a diferencia de TCP, Network Streams apoya directamente la transmisión de los tipos de datos arbitrarios sin necesidad de comprimir y descomprimir los datos en un tipo de datos intermedios. Network Streams convierte los datos de una manera compatible, que les permite a las aplicaciones que usan diferentes versiones de LabVIEWRuntimeEngine comunicarse entre sí con seguridad y éxito. Network Streams, también, ha mejorado la administración de conexiones que restaura automáticamente la conectividad de red si se produce una desconexión debido a un corte de red u otros fallos en el sistema. Streams utiliza una estrategia de comunicación basada en buffers y menor cantidad de pérdidas que asegura que los datos escritos en la secuencia nunca se pierde, incluso en entornos que tienen conectividad de red intermitente [9].

Debido a que las direcciones IP de los dispositivos conectados a la red son fijas, se facilita el intercambio de información punto a punto, en este caso entre la plataforma robótica y el computador por medio de Network Streams presente en LabVIEW.

Para configurar este protocolo, es necesario conocer con anterioridad la dirección IP del dispositivo y especificar qué función desempeñará en la comunicación (escribir o leer), adicionalmente, y con el fin de reducir tiempos y no redundar en información no deseada, se puede especificar la cantidad de bytes a leer o escribir. Paso seguido, 2 bloques de LabVIEW, llamados Read/Write Single ElementtoStream, se encargan de leer o escribir el dato o elemento en la comunicación, especificando un tiempo de espera máximo para el envío o su recepción. Finalmente, debe cerrarse la comunicación para liberar espacio en memoria. Los subprogramas de lectura y escritura implementados en LabVIEW se muestran en las figuras 10 y 11.

Figura 10. Configuración Network Stream (lectura)

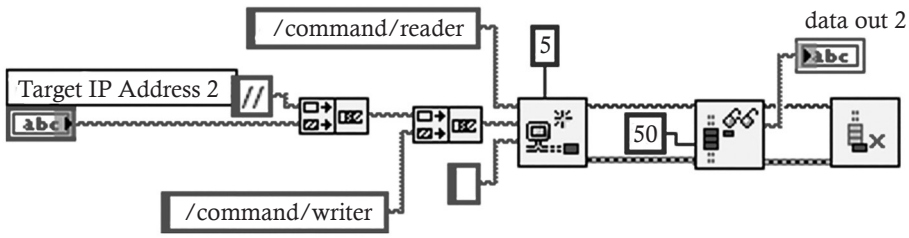
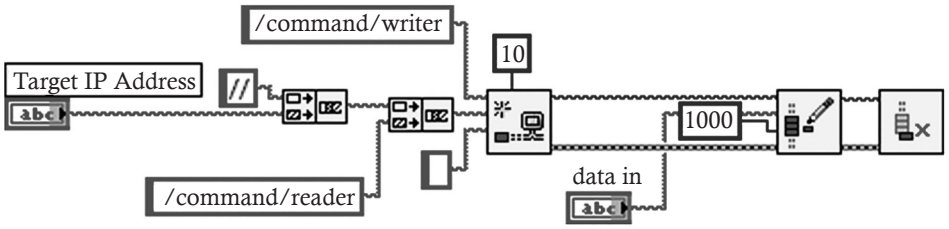


Figura 11. Configuración Network Stream (escritura)



De esta manera, se observa que previamente debe hacerse una conversión de los datos que quieran ser enviados a Strings, y una vez recibidos en el otro dispositivo, realizar el proceso inverso según los requerimientos del algoritmo por desarrollar. En caso de necesitar un ciclo continuo de envío-recepción de datos, debe añadirse un bucle tipo *while* que abarque los bloques correspondientes a lectura/escritura del protocolo Network Stream en cada uno de los dispositivos pertenecientes a la red.

Desarrollo y validación del modelo cinemático

Basados en la descripción anterior, para la plataforma robot DaNI 2.0 se desarrolló el modelado cinemático. La implementación de la cinemática de una plataforma móvil relaciona 2 problemas fundamentales: el primero consiste en poder determinar expresiones matemáticas que permitan conocer la posición y orientación final del móvil a partir de los parámetros físicos de la plataforma y el segundo, en poder determinar expresiones matemáticas que permitan conocer el comportamiento físico del móvil a partir de la posición y orientación final [12]. Estos 2 problemas son conocidos, dentro del estudio de la robótica, como la cinemática directa y la cinemática inversa. El movimiento de un móvil en el plano está dado por la ecuación 2.1. Este vector contiene la información correspondiente a la localización del móvil en un espacio de configuración 2D representado por el par ordenado y el ángulo de orientación de la plataforma (figura 12).

$$p^{\cdot} = \left[\dot{X}, \dot{Y}, \dot{\theta} \right]^{T} \tag{2.1}$$

Uno de los objetivos que se busca a la hora de modelar este tipo de estructuras es controlarlo en un marco local (2.2).

$$\dot{q}_a = J_{in} \dot{p} \tag{2.2}$$

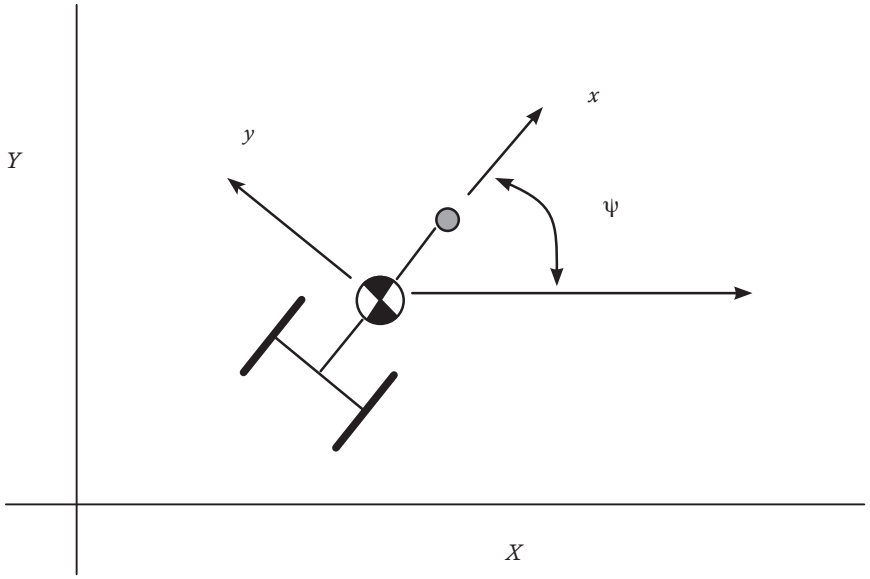
Donde

$\dot{q}_a$ = Velocidades de las ruedas

J_{in} = Es la solución de la cinemática inversa

$\dot{p}$ = Es el vector de la velocidad deseada

Figura 12. Descripción de una plataforma diferencial en el espacio



Fuente: [11].

Para el modelado de la plataforma se asumen varios factores, entre ellos la fricción mínima con el suelo y las pruebas realizadas en un ambiente ideal. Las velocidades en el marco de referencia local (fijo al cuerpo) se transforman en un marco global utilizando la matriz de la rotación ecuación (2.3), y la solución a la cinemática inversa se muestra en la ecuación (2.4) [13].

$$R(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$J_{in} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

El modelado se aplica a una plataforma diferencial con velocidades $w1$ y $w2$ de sus ruedas controlables en el marco local que incluyen directamente en el marco de referencia [14]. Las velocidades con respecto a un plano de referencia se muestran en la ecuación (2.5).

$$\dot{q}_a = \left[\dot{x}, \dot{y}, \dot{\theta}\right]^T \tag{2.5}$$

Donde

$$\begin{aligned} \dot{x}1 &= R_\omega \omega 1 \\ \dot{x}2 &= R_\omega \omega 2 \end{aligned} \tag{2.6}$$

$$\dot{x} = \frac{1}{2}\left(\dot{x}1 + \dot{x}2\right) \rightarrow \frac{1}{2} R_\omega \left(\omega 1 + \omega 2\right)$$

El movimiento lateral es limitado en la plataforma diferencial, por lo que se asegura que $y = 0$. La Velocidad de giro está compuesta por el movimiento de las 2 ruedas (2.7) [15]:

$$\dot{\theta} = \frac{R_\omega}{B}\left(\omega 1 - \omega 2\right) \tag{2.7}$$

Entonces, teniendo en cuenta ecuaciones (2.6) y (2.7), se obtiene la ecuación (2.8):

$$\dot{q}_a = \left[\frac{1}{2} R\omega\left(\omega 1 + \omega 2\right), 0, \frac{R_\omega}{B}\left(\omega 1 - \omega 2\right)\right]^T \tag{2.8}$$

Efectuando ajustes matemáticos para el vector *p* que representa las velocidades en el marco de referencia global (2.9).

$$\dot{p} = J_{in}\dot{q}_a \tag{2.9}$$

Entonces, al realizar las operaciones pertinentes entre la ecuación (2.4) y (2.9), se obtiene la ecuación (2.10).

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} \cos(\Psi) & -\sin(\Psi) & 0 \\ \sin(\Psi) & \cos(\Psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\Psi} \end{bmatrix}$$

$$\dot{p} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\Psi} \end{bmatrix} = \begin{bmatrix} \frac{R_\omega}{2}\cos(\Psi)(\omega 1 + \omega 2) \\ \frac{R_\omega}{2}\sin(\Psi)(\omega 1 + \omega 2) \\ \frac{R_\omega}{B}(\omega 1 + \omega 2) \end{bmatrix} \tag{2.10}$$

Donde la ecuación (2.10) representa el modelo de una plataforma robótica móvil tipo diferencial cuya entrada son las velocidades de los motores y como salida se obtienen las velocidades en el marco de referencia global, que después pueden ser integradas para obtener la localización del robot. Teniendo en cuenta el proceso de integración que tiene la plataforma DaNI y el *software* LabVIEW, se realizó el proceso de validación mediante la implementación de todas las expresiones matemáticas del modelo sobre este *software* que permite la sincronización entre el computador y el robot mediante enlace ethernet (figura 13). En las figuras 14, 15 y 16 se muestran las simulaciones del modelo obtenido.

Figura 13. Representación del modelo obtenido

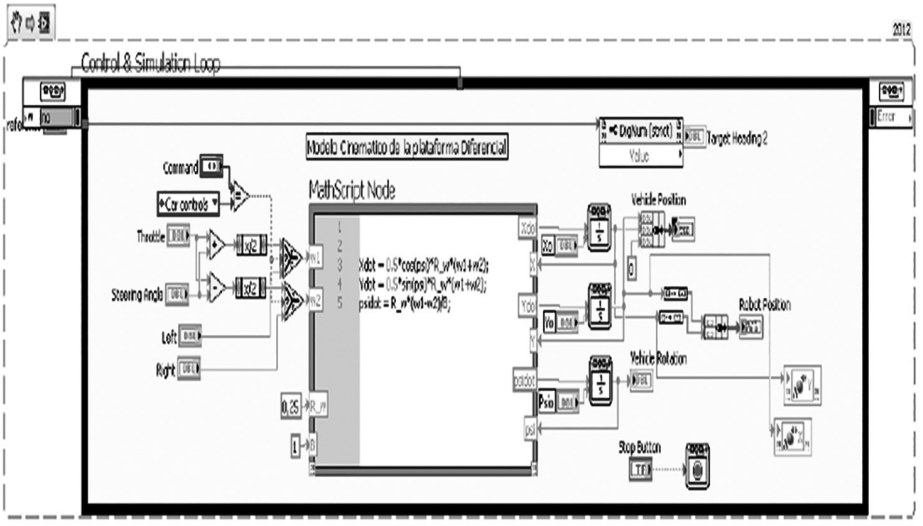


Figura 14. Comportamiento ideal en negro, comportamiento brindado por el modelo en gris, frente a una trayectoria circular

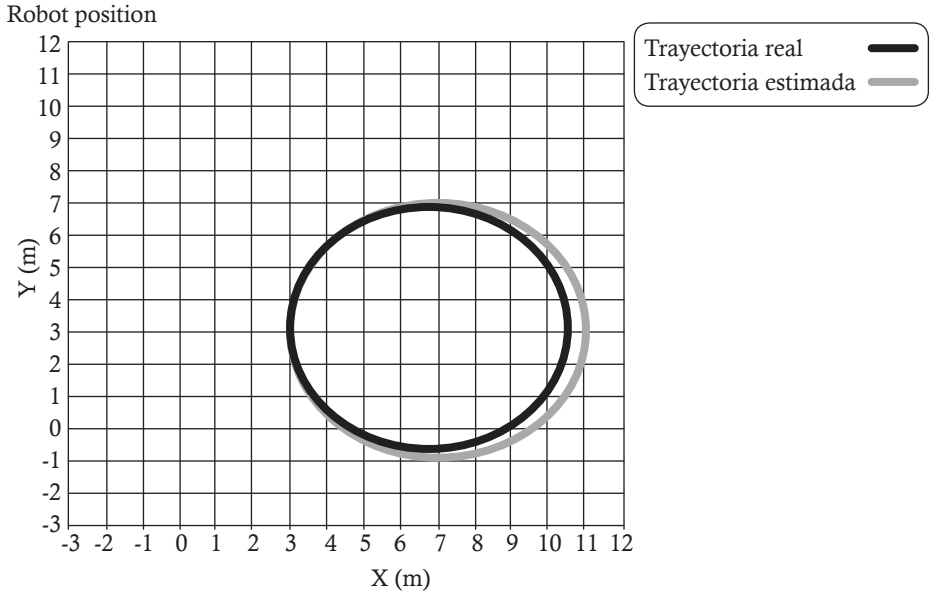


Figura 15. Comportamiento ideal en negro, comportamiento brindado por el modelo en gris, frente a unas trayectorias circulares

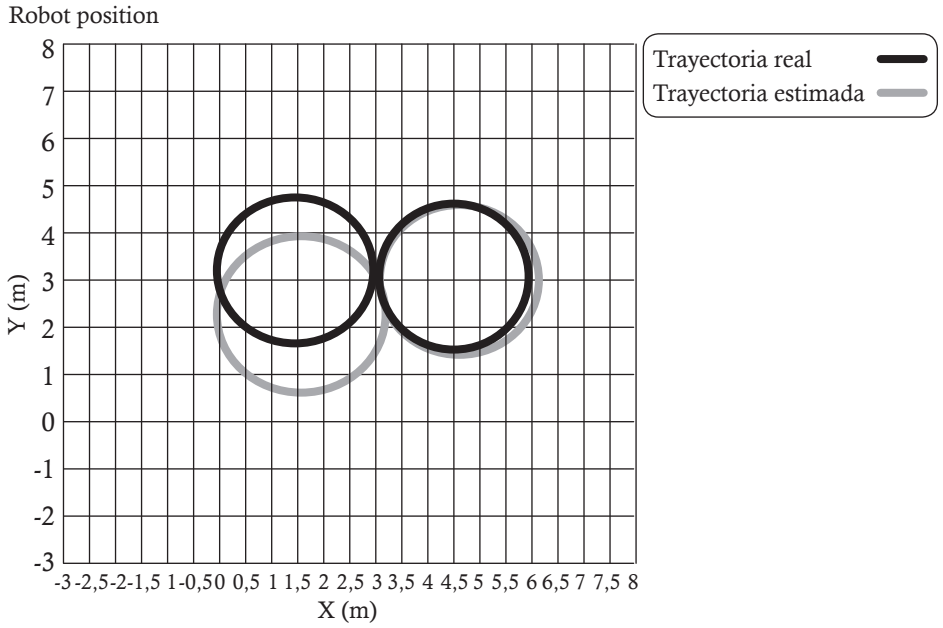


Figura 16. Se observa el comportamiento ideal en negro y la trayectoria estimada que tomó el modelo con un experimento de trayectoria aleatoria

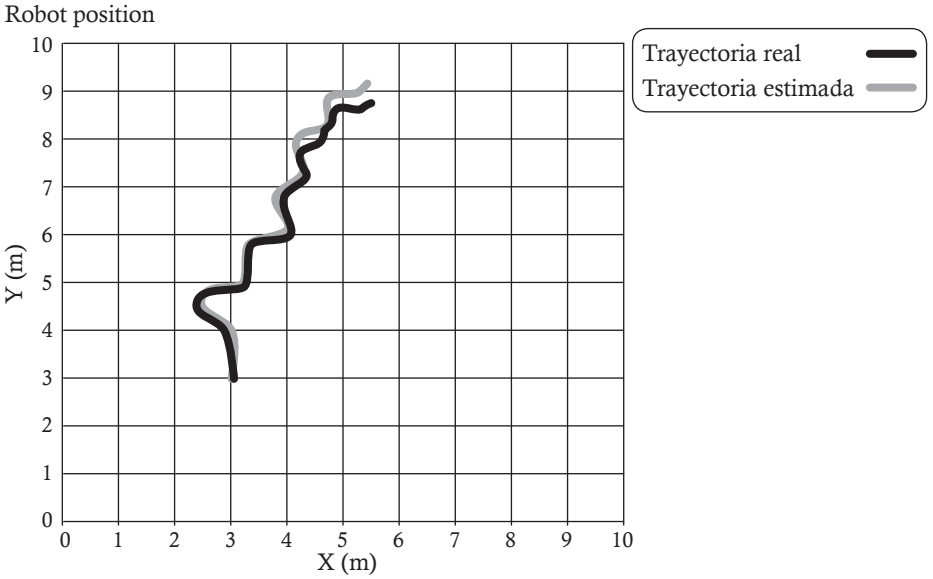
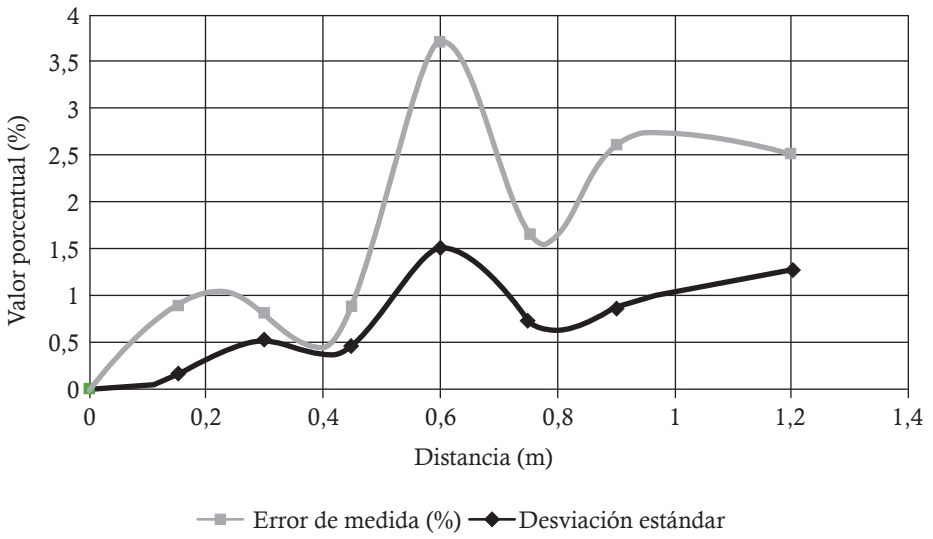


Figura 17. Error de medida y desviación estándar de los datos adquiridos del proceso de validación para el desplazamiento en línea recta

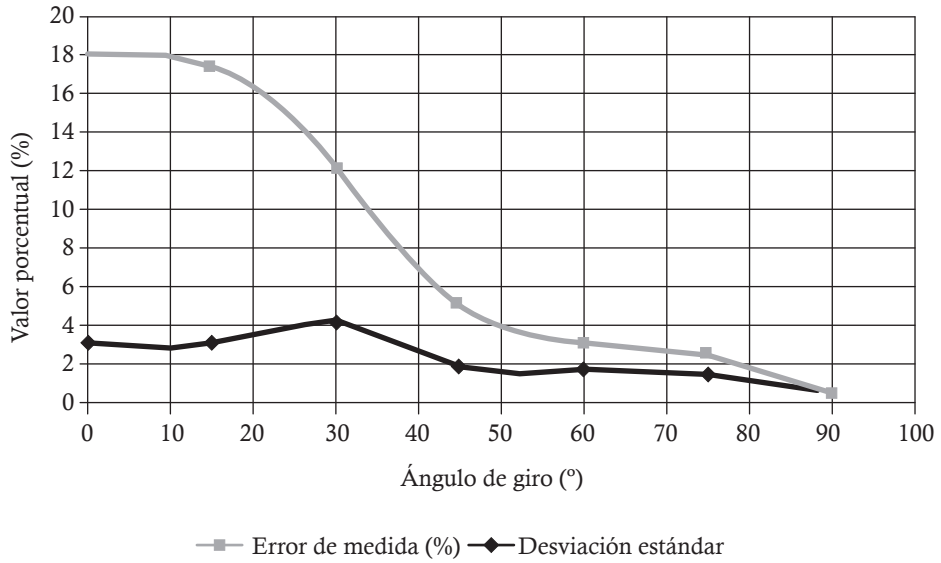


Del proceso de experimentación, validación y caracterización de la plataforma realizado para un desplazamiento lineal, se pudo determinar que el menor error sistemático se presentó con una velocidad igual a 0.2 m/s para desplazamiento no superiores

a 0.45 m. De la misma manera, se obtuvo que para desplazamientos de 0.15 m, la desviación estándar fue de ± 0.17%, con un error de medida de ± 0.93%, y para desplazamiento de 0.45 m, fue de ± 0.48% con ± 0.89%, respectivamente. Para la distancia seleccionada con el fin de realizar la mayoría de desplazamientos, se obtuvo una desviación estándar de ± 0.56% y un error de medida de ± 0.84% (figura 17). De igual manera, se llevó a cabo un proceso de experimentación para conocer el comportamiento del robot en desplazamientos rotacionales. De este, para giros muy pequeños se obtuvo la desviación estándar, que fue de ± 3.1% con un error de medida de ± 17.33%, y para giros de 90, fue de ± 0.51% con ± 0.44%, respectivamente (figura 18).

Para conocer la localización de la plataforma (figura 19), por medio de la observación y la integración consecutiva del movimiento de sus ruedas motoras en el espacio de configuración 2D, se realizó la integración de un modelo odométrico, entendido como un sistema que está permanentemente registrando la distancia recorrida por el vehículo con la finalidad de describir su evolución temporal respecto a la localización como una función de sus propias variables [9]. El modelo general de un sistema odométrico está dado por (2.11).

Figura 18. Error de medida y desviación estándar de los datos adquiridos del proceso de validación para movimientos de rotación



$$X(k+1) = f(X(k), U(k)) + v(k) \tag{2.11}$$

Donde $X(k+1)$ es la posición estimada, $X(k)$ es la posición actual, $U(k)$ es la entrada del sistema y $v(k)$ es el vector de errores sistemáticos y no sistemáticos que pueden ser asociados a la plataforma móvil.

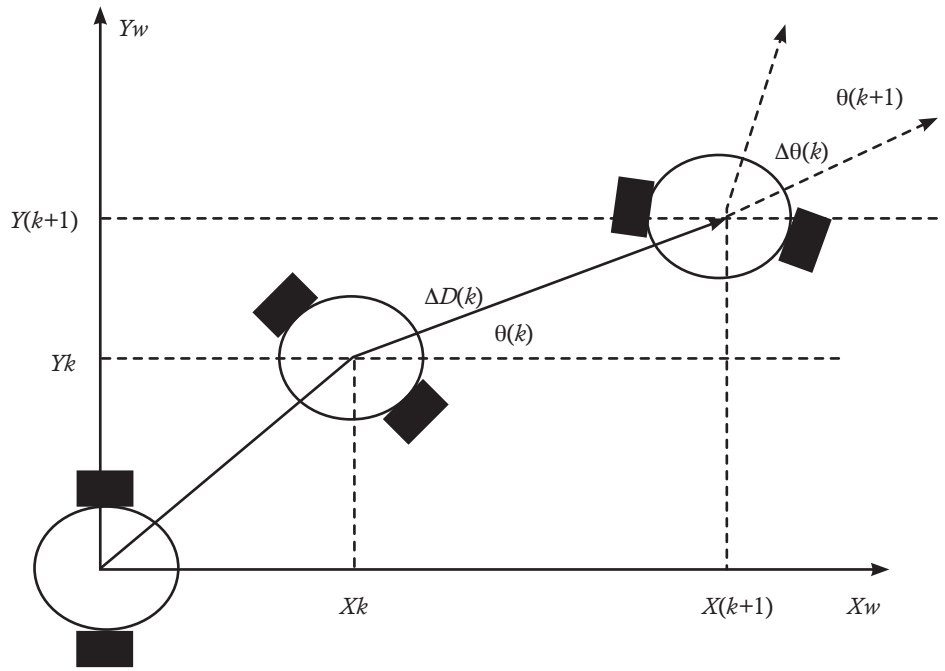
$$U(k) = [\Delta D(k)\ \Delta\theta(k)]^T \tag{2.12}$$

La entrada del sistema $U(k)$ se encuentra determinada por (2.11), donde $D(k)$ es la distancia recorrida por la plataforma en un intervalo (t_k, t_{k+1}) y $\theta(k)$ la variación de la orientación en el mismo intervalo (2.13). Para la caracterización del vector de errores de estado para una plataforma móvil $v(k)$, es asumido como $v(k) \approx N(0, Q(k))$, donde $Q(k)$ es el error característico de estado de la plataforma [16].

$$Q(k)\begin{pmatrix} Q_{11}(k) & 0 & 0 \\ 0 & Q_{22}(k) & 0 \\ 0 & 0 & Q_{33}(k) \end{pmatrix} \tag{2.13}$$

Con $Q_{11}(k) = KD(\Delta D(k) \cos \theta(k))$, $Q_{22}(k) = K_D(D(k) \sin \theta(k))$, y $Q_{43}(k) = K_{D\theta}(D(k) + K\theta(\theta(k))) \sin \theta(k))$, donde K_D es el coeficiente de error de traslación de la plataforma relativo a ΔD y expresado en [m2/m]; $K_{D\theta}$ es el coeficiente de error de rotación de la plataforma relativo a ΔD y expresado en [rad2/m]; y $K\theta$ es el coeficiente de error de rotación de la plataforma relativo a $\Delta\theta$ y expresado en [rad2/rad].

Figura 19. Iteraciones de la plataforma robótica respecto a un marco de referencia global (x, y)



Una vez desarrollado el modelo vectorial mostrado en (2.13), se realizó la validación usando los mismos procesos de experimentación presentados anteriormente (figura 19). Como zona de pruebas, fue utilizado el coliseo de la Facultad Tecnológica, lugar en donde se realizó la simulación de una zona colapsada.

Fusión sensorial

Este capítulo muestra todos los sensores que se implementaron en el sistema desarrollado; además, da a conocer la técnica que se utilizó para su lectura de información, llamada fusión sensorial. Las técnicas de fusión de datos surgen como una solución al problema de combinar de forma óptima múltiples fuentes de información redundantes; de esta manera, representan una alternativa de mejora a los sistemas basados en una sola fuente de información. Esta mejora viene dada por la redundancia inherente entre los sistemas que utilizan múltiples fuentes de información [17].

La fusión de datos es una materia multidisciplinar, en el sentido de que las técnicas y los métodos que se utilizan provienen de diversas disciplinas. Es posible realizar una clasificación de los diferentes métodos que realizan fusión de datos atendiendo a los siguientes criterios:

- Dependiendo de las relaciones existentes entre los datos de entrada. Estas pueden consistir en datos complementarios, redundantes o cooperativos.
- Con base en los tipos de datos de entrada y salida utilizados en el proceso de fusión de datos.
- Utilizando el nivel de abstracción de los datos manipulados durante el proceso de fusión, es decir, si se utilizan medidas, señales, características o decisiones.
- Basándose en el criterio de los niveles de fusión definidos por el Joint Directors of Laboratories (JDL).
- Atendiendo a los distintos tipos de arquitecturas posibles.

Sensor de calor TPA81

En la figura 20 se muestra el sensor TPA81 que consiste en un arreglo de 8 "Thermopile", que detectan ondas infrarrojas en el rango de 2 a 22 µm, que es la longitud de onda del calor radiante; por lo tanto, puede ser usado en aplicaciones de termómetros infrarrojos sin contacto. Puede medir la temperatura de 8 puntos adyacentes simultáneamente, además puede controlar un servo que mueve el módulo para obtener una imagen térmica. El TPA81 puede detectar la llama de una vela a una distancia de 2 m y sus mediciones no se afectan con la luz ambiente.

Figura 20. Sensor de calor TPA81

Fuente: [18].

En la tabla 3 se muestran las principales características del sensor.

Tabla 3. Características sensor TPA81

Característica	Especificación
Voltaje de operación	5V
Corriente	5Ma
Comunicación	I2C
Rango de temperatura	4 °C – 100 °C
Dimensiones	31 mm x 18 mm

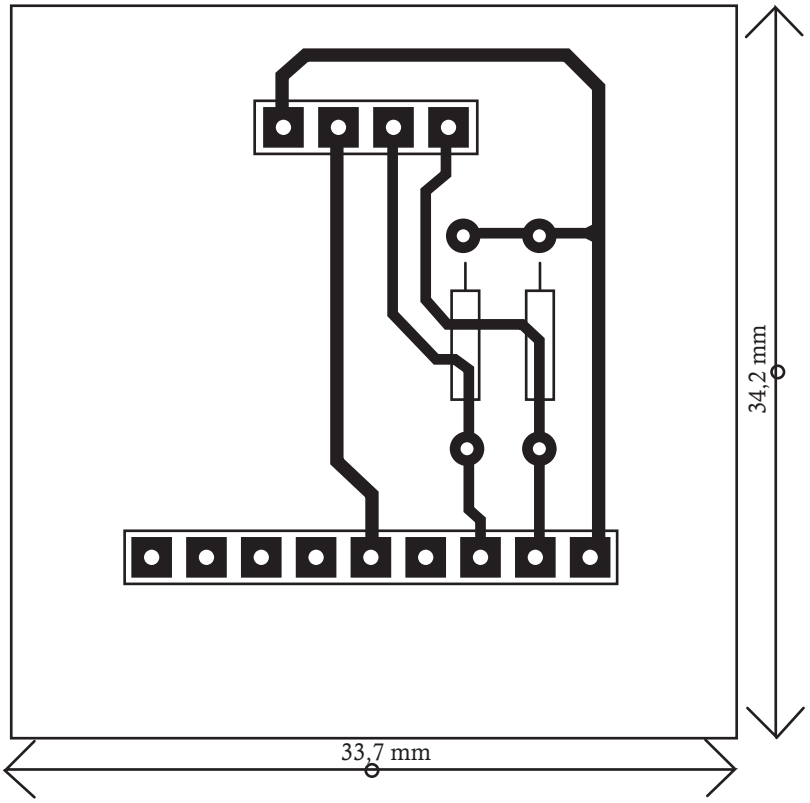
Fuente: [18].

El campo de visión normal del sensor es de 41° por 6°, lo que convierte cada uno de los 8 píxeles en 5.12° por 6°. El conjunto de 8 píxeles está orientado a lo largo de la placa de circuito impreso.

El sensor TPA81 utiliza una conexión I2C. Tiene 5 pines que se distribuyen así: el pin 1 es la tierra, el 2 no se conecta, los pines 3 y 4 corresponden a las líneas SCL y SDA con las resistencias de *pull up* conectadas a la alimentación con el fin de asegurar la comunicación en el bus. Finalmente, el pin 5 es la alimentación.

La conexión del sensor a la tarjeta SBRIO 9632 se hizo por medio de 2 pines digitales de esta (7IO2 y 7IO3), haciendo uso de la herramienta I2C de LabVIEW. La alimentación del sensor es tomada de la batería de la plataforma robótica. En la figura 21 se muestra el circuito impreso realizado en ARES para la adaptación a la tarjeta.

Figura 21. Circuito impreso del sensor TPA81



Para el proceso de caracterización de este sensor se llevaron a cabo experimentos enfocados a obtener la respuesta de este en la mínima y en la máxima distancia de detección. La fuente de calor fue una bombilla de 100 W, conectada a la red domiciliaria (120 V, 60 Hz), que simulara ser un cuerpo humano. En la figura 22 se muestran los valores arrojados por el sensor ante la presencia de la fuente de calor puesta a diferentes distancias. El proceso de experimentación tuvo como parámetros la temperatura ambiente (25 °C) y una fuente de calor de 100 W; en la figura, el eje Y presenta la distancia en metros en la que se detectó la fuente de calor, y en el eje X se tiene la temperatura en grados celsius.

En la figura, se puede observar que se obtuvieron valores de temperatura por encima de los 100 °C, que es el valor límite entregado por el fabricante, quien aclara que las medidas por encima de este valor podrían no ser confiables.

El sensor TPA81 también entrega una lectura de la temperatura ambiente del entorno; este valor junto con el que entrega el sensor SHT71 (ver sección "Sensor SHT71") serán promediados para el sistema de fusión sensorial.

Figura 22. Ángulo de detección del sensor TPA81

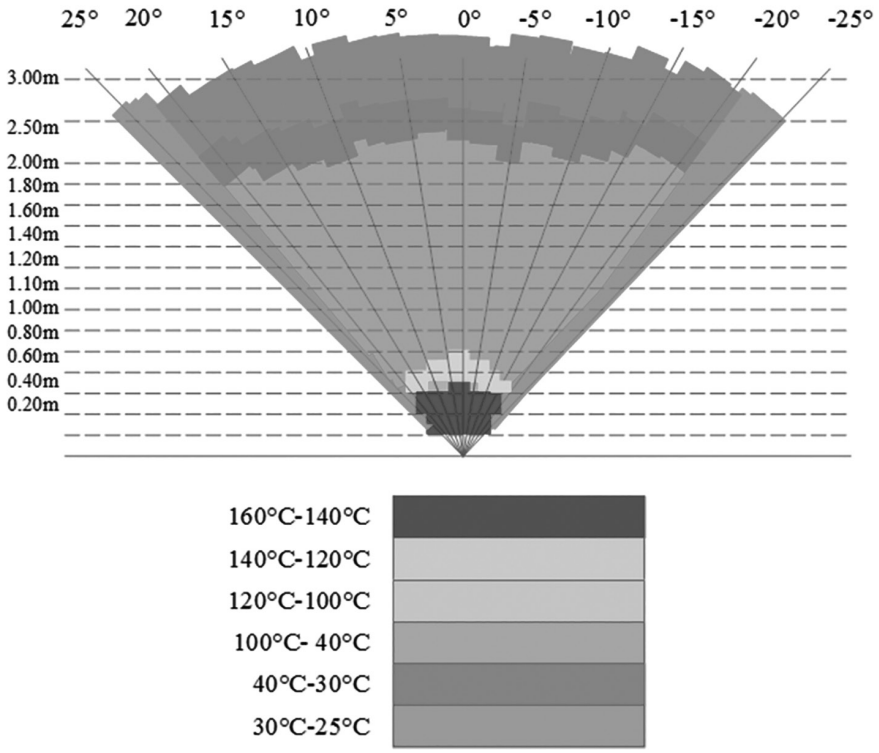
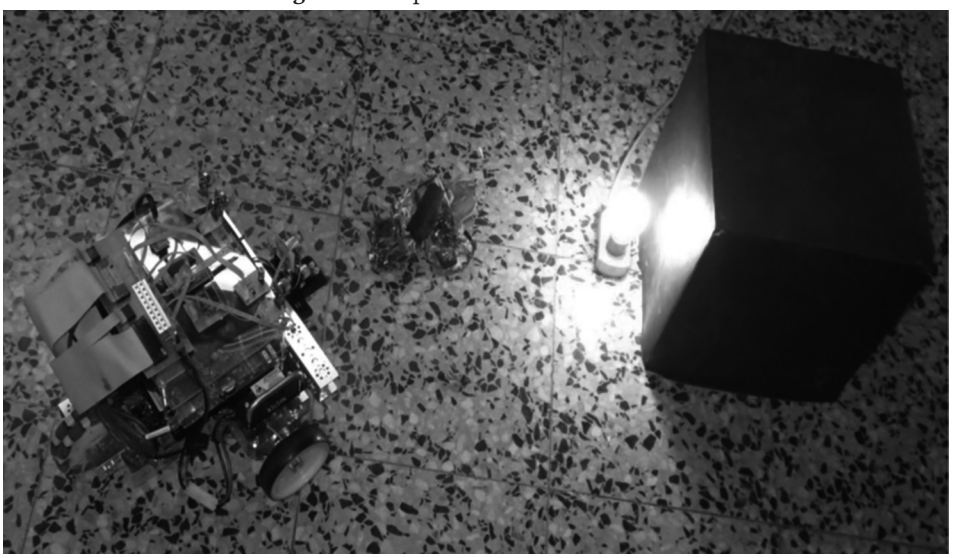


Figura 23. Exposición a fuentes de calor



Sensor para el monitoreo de la humedad relativa y la temperatura SHT71

La humedad relativa en el medio ambiente es una cantidad numérica que indica el cociente entre en la humedad absoluta del medio ambiente y la cantidad máxima de vapor de agua que admite el aire por unidad de volumen, esta cantidad se mide en porcentaje de humedad relativa (%) y su valor está normalizado de forma que la humedad relativa máxima posible del medio ambiente es del 100 %; la determinación del valor de la humedad relativa del medio ambiente está muy ligada a la temperatura del medio ambiente en el momento de la medición, por lo que es común considerar y realizar la medición de ambas variables al mismo tiempo [19].

Se deben tener en cuenta las mediciones de la humedad relativa, ya que es determinante para predecir un incendio o una explosión; dependiendo de las mediciones de temperatura y la detección de gases inflamables.

El sensor implementado es el SHT71, calibrado en fábrica y con salida digital, fabricado por Sensirion (figura 24). Es un dispositivo que posee en su interior un sensor de temperatura para compensar la medición de humedad con respecto a esta. Cuenta, también, con un calefactor interno que evita la condensación en el interior de la cápsula de medición en condiciones de niebla o cuando existe condensación. La comunicación se establece por medio de un bus serie sincrónico [21].

Figura 24. Sensor de humedad relativa y temperatura SHT71

Fuente: [20].

En la tabla 4 se muestran las principales características del sensor.

Tabla 4. Características sensor SHT71

Características	Especificaciones
Voltaje de operación	2.2 V a 5.5 V
Corriente	1 Ma
Comunicación	Protocolo propio
Temperatura de trabajo	-40 °C- 125 °C
Campo de medición	0 % - 100 % RH

Fuente: [22].

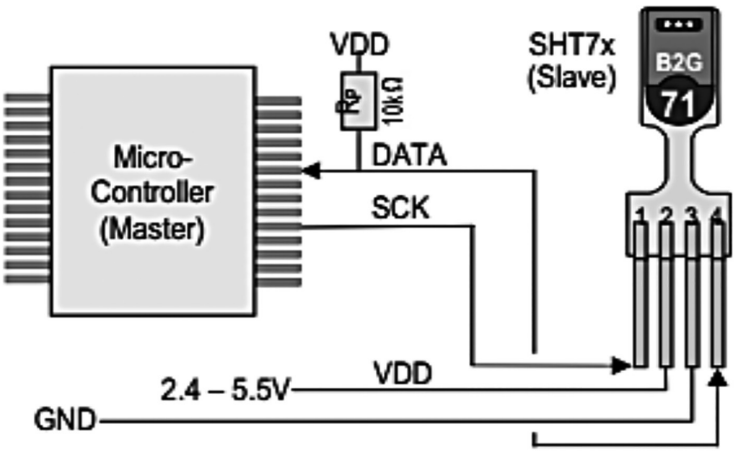
La distribución de los pines del sensor es así: pin 1 SCK que es la línea de reloj, pin 2 es la alimentación, el pin 3 es la tierra y el pin 4 es DATA, de donde se toma la lectura; el sensor entrega una lectura digital de humedad y temperatura, por medio de un protocolo de comunicación propio [21].

Debido a que la comunicación no es con los protocolos estándar, sino con uno propio, la lectura del sensor se hizo por medio del microcontrolador PIC16F873a de microchip Technology, fabricado con tecnología CMOS; su consumo de potencia es muy bajo y, además, es completamente estático, esto quiere decir que el reloj puede detenerse y los datos de la memoria no se pierden [22].

Conexión

Se obtuvo la librería en PICC para el manejo del sensor y se desarrolló un programa que leyera y transmitiera los valores por RS232 a la tarjeta SBRIO 9632; en esta, la comunicación se realizó por 2 pines digitales (7IO0-Tx y 7IO1-Rx), haciendo uso de la herramienta serie FPGA de LabVIEW. En la figura 25 se muestra el diagrama de conexión entre el sensor SHT71 y el microcontrolador PIC16F873a.

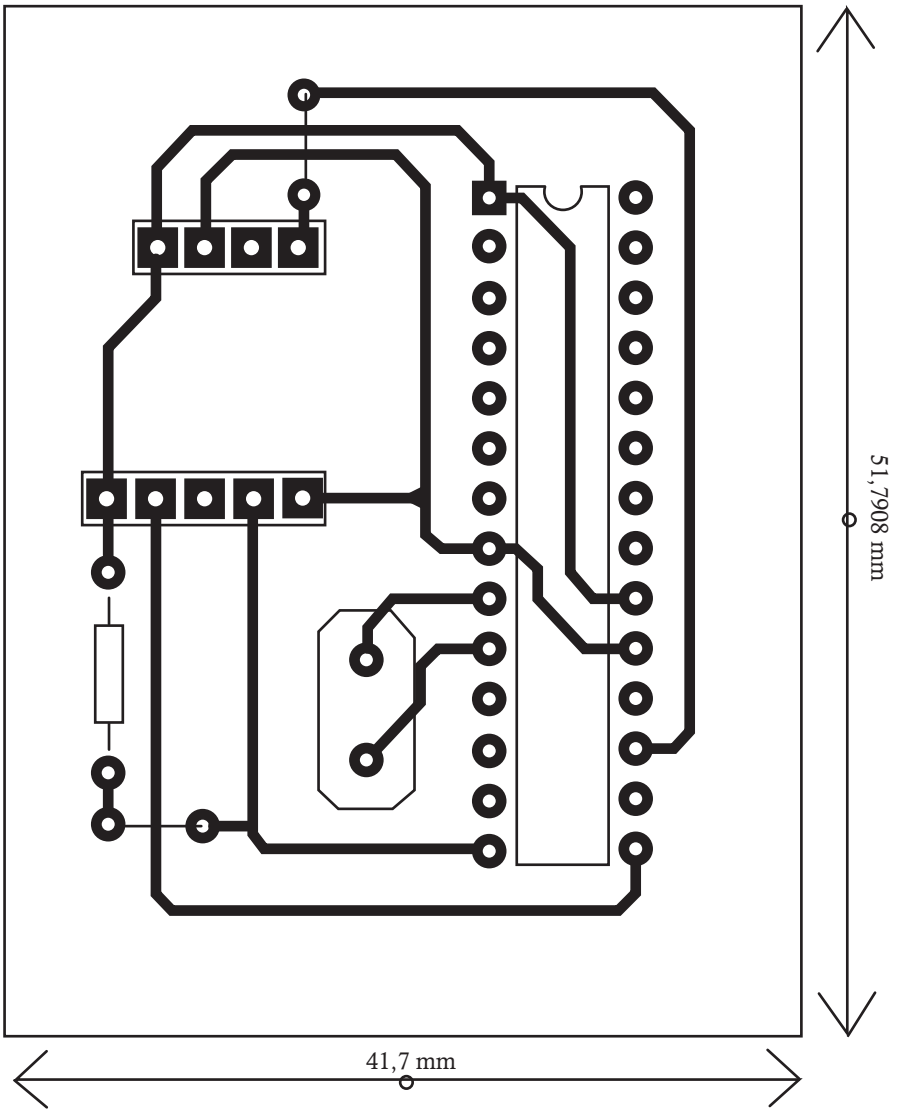
Figura 25. Conexión entre el sensor SHT71 y el microcontrolador PIC16F873a



Fuente: [21].

En la figura 26 se muestra el circuito impreso para el sensor, donde se tiene la conexión con el microcontrolador y las salidas al dispositivo embebido.

Figura 26. Circuito impreso para el sensor



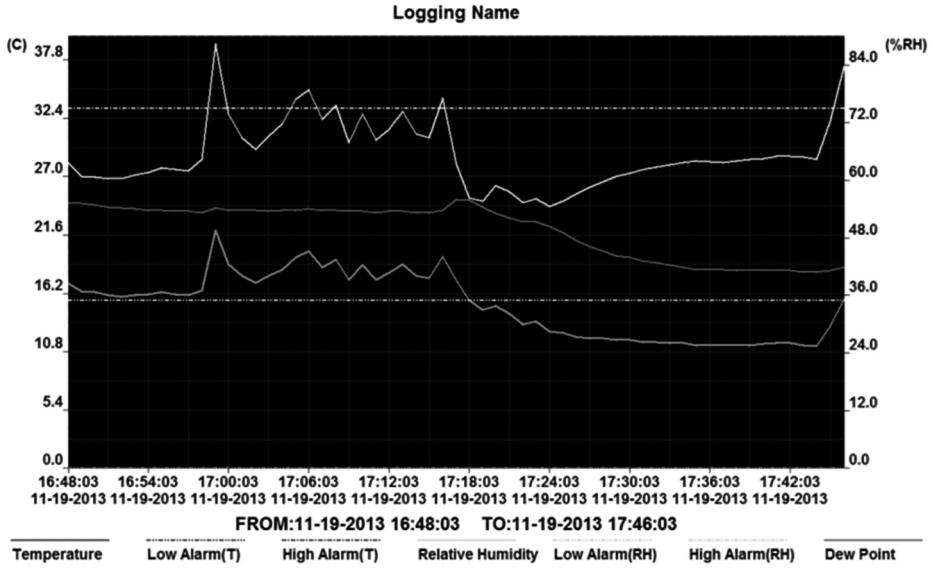
Pruebas

La caracterización del sensor se hizo tomando como patrón el datalogger RHT10 de EXTECH Instruments, que se muestra en la figura 27, el cual tiene en su interior un sensor SHT11. El instrumento permite configurar la tasa de registro, la alarma alta/baja y el modo de inicio; descargar los datos guardados conectando el módulo al puerto USB del computador, y ejecutar el *software* suministrado. Los datos de humedad relativa, temperatura y punto de rocío pueden imprimirse, graficarse y exportarse a otras aplicaciones [22].

Figura 27. Datalogger RHT10

Para tomar las medidas, se alteraron la humedad y la temperatura de un cuarto cerrado, colocando un recipiente con agua hirviendo, el cual dejaba salir el vapor en la habitación. Esta prueba tuvo una duración de una hora. En la figura 28 se muestra el gráfico entregado por el RHT10.

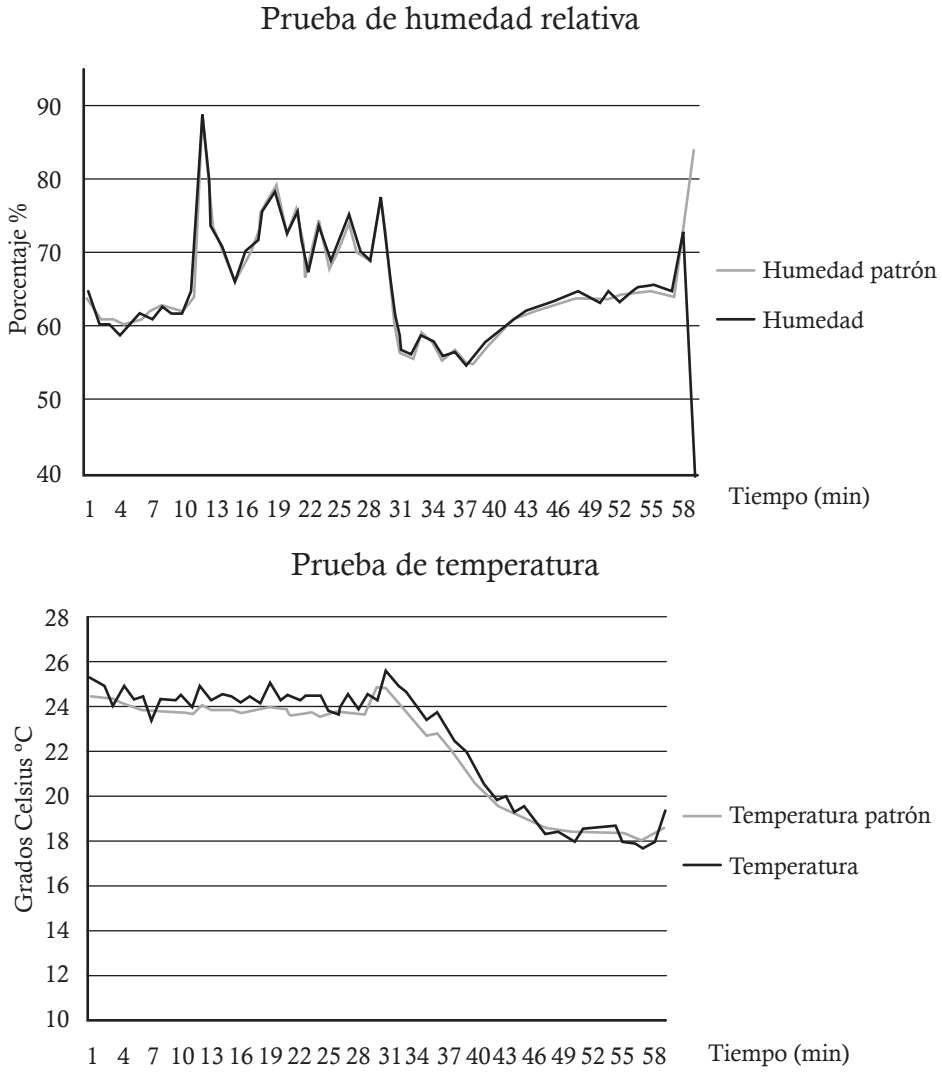
Figura 28. Gráfico entregado por el RHT10



En la gráfica entregada por el RHT10 se observan la línea superior y la línea del centro , que corresponden a las lecturas de humedad y temperatura respectivamente, la figura 29 muestra las gráficas comparativas, la línea correspondiente a la humedad y la temperatura, fue medida con el sensor SHT71, y la que representa la humedad y la temperatura fue entregada por el instrumento RHT10.

Se observa que los datos son muy similares; esto debido a que en el interior del medidor de humedad y temperatura RHT10, hay un sensor SHT11, que pertenece a la misma familia del SHT71 y es de montaje superficial y alta precisión.

Figura 29. Superior: prueba de humedad. Inferior: prueba de temperatura



Adicionalmente, se tomaron medidas con el sensor y el datalogger RHT10 en algunos puntos de la escala para hallar el error relativo y el error absoluto, y se hizo un barrido por la escala del sensor para encontrar su curva de histéresis.

Error absoluto y error relativo

La diferencia ente el valor real y el valor medido por el sensor es el error absoluto; el error relativo es el cociente entre el error absoluto y el valor verdadero y representa la fracción de imprecisión cometida en la medición. Lo que es útil para comparar mediciones llevadas sobre diferentes magnitudes. Estos valores se pueden dar en porcentaje o en la unidad que entrega el sensor [23].

En la tabla 5 se muestran 14 medidas de humedad y temperatura tomadas con el sensor. Estas fueron tomadas cada 30 s; el valor entregado por el datalogger RHT10 fue de 21.6 °C y 58.6 %.

Tabla 5. Meq10didas del sensor SHT71 en un punto de la escala

Medida	Temperatura (ºC)	Humedad (%)
1	21.63	59.656
2	21.001	58.7
3	21.945	59.49
4	21.798	59.234
5	21.693	58.732
6	21.112	58.723
7	21.564	58.821
8	21.379	58.082
9	21.127	58.256
10	21.667	58.108
11	21.765	58.0
12	21.611	57.935
13	21.609	56.423
14	21.543	56.45

Para hallar el valor del sensor en ese punto, se toma el promedio de las medidas así:

$$Humedad = \frac{816.61}{14} = 58.329\%$$

$$Temperatura = \frac{301.444}{14} = 21.531\ ^{\circ}C$$

Con los valores obtenidos, se halla el error absoluto (Ea) y el error relativo (Er), expresados en las unidades de humedad y temperatura, así:

$$E_a(hum) = 58{,}6\% - 58.329\% = 0.271\%$$

$$E_a(tem) = 21.6\ ^{\circ}C - 21.531\ ^{\circ}C = 0.271\%$$

$$E_r(hum)\frac{0.271}{58{,}6} * 100\% = 0.46\%$$

$$E_r(tem)\frac{0.069}{21{,}6} * 100\% = 0.319\%$$

Histéresis

Histéresis es la diferencia entre los valores entregados por el sensor para un valor cualquiera, cuando la variable recorre toda su escala en forma ascendente y descendente. Se expresa en porcentaje de alcance [24]. Para el cálculo de la histéresis se tomó un valor de la escala, se registró el dato ascendente y descendente, y la diferencia entre estos se divide entre la escala, así:

$$Temperatura\ 23\ ºC$$

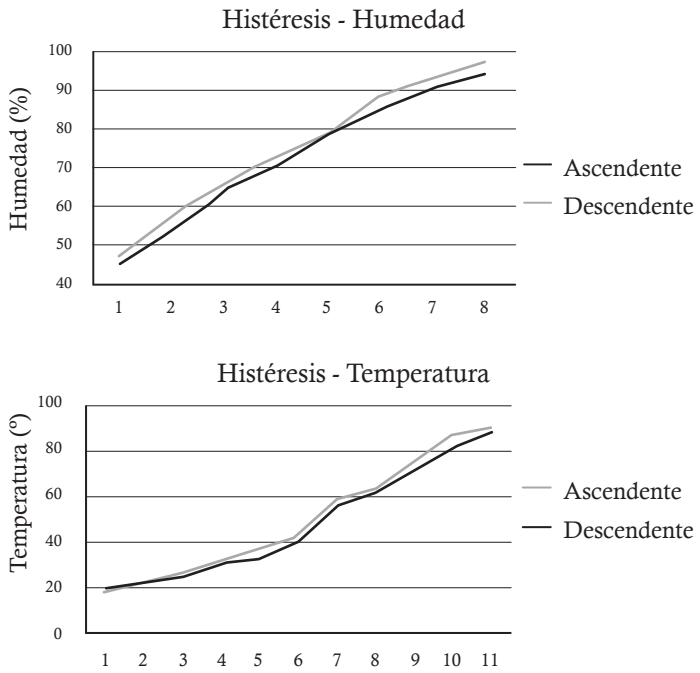
$$Histerésis(tem) = \frac{21-21}{125-(-40)} * 100\% = 2.42\%$$

$$Humedad\ 62\%$$

$$Histerésis(hum) = \frac{265-63}{100-0} * 100\% = 2\%$$

En la figura 30 se muestra el resultado del experimento de llevar el sensor desde el inicio de la escala hasta el final y posteriormente del final al inicio de esta para hallar la histéresis del sensor. Para la temperatura no se toman los valores desde el fondo de la escala, se toman desde 15 °C y hasta 100 °C; para la humedad relativa, se toman desde 40 % hasta el final de la escala.

Figura 30. Superior: histéresis para la humedad. Inferior: histéresis para la temperatura



Sensores para la detección de gases tóxicos o inflamables

El monitoreo de gases peligrosos para la calidad del aire es una tarea compleja, ya que dependiendo el gas que se mida, la sensibilidad del sensor debe cambiar; por ejemplo, si se requiere la medición de gases tóxicos, se necesitan sensores con sensibilidad a concentraciones bajas, y si se requiere la medición de un gas combustible, el sensor debe detectar altas concentraciones de este.

Debido a que hay cientos de gases y dependiendo la fuente, el sensor que se debe utilizar para su medición es distinto; cada tipo de sensor está basado en un principio de detección único y, por ende, tiene características de respuesta al gas únicas. La mayoría de los sensores no son específicos a un gas y son sensitivos a un grupo o familia de gases [25].

Sensor de gas TGS3870

Sensor de gas de semiconductor del óxido de metal para la detección de metano y de monóxido de carbono fabricado por Figaro (figura 31). Usa un gas del micro-bead que detecta la estructura; el metano y el monóxido de carbono se pueden detectar con un solo elemento del sensor por el uso periódico de 2 diversos voltajes del calentador (cielo y tierra). La miniaturización del gas que detecta el grano da lugar a un consumo de energía del calentador solamente de 38 MW (promedio) [26].

TGS3870 tiene sensibilidad baja a los vapores del alcohol (un gas típico de interferencia en el ambiente residencial) y tiene alta durabilidad, es ideal para las alarmas de gas del mercado de consumidores.

Figura 31. Sensor TGS3870

Fuente: [26].

En la tabla 6 se muestran las características del sensor.

Tabla 6. Características sensor TGS3870

Característica	Especificación
Voltaje de operación	5V VHH: 0.9V VHL: 0.2V
Consumo	38mW
Comunicación	Protocolo propio
Gases	Metano y monóxido de carbono
Rango de detección	Metano: 500 a 12500 ppm Monóxido de carbono: 50 a 1000 ppm
Temperatura de operación	20 a 22 °C
Humedad relativa	60 a 65%

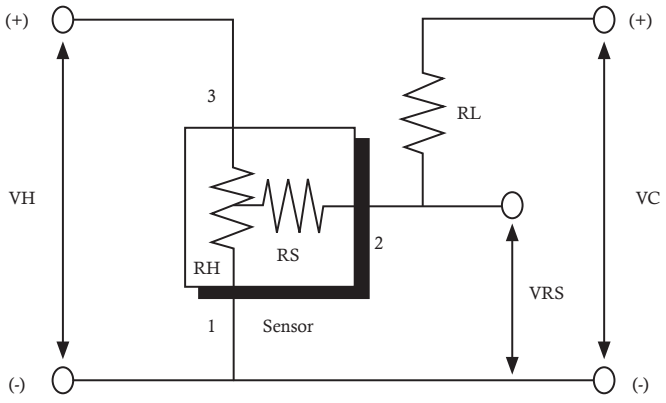
Fuente: [26].

Como el sensor mide simultáneamente los 2 gases, se hicieron pruebas con gas natural para la obtención del metano y con el humo proveniente del exosto de un automóvil para el monóxido de carbono, y se ajustó la sensibilidad del sensor con una resistencia de carga de 7 KΩ, con la cual se alejan las mediciones de cada uno de los gases [27-28].

El sensor TGS3870 maneja 2 voltajes que deben ser proporcionados al pin VH en tiempos distintos para obtener una medición; durante 5 s se debe aplicar un voltaje de 0.9 V y durante 15 s, un voltaje de 0.2 V; posteriormente, en otro pin VC, se debe aplicar la alimentación una vez hayan transcurrido los 20 s. Es decir, el sensor entrega una lectura cada 20 s.

El sensor cuenta con 3 pines de conexión, el pin 1 es la tierra, el pin 2 es VH (tensión principal) y el pin 3 es la alimentación del sensor VC. En la figura 32 se muestra el circuito básico de conexión del sensor, tomando en cuenta la resistencia de carga.

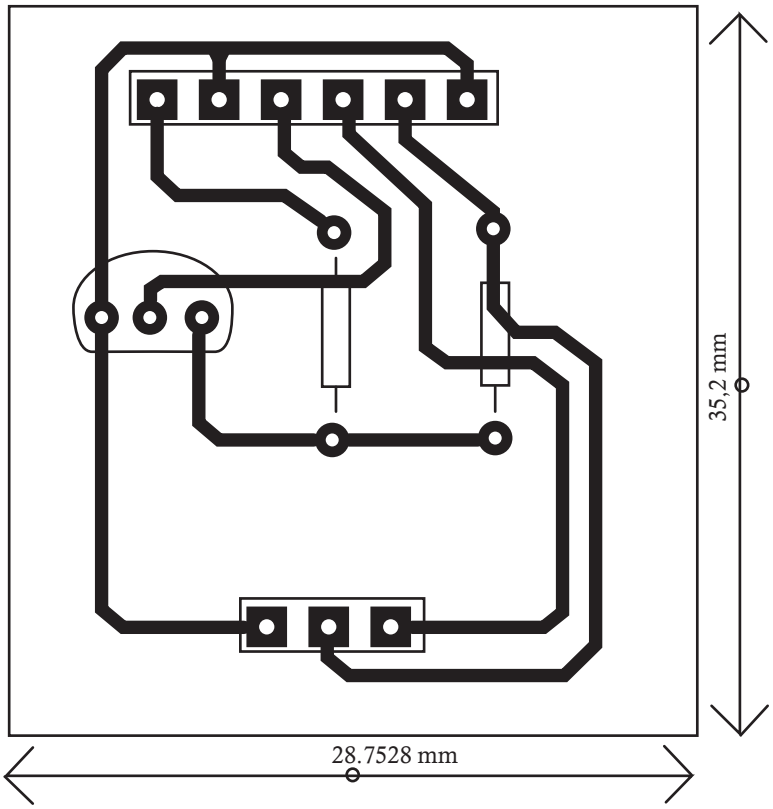
Figura 32. Circuito básico del sensor TGS3870



Fuente: [26].

Para la conexión del sensor con la tarjeta SBRIO 9632, se usaron sus canales análogos y digitales, así: AI17 y AO0 para el suministro del volteje a VH y la toma de datos, respectivamente, y 4IO8 para la alimentación del circuito. En la figura 33 se muestra el circuito impreso para el sensor.

Figura 33. Circuito impreso del sensor TGS3870



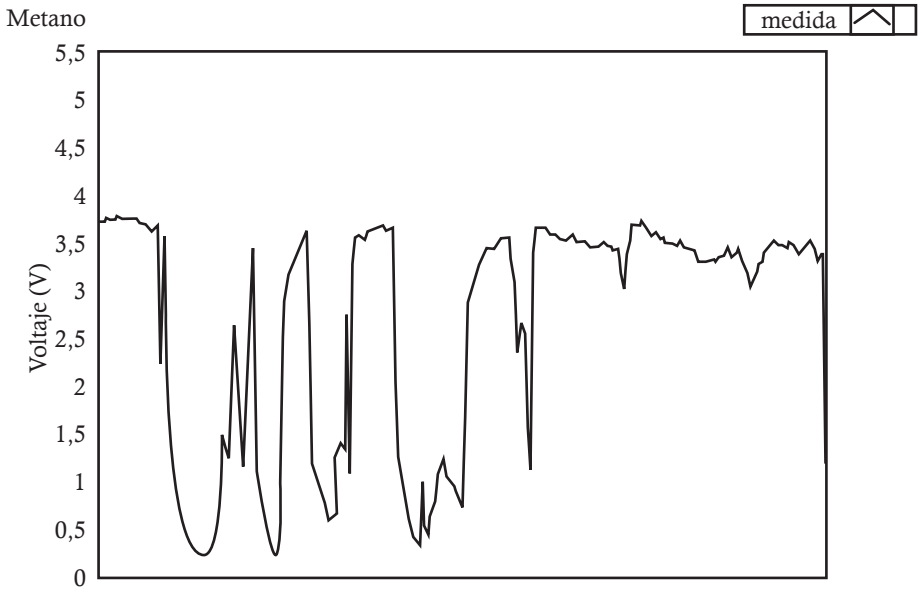
Para la realización de pruebas con el sensor no se hizo uso de un patrón, debido a que no se tuvo un ambiente controlado en el cual se pudiera modificar la concentración de metano y monóxido de carbono ni tampoco un instrumento que midiera dichas concentraciones. Por esta razón, se hicieron mediciones de fuentes que produjeran dichos gases.

