



DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

**CARRERA DE INGENIERÍA EN ELECTRÓNICA,
AUTOMATIZACIÓN Y CONTROL**

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO EN ELECTRÓNICA,
AUTOMATIZACIÓN Y CONTROL**

**TEMA: “SISTEMA ROBÓTICO MÓVIL AUTÓNOMO
COOPERATIVO PARA MAPEO 2D DE OBSTÁCULOS Y
NIVELES DE TEMPERATURA EN UN AMBIENTE
CONTROLADO”**

AUTOR: MOYA ERAZO, CARLOS ANDRES

DIRECTOR: DR. ARCENTALES VITERI, ANDRÉS RICARDO

SANGOLQUÍ

2017



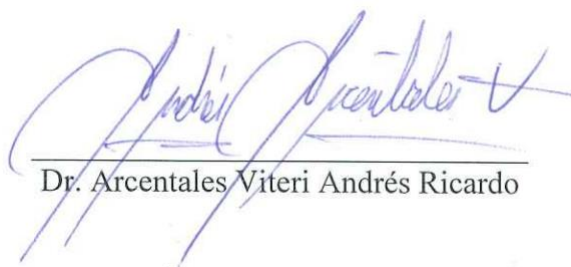
DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y
CONTROL

CERTIFICACIÓN

Certifico que el trabajo de titulación, “*SISTEMA ROBÓTICO MÓVIL AUTÓNOMO COOPERATIVO PARA MAPEO 2D DE OBSTÁCULOS Y NIVELES DE TEMPERATURA EN UN AMBIENTE CONTROLADO*”, realizado por el señor Carlos Andres Moya Erazo, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo que cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a el señor Carlos Andres Moya Erazo, para que lo sustente públicamente.

Sangolquí, Julio 2016

Atentamente.



Dr. Arcentales Viteri Andrés Ricardo



DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y
CONTROL

DECLARACIÓN DE RESPONSABILIDAD

Yo, Carlos Andres Moya Erazo, con cédula de identidad N°131374797-2, declaro que este trabajo de titulación “SISTEMA ROBÓTICO MÓVIL AUTÓNOMO COOPERATIVO PARA MAPEO 2D DE OBSTÁCULOS Y NIVELES DE TEMPERATURA EN UN AMBIENTE CONTROLADO” ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Sangolquí, Julio 2016

Moya Erazo Carlos Andres
CI: 1313747972



DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
**CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y
CONTROL**

AUTORIZACIÓN

Yo, Carlos Andres Moya Erazo, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación “SISTEMA ROBÓTICO AUTÓNOMO COOPERATIVO PARA MAPEO 2D DE OBSTÁCULOS Y NIVELES DE TEMPERATURA EN UN AMBIENTE CONTROLADO” cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Sangolquí, Julio 2016

Moya Erazo Carlos Andres
CI: 1313747972

DEDICATORIA

Abandono la idea epístola o teológica moral tradicional para dedicar un trabajo a creencias o un grupo selecto de personas científicamente abúlicas sin deseos de ofender o de perseguir egocentrismos naturales de la especie. Por el contrario, dedico este escrito a cualquier persona con la capacidad constructiva de utilizarlo con sabiduría y ética de acuerdo a las necesidades del avasallado mundo más no de la codiciosa humanidad.

Moya Erazo Carlos Andres

AGRADECIMIENTOS

Aunque pudiera agradecer epidérmicamente a mi familia en general y sus inherentes reaprehendidos concejos, agradezco infinitamente lo que me enseñaron implícitamente con sus actividades. Después de todo nuestro mayor logro siempre ha sido ser humanos.

El ejemplo más leal de amor, lo tengo de mi madre y la pasión por el conocimiento de mi padre; ambos forman el ejemplo adecuado para ser o no ser de un hijo digno de sus apellidos. Este trabajo es más suyo que mío, es el reflejo de sus tantas preocupaciones y de su incondicional presencia espiritual.

Se merecen un reglón también mis hermanas que no siempre fueron un ejemplo, pero que agradezco el estar en mi vida ya que su presencia de alguna forma creo mi carácter y establecieron este monstruo lleno de ilusiones y sinceros deseos sobre su felicidad absoluta. Taty, hoy tu espalda ya es más ligera, porque sabes lo alto que me enseñaste a volar. Mi orgullo y mayor ejemplo de vida siempre ha sido a tu lado, mi llavero mágico, mi antorcha de luz eterna, esto igualmente es tuyo.

Hay mucha gente que pudiera mencionar en esta publicación, amigos, compañeros, profesores, amores, etc pero todos tienen el común denominador llamado experiencia. Crearon entre todos, una etapa, o un momento en mi vida que llegaron a fundar lo que hoy conocen como Carlos Moya. Les agradezco a los que se metieron en mi corazón y a los que se merecieron un momento, agradezco su tiempo, su confianza y desconfianza, los retos propuestos y sus finales, agradezco a quienes conocen más de mí que yo mismo, te agradezco por enseñarme a dar todo sin esperar nada, te agradezco la sonrisa perpetua y los tantos te quiero de aparición fugaz, te agradezco por dejarme ser un cronopio y te agradezco por permanecer en mi vida aun conociendo mis demonios. Gracias.

Moya Erazo Carlos Andres

ÍNDICE DE CONTENIDO

CARÁTULA

CERTIFICACIÓN	ii
DECLARACIÓN DE RESPONSABILIDAD.....	iii
AUTORIZACIÓN	iv
AGRADECIMIENTOS	v
ÍNDICE DE CONTENIDO.....	vii
ÍNDICE DE TABLAS	xii
ÍNDICE DE FIGURAS.....	xiii
RESUMEN.....	xviii
ABSTRACT.....	xix
CAPÍTULO I.....	1
INTRODUCCIÓN	1
1.1 ANTECEDENTES	1
1.2 JUSTIFICACIÓN E IMPORTANCIA	3
1.3 ALCANCE DEL PROYECTO	5
1.4 OBJETIVOS	6
1.4.1 General	6
1.4.2 Específicos	6
CAPÍTULO II	7
ESTADO DEL ARTE	7
2.1 SISTEMAS MULTI ROBÓTICOS COOPERATIVOS.....	7
2.1.1 Principales áreas de investigación.....	7
2.1.1.1 Robots bioinspirados.....	7
2.1.1.2 Localización y mapeado.....	8
2.1.1.3 Transporte y manipulación de objetos	9

2.1.1.4	Reconfiguración de sistemas robóticos.....	10
2.1.2	Componentes de un sistema multi robótico.....	10
2.1.2.1	Agente.....	11
2.1.2.2	Control en sistemas cooperativos.....	11
2.1.2.3	Comunicación en sistemas cooperativos.....	13
2.2	PROBLEMA DE LOCALIZACIÓN Y MAPEADO SIMULTÁNEO.....	13
2.2.1	Representación de mapas.....	14
2.2.2	Hardware para la localización y mapeado simultaneo.....	15
2.2.2.1	Plataforma móvil.....	15
2.2.2.2	Dispositivos de medición de obstáculo.....	16
2.2.2.3	Estimación de posición por odometría.....	16
2.2.3	Software para la localización y mapeado simultaneo.....	17
2.2.3.1	Algoritmos SLAM.....	18
2.2.3.2	Carga computacional.....	18
2.3	PROBLEMA SLAM EN MULTI ROBÓTICA.....	19
2.3.1	Unión de mapas en sistemas MRSLAM.....	24
2.3.2	Comunicación en sistemas MRSLAM.....	27
2.4	DESCRIPCIÓN DEL PROYECTO.....	28
	CAPÍTULO III.....	30
	DISEÑO E IMPLEMENTACIÓN DE LAS PLATAFORMAS MÓVILES ROBÓTICAS.....	30
3.1	INTRODUCCIÓN.....	30
3.2	CONSIDERACIONES DE DISEÑO.....	30
3.3	PLATAFORMA ROBÓTICA MÓVIL.....	31
3.3.1	Tipo de plataforma móvil.....	31
3.3.2	Consideraciones de diseño para la elección de una plataforma diferencial ...	33
3.3.2.1	Estabilidad mecánica.....	33
3.3.2.2	Maniobrabilidad.....	34
3.3.2.3	Grados de libertad.....	35
3.3.2.4	Odometría.....	35

3.3.2.5	Selección de la plataforma robótica	36
3.3.3	Arquitectura de la plataforma móvil diferencial	36
3.3.4	Cinemática de un robot móvil diferencial	37
3.3.4.1	Geometría del robot diferencial	38
3.3.4.2	Cinemática directa.....	38
3.4	ACTUADORES	43
3.4.1	Motores DC.....	44
3.4.1.1	Driver L298N	45
3.4.2	Micro Servo SG90	45
3.5	SENSORES	46
3.5.1	Sensor ultrasónico HC-SR04	46
3.5.2	Brújula electrónica GY-271	47
3.5.3	Sensor de temperatura DS18B20	48
3.5.4	Encoder tipo optointerruptor FC-03.....	49
3.6	TARJETA PROGRAMABLE DE CONTROL	50
3.6.1	Arduino Mega 2560	50
3.6.2	Arduino Nano.....	51
3.7	SISTEMAS DE COMUNICACIONES	52
3.7.1	Comunicación Wi-Fi.....	54
3.7.1.1	Módulo Wi-Fi Esp8266	55
3.7.1.2	Protocolo de comunicación	56
3.7.1.3	Tipología y topología de la red	57
3.7.2	Comunicación UART.....	58
3.7.3	Comunicación I2C	59
3.8	FUENTE DE ALIMENTACIÓN	59
3.9	DISEÑO DE PLACAS DE CIRCUITO IMPRESO ADICIONALES.....	61
3.10	CONEXIÓN Y MONTAJE DE LA PLATAFORMA ROBÓTICA MÓVIL... ..	62
	CAPÍTULO IV.....	68
	DESARROLLO DE SOFTWARE PARA CONTROL Y VISUALIZACIÓN	
	MRSLAM... ..	68