El primer gas que se probó fue el metano (CH4). Este es un hidrocarburo y el principal componente del gas natural. Es un potente "gas de efecto invernadero", y es el mayor responsable del cambio climático a corto plazo (es decir, 10 a 15 años). El CH4 es emitido durante la producción y el transporte de carbón, gas natural y petróleo. Las emisiones de gases también resultan de la ganadería y otras prácticas agrícolas y de la descomposición del desperdicio orgánico en los vertederos de desechos sólidos municipales y de ciertos sistemas de tratamiento de aguas de desecho [29].

También se utiliza en la elaboración de muchas sustancias químicas, como acetileno y metanol [30].

La prueba de CH4 para el sensor TGS3870 se hizo con el gas natural domiciliario, ya que el 81.86 % de este es gas metano [31]. El sensor fue expuesto durante 2 minutos al gas natural proveniente de la manguera que lo proporciona a los gasodomésticos del hogar; esta manguera tiene un diámetro de 2.5 cm. En la figura 34 se muestra la respuesta del sensor; obtenida en LabVIEW, de la cual se infirió que una lectura por encima de los 3.2 V corresponderá a CH4.

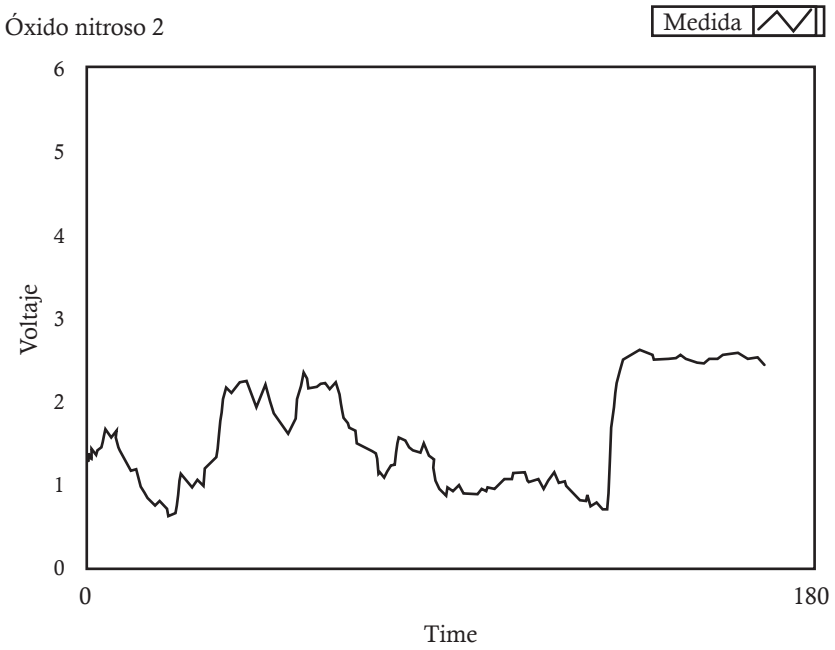
Figura 34. Respuesta del sensor TGS3870 al gas natural domiciliario



Otro gas que se probó fue el monóxido de carbono. A nivel doméstico, la producción de monóxido de carbono se debe a la quema de gas, carbón, leña, kerosén, alcohol, y cualquier otro combustible; entre los artefactos que queman gas están las estufas infrarrojas, los braseros, la leña y las parrillas y los automóviles; en la industria, la producción total de CO se da por los trabajos en metalurgia, minería y mecánica; otra fuente de CO es el tabaco, cuyo humo contiene aproximadamente 400 ppm. Finalmente, los aerosoles domésticos e industriales y quitamanchas que contienen cloruro de metileno [32].

Para esta prueba se dejó expuesto el sensor al CO proveniente del exosto de un vehículo Fiat Uno SCR modelo 94, que presentó un porcentaje del 5 % de CO en la revisión técnico mecánica del 2014 para una mezcla rica en hidrocarburos. El diámetro del tubo de escape es de 2 in. Se tomaron los valores durante 180 s para establecer el rango de voltaje en el que el sensor detecta CO, los datos se graficaron en LabVIEW. En la figura 35 se presenta la gráfica de la medición obtenida.

Figura 35. Respuesta del sensor TGS3870 al humo de un automóvil

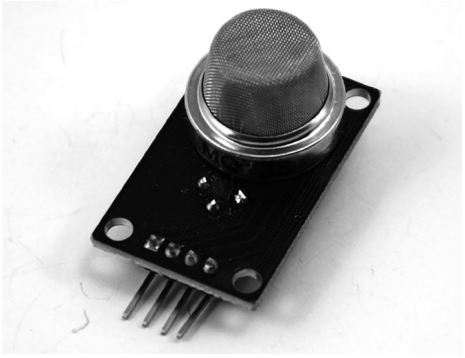


Sensor de gas MQ135

Sensor semiconductor para el control de la calidad del aire, fabricado por FUTURLEC (figura 36). Está hecho a base de dióxido de estaño (SnO_2), que presenta una continuidad mínima al aire limpio; cuando se presenta una concentración de gas combustible, la conductividad aumenta.

El MQ135 tiene una alta sensibilidad al amoníaco, al óxido nitroso o nítrico y al vapor de benceno, también al humo. Es de bajo costo y adecuado para aplicaciones donde se requiera detectar polución del aire [33].

Figura 36. Sensor MQ135



Fuente: [34].

En la tabla 7 se presentan las principales características del sensor.

Tabla 7. Características Sensor MQ135

Características	Especificaciones
Voltaje de operación	5V
Consumo	100m W
Gases	Amoniaco Benceno Óxido nitroso
Rangos de detección	Amoniaco: 10 a 300 ppm Benceno: 10 a 300 ppm Óxido nitroso: 10 a 300 ppm
Temperatura de operación	20 °C a 22 °C
Humedad relativa	60 % a 65 %

Fuente: [34].

El sensor MQ135 tiene 4 pines de conexión; el pin 1 es la alimentación, el pin 2 es la tierra, el pin 3 es la salida análoga del sensor, entrega un voltaje entre 0 y 5 V, dependiendo de la concentración de los distintos gases; y el pin 4 es la salida digital, entrega un 1 o un 0, dependiendo de la concentración del gas; esto se puede variar dependiendo de la resistencia de carga que se tenga, el sensor posee una resistencia variable que se ajusta dependiendo el gas que se vaya a detectar. La conexión con la tarjeta SBRIO 9632 se hizo por 3 canales análogos, para amoniaco, benceno y óxido nitroso; 4IO5, 4IO6 y 4IO7, respectivamente. En la figura 37 se muestra el circuito impreso para los 3 sensores de gas.

Este sensor se sometió a varias pruebas, la primera es con el amoniaco, que es una sustancia química producida por los seres humanos y la naturaleza. Consiste de una parte de nitrógeno (N) y tres partes de hidrógeno (H3). Cuando se encuentra amoniaco en niveles que pueden causar preocupación, estos probablemente se deben a una producción directa o indirecta por seres humanos.

El amoniaco es un gas incoloro de olor muy penetrante. Esta forma del amoniaco se conoce también como amoniaco gaseoso o amoniaco anhidro ("sin agua"). El amoniaco gaseoso puede ser comprimido y bajo presión puede transformarse en un líquido. La mayoría de la gente está familiarizada con el olor del amoniaco debido a su uso en sales aromáticas, detergentes de uso doméstico y productos para limpiar vidrios. El amoniaco se disuelve fácilmente en agua. Esta forma se conoce también como amoniaco líquido, amoniaco acuoso o solución de amoniaco [35].

Para esta prueba, se tomaron las mediciones dejando el sensor expuesto a hidróxido de amonio, que es amoniaco acuoso, en una cámara limpia de 1 m^3, durante 2 minutos. El registro se hizo en LabVIEW. En la figura 38 se presentan los resultados de la experimentación con distintos valores de resistencia.

Figura 37. Circuito impreso de sensor MQ135

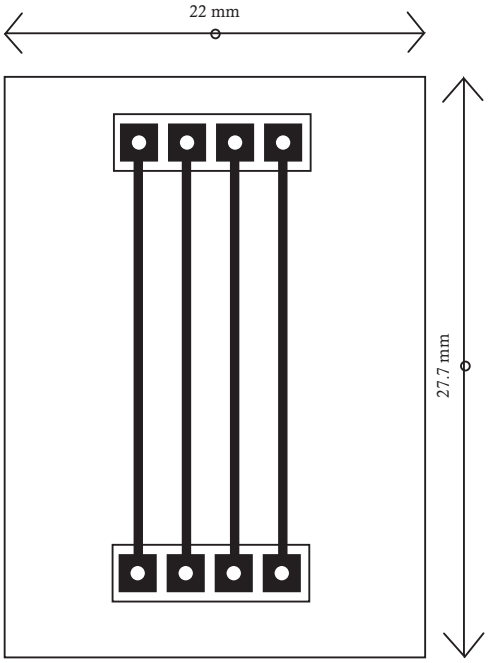
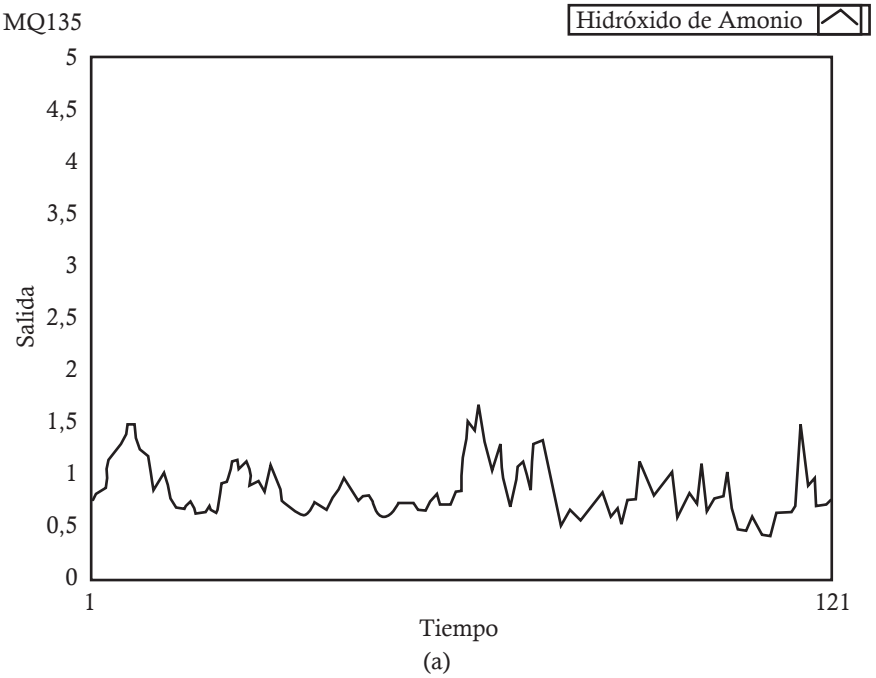
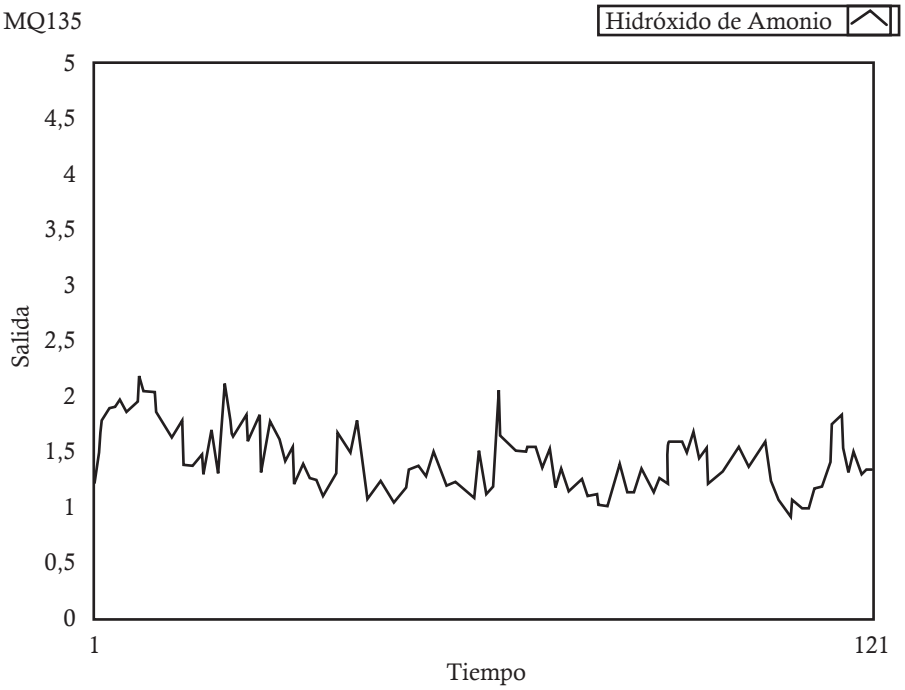


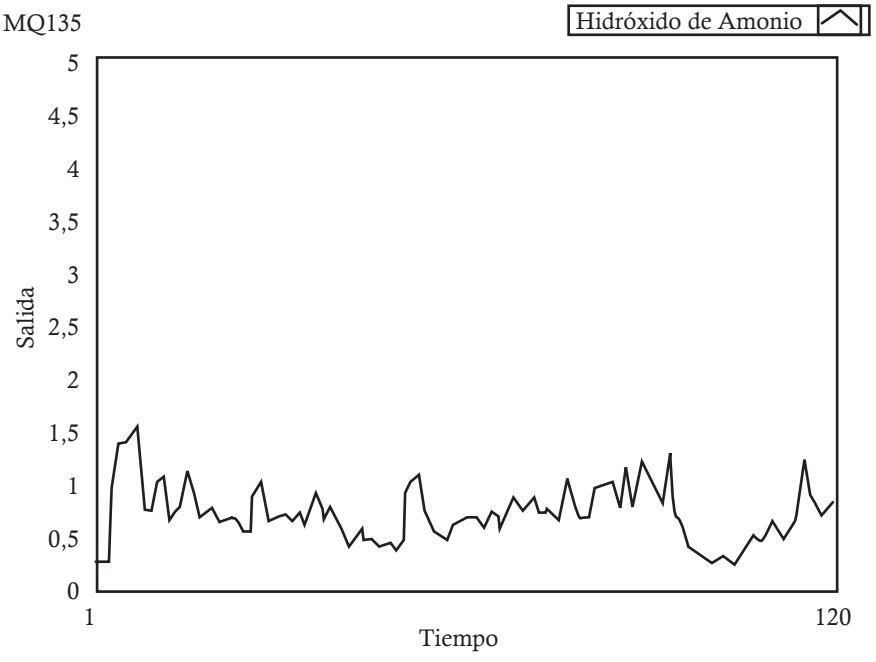
Figura 38. Respuesta del MQ135 al hidróxido de amonio. (a) R = 900 KΩ (b) R= 500 KΩ (c) R = 200 KΩ (d) R = 40 Ω



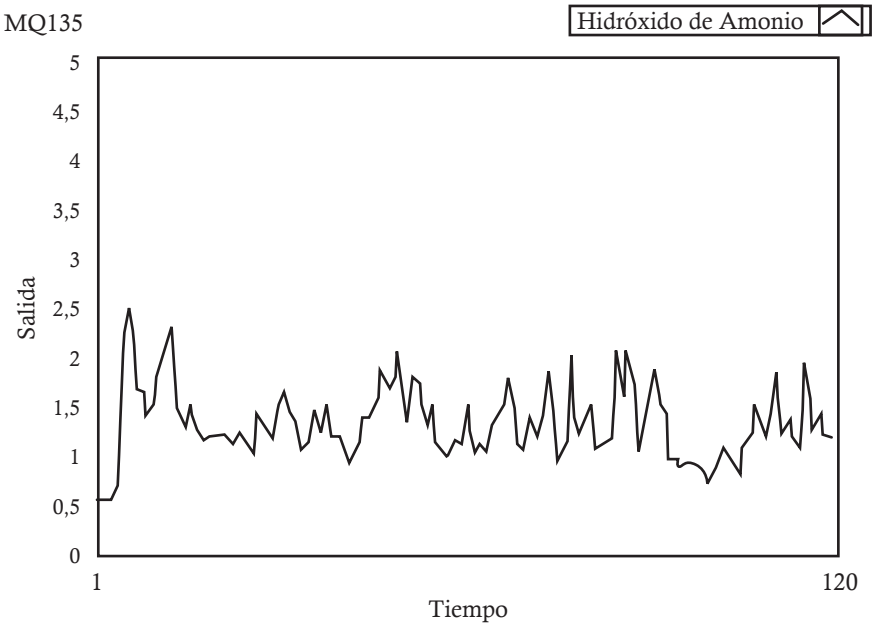
(a)


MQ135
Hidróxido de Amonio
Salida
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
1
121
Tiempo

(b)


MQ135
Hidróxido de Amonio
Salida
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
1
120
Tiempo

(c)



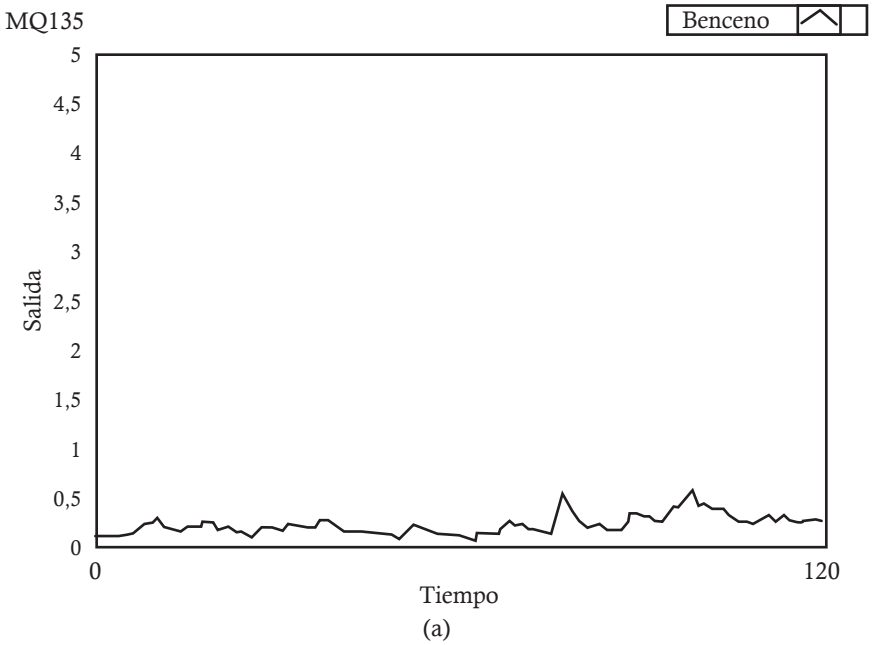
(d)

La segunda prueba que se le hizo al sensor es la del óxido de dinitrógeno, óxido nitrógeno (I) o el gas de la risa (N_2O), este gas es un gas incoloro con un olor dulce y ligeramente tóxico. Provoca alucinaciones, un estado eufórico y, en algunos casos, puede provocar pérdida de parte de la memoria.

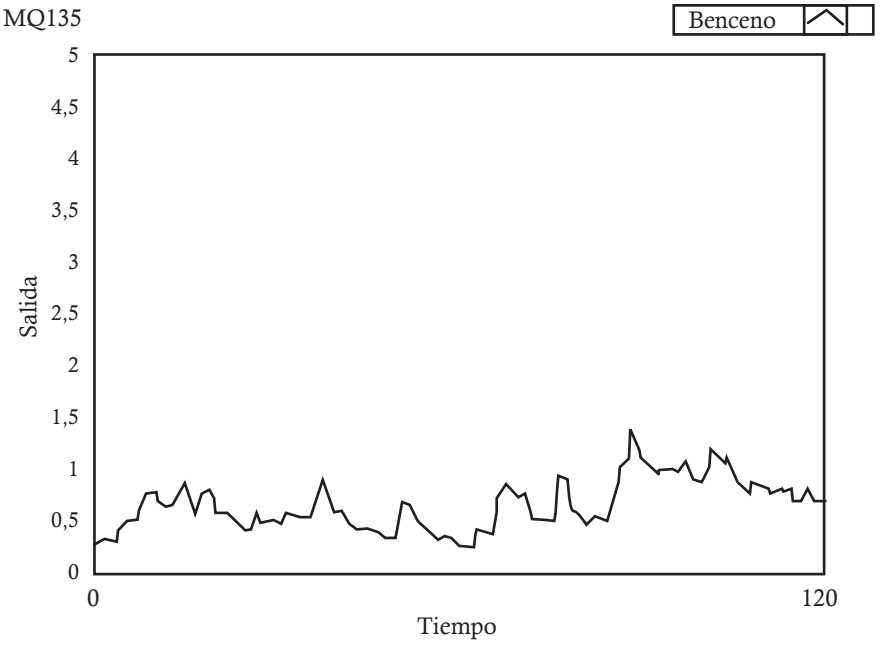
El óxido de dinitrógeno se forma en condiciones anaeróbicas a partir de abonos minerales en el suelo, y es un importante gas de efecto invernadero con una permanencia media de 100 años en la atmósfera. El óxido nitroso se encuentra en analgésicos, presente en el envasado a presión de productos alimenticios, como propelente en aerosoles, como agente de detección de fugas en equipos o instalaciones con vacío o presurizados, como refrigerante en forma gaseosa o líquida, presente en la congelación por inmersión de productos alimenticios y en la fabricación de lámparas incandescentes y fluorescentes [36, 37].

El óxido nitroso no es un combustible, pero facilita la combustión de otras sustancias. En esta prueba se tomaron las mediciones dejando el sensor expuesto a óxido nitroso, en una cámara limpia, durante 2 minutos, el registro se hizo en LabVIEW. En la figura 39 se presentan los resultados de la experimentación con distintos valores de resistencia.

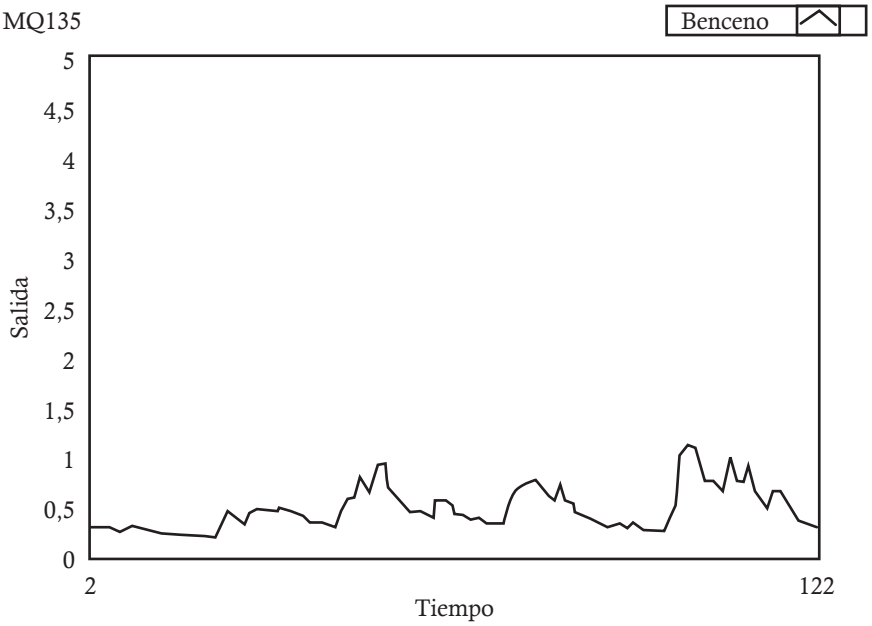
Figura 39. Respuesta del MQ135 al óxido nitroso. (a) R = 1 KΩ (b) R = 1.4 KΩ (c) R = 2 KΩ (d) R = 40 Ω



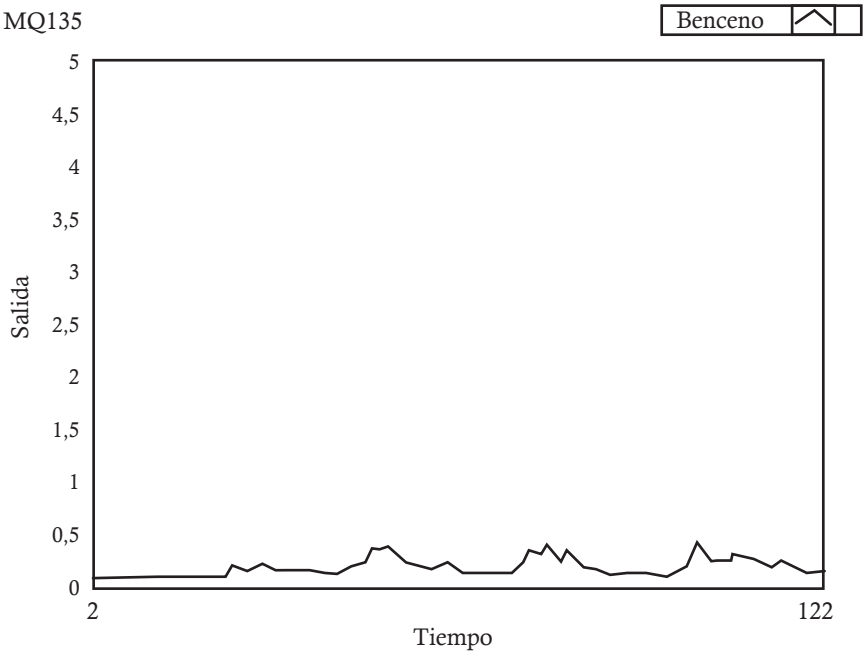
(a)



(b)


MQ135
Benceno
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
Salida
2
122
Tiempo

(c)


MQ135
Benceno
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
Salida
2
122
Tiempo

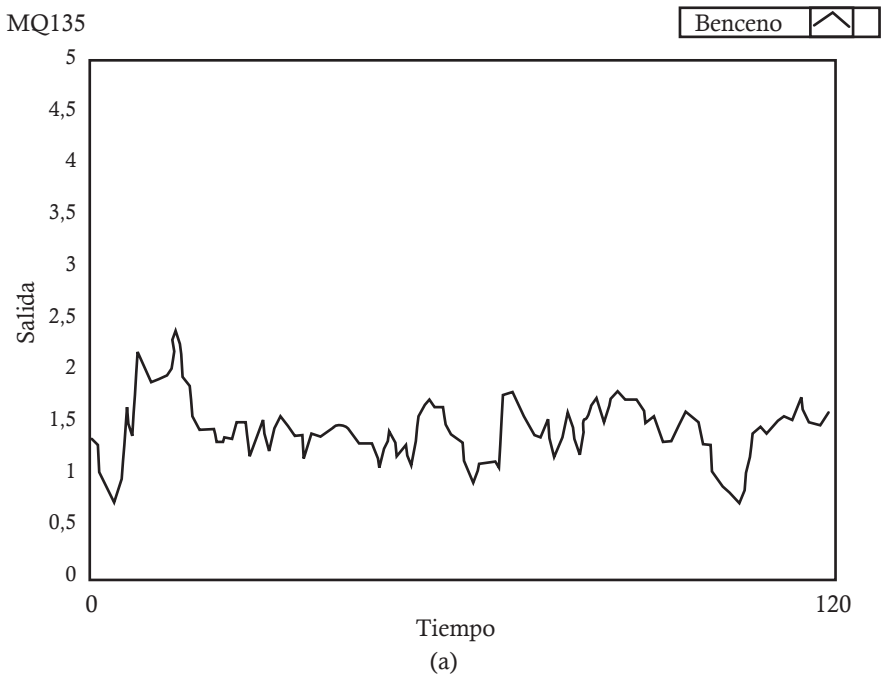
(d)

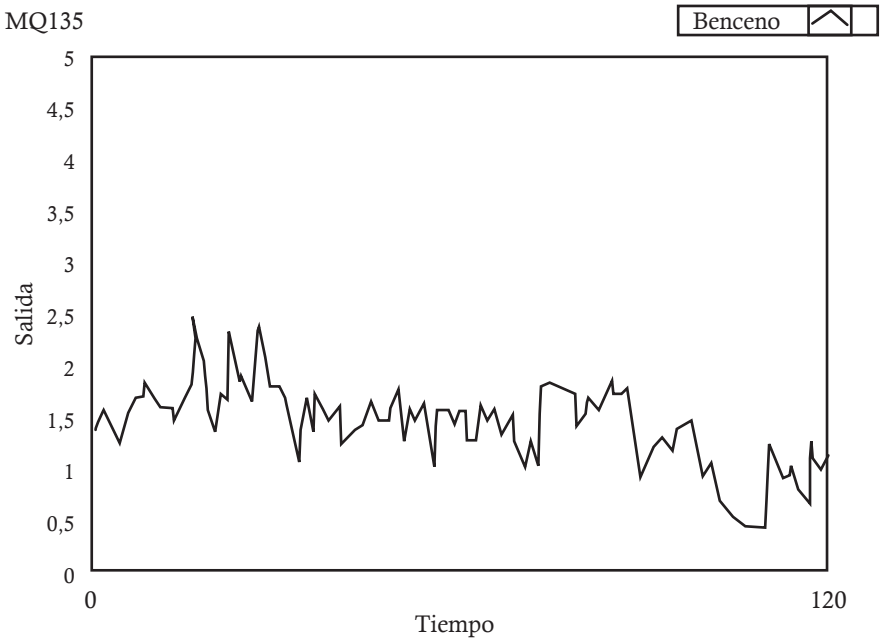
La tercera prueba del sensor se hizo con benceno. El benceno (C_6H_6) es una sustancia química líquida incolora o amarilla tenue a temperatura ambiente. Tiene un olor dulzón y es en extremo inflamable. El benceno se evapora en el aire con gran rapidez. Su vapor pesa más que el aire y puede hundirse hasta zonas muy bajas, solo se disuelve ligeramente en agua, y flotará en el agua [38, 39].

Los niveles de benceno en el aire pueden aumentar por emisiones generadas por la combustión de carbón y petróleo, los residuos de benceno, el tubo de escape de automóviles y evaporación de gasolina. El humo de tabaco es otra fuente de benceno en el aire, especialmente en el interior de viviendas. Las descargas industriales, la disposición de productos que contienen benceno y las fugas de gasolina desde tanques subterráneos liberan benceno al agua y al suelo [40].

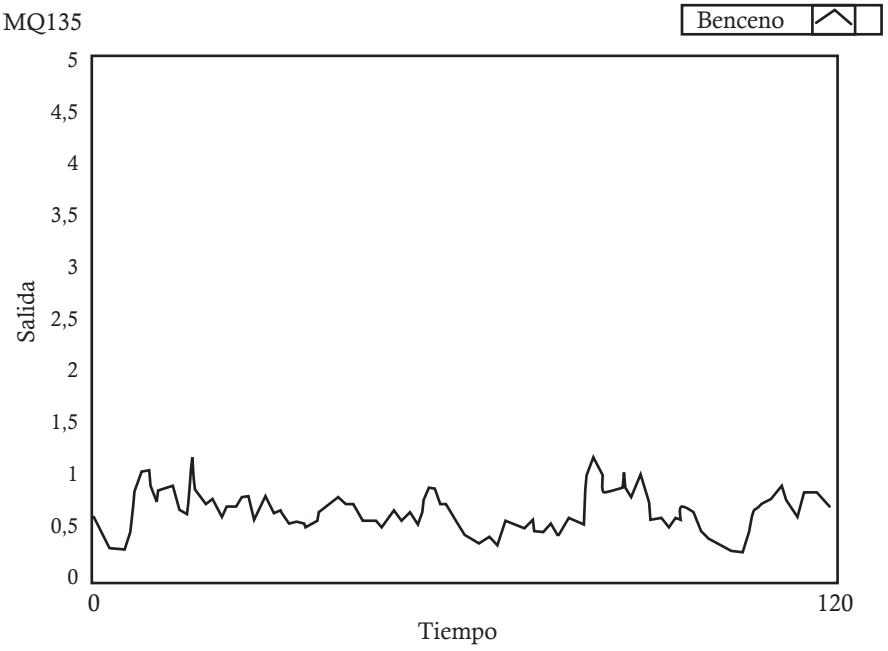
Para esta prueba se dejó expuesto el sensor a benceno líquido en una cámara limpia de 1 m^3, durante 2 minutos y se varió el valor de la resistencia para encontrar la mejor respuesta; los datos fueron tomados con gráfico en LabVIEW (figura 40).

Figura 40. Respuesta del MQ135 al benceno. (a) R = 1KΩ (b) R = 2.5 KΩ (c) R = 40 Ω (d) R = 2 KΩ

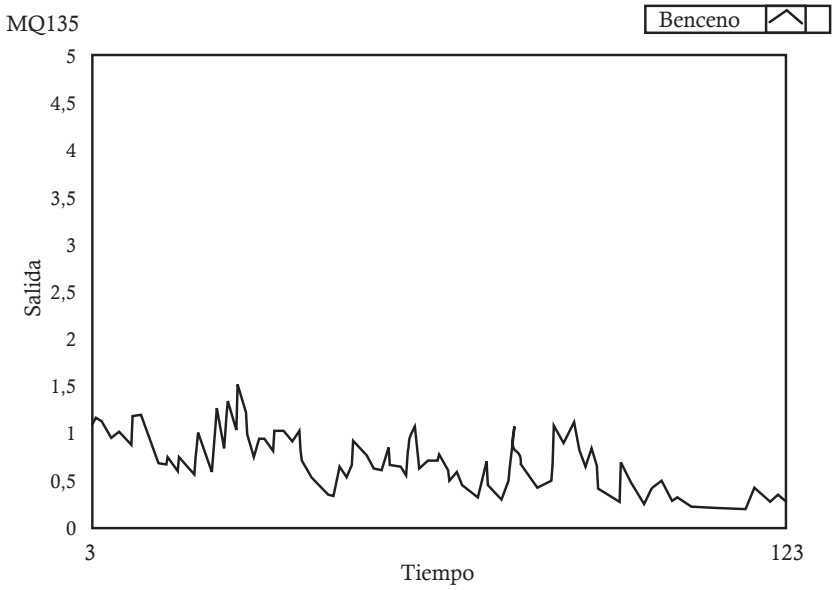



MQ135
Benceno
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
Salida
0
120
Tiempo

(b)


MQ135
Benceno
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
Salida
0
120
Tiempo

(c)



(d)

Sensores para la medición de la velocidad angular Giroscopio NGY1044

La velocidad de rotación se define como el ángulo girado por unidad de tiempo. En una plataforma móvil en relación con los ejes X, Y o Z, el conocimiento del ángulo rotado es de gran utilidad para conocer la orientación del robot, especialmente cuando no se conoce el espacio donde se moverá la plataforma [41]. El sensor NGY1044, que se muestra en la figura 41, es fabricado por Hitechic y hace parte del kit Lego Mindstorms NXT 2.0; este sensor detecta la rotación en los ejes X, Y y Z, y devuelve un valor que representa el número de grados por segundo de rotación. El sensor Gyro puede medir hasta +/- 360 ° por segundo de la rotación. El sensor Gyro está alojado en una carcasa de sensor Mindstorms estándar para que coincida con el resto de elementos del kit Mindstorms. La velocidad de rotación se puede leer hasta aproximadamente 300 veces por segundo [42].

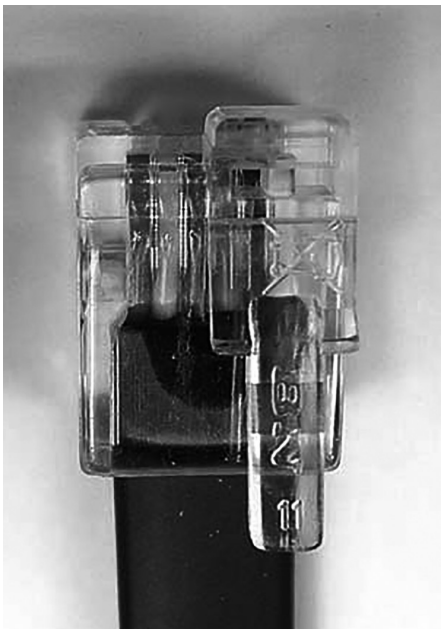
Figura 41. Giroscopio NGY1044

Fuente: [42].

El sensor Gyro está diseñado para ser conectado al brick NXT y le entrega las lecturas haciendo uso del protocolo I2C, pero cuando se usa con otro dispositivo maestro, se debe leer por un pin análogo, ya que la dirección de esclavo no está disponible.

El NGY1044 se conecta al Lego NXT con un cable RJ45, el cual se modificó en un extremo para poder obtener la lectura de voltaje que representa una velocidad angular determinada. El cable RJ45 (figura 42) tiene 6 hilos y están distribuidos así: blanco, la lectura análoga; negro, tierra; rojo, tierra; verde, la alimentación que va desde 4.5 V a 5.5 V, y el amarillo y azul que corresponden a SCL y SDA para la comunicación I2C con el brink NXT [43].

Figura 42. Conector RJ45 Lego NXT



Fuente: [43].

En la figura 43 se muestra el circuito impreso para la conexión del giroscopio a la tarjeta SBRIO9632 al pin análogo AI23.

Se hizo una prueba con otro sensor Gyro de Lego NXT, conectado al brick y el otro a la tarjeta SBRIO 9632 y se varió la rotación de ambos, con el objetivo de encontrar la equivalencia entre el voltaje entregado y el valor de la velocidad angular. En la tabla 8 se muestran los resultados de la experimentación con los 2 sensores.

Figura 43. Circuito impreso para el sensor NGY1044

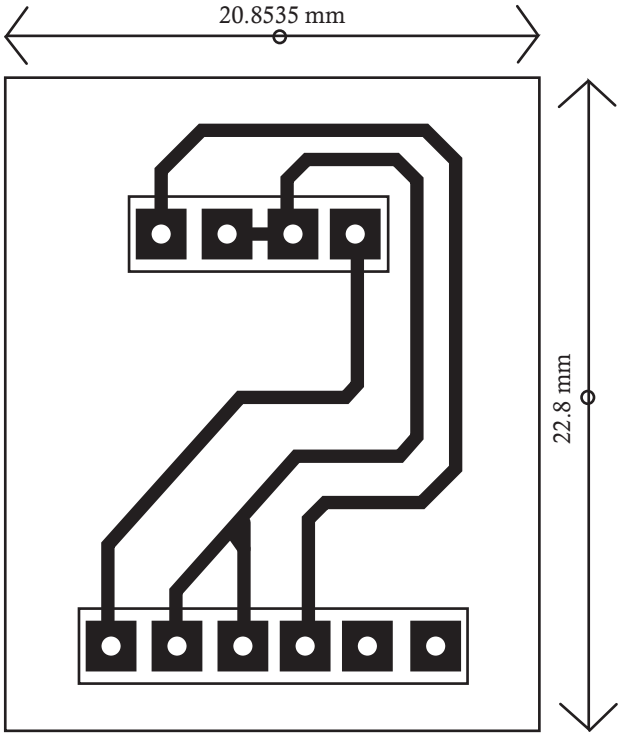


Tabla 8. Prueba al Gyro de NXT

Sensor 1 (grados/segundos)	Sensor 2 (v)
0	0
10	0.5
30	1.2
50	1.8
100	2.5
200	3
270	3.4

En la tabla 9 se muestran 10 medidas de voltaje para un ángulo de 90°.

Tabla 9. Medidas del sensor SHT71 en un punto de la escala

Medida	Voltaje
1	2.05
2	2.27
3	1.85
4	1.97
5	2.0
6	2.19
7	2.24
8	1.98
9	212
10	2.27

Para hallar el valor del sensor en ese punto, se toma el promedio de las medidas, así:

$$Voltaje(90°) = \frac{20.94}{10} = 2.094v$$

Con los valores obtenidos, se hallan el error absoluto y el error relativo expresados en voltaje, así:

$$E_a = 2v - 2.094v = -0.094$$

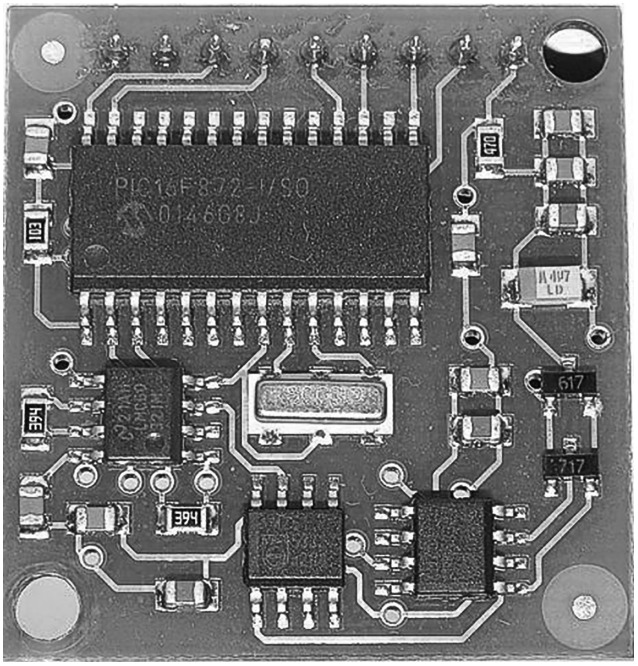
$$Er = \frac{0.094}{2} * 100\% = 4.7\%$$

Sensor para determinar la localización. Brújula digital CMPS03

Para conocer en todo instante la localización del robot fue necesario el uso de un sensor que diera información de su orientación global; de esta forma, se conoce qué área se está monitoreando y qué área falta por explorar.

La brújula digital CMPS03, que se muestra en la figura 44, fabricada por Davantech, es un sensor de campos magnéticos que está específicamente diseñado como sistema de navegación para robots. La brújula está basada en los sensores KMZ51 de Philips, que son lo suficientemente sensibles como para captar el campo magnético de la tierra. Tiene una interfaz I2C que permite obtener una lectura digital que consiste de un número único que representa la dirección de la componente horizontal del campo magnético natural [44].

Figura 44. Brújula digital CMPS03



Fuente: [44].

En la tabla 10 se presentan las características principales del sensor.

Tabla 10. Características CMPS03

Característica	Especificaciones
Voltaje de operación	5 v
Corriente	20 mA
Comunicación	I2C
Precisión	3-4 grados
Resolución	0.1 grados
Dimensiones	32 mm X 35 mm

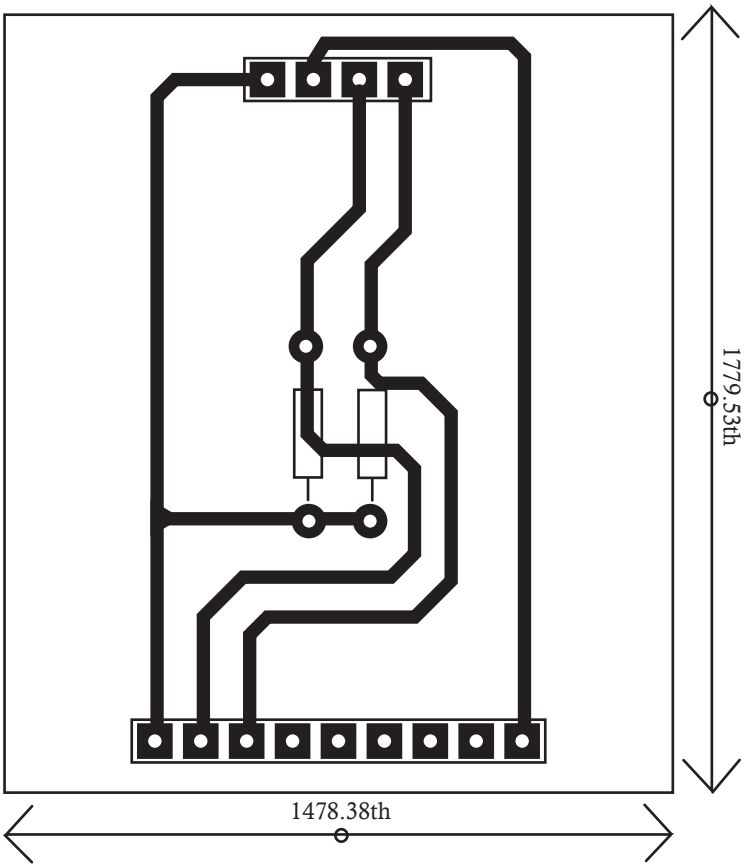
Fuente: [44].

La brújula magnética utiliza una conexión I2C. Tiene 9 pines que se distribuyen así: el pin 1 es la alimentación; el 2 y el 3 corresponden a las líneas SCL y SDA que deben de las resistencias de *pull up* conectadas a la alimentación con el fin de asegurar la comunicación en el bus; el pin 4 maneja una señal de PWM; el pin 5 y el 8 no se conectan; el 6 es el pin de calibración; el 7 es para la frecuencia, y finalmente, el pin 9 es la tierra.

La conexión del sensor a la tarjeta SBRIO 9632 se hizo por medio de 2 pines digitales (7IO4 y 7IO5), haciendo uso de la herramienta I2C de LabVIEW. La alimentación del sensor es tomada de la batería de la plataforma robótica.

En la figura 45 se muestra el circuito impreso realizado en ARES para la adaptación a la tarjeta.

Figura 45. Circuito impreso para brújula digital CMPS03



La prueba para la brújula magnética se hace tomando como referencia el sensor NXT Compass Sensor (NMC1034) que hace parte del Kit Lego Mindstorms NXT 2.0 [45]; el cual es conectado al brick del kit. El experimento consistió en variar la orientación de los dos sensores en distintos ángulos y registrar los datos obtenidos; se buscó el cero en el sensor NXT Compass y se colocó la brújula CMPS03 en la misma posición y se giró hacia la derecha. En la tabla 11 se presentan los datos tomados para diferentes orientaciones; los registros se hacen en LabVIEW.

Tabla 11. Prueba a la brújula magnética CMPS03

Sensor NMC1034 (grados)	Sensor CMPS03 (grados)
0	0
15	17.4
45	48
60	62.6
90	91.4
150	152.4
180	180.4
240	242.1
270	270.8
300	303

En la tabla 12 se muestran 10 medidas de ángulos para 150°, estos datos se tomaron haciendo girar la plataforma robótica cada vez.

Tabla 12. Medidas del sensor CMPS03 para 150°

Medida	Ángulo (grados)
1	150.32
2	155.45
3	148.87
4	145.68
5	150.56
6	151.6
7	149.09
8	156.70
9	147.7
10	152.51

Para hallar el valor del sensor en ese punto, se toma el promedio de las medidas, así:

$$Ángulo = \frac{1508.848º}{10} = 150.848º$$

Con los valores obtenidos, se halla el error absoluto y el error relativo expresados en voltaje, así:

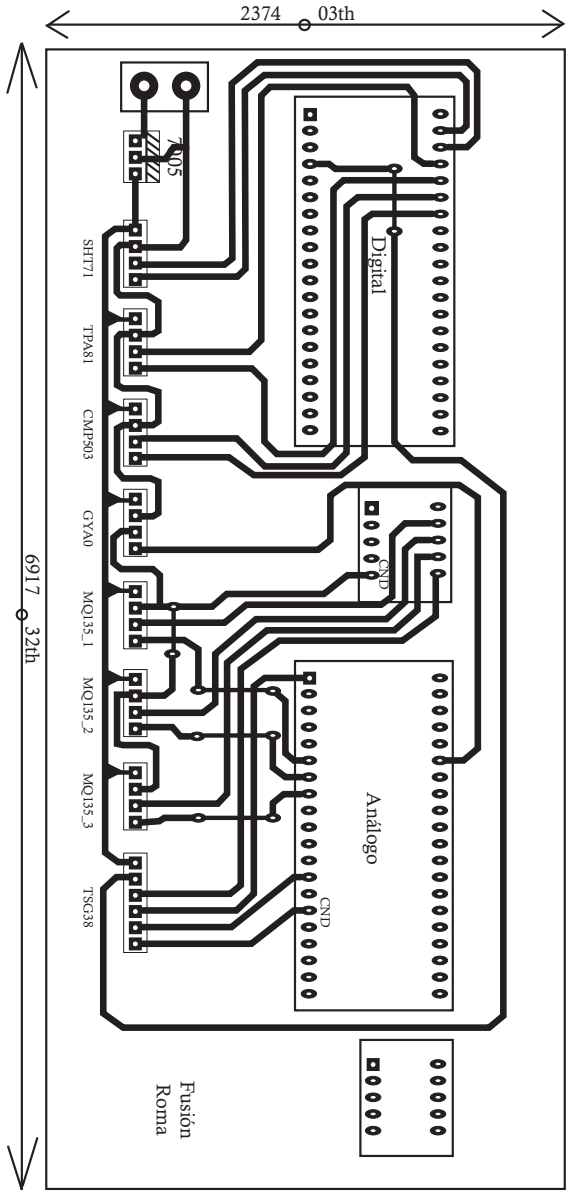
$$Ea = 150º - 150.848º = -0.848º$$

$$Er = \frac{0.848}{150} * 100\% = 0.565\%$$

Conexión principal entre los sensores y la plataforma

Cada puerto de la tarjeta SBRIO 9632 tiene 50 pines; para la conexión de los sensores, se usaron 2 puertos, uno digital y uno análogo; en la figura 46 se muestra el circuito impreso que contiene las conexiones de los puertos de la tarjeta con los conectores de todos los sensores.

Figura 46. Circuito impreso para la conexión de sensores y la tarjeta SBRIO 9632



Resultados

Durante el desarrollo de este capítulo, se obtuvieron los resultados que se muestran en la tabla 13, con su respectivo análisis.

Tabla 13. Resultados

Distancia para la detección de la fuente de calor (Sensor TPA81)	El sensor detecta fuentes de calor de hasta 3 m; sin embargo, para asegurar una lectura ideal, se toman las medidas de hasta 2 m: para que no se tome la temperatura ambiente.
Error absoluto y error relativo en las mediciones con el sensor SHT71	El error absoluto expresado en las unidades de las variables. En las mediciones de humedad, es de 0.271 % y en la mediciones de temperatura, de 0.069 °C. El error relativo expresado en porcentaje en las mediciones de humedad es de 0.046 % y en las mediciones de temperatura, de 0.319 %.
Histéresis de medición con el sensor SHT71	La histéresis encontrada en el recorrido por la escala de forma ascendente y descendente en la variable de humedad es del 2 % y la temperatura es de 2.42 %.
Rangos de detección de monóxido de carbono y metano con el sensor TGS3870	Para voltajes menores a 3 voltios, hay detección de monóxido de carbono, y para voltajes entre 3.5 voltios y 5 voltios, hay detección de metano.
Resistencia de carga para la medición del amoniaco con el sensor MQ153	La resistencia de carga ideal para la medición de concentraciones de amoniaco entre 10 y 300 ppm es de 40 Ω.
Resistencia de carga para la medición del benceno con el sensor MQ153	La resistencia de carga ideal para la medición de concentraciones de benceno entre 10 y 300 ppm es de 2.5 KΩ.
Resistencia de carga para la medición del óxido nitroso con el sensor MQ153	La resistencia de carga ideal para la medición de concentraciones de óxido nitroso entre 10 y 300 ppm es de 1.4 KΩ.
Error absoluto y error relativo en las mediciones con el sensor CMPS03	El error absoluto de la brújula digital es de 0.848 ° y el error relativo es de 0.56 %.
Equivalencia entre velocidad angular y voltaje de los sensores Gyro de LEGO	Al hacer una regresión lineal con los datos obtenidos, se obtuvo que la equivalencia. entre voltaje y velocidad angular es $\nu = 0.011 + 0.7\omega$.
Error absoluto y error relativo en las mediciones con el sensor Gyro de LEGO	El error absoluto del giroscopio es de 0.094 voltios y el error relativo es de 4.7 %.

Algoritmos de fusión de sensores

Este apartado muestra el desarrollo de un sistema de fusión sensorial para el monitoreo de las condiciones ambientales de un entorno, como la temperatura, la humedad relativa y el calor radiante; también, se determina la presencia de gases tóxicos e inflamables, como el metano, el monóxido de carbono, el benceno, el amoniaco y el óxido nitroso, con el objetivo de predecir una emergencia como incendio, explosión o intoxicación.

Para hacer la fusión de datos se usa la lógica difusa, que transforma los valores reales en entradas con variables lingüísticas que tendrán un valor de pertinencia; de la misma forma, se tienen las salidas que al final del proceso serán convertidas nuevamente a valores reales.

Se toma un nivel de fusión 3; donde se tienen en cuenta las decisiones con base en las anteriores; es decir, de acuerdo con la información obtenida se tienen reglas que darán lugar a una inferencia de mayor complejidad; donde se tiene complemento, redundancia y cooperatividad.

Propuesta de fusión sensorial

En la figura 47 se muestra el esquema general de la alternativa de solución planteada desde el inicio del proyecto; donde se tienen los diferentes sensores que hacen posible el monitoreo de las condiciones ambientales del entorno; sensores de gas, sensor de temperatura y humedad relativa y sensor de calor radiante. Otro tipo de sensores que se tienen son los que permiten la navegación de la plataforma por el entorno: encoders, brújula magnética y giroscopio.

La información suministrada por los sensores se fusiona: de forma complementaria, para obtener una nueva información a partir de la existente; de forma cooperativa, donde la información de las fuentes da como resultado datos más complejos; o de forma redundante, donde la información de una fuente es confirmada por la de las otras.

Luego del proceso de fusión, se tiene información más compleja, como las condiciones de temperatura ambiente y humedad relativa del entorno, la localización del robot y datos acerca de posibles emergencias que se pueden presentar como incendio, explosión e intoxicación [17].

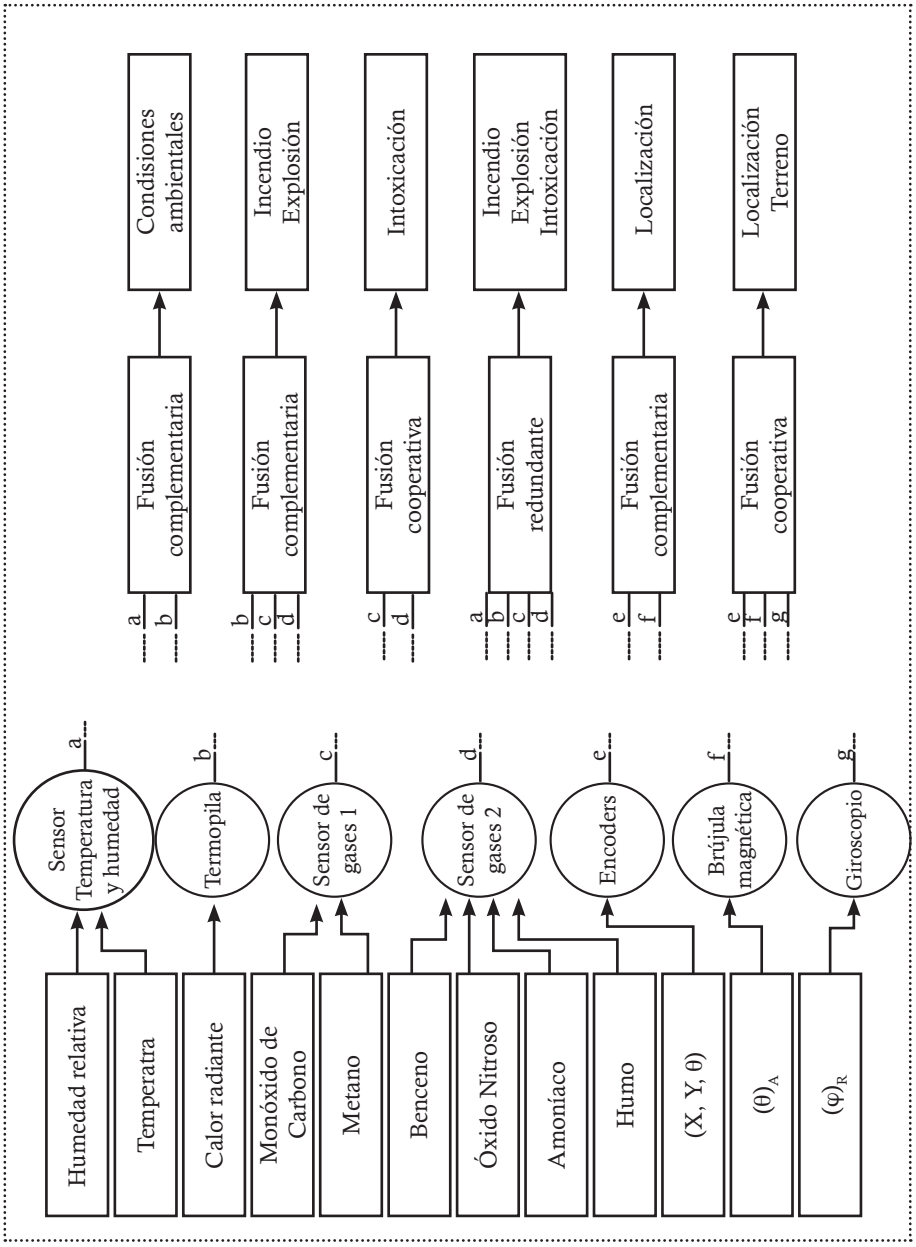


Figura 47. Propuesta de fusión sensorial

Cada una de las combinaciones y el tipo de fusión presentados en la figura 47 se fundamentan en la naturaleza de los gases, el efecto que tienen en la salud y la reacción que producen al combinarse con otros gases o al exponerse a determinadas condiciones de humedad relativa o temperatura.

Riesgo químico

En este apartado se hace una justificación de por qué es importante la detección de los gases mencionados para la prevención de emergencias y las condiciones en que explotan, provocan una intoxicación o desatan un incendio; esto con la intención de construir las reglas que rigen el sistema difuso.

El riesgo químico es el que se produce por una exposición no controlada a sustancias químicas que puede producir efectos agudos y accidentes. Los productos químicos tóxicos también pueden provocar consecuencias locales y sistémicas según la naturaleza del producto y la vía de exposición. Según la sustancia, las consecuencias pueden ser graves para la salud y los daños permanentes en el medio [46 47].

Los factores que determinan el riesgo al que se está expuesto dependen de la composición de la sustancia, el estado del producto y la exposición a la sustancia. Cualquier material nocivo, que durante su fabricación, almacenamiento, transporte o uso genere o desprenda humos, gases, vapores, polvos o fibras, se considera peligroso y dependiendo de su naturaleza, se clasifica de acuerdo con el *Reglamento de envasado y etiquetado de sustancias peligrosas* (REAL Decreto 363/1995), que establece categorías, símbolos y letras para dichas sustancias [48]:

- Explosivos: (E) R2-R3. Sustancias que pueden explotar bajo efecto de una llama, fuente de calor o que son sensibles a los choques o fricciones. Ejemplo: nitroglicerina.
- Inflamables: (F): R10. Sustancias que pueden calentarse e inflamarse en contacto con el aire a una temperatura normal o que en contacto con el agua o el aire húmedo generan gases inflamables en cantidades peligrosas. Ejemplo: benceno, etanol, acetona.
- Extremadamente inflamable: R-12. Sustancias líquidas, cuyo punto de inflamación se sitúa entre los 21 ºC y los 55 ºC. Por ejemplo: hidrógeno, etino, éter etílico.
- Comburentes: (O) R7-R8-R9. Sustancias que tienen la capacidad de incendiar otras sustancias, facilitando la combustión e impidiendo el combate del fuego. Ejemplo: oxígeno, óxido nitroso, nitrato de potasio.
- Corrosivos: (C) R34-R35. Sustancias que causan destrucción de tejidos vivos o materiales inertes. Ejemplo: ácido clorhídrico, ácido fluorhídrico.
- Irritante: (Xi) R36-R37-R38-R41. Sustancias no corrosivas que, por contacto con la piel o las mucosas, provocan una reacción inflamatoria. Ejemplo: cloruro de calcio, carbonato de sodio.
- Nocivos: (X) R20-R21-R22-R65-R68. Sustancias que por inhalación, ingestión o penetración cutánea tienen riesgos en la salud de forma temporal o alérgica. Ejemplo: etanol, dicloro-metano.

- Sensibilizantes: R42-R43. Sustancias que por inhalación o penetración cutánea ocasionan hipersensibilidad.
- Tóxicos: (T) R23-R24-R25-R39. Sustancias que por inhalación, ingestión o penetración cutánea tienen riesgos graves, agudos o crónicos a la salud. Ejemplo: cloruro de bario, monóxido de carbono, metanol.
- Muy tóxicos: R26-R27-R28-R39. Sustancias que por inhalación, ingestión o penetración cutánea causan graves problemas de salud e, inclusive, la muerte. Ejemplo: cianuro, trióxido de arsenio, nicotina.
- Carcinogénicos: R40-R45-R49. Sustancias que por inhalación, ingestión o penetración cutánea producen cáncer o aumentan su frecuencia.
- Mutagénicos: R46-R68. Sustancias que por inhalación, ingestión o penetración cutánea producen alteraciones genéticas hereditarias.
- Tóxicos para la reproducción: R60-R61-R62-R63. Sustancias que por inhalación, ingestión o penetración cutánea generan efectos negativos no hereditarios en la descendencia y afectan la capacidad reproductora.
- Radiactivos: sustancias que emiten radiaciones nocivas para la salud.
- Peligroso para el medio ambiente: (N) R50-R51-R52-R53-R54-R55-R56-R57-R58-R59. Sustancias que causan daños al ecosistema a corto o largo plazo.
- Los peligros relacionados con la seguridad en el manejo de agentes químicos se traducen en incendios, explosiones y reacciones que pueden causar daños a los seres vivos de forma permanente [47].

Al tener la clasificación de las distintas sustancias químicas, se muestra el riesgo asociado a los gases que detectarán para el sistema de fusión sensorial.

Peligro de los gases a detectar

Como primer gas mencionaremos el metano. El metano es un gas incoloro e inodoro, que se utiliza como fuente de luz y combustible, y es el principal componente del gas natural. También se utiliza en la elaboración de muchas sustancias químicas, como acetileno y metanol [49].

El metano puede afectar por inhalación; si la piel se contacta con el metano puede causar congelación. Los niveles muy altos de metano pueden disminuir la cantidad de oxígeno en el aire y causar asfixia, con síntomas de dolor de cabeza, mareo, debilidad, náusea, vómitos, pérdida de la coordinación y del juicio, aumento en la frecuencia respiratoria y pérdida del conocimiento.

El metano es un gas inflamable que presenta un grave riesgo de incendio y, debido a los gases que desprende, puede causar una explosión. Los límites de exposición al metano son de 1000 ppm en 8 horas. Una concentración de 5000 ppm es letal [50, 51].

El segundo gas es monóxido de carbono. El monóxido de carbono puede afectar si se inhala; respirarlo puede causar dolor de cabeza, mareo, sensación de desvanecimiento y cansancio; en niveles altos de exposición, causa somnolencia, alucinaciones, convulsiones, pérdida de conocimiento, cambios en la memoria y en la personalidad, confusión mental y pérdida de visión [51].

La exposición extremadamente alta al monóxido de carbono puede causar la formación de carboxihemoglobina, que reduce la capacidad de la sangre para transportar oxígeno y puede causar un color rojo brillante en la piel y las membranas mucosas, dificultad respiratoria, colapso, convulsiones, coma y la muerte [52]. Niveles altos del gas se dan debido al uso de aparatos que queman gas natural, querosén u otros combustibles y que están mal instalados o que funcionan sin ventilación adecuada.

Es un gas inflamable y presenta un grave peligro de incendio; el límite de exposición recomendado en el aire es de 35 ppm como promedio durante un turno laboral de 10 horas y de 200 ppm, que no debe sobrepasarse en ningún periodo laboral de 15 minutos [53].

El tercer gas es el óxido nitroso. El contacto con el óxido nitroso puede provocar congelación, euforia, somnolencia y pérdida del conocimiento. Puede causar envenenamiento y asfixia. Los indicios y síntomas pueden incluir dolor de cabeza, mareos y excitación, que pueden progresar a depresión del sistema nerviosos central, convulsiones y muerte [54]. No es combustible, pero facilita la combustión de otras sustancias. En caso de incendio, se desprenden humos (o gases) tóxicos e irritantes y hay un alto riesgo de incendio y explosión [55].

La inhalación del 40 % de óxido nitroso en el aire puede causar confusión y sedación, mientras que un nivel del 80 % puede causar inconsciencia en muchos individuos. Concentraciones de más de 90 mg/m^3 (50 ppm) pueden reducir la destreza, la cognición y las habilidades motoras y audiovisuales [55].

El cuarto gas es el amoniaco. El amoniaco puede afectar si se inhala, es una sustancia corrosiva y puede irritar y quemar la piel y los ojos, y causar daño permanente. La exposición al amoniaco puede irritar la nariz, la boca y la garganta, y causar tos y respiración con silbido.

Respirar amoniaco puede irritar los pulmones y causar tos o falta de aire. En niveles mayores, la exposición puede causar una acumulación de líquido en los pulmones (edema pulmonar), una emergencia médica, con una intensa falta de aire. Concentraciones bastante bajas de amoniaco producen una irritación rápida de los ojos. A concentraciones mayores, produce quemaduras graves [56].

Las mezclas de amoniaco y aire originarán explosiones si se encienden en condiciones inflamables. Es extremadamente inflamable; el calentamiento intenso puede producir aumento de la presión con riesgo de estallido. El límite recomendado de exposición en el aire es de 25 ppm, como promedio durante un turno laboral de 10 horas y de 35 ppm, que no debe excederse durante ningún periodo de trabajo de 15 minutos [57, 58].

El quinto gas es el benceno. El benceno es una sustancia carcinógena e inflamable, que al incendiarse desprende otros gases tóxicos que pueden explotar. El benceno puede afectar si se inhala; al atravesar la piel, causa irritación, sequedad, formación

de escamas, daños en los ojos y la piel. La exposición puede irritar la nariz y la garganta, causa síntomas como mareo, sensación de desvanecimiento, dolor de cabeza y vómitos [59]. Después de la exposición alta, pueden sobrevenir convulsiones y coma, o la muerte súbita por ritmo cardíaco irregular. La exposición repetida puede causar daño a los glóbulos sanguíneos (anemia aplástica) [59].

El benceno es un líquido inflamable y presenta peligro de incendio. La exposición permitida es de 0.1 ppm como promedio durante un turno laboral de 10 horas y de 1 ppm, que no debe excederse durante ningún periodo laboral de 15 minutos [60].

Entradas y salidas del sistema

El primer paso para la implementación del sistema de fusión sensorial basado en lógica difusa es obtener las entradas y las salidas de este y someterlas a un proceso de fuzzificación, es decir, pasar de valores reales a funciones con un grado de pertinencia, expresadas con variables lingüísticas.

Como la lógica difusa es una técnica que permite la expresión del razonamiento humano, haciendo uso de variables imprecisas [20], se construyeron las funciones de pertinencia para las entradas y salidas de acuerdo con la interpretación de los investigadores.

De acuerdo con lo anterior, se plantearon las entradas del sistema, como:

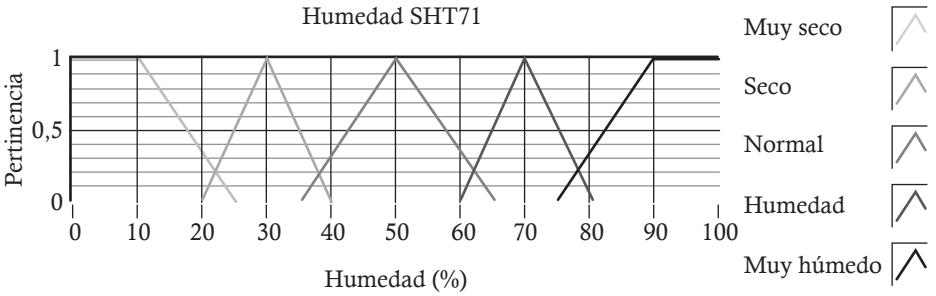
- Humedad relativa.
- Temperatura.
- Calor radiante.
- Concentración de metano, monóxido de carbono, óxido nitroso, benceno y amoniaco.

A continuación se explica la interpretación dada a los datos entregados por los sensores para la construcción de cada una de las funciones de pertinencia de las entradas y salidas del sistema.

- Humedad relativa: para la humedad relativa, hay 5 funciones de pertinencia; 3 triangulares y 3 trapezoidales para los valores límite. Entre 0 y 25% de humedad relativa se considera muy seco, entre 20 y 40% es seco, entre 35 y 65% es normal, entre 60 y 80% es húmedo y entre 75 y 100% es muy húmedo.

En la figura 48a se muestra la entrada de humedad relativa construida en LabVIEW.

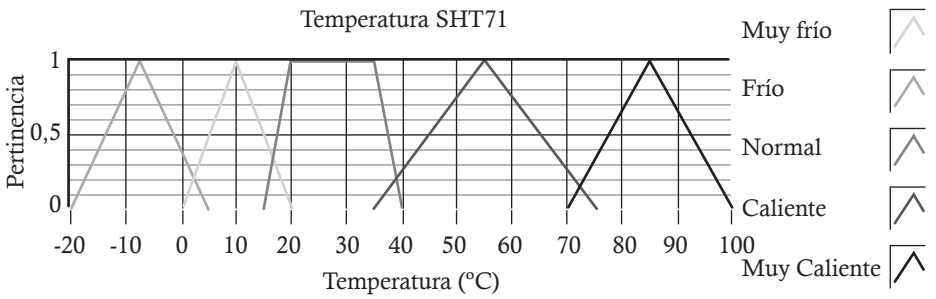
Figura 48a. Fuzzificación de la humedad relativa



- Temperatura: para la temperatura, hay 5 funciones de pertinencia; 4 triangulares y 1 trapezoidal para el valor central. Entre -20 y 5 °C se considera muy frío, entre 0 y 20 °C es frío, entre 15 y 40 °C es normal, entre 35 y 75 °C es caliente y entre 70 y 100 °C es muy caliente.

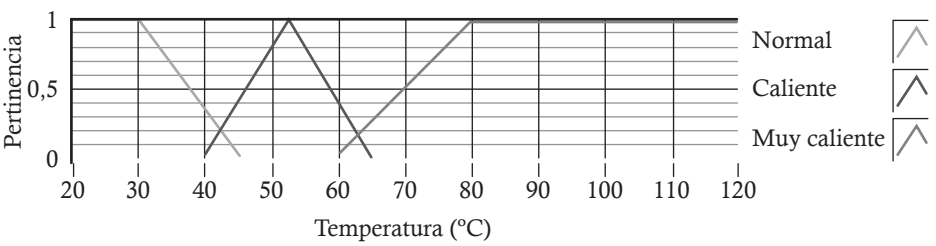
En la figura 48b se muestra la entrada de temperatura construida en LabVIEW.

Figura 48b. Fuzzificación de la temperatura



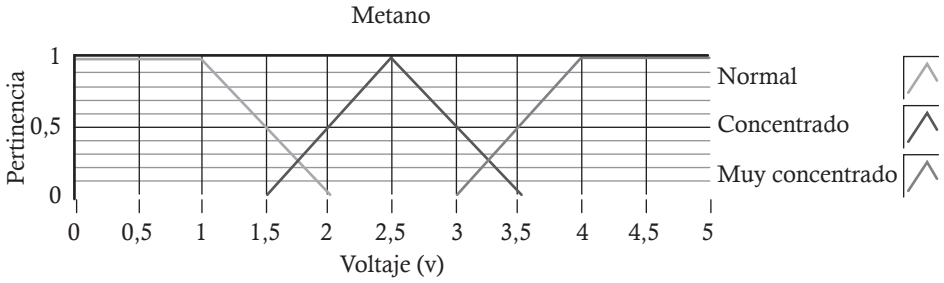
- Fuente de calor: para determinar si existe una fuente de calor, se toma el valor más alto de las 8 lecturas que entrega el sensor TPA81; para esto, hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 20 y 45 °C se considera normal, entre 40 y 65 °C es caliente, entre 60 y 120 °C es muy caliente. En la figura 49 se muestra la entrada para el calor radiante construido en LabVIEW.

Figura 49. Fuzzificación para el calor radiante



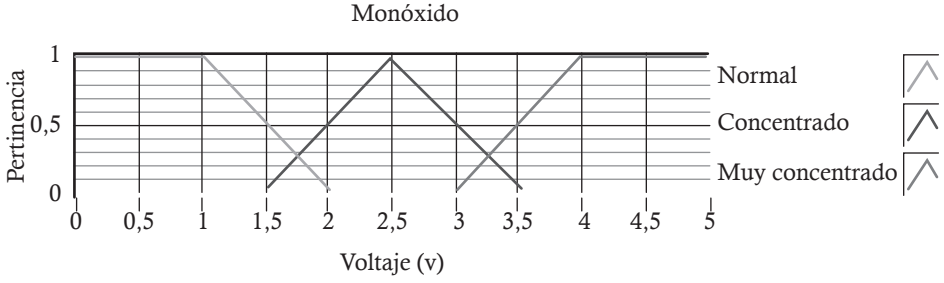
- Metano: para la detección de concentraciones de metano, se evalúa la entrada con el valor de voltaje entregado por el sensor. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 2 V se considera normal, es decir, que no hay ningún peligro asociado a ese gas; entre 1.5 y 3.5 V es concentrado y entre 3 y 5 V es muy muy concentrado. En la figura 50 se muestra la entrada para el metano construido en LabVIEW.

Figura 50. Fuzzificación para el metano



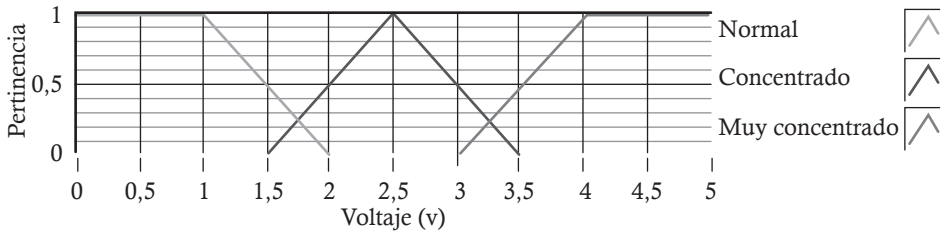
- Monóxido de carbono: para la detección de concentraciones de monóxido de carbono, se evalúa la entrada con el valor de voltaje entregado por el sensor. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 2 V se considera normal, es decir, que no hay ningún peligro asociado a ese gas; entre 1.5 y 3.5 V es concentrado y entre 3 y 5 V es muy muy concentrado. En la figura 51 se muestra la entrada para el calor radiante construido en LabVIEW.

Figura 51. Fuzzificación para el monóxido de carbono



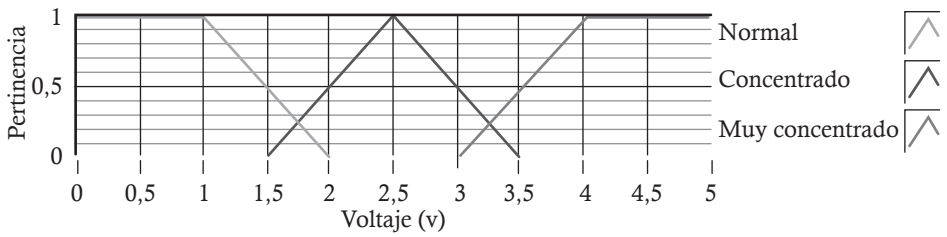
- Óxido nitroso: para la detección de concentraciones de óxido nitroso, se evalúa la entrada con el valor de voltaje entregado por el sensor. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 2 V se considera normal, es decir, que no hay ningún peligro asociado a ese gas; entre 1.5 y 3.5 V es concentrado y entre 3 y 5 V es muy concentrado. En la figura 52 se muestra la entrada para el óxido nitroso construido en LabVIEW.

Figura 52. Fuzzificación para el óxido nitroso



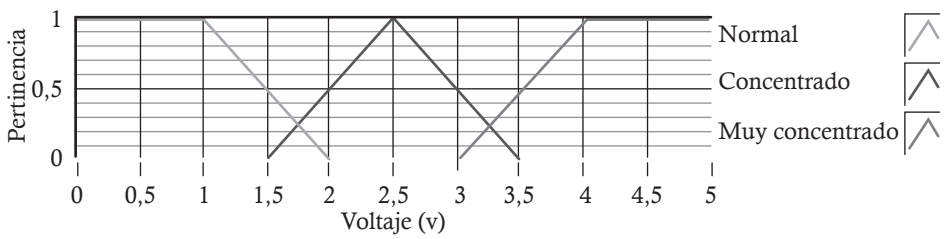
- Amoniaco: para la detección de concentraciones de amoniaco, se evalúa la entrada con el valor de voltaje entregado por el sensor. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 2 V se considera normal, es decir, que no hay ningún peligro asociado a ese gas; entre 1.5 y 3.5 V es concentrado, y entre 3 y 5 V es muy concentrado. En la figura 53 se muestra la entrada para el amoniaco construido en LabVIEW.

Figura 53. Fuzzificación para el amoniaco



- Benceno: para la detección de concentraciones de benceno, se evalúa la entrada con el valor de voltaje entregado por el sensor. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 2 V se considera normal, es decir, que no hay ningún peligro asociado a ese gas; entre 1.5 y 3.5 V es concentrado y entre 3 y 5 V es muy concentrado. En la figura 54 se muestra la entrada para el benceno construido en LabVIEW.

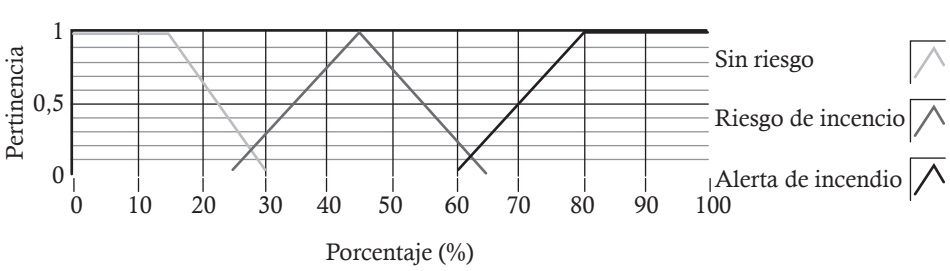
Figura 54. Fuzzificación para el benceno



Las salidas del sistema son las alarmas ante posibles emergencias; sin riesgo de incendio, alarma de incendio, intoxicación, alarma de intoxicación, explosión, riesgo de explosión y alarma de explosión.

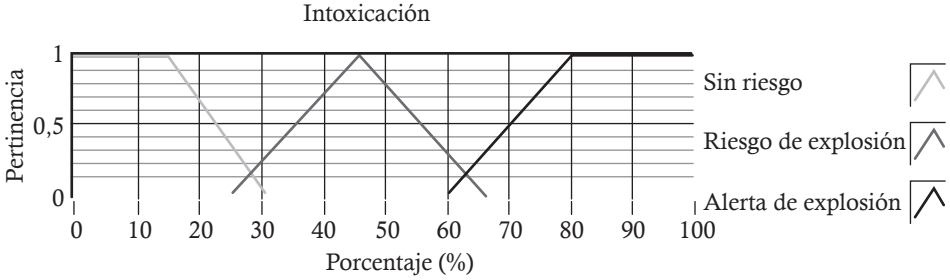
- Incendio: para la predicción de incendio causado por las distintas combinaciones entre las entradas, se evalúa en porcentaje de 0 a 100%. Se tienen 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 30% se considera que no existe riesgo, entre 25 y 65% existe el riesgo de incendio y entre 60 y 100% hay una alarma de incendio. En la figura 55 se muestra la salida para incendio en LabVIEW.

Figura 55. Fuzzificación para incendio



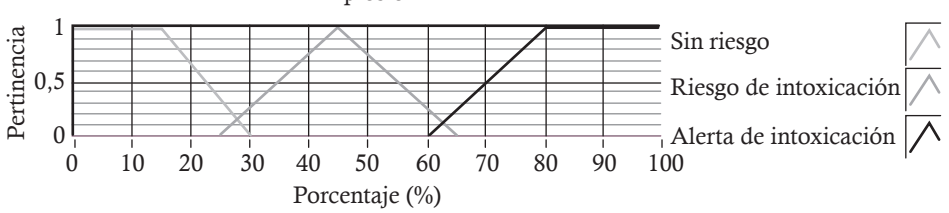
- Intoxicación: para la predicción de intoxicación causada por las distintas combinaciones, entre las entradas se evalúa en porcentaje de 0 a 100%. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 30% se considera que no existe riesgo, entre 25 y 65% existe el riesgo de intoxicación y entre 60 y 100% hay una alarma de intoxicación. En la figura 56 se muestra la salida en LabVIEW.

Figura 56. Fuzzificación para intoxicación



- Explosión: para la predicción de explosión causada por las distintas combinaciones, entre las entradas se evalúa en porcentaje de 0 a 100%. Hay 3 funciones de pertinencia; 1 triangular y 2 trapezoidales para los valores límite. Entre 0 y 30% se considera que no existe riesgo, entre 25 y 65% existe el riesgo de explosión y entre 60 y 100% hay una alarma. En la figura 57 se muestra la salida en LabVIEW.

Figura 57. Fuzzificación para explosión



Reglas

Se construyeron 16 reglas que son la base del sistema difuso y sobre ellas se hace la evaluación de las emergencias; las condiciones usan un conector AND (mínimo) en las entradas. La defuzzificación se hace por el método del centroide con un sistema difuso tipo Mandani, que toma el valor que ingresa al sistema difuso y es convertido en un nivel de pertinencia, posteriormente se asocia a una regla, obteniendo la salida que finalmente se convierte nuevamente a un valor real.

Como se explicó, dependiendo de las condiciones de temperatura y humedad relativa a la que sean expuestos los gases metano, monóxido de carbono, óxido nitroso, benceno y amoniaco, se pueden desatar distintas situaciones peligrosas. Inicialmente, se tienen las reglas que darán como salida la situación de no riesgo, donde todas las entradas están en valores normales; después, se encuentran las salidas de riesgo de incendio provocadas por condiciones ambientales fuera del rango que se tiene como normal, como lo son alta temperatura y alta humedad.

Posteriormente, se hacen las combinaciones entre la alta temperatura, la alta humedad y cada uno de los gases en condiciones de concentración y alta concentración. Y, finalmente, se presentan las reglas para las situaciones en las cuales las condiciones ambientales son normales, pero las concentraciones de los gases tóxicos e inflamables son medias y altas; en estos casos existen las alertas de intoxicación y explosión; además, se presentan los casos particulares en los cuales el óxido nitroso que no es nocivo por sí solo se combina con los demás gases, lo que genera las emergencias.

- IF ‘temp tpa’ IS ‘normal’ AND ‘Temperatura’ IS ‘normal’ AND ‘Humedad’ IS ‘normal’ AND ‘oxido nitroso’ IS ‘normal’ AND ‘amoniaco’ IS ‘normal’ AND ‘benceno’ IS ‘normal’ AND ‘metano’ IS ‘normal’ AND ‘monoxido de carbono’ IS ‘normal’ THEN ‘incendio’ IS ‘sin riesgo’ ALSO ‘explosion’ IS ‘sin riesgo’ ALSO ‘intoxicacion’ IS ‘sin riesgo’.
- IF ‘Humedad’ IS ‘humedo’ AND ‘Temperatura’ IS ‘caliente’ AND ‘oxido nitroso’ IS ‘normal’ AND ‘monoxido de carbono’ IS ‘normal’ AND ‘amoniaco’ IS ‘normal’ AND ‘benceno’ IS ‘normal’ AND ‘metano’ IS ‘normal’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘sin riesgo’ ALSO ‘intoxicacion’ IS ‘sin riesgo’.
- IF ‘Humedad’ IS ‘muy humedo’ AND ‘Temperatura’ IS ‘muy caliente’ AND ‘oxido nitroso’ IS ‘normal’ AND ‘monoxido de carbono’ IS ‘normal’ AND ‘amoniaco’ IS ‘normal’ AND ‘benceno’ IS ‘normal’ AND ‘metano’ IS ‘normal’ THEN ‘incendio’ IS ‘alerta’ ALSO ‘explosion’ IS ‘sin riesgo’ ALSO ‘intoxicacion’ IS ‘sin riesgo’.

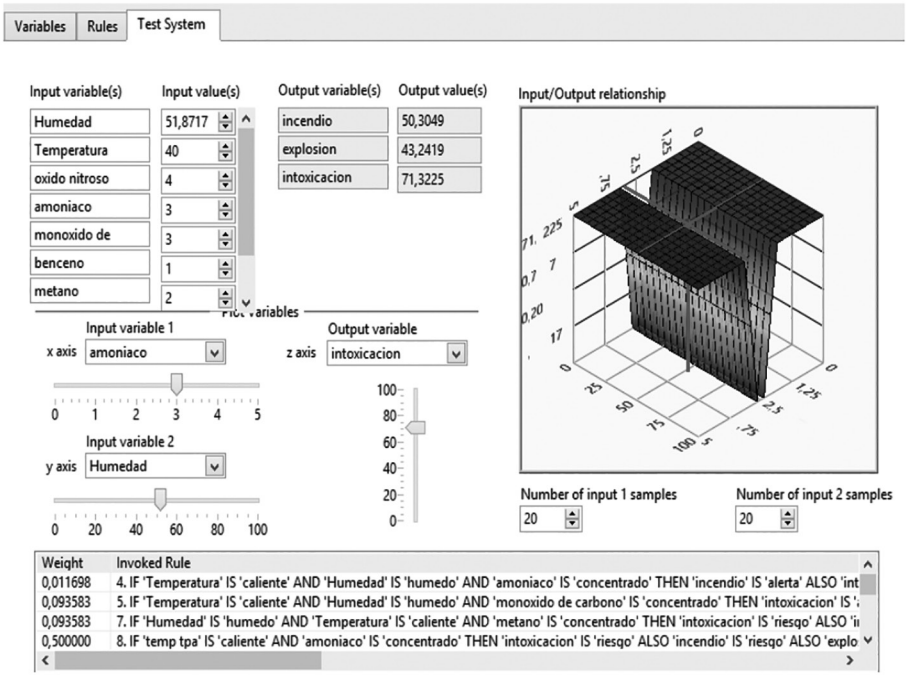
- IF ‘Temperatura’ IS ‘caliente’ AND ‘Humedad’ IS ‘humedo’ AND ‘amoniaco’ IS ‘concentrado’ THEN ‘incendio’ IS ‘alerta’ ALSO ‘intoxicacion’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’.
- IF ‘Temperatura’ IS ‘caliente’ AND ‘Humedad’ IS ‘humedo’ AND ‘monoxido de carbono’ IS ‘concentrado’ THEN ‘intoxicacion’ IS ‘alerta’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘incendio’ IS ‘riesgo’.
- IF ‘Humedad’ IS ‘humedo’ AND ‘Temperatura’ IS ‘caliente’ AND ‘benceno’ IS ‘concentrado’ THEN ‘intoxicacion’ IS ‘alerta’ ALSO ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’.
- IF ‘Humedad’ IS ‘humedo’ AND ‘Temperatura’ IS ‘caliente’ AND ‘metano’ IS ‘concentrado’ THEN ‘intoxicacion’ IS ‘riesgo’ ALSO ‘incendio’ IS ‘alerta’ ALSO ‘explosion’ IS ‘riesgo.
- IF ‘temp tpa’ IS ‘caliente’ AND ‘amoniaco’ IS ‘concentrado’ THEN ‘intoxicacion’ IS ‘riesgo’ ALSO ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘alerta’.
- IF ‘temp tpa’ IS ‘caliente’ AND ‘monoxido de carbono’ IS ‘concentrado’ THEN ‘intoxicacion’ IS ‘alerta’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘incendio’ IS ‘riesgo’.
- IF ‘temp tpa’ IS ‘caliente’ AND ‘benceno’ IS ‘concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘alerta’ ALSO ‘intoxicacion’ IS ‘riesgo’.
- IF ‘temp tpa’ IS ‘caliente’ AND ‘metano’ IS ‘concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘alerta’ ALSO ‘intoxicacion’ IS ‘riesgo’.
- IF ‘oxido nitroso’ IS ‘concentrado’ OR ‘oxido nitroso’ IS ‘muy concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘sin riesgo’ ALSO ‘intoxicacion’ IS ‘alerta’.
- IF ‘amoniaco’ IS ‘concentrado’ OR ‘amoniaco’ IS ‘muy concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘intoxicacion’ IS ‘alerta’.
- IF ‘monoxido de carbono’ IS ‘concentrado’ OR ‘monoxido de carbono’ IS ‘muy concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘intoxicacion’ IS ‘alerta’.
- IF ‘benceno’ IS ‘concentrado’ OR ‘benceno’ IS ‘muy concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘intoxicacion’ IS ‘alerta’.
- IF ‘metano’ IS ‘concentrado’ OR ‘metano’ IS ‘muy concentrado’ THEN ‘incendio’ IS ‘riesgo’ ALSO ‘explosion’ IS ‘riesgo’ ALSO ‘intoxicacion’ IS ‘alerta’.

El sistema difuso se elaboró por medio de la herramienta Fuzzy Sistema Design de LabVIEW, que permite ingresar la entradas, salidas y reglas; y ofrece opciones para el tipo de sistema difuso entre Mandani y Takagui Sugeno y para el método de fuzzificación, entre centro de área, centro de sumas y el máximo.

En la figura 58 se muestra la simulación que entrega la herramienta Fuzzy Sistema Design, donde se pueden ingresar cada uno de los valores de las entradas y se genera un área de decisión que dará la salida. En este caso, se ingresó temperatura

caliente, humedad normal, óxido nitroso muy concentrado, amoniaco y monóxido de carbono concentrado, benceno y metano normal. El sistema da como salida un riesgo de incendio y explosión, y una alerta de intoxicación.

Figura 58. Test System de LabVIEW



Pruebas del sistema de fusión sensorial

Para hacer la evaluación del sistema de fusión sensorial se dispuso de un entorno de 10 m^2, donde se tuvieron los distintos gases tóxicos e inflamables que se miden, una fuente de calor y agua hirviendo para producir vapor y elevar la temperatura ambiente y la humedad relativa. A continuación, se hace una descripción de los elementos usados en la prueba y los gases que se obtienen de ellos.

Cigarrillo

El humo emanado del cigarro libera una gran cantidad de gases tóxicos, además de una gran cantidad de partículas dañinas [61]. En la tabla 14 se muestran los gases contenidos en el humo del cigarrillo tanto exhalado como emanado y su proporción.

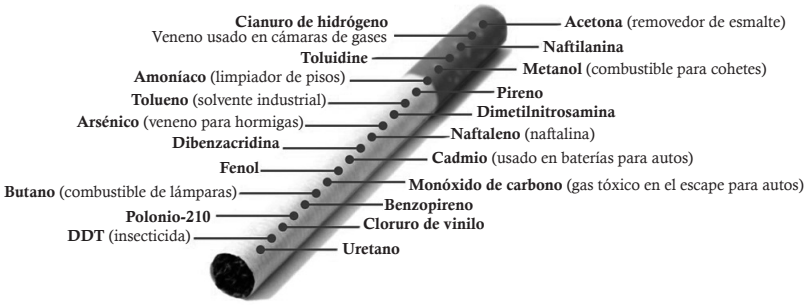
Tabla 14. Composición del humo del cigarrillo

Compuesto	Contenido	Humo exhalado (%)	Humo emanado (%)
Co	10-30 mg	2.5	4.7
Co2	20-40 mg	8	11
Benceno (B)	12-48 g	5	10
Acetona	100-250 g	2	5
Ác. Cianhídrico	400-500 g	0.1	0.25
Amoniaco	50-130 g	40	170
Piridina	16-40 g	6.5	20
N-Nitrosodimetilamina (C)	10-40 ng	20	100

Fuente: [61].

En la figura 59 se muestran otros gases contenidos en el cigarrillo, tanto en el filtro como en el cuerpo.

Figura 59. Componentes del cigarrillo



Fuente: [62].

Madera

El humo que libera la madera al quemarse es una combinación de muchos gases, que pueden ser peligrosos. Las concentraciones exactas de cada gas dependerán del tipo de madera y su estado. La madera seca y acondicionada generalmente produce menor cantidad de humo peligroso y emite la mayor cantidad de calor. El óxido nitroso es un compuesto que se libera al quemar la madera y hace parte de los ácidos que se combinan con el agua en la atmósfera, formando lluvia ácida [63].

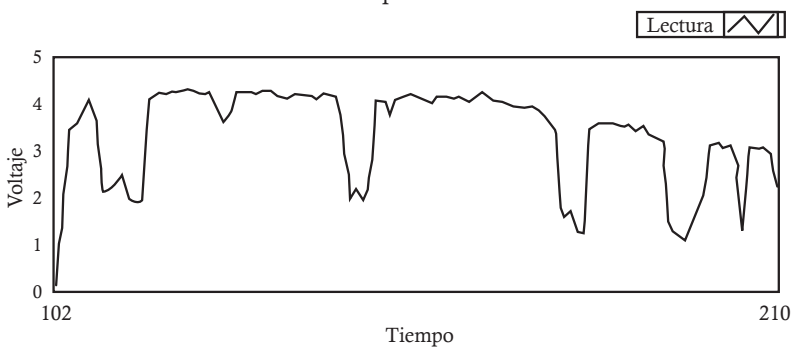
Se hizo una prueba con 10 cigarrillos (humo emanado), en la que se encuentra amoniaco, benceno, metano y monóxido de carbono; un pedazo de madera ardiendo con gasolina para captar el óxido nitroso; un bombillo de 100 W para simular una fuente de calor y agua hirviendo para cambiar la humedad relativa y la temperatura.

Se dejó la plataforma expuesta a las emanaciones en el entorno y se observó la respuesta de cada uno de los sensores y la activación de las alarmas y los riesgos, de acuerdo con las reglas de Fuzzy. A continuación, se muestra la respuesta de cada uno de los sensores durante la prueba.

Sensor MQ135 (óxido nitroso)

En la figura 60 se observa la respuesta del sensor expuesto al humo emanado al quemar carbón con gasolina. De acuerdo con los datos obtenidos, se modificó la entrada del Fuzzy para el óxido nitroso; se redujo el valor para la función de pertinencia normal en 0,5 V, es decir que hasta 1 V se considera que no hay riesgo con la presencia del gas.

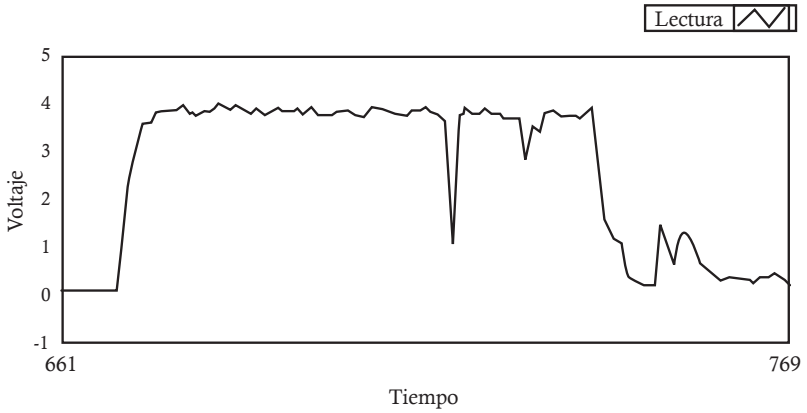
Figura 60. Respuesta del sensor MQ135 al óxido nitroso contenido en el humo emanado por el carbón



Sensor MQ135 (amoniaco)

En la figura 61 se observa la respuesta del sensor expuesto al humo emanado por los cigarrillos. De acuerdo con los datos obtenidos, no se modificó la entrada del amoniaco al Fuzzy; se tiene que valores por debajo de 2 V son normales, entre 2 y 3.5 V es concentrado y por encima de 3.5 V es muy concentrado.

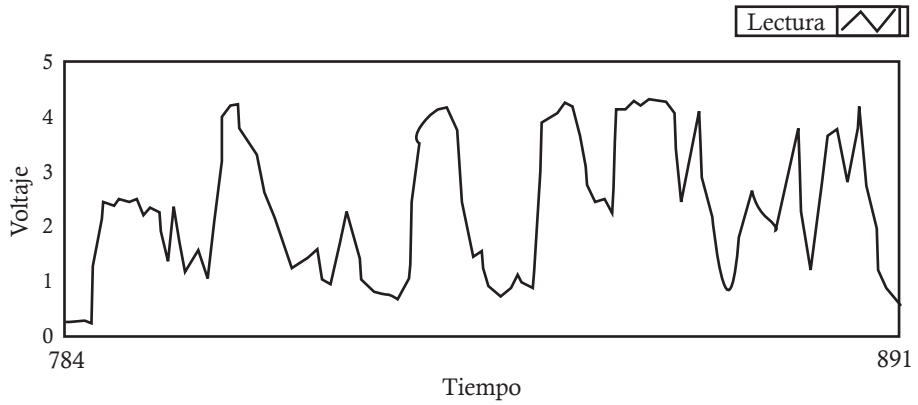
Figura 61. Respuesta del sensor MQ135 al amoniaco contenido en el humo emanado por los cigarrillos



Sensor MQ135 (benceno)

En la figura 62 se observa la respuesta del sensor expuesto al humo emanado por los cigarrillos. De acuerdo con los datos obtenidos, no se modificó la entrada del amoniaco al Fuzzy; se tiene que valores por debajo de 2 V son normales, entre 2 y 3.5 V es concentrado y por encima de 3.5 V es muy concentrado.

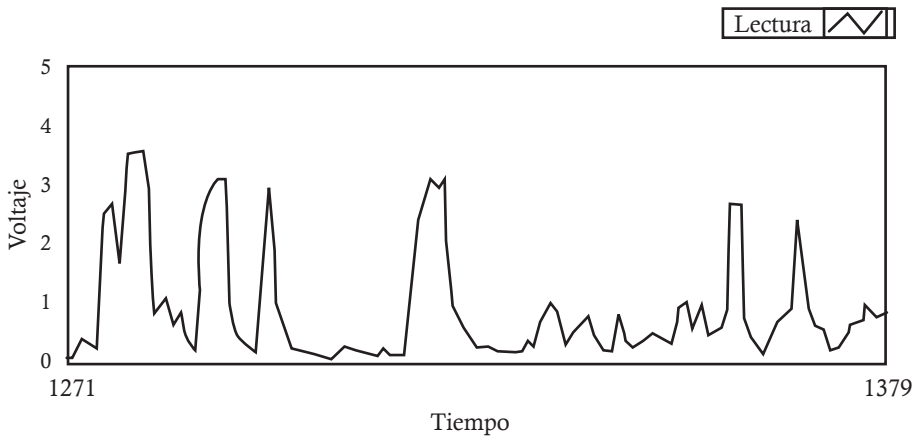
Figura 62. Respuesta sensor MQ135



Sensor TGS3870 (metano y monóxido de carbono)

En la figura 63 se observa la respuesta del sensor expuesto al humo emanado por los cigarrillos y al gas de un encendedor, este último se usó para quemar la gasolina en su interior y generar monóxido de carbono. Los datos por encima de 3 V corresponden al metano y los que están entre 1.5 y 3 V corresponden al monóxido de carbono.

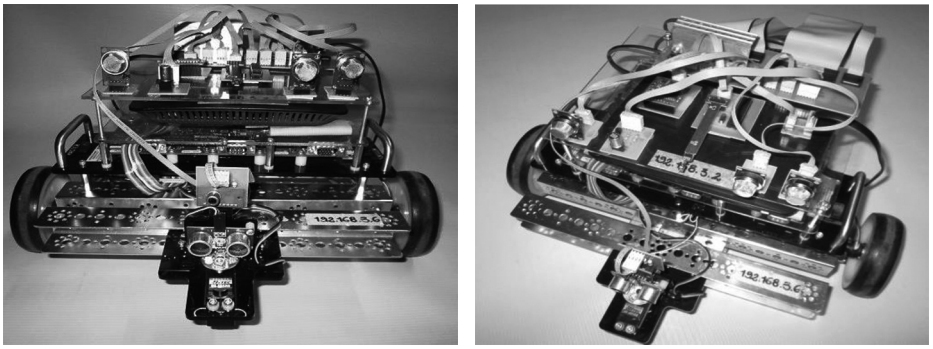
Figura 63. Respuesta del sensor TGS3870 al humo emanado por los cigarrillos



Fotografías

A continuación, se muestra un registro fotográfico de la plataforma, adaptación de los sensores y conectores, y de las pruebas realizadas al sistema, sistema de fusión sensorial y algoritmo de evasión de obstáculos. En la figura 64 se muestra la plataforma robótica DaNI 2.0 con la adaptación de los sensores y el *router* para la comunicación con el PC.

Figura 64. Plataforma robótica DaNI 2.0



En las figuras 65 y 66 se muestran las pruebas realizadas al sistema de fusión sensorial, exponiendo los sensores a gases tóxicos e inflamables, a fuentes de calor y cambios de humedad y temperatura.

Figura 65. Exposición a cigarrillo y carbón

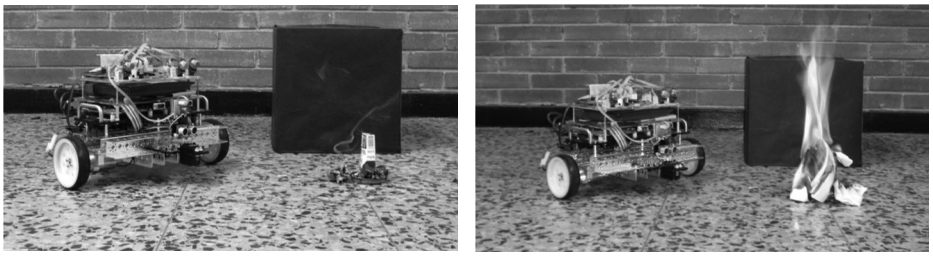
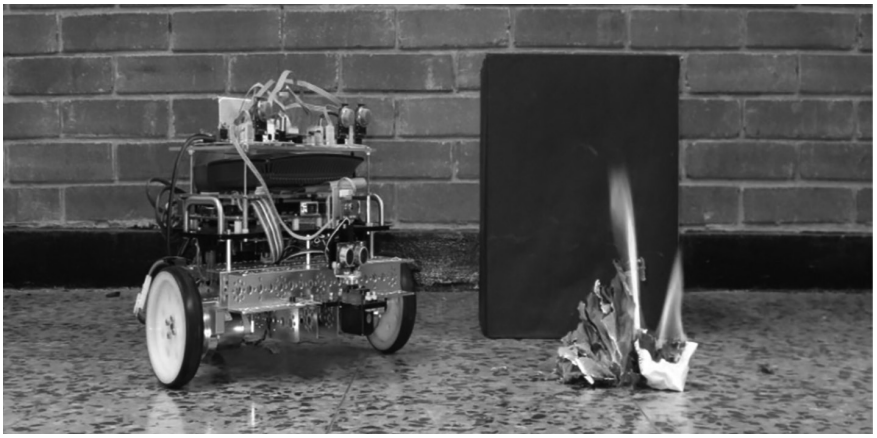
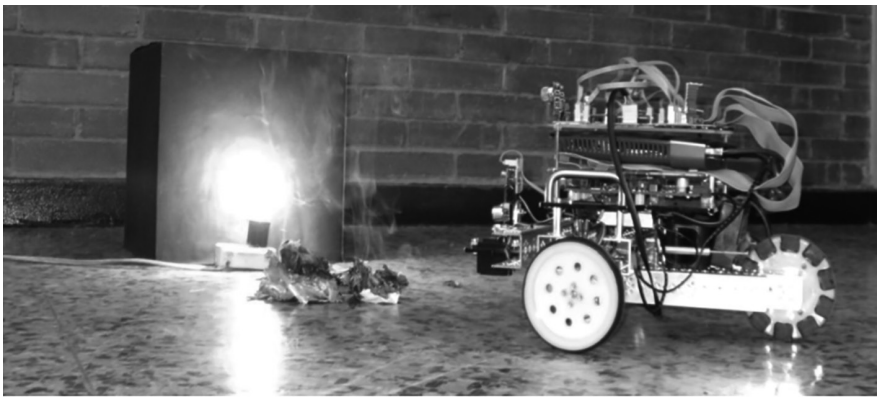


Figura 66. Exposición a fuentes de calor y cambios de temperatura y humedad relativa





Navegación del robot por filtros de Kalman

La navegación del robot móvil permite conocer su ubicación en la zona de exploración y el desarrollo de planificación de tareas para alcanzar el objetivo planteado. En nuestro caso, la exploración de su entorno permitiría detectar víctimas, condiciones de la zona de emergencia y apoyar a los grupos de rescate con información de localización necesaria y condiciones ambientales, entre otras.

Dentro de los sistemas sensoriales con que cuenta la plataforma, se desarrolló el sistema de posicionamiento a partir de la información que registran *encoders,* brújula magnética y giroscopio. Este sistema de posicionamiento cuenta con relaciones matemáticas establecidas a partir de modelos cinemáticos y odométricos que permiten conocer la distancia recorrida por la plataforma y su posición en todo momento.

Dado que estos modelos presentan errores acumulativos ocasionados, principalmente por características sistémicas dadas por la misma plataforma, es necesario integrar los modelos matemáticos y transformarlos en una aproximación lineal que permita estimar parámetros de desplazamiento de la plataforma mediante la implementación de un filtro de Kalman.

Determinación de parámetros para el filtro de Kalman

Modelo cinemático

El desarrollo de un modelo cinemático permite conocer la posición y la orientación del móvil a partir de sus parámetros físicos, que, por medio de expresiones matemáticas, faciliten conocer el comportamiento del movimiento que posibilite la estimación de posiciones que puedan ser planificadas [66].

Para la plataforma usada, el desplazamiento total y su orientación están dados por las ecuaciones (4.1) y (4.2), respectivamente, donde se presentan expresiones generales obtenidas a partir de las características de la plataforma, donde Xm_L y θ_L corresponden al desplazamiento y la orientación de la rueda izquierda, Xm_R y θ_R corresponden al desplazamiento y la orientación de la rueda derecha y *b* corresponde al eje de tracción de la plataforma.

$$\Delta D = \frac{Xm_L - Xm_R}{2} = 0.0244\left(\theta_L + \theta_R\right) \tag{4.1}$$

$$\Delta\theta = \frac{Xm_L - Xm_R}{b} = 0.2856\left(\theta_L + \theta_R\right) \tag{4.2}$$

Modelo odométrico

El modelo odométrico permite estimar el movimiento registrado a partir de un planificador de tareas por medio del modelo cinemático que sea obtenido de sus parámetros constructivos. En este sentido, el modelo odométrico utilizado presenta un comportamiento a partir de la ecuación (4.3).

$$X(k+1) = f(X(k)) + v(k) \tag{4.3}$$

Donde X(K+1) es la posición estimada, X(K) es la posición actual, U(K) es la entrada del sistema, V(K) es el vector de errores sistemáticos y no sistemáticos que pueden ser asociados a la plataforma móvil y K es el índice de tiempo de la iteración.

La entrada del sistema U(k) se encuentra determinada por la ecuación (4.4), donde ΔD(k) es la distancia recorrida por la plataforma en un intervalo (t_K, t_{K+1}) y Δθ(k) es la variación de la orientación en el mismo intervalo.

$$U(k) = [\Delta D(k)\ \Delta\theta(k)]^T \tag{4.3}$$

La caracterización del vector de errores de estado V(K) para la plataforma móvil es asumido como V(K)≈ N(0 , Q(K)), suponiendo que no hay ninguna entrada para el sistema, donde N(a,b) es la distribución normal con media a y una desviación estándar b [65], Q(K) es el error característico de estado de la plataforma que no es colineal y se encuentra dado por la ecuación (4.5).

$$Q(k) = \begin{bmatrix} k_D\left(D(k)\cos\theta(k)\right) & 0 & 0 \\ 0 & k_D\left(D(k)\sin\theta(k)\right) & 0 \\ 0 & 0 & k_{D\theta}\left(D(k)\right) + k_\theta\left(\theta(k)\right) \end{bmatrix} \tag{4.5}$$

Siendo K_D el coeficiente de error de traslación de la plataforma relativo a ΔD y expresado en [m^2 / m]; $K_{D\theta}$ el coeficiente de error de rotación de la plataforma relativo a ΔD y expresado en [rad^2/m]; y el K_θ coeficiente de error de rotación de la plataforma relativo a Δθ y expresado en [rad^2/rad].

Teniendo en cuenta los procesos de experimentación realizados para los movimientos individuales y a partir de una serie de aproximaciones lineales, como mínimos cuadrados, se pudieron determinar que los 3 coeficientes con mejores resultados fueron: $K_D = 0.5$, $K_{D\theta} = 0.08$, $K_\theta = 0.3$.

Con el modelo cinemático y el modelo odométrico vectorial mostrado en la ecuación (4.6) se realizó la implementación del filtro de Kalman.

$$\begin{bmatrix} X(k+1) \\ Y(k+1) \\ \theta(k+1) \end{bmatrix} = \begin{bmatrix} x(k) + D(k)\cos\left(\theta(k) + \frac{\theta(k)}{2}\right) \\ y(k) + D(k)\sin\left(\theta(k) + \frac{\theta(k)}{2}\right) \\ \theta(k) + \frac{\theta(k)}{2} \end{bmatrix} + v(k) \tag{4.6}$$

Fases del filtro de Kalman

El filtro de Kalman calcula el estado del proceso en un instante y obtiene información (se realimenta) de la medida; las ecuaciones del filtro se presentan como actualización del tiempo (predicción) y actualización de las medidas (corrección) [64].

Fase de predicción

En esta fase de filtro de Kalman se calcula la estimación, es decir, la posición en que se encontrará la plataforma según el modelo cinemático y odométrico. Esta estimación se expresa usando su matriz característica jacobiana para el sistema, de la siguiente forma: donde *A* representa el estado actual del sistema y *B*, las entradas al sistema.

$$fx = X_{k+1} = X_k + \Delta D\cos\left(\theta + \frac{\Delta\theta}{2}\right)$$

$$f\theta = \theta_{k+1} = \theta_k + \frac{\Delta\theta}{2}$$

$$A = \begin{bmatrix} 1 & 0 & -\Delta D\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 1 & \Delta D\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$fx = X_{k+1} = X_k + \left(\Delta D + \varepsilon D\right)\cos\left(\theta + \left(\frac{\Delta\theta}{2} + \varepsilon\Delta\theta\right)\right)$$

$$fx = Y_{k+1} = X_y + \left(\Delta D + \varepsilon D\right)\sin\left(\theta + \left(\frac{\Delta\theta}{2} + \varepsilon\Delta\theta\right)\right)$$

$$f\theta = \theta_{k+1} = \theta_k + \left(\frac{\Delta\theta}{2} + \varepsilon\Delta\theta\right)$$

$$B = \begin{bmatrix} \cos\left(\theta + \frac{\Delta\theta}{2}\right) & -\Delta D\sin\left(\theta + \frac{\Delta\theta}{2}\right) \\ \sin\left(\theta + \frac{\Delta\theta}{2}\right) & \Delta D\cos\left(\theta + \frac{\Delta\theta}{2}\right) \\ 0 & 0 \end{bmatrix} \tag{4.8}$$

Estas son las ecuaciones del filtro de Kalman que se van a aplicar para calcular la posición de la plataforma móvil que incorpora el modelo odométrico, que da los incrementos en la posición del punto central del móvil respecto de unos ejes de referencia fijos; dichos incrementos se toman como entradas directas al sistema [65].

Fase de estimación

En esta parte del filtro, se corrige la posición de la plataforma con la información que se recibe de los sensores, y considerando que se mide directamente la posición y orientación del robot, se tiene H que es el jacobiano en función de la medida, y se expresa así:

$$\begin{aligned} H_\theta &= \theta \\ Hy &= y \\ Hx &= x \end{aligned} \tag{4.9}$$

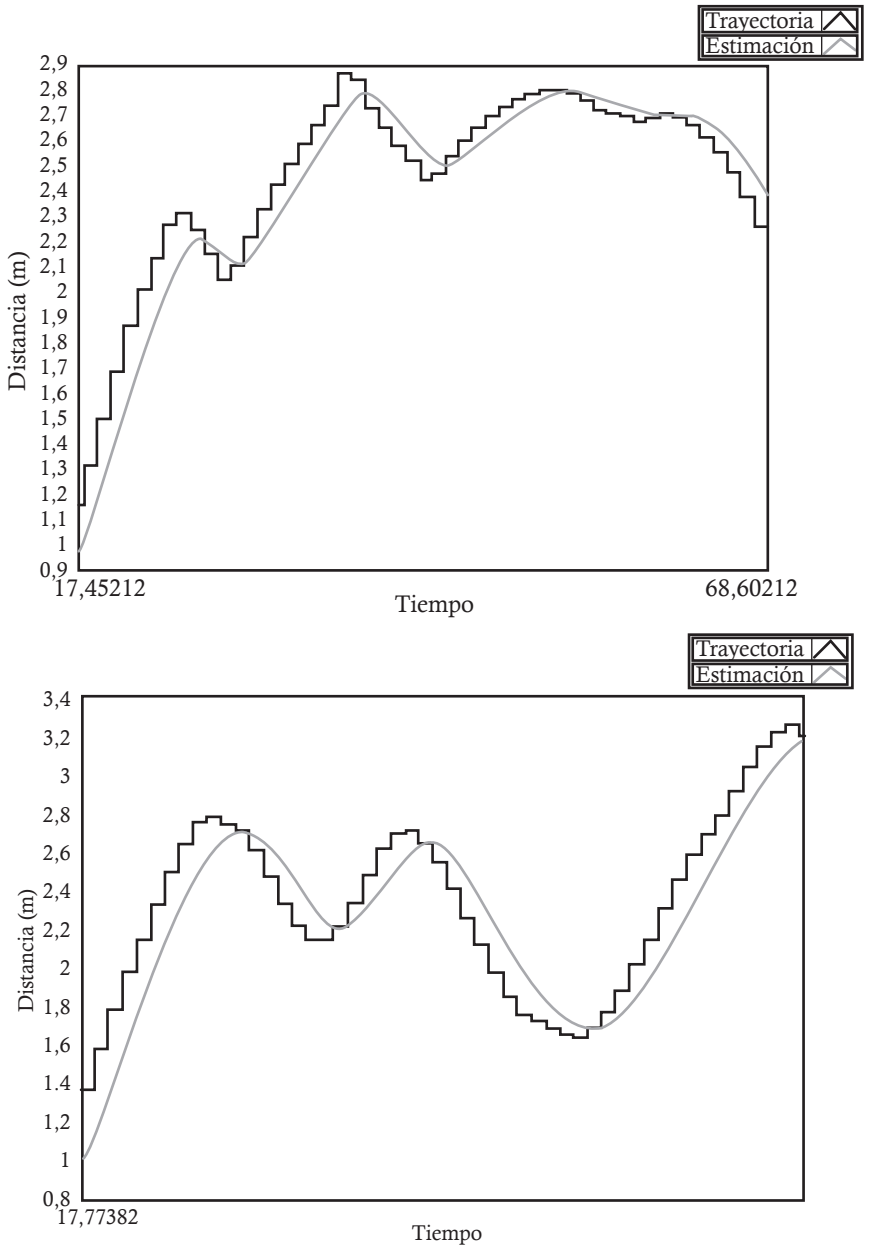
$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

Implementación del filtro

El filtro de Kalman se implementó en LabVIEW con la herramienta de control, diseño y simulación; el bloque tiene como entrada las matrices A, B y H que se hallaron, la distancia recorrida por la plataforma y la lectura de los sensores, *encoders*, giroscopio y brújula magnética.

En la figura 67 se muestran las respuestas del filtro durante la navegación del robot; la línea negra es la trayectoria del robot, que se ve como escalones debido a los valores entregados por el giroscopio, y la línea gris es la estimación que hace el filtro; de aquí se toma la lectura de la distancia recorrida y se obtienen las coordenadas X y Y del robot.

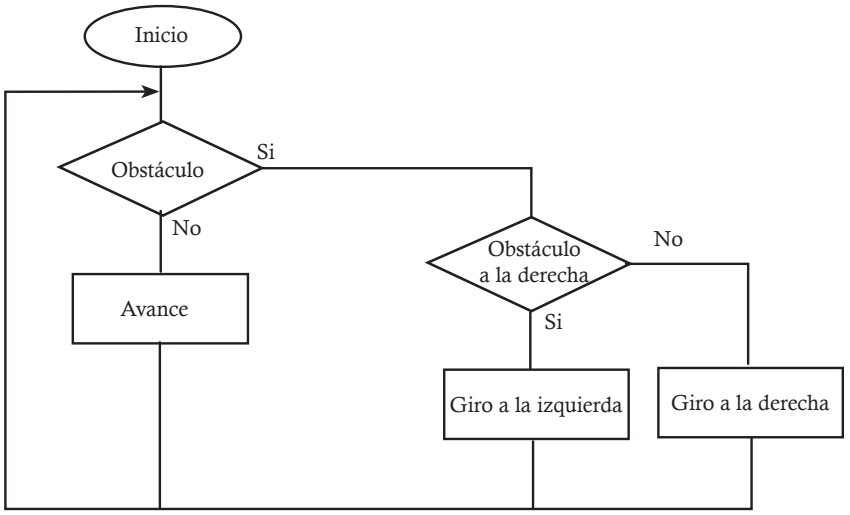
Figura 67. Respuesta del filtro de Kalman



Algoritmo de evasión de obstáculos

Dentro del proceso de navegación implementado, se debe garantizar la evasión de todo tipo de obstáculo y, para ello, fue desarrollado un algoritmo que basa su funcionamiento en la lectura de un radar construido basado en el sensor de ultrasonido. En la figura 68 se presenta el diagrama de flujo del algoritmo implementado; el robot avanza en todo momento y cuando detecta un obstáculo a 35 cm o menos, la plataforma gira hacia el lado donde encuentre menor distancia a otro objeto, ya que el sensor se encuentra montado sobre un servomotor de la plataforma robótica DaNI.

Figura 68. Algoritmo de evasión de obstáculos



Para comprobar el funcionamiento del algoritmo, se desarrollaron varias pruebas en las cuales el espacio de configuraciones se conformó a partir del uso de 3 obstáculos (2 cubos y un cilindro), con 3 distribuciones aleatorias, lo que permitió observar el desempeño del algoritmo implementado sobre la plataforma.

Primera prueba

Esta prueba inició ubicando la plataforma frente al cilindro con el objetivo de alcanzar la pared y realizar un retorno. Con esta distribución de obstáculos, la plataforma realiza un primer giro hacia la derecha y lo rodea hasta regresar al punto inicial por la izquierda. El robot no chocó, ni tocó ninguno de los objetos en el entorno. La duración de esta prueba fue de 65 s.

En la figura 69 se muestra la distribución de los obstáculos en el espacio de configuración distribuido de forma aleatoria, y en la figura 70 se muestran la trayectoria realizada por el robot partiendo de la posición (0,0) y la respuesta del filtro de Kalman para la estimación de la posición que es tomada como verdadera para la trayectoria realizada por el robot.

Figura 69. Distribución de obstáculos-prueba 1

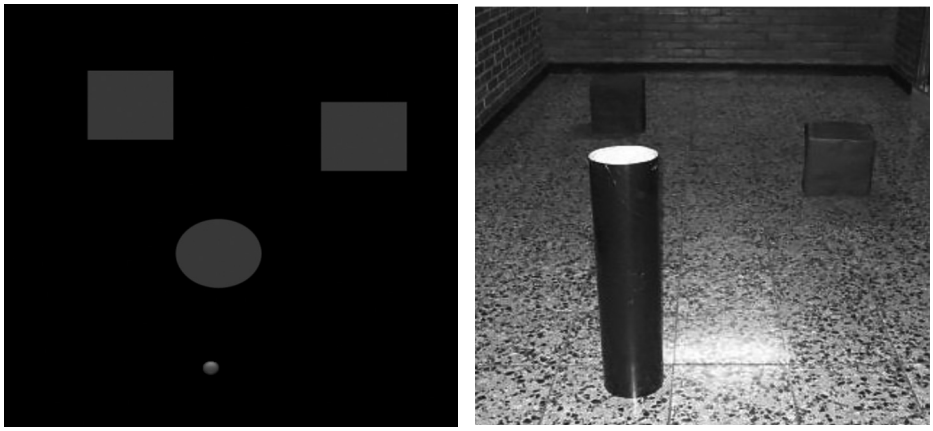
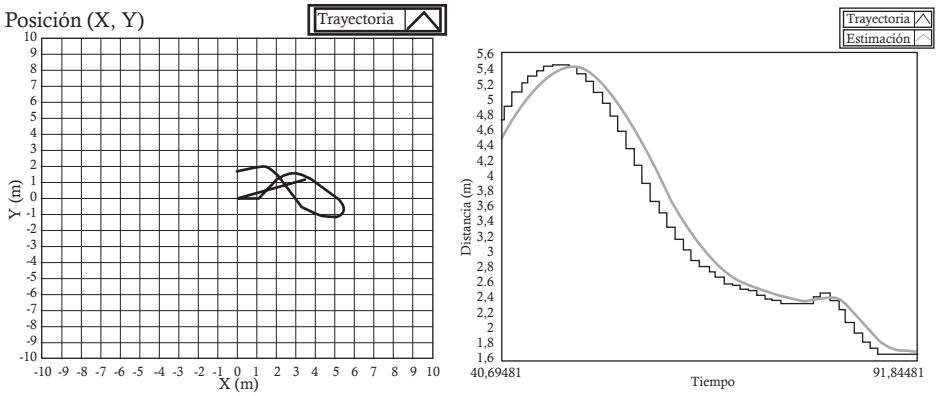


Figura 70. Trayectoria realizada por la plataforma para la primera prueba; estimación de la posición realizada por el filtro de Kalman



Segunda prueba

Esta prueba inició colocando la plataforma frente al cuadrado para que llegara hasta la pared y regresara. Con esta distribución de obstáculos, la plataforma toma la izquierda y sigue derecho entre los obstáculos hasta encontrar la pared, donde gira y retrocede para regresar. Durante el recorrido realizado por la plataforma, choca debido a que no hay detección por su parte posterior. La duración de esta prueba fue de 75 s.

En la figura 71 se muestra la distribución de los obstáculos en el espacio de configuración distribuido de forma aleatoria y en la figura 72 se muestran la trayectoria realizada por el robot partiendo de la posición (0,0) y la respuesta del filtro de Kalman para la estimación de la posición que es tomada como verdadera para la trayectoria realizada por el robot.

Figura 71. Distribución de los obstáculos-prueba 2

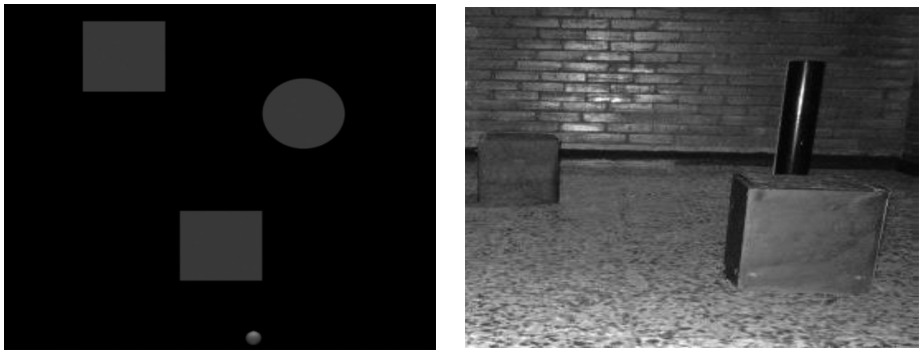
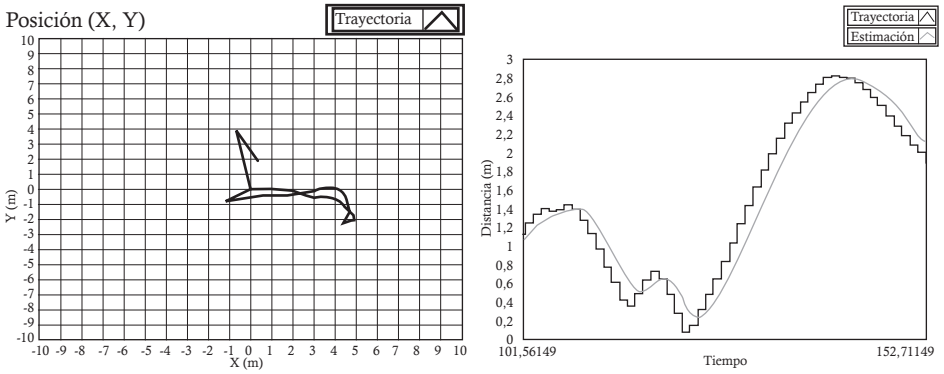


Figura 72. Prueba 2. Izquierda: trayectoria de la plataforma.
Derecha: estimación de la posición



En esta segunda prueba, la plataforma chocó mientras hacía un giro y, por ende, en la figura 72, en la gráfica de estimación del filtro de Kalman, se puede observar cómo la trayectoria se aleja de la trayectoria estimada, lo que fue ocasionado, principalmente, por errores sistemáticos y no sistemáticos presentes en el momento del choque.

Tercera prueba

Esta prueba inició colocando la plataforma de frente al cilindro y uno de los cubos para que los atravesara y llegara hasta la pared y regresara. Durante el recorrido de regreso a la posición inicial, la rueda derecha chocó con un cubo y la plataforma empuja el objeto hasta que logra seguir hacia adelante. La duración de esta prueba fue de 100 s.

En la figura 73 se muestra la distribución de los obstáculos en el espacio de configuración, distribuido de forma aleatoria, y en la figura 74 se muestran la trayectoria realizada por el robot partiendo de la posición (0, 0) y la respuesta del filtro de

Kalman para la estimación de la posición que es tomada como verdadera para la trayectoria realizada por el robot.

Figura 73. Distribución de obstáculos-prueba 3

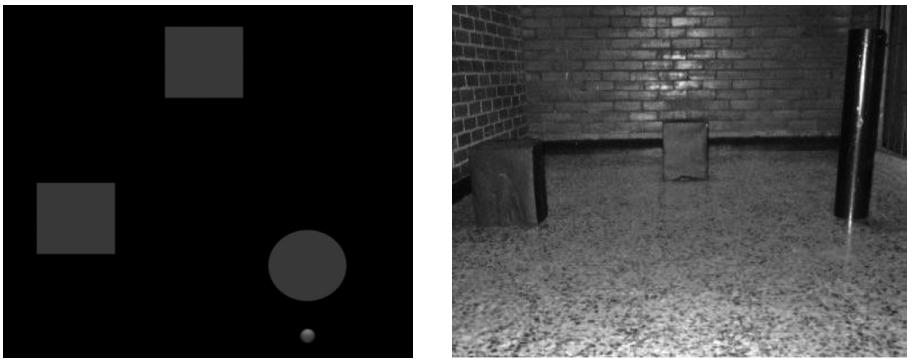
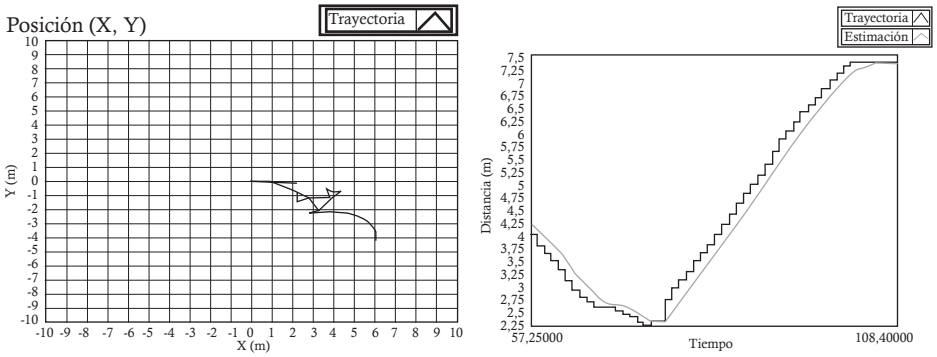
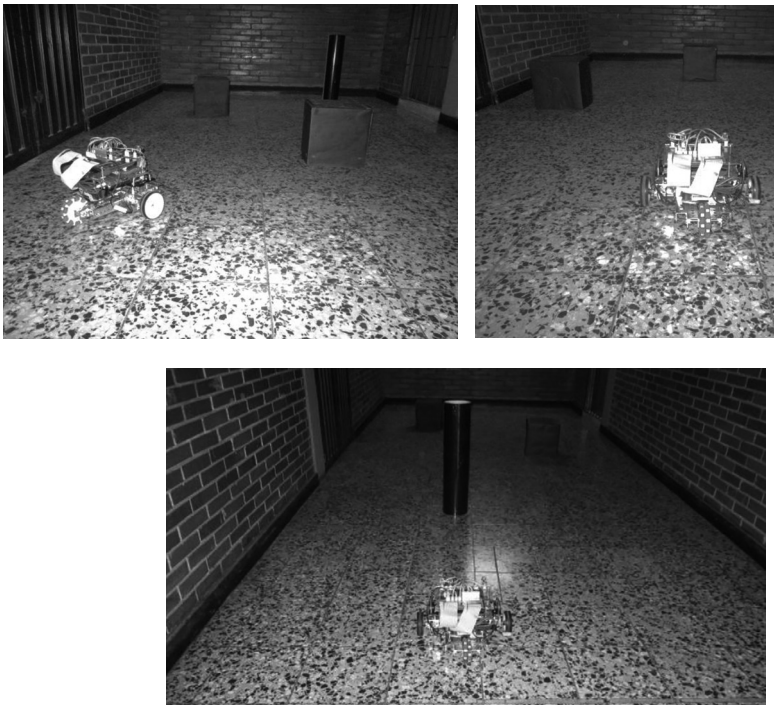


Figura 74. Prueba 3. Izquierda: trayectoria de la plataforma. Derecha: estimación de la posición



En la figura 75 se muestran algunas fotos de las pruebas realizadas al algoritmo de evasión de obstáculos implementado, y se puede observar la distribución aleatoria de los objetos usados.

Figura 75. Pruebas al algoritmo de evasión de obstáculos



Visión artificial

Es de vital importancia para el desarrollo del proyecto incluir algoritmos basados en visión artificial que permitan la detección y la evasión de obstáculos de modo que no afecte las plataformas robóticas. Al tomar como base que en el corazón de la visión artificial la primera parte radica en los dispositivos de captura de imágenes, se plantean 2 opciones, una cámara USB marca Logitech; donde se muestra un algoritmo para la detección de obstáculos basado en segmentación de imagen, y el segundo muestra la implementación de un Kinect de Microsoft para la captura de imágenes de profundidad y la detección de marcas naturales por medio de estas.

Detección de obstáculos basado en cámara web

La implementación de las cámaras en el campo de la robótica es cada día más común debido a su amplia versatilidad como sensor para el robot por medio de la visión artificial. Por su parte, la detección de obstáculos es uno de los temas clave a la hora de trabajar con la planeación de rutas y movimientos en el robot; por tal motivo, se han desarrollado una gran cantidad de algoritmos que permiten la identificación de obstáculos mediante el procesamiento de imágenes. Por esta razón, es fundamental realizar una captura de imágenes por medio de un dispositivo que permita desempeñar esta función; al tratarse naturalmente de algoritmos de visión artificial, dichos dispositivos consisten en cámaras ubicadas en las plataformas robóticas y que proporcionan un buen rango de visión dependiendo de los requerimientos en cada aplicación.

En el mercado actual se puede conseguir un amplio número de cámaras USB de diferentes resoluciones y que se adapten a las necesidades de cada problema en particular; de esta manera, se encuentran las cámaras HD, las cuales poseen como característica principal una adaptación a los cambios repentinos de luz en un entorno. Para el desarrollo de este sistema se implementó una cámara USB de referencia Logitech Quick Cam 9000 pro (figura 76), una cámara HD que, gracias a su buena resolución, adaptación a los cambios repentinos de luz, autoenfoque, entre otras características, fue utilizada como sensor primario para la obtención de imágenes utilizadas en los algoritmos de visión artificial.

Algunas de las especificaciones técnicas son descritas a continuación:

- Marca: Logitech.
- Modelo: QuickCam Pro 9000.
- Resolución real del sensor: 2.0 Mpx.
- Resolución máxima: 8.0 Mpx por *software.*
- Resolución de video: hasta 1600*1200 píxeles.
- Cuadros por segundo: 30 FPS.
- Interfaz: USB 2.0.

Dentro de sus características especiales, se encuentran:

- Sensor nativo HD de 2.0 Mpx.
- Lente Carl Zeiss con autoenfoque para una imagen más nítida y rica en colores.
- Micrófono integrado con tecnología RightSound de Logitech.
- Clip de montaje universal para *laptops*, *netbooks*/*notebooks* y monitores LCD y CRT.
- *Software* incluido para crear divertidas animaciones y efectos.

Figura 76. Logitech QuickCam 9000 pro

Fuente: [67].

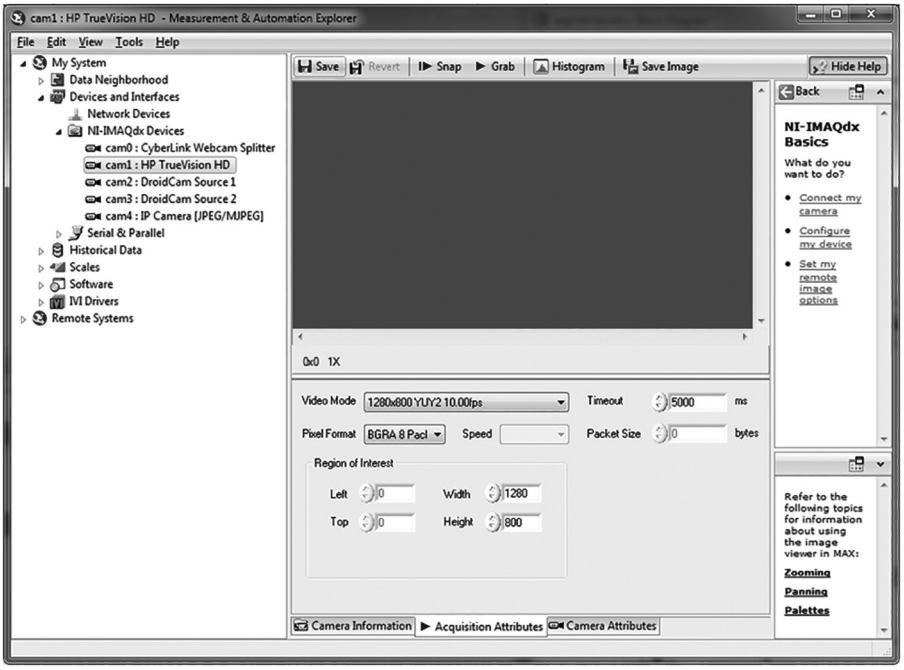
Adquisición de imágenes en LabVIEW

La etapa de adquisición y visualización de imágenes se lleva a cabo gracias al módulo NI-IMAQdx, instalado en el programa LabVIEW de National Instruments,

y permite administrar el dispositivo para cambiar formatos de imágenes, calidad y resolución de estas, entre otras características.

En primer lugar, el dispositivo o cámara debe ser compatible con LabVIEW y reconocido por el programa Measurement & Automation Explorer, en la pestaña Devices and Interfaces, en el submenú NI-IMAQdx devices (figura 77). Una vez allí, se debe seleccionar el dispositivo por utilizar, y en la parte inferior central del programa se pueden configurar los atributos de adquisición para esa cámara en particular.

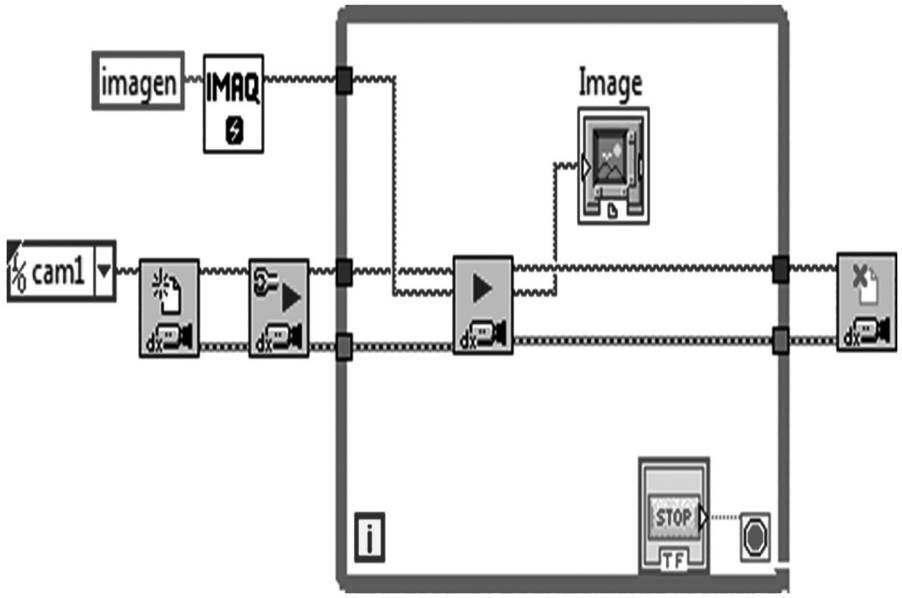
Figura 77. Configuración de atributos de adquisición



El algoritmo implementado en LabVIEW para la captura y visualización de las imágenes adquiridas por la cámara seleccionada, mediante las funciones del módulo NI-IMAQdx, se muestra en la figura 78.

Se observa, de esta manera, que el procedimiento general consiste en inicializar la cámara con los ajustes previamente establecidos en el NI MAX; crear un *buffer,* en el cual serán almacenadas temporalmente las imágenes capturadas por la cámara; en un bucle se configura el tipo de adquisición (continua o ráfaga de imágenes) y, por último, se finaliza la cámara para liberar espacio en memoria. Por lo tanto, debe seguirse este procedimiento básico para la adquisición de imágenes en LabVIEW y añadir un visualizador de imágenes en el panel frontal del programa.

Figura 78. Algoritmo para la adquisición de imágenes



Algoritmo segmentación de imágenes

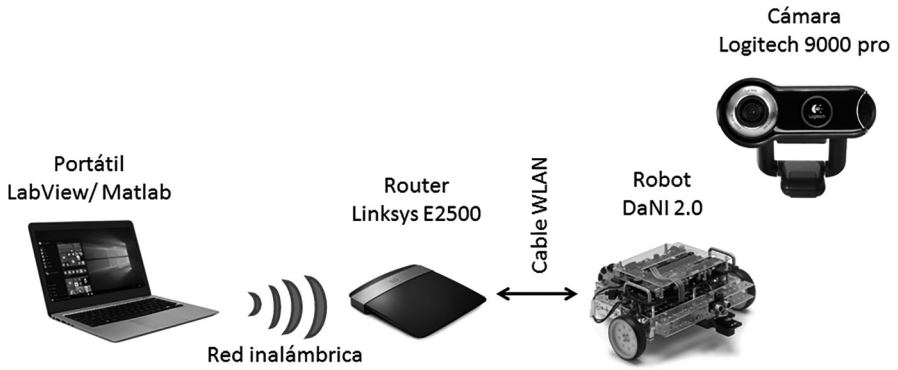
El algoritmo desarrollado se basa en un método denominado "segmentación de imágenes" y consiste en capturar imágenes por medio de una cámara, realizar un proceso de filtrado y binarización, y posteriormente, tomar partes de cada imagen (segmentos) y analizarlas por separado para de esta manera determinar los movimientos o las trayectorias que pueda tomar el robot. Las salidas del programa corresponden a las velocidades para cada motor de la plataforma robótica, que son enviadas mediante una comunicación wifi y que permiten el desplazamiento del robot en diferentes direcciones con el fin de evitar colisionar con los obstáculos. Por su parte, otro algoritmo descargado en la tarjeta Single-Board RIO de la plataforma robótica permite la recepción de los datos enviados para posteriormente tomar la información de las velocidades para cada uno de sus motores; de esta manera, se logra el movimiento del robot y la evasión de obstáculos con cámara a bordo, el diagrama general del sistema se puede ver en la figura 79.

Debido al gran número de aplicaciones desarrolladas por medio de la teoría de procesamiento de imágenes, se usa un *software* especializado que permita el análisis de dichas imágenes para posteriormente convertirlas en datos que puedan ser usados para algún fin en específico. Actualmente, 2 de los programas más comunes y de gran potencial para el desarrollo de aplicaciones de visión artificial son Matlab y LabVIEW, los cuales poseen librerías ya predefinidas para facilitar la programación en dichos entornos.

Matlab, acrónimo en inglés de Matrix Laboratory, es un sistema interactivo de programación para realizar cálculos numéricos con vectores y matrices. Muchos de los

programas que vienen con el sistema son funciones "internas" (*built-in functions*), diseñadas para resolver problemas generales, y otros conforman librerías especializadas (*toolboxes*) para resolver problemas más concretos. Una ventaja de Matlab es la sencillez de su lenguaje de programación: muchos programas que resultan difíciles de implementar en lenguajes como C, Fortran, etc., se implementan con relativa facilidad en Matlab [69].

Figura 79. Diagrama general del sistema implementado



Fuente: [68].

La principal característica de Matlab es que permite realizar operaciones matriciales de manera rápida y efectiva. Partiendo de la base que las imágenes al ser adquiridas o procesadas son matrices, donde cada elemento (imágenes monocromáticas) o una serie de elementos (imágenes policromáticas) constituyen un pixel, se puede comprender la importancia de un *software* como Matlab en el campo de la visión artificial; por este motivo, cuenta con grandes librerías desarrolladas y perfeccionadas para realizar aplicaciones que demandan un análisis de las imágenes pixel a pixel por medio de comandos de texto.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) es un entorno de desarrollo basado en programación gráfica denominado "Lenguaje G". Utiliza símbolos gráficos en lugar de lenguaje textual para describir acciones de programación [70].

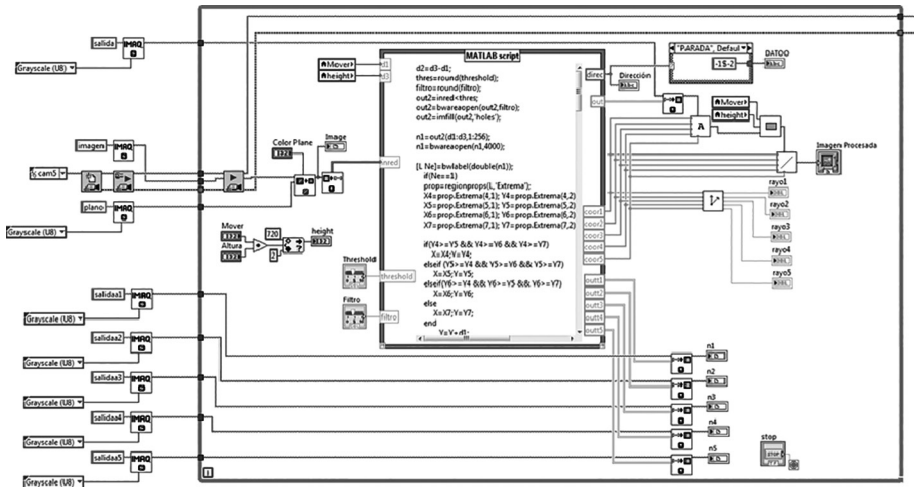
Dentro de sus principales ventajas, se encuentra que permite la ejecución de tareas en tiempo real, lo cual facilita el procesamiento de las imágenes y la toma de decisiones para determinar los movimientos que deba realizar el robot. Otra de las ventajas más importantes que tiene este lenguaje de programación es que permite una fácil integración con *hardware*, específicamente con tarjetas de medición, adquisición y procesamiento de datos, incluyendo adquisición de imágenes y sistemas embebidos, presentes en las plataformas robóticas implementadas.

Para aprovechar las características principales de cada lenguaje de programación, en el sistema se hizo una fusión entre estos 2 lenguajes, para así beneficiarse de las

ventajas de cada uno. Actualmente, existe una función dentro del LabVIEW, la cual permite dicha fusión, esta se llama Matlab Script Node, y permite invocar las funciones del Matlab por medio de una ventana virtual creada por LabVIEW, a través de un ActiveX; por esta razón, la función puede ser usada únicamente en sistemas operativos Windows. Adicionalmente, se requiere tener instalada una versión licenciada de Matlab igual o superior a la versión 6.5.

Esencialmente, el Matlab Script Node está basado en líneas de texto que contienen comandos propios de Matlab y se ejecutan de forma serial de la misma manera que se ejecutaría en el programa como tal. Adicionalmente, se pueden agregar entradas y salidas a este bloque dependiendo del uso que se requiera. Dichas entradas o salidas deben estar acorde a los tipos de datos predefinidos por la función y que son descritos en la tabla 15.

Figura 80. Matlab Script Node



De esta manera, se hizo uso de esta función tomando como entrada la imagen en escala de grises convertida en array de 8 bits para ser binarizada, filtrada y posteriormente analizada. Asimismo, como salidas del bloque, se tienen la imagen binarizada, la cual es un 2-D *array of real* que se utiliza posteriormente para ser visualizada, y la dirección que debe tomar el robot, que es un dato tipo *String* que se envía por medio de un protocolo TCP/IP hacia el robot.

Tabla 15. Tipos de datos del Matlab Script Node

LabVIEW Data Type	Matlab® Script Node Data Type	MathScript Node Data Type
DBL Double-precision, floating-point numeric	Real	Scalar»DBL

LabVIEW Data Type	Matlab® Script Node Data Type	MathScript Node Data Type
CDB Complex double-precision, floating-point numeric	Complex	Scalar»CDB
DBL 1D array of double-precision, floating-point numeric	1-D Array of Real	1D-Array»DBL 1D
CDB 1D array of complex double-precision, floating-point numeric	1-D Array of Complex	1D-Array»CDB 1D
DBL Multidimensional array of double-precision, floating-point numeric	2-D Array of Real	Matrix»Real Matrix (2D only)
CDB Multidimensional array of complex double-precision, floating-point numeric	2-D Array of Complex	Matrix»Complex Matrix (2D only)
abc String	String	Scalar»String
Path	Path	N/A
[abc 1D array of string	N/A	1D-Array»String 1D

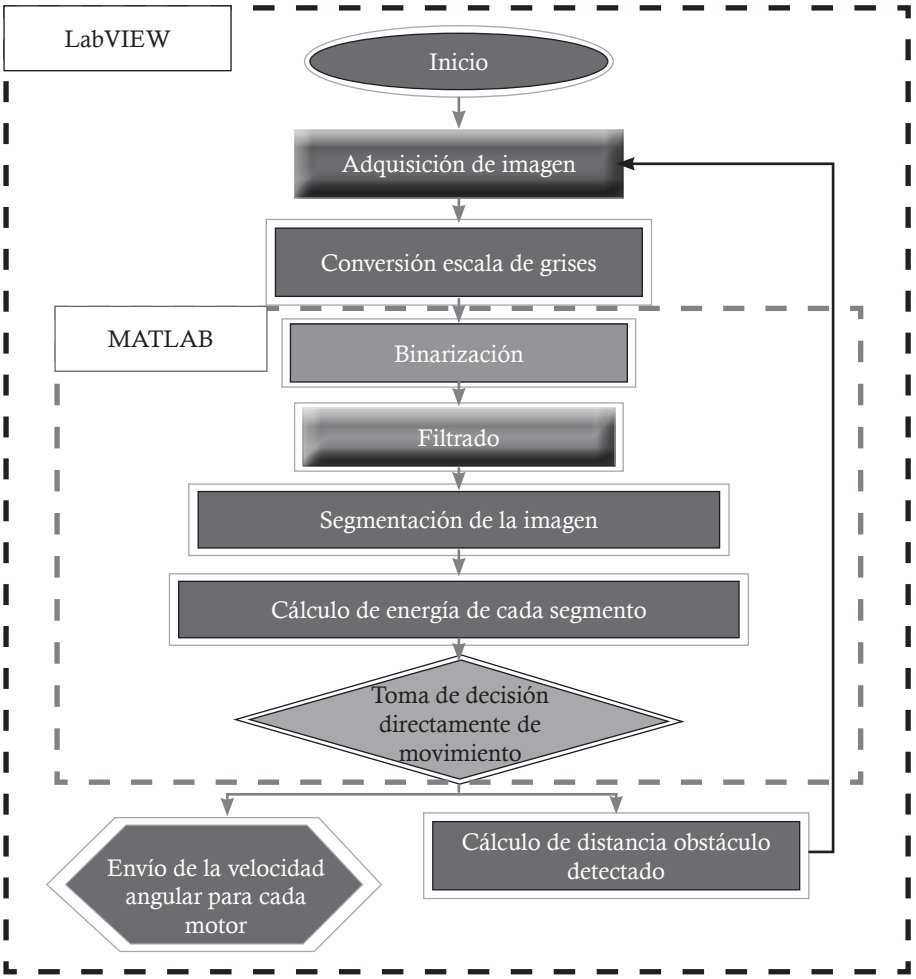
Fuente: [70].

El algoritmo de segmentación de imágenes se dividió principalmente en 2 partes; en este caso, el algoritmo del maestro se refiere al algoritmo desarrollado en el computador y que permite realizar la adquisición, el procesamiento y el análisis de las imágenes, así como el envío de datos hacia la plataforma robótica , y el algoritmo del esclavo, que es el algoritmo desarrollado en la plataforma robótica DaNI 2.0 que consiste fundamentalmente en un protocolo basado en TCP/IP llamado "Network Streams" el cual permite el intercambio de datos entre el computador y el robot a través de LabVIEW.

La figura 81 muestra el algoritmo implementado en el PC (maestro); se ve claramente la parte implementada en LabVIEW y la parte implementada en Matlab. Lo primero que hace el algoritmo es capturar una foto de la cámara web, el tamaño de esta foto es de (1280X720X3) a 5 fotogramas por segundo, y el formato es YUY2, propio de Microsoft. El siguiente paso es convertir las 3 matrices en una sola, por lo cual, después de esto, se obtiene la imagen en escala de grises (figura 83a); después de obtener la imagen a escala de grises, los pasos siguientes se hacen en Matlab, la binarización es un proceso por el cual una matriz que contiene elementos con valores de 0 a 255 se convierte en una matriz que solo posee elementos de 0 a 1, como lo muestra la figura 83b; este procedimiento se hace por medio de un valor umbral que puede ser calculado automáticamente por Matlab o manualmente por el usuario; la instrucción utilizada para esto se muestra a continuación:

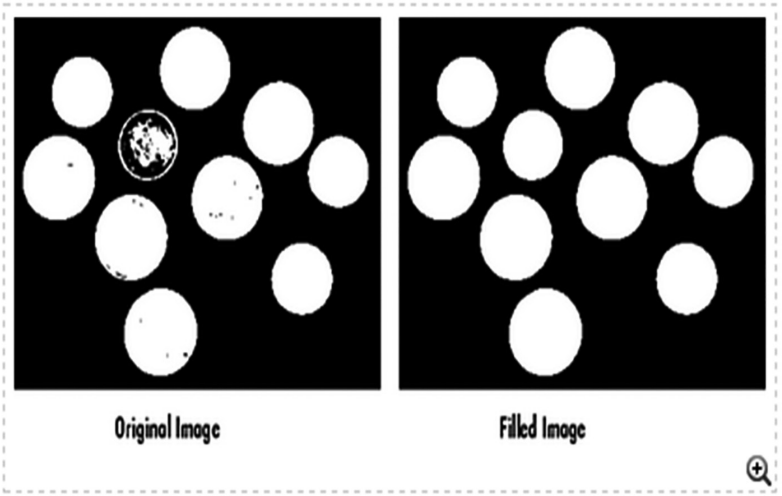
imagenBW = im2bw(imagenGris,threshold); (5.1)

Figura 81. Diagrama de flujo del algoritmo de captura, detección y segmentación de imágenes



Después de la binarización, se procede al filtrado; se aplicaron 2 filtros en serie, uno de elementos y uno de huecos. El filtro de huecos se aplicó con la función "imfill" (imagenBW, 'holes')" [71], la cual elimina todos los elementos 0 en una imagen que estén rodeados por valor 1, esto quiere decir que cualquier fondo que esté rodeado por un objeto lo considera como ruido (figura 82).

Figura 82. Filtro de huecos, imagen tomada de Matworks

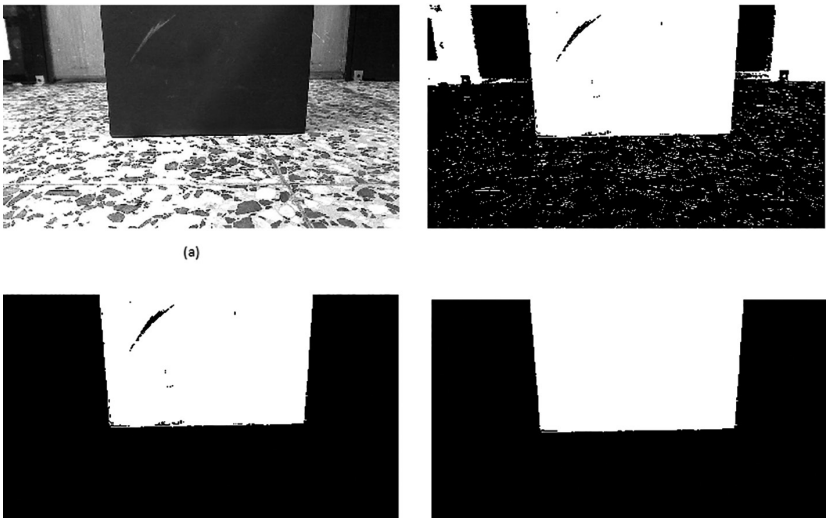


Otro filtro aplicado es la función mostrada en (5.2), que elimina todos los objetos menores a P pixeles [71].

BW2 = bwareaopen (BW, P) (5.2)

La figura 83 muestra un recorrido desde la toma de la imagen hasta la imagen resultante después de los filtros.

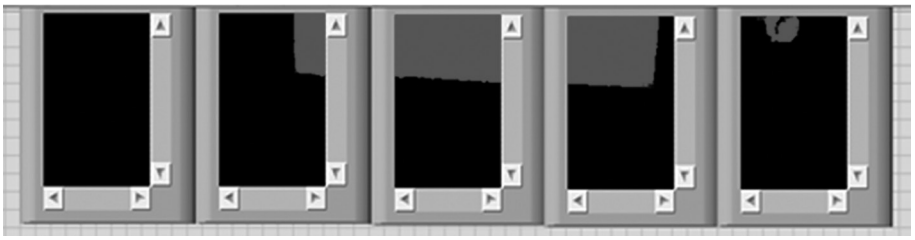
Figura 83. (a) Imagen tomada por la cámara, (b) Imagen binarizada, (c) Imagen resultante filtro elementos, (d) Imagen resultante filtro huecos



Fuente: elaboración propia.

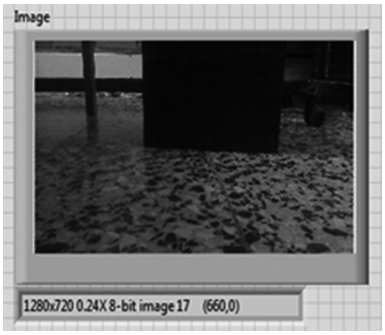
El siguiente paso es la segmentación de imágenes, el proceso de segmentación se hace con el fin de analizar la imagen en segmentos más pequeños, por consiguiente, tiene menos información y su peso computacional es menor. Otra ventaja de la segmentación de imágenes es que no se procesa información irrelevante, en nuestro caso obstáculos que estén muy lejos del robot para ser una amenaza (figura 84).

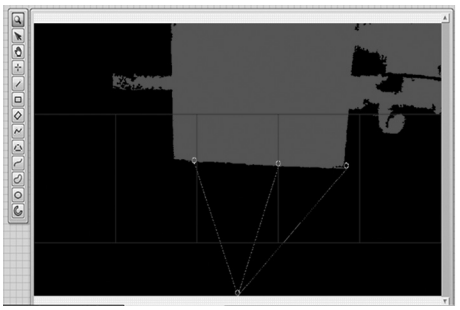
Figura 84. Segmentación de imágenes



El último paso del algoritmo es detectar los objetos dentro de las áreas segmentadas y trazar los vectores de distancia; el resultado final se ve en la figura 85b.

Figura 85a. Imagen a escala de grises **Figura 85b.** Resultado final del algoritmo





Después de analizar el algoritmo implementado en el maestro, ahora analizamos el del esclavo, este es el encargado de mover los motores de la plataforma para evitar los obstáculos, y se basa en los datos suministrados por el algoritmo maestro.

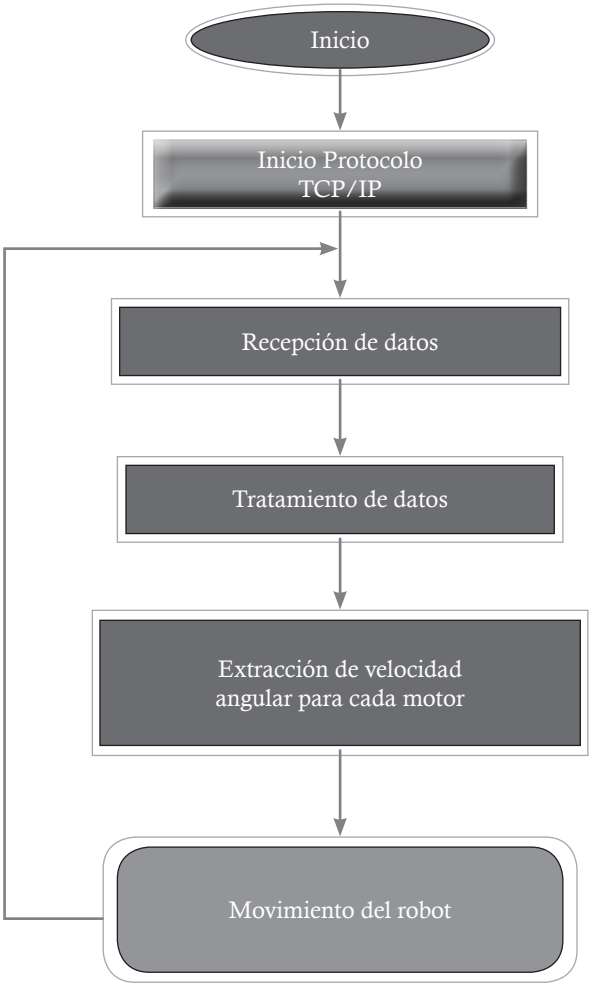
El algoritmo desarrollado en la plataforma robótica DaNI 2.0 consiste, fundamentalmente, en un protocolo basado en TCP/IP llamado "Network Streams", el cual permite el intercambio de datos entre el computador y el robot por medio de LabVIEW.

La plataforma robótica cuenta con una dirección IP fija asignada previamente y con la cual se puede establecer la comunicación con el computador por medio de tecnología wifi mediante el *router* adaptado al robot; por lo tanto, en el algoritmo maestro, se debe introducir la dirección IP de la plataforma robótica y de la misma manera en el algoritmo esclavo, la dirección IP del computador. La trama recibida por el protocolo consiste en un *String* que tiene la siguiente configuración:

$-V_{LM}\$V_{RM}$

En donde V_{LM} es la velocidad angular del motor izquierdo del robot y V_{RM} es la velocidad angular del motor derecho, ambas dadas en rad/s, el símbolo "$" se utiliza para identificar en el *String* cada una de las velocidades. El algoritmo permite realizar la desconcatenación de los números que representan las velocidades angulares correspondientes y extraerlos de manera que puedan convertirse en variables tipo *float*. Paso seguido, estas velocidades ingresan a un bloque de LabVIEW Robotics llamado "NI_Robotics_Starter Kit Host.lvclass: Write DC Motor Velocity Setpoints", el cual realiza el control necesario para garantizar dicha velocidad angular en cada uno de los motores de la plataforma robótica y cuya máxima velocidad en cada uno de los motores es de 15.7 rad/s.

Figura 86. Diagrama de flujo del algoritmo esclavo



Detección de marcas naturales con Kinect de Microsoft

El segundo sensor que analizaremos como elemento de captura de imágenes para sistemas de visión artificial es el sensor Kinect de Microsoft (figura 87); este dispositivo, aunque nuevo en el mercado, ha tenido una gran acogida entre los sistemas de captura de imágenes utilizados para sistemas de visión debido a su cámara de profundidad, ya que por medio de ella se eliminan la mayoría de fuentes de ruidos que afectan al sistema que utiliza cámaras convencionales.

Figura 87. *Kinect for Windows de Microsoft*

Fuente: [72].

El sensor de Kinect es una barra horizontal de aproximadamente 23 cm (9 in) conectada a una pequeña base circular con un eje de articulación de rótula, y está diseñado para ser colocado longitudinalmente por encima o por debajo de la pantalla de video.

El dispositivo cuenta con una cámara RGB, un sensor de profundidad, un micrófono de múltiples matrices y un procesador personalizado que ejecuta el *software* patentado, que proporciona captura de movimiento de todo el cuerpo en 3D, reconocimiento facial y capacidades de reconocimiento de voz. El micrófono de matrices del sensor de Kinect permite a la Xbox 360 llevar a cabo la localización de la fuente acústica y la supresión del ruido ambiente [72].

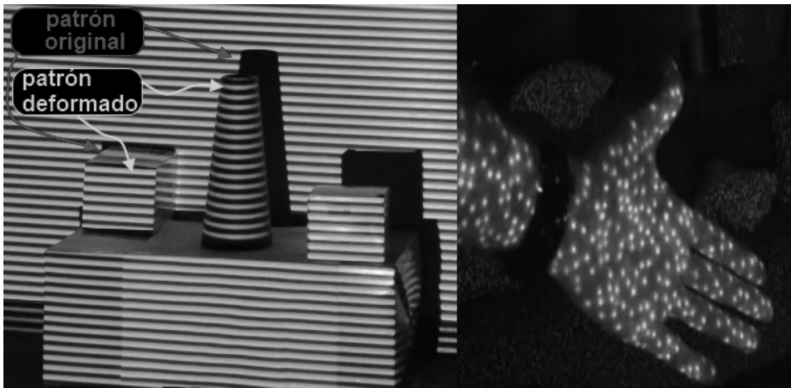
Principio de funcionamiento

Las técnicas de luz estructurada[1] usan un proyector para proyectar un patrón sobre la escena por escanear. Al hacer corresponder la imagen captada con el patrón original, es posible triangular la posición de cada píxel y determinar su profundidad con respecto al plano perpendicular a la cámara [73].

En la figura 88a se puede observar cómo se emplea la técnica de triangulación para el caso de patrones de líneas en una superficie plana, la figura 89 muestra el principio de triangulación.

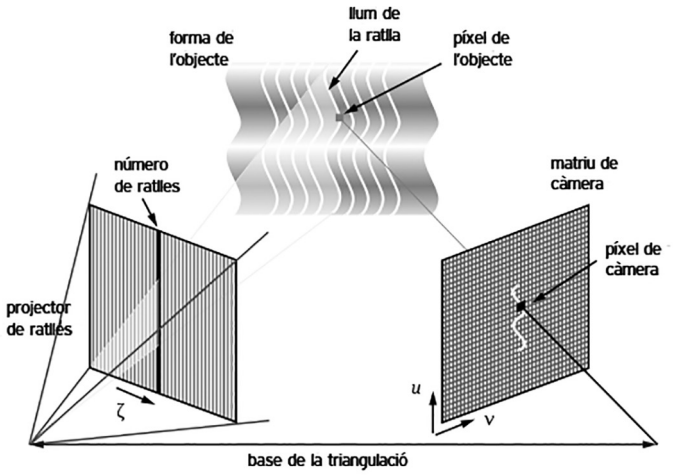
1 La técnica de luz estructurada consiste en proyectar un patrón de luz sobre la escena y observar la deformación del patrón en la superficie de los objetos. El patrón de luz es proyectado por un proyector LCD (luz no coherente) o por un barrido láser. Una cámara desplazada ligeramente respecto del proyecto captura la deformación de las líneas (o puntos) y calcula la distancia de cada punto utilizando una técnica similar a la de triangulación [85].

Figura 88. Patrones de luz estructurada a) Líneas, b) Puntos sensor Kinect



Fuente: [74].

Figura 89. Principio de triangulación aplicado a líneas



Fuente: [74].

Composición del Kinect

El sensor Kinect de Microsoft cuenta con 4 partes principales (figura 90), entre las cuales se encuentran:

Sensor de profundidad infrarrojo (emisor y receptor): el sensor de profundidad está formado por 2 componentes: un generador de láser de infrarrojos y un sensor CMOS monocromo de luz infrarroja. Esta composición le permite a la cámara capturar el video con información en 3 dimensiones bajo cualquier condición de iluminación. A diferencia de la cámara RGB, el sensor de profundidad emplea tecnología CMOS en su sensor de imagen. La principal diferencia radica en la velocidad de respuesta de un sensor CMOS, lo que permite un tiempo de exposición menor y, por lo tanto, una mayor sensibilidad a la luz.

Figura 90. Partes principales del Kinect de Microsoft

Fuente: [75].

La imagen en 3 dimensiones es construida capturando la luz infrarroja proyectada por el láser, en forma de parrilla, y calculando la profundidad con base en el tiempo que ha tardado la luz en volver al sensor.

El sensor de distancia produce imágenes con una resolución de 640 x 480 (VGA), con una información de profundidad de 11 bits, que es traducida a una imagen en escala de grises de 16 bits. Para facilitar la sincronización con la cámara RGB, también emite video a 30 fotogramas por segundo [76].

Por lo tanto, el valor de cada pixel en la imagen entregada por el sensor de profundidad equivale a la medida de profundidad realizada en milímetros desde el sensor hasta donde se encuentra el objeto detectado. En caso de presentarse un valor de 0 (cero), significa que el haz de luz emitido no fue detectado por el sensor receptor; esto se presenta cuando la distancia es mayor o menor a los límites establecidos por el Kinect, cuando un haz de luz es difractado debido a la geometría del objeto (por ejemplo, una esquina saliente) o cuando el color del objeto es negro y su textura no permite el reflejo de los haces de luz, sino que, por el contrario, los absorbe.

Cámara VGA: la captura de imágenes a color es realizada por Kinect mediante una pequeña cámara RGB situada en la parte central del artefacto. Estas cámaras funcionan utilizando unos sensores capaces de convertir la señal recibida en forma de fotones a una señal electrónica digital que descompone la luz capturada en 3 componentes: rojo, verde y azul (o *red, green, blue*, de donde se extraen sus sílabas en inglés).

En el caso de Kinect, el sensor encargado de realizar esta conversión es un sensor dispositivo de carga acoplada o CCD. Estos sensores están compuestos por una matriz de condensadores. El número de condensadores contenidos en el sensor determina su capacidad de resolución, que se mide en píxeles.

Un sensor CCD está basado en el efecto fotoeléctrico, un fenómeno físico por el cual la luz recibida es convertida en corriente eléctrica en algunos materiales. De esta forma, alterando la composición del material en el que se fabrican cada uno de los condensadores, es posible crear celdas que reaccionan ante determinadas frecuencias de la luz (en este caso, el espectro verde, el rojo y el azul).

De esta forma, para conseguir la conversión de la imagen física en imagen digital, la mayoría de cámaras CCD utilizan una máscara de Bayer que proporciona una trama para cada conjunto de 4 píxeles de forma que un píxel registra luz roja, otra luz azul y 2 píxeles se reservan para la luz verde (el ojo humano es más sensible a la luz verde que a los colores rojo o azul). El resultado final incluye información sobre la luminosidad en cada píxel, pero con una resolución en color menor que la resolución de iluminación.

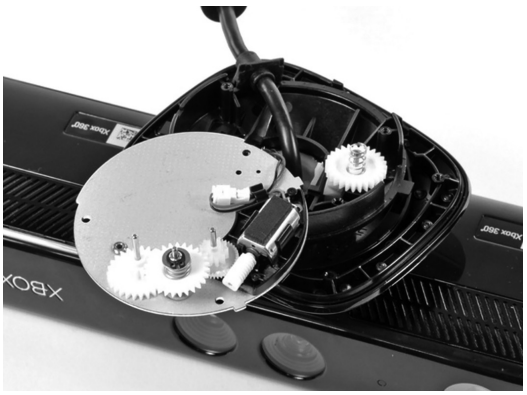
La información de color para cada píxel de la fotografía digital formada se calcula interpolando la señal de 2 píxeles verdes, 1 rojo y 1 azul. De esta manera, aunque se pierde resolución de color, se gana resolución de iluminación, lo que reduce el tiempo necesario de exposición del sensor a la luz y, por lo tanto, el ruido generado por la sobrecarga de los propios condensadores por la temperatura generada por la corriente inducida que los atraviesa [76].

En particular, la cámara RGB equipada en el Kinect tiene una resolución de 640 x 480 píxeles (VGA) y es capaz de procesar imágenes a una velocidad de 30 fotogramas por segundo, con una resolución de color de 32 bits (1 byte por cada color, 16.7 millones de colores). Adicionalmente, en el SDK utilizado actualmente se presentan resoluciones de 1280 x 960 píxeles 12 Fps (RGB), 1280 x 960 píxeles 12 Fps (RawBayer), 640 x 480 píxeles 30 Fps (RGB), 640 x 480 píxeles 15 Fps (YUV), y como función añadida se encuentra 640 x 480 píxeles 30 Fps (Infrared), la cual consiste en combinar los haces de luz infrarroja del emisor y capturarlos con la cámara VGA para obtener una imagen en escala de grises de 16 bits que funciona con o sin presencia de luz visible.

Base motorizada: tras investigaciones para ver las diferentes configuraciones de espacios de vida en toda América, Europa y Asia, Microsoft llegó a la conclusión de que era necesario dotar a la cámara del Kinect de la posibilidad de moverse hacia arriba o hacia abajo con el objetivo de calibrar cada espacio concreto [77].

El ajuste de inclinación para el sensor Kinect está dado por un pequeño servomotor (figura 91), el cual amplía su campo de visión vertical. Se encuentra ubicado en la base del Kinect y es controlado mediante el SDK presente en el computador [34, p45].

Figura 91. Motor para ajuste de inclinación del Kinect

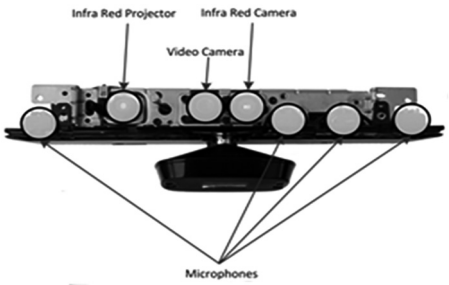


Fuente: [78].

La inclinación motorizada le permite entonces al Kinect tener un rango de visión horizontal de 57.5° aproximadamente (constante) y un rango de visión vertical de 43° aproximadamente (constante) con un ajuste de inclinación física adicional entre 27° y -27°; por lo tanto, cuando se ajusta un ángulo de 0° para el motor, el Kinect está totalmente paralelo al suelo, en caso contrario se hace el ajuste respectivo de inclinación del Kinect.

Micrófono Multi-Array (Arreglo de 4 Micrófonos Unidireccionales): Micrófono Multi-Array de 4 micrófonos ubicados a los extremos del sensor, cada canal procesa 16-bit (Kinect Xbox360) y 24-bit (Kinect for Windows) y un rango de muestreo de 16 KHz. Estos micrófonos son la única razón por la cual el sensor es tan ancho y se monta como un solo micrófono y se usa para reconocimiento de voz y charlas [77].

Figura 92. Ubicación de los micrófonos del Kinect de Microsoft



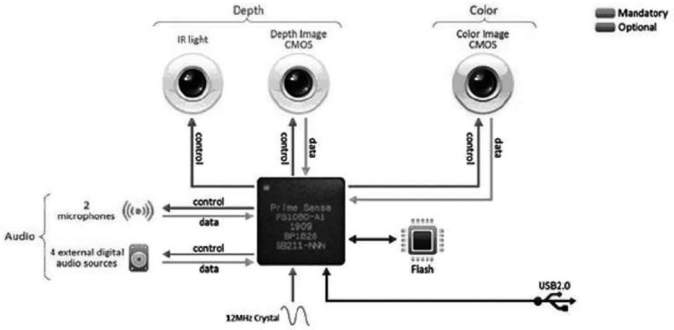
Fuente: [79].

El hecho de que el sensor Kinect posea 4 micrófonos unidireccionales a lo ancho de su estructura permite que el reconocimiento de la fuente de voz sea más preciso y se pueda estimar el ángulo con el cual dicha fuente incide respecto al Kinect, esto con el fin de crear posibles aplicaciones en las cuales esta función sea útil, o simplemente realizar ajustes de voz para los sonidos capturados. En el Toolkit para desarrolladores incluido en las últimas versiones del SDK se incluyen algunas aplicaciones de reconocimiento de voz mediante la plataforma Kinect Speech Platform, además de los paquetes de idioma adicionales Kinect Speech Language Packs, los cuales permiten el reconocimiento de palabras y los procesa como comandos de control.

Partes adicionales: el sensor Kinect cuenta, además, con una serie de partes adicionales que permiten acoplar las partes mencionadas anteriormente o mejorar el rendimiento del dispositivo. Entre estas se encuentran:

- Memoria RAM de 512 Mb.
- Acelerómetro para estabilizar la imagen cuando se mueve.
- Ventilador, no está encendido continuamente para no interferir con los micrófonos.

Figura 93. Esquema general del Kinect de Microsoft



Fuente: [80].

Kinect for Windows y Matlab

Matlab es un entorno de programación orientado al cálculo matemático especialmente con vectores y matrices; sin embargo, por medio de sus nuevas versiones, se ha ofrecido el soporte para *hardware* compatible mediante sus *toolboxes*; este es el caso del Kinect for Windows, para el cual Matlab desde su versión 2013a hace uso de su Image Acquisition Toolbox con el fin de adquirir y manipular las imágenes capturadas por el Kinect como si se tratase de cámaras USB conectadas al computador. Para poder hacer uso de estas funciones, es necesario instalar desde Matlab el Kinect for Windows Runtime.

La instalación de este paquete de soporte permite usar las funciones propias del Image Acquisition Toolbox, ejecutadas en el sensor Kinect mediante comandos de Matlab. Es importante recalcar que el Kinect cuenta con una cámara VGA y un sensor de profundidad, los cuales se comportan en Matlab como 2 dispositivos de captura de imágenes independientes; por tal motivo, la adquisición por medio de estos se realiza de la misma manera. Algunas de las funciones para la adquisición de imágenes desde el Kinect se describen a continuación.

- *imaqhwinfo:* brinda información sobre los dispositivos de captura de imágenes asociados al computador. En el caso del Kinect, se enumera la cámara VGA como 1 y el sensor de profundidad como 2. Adicionalmente, muestra información relacionada con los formatos soportados por el Kinect; para el caso de la cámara VGA, son: 'RawYUV_640x480', 'RGB_1280x960', 'RGB_640x480', 'YUV_640x480', 'Infrared_640x480', 'RawBayer_1280x960', y 'RawBayer_640x480', y en el caso del sensor de profundidad, son: 'Depth_640x480', 'Depth_320x240' y 'Depth_80x60'.
- *videoinput:* crea un elemento de entrada de video mediante el dispositivo y formato seleccionados.
- *getselectedsource:* crea un elemento de fuente de video por medio del elemento de video seleccionado. Permite la visualización de los parámetros de los dispositivos conectados.

- *set:* permite el ajuste individual de los parámetros de los dispositivos de captura de imágenes, conectados siempre y cuando los dispositivos no estén adquiriendo imágenes. En el caso del Kinect, se pueden realizar ajustes propios del SDK, como ángulo de elevación de la cámara, modo de profundidad, postura del cuerpo, esqueletización, entre otros.
- *preview:* en una ventana nueva, se realiza la visualización del dispositivo seleccionado con los parámetros ajustados previamente.
- *FramesPerTrigger y TriggerRepeat:* FramesPerTrigger configura la cantidad de *frames* o fotogramas que serán adquiridos cada vez que se ejecute el comando de adquisición (Trigger); y la función TriggerRepeat configura el número de veces que se ejecutará el Trigger, una vez que llegue a este número el bucle terminará, por este motivo, si se desea un bucle infinito, se debe ajustar este parámetro como *Inf.*
- *triggerconfig:* configura el tipo de Trigger requerido de acuerdo con la orden de ejecución que se necesita dependiendo del programa a desarrollar. Usualmente, para que la adquisición se realice cada vez que se ejecute el comando, se selecciona tipo manual.
- *start:* inicia el dispositivo con los parámetros configurados previamente.
- *trigger:* usualmente, este comando se utiliza dentro de un bucle, ya que es el responsable de la adquisición de imágenes en cada iteración. En caso de requerir la adquisición simultánea de las imágenes de color y profundidad en el Kinect, este comando ejecuta dicha adquisición en ambas cámaras en el mismo instante de tiempo; esto marca una gran diferencia respecto al comando *getsnapshot,* ya que este último realiza la adquisición de manera consecutiva, lo que genera corrimientos de imagen entre la información de color y la de profundidad.
- *getdata:* realiza el proceso de adquisición de datos correspondientes al *frame* o fotograma actual. Este comando debe ejecutarse para cada dispositivo por separado.

  En este caso, los elementos ImagenColor e ImagenDepth corresponden a las matrices generadas por las imágenes capturadas por la cámara VGA y el sensor de profundidad, respectivamente. Los números ts_color y ts_depth indican el tiempo en segundos transcurrido desde la adquisición del primer *frame* hasta el momento actual. Los elementos metaData_Color y metaData_Depth corresponden a estructuras de Matlab que contienen información propia del Kinect, como tiempo absoluto y, en el caso del sensor de profundidad, información relacionada con el Skeletal_Stream.
- *stop:* en caso de que no se haya cumplido la condición configurada en TriggerRepeat, se debe usar esta función para detener el dispositivo o, en general, para evitar errores la siguiente vez que se ejecute el programa.

Kinect for Windows y LabVIEW

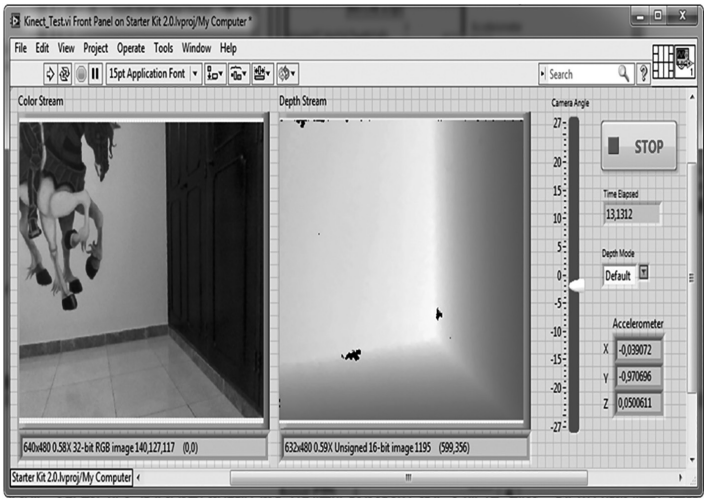
LabVIEW es un entorno de programación gráfico y presenta, dentro de sus más importantes ventajas, la ejecución de tareas de manera paralela; por lo tanto, la visualización y el procesamiento de imágenes se pueden realizar en tiempo real. Por otro lado, Matlab cuenta con el soporte para Kinect for Windows y funciones para la adquisición de imágenes capturadas por dicho sensor; estas funciones no están presentes de manera directa en LabVIEW, lo que se convierte en un problema cuando se requiere trabajar con el Kinect en dicho entorno de programación. Como solución a esta problemática se usa de la fusión Matlab-LabVIEW, mencionada anteriormente, y mediante el uso del Matlab Script Node se ejecutan los comandos de adquisición de imágenes con el Kinect y ajuste de parámetros de este.

En el desarrollo del presente apartado, se incluye un programa realizado en el entorno de programación LabVIEW 2012, llamado Kinect_Test.vi (figura 94), el cual permite la visualización simultánea tanto de la cámara VGA como del sensor de profundidad, presentes en el Kinect for Windows, además de ajuste de parámetros, como ángulo de inclinación del Kinect (entre -27° y 27° verticales) y modo de profundidad (*Near* o *Default*) y visualización del tiempo transcurrido y los valores medidos por el acelerómetro interno del Kinect for Windows. El objetivo de este programa es verificar el correcto funcionamiento del sensor Kinect for Windows de Microsoft, así como la implementación de la fusión Matlab-LabVIEW y comandos de control mostrados anteriormente.

La ejecución del programa Kinect_Test.vi se realiza desde su panel frontal y consta de 3 etapas: verificación de conexión del Kinect, visualización de imágenes y ajuste de parámetros, detención del Kinect.

- La primera etapa incluye una ventana emergente, en la cual se indica si el Kinect for Windows ha sido conectado o detectado por el programa. En caso de no ser detectado la ventana emergente (figura 94) muestra el mensaje "Kinect no Conectado", e incluye 2 botones con las opciones "Cancelar" (con la cual se detiene la ejecución del programa) y "Reintentar" (con la cual se ejecuta el programa de nuevo para verificar el estado de conexión del Kinect).
- La segunda etapa del programa consiste en un bucle infinito en el cual se realiza la visualización de las imágenes adquiridas por la cámara VGA y el sensor de profundidad, respectivamente, mediante indicadores en el panel frontal del programa, ajuste del ángulo de inclinación del Kinect mediante un *slider* y el modo de profundidad mediante un menú desplegable y visualización del tiempo transcurrido y del acelerómetro interno del Kinect en sus ejes X, Y y Z (figura 94).
- La tercera y última etapa consiste en detener el bucle infinito mediante el botón "STOP", ubicado en el panel frontal, en la esquina superior derecha. Esto permite dejar de adquirir imágenes y ejecutar el comando *stop* de Matlab para detener el Kinect y apagar el emisor infrarrojo. La no ejecución de esta función puede repercutir en errores en el momento de volver a correr el programa.

Figura 94. Panel Frontal Kinect_Test.vi



Obtención de parámetros para ajuste de coordenadas x, y en la matriz de profundidades

Las imágenes adquiridas en Matlab corresponden a matrices de 2 dimensiones, cuyo tamaño (número de filas y columnas) equivale a la resolución de la imagen y donde cada celda equivale a un pixel y su valor depende del formato seleccionado; en caso de las imágenes a color, por ejemplo el formato RGB, cada una de las imágenes está compuesta por 3 matrices del mismo tamaño, donde la primera equivale al componente R (*red*), la segunda al componente G (verde) y la tercera al componente B (azul). Para el caso específico del sensor de profundidad del Kinect, la imagen resultante consiste en una matriz de 16 bits, donde el valor de cada una de las celdas o píxeles corresponde a la medición de profundidad en milímetros realizada por el sensor.

Las últimas versiones del Kinect for Windows SDK ofrecen una resolución adicional para la cámara VGA, basadas en cámaras infrarrojas. Aprovechando los haces de luz infrarroja emitidos por el sensor de profundidad, se desarrolló una función que permite capturar dicha luz mediante la cámara VGA, con el fin de crear una cámara que permitiera una buena visibilidad aun en ausencia de luz visible. La imagen resultante corresponde a una imagen monocromática de 16 bits (escala de grises) con la ventaja adicional de que cada píxel coincide espacialmente con el mismo número de píxeles de la imagen generada por el sensor de profundidad.

Utilizando los comandos mencionados en el apartado anterior, y bajo el entorno de programación Matlab, se realizaron capturas con la cámara VGA del sensor Kinect, en el formato 'Infrared_640x480', y mediante una cuadrícula elaborada en un contraste blanco/negro, se efectuaron mediciones para obtener los parámetros que permitieran realizar la transformación de coordenadas de píxeles a coordenadas x, y, que correspondieran a valores lo más cercanos posible a los valores exactos del espacio medido a través del Kinect.

Se tomó como referencia una pared, a la cual se adhirió la cuadrícula mencionada y desde la cual se ubicó el sensor Kinect de Microsoft a una distancia aproximada de 1.3 m, con una altura de 1.2 m. La imagen obtenida por la cámara VGA se muestra en la figura 95.

Gracias a las herramientas de Matlab, se pueden poner marcadores en las imágenes, los cuales indican las coordenadas en píxeles y el valor de dicho píxel. Mediante el uso de estos marcadores se realizaron las mediciones sobre la imagen y las mediciones en el mundo real o espacio se realizaron mediante un flexómetro comercial. De esta manera, se obtuvo:

- Medición ancho de cuadrícula: 1360 mm.
- Ancho de cuadrícula en píxeles: 622 px – 23 px = 599 px.
- Valor aproximado de mundo real por píxel: 1360 mm/599 px = 2.27045 mm/px.

Al hacer uso de funciones de procesamiento de imágenes de Matlab, se tomó la imagen de referencia y se transformó en una imagen binaria, con el fin de tener una mejor claridad sobre el valor de cada píxel capturado por el Kinect. La imagen resultante se muestra en la figura 96.

Aplicando una función de *zoom* sobre la imagen de referencia binarizada (figura 96) y utilizando los mismos marcadores usados previamente, se obtuvieron las siguientes mediciones:

Figura 95. Imagen referencia para la obtención de parámetros

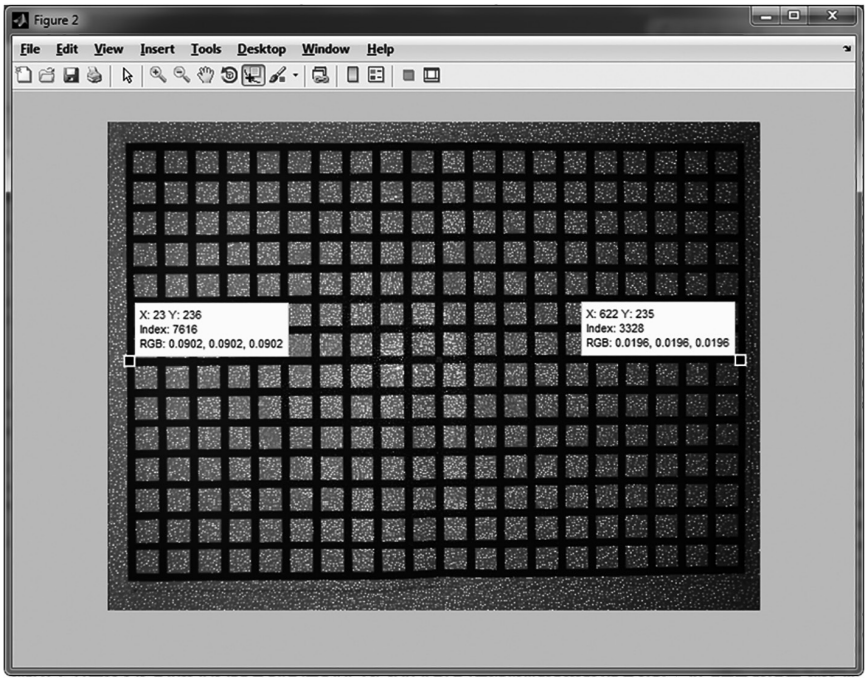
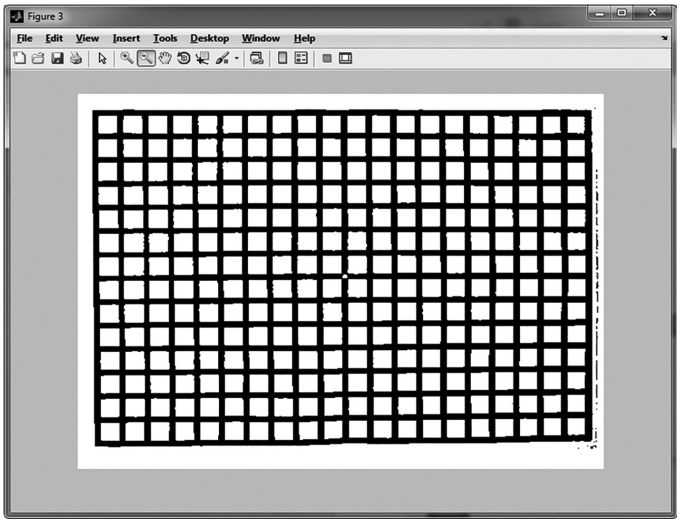
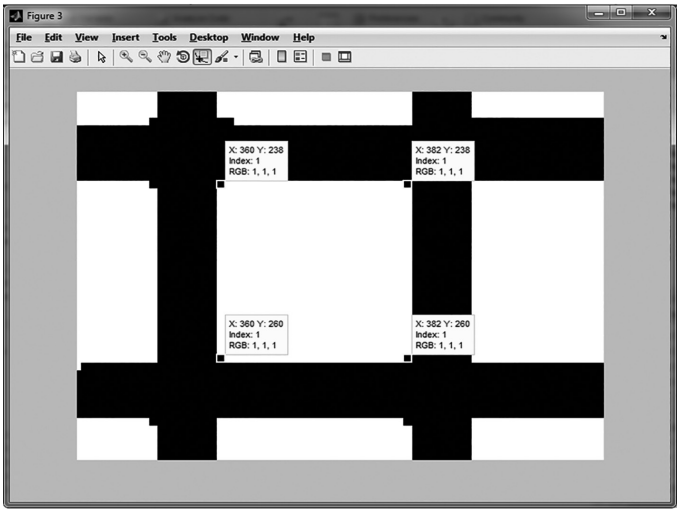


Figura 96. Imagen referencia binarizada



- Medición cada cuadro: *50 mm.*
- Medición cada cuadro en píxeles: *382 px – 360 px = 22 px.*
- Aplicando valor por píxel: *22 px * 2.27045 = 49.95 mm.*
- Medición ancho de línea: *18 mm.*
- Medición cada línea en píxeles: *8 px.*
- Aplicando valor por píxel: *8px * 2.27045 = 18.08 mm.*

Figura 97. Imagen referencia binarizada y ampliada



Posteriormente, se realizaron mediciones en los extremos de la cuadrícula, tanto en la imagen (figura 98) como en el mundo real (tabla 16).

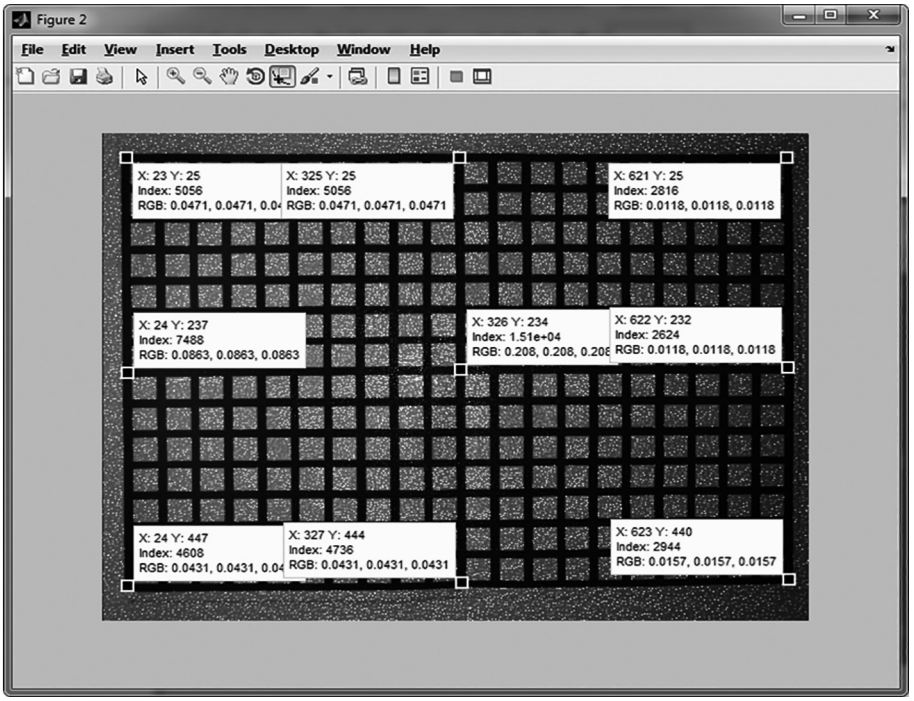
Tabla 16. Mediciones realizadas en los extremos de la cuadrícula dadas en milímetros

Esq. Sup. Derecha	Central Superior	Esq. Sup. Izquierda
1590	1405	1535
Central Derecha	Centro	Central Izquierda
1510	1320	1457
Esq. Inf. Derecha	Central Inferior	Esq. Inf. Izquierda
1585	1402	1530

Tomando como referencia el eje horizontal central, se puede suponer la visibilidad del Kinect como un triángulo escaleno y, de esta manera, hallar el ángulo de visión del ancho de la cuadrícula de la siguiente manera:

Aplicando teorema del Coseno: $a^2 = b^2 + c^2 - 2 * b*c* \cos(\alpha)$

Figura 98. Imagen referencia con mediciones en los extremos de la cuadrícula



Donde a = 1360 mm, b = 1457 mm, c = 1510 mm. Despejando α:

$$\alpha = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2 * b * c}\right) = \cos^{-1}\left(\frac{1457^2 + 1510^2 - 1360^2}{2 * 1557 * 1510}\right) \approx 54.52^{o}$$

Dividiendo entre el número de píxeles y multiplicando por el ancho de la imagen:

$$\frac{54.52}{599} * 632 = 57.52^{o} \approx 57.5^{o}$$

El resultado obtenido de la operación anterior igual a 57.5° es el ángulo de visión horizontal entregado en la hoja técnica del Kinect [81].

Para hallar el ángulo de visión vertical, se procede de la misma manera. Aplicando el teorema del Coseno:

$a^2 = b^2 + c^2 - 2 * b*c* \cos(\alpha)$

donde a = 952mm, b = 1405mm, c = 1402mm. Despejando α:

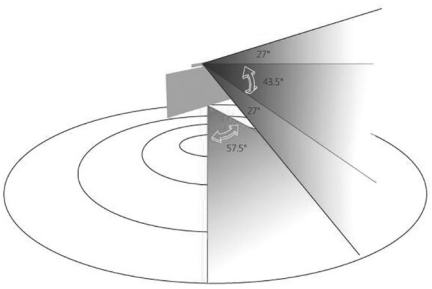
$$a = \cos^{-1}\left(\frac{b^2 + c^2 - a^2}{2 * b * c}\right) = \cos^{-1}\left(\frac{1405^2 + 1402^2 - 952^2}{2 * 1405 * 1402}\right) \approx 39.5^{o}$$

Dividiendo entre el número de píxeles y multiplicando por el alto de la imagen:

Cateto Opuesto = *Cateto Adyacente* * tan (θ)

El resultado obtenido de la operación anterior igual a 45.2° es el ángulo de visión vertical que es aproximadamente igual al ángulo entregado en la hoja técnica del Kinect que es 43.5° [81].

Figura 99. Ángulos de visión del Kinect de Microsoft



Fuente: [82].

Mediante las mediciones realizadas anteriormente, y gracias a los resultados obtenidos, se genera un vector cuya longitud corresponde al ancho de la matriz de profundidad y cuyos datos corresponden al ángulo horizontal obtenido dividido equitativamente. Mediante Matlab, el código utilizado para obtener dicho vector se muestra a continuación:

```
a = acosd((1443^2+1447^2-1360^2)/(2*1443*1447));
for i=1:length(ImagenDepth)
 angulos(i)=(a/621)*(i-317);
end
```

El vector de ángulos se genera una sola vez al inicio del programa, debido a que dichos ángulos son estáticos y constantes para el Kinect. Una vez obtenido, se parte del principio de transformación de sistemas de coordenadas (transformar los datos obtenidos en coordenadas polares a coordenadas rectangulares), lo cual se logra mediante trigonometría convencional. Sin embargo, la matriz entregada por el sensor de profundidad del Kinect corresponde a las coordenadas *'y'* medidas, ya que internamente el Kinect hace un procesamiento de dichos datos; se usó la función trigonométrica tangente para conocer las coordenadas *'x'* a partir de dicha matriz de profundidades. Suponiendo un triángulo rectángulo, la función trigonométrica tangente se define como:

$$\tan(\theta) = \frac{Cateto\ opuesto}{Cateto\ adyacente}$$

Se supone el cateto adyacente como la distancia obtenida en cada una de las celdas de la matriz de profundidades. Por este motivo, se requiere hallar el cateto opuesto y se despeja de la ecuación anterior.

$$Cateto\ Opuesto = Cateto\ Adyacente * \tan(\theta)$$

La distancia hallada en el cateto opuesto corresponde a la coordenada *'x'* y la distancia del cateto adyacente, a la coordenada *'y'*. Debido a que la imagen entregada por el sensor de profundidad es una matriz, basta con calcular la función tangente de cada elemento del vector de ángulos y multiplicarlo por cada una de las columnas de la matriz de profundidades, el resultado es una matriz que contiene las coordenadas *'x'* de la imagen capturada y, en conjunto con la matriz de profundidades, se obtienen las coordenadas *'x, y'* requeridas; esto se logra en Matlab de la siguiente manera:

```
for i=1: length(ImagenDepth)
        x_Depth(:,i) = tand(angulos(i))*ImagenDepth(:,i);
end
```

Para graficar los puntos obtenidos con las coordenadas *'x, y'* halladas, se utiliza la función *plot* de Matlab.

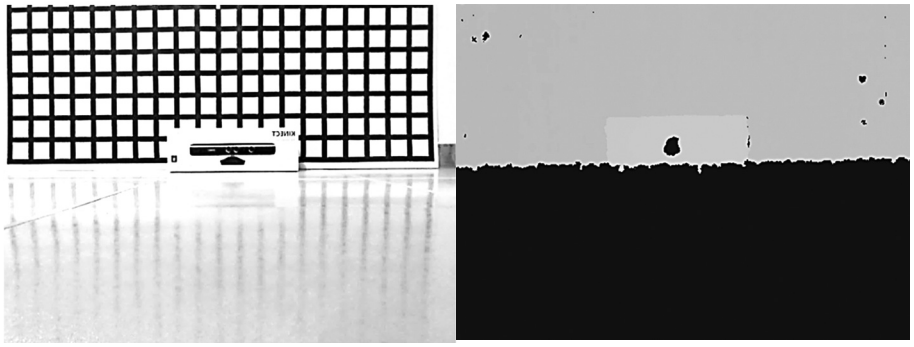
```
plot(x_Depth,ImagenDepth,'.r','MarkerSize',1)
```

Las imágenes obtenidas por la cámara VGA, el sensor de profundidad y la gráfica resultante en coordenadas rectangulares se muestran en la figura 100.

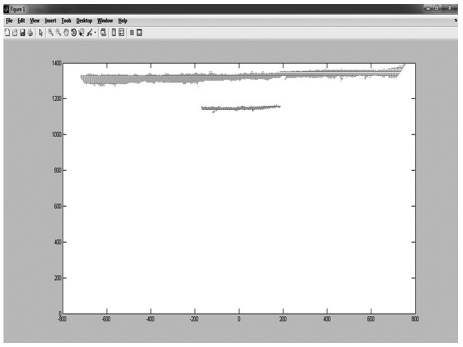
Adaptación del Kinect a la plataforma robótica

Actualmente, en el mercado se consiguen accesorios diseñados con el fin de instalar correctamente los dispositivos fabricados por una empresa, esto hace parte de los productos complementarios que pretenden dar soluciones desde la industria a pequeñas problemáticas. Por este motivo, muchos de estos accesorios son fabricados por la misma empresa que fabrica el dispositivo principal. Este es el caso de Microsoft, que ofrece, por medio de la página oficial de su tienda en línea, una serie de accesorios para sus consolas o dispositivos ligados a ellas, incluido el Kinect. En la figura 101 se muestra el gancho de Kinect para montaje sobre televisor, pensado inicialmente para Kinect, para Xbox360, de manera que pudiera ser ubicado de manera segura en la parte superior de cualquier televisor.

Figura 100. Imágenes obtenidas por (a) cámara VGA (b) sensor de profundidad (c) gráfica en coordenadas rectangulares



(a) (b)



(c)

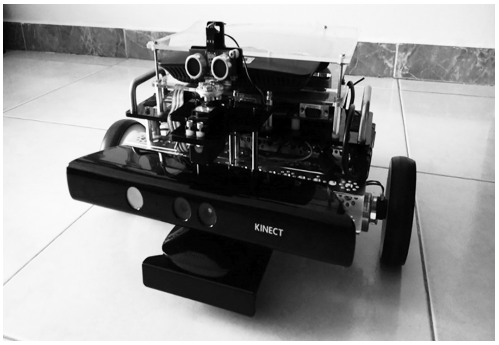
Con el fin de no alterar el Kinect for Windows, proporcionado por Roma, de la Universidad Distrital Francisco José de Caldas, Facultad Tecnológica, se usó dicho accesorio para sujetar el Kinect a la plataforma robótica DaNI 2.0, el cual permite su desmontaje y montaje de manera rápida y segura mediante los botones ubicados en la parte lateral del Kinect Sensor TV Mounting Clip. De las modificaciones hechas a este accesorio se tomó únicamente su parte superior y mediante la extracción de tornillos, se descartó el resto del gancho. La parte utilizada se fijó a una base acrílica previamente instalada a la plataforma robótica.

Figura 101. Kinect Sensor TV Mounting Clip

Fuente: [83].

Adicionalmente, se requería un sistema de alimentación eléctrica para el Kinect for Windows que permitiera el libre movimiento de la plataforma robótica, debido que la fuente de alimentación con la cual viene originalmente el Kinect consiste en un adaptador de 12VDC que debe conectarse a la red eléctrica doméstica, por lo tanto se originan problemas en cuanto a limitaciones de espacio y extensiones eléctricas que utilizar. Pensando en esto, se instaló una batería común (batería seca) de 12VDC a la plataforma robótica y usando el mismo tipo de adaptador del Kinect for Windows, se modificó para mantener el puerto hembra de Kinect y el puerto USB macho pero cuya alimentación fuera tomada desde la batería instalada.

Figura 102. Foto de la plataforma con el Kinect



Extracción de marcas naturales

Las marcas o balizas son características del entorno que se emplean para localizar el robot. Pueden ser naturales o ser introducidas artificialmente en el entorno. Esta técnica se sitúa entre las que interpretan el entorno, aunque la estimación de la posición en realidad no se hace dependiendo del entorno, sino por las marcas detectadas [84].

Las marcas naturales son objetos o características del entorno y no tienen la funcionalidad de posicionar el robot, aunque se empleen para eso. En la navegación con marcas naturales, una de las tareas más difíciles es la detección de las marcas. Por lo general, en la búsqueda de marcas naturales se emplean las cámaras de video y, como marcas, se trata de detectar bordes verticales de gran dimensión que pueden corresponder, por ejemplo, a marcos de puertas o paredes [85].

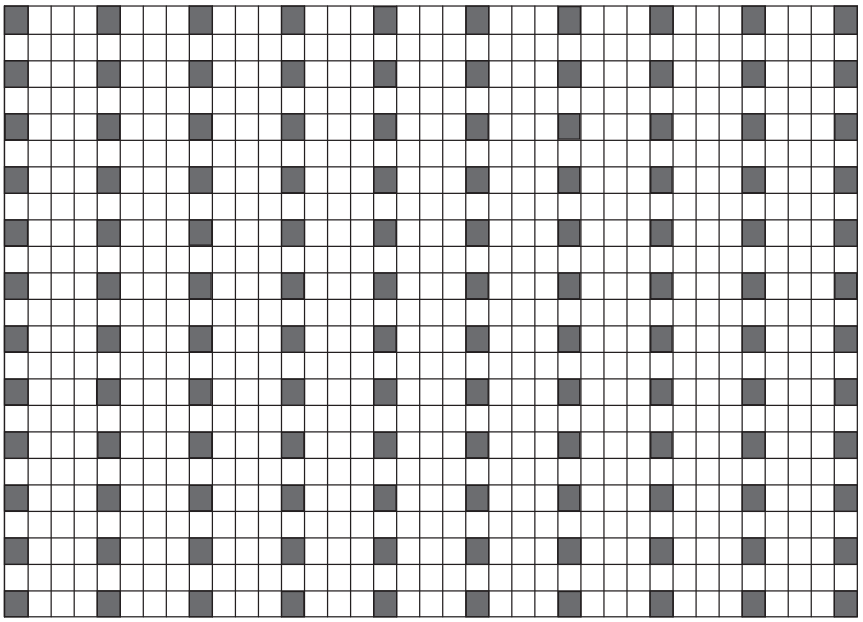
A pesar de que los métodos de extracción de marcas naturales mencionados anteriormente utilizan cámaras de video como dispositivo principal, se basan en variaciones abruptas de color e intensidad tanto en barridos verticales como horizontales, lo que permite la identificación de esquinas o bordes; sin embargo, dichos sistemas presentan desventajas en cuanto a la adecuada iluminación del espacio y un alto costo computacional, debido a la complejidad de los modelos planteados para estimar la posición de dichas marcas.

La implementación del Kinect for Windows de Microsoft permite la extracción de marcas naturales mediante su sensor de profundidad, ya que brinda información de la distribución geométrica espacial del entorno. Como componente fundamental en el desarrollo del algoritmo de exploración de espacios planteado, se requiere la extracción de marcas naturales de dicho entorno, ya que permite determinar características del espacio por explorar y, de esta manera, guiar al robot mediante la planeación de rutas que a su vez permitan la evasión de obstáculos mientras se realiza su navegación autónoma.

El tipo de marca natural seleccionado fueron las esquinas denominadas también "Spike Landmarks" [86], ya que planteando una situación de riesgo en la que se necesite un agente *USAR* [87], en una zona colapsada, los elementos más propensos a permanecer estáticos son las columnas de las construcciones o sus esquinas, por este motivo, mediante la identificación de esquinas como marcas naturales, se puede obtener un entorno semiestático para la ejecución del algoritmo planteado.

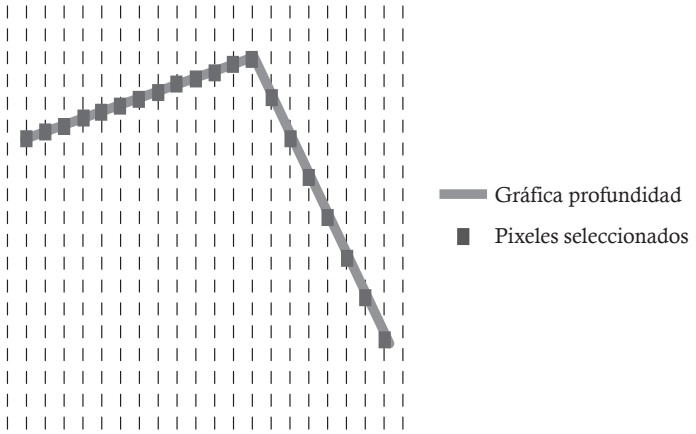
El algoritmo de detección de esquinas implementado toma como base la imagen entregada por el sensor de profundidad del Kinect, esto en sí consiste, específicamente, en una matriz, donde cada celda contiene el valor correspondiente a la medición de profundidad realizada y dada en milímetros. Con el fin de reducir tiempos de cómputo, dicha imagen es discretizada de manera que se puedan tomar puntos o píxeles separados entre sí con un ancho y alto constante respecto a la posición de dicho píxel en la imagen. Este proceso se puede apreciar en la figura 103, donde la cuadrícula corresponde al número de píxeles de la imagen de profundidad, las casillas de color gris corresponden a las posiciones de los píxeles seleccionados y las casillas de color blanco se descartan.

Figura 103. Discretización de la imagen de profundidad



El proceso de análisis de los píxeles seleccionados en busca de detección de esquinas se realiza mediante un barrido horizontal, esto quiere decir que para cada iteración, se usan los puntos seleccionados que estén en una misma fila de la matriz original. La figura 104 muestra un ejemplo de una de estas iteraciones, si se graficaran en coordenadas cartesianas cada uno de los píxeles dispuestos en la misma fila de la imagen de profundidad (en color gris claro) y los píxeles seleccionados mediante el proceso de discretización mostrado anteriormente (en color gris oscuro).

Figura 104. Gráfica ejemplo de una fila de la matriz de profundidades en coordenadas cartesianas



El principio matemático utilizado en el algoritmo consiste en el cálculo de las pendientes entre pares de puntos seleccionados y, analizando continuamente dichas pendientes, identificar cambios de signo, lo cual indica un punto de la detección de una esquina. La siguiente ecuación muestra la manera de hallar el valor de la pendiente *"m"* a partir de 2 puntos en coordenadas cartesianas.

$$m = \frac{y_1 - y_2}{x_1 - x_2} = \frac{\Delta y}{\Delta x}$$

Los valores respectivos para Δx son constantes, debido a que corresponden a un ancho fijo y que dependen de la posición del píxel en la imagen; por otro lado, los valores '*y'* planteados en la ecuación corresponden a las mediciones de profundidad almacenadas en dichos píxeles y dadas en milímetros. Por tal motivo, la ecuación anterior en función de las iteraciones se puede escribir como:

$$m_k = Depth_{i,j} - Depth_{i,j+\Delta x}$$

Donde '*Depth'* es la matriz de profundidades, *'i'* es el número de fila y *'j'* es el número de columna de la matriz de profundidades, 'Δx' es un valor constante y '*mk*' es el valor de la pendiente para cada par de puntos dados.

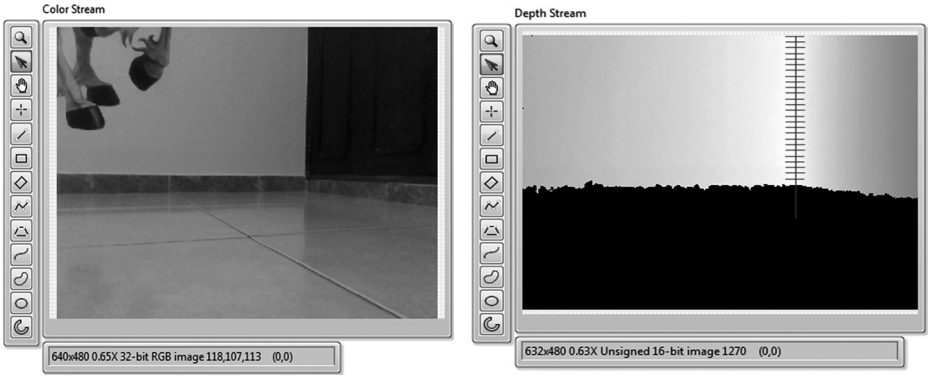
El siguiente paso es la comparación de los valores m_k consecutivos, para lo cual se usa la *ley de signos,* estudiada en algebra básica. Fundamentalmente, esta ley señala que la multiplicación entre 2 números del mismo signo da como resultado un número de signo positivo, y la multiplicación entre números con signo opuesto da como resultado un número de signo negativo. Por tanto, se genera una variable llamada *signo:*

$$signo = m_k * m_{k+1}$$

En caso que la variable signo sea negativa o menor que cero, se identifica una marca natural potencial, y se almacena un valor de uno en una matriz que inicialmente está completamente en cero. Una vez se han tomado todos los puntos seleccionados y se ha completado la matriz de unos y ceros anteriormente mencionada, se procede a sumar los valores de las columnas y, estableciendo un umbral, se puede saber si se trata de una esquina o no. En caso afirmativo, se calculan las coordenadas cartesianas y se almacenan los datos en un vector que es utilizado en otra etapa del algoritmo.

En la figura 105 se puede apreciar la detección de una marca natural tipo esquina; en la imagen de la izquierda se muestra la imagen capturada por la cámara VGA del sensor Kinect for Windows de Microsoft y en la imagen de la derecha se muestra la imagen generada por la matriz de profundidades adquirida por el Kinect en el mismo instante de tiempo. En dicha imagen se indican mediante líneas horizontales de color gris oscuro los puntos de inflexión hallados, en los cuales se evidencia el proceso de discretización mencionado anteriormente; se indica, además, mediante una línea vertical, la marca detectada como resultado de la aplicación del umbral establecido para el número de puntos de inflexión hallados.

Figura 105. Imagen de una esquina capturada con el Kinect (izquierda). Detección de marcas naturales (derecha)



Control del robot basado en emociones

Investigadores en ciencias de la cognición y áreas afines han propuesto modelos computacionales de las emociones, los cuales pueden ayudar a emular aspectos de la cognición tan importantes como la memoria, la adaptación y la generación de autonomía. Estos modelos son útiles para probar o refutar teorías sobre el aprendizaje y la toma de decisiones. Uno de los más completos, basado en aportes de la neurociencia, estudia la interacción entre la atención y la memoria, y un número menor modela el proceso cognitivo de la toma de decisiones, en especial aplicado a la robótica y a los agentes virtuales, en donde se deben planear los movimientos de manera autónoma.

Selección de un modelo computacional de las emociones

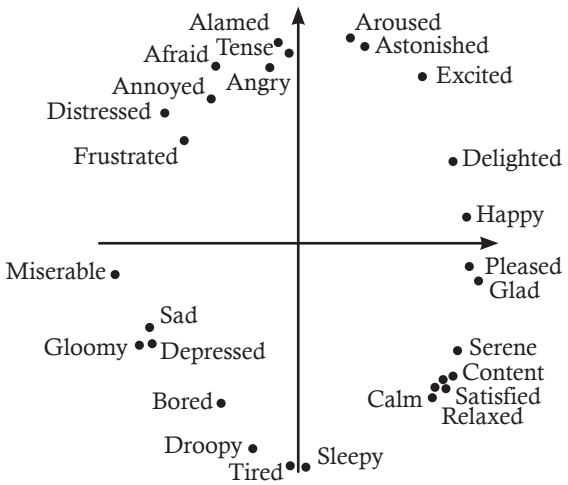
Esta sección explica la selección de un modelo computacional de emociones, el cual puede ser usado como el núcleo de una arquitectura de control basada en emociones, como se explica en la sección siguiente. Una clasificación posible de los modelos de las emociones, las distinguen entre discretas y continuas. Los modelos discretos definen un conjunto de emociones básicas. Por ejemplo los autores en [88] y [89] definen exclusivamente ira y temor para definir un algoritmo de búsqueda, el cual genere autonomía en sistemas de control. Los modelos discretos han inspirado muchas aplicaciones en ingeniería, y algunos de los teóricos principales son Robert Plutchik, Paul Ekman, and Nico Frijda [90]. Sus modelos tienen ocho, seis, y seis emociones básicas, respectivamente.

Opuesto a los modelos discretos, están principalmente dos modelos continuos, el uno viene de lo que se conoce como teoría de la evaluación, y el otro como la teoría dimensional del afecto. Psicólogos como Richard Lazarus y Craig Smith son dos de los principales contribuyentes a la teoría dimensional. Ellos buscan definir emociones como el resultado de la evaluación de una situación. La definición de la emoción en la teoría de evaluación toma en cuenta aspectos como lo sorpresivo de una situación, la importancia de la meta, el control que se pueda tener, la energía disponible, entre muchos aspectos. Algunos proponen entre 5 y 16 variables [91]. Recientemente, el trabajo relacionado con las teorías discretas ha sido enfocado a encontrar soporte desde la teoría de la neurociencia, y también desde el área de la dinámica de sistemas [92].

El segundo punto de vista tiene que ver con los modelos continuos de las emociones, y es conocido como la teoría dimensional. Esta es la teoría seleccionada para emular las emociones en este trabajo. La teoría dimensional fue propuesta por Russel en 1980, cuando midió los estados emocionales de humanos, por medio dos variables, valencia y activación. La valencia indica si la experiencia emocional es positiva o negativa, como alegría e ira. De otra parte, la activación va del rango más activado a menos activa, dos extremos pueden ser excitación y aburrimiento [93].

Cuando Rusell midió los estados emocionales, representó los resultados en un plano cartesiano, con valencia en el eje horizontal y activación en el vertical. Resultó que las emociones ocupan una región circular, lo que le da nombre al modelo, el modelo circumplejo del afecto. Por ejemplo alegría (figura 106) es la combinación de una valencia positiva y un nivel medio de activación. Tristeza, de otra parte, es altamente negativa en valencia y baja en activación. Aun cuando esta teoría fue propuesta desde la psicología, en los últimos años ha tenido soporte desde la neurociencias [94].

Figura 106. Modelo circumplejo de las emociones

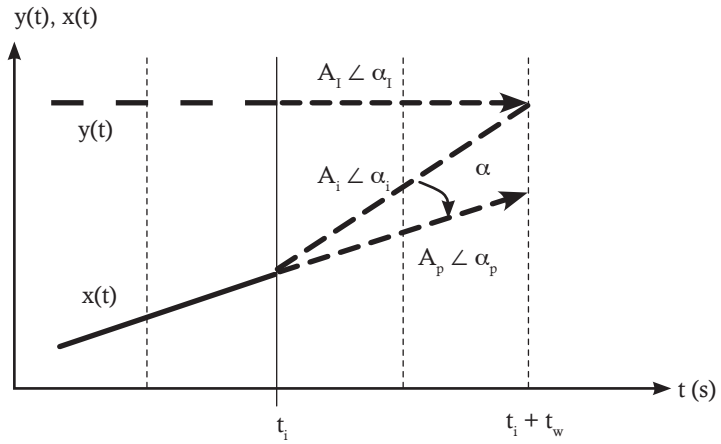


El modelo circumplejo del afecto ha servido de base para contribuciones desde la medicina, psicología, análisis del lenguaje, música, entre otras áreas. Trabajos recientes incluyen la medida de los estados emocionales, para cambiar como respuesta el color de la pantalla de un teléfono [95], y la medida del estado emocional de una persona durante un video juego [96]. Otro trabajo, en robótica, es el diseño y construcción del robot llamado EDDIE (del inglés *Emotion-Display with Dynamic Intuitive Expressions*). Este robot regula la acción delos servomotores, los cuales controlan el movimiento de las orejas, ojos y boca para la generación de estados emocionales [97]. Basado en el análisis de estas aplicaciones, se concluye que la teoría dimensional puede ser usada para construir una estrategia de control basada en emociones.

Arquitectura del control basado en emociones

Se propone la definición de un estado emocional, con relación al modelo circumplejo de las emociones, por medio del ángulo α. La anticipación es el núcleo de la estrategia de control propuesta, donde la diferencia entre la predicción de la posición futura de la planta y la predicción de la posición del modelo de referencia, permite definir el estado emocional α (figura 107). En esa figura, $x(t)$ representa el movimiento de la planta, mientras $y(t)$, el del modelo de referencia. El instante actual es t_i, y la ventana de tiempo de predicción es t_w.

Figura 107. Definición del estado emocional α



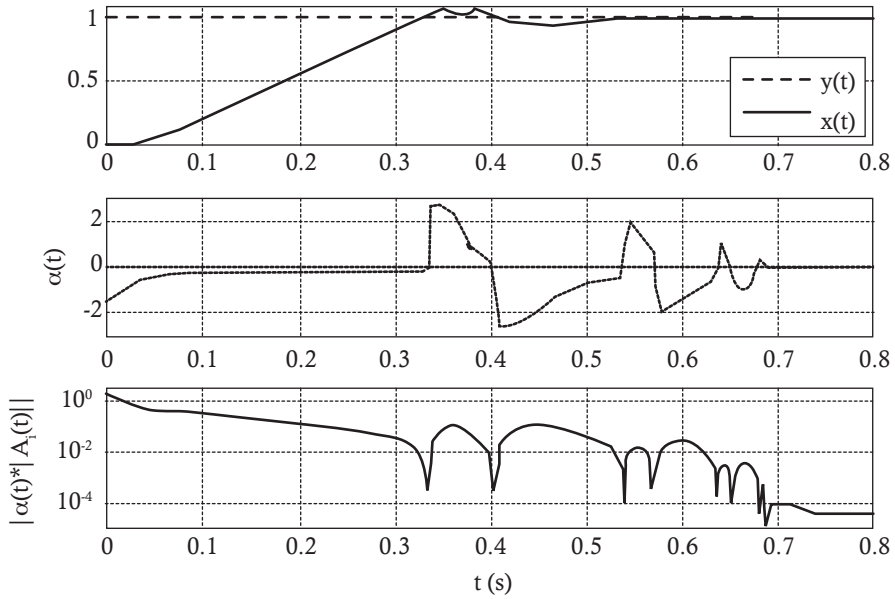
La anticipación en la estrategia de control consiste en calcular el valor futuro de la dinámica del modelo de referencia, y lo mismo para la planta, para lo cual se usa la tasa de cambio en el instante t_i (actual), se supone que la dinámico no presentará cambios, por lo cual la predicción puede escribirse como $y(t_i+t_w) = y(t_i) + mt_w$, donde m es la tasa de cambio en el instante actual. El tiempo t_w puede ser un múltiplo del tiempo de muestreo que se utilice. Por ejemplo, la figura 107 muestra $t_w = 2dt$.

Además de la definición del estado emocional, α, se requiere la definición de la intensidad del estado emocional. El conjunto entre estas dos constituye lo que se conoce en control tradicional como error. La magnitud ideal, A_i en la figura 107, es la intensidad de la emoción. Así, entre más grande sea la diferencia entre la planta y el modelo de referencia, mayor será la intensidad de la emoción.

Una vez se ha calculado el estado emocional y la intensidad del estado emocional, se pondera este resultado, de forma proporcional, y también de manera proporcional a la acumulación de esta definición nueva de error "emocional", como sigue: $u = KpA_i\alpha + Ki\int A_i\alpha dt$. La parte proporcional, $KpA_i\alpha$, controla el error dinámico instantáneo, mientras que la componente integral, $Ki\int A_i\alpha dt$, puede verse como la medida de la inercia del sistema, y sirve para controlar el sistema con base en la historia de la dinámica del sistema.

Los componentes del controlador basado en emociones, en forma gráfica, se observan en la figura 108. En esa figura se supone un modelo de referencia muy simple, este es $y = 1$, con el propósito de observar el efecto de la definición emocional exclusivamente. De otra parte, se utilizará la planta $H(s) = 100/(s^2 + 10s + 100)$. Los parámetros de controlador son los siguientes: tiempo de muestreo $dt = 1\times10^{-3}$, ventana de predicción $t_w = 2dt$, ganancias $Kp = 1\times10^4$ y $Ki = 1\times10^5$. Una característica importante del estado emocional α es que este decrece con el error, de tal manera que el sistema dinámico tiene a un estado emocional de calma, como se explicará en la sección siguiente. Además, el estado emocional, α, detecta el signo del error. Así, el estado emocional, α, puede ser positivo o negativo. La simulación en la figura 108 muestra como $A_i\alpha$ se reduce a medida que el tiempo pasa.

Figura 108. Rendimiento de la arquitectura de control nueva

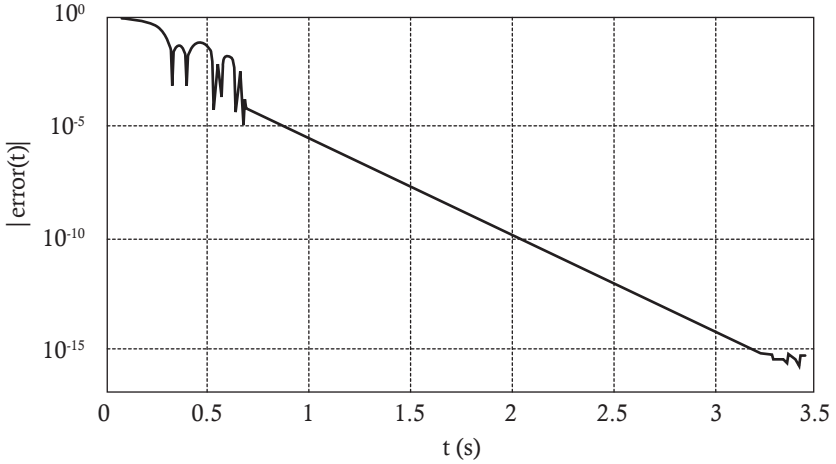


El error, $x_r - x_p$, donde x_r es el desplazamiento del modelo de referencia, mientras x_p es el desplazamiento del modelo de la planta, en la figura 109 sirve para mostrar la estabilidad que se produce al utilizar el controlador. El error decrece exponencialmente hasta alcanza el mínimo que Matlab tiene para el cálculo de operaciones a punto flotante, y esto lo hace luego de tres segundos de tiempo simulado.

Definición de los estados emocionales para el controlador basado en emociones

Generalmente, tareas complejas requieren un estado emocional con un bajo nivel de activación. Este hecho es conocido como la ley Yerkes-Dodson [98]. Por esto se define $\alpha = 0$, como el estado emocional ideal, equivalente a la calma, de acuerdo con la ley enunciada, y al modelo circumplejo de las emociones.

Figura 109. Demostración experimental de estabilidad.



Dado que es muy ambicioso definir todos los estados emocionales que experimenta un humano, para tareas de control como las expuestas en este capítulo, entonces se utilizará sólo una parte de todos estos estados emocionales, los cuales se considera pueden ayudar a realizar las tareas de control requeridas. En particular, desordenes emocionales, tales como estrés, fobias, manías, y otras, son excluidas. De la misma manera, comportamientos más complejos, los cuales pueden incluir la mezcla de varios estados emocionales, tampoco serán emulados.

Una manera de visualizar un estado emocional en un controlador, corresponde a imaginar lo que experimentaría una persona que lleva a cabo la misma tarea del controlador. Una persona puede experimentar alegría cuando encuentra un resultado positivo, dadas las acciones de control que está tomando, de otra parte, la persona puede experimentar temor o ira cuando el resultado es contrario. Estas emociones pueden ser caracterizadas en términos del estado emocional α como se muestra en la tabla 17. Esta tabla incluye la definición de los estados emocionales más representativos; desde los más positivos, como calma, hasta llegar al más negativo, la ira. Para simplificar la presentación, se asume que el estado emocional del controlador depende exclusivamente del valor instantáneo del estado emocional α y de la intensidad del estado emocional.

El estado emocional α será utilizado para tomar decisiones en cuanto a hacia dónde debe dirigirse la dinámica de un sistema bajo control. Esto, por medio de la salida del controlador, también llamada variable de corrección, u, la cual se aplica a la planta, bien sea directamente, o a través de una etapa de potencia.

Tabla 17. Definición de los estados emocionales

Emoción	Definición
Calma	El estado emocional $\alpha = 0$, se ha alcanzado
Satisfacción	La meta, $\alpha = 0$, no ha sido alcanzada aún, pero se está en esa dirección ($\alpha = \pi/6$).
Alegría	La distancia entre el modelo de referencia y la planta decrece, y de seguir esta tendencia, eventualmente será cero ($\alpha = \pi/3$).
Excitación	La planta oscila alrededor de la referencia, sin incrementar el valor de la intensidad de la emoción ($\alpha = \pi/2$).
Temor	La distancia entre el modelo de referencia y el sistema aumenta ($\alpha = 3\pi/4$).
Ira	Este es el peor caso posible. El sistema se dirige en sentido contrario al modelo de referencia, y por tanto, de seguir así, nunca se alcanzará la meta ($\alpha = \pi$).

Caracterización en la frecuencia del controlador basado en emociones

Esta sección compara el rendimiento del controlador basado en emociones, con un controlador tradicional PI. Este estudio se enfoca en el análisis del comportamiento en la frecuencia para los dos controladores, cuando están expuestos a restricciones iguales. El análisis en la frecuencia calcula el efecto antes los cambios de amplitud y frecuencia de la referencia, sobre el error del controlador, y también sobre la señal u, utilizada para medir el esfuerzo que hace el controlador. Una medida tradicional del error utiliza la diferencia entre la señal de referencia y la salida de la planta, una vez esta salida ha sido transformada por el sensor. Dado el controlador emocional, además se analizará otro error, el emocional, como resulta del producto entre el estado emocional y la intensidad de la emoción, en las ecuaciones 6.1 y 6.2, las cuales se basan en la figura 107.

$$\alpha = a\tan\left(\frac{y\left(t_i + t_w\right) - x\left(t_i\right)}{tw}\right) - a\tan\left(\frac{x\left(t_i + t_w\right) - x\left(t_i\right)}{t_w}\right) \tag{6.1}$$

$$A_i = \sqrt{\left(y\left(t_i + t_w\right) - x\left(t_i\right)\right) + t_w^2} \tag{6.2}$$

Basado en las ecuaciones anteriores, el indicador de la evolución del error emocional corresponde a la integral del valor absoluto del error (*ITAE*), como se define en la ecuación 6.3.

$$\text{ITAE definición emocional} = \int_0^{20} t\left|\alpha A_i\right| dt \tag{6.3}$$

El índice *ITAE* calcula el comportamiento del error, dado un rango de tiempo, y reduciendo todo el comportamiento a un solo número, penalizando en especial el en función del tiempo en el cual se presenta el error. La descripción del índice *ITAE* para la definición tradicional del error se encuentra en la ecuación 6.4.

$$\text{ITAE definición tradicional} = \int_0^{20} t\left|y - x\right| dt \tag{6.4}$$

Como ya fue dicho, en adición al análisis del error, el cual corresponde a cuán bueno es el controlador para dirigir la planta, se observará el esfuerzo que hace el controlador. Este esfuerzo es resumido en la integral del valor absoluto de la señal de actuación, *IAU*, como se define en la ecuación 6.5. Una definición tradicional de *IAU* no incluye la constante *b*, pero dado que el valor estable del sistema, bajo una referencia senoidal, es oscilatoria, entonces, esa constante se utiliza para retirar el promedio, y distinguir de manera más clara el efecto del controlador. Esta constante *b* corresponde al valor de la señal de actuación que produce error nulo a una referencia constante, cuando esta referencia es igual al valor promedio de la señal senoidal de referencia.

$$IAU = \int_0^{20} t\left|u - b\right| dt \tag{6.5}$$

En adición a los tres indicadores en las ecuaciones 6.3 a 6.5, se utiliza un índice adicional, este es la medida en decibeles, como se define en la ecuación 6.6. Este índice asocia la amplitud de la asociación de salida con la amplitud de la oscilación de la referencia. Esta medida permite calcular el ancho de banda del sistema de control.

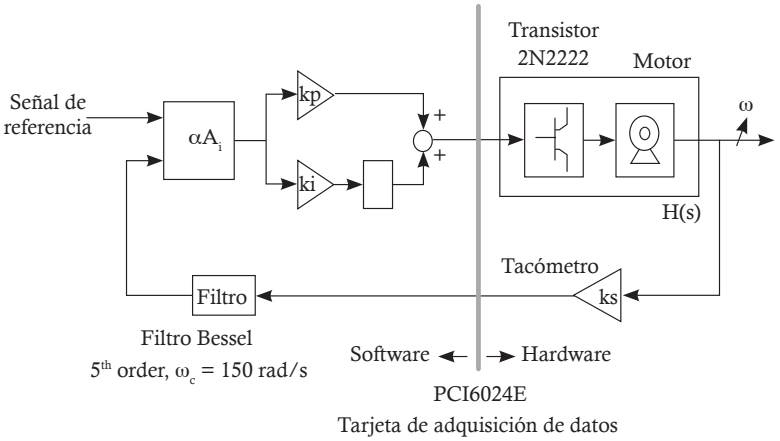
$$decibeles = 20\log\left(\frac{Aout}{Ain}\right)[db] \tag{6.6}$$

La planta de prueba en esta sección es un motor de corriente directa, como el que utiliza la plataforma robótica en cada rueda, y según se muestra en la figura 110. Esta figura deja ver la configuración de laboratorio. El proceso para obtener la planta usa voltajes provenientes de un transistor y la velocidad en el motor que resulta. Estos datos sirven como entrada programa de identificación de sistemas de Matlab. El modelo resultante es $H(s) = 7470/(s^2 + 14s + 37)$ (rad/s)/V. Otras características del motor son: voltaje nominal de 12 V, con 4.3 W de potencia, velocidad nominal de 4060 rpm, y torque nominal de 10 mNm, con referencia comercial 1.13.78.xxx de Bühler.

La salida del sensor de velocidad, un tacómetro, es una señal senoidal con frecuencia y amplitud variable de acuerdo con la velocidad de giro. El acondicionamiento de la señal comienza con un puente rectificador de onda. El segundo paso es el filtrado mediante un circuito RC. Finalmente, la señal pasa por un filtro pasabajos

en Simulink, como se muestra en la figura 110. El modelo del tacómetro puede ser resumida a una constante *ks* = 0.006 V/(rad/s).

Figura 110. Montaje para evaluar el rendimiento del controlador emocional



El análisis del comportamiento en la frecuencia utiliza la señal de referencia $r = A\text{sen}(\omega t) + dc$. El valor *dc* es igual a 5.5, el cual corresponde a la mitad del rango de entrada del motor. De igual manera, *a* cambia de 0.5 a 2.5, para cubrir todo el rango de entrada del motor. El rango de frecuencia comienza en 0.628 rad/s y va hasta 100 rad/s. El límite inferior garantiza tener dos ciclos de oscilación en 20 *s* de experimentación, mientras que el límite alto corresponde a la frecuencia máxima a la cual el tacómetro puede transmitir datos por la tarjeta de adquisición de datos. Las ganancias *kp* y *ki* en la figura 110 fue definida como 2 y 5, respectivamente, según propone un algoritmo de auto sintonización en Matlab. El resultado de la caracterización de frecuencia se muestra en la figura 111.

La columna izquierda en la figura 111 muestra las simulaciones del análisis del comportamiento en la frecuencia. En la columna derecha se muestra el resultado de la experimentación en laboratorio. Cada simulación y su contraparte en laboratorio usa un tiempo de muestre de 2 ms, y una ventana de predicción de 10 ms. Cada experimento se corre tres veces. Así el resultado final en la figura corresponde al promedio. Los resultados en la primera fila muestran el análisis para la definición emocional del error, mientras que la segunda fila muestra los resultados para la definición tradicional del error. El color claro representa el control tradicional PI, y el oscuro representa el resultado con el controlador basado en emociones.

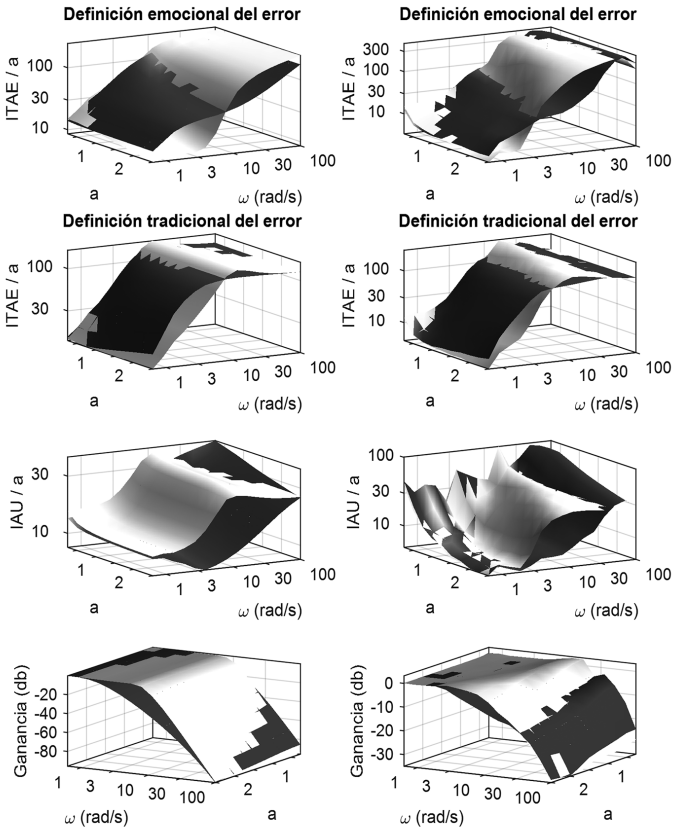
Una primera observación de los resultados en la figura 111 contrasta toda la columna izquierda con la derecha, en otras palabras, simulaciones versus experimentos. Se observa una similitud entre las dos columnas, lo cual garantiza que el modelo de la planta y el modelo del sensor capturan apropiadamente la dinámica evaluada, y al tiempo sirve para garantizar la veracidad de los datos experimentales.

El análisis de las dos primeras filas de la figura 111, correspondiente al índice *ITAE* muestra una dependencia cuasi lineal entre el índice y la amplitud, como

se espera para plantas lineales. Además, estas gráficas muestran que el incremento de frecuencia hace que incremente el valor del índice, lo cual significa que mayores valores de frecuencia implican mayores errores para ambos controladores. Una tercera observación se refiere a la comparación entre controladores. El controlador PI tradicional resulta mejor a bajas frecuencias, mientras que el controlador emocional, en general, resulta mejor a altas frecuencias.

El diagrama de Bode de magnitud (la cuarta fila de la figura 111) muestra el corte de frecuencia, el cual sirve para dividir frecuencias bajas de frecuencias altas, y este resultado es alrededor de 5 rad/s para PI tradicional, mientras es 3 rad/s para el emocional. El valor bajo para el controlador emocional contrasta con el valor a frecuencias altas, donde el controlador emocional resulta mejor. Como se concluyó en el párrafo anterior, el controlador emocional resulta mejor a frecuencias altas. De otra parte, en general, el controlador basado en emociones dirige la planta con menos esfuerzo de lo que lo hace el control tradicional, como indica valores más bajos del índice *IAE*.

Figura 111. Caracterización en la frecuencia



Modelo de la plataforma diferencial

El modelo del robot DaNI 2.0 realizado para probar las estrategias de control tiene una parte dinámica y una parte cinemática. Se comenzará describiendo la parte cinemática. Esta define la posición del robot, y el ángulo de dirección θ, con respecto al plano XY, según se muestra en las ecuaciones siguientes, en las cuales $x(t)$, $y(t)$, $\theta(t)$, son las coordenadas de posición en el instante de tiempo t; $vL(t)$ y $vR(t)$ son las velocidades lineales derecha e izquierda, respectivamente, de las ruedas del robot; de otra parte x_0, y_0, θ_0 son las coordenadas iniciales.

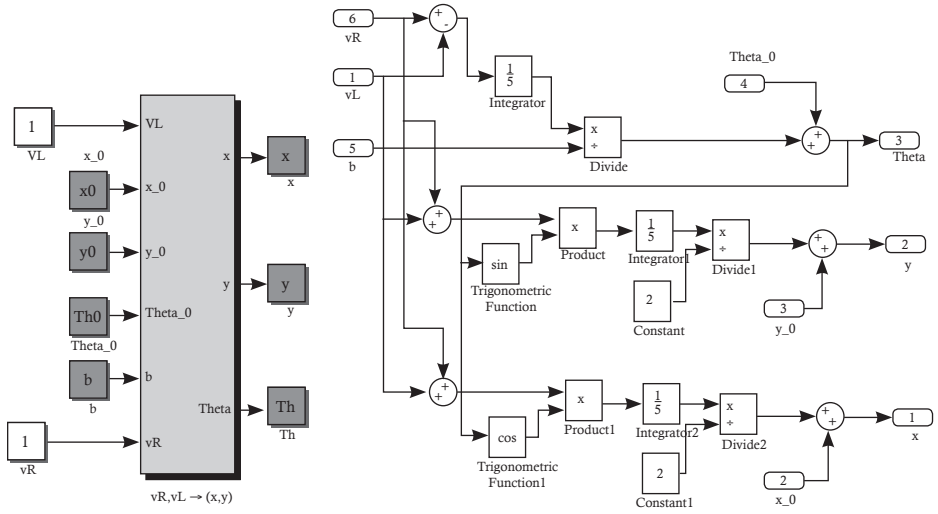
$$x(t) = x_0 + \int \left(vL(t) + vR(t) \right) \cos(t)(\theta)\, dt \tag{6.7}$$

$$y(t) = y_0 + \int \left(vL(t) + vR(t) \right) \operatorname{sen}(t)(\theta)\, dt \tag{6.8}$$

$$\theta(t) = \theta_0 + \frac{1}{b} \int \left(vL(t) - vR(t) \right) dt \tag{6.9}$$

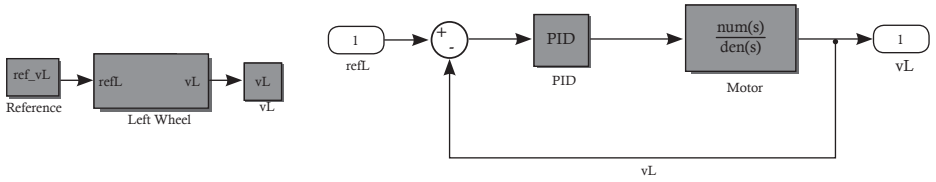
La figura 112 muestra la implementación de estas ecuaciones en Simulink. La parte izquierda muestra las entradas del modelo, es decir, las velocidades lineales de las ruedas de la plataforma, más las condiciones iniciales, según definen las ecuaciones. El lado derecho de la figura presenta la implementación en bloques de las ecuaciones 6.7 a 6.9.

Figura 112. Modelo del robot en Simulink



La segunda parte del modelo, la parte dinámica, consiste en una idealización del comportamiento de los motores que mueven la plataforma. Estos son motores de corriente directa, y por facilidad se supondrán exactamente iguales. Si bien pudo haberse utilizado un algoritmo de identificación, lo que se hizo fue observar la respuesta de cada motor; observando el tiempo que tarda el motor en pasar de reposo a alguna referencia de velocidad, lo cual indica la constante de tiempo del sistema, de otra parte, para identificar la ganancia del sistema se observó la velocidad aproximada que sale de tratar las señales de los encoders incrementales, acoplados a cada motor. Se sabe además que cada motor tiene un controlador PID sintonizado para regular la velocidad, por lo cual se simula este controlador también, como se indica en la figura 113.

Figura 113. Modelo de los motores del robot

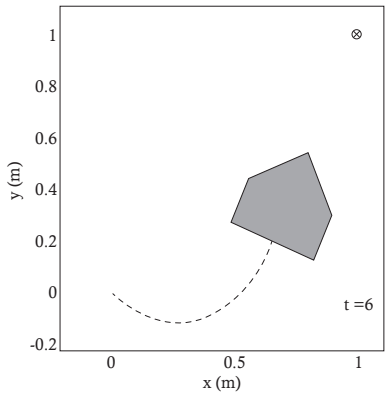


La función de transferencia del motor está indicada en la ecuación 6.10. Las ganancias del controlador PID son: ganancia proporcional $Kp = 1$, ganancia integral $Ki = 1$, ganancia diferencial $Kd = 0$.

$$H(S) = \frac{6{,}93}{s^2 + 5.26s + 6.93} \tag{6.10}$$

De esta forma se puede emular el comportamiento del robot. Código adicional en Matlab permite capturar las coordenadas y visualizar el movimiento de la plataforma, como se muestra en la figura 114. En esa figura se parte de la coordenada (0,0,-π/4), y se agrega la meta de ir a (1,1), de otra parte, se muestra una instantánea del recorrido a los 6 segundos.

Figura 114. Simulación del movimiento del robot



Control clásico del robot

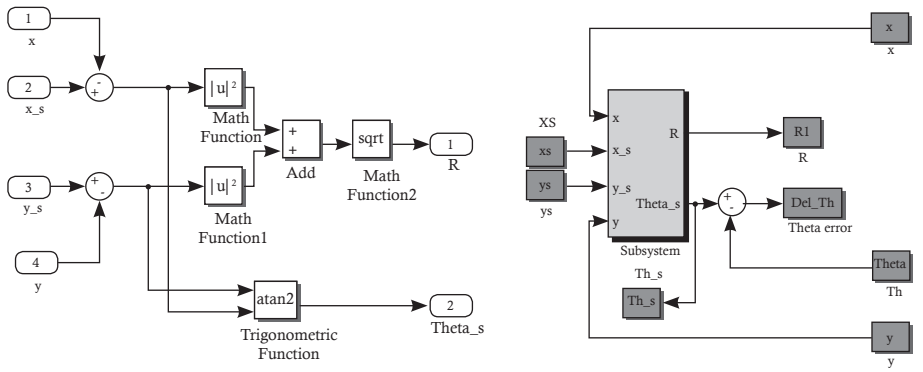
En esta sección se explica la forma de emular la implementación de un control clásico, es decir, PID, para la plataforma robótica. Esta explicación servirá de base para exponer el control emocional, objeto final de este capítulo. Los objetivos de control utilizados en la estrategia de control de posición del robot implica observar la distancia entre el robot y una meta, conocida como radio, según indica la ecuación 6.11; de otra parte, se observa la diferencia entre el ángulo de la línea directa de vista entre el robot y la meta, con el ángulo que en realidad tiene el robot, según se indica en la ecuación 6.12.

$$R = \sqrt{\left(x - xs\right)^2 + \left(y - ys\right)^2} \tag{6.11}$$

$$\Delta\theta = a\tan\left(\frac{y - ys}{x - xs}\right) - \theta \tag{6.12}$$

En la ecuación 6.11 *xs* se refiere a la coordenada *x* de la meta, mientras que *ys* se refiere a la coordenada y de la meta. La forma de implementar estas ecuaciones en Simulink se presenta en la figura 115. Véase que la evaluación del ángulo sólo incluye la evaluación de la función arco tangente, la diferencia con el ángulo se presenta en el diagrama de bloques a la derecha de la figura 115. El propósito del control es hacer que el radio, *R*, sea cero, y a su vez el delta de ángulo, Δθ, debe llegar a ser cero. Estos dos valores en condiciones ideales, en el control de la planta real lo que se busca es que esas distancias se aproximen a cero.

Figura 115. Evaluación del radio y ángulo de dirección del robot

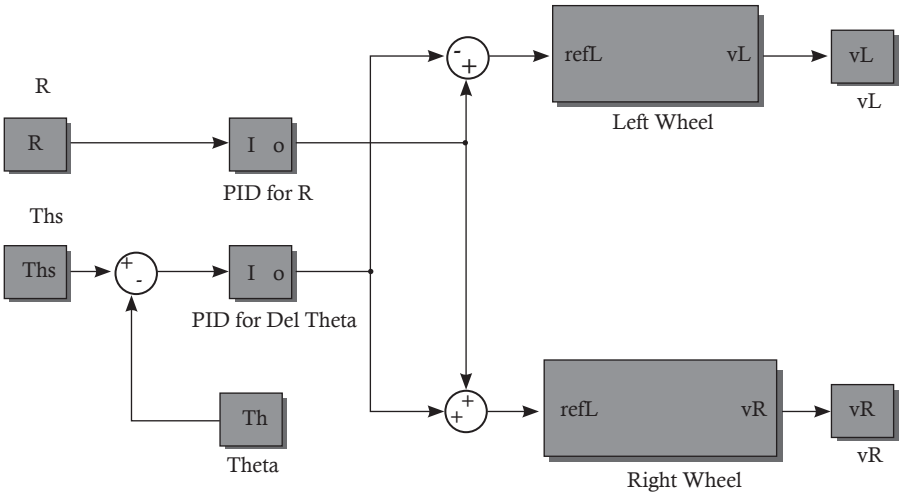


Supóngase ahora una condición ideal, esto es que Δθ es cero, entonces el control (la reducción del radio), consiste en dirigir al robot hacia adelante. Las variables con que se cuentan para manejar el robot son sólo dos, estas son las velocidades de las ruedas del robot, las cuales están comandadas por los motores en la figura 113. Así entonces, si la lectura del radio se aplica a un controlador PID, como entrada, la salida de este

controlador indicará la referencia de velocidad de las dos ruedas. En otras palabras, la misma corrección alimenta cada rueda.

La corrección del ángulo implica que la corrección que se dé a cada rueda sea diferente. Si se quiere que la plataforma gire en sentido horario, la forma más rápida es indicarle a la rueda izquierda que avance, mientras se le indica a la rueda derecha que retroceda, es decir, velocidad negativa. De la misma manera, si se quiere que el robot gire en sentido contra horario, entonces la rueda derecha debe ir hacia adelante, mientras que la rueda trasera debe ir hacia atrás. Como conclusión, la corrección que provenga del controlador que regula el ángulo debe ir positiva para una rueda y negativa para la otra, según se muestra en la figura 116.

Figura 116. Configuración de los dos controladores para manipular al robot

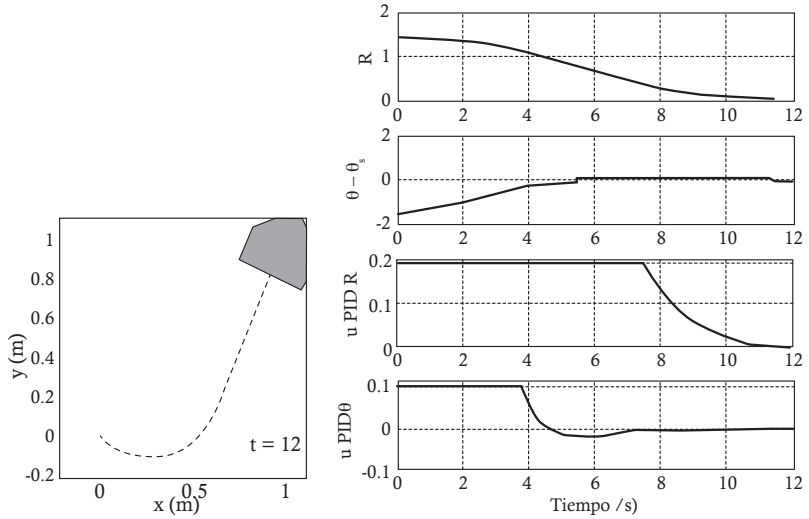


Supóngase una condición ejemplo para mostrar el trabajo del control durante la operación del robot. Si la condición inicial es <0,0,-π/4> y el objetivo es <1,1,π/4>. El recorrido del robot se presenta en la parte izquierda de la figura 117. Véase cómo luego de 12 segundos, se ha alcanzado el destino trazado en un comienzo. En la parte derecha se encuentra el comportamiento de las variables a controlar y de la salida de los controladores. Primero se verá la figura superior, el radio *R*. Esta distancia inicial es √2, correspondiente al radio entre el punto inicial y la meta. A medida que el robot avanza, esta distancia disminuye, y luego de unos 11 segundos es cercana a cero. Es importante agregar un condicional que indique que si se da esta condición, el robot muy cerca de la meta, entonces el robot se debe detener.

La segunda parte de la parte derecha de la figura 117 muestra el comportamiento del ángulo. Véase que el error inicial es igual a -π/4 y la actuación de los controladores lleva este valor a aproximarse a cero, como se espera. La señal *R* y la diferencia de ángulo son las entradas a dos controladores PID. La observación de la señal *u* PID *R*, que es la salida del controlador PID que regula el radio se mantiene en 0.2 durante algo más de 7 segundos. Esto porque un condicional, no mostrado en la figura 116

hace que la salida máxima sea 0.2. Es decir, se programa una saturación. Esa saturación también ocurre en el PID que regula el ángulo.

Figura 117. Control de posición



El código necesario para correr esta simulación se presenta en la siguiente tabla.

Tabla 18. Código Matlab para simular el robot, parte a

```
%Danilo Rairán, Mar/18/2015
clear, clc, close all
x0 = 0; %Initial x position
y0 = 0; %Initial y position
Th0 = -pi/4; %Initial angle
b = 0.37; %separation between wheels
Ts = 1e-3; %Sampling time
ft = 12; %Stop time
Rlim = 25e-3; t = 0:Ts:ft; %Time
xs = ones(1,length(t)); ys = ones(1,length(t)); xs = [t;xs]'; ys = [t;ys]';
%Figure Parameters
n = 1/50; %One point every # points, for Simulation. 1/3 is one each three.
p = 0.001; %Pause
ti = 0; %Initial Time to be Presented
tf = ft; %Final time to be Presented
tw = ft; %Visible part of the signals. Last tw seconds
%PID to control radio R
kp_PID_R = 1; ki_PID_R = 0.0; kd_PID_R = 1; maxI_PID_R = 0.2; minI_PID_R = 0;
maxOut_PID_R = 0.2; minOut_PID_R = 0; %PID to control theta angle
kp_PID_Theta = 1.0; ki_PID_Theta = 0.0; kd_PID_Theta = 0.5; maxI_PID_Theta = 0.1;
minI_PID_Theta = -0.1; maxOut_PID_Theta = 0.1; minOut_PID_Theta = -0.1;
%Left Wheel
kp_PID_vL = 1; ki_PID_vL = 1; kd_PID_vL = 0; num_vL = 1; den_vL = [0.2 1];
%Rigth Wheel
kp_PID_vR = 1; ki_PID_vR = 1; kd_PID_vR = 0; num_vR = 1;
```

Tabla 19. Código Matlab para simular el robot, parte b

```
den_vR = [0.2 1]; sim('robot_model_4')
%Resampling of all data
sample = 1:1/n:((ft/Ts) + 1);
t = t(sample); x = x(sample); y = y(sample); Th_s = Th_s(sample); R = R(sample);
PID_R_out = PID_R_out(sample); PID_Th_out = PID_Th_out(sample);
refL = refL(sample); refR = refR(sample); vL = vL(sample); vR = vR(sample);
Th = Th(sample); xs = xs(sample,2); ys = ys(sample,2); eTh = (Th_s - Th)*(180/pi);
Ts = Ts*(1/n); ti = round(ti/Ts)*Ts; tf = round(tf/Ts)*Ts; tw = round(tw/Ts)*Ts;
Xmin = min([x;y;xs;ys]) - (1/10)*(max([x;y;xs;ys]) - min([x;y;xs;ys]));
Xmax = max([x;y;xs;ys]) + (1/10)*(max([x;y;xs;ys]) - min([x;y;xs;ys]));
Ymin = min([x;y;xs;ys]) - (1/10)*(max([x;y;xs;ys]) - min([x;y;xs;ys]));
Ymax = max([x;y;xs;ys]) + (1/10)*(max([x;y;xs;ys]) - min([x;y;xs;ys]));
rel_hw = (Ymax - Ymin)/(Xmax - Xmin); scrsz = get(0,'ScreenSize');
if rel_hw<scrsz(4)/scrsz(3)
w_w = scrsz(3)*0.8; %This is to see equeal axes
w_h = 2*0.5*w_w*rel_hw;
else
w_h = scrsz(4)*0.8; %This is to see equeal axes
w_w = 2*0.5*w_h/rel_hw;
end
H = figure(1); clf, set(H,'position',[scrsz(3)*0.05 scrsz(4)*0.05 w_ww_h])
% set(H,'position',[scrsz(3)*0.05 scrsz(4)*0.05 700 700])
axis_1 = [XminXmaxYminYmax]; axis(axis_1)
f = get(gcf,'position'); a = get(gca,'DataAspectRatio');
ks = (a(2)*f(3))/(a(1)*f(4)); %k by screen size and axis definition
```

Tabla 20. Código Matlab para simular el robot, parte c

```
if ti == 0, ti = Ts; end
if tf>ft, tf = ft; end
if ti< 0, ti = 0; end
if ti>tf, ti = 0; tf = ft; end
if tw< 0, tw = ft; end
if tw>ft, tw = ft; end
j = 1;
for i = ti/Ts + 1:(tf/Ts) + 1
clf, ini_p = round(i - tw/Ts);
if (i - tw/Ts) < 1, ini_p = 1; end
figure(1), hold on
plot(xs(ini_p:i),ys(ini_p:i),'r','linewidth',3), plot(x(ini_p:i),y(ini_p:i),'b--','linewidth',2)
antTh = atan2((y(i) - y(i - 1)),(x(i) - x(i - 1)));
xAntPos = x(i) + (b/2)*cos(antTh); yAntPos = y(i) + (b/2)*sin(antTh);
[X,Y] = robotPosition(x(i),y(i),Th(i),b,1); patch(X,Y,[0.5 0.5 1.0]), alpha(0.95)
line([x(i) xAntPos],[y(i) yAntPos],'Color','g'), line([x(i) X(3)],[y(i) Y(3)],'Color','g')
plot(xs(i),ys(i),'xk','markersize',10,'linewidth',2)
plot(xs(i),ys(i),'ok','markersize',10,'linewidth',2)
axis(axis_1), box on, hold off
text(Xmax,0,['t = ',num2str(t(i),2),''],…
'VerticalAlignment','Top','HorizontalAlignment','Right','FontWeight','bold','FontSize',24);
grid off, xl1 = xlabel('x (m)'); yl1 = ylabel('y (m)');
set(gca,'FontSize',24), set(xl1,'FontSize',24), set(yl1,'FontSize',24), set(gcf,'Color','w')
box on, draw now, pause(p)
end
```

El código en la tabla 18 corresponde a la inicialización de variables, dentro de ellas los valores de cada PID. La tabla 19 toma los resultados de correr el modelo en Simulink, y los re-muestrea para facilitar la presentación de los resultados. La tabla 20 contiene la presentación, a manera de animación de la simulación del movimiento del robot.

Modelo de referencia

Un enlace entre el control tradicional, ya presentado, y el controlador basado en emociones es el modelo de referencia. La idea de este modelo es no presentar una única coordenada final, como meta, de la forma que se ha hecho hasta ahora, sino por el contrario, presentar durante cada instante de tiempo cuál debería ser la posición ideal de la plataforma. Es equivalente a presentar una meta cada instante de tiempo, en forma dinámica, desde la condición inicial y moviéndose hasta el punto final. Este modelo de referencia facilita el trabajo de control, porque la distancia entre el robot y el modelo siempre es pequeña y lo mismo con el ángulo. La idea es presentar metas pequeñas al controlador, en lugar de una sola.

El modelo de referencia no solo consiste en una secuencia de pequeñas metas enlazadas, sino que ésta tiene en cuenta la dinámica del robot. Así entonces, para llegar a la meta, presenta transiciones suaves, realizables por el robot. Un trabajo importante dentro de esta investigación fue la evaluación de varias fuentes sigmoidales, con el fin de determinar la que presente transiciones suaves, y a su vez la menor distancia recorrida.

En esta sección se presentará el resultado de un modelo de referencia, el cual resulte exigente para la plataforma robótica, y por lo tanto es útil para medir las capacidades de control de cualquier algoritmo de control. Se trata de la curva descrita por lo que se conoce como curva de Lissajous. Esta curva resulta ideal porque exige al robot curvaturas a la izquierda y a la derecha, con ángulos cerrados y abiertos, y de otra parte velocidades variables, lo cual emula el comportamiento normal de una plataforma robótica. El modelo se describe en las ecuaciones 6.13 y 6.14. En estas ecuaciones T representa el periodo, el cual será utilizado como 14π.

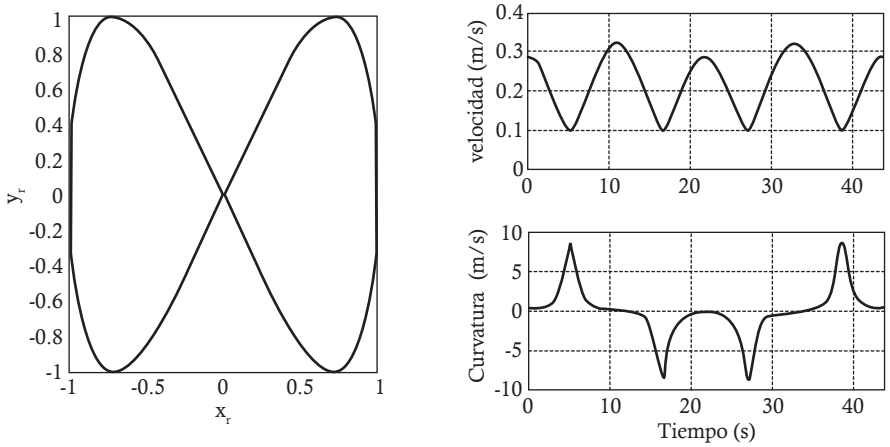
$$x_r(t) = \cos\left(\frac{2\pi}{T}t\right) \tag{6.13}$$

$$y_r(t) = \sin\left(2\frac{2\pi}{T}t\right) \tag{6.14}$$

La curva izquierda en la figura 118 es la referencia dinámica para el robot. El propósito es probar el comportamiento de las estrategias de control ante velocidades variables a radios de curvatura variable. La curva comienza en $t = 0$ en la coordenada <1,0>, comienza a avanzar en sentido contrario a las agujas del reloj, hasta llegar al punto de origen. La parte derecha de la figura 118 muestra la velocidad, es máxima cuando el radio de curvatura es mínima, y viceversa.

La velocidad, dadas las componentes *x* y *y* de la posición está dada en la ecuación 6.15. La variable $\dot{x}_r$, en la ecuación 6.15, es la componente en *x* de la velocidad, e igual a la derivada de la posición, en *x*. La señal $\dot{y}_r$ es la componente en *y* de la velocidad. El radio de curvatura, *k*, se muestra en la figura 118 y se calcula mediante la expresión en la ecuación 6.16. Las curvas a la izquierda presentan radio de curvatura positivo, mientras las curvas a la derecha presentan radio negativo.

Figura 118. Modelo de referencia



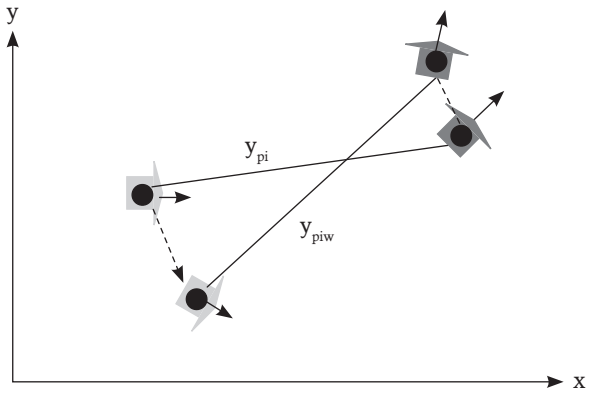
$$\left|\dot{r}\right| = \sqrt{\left(\dot{x}_r\right)^2 + \left(\dot{y}_r\right)^2} \tag{6.15}$$

$$k = \frac{\left|\dot{r} \times \ddot{r}\right|}{\left|\dot{r}\right|^3} \tag{6.16}$$

Estrategia de control basado en emociones aplicada al robot

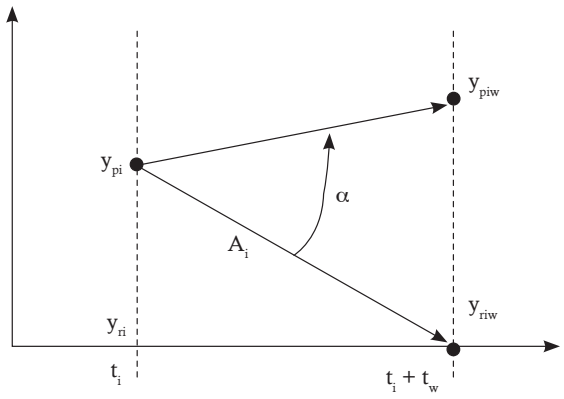
Para simplificar la explicación de esta estrategia, se presenta un ejemplo, en el cual se regula el radio entre el robot y la posición del modelo de referencia. En cada instante de tiempo el robot y el modelo de referencia ocupan una posición en el espacio. La distancia en el instante actual de observación es llamada y_{pi}, mientras la distancia estimada alguna ventana de tiempo posterior es y_{piw}, para calcular esta distancia, y_{piw}, se estima la posición nueva para el robot, y también se estima la posición que tendría el modelo de referencia. El robot está a la izquierda en la figura 119, mientras el modelo de referencia está a la derecha. El subíndice *p* en las dos distancias indica que es la medida referente a la planta.

Figura 119. Robot y modelo de referencia



Ahora, observando el movimiento del robot y del modelo de referencia, en el tiempo, resulta el comportamiento en la figura 120. En ella t_i es el instante de tiempo actual. En ese instante de tiempo la distancia de la planta al modelo de referencia es y_{pi} mientras la distancia ideal es cero, es decir, $y_{ri} = 0$. La clave de la emulación de emociones por parte del comportamiento de un sistema dinámico consiste en anticipar lo que va a ocurrir con el sistema, y basado en la diferencia entre la anticipación, y la anticipación de lo que realmente está pasando, entonces se tiene un estado emocional. Así, anticipando la posición del robot y del modelo de referencia, es posible determinar que la distancia a que se llegará luego de una ventana de tiempo será y_{piw}, mientras que la distancia ideal anticipada es y_{riw}, la cual se mantiene en cero por cuanto el objetivo de control no cambia. El ángulo α en la figura 120 representa el estado emocional, y la magnitud entre el punto y_{pi} y y_{riw} representa la intensidad del estado emocional.

Figura 120. Definición del estado emocional del sistema dinámico



Del análisis de la figura 120, puede definirse el estado emocional, α, y la intensidad de esta emoción, $||A_i||$.

$$\alpha = tg^{-1}\left(\frac{y_{pix} - y_{pi}}{t_w}\right) - tg^{-1}\left(\frac{y_{riw} - y_{pi}}{t_w}\right) \tag{6.17}$$

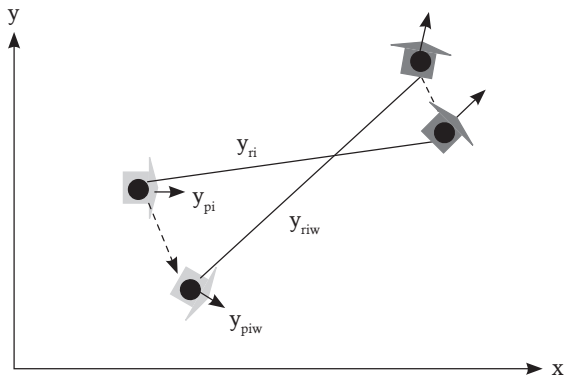
$$\left\|A_i\right\| = \sqrt{t_w^2 + \left(y_{riw} - y_{pi}\right)^2} \tag{6.18}$$

El estado emocional, α, y su intensidad, $||A_i||$, se combinan para generar la señal de error de entrada a un controlador, como se presenta en la ecuación 6.19, donde la letra *e* representa el error.

$$e_R = \alpha\left\|A_i\right\| \tag{6.19}$$

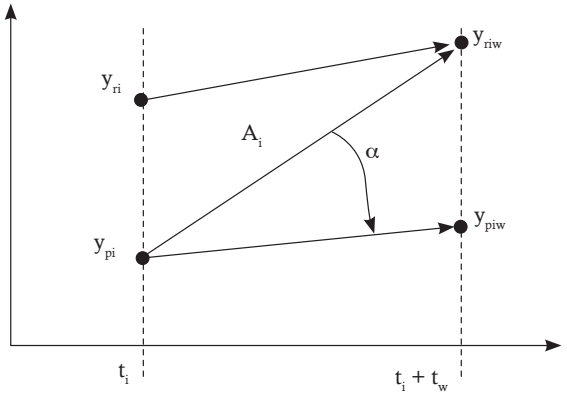
Ahora se mostrará la formulación de la ecuación 6.19, pero para el control del ángulo de dirección, θ, en lugar del radio, *R*. Se comenzará observando tanto al robot como al modelo de referencia en el instante actual y también la anticipación de lo que será la posición de la plataforma y de su modelo de referencia. En esta ocasión, en lugar de medir radios, se medirá el ángulo, como muestra la figura 121. Se distinguen entonces dos instantes de tiempo, más robot y modelo, lo que equivale a tener cuatro medidas de ángulo. El ángulo de referencia no es el ángulo que tiene la referencia, sino que es el ángulo de visión directa entre el robot y el modelo de referencia.

Figura 121. Medición de ángulos



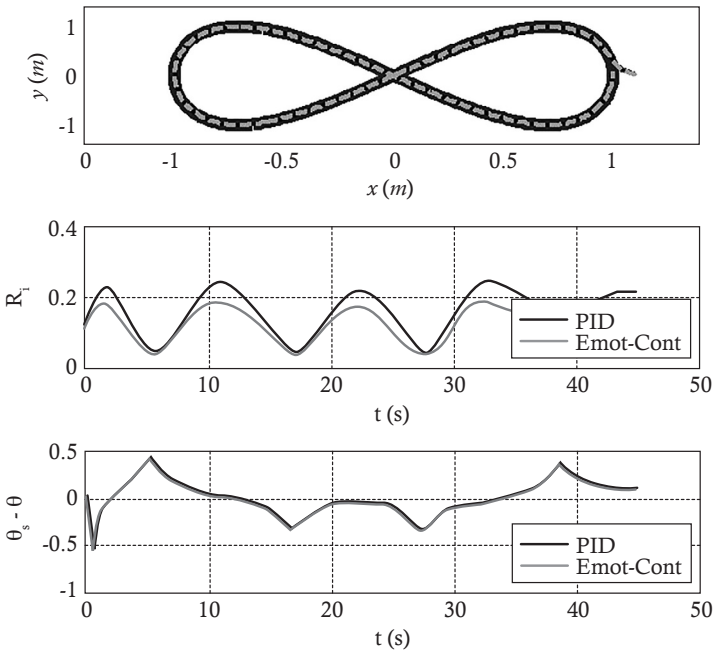
De acuerdo con las definiciones en la figura 121, se puede hacer el análisis en la figura 122. Nótese que la referencia no es cero, como en el caso del radio, sino que puede tener otro valor. Una vez más, se define el estado emocional como la diferencia en ángulo entre el vector que apunta al valor anticipado del modelo, y_{riw}, y el vector que apunta al valor anticipado de la planta, y_{piw}, donde el origen de estos dos vectores es el valor actual, y_{pi}. La intensidad emocional es la longitud del vector *Ai*. Las definiciones en las ecuaciones 6.17, 6.18 y 6.19 son iguales para el caso del control del ángulo. Lo único que se cambia es el subíndice para la señal de error, a e_θ.

Figura 122. Definición del estado emocional del sistema dinámico para el ángulo



Con el propósito de comparar el efecto de transformar la señal de error, de la definición tradicional, como en las ecuaciones 6.11 y 6.12 a la definición que utiliza la emulación de emociones humanas, enseguida se presenta la simulación de las dos definiciones, cuando estas señales alimentan controladores tradicionales PID. En la parte superior se ve el recorrido del modelo de referencia, en línea punteada, así como el recorrido de la plataforma robótica controlada por el PID tradicional, mientras que en color claro está el resultado usando la definición del error basada en emociones. Las coordenadas iniciales del modelo de referencia en la figura 123 son <1,0,pi/2>, mientras las condiciones iniciales del robot se han definido como <1.1,0,0>.

Figura 123. Comparación entre las dos definiciones de error



La parte intermedia de la figura 123 muestra el error en el radio. Puede verse cómo el radio con la definición basada en emociones resulta menor que con la definición tradicional de error. En la parte inferior se observa el ángulo de diferencia entre el ángulo ideal, θs, y el que realmente tiene el ángulo, con el ángulo del robot, θ. Si bien la diferencia es pequeña, aun así, el comportamiento con la definición de error basada en emociones es inferior, es decir, mejor.

Aprendizaje por refuerzo

El aprendizaje por refuerzo en este capítulo se va a llevar a cabo mediante el cambio de los pesos de una red neuronal tipo perceptron multicapa, con entrenamiento en línea. Esta está diseñada específicamente para el control de posición del robot, cuando este sigue lo que se ha llamado el modelo de referencia.

El algoritmo de entrenamiento para las redes con conexiones hacia delante es conocido como backpropagation, del inglés para propagación hacia atrás. El algoritmo tiene dos partes, en la primera, cada grupo o vector de entradas es presentado a la red y esta, usualmente inicializada con pesos aleatorios, genera algunas salidas. Estas salidas de la red son comparadas con las salidas que se espera que la red asocie a esas entradas, y se calcula la diferencia o error. Enseguida se presenta el segundo vector de entrada, también conocido como ejemplo; nuevamente se propaga la red, y se evalúa el error entre la salida de la red y las salidas que se esperarían. El error debido a todos los ejemplos es acumulado, y se repite esto con todos los ejemplos que incluya el conjunto de datos que se tenga. En la segunda parte del algoritmo de entrenamiento, con base en el error acumulado, se empiezan a cambiar los pesos de cada conexión en la capa de salida, de manera que el error disminuya (para mayores detalles con respecto al algoritmo de aprendizaje ir a la referencia [99]). Luego se cambian también los pesos de todas las otras capas. Cuando se han realizado estas dos partes, se dice que se ha implementado una época, luego de lo cual se repite la parte uno, luego la dos, y se repite el procedimiento por tantas épocas como sea necesario, hasta que el error esté en un rango aceptable.

El proceso ya descrito se conoce como entrenamiento en lote, o fuera de línea, porque se tienen todos los ejemplos y se dispone de tiempo para ajustar los pesos por medio del algoritmo de aprendizaje, y una vez se considera que la red tiene el conjunto de pesos adecuado, se detiene el entrenamiento y se utiliza la red. En este capítulo se trata un problema relacionado, pero más difícil, debido a que el aprendizaje debe hacerse en línea. La red neuronal debe aprender, debe hacerlo rápido, y además debe hacerlo a partir de un ejemplo por época.

En su forma más simple, lo que la red aprende es la dinámica inversa de la planta, como en [100]. Los ejemplos que se le presentan a la red, compuestos por duplas de datos correspondientes a la entrada y la salida de la planta, se intercambian, de tal manera que la entrada de la red es la salida de la planta y la salida de la red es la entrada de la planta.

En general, en la capa de entrada de la red pueden tenerse n conexiones, pero para simplificar la explicación en la figura 224 se suponen sólo cuatro: p_1 a p_4. Entre

la capa de entrada y la primera capa oculta existen conexiones entre todas las entradas y todas las neuronas, y cada conexión tiene asociado un peso, $iw^{1,1}$, denotado así por enlazar la capa de entrada con la primera capa oculta. De otra parte, cada peso tiene un subíndice, el cual indica la neurona destino y la entrada conectada por la conexión; para el ejemplo en la figura uno de los subíndices es 3,4. Cada neurona tiene un valor de ajuste, por ejemplo en la primera neurona es b^1_1, lo cual indica que es la primera neurona en la primera capa oculta.

La entrada de la función de activación de la capa oculta 1, es decir f^1, está en la ecuación 6.20.

$$n^1 1 = \sum_{R=1}^{4} iw^{1,1} 1, R * P_R + b^1 1 \tag{6.20}$$

La función de activación por lo general es sigmoidea, aunque en realidad puede ser cualquier función diferenciable.

$$a^1 1 = f^1 \left(\sum_{R=1}^{4} iw^{1,1} 1, R * P_R + b^1 1 \right) \tag{6.21}$$

La salida de la capa oculta es llamada a^1_1, como se indica en la figura 124. Este valor es una de las entradas para la capa de salida, en la cual se suman las contribuciones de las neuronas de la capa oculta, como se indica en la ecuación 6.22, y finalmente se calcula la salida, utilizando f^2.

$$n^2 1 = \sum_{S=1}^{3} lw^{2,1} 1, S * a^1 S + b^2 1 \tag{6.22}$$

$$a^2_1 = f^2(n^2_1) \tag{6.23}$$

En la ecuación 6.23 a^2_1 es la salida de la red, la cual se compara con la salida deseada, y_d, para determinar la desviación entre la red y la salida esperada.

$$e = \frac{\left(y_d - a^2 1 \right)^2}{2} \tag{6.24}$$

En la ecuación 6.24 e es el error cuadrático medio, según [117], con el cual se calcula la corrección que tendrán los pesos de toda la red. δ^2 para la capa de salida y δ^1 para la capa oculta, como se expresa en las ecuaciones 6.25 y 6.26.

$$\delta^2_1 = \left(y_d - a^2_1 \right) * f^{2'} \left(n^2_1 \right) \tag{6.25}$$

$$\delta^1_1 = f^{1'} \left(n^1_1 \right) * \delta^2_1 * lw^{2,1}_{1,1} \tag{6.26}$$

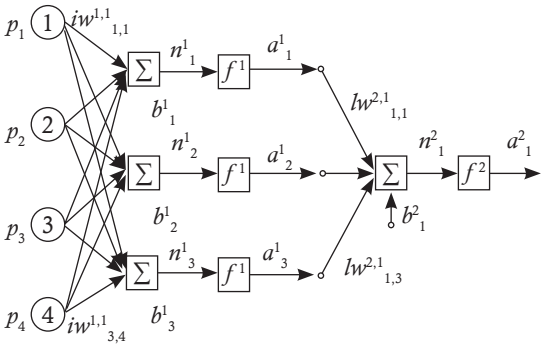
$f^{1'}$ y $f^{2'}$ son las derivadas de las funciones de activación. Finalmente, el valor de los pesos corregidos es el que se indica en las ecuaciones 6.27 y 6.28.

$$lw^{2,1}_{1,1}\left(t+1\right) = lw^{2,1}_{1,1}\left(t\right) + \alpha * \delta^{2}_{1} * a^{1}_{1} \tag{6.27}$$

$$iw^{1,1}_{1,1}\left(t+1\right) = iw^{1,1}_{1,1}\left(t\right) + \alpha * \delta^{1}_{1} * P_{1} \tag{6.28}$$

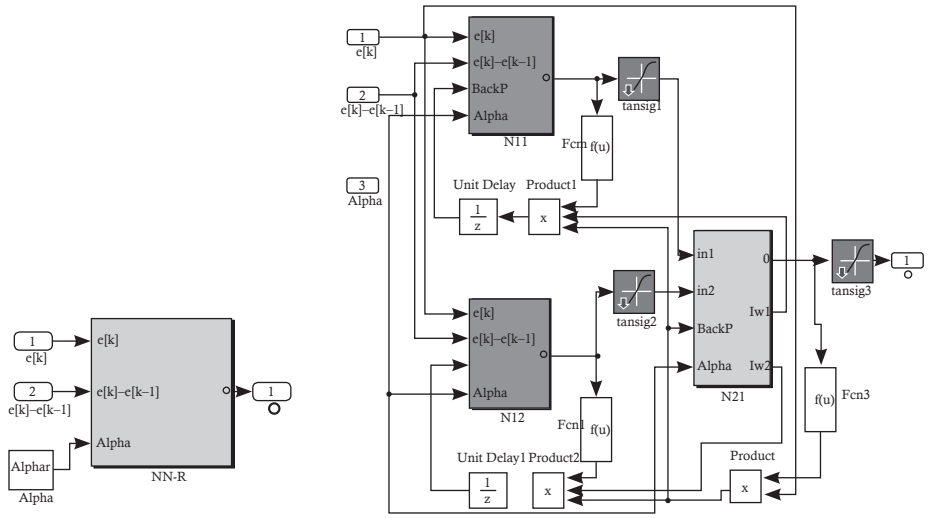
La taza de aprendizaje, α, es la encargada de definir la velocidad de aprendizaje. Para mayor detalle sobre las diferentes deducciones del algoritmo *backpropagation* en línea ir a [99], [101], [102]. Una configuración más completa se presenta en el anexo 1 en este capítulo.

Figura 124. Red neuronal perceptron multicapa



Ahora se explica la configuración de la red con aprendizaje por refuerzo para el control del radio, y luego para el control del ángulo. La red neuronal tiene dos entradas, la primera es el error actual, denotado como $e(k)$, y la segunda es la diferencia entre el error actual y el anterior. Esta segunda entrada tiene como propósito observar el signo del error, y de acuerdo con esto modificar los pesos de la red neuronal. La taza de aprendizaje fue fijada en 0.01. La figura 125 muestra además el interior de la red neuronal, la cual cuenta con dos neuronas en la capa oculta y una neurona en la capa de salida. Las funciones de activación en todos los casos probados son tangentes sigmoidales. La función que define el delta de peso cada instante de tiempo depende de la misma salida de cada neurona, de la salida de la capa anterior y el error, que a su vez es la primera entrada, según indican las ecuaciones 6.24 a 6.28.

Figura 125. Configuración de la red neuronal



La función *f(u)* en la figura 125 tiene la forma de una campana, y corresponde a la derivada de la función de activación, según indica la ecuación 6.24 y 6.25. Ahora se observa el interior de una red, en la cual se hace el cálculo del incremento o decremento del peso de cada conexión, lo que emula el aprendizaje por refuerzo. En la figura 126 pueden verse las conexiones y los bloques con los cuales se actualiza el peso, por refuerzo, de cada conexión. En el lado derecho de la figura 126 se observa al interior del bloque *w11*, es decir, en el cual se actualiza el peso *w11*. El bloque *w11_0r* es el valor inicial de la red, el cual puede definirse como un número aleatorio, entre -1 y 1. De ahí en adelante, el valor del peso es igual al valor del peso actual, más el delta de peso, según indican las ecuaciones 6.27 y 6.28.

Ahora se muestra el resultado de aplicar el controlador por refuerzo diseñado en esta sección, y aplicado tanto al control del radio, como al control del ángulo. Los valores con que se inicializaron las redes son:

w11_0r = 0.6363; *w12_0r* = 0.6351; *w21_0r* = 0.4449; *w22_0r* = -0.7030; *lw1_0r* = 0.3192;

lw2_0r = 0.0372; *b1r* = 0.9459; *b2r* = 0.2980; *blr* = 0.6007; *Alphar* = 0.001;

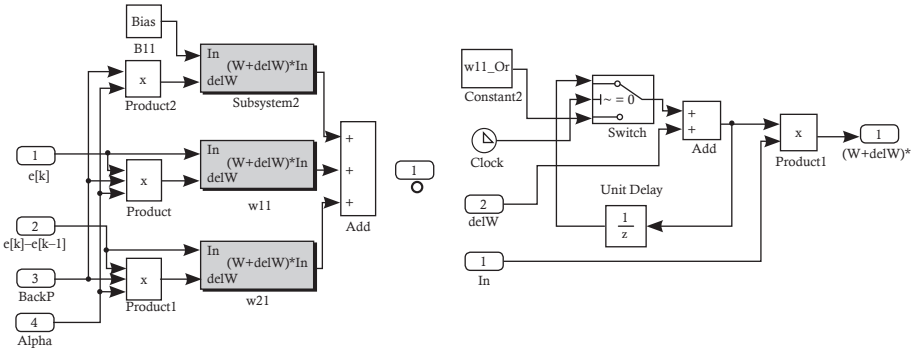
Para la red que controla el ángulo, y:

w11_0 = 0.4254; *w12_0* = 0.0009; *w21_0* = -0.0578; *w22_0* = -0.8808; *lw1_0* = 0.3639;

lw2_0 = -0.9151; *b1* = -0.8571; *b2* = 0.0433; *bl* = -0.8065; *AlphaTh* = 0.001;

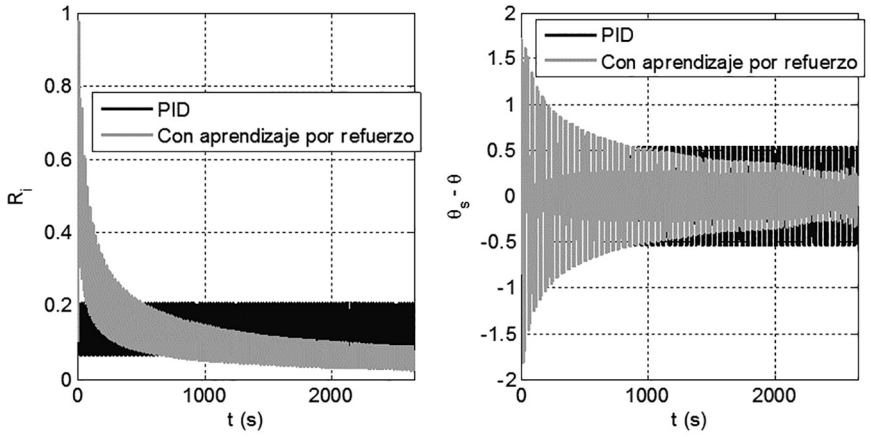
para la red que controla el radio.

Figura 126. Conexión de la neurona *N11*.



Estos valores iniciales hacen que el sistema presente el comportamiento en color claro en la figura 127. En un comienzo el error es alto, casi de un metro, y a medida que el tiempo pasa, la red aprende, y el radio se reduce, hasta incluso ser menor que el error de distancia que produce un PID tradicional. El comportamiento con el error de ángulo, también en la figura 127, es similar al comportamiento del radio, es decir, reduce con el tiempo. Si bien la red aprende, necesita tiempo para hacerlo, dado que en lugar de tener muchos datos en cada época, se tiene un solo ejemplo por época. En la simulación se presentan 60 vueltas del modelo de referencia, con una duración aproximada de 44 s por vuelta.

Figura 127. Control con aprendizaje por refuerzo.



Controlador basado en emociones con aprendizaje por refuerzo

Esta sección presenta los resultados de implementar la definición del error utilizando la emulación de emociones humanas, y de otra parte, se anexa la componente de

aprendizaje por refuerzo, explicada en la sección anterior. En la figura 128 puede verse el esquema final, en el cual se diferencian tres partes del sistema de control. La primera es la definición de la señal de error, utilizando emociones emuladas. El bloque que tiene como salida el error para el ángulo requiere muchas entradas, y para simplificar el diagrama no se presentan. Recuérdense los valores requeridos en la figura 122, y las ecuaciones 6.17 a 6.19. El segundo componente se refiere a la red neural, en la cual los pesos cambian, por medio del refuerzo, representando aprendizaje. El tercer componente es la plataforma diferencial en sí. De esta plataforma se enfatizan dos partes: la parte dinámica del modelo, con el modelo de los motores que mueven las ruedas; y de otra parte, el modelo cinemático del robot.

El resultado del control cuando las ganancias del PID tradicional son <1.6,0,0> para el control del radio, y <0.6,0,0> para el control del ángulo. Esta prueba implica, al igual que en la sección anterior, el seguimiento del modelo de referencia por 60 vueltas, cuando cada vuelta tarda 44 s. El resultado del controlador PID es igual, sin importar el paso del tiempo, dado que el sistema puede considerarse invariante en el tiempo, y porque las ganancias del controlador también son constantes. Esto es cierto tanto para el radio como para el ángulo. En contraste, el controlador basado en emociones, con aprendizaje por refuerzo, comienza con error bastante alto, comparado con los PID, pero el paso del tiempo cambia la situación.

Figura 128. Controlador emocional con aprendizaje por refuerzo

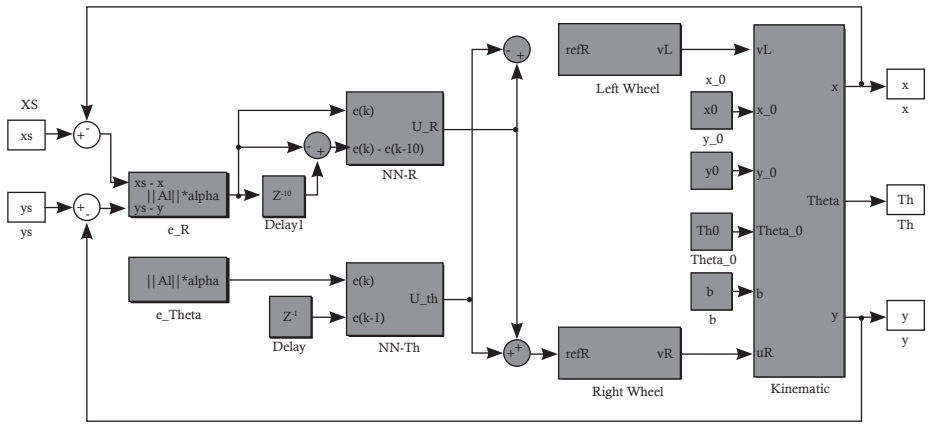
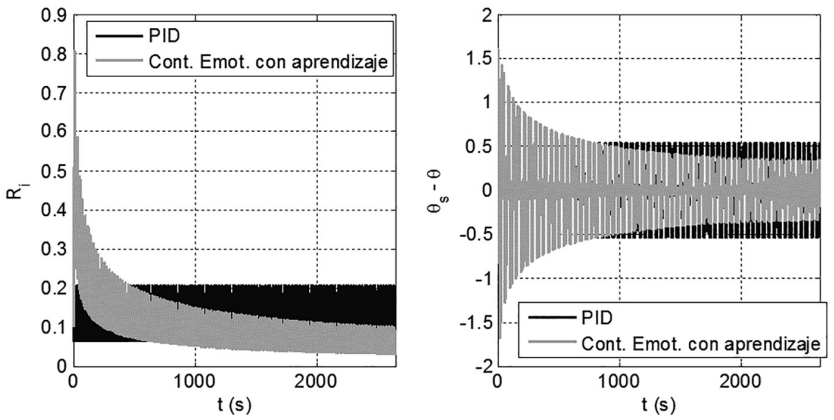
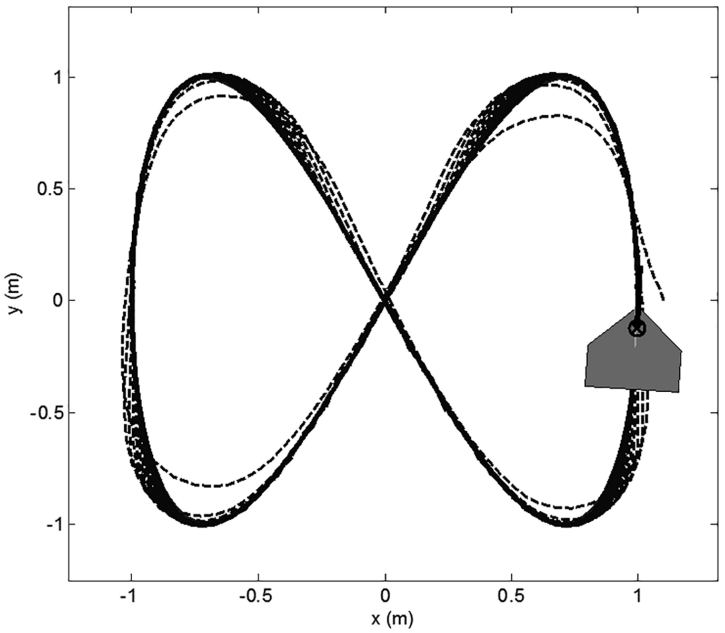


Figura 129. Resultado del control basado en emociones con aprendizaje por refuerzo



La figura 130 permite ver de una forma diferente las curvas en la figura 129 el aprendizaje del controlador basado en emociones. En un comienzo, el recorrido del robot se distancia del modelo de referencia, pero a medida que se completa el giro varias veces, el error reduce, y luego de unas vueltas, el robot pasa sobre el recorrido del modelo de referencia.

Figura 130. Recorrido de la plataforma con el controlador final



Conclusiones y trabajo futuro

El controlador basado en emociones, con ganancias fijas, permite que el robot pueda seguir el perfil, o ruta, de referencia, con mejoras que superan el 10% con respecto a la definición tradicional del error, y la aplicación de un controlador tradicional PID, con las mismas ganancias del controlador basado en emociones.

El aprendizaje por refuerzo permite que la plataforma robótica se aproxima al modelo de referencia, sin importar si en un comienzo el error de posición y ángulo es grande. Se observó la influencia de la tasa de aprendizaje, y se concluye que no necesariamente aprender más rápido significa un mejor resultado.

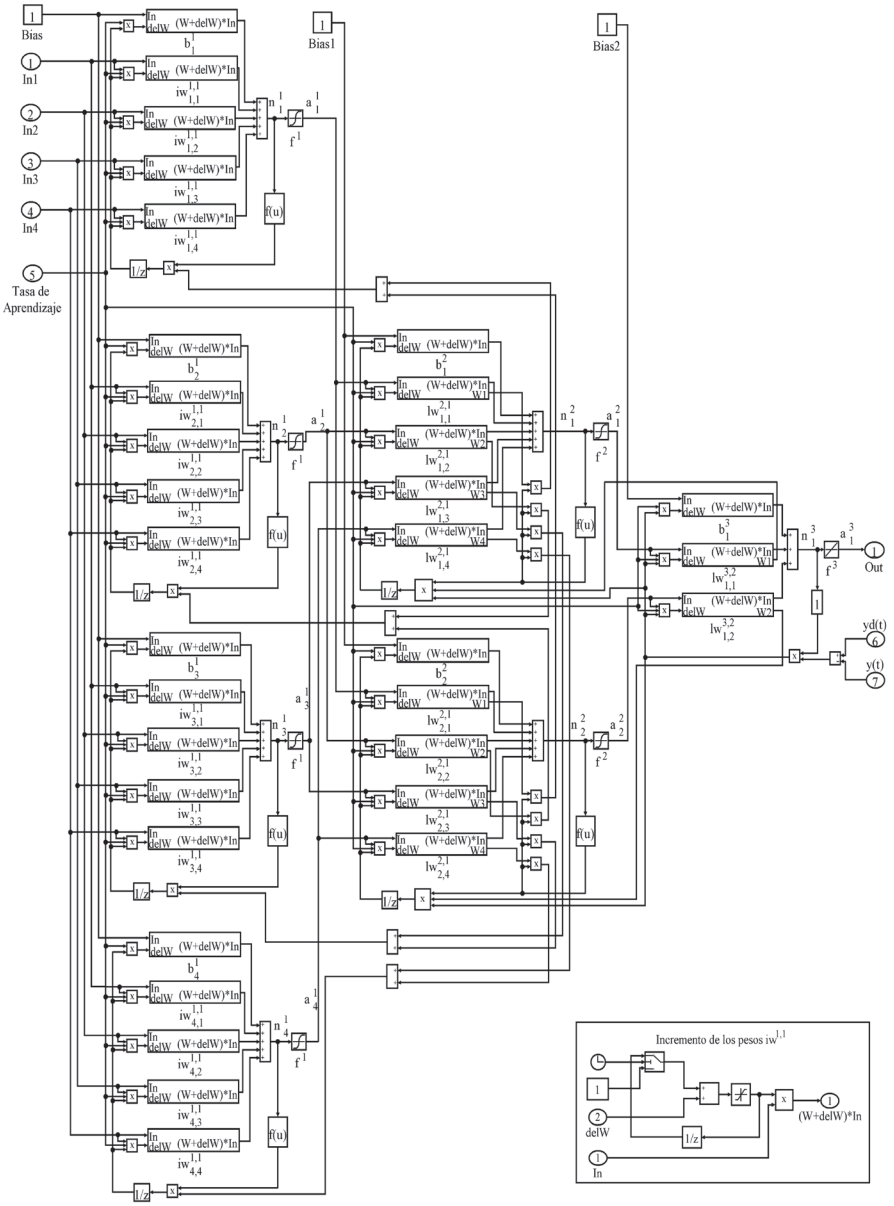
Los valores iniciales de los pesos de la red neuronal influyen sobre el comportamiento de la red, por tanto, en términos generales, no es posible aplicar pesos iniciales aleatorios, sino que por ensayo error deben seleccionarse. Si bien esto no representa darle la respuesta a la red neuronal, sí implica que la red no puede estar muy lejos de la respuesta para encontrarla.

Se pudo verificar que es apropiado seleccionar la figura de Lissajous como modelo de referencia, dado que exige dinámicas lentas y rápidas, además de distintos radios de curvatura, positivos y negativos. Esto permite evaluar el rendimiento de cualquier controlador, con movimientos en espacio reducido.

Es importante explorar en trabajos futuros el incluir una memoria, de manera que la mejor red neuronal pueda ser almacenada, y que esta puede ser llamada nuevamente cuando las condiciones dinámicas se repitan, sin necesidad de repetir el aprendizaje por refuerzo.

Estudios posteriores deben explorar el proceso de cambio de modelo de referencia cuando se presentan obstáculos durante el recorrido ideal, lo cual hará que el robot pueda evitarlos. De igual manera, es importante que se continúe evaluando otras estrategias de control, resultando así una base de la cual puede compararse y escoger el control más adecuado en cada caso.

Anexo 1. Diagrama de conexiones de una red neuronal con aprendizaje en línea, 4,4,2,1

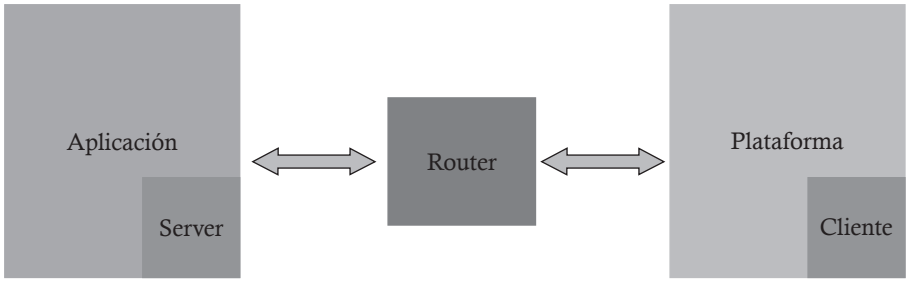


Protocolo de comunicación del sistema Multirobot

El continuo avance tecnológico, con respecto a las tecnologías de la información, en la actualidad, hace necesario la implementación de herramientas que permitan el acceso a la información de manera rápida, versátil y eficaz; para ello, las plataformas y las comunicaciones inalámbricas proporcionan los instrumentos para el manejo, la manipulación y el control de la información. Esto permite proyectar una solución confiable y centralizada basada en datagramas TCP (*sockets*) soportada sobre una infraestructura tipo cliente-servidor.

Buscando un proceso de comunicación robusto y fiable para la transferencia de datos entre todas las plataformas robóticas que permita de forma permanente estar intercambiando información, se hace necesario establecer un protocolo de comunicación integrado al uso de comunicación IP soportada sobre tecnología inalámbrica WIFI. En la figura 131 se muestra un esquema básico de la infraestructura cliente servidor que fue implementada.

Figura 131. Infraestructura tipo cliente-servidor



El bloque Aplicación es el bloque principal del sistema, ya que es el encargado de manejar la interfaz gráfica del usuario GUI por medio de una arquitectura cliente-servidor únicamente accediendo desde un computador. Este bloque tiene las siguientes características:

- Debe ser diseñado en Java.
- Tiene una interfaz gráfica con acceso a:

– Conexión y desconexión al módulo ethernet de la interfaz de *hardware.*
– Intercambio de información mediante *sockets* entre cualquier integrante de la red.
– Debe permitir una comunicación cliente-servidor por medio de *sockets* que transmitan y reciban comandos de la interfaz de *hardware* implementada.

El bloque Router es el encargado de establecer la comunicación entre todas las interfaces de *hardware* implementadas en el sistema, permite la asignación de una dirección IP a la interfaz de *hardware* que estará dentro de la máscara de red, al cual esté conectado el *router.*

La interfaz de *hardware* que es implementada para cada plataforma robótica cumple la función de generar y adecuar las condiciones para el manejo de la información recibida desde el bloque Aplicación, instalado en el equipo servidor, para el control del *hardware* encargado del cambio de estado de las funciones operativas que se puedan tener de salida. De igual forma, el bloque Plataforma debe estar notificando e indicando el cambio de estado de sus funciones y enviar respuesta o confirmación de las órdenes generadas por el bloque Aplicación. Las principales características de este bloque son:

- Comunicación-implementación sobre protocolo TCP–IP sobre puertos de conexión ethernet.
- Posee puertos de entrada (análogos y digitales) controlados por medio de un microcontrolador o microprocesador, según fuese requerido.

Implementación de la comunicación por *sockets*

Para la implementación de la comunicación se necesita desarrollar un algoritmo que permita el manejo de los protocolos, y la decisión implementada por el equipo de investigación fue desarrollar *sockets* bidireccionales donde el bloque Aplicación manipule y envíe caracteres de cambio al bloque Plataforma, que, a su vez, deberá enviar confirmación de recepción correcta de información.

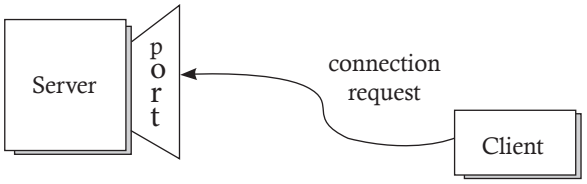
Teniendo en cuenta el concepto de *socket* como mecanismos de interfaz entre programas a través de una red TCP/IP, pueden ser implementados dentro de la misma máquina o por medio de la misma red. Generalmente, son usados en forma cliente-servidor y son desarrollados en Java, dado que proporcionan clases ServerSocket y Socket que ayudan a los procesos de implementación. Al realizar la implementación de la comunicación por *sockets*, se tuvo en cuenta el modelo cliente-servidor para mantener siempre activa la conexión entre el boque Aplicación (servidor) y el bloque Plataforma (cliente).

Para establecer la comunicación por medio de *sockets* –entendidos como puntos finales de enlaces de comunicaciones que intercambian datos entre procesos–, se debe programar el servidor y, por otro lado, el cliente. El tipo de *sockets* describe la forma en la que se transfiere información por medio de ellos. Normalmente, un servidor se ejecuta sobre un computador específico y tiene un *socket* que responde en un puerto

específico. El servidor únicamente espera, escuchando a través del *socket* a que un cliente haga una petición.

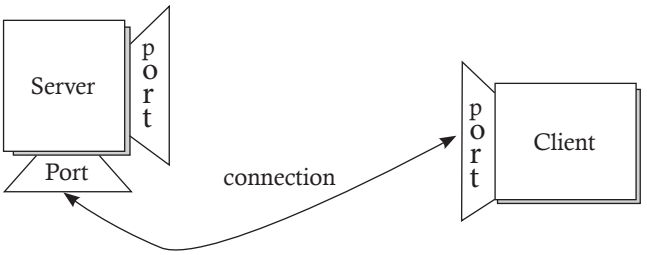
En el lado del cliente, se conoce el nombre del *host* de la máquina, en la cual el servidor se encuentra ejecutándose y el número de puerto en el cual el servidor está conectado. Para realizar una petición de conexión, el cliente intenta encontrar el servidor realizando una petición a la máquina en el puerto especificado (figura 132).

Figura 132. Petición de conexión arquitectura cliente-servidor



La petición de conexión será aceptada por el servidor, quien obtiene un nuevo *socket* sobre un puerto diferente. Este proceso se debe a que el servidor requiere seguir atendiendo al *socket* original para peticiones de conexión mientras atiende las necesidades del cliente conectado anteriormente (figura 133).

Figura 133. Comunicación cliente-servidor



Por parte del cliente, si la conexión es aceptada, un *socket* crea formas satisfactorias y puede usarlo para comunicarse con el servidor. Es importante darse cuenta de que el *socket* en el cliente no está utilizando el número de puerto usado para realizar la petición al servidor. En lugar de este, el cliente asigna un número de puerto local a la máquina en la cual está siendo ejecutado. Ahora el cliente y el servidor pueden comunicarse escribiendo o leyendo en o desde sus respectivos *sockets*.

Java sockets

El paquete java.net de la plataforma Java proporciona una clase *socket,* la cual implementa una de las partes de la comunicación bidireccional entre un programa Java y otro programa en la red. La clase *socket* se sitúa en la parte más alta de una implementación de pendiente de la plataforma, ocultando los detalles de cualquier sistema particular al programa Java. Al usar la clase java.net.Socket en lugar de utilizar un

código nativo de la plataforma, los programas Java pueden comunicarse por medio de la red de una forma totalmente independiente de la plataforma. De forma adicional, java.net incluye la clase ServerSocket, la cual implementa un *socket,* que los servidores pueden utilizar para escuchar y aceptar peticiones de conexión de clientes.

Modelo de comunicaciones con Java

El modelo de *sockets* más simple que puede ser implementado en Java tiene las siguientes características (figura 134):

- El servidor establece un puerto y espera durante cierto tiempo (*time out* segundos) a que el cliente establezca la conexión. Cuando el cliente solicite una conexión, el servidor abrirá la conexión *socket* con el método *accept()*.
- El cliente establece una conexión con la máquina *host* por medio del puerto que se designe *puerto#.*
- El cliente y el servidor se comunican con manejadores InputStream y OutputStream.

Librerías para manejo de tareas de red

En el bloque Aplicación, inicialmente, se debe agregar las librerías y los objetos que se utilizarán para establecer la comunicación. En la figura 135 se muestran las librerías tipo io, y net que se utilizaron. Asimismo, en la misma figura se puede observar los métodos de transferencia de datos incluidos en la librería *socket*, como InputStream y OutputStream, que nos dan respectivamente un InputStream y un OutputStream, por medio de los cuales se puede transferir y manejar los datos.

Figura 134. Modelo de comunicación por *sockets* en Java

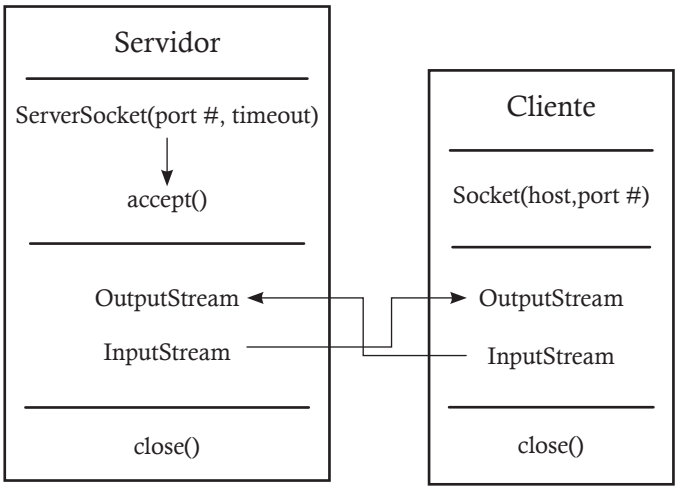


Figura 135. Librerías implementadas en código fuente

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.SocketAddress;
import java.net.UnknownHostException;
```

Luego de definir los objetos, se realiza la configuración de una comunicación simple por medio de *sockets* que manejan siempre un concepto, como se detalla en la figura 136. Inicialmente, se crea un objeto de tipo *socket,* sobre el cual se establecerá la conexión, además se incluyen los datos del servidor como la dirección IP y el puerto que maneja la comunicación, luego se crean y se definen los objetos de entrada y salida de datos, al igual que sus respectivos *buffers*; por último, el programa automáticamente cierra el *socket* y finaliza el proceso de transmisión de paquetes mediante *sockets.* Cualquier comunicación mediante *sockets* maneja esta estructura; se pueden agregar otros atributos, como el tiempo de espera de la conexión y los condicionales de conexión según el algoritmo diseñado para dichas implementaciones.

Figura 136. Sintaxis de implementación para comunicación TCP/IP

```
try {
    //create a new socket instance
    SocketAddress sockaddr = new InetSocketAddress("192.168.0.20",8888);
    nsocket = new Socket();
    nsocket.connect(sockaddr, 5000);//connect and set a 10 second connec
    if (nsocket.isConnected()) {//when connected
        nis = nsocket.getInputStream();//get input
        nos = nsocket.getOutputStream();//and output stream from the socl
        inFromServer = new BufferedReader(new InputStreamReader(nis));//
    }
//catch exceptions
} catch (IOException e) {
    e.printStackTrace();
    result = true;
} catch (Exception e) {
    e.printStackTrace();
    result = true;
} finally {
    closeSocket();
}
return result;
```

El primer paso para establecer la comunicación entre la aplicación y la interfaz de *hardware* es implementar un evento que ejecute la conexión, es decir, que desde la aplicación solicite al servidor la conexión. Para ello, se implementó el código mostrado en la figura 137, en el que se tuvo en cuenta una característica especial sobre el manejo de tareas en segundo plano, aspecto importante para el correcto funcionamiento del bloque Aplicación.

Figura 137. Evento de conexión

```
private OnClickListener buttonConnectOnClickListener = new OnClickListener() {
    public void onClick(View v){
        if(!connected){//if not connected
            outputText("Conectando al Servidor");
            networktask = new NetworkTask(); //New instance of NetworkTask
            networktask.execute();
        }else{
            outputText("Desconectado del Servidor");
            if(networktask!=null){
                networktask.closeSocket();
                networktask.cancel(true);
            }
        }
    }
};
```

Se puede ver que el evento está asociado a un elemento básico de una GUI, este permitirá la conexión con el servidor; teniendo en cuenta un estado booleano de *true* o *false*, el evento estará escuchando constantemente algún cambio en el estado del botón; además, nótese que en este fragmento de código se llama una función *networktask.* Además, se puede observar que se asocia a otras funciones directas de *networktask*.

Para establecer la comunicación entre los bloques Aplicación y Plataforma, se debe tener en cuenta que la información o los comandos de manejo deben estar disponibles en tiempo real, por tanto, para esta implementación se manejará el concepto de comunicación TCP/UDP Cliente-Servidor. En la tabla 21 se muestran las configuraciones principales de las funciones más comunes utilizadas en este tipo de comunicaciones.

Tabla 21. Configuración de las principales funciones que se utilizan para realizar una comunicación cliente-servidor

Etapa	Función	Descripción
Inicialización	begin()	Inicializa la librería y la configuración de red ethernet; la librería es compatible con DHCP, por tanto, se puede obtener la dirección MAC y asignarle a la placa una dirección IP.
	localIP()	Obtiene la dirección IP de Shield Ethernet, es útil cuando la dirección es asignada por medio de DHCP.

Etapa	Función	Descripción
Direccionamiento	IPAddress()	Define una dirección IP. Se puede utilizar para declarar direcciones locales y remotas.
Comunicación	Server	Crear un servidor que escucha las conexiones entrantes en el puerto especificado.
	EthernetServer()	Crear un servidor que escucha las conexiones entrantes en el puerto especificado.
	begin()	Indica al servidor para empezar a escuchar las conexiones entrantes.
	write()	Escribir datos en todos los clientes conectados a un servidor.
	print()	Imprimir los datos a todos los clientes conectados a un servidor. Imprime los números como una secuencia de dígitos, cada carácter ASCII (por ejemplo, el número 123 se envía como los 3 personajes '1 ', '2', '3 ').
	println()	Imprimir los datos, seguidos por un salto de línea, a todos los clientes conectados a un servidor. Imprime los números como una secuencia de dígitos, cada carácter ASCII (por ejemplo, el número 123 se envía como los 3 personajes '1 ', '2', '3 ').

Para la comunicación con el servidor, se debe establecer un protocolo sobre la capa de transporte, que, como se sabe, tiene como finalidad comunicar las aplicaciones entre sí (comunicación de extremo a extremo), apoyándose para ello en los servicios proporcionados por la capa de red (protocolo IP). Son 2 los protocolos empleados generalmente a este nivel: TCP y UDP.

Para que 2 aplicaciones se puedan comunicar, es necesario disponer de algún mecanismo de identificación que le permita a una aplicación en una máquina origen identificar a otra aplicación en otra máquina remota con la que desea comunicarse. Esto se realiza mediante los llamados "puntos de acceso al servicio", que en TCP/UDP se denominan puertos. Para la configuración del socket, se deben definir sus propios parámetros: puerto, dirección IP de destino, tamaño del *buffer*, entre otros.

Figura 138. Parámetros de configuración de red de la interfaz ethernet

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddress serverIP(192,168,0,20);
int serverPort=8888;
```

De la figura 138 se puede observar que se introduce una dirección IP y un número de puerto para el *socket*, de tenerse en cuenta que la dirección IP depende de la dirección

de red del *router*; además, en esta ocasión, se eligió el puerto 8888. Luego se da inicio a la conexión mediante la siguiente sintaxis de código, en la cual se ingresan los datos de la dirección MAC y la IP.

Teniendo ya la conexión, se define el modo de configuración y, para ello, se debe crear una variable, que para este caso va a hacer el servidor, que se encargará de escuchar las peticiones del usuario y la asignación del puerto. Por último, para que la configuración de la comunicación quede completa, se crean ciertos condicionales de conexión, es decir, cuando se establezca el servidor y se crea la conexión con algún cliente, se ejecuta el algoritmo principal.

Después de configurar el módulo ethernet del bloque Plataforma, y establecer la comunicación con el bloque Aplicación, se realiza la transmisión de datos por medio de varias funciones que permiten el envío y la lectura de datos. En la figura 139 se muestran las funciones utilizadas para la lectura y la escritura de datos. Por ejemplo, nótese que la opción de lectura se maneja de forma similar a la de una comunicación serial, pero como son varios los datos recibidos, es necesario crear una variable que acumule los datos que sean trasmitidos.

Figura 139. Transmisión de datos (sintaxis)

```
String commandStr ="";//Connamdstring whe
char c = client.read();
commandStr+=c;//and ands')) {//if a client
server.println("setPoti:"+String(state));
```

Una de las ventajas directas de utilizar tecnología IP para el proceso de comunicación es la de no tener límites definidos para la cantidad de bytes transmitidos; de igual forma, la velocidad establecida puede estar por el orden de los 11 megabytes por segundo basados en el soporte de la tecnología wifi. Esta robustez en el proceso de comunicación nos lleva a diseñar una trama de comunicaciones convencional que pueda ser usada y leída desde y hacia cualquier plataforma robótica utilizada en el proyecto. Por esta razón, para la implementación del sistema Multirobot fue proyectado el uso de la trama de comunicaciones mostrada en la figura 140.

Figura 140. Trama general de comunicaciones

Inicio de trama	ID Plataforma	Coor X	Coor Y	Sentido	Flag	Info	Fin de trama

Esta trama está conformada por 8 segmentos y cada segmento se separa por un caracter especial que determina los límites de cada uno de ellos. El caracter implementado fue '%' y conduce a su no utilización dentro de la información transmitida desde y hacia las plataformas robóticas. En la tabla 22 se describe el formato de trama usado.

Tabla 22. Formato de trama

Inicio de trama	**Determina el inicio del formato de trama; está compuesta por el carácter '@'.**
ID plataforma	Determina el parámetro de identificación de cada una de las plataformas robóticas en el área de trabajo EjR1.
Coor X/Coor Y	Estos 2 bloques entregan las coordenadas tanto en X como en Y de la ubicación de la plataforma robótica.
Sentido	Entrega el sentido o ángulo de dirección de desplazamiento de la plataforma.
Flag	Este segmento de la trama posee 2 valores 1 o 0, esta bandera identificará si se habilita el segmento de información en el cual se puede transmitir cualquier tipo de información codificada en ASCII.
Info	Segmento de trama en el cual se puede colocar cualquier tipo de información relevante que necesita ser transmitida desde el servidor hacia el cliente o viceversa.
Fin Trama	Determina el inicio del formato de trama; está compuesta por el carácter '#'.

Ejemplo general de trama de comunicación:

@%R1%23,50%150,67%75%1%sensor1_135,3cm%#

Sistema Multirobot cooperativo

Siguiendo el enfoque tomado por Brooks en el desarrollo de la arquitectura subsumida, este capítulo muestra el diseño de sistemas de control comportamental a nivel de un robot y a nivel del sistema Multirobot. En ellos se evidencia el diseño y la implementación de capas comportamentales constituidas por comportamientos básicos que son coordinados por medio de los mecanismos de inhibición y supresión. En cada uno de los comportamientos desarrollados se muestran las características más relevantes de cada uno y su función dentro de la tarea general de búsqueda y localización de una fuente de calor, así como los resultados y el análisis de cada uno de los procesos de experimentación llevados a cabo.

En la actualidad, las aplicaciones en robótica móvil están orientadas al diseño de sistemas cooperativos que de gran manera potencializan muchas de las habilidades que pudiera presentar por sí solo un solo robot. Como se había mencionado, las arquitecturas clásicas de control imposibilitan de gran manera implementaciones en el campo de la robótica cooperativa, ya que por su estructura requieren de modelos abstractos de su entorno [103], lo que los hace lentos y altamente costosos a nivel computacional.

El paradigma reactivo introdujo la idea de diseñar robots capaces de interactuar con el humano y que fuesen tan sencillos y tan asequibles como se lograra. Bajo este paradigma, surgieron enfoques distintos que dieron origen a arquitecturas reactivas. Una de ellas es la arquitectura subsumida derivada del trabajo de Brooks, que por medio de mecanismos como la supresión y la inhibición logra la coordinación del sistema haciendo de esta una jerarquía de capas comportamentales que permiten diseños incrementales basados en estructuras tan simples y primitivas como el diseñador las pueda imaginar [103,104,105,106].

Muchas de las tareas que hoy en día puede desarrollar un robot dentro de un equipo de búsqueda y rescate urbano *USAR* son bastante peligrosas y, por lo mismo, requieren de una alta capacidad por parte del prototipo de responder a los estímulos tan rápido como se presente la situación; se llevó a cabo el diseño y la implementación de un sistema cooperativo enfocado en la búsqueda y localización de fuentes de calor siguiendo el enfoque de Brooks.

Técnica de rastrillaje

La descripción de tareas y técnicas utilizadas por los equipos *USAR* en zonas colapsadas (leer capítulo 1) es el punto de partida de este trabajo de investigación. Como se muestra en el primer capítulo, existen labores de búsqueda y localización definidas para espacios abiertos y espacios cerrados, y en las cuales se implementan una serie de técnicas caracterizadas para cada una de ellos. La técnica de rastrillaje aplicada a los 2 tipos de situaciones es la que se decidió implementar en este proyecto. Algunas de las tareas específicas que se realizan en la técnica de rastrillaje son:

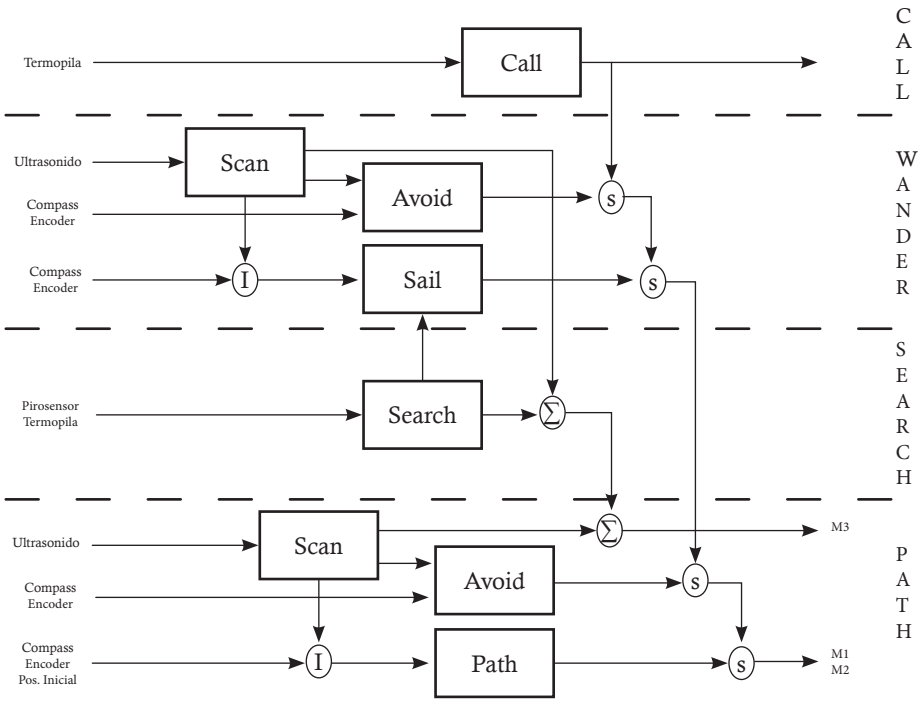
- Ubicación del equipo de búsqueda en un punto inicial.
- Enlistamiento de cada uno de los miembros del equipo.
- El líder del equipo ordena a cada uno de los miembros del equipo avanzar en línea recta a una determinada distancia.
- Una vez recorrida dicha distancia, cada uno de los miembros realiza un barrido desde su punto.
- Si alguno de los rescatistas ubica o localiza una posible víctima, le notifica al líder, informándole la situación.
- De presentarse la situación anterior, el liderazgo de la situación es entregado al rescatista que encontró la víctima.
- Luego se dirige libremente hacia la víctima, teniendo cuidado con cada uno de los obstáculos y escombros que hallan en el camino.
- Mientras tanto, los demás rescatistas mantienen su posición a la espera de nuevas órdenes. Una vez localizada la víctima, se informa la ubicación a los demás rescatistas.
- Posteriormente, cada uno de ellos se dirige hacia la baliza. De no encontrarse ninguna víctima, se vuelve a realizar el avance previa orden del líder.

Comportamientos reactivos básicos implementados a nivel de un robot

La descripción anterior de tareas se ejecuta por parte de un equipo humano de búsqueda y rescate, lo que aún para ellos resulta ser una labor bastante dispendiosa y peligrosa. El desarrollo de comportamientos compatibles con este tipo de situaciones se puede lograr con la integración de tareas simples desde el punto de vista robótico y por medio de la integración de cada una de ellas en capas comportamentales con mecanismos de coordinación (supresión e inhibición) [103].

Esta descripción se tradujo en la estructura comportamental mostrada en la figura 141 y que está a nivel de un solo robot. El objetivo de la descripción de los comportamientos básicos que serán implementados a nivel del robot es poder establecer un punto de partida para el diseño de las máquinas de estado que serán diseñadas para el desarrollo de todo el proceso de un robot como objeto individual.

Figura 141. Descripción de comportamientos a nivel de un solo robot



Capa Path

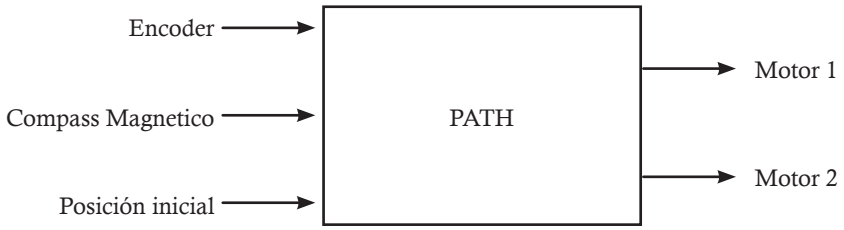
Es la capa que garantiza que el robot siga una trayectoria preestablecida dentro del proceso de búsqueda de víctimas, evitando los obstáculos que se le presenten durante su recorrido. Esta capa está conformada por 3 comportamientos básicos: Path, Scan y Avoid.

Comportamiento Path

En el proceso de búsqueda de la víctima, se han definido 4 movimientos típicos de los rescatistas en zonas colapsadas interiores cumpliendo la normativa internacional [104]. Con base en lo definido, se implementó en el robot una trayectoria que debe ser preestablecida, y que permita el cumplimiento de una formación básica. La información requerida por este comportamiento es suministrada por los *encoders* que alimentan las ecuaciones cinemáticas del robot, que calcula la velocidad lineal y angular del recorrido de la plataforma.

Un compás magnético establece el sistema de referencia del robot necesario para determinar su orientación y la posición inicial que es requerida para la estimación de posiciones, y es punto de partida para la formación de todo el sistema multiagente. La salida del comportamiento debe ser entendida como señales de estímulo aplicado a cada motor (M1, M2) implementado en la plataforma (figura 142).

Figura 142. Descripción I/O para el comportamiento Path



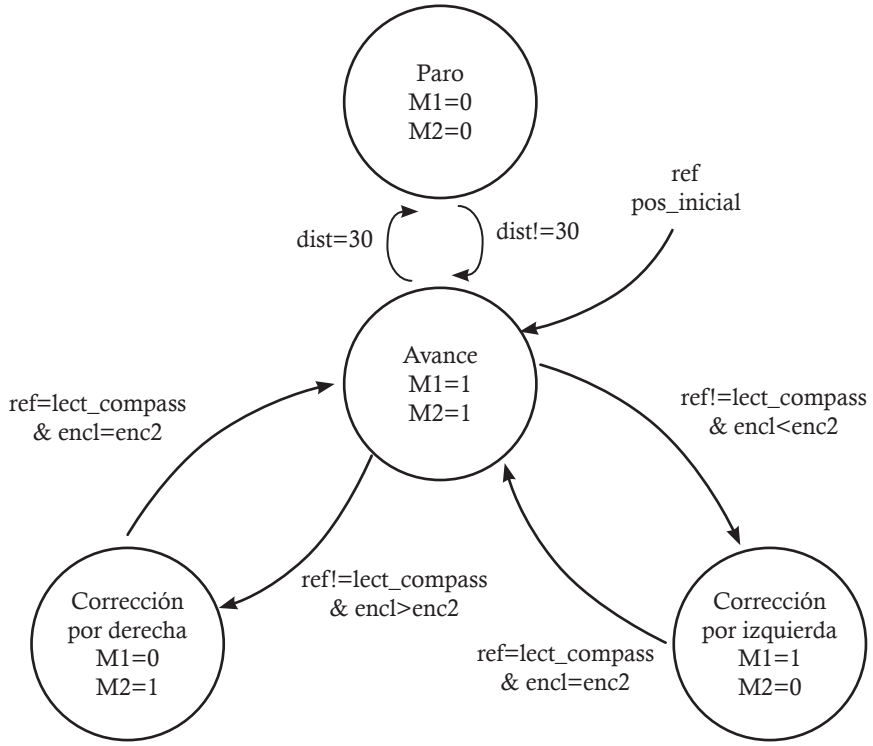
Para la validación del comportamiento, se implementó el algoritmo que se muestra en la figura 143, representado por un diagrama de autómatas finitos. En este, se muestra la dinámica que se sigue por parte del comportamiento Path durante su ejecución, y se distinguen 4 estados: paro, avance, corrección por derecha y corrección por izquierda, y cada uno de ellos es activado y desactivado por una serie de señales que a continuación se describen de manera detallada:

- *dist:* esta variable permite el monitoreo del valor de distancia recorrida por el robot en una trayectoria en línea recta. Cuando el valor de dist se hace 30, indica que el robot cumplió 30 cm de su recorrido y activa el estado paro, de lo contrario mantiene al comportamiento bajo la influencia de estado avance.
- *ref:* esta variable indica el valor en el cual el robot configura su rumbo. Está dado por el valor sensado por el compás magnético y permite el control de la orientación tomada por el robot durante la ejecución del comportamiento Path.
- *pos inicial:* hace alusión al punto de origen con coordenadas (0, 0) para el robot.
- *enc 1:* esta variable registra en todo momento el valor del *encoder* del motor de la rueda derecha. Por medio de la comparación con la variable enc_2, permite conocer el movimiento que está presentando el robot en todo instante de tiempo. También es una variable clave en el proceso de cálculo cinemático para la plataforma robot.
- *enc 2:* esta variable registra en todo momento el valor del *encoder* del motor de la rueda izquierda. Por medio de la comparación con la variable enc_1, permite conocer el movimiento que está presentando el robot en todo instante de tiempo. También es una variable clave en el proceso de cálculo cinemático para la plataforma robot.
- *M1*, *M2:* corresponde al motor derecho (M1) y el motor izquierdo (M2).
- *lect_compass:* corresponde al valor registrado en todo momento por el compás magnético.

Durante el proceso de validación del comportamiento, se registraron los valores de distancia recorridos por el robot. Durante la experimentación, que consistió en la ejecución de 2 pruebas en las que se registraron los valores de distancia recorrida por el robot en una trayectoria recta de 2 m, se obtuvieron los errores porcentuales para

intervalos de 10 cm y la variación en el cambio de distancia en esos mismos intervalos. En la figura 144 se muestra el análisis de las muestras tomadas, en ellas se observa que los valores medidos corresponden altamente a las medidas seleccionadas, y presentan, en el mayor de los casos, un error porcentual de 9.05% y variaciones entre las muestras tomadas de 1.41%.

Figura 143. Máquina de estados para la descripción del comportamiento Path



Comportamiento Scan

La búsqueda y detección de un obstáculo dentro de un recorrido mientras se realiza otra actividad es el objetivo principal de este comportamiento, que es activado después de realizar un recorrido preestablecido por el robot (para el caso fue de 30 cm), mediante una señal enviada por el comportamiento Path. El comportamiento Scan, mediante señal enviada por el sensor de ultrasonido (este sensor está montado sobre un motor denominado M3 para implementar un radar sencillo), determina la distancia y ubicación de un obstáculo respecto a la posición del robot, que permite activar el comportamiento Avoid e inhibir el comportamiento Path, que permite al robot sortear el obstáculo existente y buscar la trayectoria original para continuar su recorrido con los demás miembros del equipo (figura 145).

Figura 144. Resultados del proceso de validación para el comportamiento Path

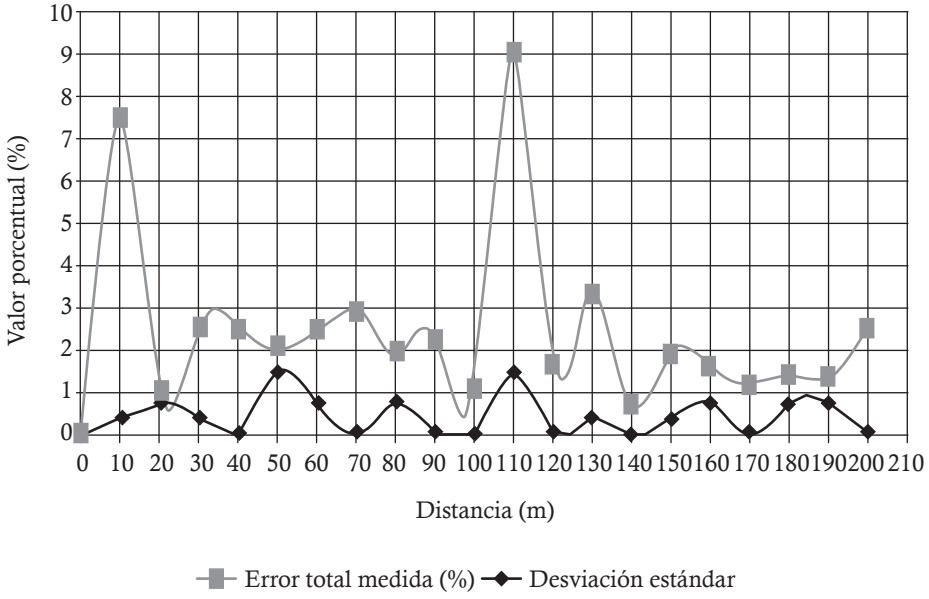
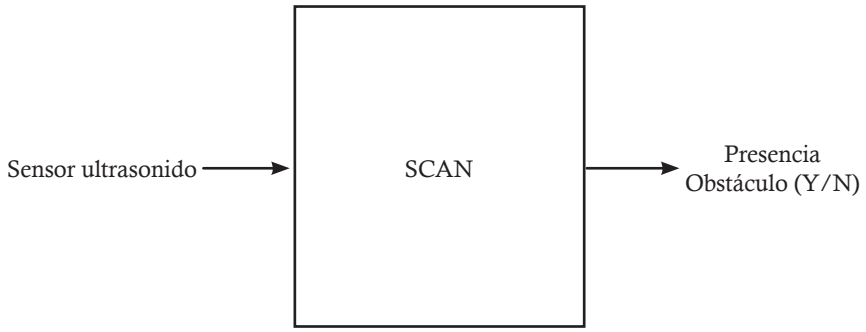
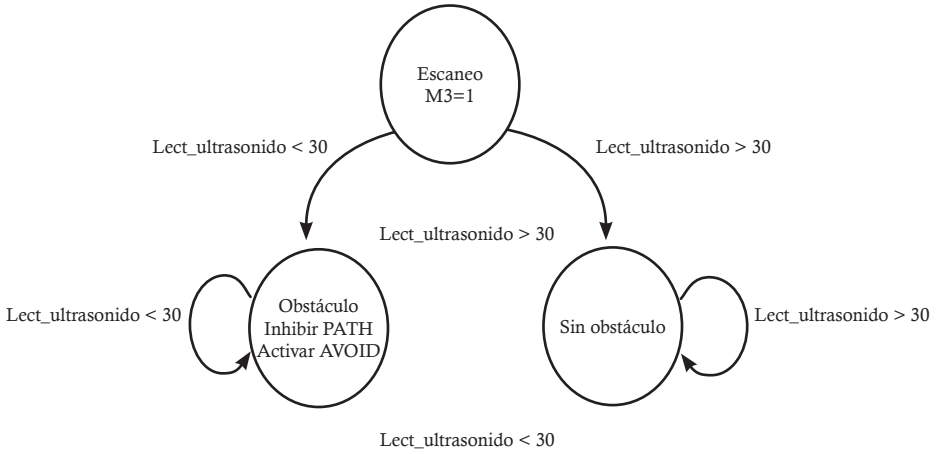


Figura 145. Descripción I/O para el comportamiento SCAN



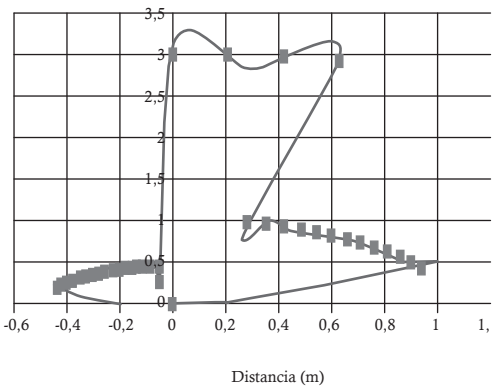
En la figura 146 se muestra el diagrama de máquinas de estado finitas que sirvió para modelar el algoritmo de validación del comportamiento SCAN. En este hay 3 estados básicos: escaneo, obstáculo y sin obstáculo; en cada uno de ellos se presentan procesos de activación y desactivación gobernados por la variable lect_ultrasonido, la cual registra de manera continua el valor consignado por el sensor ultrasonido y permite ejercer el control sobre el comportamiento.

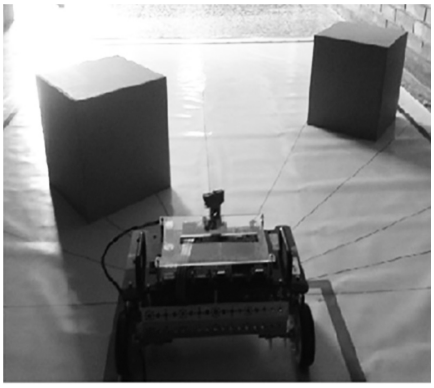
Figura 146. Diagrama de flujo para la implementación del comportamiento SCAN



Durante el proceso de experimentación en LabVIEW, para el comportamiento SCAN, el robot tomó muestras con la ayuda del motor M3, en un rango de apertura de ±90° y comparó los valores entregados por el sensor ultrasonido para así determinar la dirección y distancia del obstáculo. De encontrar presencia de algún obstáculo, el comportamiento genera una señal de tipo booleano que permite la inhibición del comportamiento Path y activa el comportamiento Avoid (figura 147). Los datos registrados por el robot permitieron conocer el desempeño del comportamiento frente a distintas situaciones. En la figura 147 se muestra un resultado de detección en donde alrededor de los 50 cm se ubican una serie de puntos que representan la superficie de un obstáculo detectado por izquierda y a un metro por derecha.

Figura 147. Experimentación y validación del comportamiento SCAN

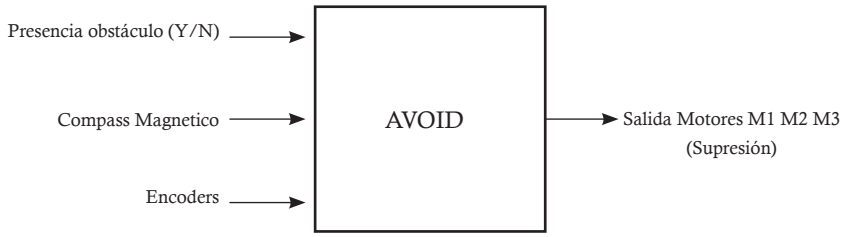




Comportamiento Avoid

Con la activación dada por el comportamiento SCAN, el comportamiento Avoid tiene como objetivo principal evadir el obstáculo, pero sin olvidar que debe regresar a la ruta original. Para hacer esto, este comportamiento suprime el comportamiento Path y debe calcular el error de su movimiento con respecto a la ruta preestablecida para garantizar el retorno a la ruta original. Para este fin, utilizó el algoritmo VFH+. Las entradas de este comportamiento son las señales de los *encoders* y del compás necesarias para calcular las posiciones del robot y sus salidas son estímulos a los motores M1 y M2. Una vez se regrese a la ruta original, este comportamiento debe liberar el comportamiento Path que será el que sigue ejecutándose (figura 148).

Figura 148. Descripción I/O para el comportamiento Avoid



Para la implementación de este comportamiento, se usó el algoritmo VFH+ [107], el cual está basado en el método de campos de fuerza virtuales, lo cual es una combinación del concepto de los campos de potencial con otro concepto que son las celdas de certidumbre. El método de celdas de certidumbre para la representación de obstáculos permite agregar y retirar datos durante la ejecución, y admite una fácil integración de múltiples sensores. Para crear una cuadrícula de certidumbre, el área de trabajo del robot se divide en varias celdas, y conforma una cuadrícula general conocida, donde cada celda (i, j) contiene un valor de certidumbre $C(i, j)$, que indica el grado de confianza de la localización de un obstáculo dentro del área de la celda. Mientras más grande sea el valor de certidumbre $C(i, j)$, mayor será el nivel de confiabilidad.

Basado en el concepto de celdas de certidumbre [108], en el cual se buscan huecos en histogramas polares construidos localmente, los valores del histograma son inversamente proporcionales a las distancias de los obstáculos alrededor del robot. Esto se refleja en el entorno, donde una ruta libre de obstáculos en el histograma es reflejada como un valle que le permite al algoritmo ubicar el valle más amplio con el objetivo de determinar la nueva dirección del recorrido que debe seguir el robot. Con base en el concepto propio del algoritmo, este fue diseñado para buscar aberturas que pudiesen representar un espacio libre por el cual el robot pueda moverse libremente mediante la asignación de valores o pesos a cada una de las posibles rutas o caminos libres, donde la que menor valor de peso tenga será la que el robot debe elegir para seguir el recorrido mediante la implementación de cuatro fases [109, 110].

La primera de ellas realiza la construcción de un primer histograma H^p por medio de (8.1).

$$H_k^P = \sum_{i,j \in C^*}^{n} M_{ij} \cdot h_{ij} \tag{8.1}$$

En donde: k es el valor numérico que representa el valor; i, j son las coordenadas de la celda activa en la ventana activa de acción C^*; m_{ij} es el valor que se afecta directamente del movimiento del robot y está dado por (8.2).

$$m_{ij} = C_{ij}^* \left(a - b d_{ij} \right) \tag{8.2}$$

Donde C_{ij}^* es el valor de la celda activa en la posición i, j e influye directamente en $d_{i,j}$, $d_{i,j}$ es la distancia que hay entre la celda activa y el robot; a, b son constantes positivas para configurar un valor de 0 en celdas inactivas.

En (8.1) el valor de $h_{i,j}$ especifica si la celda C^* pertenece o no al sector activo. Por medio de este mecanismo es posible considerar el tamaño del robot (figura 149) y se define que cada una de las celdas construidas al ser ampliadas resultan ser un círculo con radio r_{r+s} (8.3).

$$r_{r+s} = r_r + d_s \tag{8.3}$$

Donde r_r es el radio del robot; d_s representa una distancia segura desde un obstáculo hacia el robot.

Con el fin de conocer el histograma real en el que se ve involucrado el robot, se calcula el valor de $y_{i,j}$, y está dado por (8.4):

$$y_{i,j} = sen^{-1}\left(\frac{r_{r+s}}{d} \right) \tag{8.4}$$

El valor de $h_{i,j}$ se calcula por medio de (8.5).

$$h_{ij} = 1 i_{fk} * \alpha(\beta_{ij} - y_{ij}, \beta_{ij} + y_{ij}) \tag{8.5}$$

En donde $\beta_{i,j}$ es el ángulo de dirección desde la celda C^* a la posición real del centro del robot $x_0 y_0$, y es calculado por medio de (8.6).

$$\beta_{i,j} = \tan^{-1}\left(\frac{y_j - y_i}{x_j - x_i} \right) \tag{8.6}$$

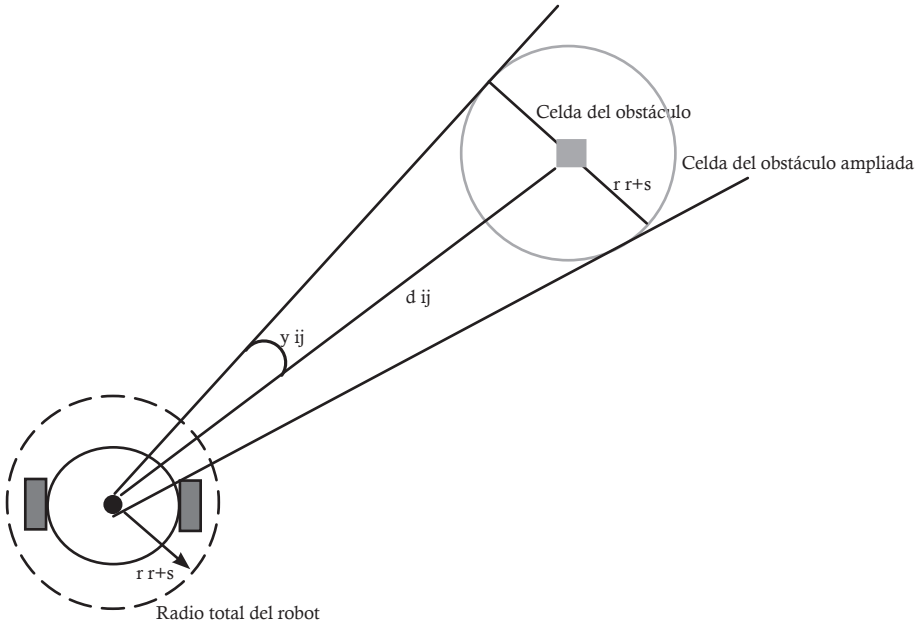
Para determinar las zonas que están ocupadas o libres se ejecuta la segunda fase del algoritmo. En este se crea un segundo histograma H^b con la ayuda del primero y utiliza 2 valores de umbral, que son τ_{high}, τ_{low}, y está dada por las ecuaciones (8.7), (8.8) y (8.9).

$$H_{k,j}^b = 1 if H_{k,j}^p > \tau_{high} \tag{8.7}$$

$$H^{b}_{k,j} = 0 if H^{p}_{k,j} > \tau_{low} \tag{8.8}$$

$$H^{b}_{k,j} = 1 if H^{p}_{k,j} > \tau_{high} \tag{8.9}$$

Figura 149. Relación entre la celda de obstáculo y el robot

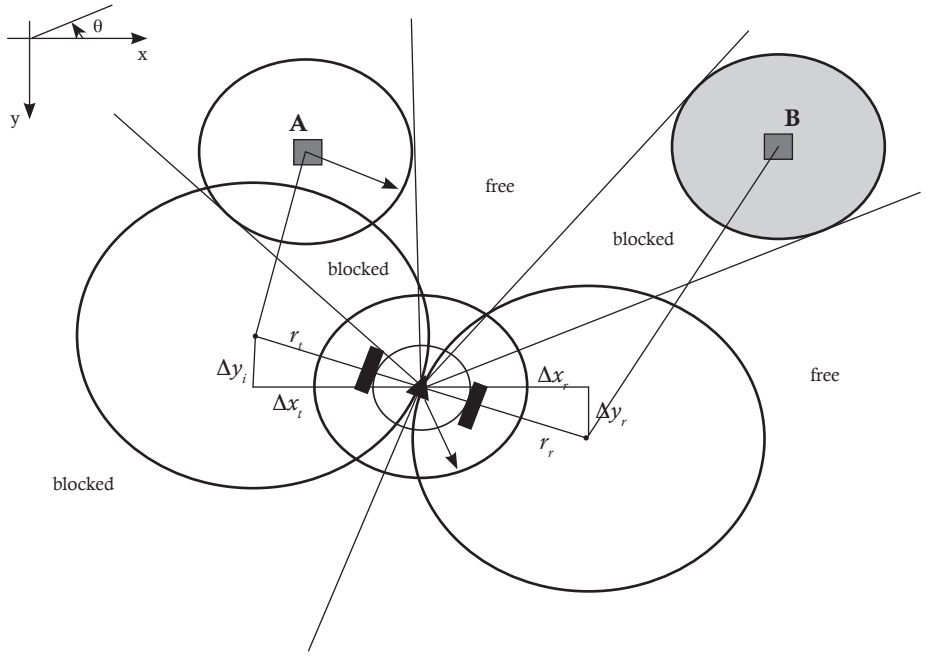


Un tercer histograma H^{m} es calculado para determinar la dirección tomada por el robot para superar el obstáculo, y se constituye como la tercera y cuarta fase respectivamente. Se basa en las restricciones cinemáticas del robot. Este se denomina *histograma polar enmascarado*. Al combinarlo con el histograma H^{b}, se puede conocer el radio de acción del robot en el cual supera los obstáculos sin llegar a colisionar (figura 150). En este los radios de los círculos *rl.rr* dependen directamente de la velocidad del robot. Si se presenta la situación en la cual el círculo de la zona izquierda intersecta el radio de una celda de obstáculo ampliada, todas las acciones que pueda ejecutar el robot en esta dirección se bloquean y se hace un retroceso del robot. De igual manera, sucede con el círculo de acción por la derecha.

El proceso de validación para el comportamiento se realizó con la implementación del algoritmo VFH+ en LabVIEW. En las figuras 151 y 152 se evidencian una serie de datos y gráficas, los cuales representan la evolución del comportamiento en ejecución a través del tiempo. Inicialmente, para entender un poco la terminología de estas gráficas, se traducen algunos términos propios:

- Datos sin procesar de LIDAR (Raw LIDAR): está representado por pequeños cuadros de color negro; se traduce como la proyección de puntos que hace el algoritmo para representar el obstáculo detectado por el robot.
- Umbral interior (*inner threshold*): está representado por una línea delgada de color gris oscuro, la cual se interpreta como el radio interno efectivo del robot; se considera como el radio crítico de acción del robot y al cual se presentaría un inminente choque con los obstáculos en el entorno.
- Umbral exterior (*outer threshold*): está representado por una línea delgada de color gris claro, la cual se interpreta como el radio externo efectivo del robot. Sobre este valor la navegación del robot se considera segura.

Figura 150. Naturaleza del proceso de selección de dirección en el algoritmo VFH+



Fuente: [109].

- Histograma (*histogram*): está representado por pequeños cuadros de color gris medio; esta información es como tal el histograma o la reconstrucción de zonas libres y zonas ocupadas que realiza el algoritmo. En esta se consideran tanto las proyecciones de los obstáculos como la de los posibles caminos que puede tomar el robot para evadir el obstáculo [110].
- Objetivo (*target*): está representado por una línea gruesa de color gris claro sobre una circunferencia media que denota el frente de robot; cuando esta circunferencia

media se torna de color gris oscuro, se refiere a la presencia de uno o varios obstáculos sobre la trayectoria del robot.

- Comando de dirección (*command heading*): está representado por una línea gruesa de color gris oscuro sobre una circunferencia media; denota la dirección en la cual el algoritmo detecta un obstáculo y a la cual no se dirige el robot. Esta información es útil a la hora de considerar hacia qué dirección no se debe tomar un futuro camino o ruta, ya que es altamente probable una colisión.
- Rumbo actual (*current heading*): está representado por una línea delgada de color gris muy claro, la cual denota la dirección tomada por el robot sobre el camino. Esta información es útil para conocer la trayectoria escogida por el robot.
- Rumbo despejado (*open headings*): está representado por pequeños cuadros de color gris claro y se interpretan como aquellas zonas libres a la cuales el robot puede dirigirse sin ningún riesgo de una posible colisión contra un obstáculo.
- Rumbo bloqueado (*blocked headings*): está representado por pequeños cuadros de color gris oscuro y se interpretan como aquellas zonas inaccesibles a la cuales el robot no puede dirigirse debido a la inminente posibilidad de una colisión con un obstáculo.

En la figura 151 se presenta la primera secuencia de 4 imágenes que recrean la información entregada por el comportamiento al usuario. En el primer cuadro (parte superior izquierda), el robot se encuentra ejecutando una rutina de avance en línea recta sobre un entorno sin obstáculos. En el cuadro ubicado en la parte superior derecha, se evidencia la información del histograma detectando una posible presencia de obstáculo. En el cuadro ubicado en la parte inferior izquierda se muestra cómo el robot está cada vez más cerca de un obstáculo, proyectando sobre la imagen una serie de pequeños cuadros negros. Finalmente, en el último cuadro, el algoritmo confirma la presencia de un obstáculo muy cerca del umbral externo del robot.

La figura 152 es la continuación de la secuencia de imágenes y en ella se sigue la evolución del comportamiento Avoid sobre el robot; en esta, el primer cuadro (parte superior izquierda) muestra el instante en el que la información del histograma refleja la presencia de un obstáculo sobre el umbral mínimo de detección, a lo cual el comportamiento responde con el inicio de la rutina de evasión como tal. En los siguientes cuadros, se observa cómo la información del histograma refleja la cercanía con la que el robot evade el obstáculo, y cómo por medio de esta información reconstruye una silueta aproximada de obstáculos que está superando.

Figura 151. Resultados del proceso de validación del comportamiento Avoid

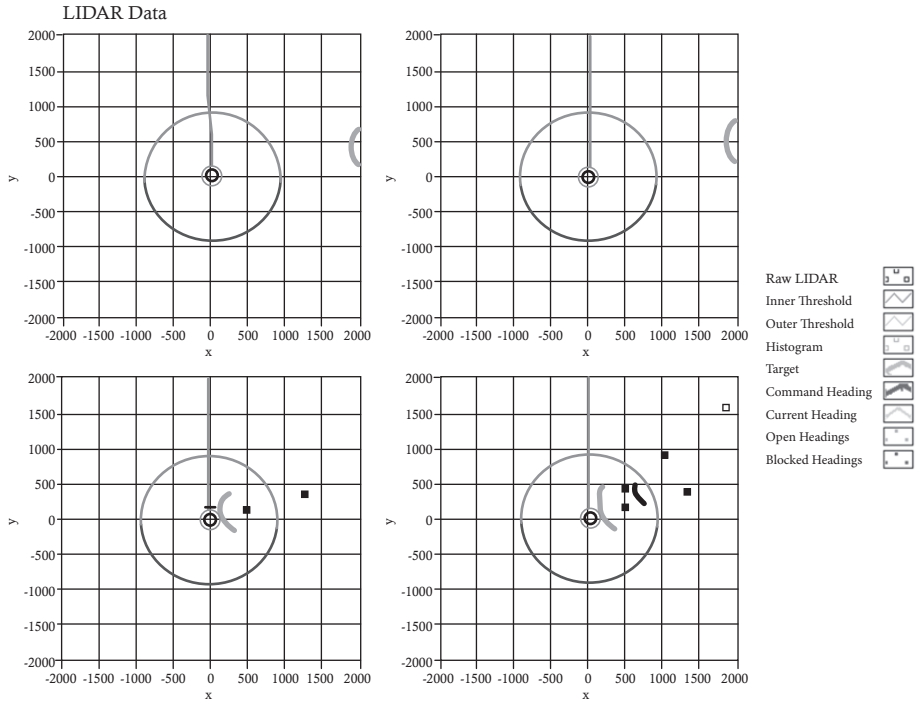
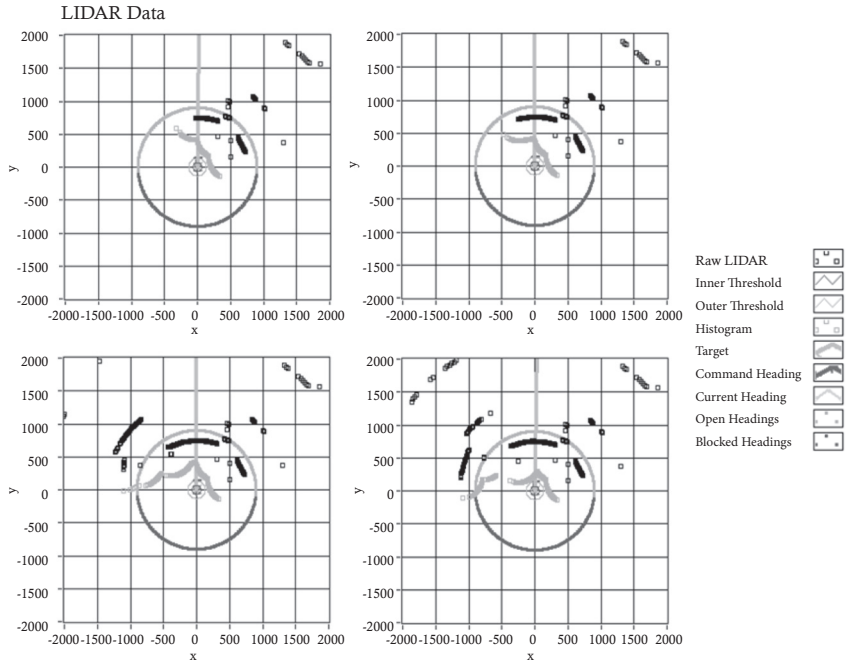


Figura 152. Resultados del proceso de validación del comportamiento Avoid



Capa Search

Es la capa que garantiza que el robot efectúe la búsqueda de la víctima dentro de la zona de prueba establecida. Esta capa está conformada por un comportamiento básico que es Search.

El comportamiento básico que conforma esta capa tiene como objetivo principal la búsqueda y detección de la víctima que para la aplicación es representada como una fuente de calor. Este comportamiento toma la señal del sensor TPA81 para determinar la dirección hacia dónde se encuentra la víctima y la distancia a ella (figura 153). A su vez, permite el control sobre el motor M3 (motor dedicado a la implementación de un radar). Este comportamiento genera una señal de control para activar el comportamiento Sail.

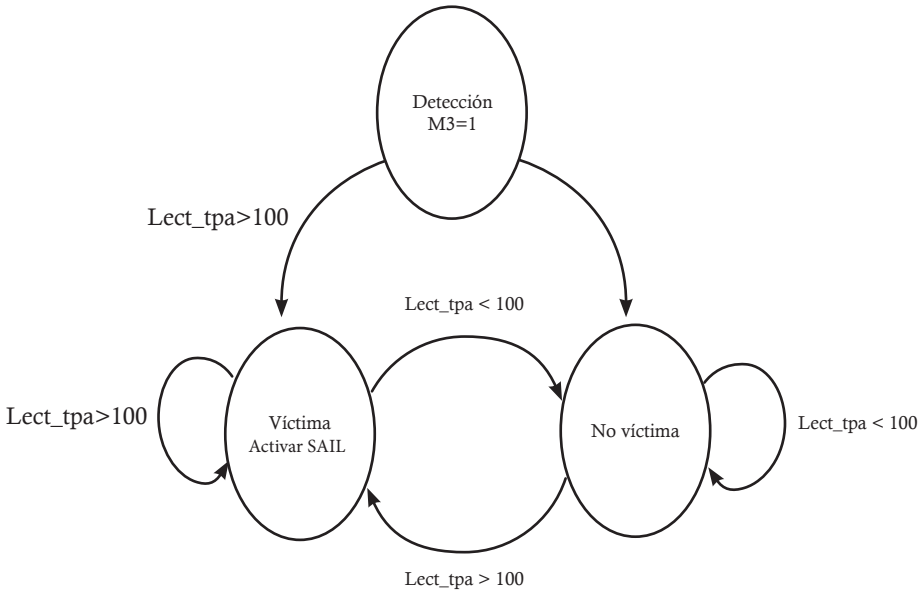
Figura 153. Descripción I/O para el comportamiento Search

En la figura 154 se muestra el diagrama de autómatas finitos para el comportamiento Search; en este, se evidencia la presencia de 3 estados: detección, víctima y no víctima, respectivamente. Cada uno de ellos está influenciado por la variable Lect_tpa, la cual registra continuamente los valores tomados por el sensor termopila TPA81 y dependiendo de ellos, toma acciones sobre cada uno de los estados del comportamiento.

Durante el proceso de validación para el comportamiento implementado en LabVIEW, se realizó un barrido de la zona y, por medio de la lectura del sensor termopila TPA81, de la cual se obtuvo una gráfica la que se refleja tanto la dirección como la intensidad de la fuente de calor; en la figura 155 se muestra un pequeño mapa termal muy sencillo que retrata los aspectos mencionados anteriormente por medio de pequeños cuadros sobre el eje vertical, cada uno sobre 9 regiones que representan las zonas activas de detección del robot. Los colores de cada uno de estos cuadros determinan el valor de temperatura en cada una de las regiones, a partir de las siguientes pautas:

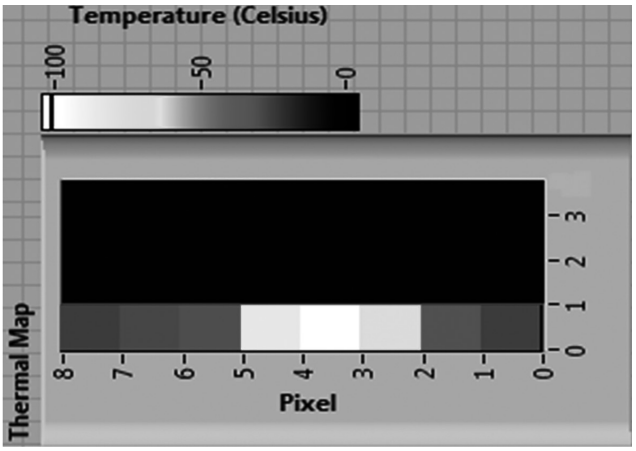
- Tonos en escala de negros y oscuros: representan temperaturas en el rango de los 0 °C y los 35 °C.
- Tonos en escala de grises: representan temperaturas en el rango de los 36 °C y los 75 °C.
- Tonos en escala de blancos: representan temperaturas superiores a los 75 °C.

Figura 154. Diagrama de autómatas finitos para el comportamiento Search



Siguiendo las pautas descritas anteriormente, en el mapa termal de la figura 155 se observa la presencia de una fuente de calor en las zonas 3, 4 y 5 respectivamente, y con el valor de intensidad registrado con la tonalidad de colores se concluye que la dirección de la fuente de calor se ubica sobre la región 4 del sensor.

Figura 155. Resultado del proceso de validación del comportamiento Search



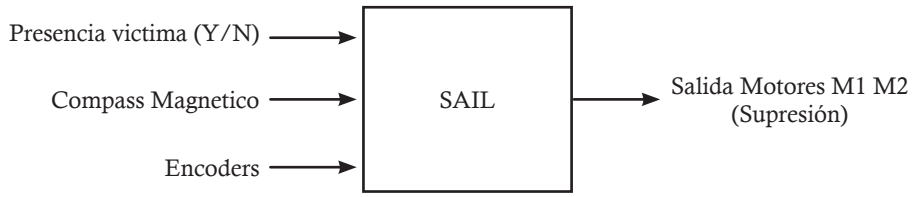
Capa Wander

Esta capa garantizó el recorrido del robot hacia la víctima sin una ruta preestablecida y donde el robot estuvo en la capacidad de superar la presencia de posibles obstáculos presentes en el camino. Está conformada por 3 comportamientos básicos: Sail, Scan y Avoid.

Comportamiento Sail

Este comportamiento es activado por el comportamiento Search ante la presencia de la víctima. Permite que el robot rompa la formación inicial, entendido esto como la salida de la ruta preestablecida, para así dirigirse hacia la víctima localizada, sin seguir una trayectoria preestablecida o algún parámetro definido. Las entradas de este comportamiento son las señales del sensor compás y de los *encoders* que le permitieron al robot conocer su posición en todo momento. La salida de este comportamiento suprime el comportamiento Avoid establecido en la capa Path que les permitió establecer estímulos a los motores M1 y M2. A su vez, este comportamiento es inhibido por el comportamiento Scan cuando el robot en su recorrido detectó un obstáculo (figura 156).

Figura 156. Descripción I/O para el comportamiento Sail



Para el proceso de validación del comportamiento se implementó el algoritmo que se destaca en la figura 157. En este se muestra el diagrama de autómatas finitos para el comportamiento Sail y que está gobernado por 4 estados: navegar, obstáculo, no obstáculo y víctima, los cuales, por medio de las siguientes señales, presentaron procesos de activación y desactivación, a saber:

- *Lectura ult:* indica el registro del sensor ultrasonido y permite determinar la presencia de obstáculos sobre el camino.
- *enc 1:* esta variable registra en todo momento el valor del *encoder* del motor de la rueda derecha. Por medio de la comparación con la variable enc_2, permite conocer el movimiento que está presentando el robot en todo instante de tiempo. También es una variable clave en el proceso de cálculo cinemático para la plataforma robot.
- *enc 2:* esta variable registra en todo momento el valor del *encoder* del motor de la rueda izquierda. Por medio de la comparación con la variable enc_1, permite conocer el movimiento que está presentando el robot en todo instante de tiempo.

También es una variable clave en el proceso de cálculo cinemático para la plataforma robot.

- *M1*, *M2:* corresponden al motor derecho (M1) y al motor izquierdo (M2).

Durante el proceso de experimentación, inicialmente al recibir un estímulo de la capa comportamental Search, que le notificó sobre la presencia de la víctima, el robot describió una ruta como la mostrada en la figura 158, donde se evidencian 2 trazos, uno de color gris claro, el cual representa la ruta ideal que debería tomar el robot para dirigirse a la víctima y el trazo de color negro que representa la ruta tomada por el robot para llegar hasta la fuente de calor. Teniendo en cuenta la descripción del comportamiento, el desplazamiento realizado por el robot es válido, ya que no se debe seguir ningún tipo de patrón de ruta o camino para llegar al punto donde se encuentra la fuente de calor, el trazo de la ruta deseada no es más que una guía para indicar un posible "camino óptimo".

Figura 157. Máquina de estado para la implementación del comportamiento Sail

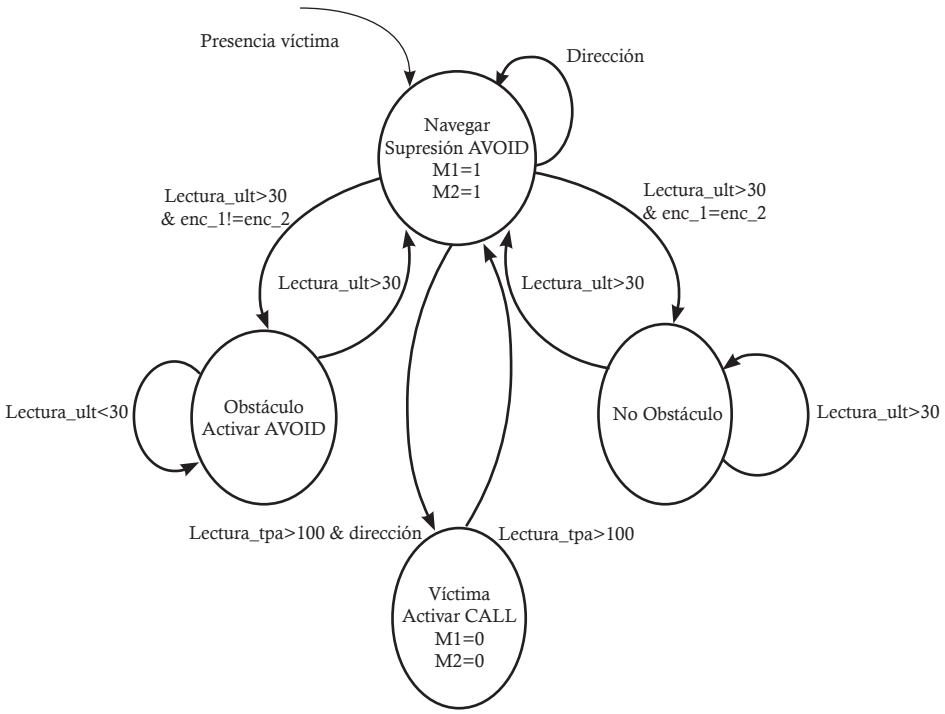
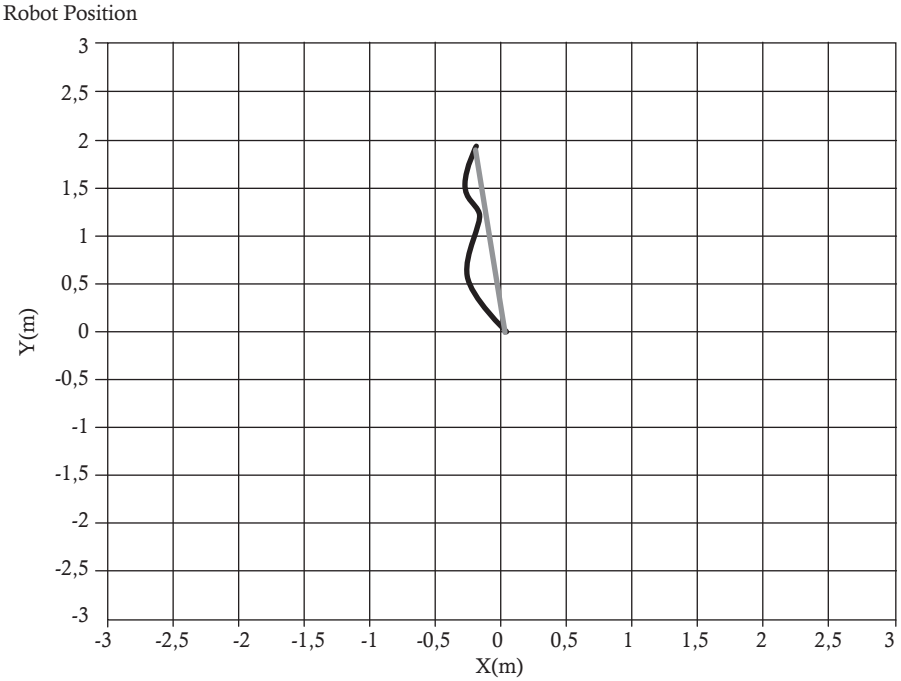


Figura 158. Ruta realizada por el robot como parte del proceso de experimentación para la validación del comportamiento Sail



Comportamiento Scan

Este comportamiento tiene como objetivo la búsqueda y la detección de un obstáculo mientras se está ejecutando el comportamiento Sail, tomando la señal del sensor de ultrasonido (que permite la medición de la distancia y ubicación del obstáculo con respecto al robot) y generando una señal de control al motor M3 (motor dedicado para la implementación de un radar básico). El comportamiento Scan activa el comportamiento Avoid e inhibe el comportamiento Sail, permitiendo que el robot pueda sortear el obstáculo y seguir el recorrido hacia la víctima localizada, una vez sea superado el obstáculo. Es importante aclarar que, aunque este comportamiento presenta un funcionamiento similar al descrito con el mismo nombre en la capa Path, este se encuentra ubicado en una capa de mayor jerarquía y su funcionamiento no es exactamente igual.

En la descripción del anterior comportamiento, el robot solo tenía la capacidad de dirigirse a la fuente de calor, siempre y cuando no se presentara ningún tipo de obstáculo en su camino; sin embargo, este tipo de situación no suele presentarse con frecuencia en una tarea de búsqueda de víctimas en un entorno colapsado, ya que por la misma naturaleza del evento el camino hacia la fuente de calor no está libre. Por ello, y para validar el funcionamiento del comportamiento SCAN, en el proceso de experimentación, el robot escaneó su entorno bajo distintas situaciones: el robot describiendo una ruta en línea recta en un pasillo (figura 160), el robot ante un obstáculo de frente (figura 161) y, por último, en presencia de la esquina de un cuarto (figura 162).

Figura 159. Descripción I/O para el comportamiento SCAN

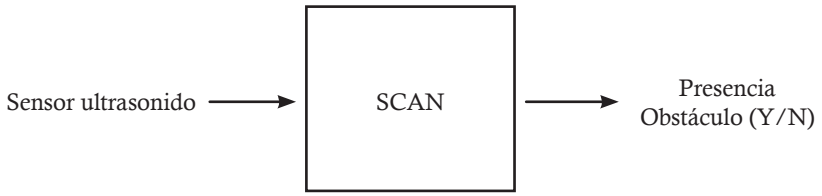


Figura 160. Escaneo del entorno realizado por el robot cuando se encuentra describiendo una trayectoria en línea recta por un pasillo

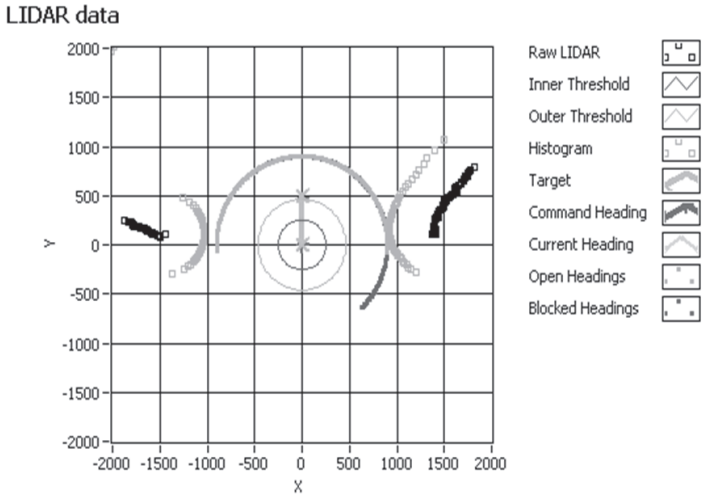


Figura 161. Escaneo del entorno realizado por el robot ante un obstáculo al frente

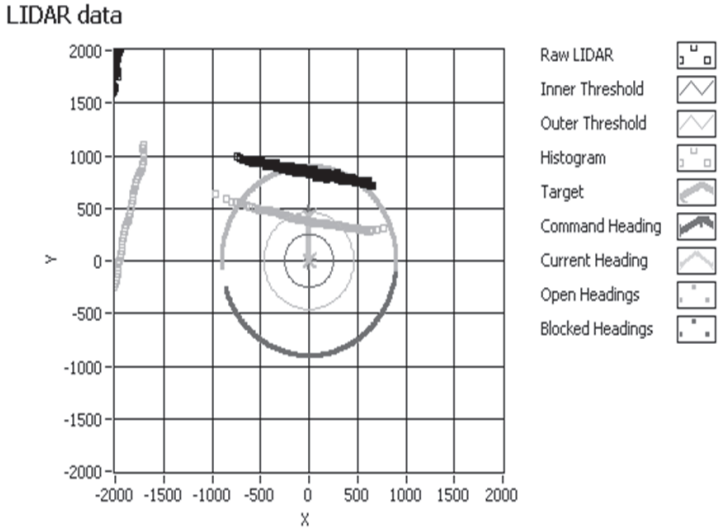
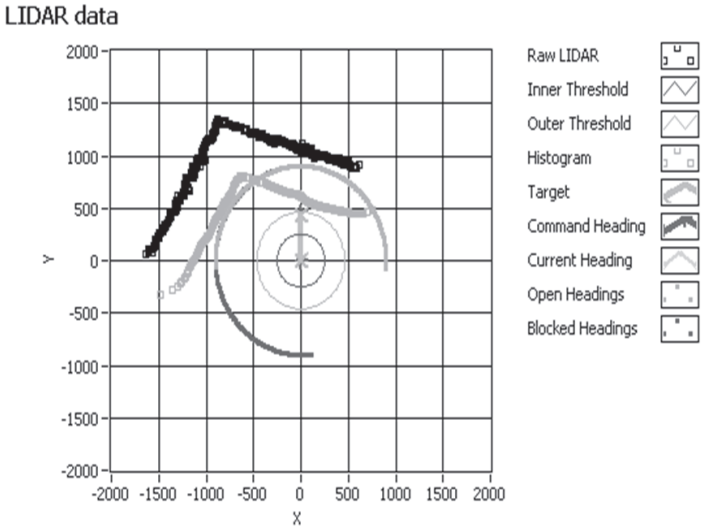


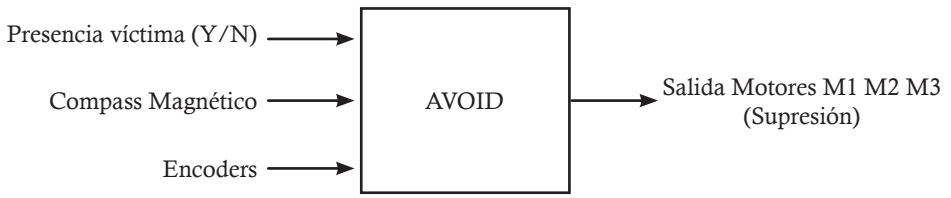
Figura 162. Escaneo del entorno realizado por el robot cuando se encuentre en una esquina interior



Comportamiento Avoid

Con la activación establecida por el comportamiento Scan, el comportamiento Avoid tiene como objetivo principal evadir el obstáculo sin olvidar que debe dirigirse hacia la víctima. Para hacer esto, este comportamiento suprime el comportamiento Sail. El desplazamiento del robot está determinado por la información suministrada por el algoritmo VFH+. Las entradas de este comportamiento son las señales de los *encoders* y del compás magnético necesarias para determinar la localización del robot, y sus salidas son estímulos a los motores M1 y M2. Una vez sea superado el obstáculo, este comportamiento debe liberar el comportamiento Sail que sigue ejecutándose (figura 163).

Figura 163. Escaneo del entorno realizado por el robot cuando se encuentre en una esquina interior



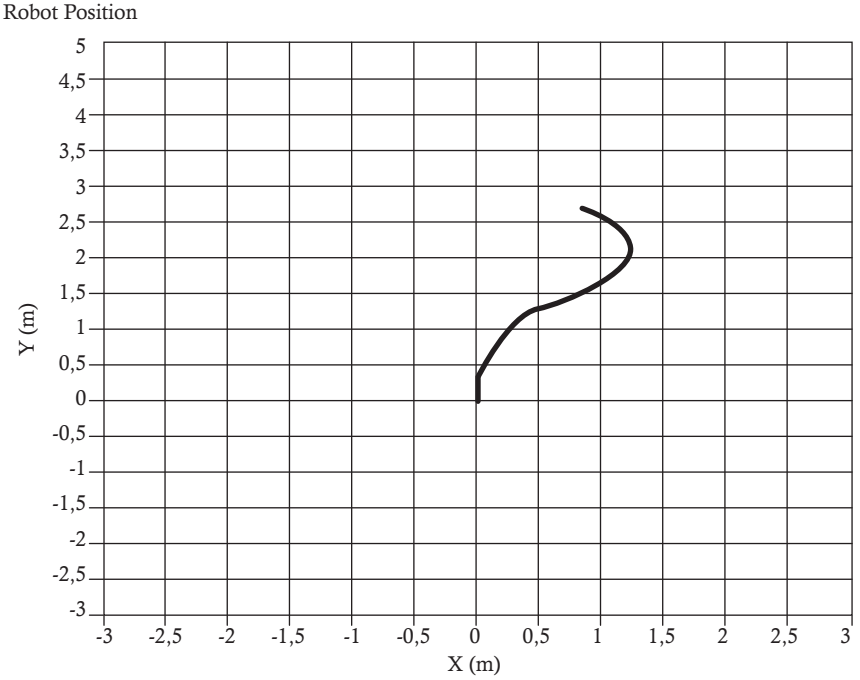
Con la implementación del comportamiento Avoid, se completó el esquema de la capa comportamental Wander; el proceso de experimentación permitió conocer el funcionamiento completo del robot con 3 de las 4 capas comportamentales que rigen el esquema completo de búsqueda y localización de fuentes de calor. En la figura

164 se muestra la trayectoria descrita por el robot que partió del origen (0, 0) con un obstáculo ubicado en la coordenada (0, 1.5) hasta el punto final (0.9, 2.8), donde se localizó la fuente de calor. En la ruta descrita por el robot se puede evidenciar el camino tomado por este para evadir el obstáculo y así llegar a la fuente de calor.

En la figura 164 se muestra una ruta descrita por la plataforma, donde se evidencia el camino tomado por el robot para evadir el obstáculo y así llegar a la fuente de calor. En esta prueba, se puso al robot de frente a su entorno con un obstáculo ubicado a 1.5 m de distancia del robot y una bombilla de 100 W como fuente de calor, ubicada en el punto (0.9, 2.5). Se observó el funcionamiento de las capas comportamentales Path, Search y Wander, actuando en conjunto para la búsqueda y la localización sobre un plano de una fuente de calor.

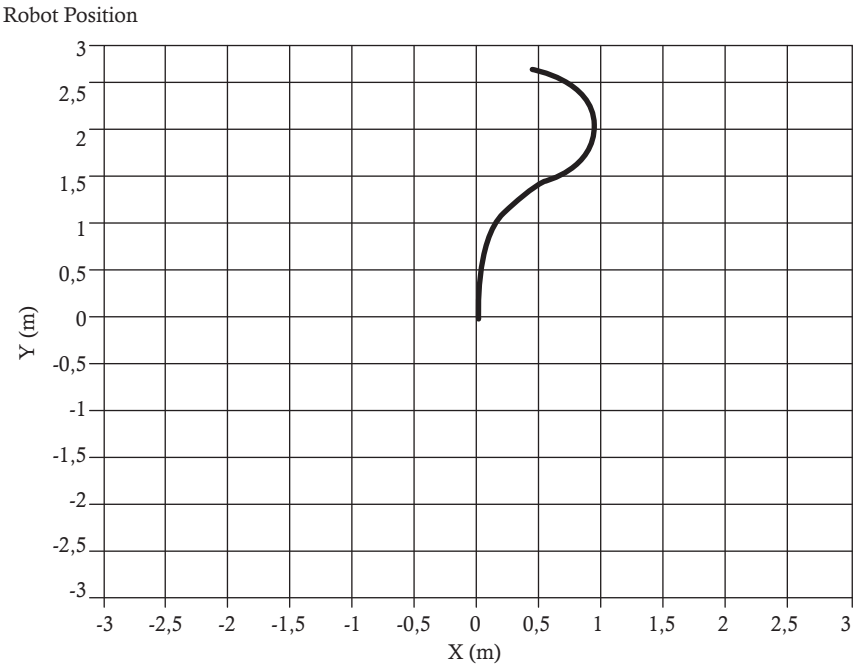
Dentro de este recorrido, cuando el robot detectó el obstáculo, activó el comportamiento Avoid, y una vez superado el obstáculo alrededor de los 2.5 m, el comportamiento Sail toma el mando sobre el robot permitiendo la navegación hacia la fuente de calor, aproximadamente sobre la coordenada (0.8, 2.7). Considerando esta coordenada y la coordenada inicial en donde se situó la fuente de calor (0.9, 2.5), el proceso de búsqueda y localización de la fuente de calor arrojó un error de localización de 11.1 % para el eje *X* y de 8 % para el eje *Y*, porcentajes que son mínimos considerando las distancias recorridas por el robot y que, como resultado principal, es un buen balance de funcionamiento para todas las capas comportamentales implementadas.

Figura 164. Trayectoria descrita por el robot en el proceso de búsqueda y localización de una fuente de calor



En la figura 165 se muestra un resultado más de la integración de 3 de las 4 capas comportamentales que conforman la estructura de control del robot. En ella se muestra la ruta hecha por este con condiciones de prueba similares. Se puede evidenciar el desarrollo de los comportamientos más básicos en la estructura comportamental; así, por ejemplo, el comportamiento Path se muestra sobre el primer metro de recorrido. Luego de este recorrido, el robot detectó un obstáculo y procedió con la activación del comportamiento Avoid. Asimismo, una vez superado el obstáculo (alrededor de los 2 m), el comportamiento Sail es el que toma el control del robot para llevarlo hasta el punto donde se ubicó la fuente de calor (aproximadamente sobre la coordenada (0.5, 2.6). Considerando estos datos y coordenadas donde se situó la fuente de calor (0.5, 2.5), el proceso de búsqueda y localización arrojó un error de localización de 4 % sobre el eje *Y*, porcentajes que son mínimos considerando las distancias recorridas por el robot. Nuevamente, como resultado general de la prueba realizada, se obtiene un buen balance de funcionamiento del conjunto de capas comportamentales implementadas sobre un robot a partir del funcionamiento de los comportamientos simples implementados.

Figura 165. Trayectoria descrita por el robot en el proceso de búsqueda y localización de una fuente de calor

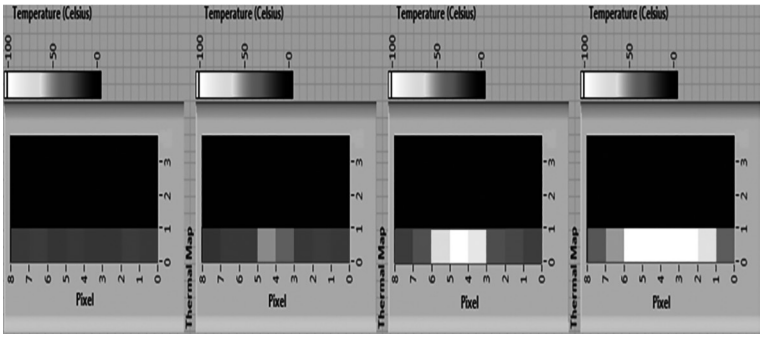


El funcionamiento del comportamiento Search se monitoreó constantemente por medio de todo el proceso de búsqueda y localización de la fuente de calor. En la figura 166 se muestra la secuencia de imágenes que corresponden a la evolución de la información suministrada por el comportamiento Search y, para su entendimiento,

se le sugiere al lector revisar el apartado donde se hace una descripción detallada de la interpretación para cada una de las gráficas generadas por el comportamiento Search. En el primer cuadro (izquierda), el mapa termal muestra sobre cada una de las 8 zonas de detección un color claro que denota la temperatura ambiente, esto sucede en el instante en el cual las acciones del robot están comandadas por el comportamiento Path. En el segundo cuadro, el mapa termal muestra el aumento de temperatura sobre las zonas 3 y 4, respectivamente, lo que indica una fuente de calor en esa dirección. Si existe la presencia de un obstáculo, se ejecuta el comportamiento Avoid; sin embargo, en este cuadro no se presenta ningún tipo de obstáculo.

En el tercer cuadro, el robot está bajo la influencia del comportamiento Sail, evidenciado en el aumento de la intensidad en la lectura del sensor termopila que registra una alta temperatura alrededor de la cuarta zona de detección. Finalmente, en el cuarto cuadro (derecha) de la figura 166, se muestra la información suministrada por el comportamiento en el momento en el cual el robot ha detectado y localizado la fuente de calor y que por la cercanía del robot sobre la fuente se registra un gran aumento de la temperatura en las 8 zonas de detección del sensor termopila que registra temperaturas por encima de los 100 °C.

Figura 166. Path, Scan, Avoid, Sail



Capa Call

El objetivo de esta capa fue garantizar el llamado del otro robot una vez localizada la víctima. Esta capa comportamental se constituyó de un solo comportamiento denominado con el mismo nombre.

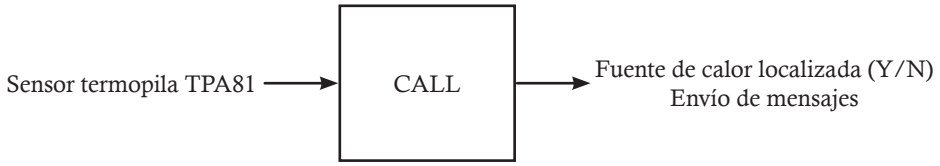
Comportamiento Call

Esta capa comportamental tiene como objetivo garantizar el llamado a los demás miembros del equipo de búsqueda y rescate, una vez sea localizada la víctima, y fue diseñada mediante la implementación de un solo comportamiento básico denominado con el mismo nombre.

Este comportamiento tomó como señal de entrada la información suministrada por la termopila TPA81 para conocer la distancia que presentaba el robot con la víctima. Una vez se encuentra el robot a una distancia menor a 10 cm, se envió un

mensaje de llamada al otro robot, en el cual se enviaban las coordenadas respectivas de la localización del robot. Este comportamiento suprime el comportamiento Avoid de la capa Wander (figura 167).

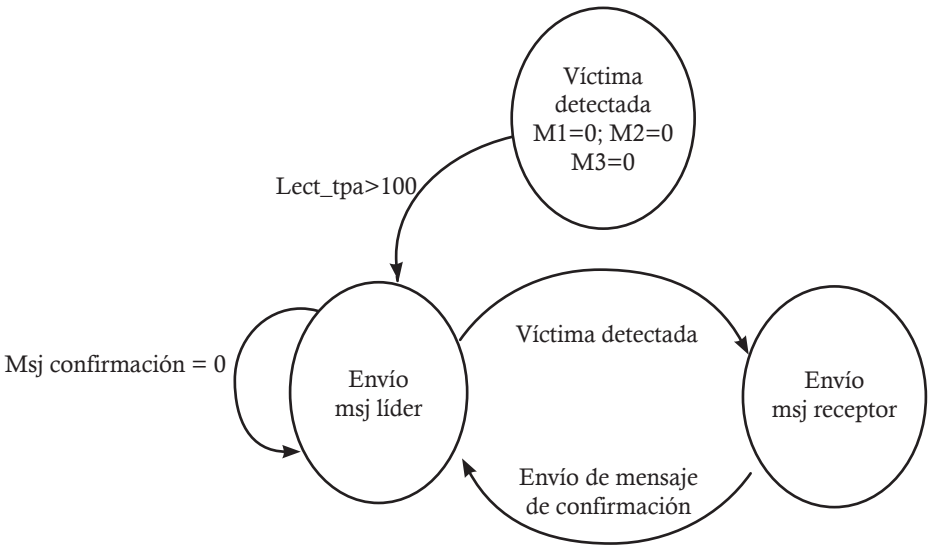
Figura 167. Descripción I/O para el comportamiento Call



En la figura 168 se muestra el diagrama de autómatas finitos que representa el funcionamiento del comportamiento Call; en este, se presentan 3 estados: víctima detectada, Envio msj lider y Envio msj receptor, los cuales se activan o desactivan dependiendo del estado de las siguientes señales de control:

- Msj confirmación: es el mensaje que envía el robot, que cumple el rol de esclavo cuando recibe un mensaje del líder de grupo.
- Lect tpa: es el valor registrado por el sensor termopila TPA81 durante todo el proceso de búsqueda y localización de una fuente de calor.
- Víctima detectada: es el mensaje que envía el líder de grupo una vez localizada la fuente de calor.

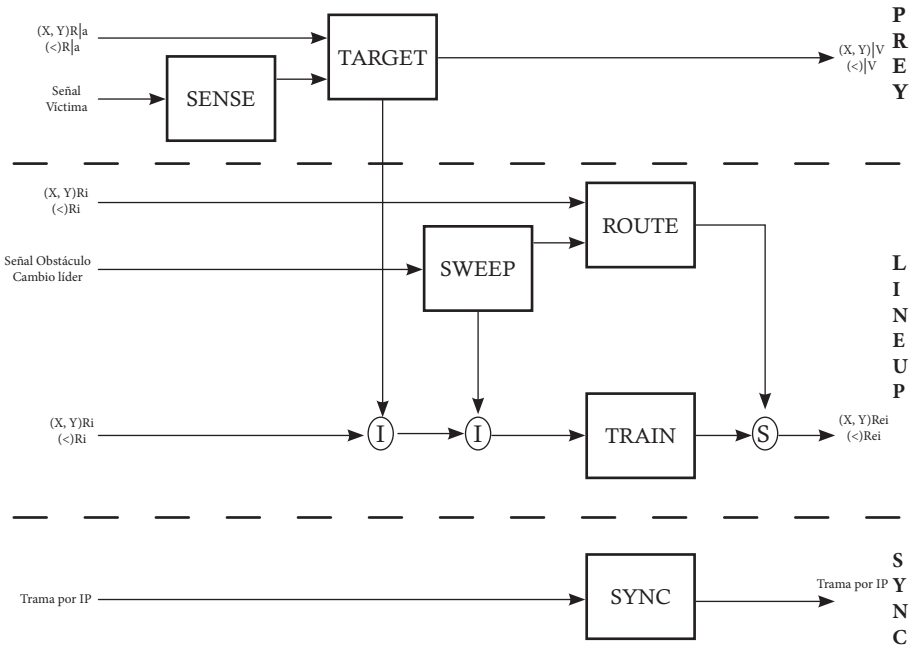
Figura 168. Descripción de máquinas de estado para el comportamiento Call



Comportamientos reactivos básicos implementados a nivel Multirobot

A nivel Multirobot, el proyecto decidió implementar una misma estructura de comportamientos que permitiera la sincronización, la coordinación y la colaboración de todos los miembros del equipo de búsqueda y rescate de víctimas con el objetivo de establecer el punto de partida para el diseño y el desarrollo de toda la coordinación y la comunicación del sistema en forma general (figura 169).

Figura 169. Descripción de comportamientos a nivel Multirobot

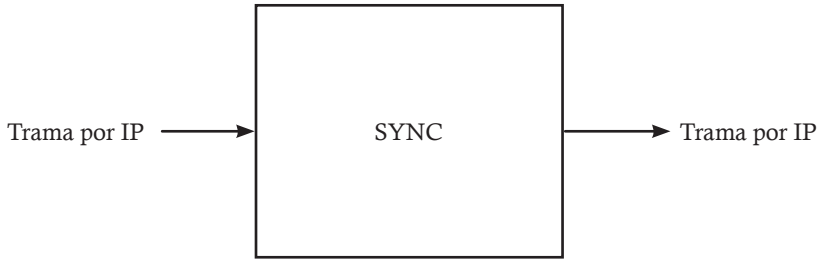


Capa Sync

Esta capa de comportamientos está conformada por un comportamiento básico que se encarga de desarrollar el proceso de sincronización entre todos los robots que conforman el equipo y que permite, entre otras cosas, iniciar al mismo tiempo el funcionamiento de todo el sistema Multirobot, verificando la formación por ejecutar y garantizando el enlace entre los robots y el equipo de control y seguimiento que vigila el intercambio de información a través del sistema de comunicación implementado.

El comportamiento básico denominado con el mismo nombre de la capa desarrolla el enlace entre los miembros del equipo Multirobot y el computador que permite establecer quién será el robot líder, encargado de la coordinación de tareas y procesos de coordinación requeridos por todo el sistema. En este comportamiento se tienen como señales de entrada y salida la información a transmitir mediante la implementación de la trama diseñada para tal fin (figura 170).

Figura 170. Descripción I/O para el comportamiento Sync



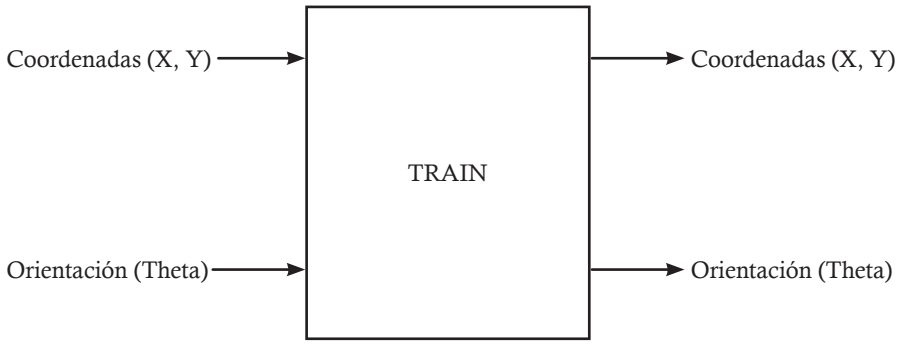
Capa Lineup

Esta capa de comportamiento, conformada por 3 comportamientos básicos (Train, Sweep y Route) tiene como objetivo mantener la formación preestablecida inicialmente para el desarrollo de la búsqueda de la víctima en la zona de prueba seleccionada por medio de la detección de obstáculos que permitan asegurar el recorrido, manteniendo, prioritariamente, la formación preestablecida para el equipo de búsqueda y rescate.

Comportamiento Train

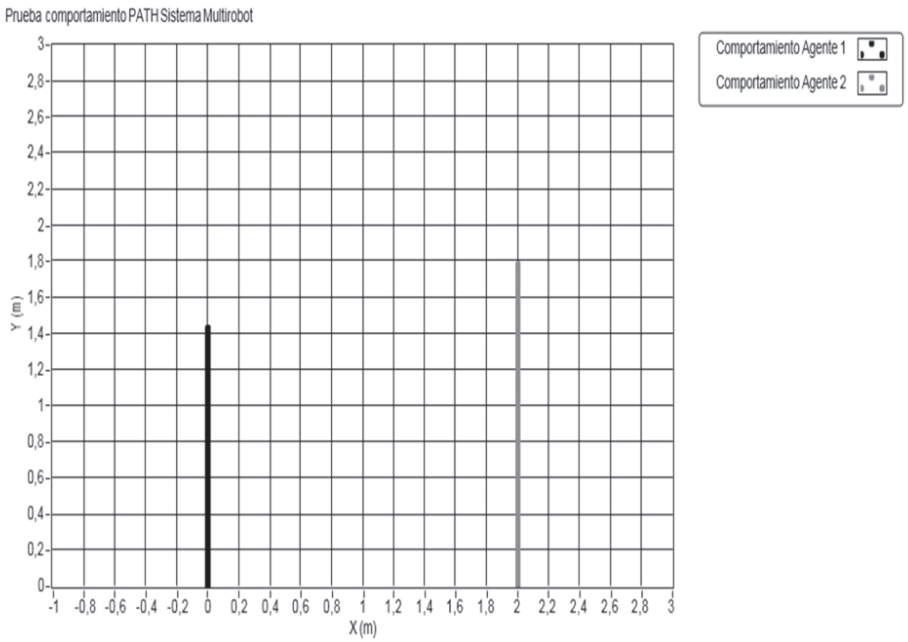
Este comportamiento garantiza la distancia de los robots por medio de la trayectoria establecida, que permita asegurar la formación determinada inicialmente que utiliza la posición (*X, Y*) y la orientación θ de cada robot para el cómputo de la posición relativa, cada uno de ellos permite determinar diferencias de desplazamiento en sus recorridos y las respectivas correcciones que permitan mantener la formación y la ruta preestablecida. La salida del comportamiento Train es una posición (*X, Y*) y orientación θ para cada robot que le permita corregir su posición de forma individual con respecto a los demás miembros que conforman el equipo. Este comportamiento es inhibido por el comportamiento Sweep cuando es detectado un obstáculo en el recorrido de la trayectoria (figura 171).

Figura 171. Descripción I/O para el comportamiento Train



En la etapa de pruebas fue implementada la formación de rastrillaje, técnica que se escogió como patrón de búsqueda de la fuente de calor. Durante el proceso de experimentación, se pusieron los robots sobre un plano (*X, Y*), separados por una distancia de 1 m y configurando el sistema Multirobot para que realizara una trayectoria en línea recta sobre el entorno. En la figura 172 se presenta el recorrido realizado por el equipo Multirobot en el cual, la línea de color gris claro representa el recorrido realizado por el robot 1, mientras que la línea roja representa el recorrido realizado por el robot 2. De la figura se puede apreciar que la distancia de separación entre los 2 robots se mantiene durante la ejecución de la ruta en línea recta. Asimismo, al finalizar la prueba, se encontró una diferencia de 20 cm sobre la coordenada final *Y* de cada robot, lo que no representa ningún tipo de alteración sobre el funcionamiento del sistema Multirobot, considerando los errores no sistemáticos que se pueden presentar en todo el sistema.

Figura 172. Prueba Path Multirobot



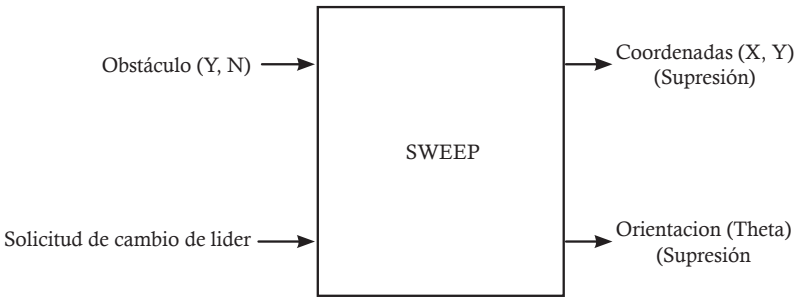
Comportamiento Sweep

El objetivo de este comportamiento es revisar mediante los 2 robots la presencia de un obstáculo que impida el seguimiento de la trayectoria y, por ende, la formación preestablecida. Esta señal puede ser enviada por cualquiera de los robots que, a su vez, realizan la activación del comportamiento Route.

Durante la ejecución del comportamiento Sweep se presenta la ruptura de la formación cuando es detectado el obstáculo y habrá una petición del cambio de líder de la formación originada por el(los) robot(s) que detecte el obstáculo, con el objetivo

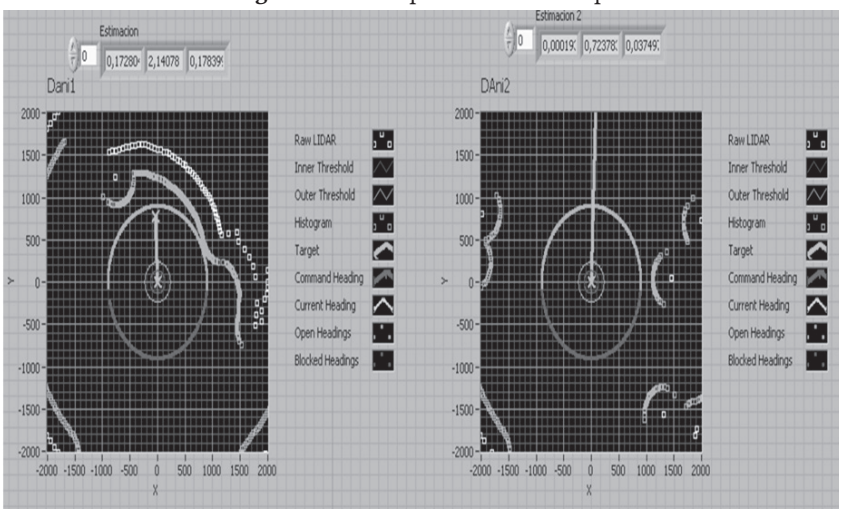
de mantener la formación en todo momento y no perder un integrante en el recorrido por un obstáculo presente. El otro robot esperará al líder temporal hasta que sea superado el obstáculo. Las entradas del comportamiento son las señales de detección de obstáculo enviadas por un robot (comportamiento SCAN) y la solicitud de cambio de líder temporal (figura 173).

Figura 173. Descripción I/O para el comportamiento Sweep



En el proceso de experimentación para la validación del comportamiento se monitoreó por medio de LabVIEW la información suministrada por cada uno de los integrantes del equipo durante la ejecución de una ruta libre sobre el entorno de pruebas. En la figura 174 se muestra una parte de la interfaz de usuario donde se notifica el estado actual de cada uno de los robots respecto a la detección de obstáculos sobre el camino. A la izquierda de la figura 174 se muestra el momento en el que el robot 1 detectó un obstáculo sobre su camino; por su parte, el robot 2 (derecha) no evidencia ningún tipo de obstáculo sobre su camino. La información suministrada por cada uno de los robots hace parte del comportamiento Sweep y es utilizada para activar el comportamiento Route.

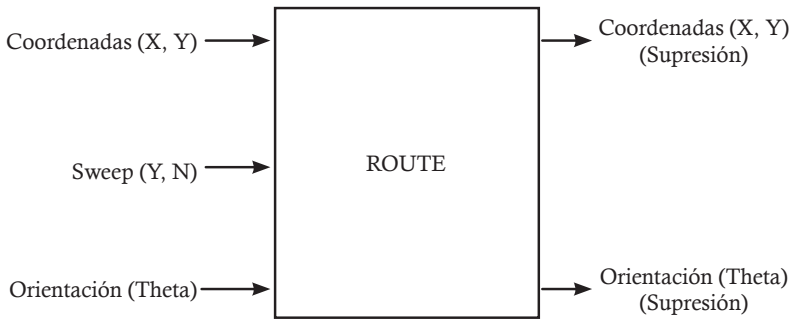
Figura 174. Comportamiento Sweep



Comportamiento Route

El comportamiento Route es activado por el comportamiento Sweep, con un objetivo fundamental: mantener coordinado el funcionamiento del sistema Multirobot. Para ello, mientras que un robot supera el obstáculo que ha detectado, los demás integrantes del equipo deben disminuir su velocidad con el objetivo de dar una espera al integrante que está superando el obstáculo y así mantener la formación previamente establecida (figura 175).

Figura 175. Descripción I/O para el comportamiento Route



Para la validación del comportamiento Route se realizaron 2 tipos de pruebas, en las que se verificó el desempeño del sistema Multirobot. La primera de ellas (figura 176) representa la situación en la que el robot 1 (línea negra) detectó la presencia de un obstáculo sobre su camino y procedió a ejecutar la rutina de evasión mientras que el robot 2 (línea gris claro) describió una ruta en línea recta por un camino libre, sin obstáculos. Una vez el robot 1 detecta un obstáculo y empieza a ejecutar la rutina de evasión, el robot 2 suspende su desplazamiento sobre el plano hasta recibir una notificación por parte del líder (robot 1) de que ha superado exitosamente el obstáculo. En la figura también se pueden evidenciar un par de puntos aislados que representan la proyección del camino al cual deben regresar cada uno de los robots.

La segunda situación se evidenció con la presencia de obstáculos sobre el camino de cada uno de los robots. Aquí el comportamiento Route tomó la información suministrada por cada uno de los integrantes del equipo y coordinó la respuesta de cada uno de los robots haciendo que se realizara la tarea de evasión de obstáculos por cada uno de ellos para posteriormente retomar la ruta de búsqueda. En la figura 177 se muestran los resultados de este proceso de experimentación, en los que se dan a conocer las trayectorias seguidas por el sistema Multirobot y el desempeño por medio de su ejecución. De nuevo, y al igual que en la situación anterior, los puntos aislados representan la proyección del camino que deben retomar cada uno de los integrantes del sistema Multirobot después de evadir el obstáculo. En la figura se presenta un pequeño error entre el punto y la coordenada final proyectada por cada robot cercano a 20 cm, lo que se considera como normal teniendo en cuenta las distancias recorridas por el sistema Multirobot.

Figura 176. Prueba Avoid Multirobot

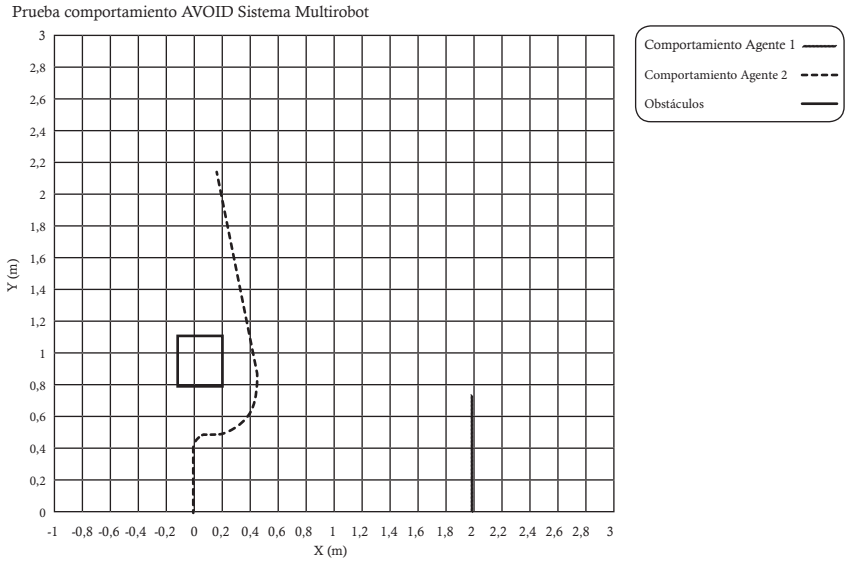
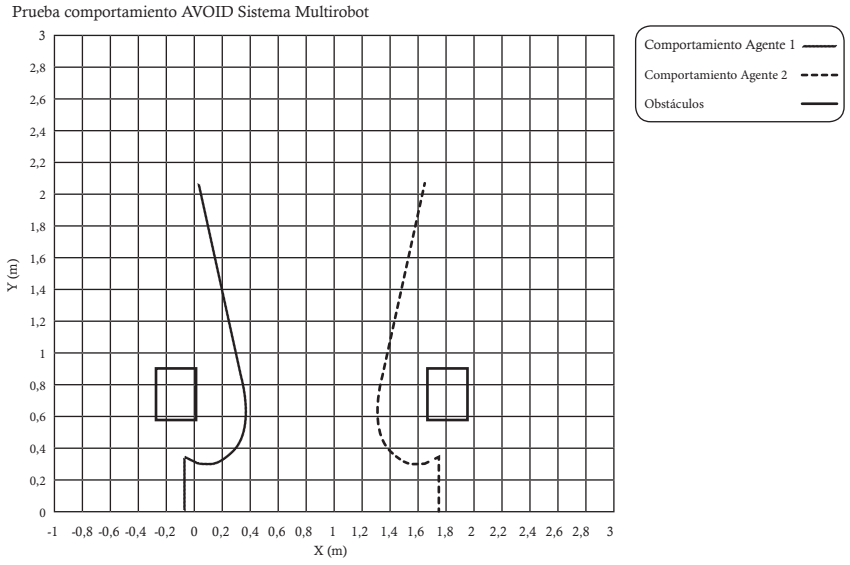


Figura 177. Segunda prueba Avoid Multirobot



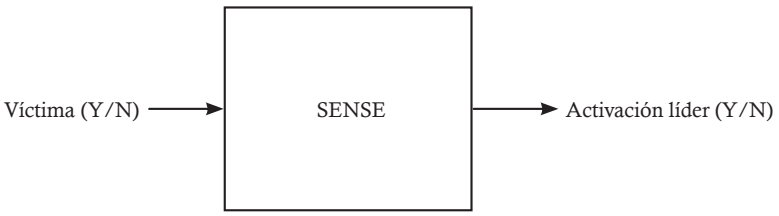
Capa Prey

Esta capa comportamental, conformada por 2 comportamientos básicos (Sense y Target), es la encargada de llevar todo el equipo Multirobot a la posición de la víctima localizada, y así poder desarrollar, posteriormente, las operaciones de rescate.

Comportamiento Sense

Este comportamiento básico está encargado de la supervisión de los robots con respecto a la presencia de la víctima en la zona de búsqueda por medio de la señal que puede ser enviada por un robot a los demás miembros del equipo. El robot que active el comportamiento Sense será autorizado para romper la formación de búsqueda; se convierte en el líder absoluto del equipo Multirobot. Este comportamiento permite la activación del comportamiento Target. La señal de entrada del comportamiento Sense es una señal enviada desde cualquier miembro del equipo Multirobot (originada por el comportamiento Search) inferior del robot (figura 178).

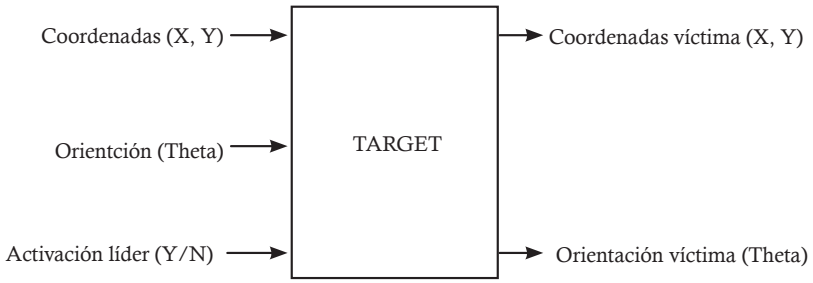
Figura 178. Descripción I/O para el comportamiento Sense



Comportamiento Target

Este comportamiento es activado por el comportamiento Sense y establece el liderazgo absoluto del robot que detecta la víctima, permitiéndole romper la formación preestablecida para dirigirse, en el menor tiempo, hacia la víctima, y superando los obstáculos que se le presenten en el recorrido. Cuando el robot líder global localice y determine la posición de la víctima, deberá enviar la información de su posición (X, Y) y su orientación (θ) a los demás miembros del equipo, suprimiendo el comportamiento Train, para que ellos se dirijan inmediatamente hacia la víctima localizada. La entrada a este comportamiento es la posición (X, Y) y orientación (θ) de cada robot, y la salida del comportamiento es la posición (X, Y) y la posición (θ) de la víctima. Cuando todos los robots se ubiquen sobre la posición de la víctima y en una zona de seguridad definida (una distancia mínima a la posición), el comportamiento Target enviará señal a todos los robots para que detengan su funcionamiento (figura 179).

Figura 179. Descripción I/O para el comportamiento Target



El proceso de validación que se llevó a cabo fue para la capa comportamental Prey en conjunto, debido a que la integración de los comportamientos presentes en ella permite una mejor respuesta en cuanto a los resultados mostrados. En el proceso de validación, se monitoreó el desempeño del sistema bajo la presencia de 2 situaciones: en la primera no se presentaron obstáculos sobre el camino del sistema Multirobot y en la segunda se evidenció la presencia de obstáculos sobre el camino del sistema Multirobot.

En la figura 180 se muestra cómo el sistema empieza su rutina de rastrillaje en línea recta y cómo uno de los integrantes del sistema detectó la presencia de una fuente de calor; posterior a ello, se dirige a ella y una vez localizado el punto de la fuente, solicita el llamado del otro integrante del sistema Multirobot.

La segunda situación (figura 181) muestra el desempeño del sistema Multirobot ante la presencia de obstáculos sobre el camino. En este caso, de encontrar obstáculo, cada uno de los integrantes estuvo en la capacidad de evadirlo para posteriormente seguir con el proceso de búsqueda. Una vez detectada la víctima por parte de uno de los robots (robot 2), este tomó el liderazgo del sistema y se dirigió a la fuente, mientras que el otro robot (robot 1) mantuvo su posición hasta la notificación con la que se envió la información concerniente a las coordenadas donde se ubicó la fuente de calor. Por último, el sistema se reunió alrededor de la fuente de calor, terminó así el proceso de búsqueda y localización de víctimas en entornos dinámicos.

Figura 180. Validación de la capa comportamental Prey

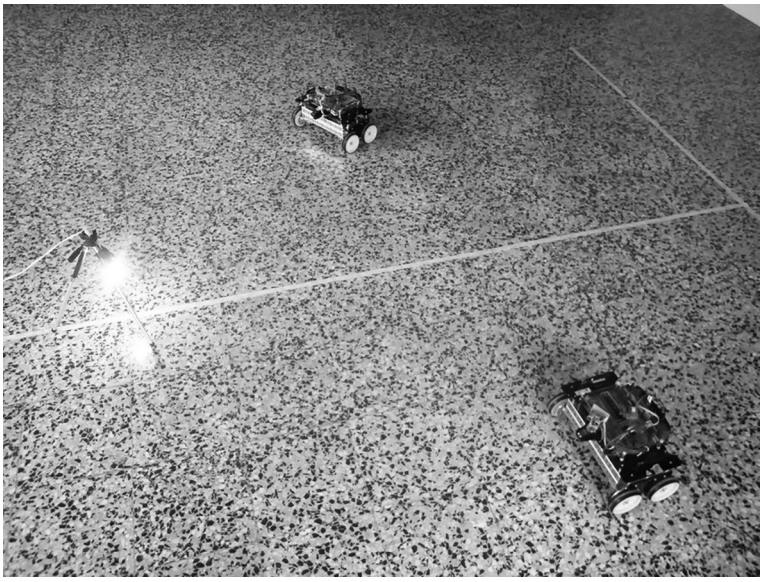
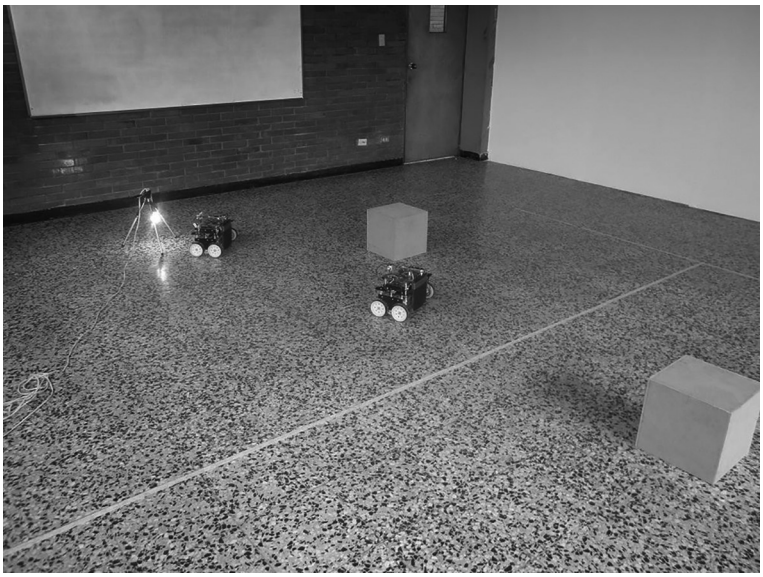


Figura 181. Validación de la capa comportamental Prey



Conclusiones

- La inclusión de agentes robots como miembros de un equipo de búsqueda y rescate urbano es, sin duda, un avance significativo en pro de atender de una forma más ágil y eficiente los desastres. Un robot puede dotarse de un sinnúmero de sensores que podrán ser más eficientes que los sentidos (tanto humanos como animales), lo cual permitirá la detección de un sinnúmero de peligros, con lo cual los demás integrantes del equipo estarán preparados y se reducirán las posibilidades de encontrarse con situaciones inesperadas.
- La ventaja más significativa de tener robots en los equipos de rescate es que se salvaguarda la integridad física del grupo humano, ya que se tiene conocimiento previo del terreno y de las condiciones a las que se enfrentan en el desastre. Además del tiempo de exposición a las condiciones adversas.
- En el desarrollo del proyecto fue necesario implementar una fusión de sensores para conocer las condiciones ambientales del entorno, donde la temperatura y la humedad desempeñan un papel fundamental, ya que dependiendo de estas 2 condiciones y la presencia de gases o fuentes de calor, se puede originar una emergencia.
- La caracterización del sensor se llevó a cabo con el datalogger RHT10, que tiene en su interior un sensor de la misma familia del SHT71, de alta precisión; por esta razón, las lecturas en las pruebas son cercanas.
- Se evaluó el sistema cooperativo basado en un proceso de experimentación donde se realizaron varias pruebas para la validación de cada uno de los comportamientos pertenecientes a las capas comportamentales tanto a nivel de un robot como a nivel Multirobot. Con base en esto, se dedujo el desempeño de cada uno de ellos tomando como referencia los errores presentados en la medida y las acciones que deberían ejecutar idealmente por medio de la interacción con el entorno.
- Se implementó un sistema de monitoreo en LabVIEW, en el cual el usuario puede observar la evolución de las tareas de búsqueda y localización de fuentes de calor ejecutadas por el sistema Multirobot. Algunos retardos se presentaron en la transmisión de datos que no pudieron ser registrados. Sin embargo, estos retardos resultaron irrelevantes para el sistema de monitoreo en tiempo real del sistema Multirobot.

- El filtro de Kalman mejoró el proceso de localización del robot sobre la zona de exploración, ya que para el seguimiento de rutas predefinidas, los errores de edometría ocasionaban lecturas incorrectas durante la búsqueda de la víctima. La técnica recursiva del filtro de Kalman permitió estimar la próxima posición de la plataforma correctamente con el ruido presente en el entorno.
- Una vez corregidos los problemas de localización de cada plataforma, se realizará una comunicación ordenador-robot con el objetivo de monitorear la posición del robot en instantes de tiempo determinados, lo cual permitirá dirigir las plataformas a puntos específicos en el entorno; además, se desarrollarán algoritmos de fusión sensorial implementados sobre LabVIEW, que redundarán la información para hacer una estimación aún más exacta.
- El problema de posicionamiento y planeación de rutas para un equipo Multirobot es complejo y de muchas aéreas del conocimiento; por esto, en esta investigación se probaron diferentes métodos y algoritmos que contribuyen a la solución de esta tarea mayor; uno de estos métodos es la visión artificial, la cual resultó muy conveniente, ya que ofrece un posicionamiento del robot en cada instante de tiempo.
- Las marcas artificiales implementadas en los robots fueron de gran ayuda, ya que ofrecieron una correcta solución al problema de posicionamiento en el espacio de los robots.
- Los sistemas de visión artificial son altamente susceptibles a cambios de iluminación; por esto, los algoritmos de filtrado son esenciales, en estos documentos se implementan 2 métodos que presentan una gran tasa de éxito; dichos algoritmos son los de filtrados por huecos y elementos.
- Network Streams es una herramienta de LabVIEW que permite el envío y la recepción de datos entre dispositivos; en este caso, entre el computador y la plataforma robótica, por medio de las direcciones IP de cada dispositivo; sin embargo, a diferencia del protocolo TCP/IP, se garantiza el intercambio de datos aun en la presencia de intermitencias de señal wifi.
- La plataforma robótica DaNI 2.0 fue probada en espacios cerrados y abiertos medianamente uniformes y la respuesta presentada fue adecuada, ya que cuenta con elementos que la hacen robusta e idónea para la aplicación que se deseaba; en este caso, cumplió a cabalidad con el objetivo del algoritmo que era la detección y evasión de obstáculos en los entornos a explorar.
- La fusión entre los programas Matlab y LabVIEW, a pesar de ser 2 plataformas diferentes, presentó una buena compatibilidad en la respuesta deseada, ya que se permite hacer uso de las funciones propias de Matlab dentro de un bloque de LabVIEW; por tal motivo, se facilita la migración de un lenguaje a otro, lo que permitió utilizar la combinación entre los 2 programas.
- El controlador basado en emociones, con ganancias fijas, permite que el robot pueda seguir el perfil, ruta, de referencia, con mejoras que superan el 10% con respecto a la definición tradicional del error, y la aplicación de un controlador tradicional PID, con las mismas ganancias del controlador basado en emociones.

- El aprendizaje por refuerzo permite que la plataforma robótica se aproxime al modelo de referencia, sin importar si en un comienzo el error de posición y ángulo es grande. Se observó la influencia de la tasa de aprendizaje, y se concluye que no necesariamente aprender más rápido significa un mejor resultado.
- Los valores iniciales de los pesos de la red neuronal influyen sobre el comportamiento de la red; por lo tanto, en términos generales, no es posible aplicar pesos iniciales aleatorios, sino que por ensayo error deben seleccionarse. Si bien esto no representa darle la respuesta a la red neuronal, sí implica que la red no puede estar muy lejos de la respuesta para encontrarla.
- Se pudo verificar que es apropiado seleccionar la figura de Lissajous como modelo de referencia, dado que exige dinámicas lentas y rápidas, además de distintos radios de curvatura, positivos y negativos. Esto permite evaluar el rendimiento de cualquier controlador, con movimientos en espacio reducido.
- Es importante explorar en trabajos futuros el incluir una memoria, de manera que la mejor red neuronal pueda ser almacenada, y que esta puede ser llamada nuevamente cuando las condiciones dinámicas se repitan, sin necesidad de repetir el aprendizaje por refuerzo.
- Estudios posteriores deben explorar el proceso de cambio de modelo de referencia cuando se presentan obstáculos durante el recorrido ideal, lo cual hará que el robot pueda evitarlos. De igual manera, es importante que se continúen evaluando otras estrategias de control, que resulten en una base de la cual pueden compararse y escoger el control más adecuado en cada caso.

Referencias

[1] FEMA. www.fema.gov. 2012.

[2] Galvez J., "Manual de búsqueda y rescate – Bomberos", Voluntarios de la Provincia de Entre Ríos, 2012.

[3] Arkin R. C., Fujita M., Takagi T., and Hasegawa R., "An ethological and emotional basis for human-robot interaction, Robotics and Autonomous Systems", vol. 42, no. 1,2003, pp. 35

[4] Breazeal C., "Sociable machines, Expressive social exchange between humans and robots Doctoral Dissertation", Department of Electrical Engineering and Computer Science, MIT., vol. 1, no. 1, 2000.

[5] Wilkes D., Alford A., Cambron M., Rogers T., Peters R., and Kawamura K., "Designing for human-robot symbiosis", Industrial Robot, vol. 1, no. 26, 1999, pp. 4758.

[6] Burke J. L., "Moonlight in Miami: A Field Study of Human-Robot Interaction in the Context of an Urban Search and Rescue Disaster Response Training Exercise", Master of Arts, College of Arts and Sciences- University of South Florida, 2003.

[7] National Instruments, "Robotics Platform for Teaching, Research, and Prototyping", NI LabVIEW Robotics Starter Kit. Tomado de http://sine.ni.com/ds/app/doc/p/id/ds-217/lang/es. Consultado el 13 de abril de 2013.

[8] National Instruments. "NI LabVIEW Robotics Starter Kit, Robotics Platform for Teaching, Research, and Prototyping". Tomado de http://sine.ni.com/ds/app/doc/p/id/ds-217/lang/es. Consultado el 13 de abril de 2013.

[9] Figueroa O., Junior R., Palaguachi A., and Angel A., "Diseño y Elaboración de Guías de Laboratorio para la plataforma móvil LabVIEW Robotics Sbrio Starter Kit de NationalInstrument". Escuela Politécnica del Ejército, Departamento de Eléctrica y Electrónica. Sangolquí, Ecuador, 2012, pp. 73-75.

[10] FEMA, _www.fema.gov,_2012.

[11] Longoria R., "Teach Vehicle Steering and Simulation with LabVIEW Robotics Starter Kit (DaNI)", Tomado de http://www.me.utexas.edu/_longoria/VSDC/handouts/Vehicle_Turning_and_Simulation.pdf, Consultado en 2012.

[12] Ribeiro M., and Lima P., "Kinematics models of mobile robots", 2002, pp. 1000-1049.

[13] Gillespie T., "Fundamentals of vehicle dynamics", Society of Automotive Engineers Warrendale, PA, vol. 400, 1992.

[14] Dixon J., "Tires, suspension, and handling", 1996.

[15] Milliken W., and Milliken D., "Race car vehicle dynamics", SAE International, 1995.

[16] Lew J., Liu Y., Wu X., and Zhu J., "Omni-directional mobile robot controller design by trajectory linearization", American Control Conference, 2003. Proceedings, vol. 4, no. 1, 2003, pp. 3423_3428.

[17] Sotela, F. C. (n.d.). "Fusión de Datos Distribuida en Redes de Sensores Visuales Utilizando Sistemas Multi-Agente". Tomado de http://e archivo.uc3m.es/bitstream/10016/9495/1/Castanedo-thesis-Phd.pdf, 2010.

[18] LACOMET, "Mediciones de Humedad Relativa", Laboratorio Costarricense de Metrología. Tomado de http://www.lacomet.go.cr/index.php?option=com_content&view=article&id=214:mediciones&catid=86:humedad&Itemid=268

[19] Pérez M., Costa C., Flores G, and Carrillo J, "Sensor de temperatura y humedad sht11", Universidad Politécnica de Valencia, 2009.

[20] LACOMET, "Mediciones de Humedad Relativa", Laboratorio Costarricense de Metrología. Tomado de: http://www.lacomet.go.cr/index.php?option=com_content&view=article&id=214:mediciones&catid=86:humedad&Itemid=268

[21] utp, "El Microcontrolador PIC16F873", 2002, p. 22.

[22] Extech, "Manual del usuario Registrador de Humedad / Temperatura", Extech Instruments. Tomado de http://www.extech.com/resources/RHT10_UM-es.pdf, p. 12.

[23] Barragán A. J., "Instrumentación y Control Industrial, Introducción a los Sistemas de Medida y Control, errores en las mediciones". Tomado de: http://www.uhu.es/diego.lopez/ICI/tema1.pdf, 2014.

[24] "Histéresis, Capítulo 2, Instrumentación industrial". Tomado de: http://web.udl.es/usuaris/w3511782/Control_de_procesos/Unidades_files/Cap02_10-11.pdf, 2014.

[25] Chou, J., "Sensores Electroquímicos Principio de Operación Características y Aplicaciones". Tomado de: http://www.atsintech.com/tablas/ISTBook.pdf, 2014.

[26] Figaro, "tgs 3870 - for the detection of both methane and carbon monoxide applications: basic measuring circuit: Product information". Tomado de: http://www.figarosensor.com/products/docs/TGS%203870-B00%281213%29.pdf, 2016.

[27] I. Morsi, "A Microcontroller Based on Multi Sensors Data Fusion and Artificial Intelligent Technique for Gas Identification", IECON-33rd Annu. Conf. IEEE Ind. Electron. Soc., 2007, pp. 2203–2208.

[28] I. Morsi, "Discrimination between Butane and Propane in a Gas Mixture UsingSemiconductor Gas Sensors and Neural Networks", Arab Academy for Science and Technology, 2008.

[29] Iniciativa global del metano, "Emisiones mundiales de metano y oportunidades de atenuación", 2010, pp.1–4.

[30] Departamento de Salud y Servicios para Personas Mayores de New Jersey, "(METHANE) Efectos agudos sobre la salud. Efectos crónicos sobre la salud. Riesgo de cáncer. Riesgo para la reproducción. Otros efectos a largo plazo. Exámenes médicos", New Jersey, 2003.

[31] Gas Natural Fenosa, "Qué es el gas natural fenosa, Composición típica del gas natural". Tomado de: http://www.gasnaturalfenosa.com.co/co/hogar/el+gas+natural/1297102453941/que+es.htm, 2014.

[32] Saraco S., "Recomendaciones para la atención de las intoxicaciones por monóxido de carbono", Ministerio de Salud Gobierno Mendoza, 2012.

[33] VegaROBOkit, "MQ155 gas sensor, Character", Application. Tomado de: http://www.vegarobokit.com/index.php?route=product/product&product_id=432, 2014.

[34] R. Tem, "MQ-135 Gas Sensor", Technical Data, vol. 1, pp. 3–4.

[35] Agencia para Sustancias Tóxicas, "Resumen De Salud Pública", Madrid, 2004.

[36] Consejería de Sanidad. Dirección General de Salud Pública, "Riesgo quimico- accidentes grabes, óxido nitroso", Región de Murcia, Tomado de: https://www.murciasalud.es/recursos/ficheros/113921-Oxido_nitroso.pdf, 2007.

[37] Messer L.B, "Óxido nitroso", Buenos Aires, 2012.

[38] Centro para el Control y la Prevención de Enfermedades, "Realidades del Benceno". Tomado de: http://www.bt.cdc.gov/agent/benzene/espanol/facts.asp,2014.

[39] A. F. P. SEGUROS and S.A., "Benceno", Buenos Aires, 2001.

[40] Agencia para Sustancias Tóxicas y Registro de Enfermedades, "Resumen de Salud Pública – Benceno (Benzene)", Madrid, 2007.

[41] Ávila M., "Sensores de velocidad". Tomado de: file:///C:/Users/Multibot/Downloads/Sensores%20de%20velocidad.pdf, 2005.

[42] Lego, Mindstorms, "NXT Gyro Sensor for LEGO Mindstorms NXT", HiTechnic. Tomado de: http://www.hitechnic.com/cgibin/commerce.cgi?preadd=action&key=NGY1044, 2014.

[43] Synthesize-your-own NXT, "connector plug". Tomado de: http://www.philohome.com/nxtplug/nxtplug.htm, 2014.

[44] Super robótica, Robótica fácil, "sensor brújula digital CMPS03 s320160". Tomado de: http://www.superrobotica.com/s320160.htm,2014.

[45] Lego Mindstorms, "NXT Compass Sensor (NMC1034)", Robotics Invention System, RCX are trademarks of the LEGO Group. Tomado de: http://www.hitechnic.com/cgi-bin/commerce.cgi?preadd=action&key=NMC1034, 2014.

[46] Blogspot, "Factor de riesgo químico". Tomado de: http://factorderiesgoquimico.blogspot.com/2009/07/factor-de-riesgo-quimico.html, 2009.

[47] Aguilar F, Bernaola M., Gálvez V., Rams P., Sánchez M. T., Sousa M. E., Tejedor, J. N. "Riesgo químico: sistemática para la evaluación higiénica". Ministerio de Trabajo e Inmigración, Centro Nacional de Nuevas Tecnologías, Ed., 2010. pp. 242.

[48] ISTAS, "Riesgo químico, agentes químicos peligrosos, los peligros de los productos químicos y sus símbolos", Madrid. Tomado de: http://www.istas.net/web/index.asp?idpagina=3445, 2014.

[49] Departamento de Salud y Servicios para Personas Mayores de New Jersey, "Efectos agudos sobre la salud, efectos crónicos sobre la salud, riesgo de cáncer, riesgo para la reproducción, otros efectos a largo plazo. Exámenes médicos", vol. 1971, New Jersey. Tomado de http://www2.udec.cl/matpel/sustanciaspdf/m/METANO.pdf , 2002, pp. 6.

[50] Global Methane Initiative, "Emisiones mundiales de metano y oportunidades de atenuación. Tomado de: https://www.globalmethane.org/documents/analysis_fs_spa.pdf, 2010, pp. 1-4.

[51] Saracco S. "Recomendaciones para la atención de las intoxicaciones por monóxido de carbono", Centro de Información y Asesoramiento Toxicológico Mendoza, Plan de Emergencias Médicas y Catástrofes, Ministerio de Salud, Gobierno de Mendoza, Tomado de: http://www.hazmatargentina.com/descargas/toxicologia/atencion_monoxido.pdf, 2012, pp. 1– 11.

[52] Departamento de Salud y Servicios para Personas Mayores de New Jersey, "Riesgo de cáncer, riesgo para la reproducción y otros efectos a 115 años largo plazo", New Jersey. Tomado de: http://www2.udec.cl/matpel/sustanciaspdf/m/MONOXIDODECARBONO.pdf, 2006, pp. 6.

[53] ATSDR. "Resumen de Salud pública-monóxido de carbono", Madrid. Tomado de: http://www.atsdr.cdc.gov/es/phs/es_phs201.pdf, 2012, pp.1-7.

[54] Messer. "Óxido nitroso", Buenos Aires. Tomado de: http://www.infrasal.com/attachments/article/49/Oxido_nitroso INFRASAL.pdf, 2012, pp.1-2.

[55] Infra del salvador S.A. "OXIDO NITROSO N 2 O", Salvador. Tomado de: http://www.infrasal.com/attachments/article/49/Oxido_nitroso INFRASAL.pdf, 2004, pp.4.

[56] Departamento de Salud y Servicios para Personas Mayores de New Jersey. "Derecho a Saber Hoja Informativa sobre Sustancias Peligrosas", New Jersey. Tomado de: http://nj.gov/health/eoh/rtkweb/documents/fs/0084sp.pdf, 2009, p. 6.

[57] New Jersey Departament of Health. "Exámenes médicos Efectos crónicos sobre la salud Riesgo de cáncer Riesgo para la reproducción Otros efectos a largo plazo", (Vol. 1005, New Jersey. Tomado de: http://www2.udec.cl/matpel/sustanciaspdf/a/AMONIACO.pdf, 1998, p. 6.

[58] Ministerio de Ambiente, "Amoniaco", Colombia. Tomado de: http://www.minambiente.gov.co/documentos/Guia5.pdf, pp 97-112.

[59] UNAM. "Hoja de seguridad del benceno", Universidad Autónoma de México, Vol. 8789, Ciudad de México. Tomado de: http://www.quimica.unam.mx/IMG/pdf/5benceno.pdf, pp. 1–6.

[60] Centro para el Control y la Prevención de Enfermedades. "Realidades del Benceno". Tomado de: http://www.bt.cdc.gov/agent/benzene/espanol/facts.asp

[61] Centro Landivar para el control del tabaco, "Los contaminantes del tabaco". Tomado de: http://www.url.edu.gt/otros_sitios/noTabaco/01-02cont.htm, 2014.

[62] Lacoma T., "Cuál es el gas emitido al quemar madera, material particulado, NOx y COVs". Tomado de: http://www.ehowenespanol.com/gas-emitido-quemar-maderasobre_ 314829/, 2014.

[63] Martínez41338, "Consecuencias del tabaquismo, Composición del cigarrillo". Tomado de: http://martinez41338.wordpress.com/consecuencias-del-tabaquismo-5/, 2014.

[64] Muñoz, P. R., Antonio, T., and Montemayor, S., "Procesamiento de Imagines", Universidad Rey Juan Carlos, España. Tomado de: http://www.etsii.urjc.es/~asanz/documentos/MemoriaKalmanJun03.pdf, 2003.

[65] Bermúdez G., "Modelamiento cinemático y odométrico de robots móviles: aspectos matemáticos", Revista Tecnura, vol. 20, no. 12, 2003.

[66] Sanz A., "Filtro de kalman". Tomado de: http://www.etsii.urjc.es/~asanz/documentos/MemoriaKalmanJun03.pdf, 2013.

[67] Logitech. "Logitech QuickCam 9000 pro review". Tomado de: http://www.logitech.com/repository/1403/pdf/25618.1.0.pdf. Consultado el 7 de octubre de 2013.

[68] Bermúdez G, Pérez M, and Vanegas A. "Algoritmo de detección de obstáculos, basado en segmentación de Imágenes con cámara abordo en el robot Dani

2.0". Universidad Distrital Francisco José de Caldas, Facultad Tecnológica. Revista Visión Electrónica. 13 de noviembre de 2013.

[69] Benítez J., Hueso J, "Introducción a Matlab", Departamento de Matemáticas aplicada, Universidad politécnica de valencia, s. f.

[70] "Introducción al labView", Página web: http://ftp.ehu.es/cidira/dptos/depjt/Instrumentacion/BKANGEL/10_LabVIEW/Introducci%F3n. PDF

[71] MathWorks, "Documentation bwreadopen". Tomado de: http://www.mathworks.com/help/images/ref/bwareaopen.html, visitada el 07-11-2013.

[72] Microsoft, "Manual del Sensor Kinect". Tomado de: http://download.microsoft.com/download/f/6/6/f6636beb-a352-48ee-86a3- bd9c0d4492a/kinect-manual.pdf. Visitado el 17-05-2014.

[73] Ayuso L. F., "Adquisición de información de profundidad mediante la técnica Structured Light, Three Phase-Shift", Madrid, España, 2011.

[74] Córdova F. A., "Detección de robo/abandono de objetos en interiores utilizando Cámaras de Profundidad", Madrid, España, 2012.

[75] Gizmologia, "Gizmologia.com", Febrero 2013. Tomado de: http://i0.wp.com/gizmologia.com/files/2013/02/kinect-21.jpg?resize=934%2C454. Visitada el 23-09-2014.

[76] Viñals J., "Localización y generación de mapas del entorno (SLAM) de un robot por medio de una Kinect", Valencia, 2012, pp. 39-40.

[77] Ilvay R. B., "Sistema de educación para niños de 3 a 5 años, mediante un robot controlado por el sensor Kinect", Riobamba, Ecuador, 2014, pp. 38-45.

[78] Socialphy, "Do it yourself: Kinect teardown", 2012. Tomado de: http://www.socialphy.com/posts/do-it-yourself/7717/Do-it-yourself_-Kinect-teardown.html. Visitada el 20-08-2014.

[79] WIX, "Animus Project Hardware" 2012. Tomado de: http://animusproject.wix.com/web/apps/blog/tag/hardware. Visitada el 20-08-2014.

[80] Pensamientos Computables, "Kinect: Como Funciona su 3D Body Tracking", 2010. Tomado de: http://www.pensamientoscomputables.com/img/kinect/esquema.jpg. Visitado el 17-09-2014.

[81] Mathe L., Samban D., and Gómez G., "Estudio del funcionamiento del sensor Kinect y aplicaciones para bioingeniería", Córdoba, Argentina, 2012.

[82] Microsoft, "Kinect for Windows SDK v1.8". Tomado de: http://www.microsoft.com/en-us/download/details.aspx?id=40278&751be11f-ede8-5a0c-058c-2ee190a24fa6=True. Visitado el 23-07-2014.

[83] Microsoft, "Kinect Sensor TV Mounting Clip". Tomado de: http://www.microsoftstore.com/store/msusa/en_US/pdp/Kinect-Sensor-TV-Mounting-Clip/productID.253726200. Visitado el 23-08-2014.

[84] Pizarro D., "Localización de robots móviles en espacios inteligentes utilizando cámaras externas y marcas naturales", Alcalá de Henares, Madrid, España, 2008.

[85] Aristondo J. "Algoritmo de reconocimiento de forma y color para una plataforma robótica", San Sebastián, España, 2010.

[86] Blackmore B. S. and Choudhury S., "Fuzzy Landmark Detection in Simultaneous Localisation and Mapping for Agricultural Robots", Pittsburgh, Pensilvania, Estados Unidos, 2008.

[87] Federal Emergency Management Agency FEMA, "Urban Search & Rescue". Tomado de: http://www.fema.gov/urban-search-rescue. Visitado el 23-03-2014.

[88] Cañamero L., "Designing emotions for activity selection in autonomous agents", Emotions in Humans and Artifacts, Cambridge, MA: The MIT Press, 2003, pp. 115-148.

[89] Scheutz M., "Useful roles of emotions in artificial agents: a case study from artificial life", Proceedings of AAAI 2004, AAAI Press, 2004, pp. 31-40.

[90] Scherer, K. R. "Psychological models of emotion". In Joan C. Borod (Ed.), The Neuropsychology of Emotion", New York: Oxford University Press. 2000, pp. 137-162.

[91] Marinier R., and Laird J., "Toward a comprehensive computational model of emotions and Feelings", 6th International Conference on Cognitive Modeling, 2004, pp. 1- 6.

[92] Sander D., Grandjean D., and Scherer K., "A systems approach to appraisal mechanisms in emotion", Neural Networks, Science Direct, vol. 18, 2005, pp. 317- 352.

[93] Russell J., and Barrett F., "Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant", Journal of Personality and Social Psychology, vol. 76, no 5, 1999, pp. 805-819.

[94] Posner J., Russell J., and Peterson B., "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology", Development and Psychopathology, vol. 17, 2005, pp. 715-734.

[95] Fagerberg P., Ståhl A., and Höök K., "eMoto: emotionally engaging interaction", Personal and Ubiquitous Computing, Vol. 8, 2004, pp. 377-381.

[96] Peter C., and Herbon A., "Emotion representation and physiology assignments in digital systems", Interacting with Computers, vol. 18, 2006, pp. 139-170.

[97] Sosnowski, S., Kühnlenz, K., and Buss, M., "EDDIE - An emotion-display with dynamic intuitive expressions", The 15th IEEE International Symposium on

Robot and Human Interactive Communication (RO-MAN06), Hatfield, UK, Sep. 2006, pp. 569-574.

[98] Gore B., and Jarvis P., "Modeling the complexities of human performance", Systems, Man and Cybernetics, 2005 IEEE International Conference, vol. 2, Oct. 2005, pp. 1604- 1609.

[99] Kadwane, S.; Kumar, A. y Karan B.; Ghose T. "Online trained simulation and DSP implementation of dynamic back propagation neural network for buck converter", ACSE Journal, Enero 2006, vol. 6, no 1, pp. 27-34.

[100] Nouri, K.; Dhaouadi, R. y Braiek, N. Nonlinear speed control of a dc motor drive system with online trained recurrent neural network. 9th IEEE International Workshop on Advanced Motion Control. 2006, pp. 704-708.

[101] Sztipanovits, J. "Dynamic backpropagation algorithm for neural network controlled resonator-bank architecture", IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, febrero 1992, vol. 39, no 5, pp. 99-108.

[102] Xiaosong, D.; Popovic, D. y Schulz-ekloff G. "The backpropagation neural network being capable of real-time identification", Proceedings of the 4th IEEE Conference on Control Application, septiembre 1995, pp. 572-577.

[103] Brooks R., "A Robust Layered Control System for a mobile robot". IEEE Journal on Robotics and Automation, vol. 2, no. I, 1986, pp. 14-23.

[104] Arkin R. C., and Balch T., "Cooperative Multiagent Robotic Systems 1 Introduction 2 Related Work and Background". Mobile Robot Laboratory, vol. 1, no. 1, 1998, pp. 1-16.

[105] Arkin R. C., "Chapter 4. Behavior Based Architectures". In Behavior Based Robotics, M. Dorigo, Ed. London, England: The MIT Press, 1998, pp. 123-173.

[106] Arkin R. C., "Chapter 3. Robot Behavior". In Behavior Based Robotics, M. Dorigo, Ed. London, England: The MIT Press. 1998, pp. 65-120.

[107] Babinec A., Vitko A., Duchon F., and M. Dekan, "Reactive Navigation of Mobile Robot Using Vector Field Histogram+". Modelling of Mechanical and Mechatronics Systems 4th International Conference, vol. 1, no. 1, 2011, pp. 1-10.

[108] Borenstein J., Koren Y., and Member S., "The Vector Field Histogram Fast Obstacle A voidance for Mobile Robots". IEEE Transactions on Robotics, vol. 7, no. 3, 1991, pp. 278-288.

[109] Ulrich I., and Borenstein J., "VFH+: reliable obstacle avoidance for fast mobile robots". Proceedings. EEE International Conference on Robotics and Automation (Cat. No.98CH36146), vol. 2. Tomado de: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=677362, 1998. pp. 1572-1577.

[110] Borenstein J., Koren Y., Mechanics A., and Arbor A., "Histogramic in motion mapping". IEEE Journal on Robotics and Automation, vol. 7, no. 4, 1991, pp. 535-539.

Autores

Miguel Ricardo Pérez Pereira

Ingeniero en Control e Instrumentación Electrónica de la Universidad Distrital Francisco José de Caldas con Maestría en Ciencias de la Educación de la Universidad San Buenaventura. Docente categoría Asociado en la Universidad Distrital Francisco José de Caldas, adscrito a la Facultad Tecnológica. Docente investigador del Grupo de Robótica Móvil Autónoma (Roma) clasificado con categoría C en Colciencias. Autor de artículos científicos en las áreas de visión artificial, procesamiento de señales, educación, digitales.

Giovanni Rodrigo Bermúdez Bohórquez

Ingeniero Electricista de la Universidad Nacional de Colombia con Maestría en Ingeniería Electrónica y de Computadores de la Universidad de Los Andes. Docente categoría Asociado en la Universidad Distrital Francisco José de Caldas, adscrito a la Facultad Tecnológica. Director del Grupo de Investigación en Robótica Móvil Autónoma (Roma) clasificado con categoría C, e investigador junior ante Colciencias. Autor de varios libros y artículos científicos en las áreas de instrumentación, sensórica y robótica móvil.

José Danilo Rairán Antolines

Doctor en Ingeniería de Sistemas y Computación de la Universidad Nacional de Colombia. Docente categoría Titular en la Universidad Distrital Francisco José de Caldas. Director del Grupo de Investigación en Control Electrónico (Gice). Autor de varios libros en las áreas de circuitos eléctricos, máquinas eléctricas y control. Autor de más de veinte artículos en congresos y revistas internacionales.

Este libro se
terminó de imprimir
en diciembre de 2017
en la Editorial UD
Bogotá, Colombia