4.1	INTRODUCCIÓN	68
4.2	DISEÑO DE INTERFAZ GRÁFICA	68
4.2.1	Descripción de funcionamiento.....	69
4.2.1.1	Pantalla de mapas	69
4.2.1.2	Pantalla de históricos.....	72
4.2.2	Clases y métodos desarrollados	73
4.3	DESARROLLO DEL CONTROLADOR COOPERATIVO CENTRALIZADO PARA MRSLAM.....	76
4.3.1	Consideraciones de diseño	77
4.3.2	Diseño del algoritmo de control.....	78
4.3.3	Entradas y salidas del sistema de control.....	80
4.3.4	Diseño de filtros espaciales	82
4.3.5	Clases y métodos Desarrollados	84
4.4	DESARROLLO DEL CONTROL INDEPENDIENTE DE LOS AGENTES ROBÓTICOS	86
4.4.1	Consideraciones de diseño	86
4.4.2	Diseño del algoritmo de control.....	87
4.4.3	Control de recorrido	90
4.4.4	Control de giro	94
4.4.5	Librerías y funciones desarrolladas.....	95
	CAPÍTULO V	98
	PRUEBAS Y RESULTADOS EXPERIMENTALES	98
5.1	INTRODUCCIÓN.....	98
5.2	PRUEBAS Y CALIBRACIONES DE SENSORES	98
5.2.1	Pruebas con sensores ultrasónicos	98
5.2.2	Pruebas con sensores encoders	100
5.2.3	Pruebas con brújulas electrónicas	105
5.3	Pruebas de funcionamiento del sistema MRSLAM.....	107
5.3.1	Evolución del sistema MRSLAM durante las pruebas	109
5.4	RESULTADOS DEL SISTEMA MRSLAM.....	114

CAPÍTULO VI.....	122
CONCLUSIONES Y RECOMENDACIONES	122
6.1 DISEÑO E IMPLEMENTACIÓN DE LAS PLATAFORMAS MÓVILES ROBÓTICAS	123
6.2 DESARROLLO DE SOFTWARE PARA CONTROL Y VISUALIZACIÓN MRSLAM... ..	124
6.3 PRUEBAS Y RESULTADOS EXPERIMENTALES.....	126
6.4 FUTUROS TRABAJOS	129
BIBLIOGRAFIA	132

ÍNDICE DE TABLAS

Tabla 1. Tipos de robots móviles con ruedas (Fabrizzio et al., 2013)	32
Tabla 2. Comparativa entre tecnologías inalámbricas (Björn Jürgens et al., 2008)...	53
Tabla 3. Comunicación interna de un robot.	54
Tabla 4. Protocolos usados según el modelo OSI.	57
Tabla 5. Resumen de consumo energético de un robot.....	59
Tabla 6. Configuración de pines del Arduino Mega 2560.	63
Tabla 7. Configuración de pines del Arduino Nano.	63
Tabla 8. Costos finales de la implementación de las plataformas robóticas	66
Tabla 9. Clases usadas para la interfaz gráfica.	73
Tabla 10. Descripción de métodos para la Interfaz Gráfica.	74
Tabla 11. Entradas y salidas del controlador central del sistema MRSLAM.	81
Tabla 12. Clases usadas para el controlador central.	84
Tabla 13. Descripción de métodos para el controlador central.....	85
Tabla 14. Resultados de las pruebas del error de recorrido.	93
Tabla 15. Corrección lineal de error en recorrido en los 3 robots móviles.....	94
Tabla 16. Descripción de librerías usadas para el controlador independiente de cada agente robótico.	96
Tabla 17. Funciones desarrolladas para el controlador independiente de cada agente robótico.	96
Tabla 18. Comparación de los tiempos de procesamiento de los 3 códigos usados.	100
Tabla 19. Resultados de pruebas con sensores encoders.	101
Tabla 20. Resultados de pruebas con 2 sensores encoders simultáneamente a velocidad máxima.....	102
Tabla 21. Resultados de las pruebas de funcionamiento en el primer escenario.	120
Tabla 22. Resultados de las pruebas de funcionamiento en el segundo escenario. .	120

ÍNDICE DE FIGURAS

Figura 1. Enjambre de iRobots del MIT. (Miner, 2007)	8
Figura 2. Ejemplo de localización y mapeado mediante un robot. (Lemus, Díaz, Gutiérrez, Rodríguez, & Escobar, 2014).....	9
Figura 3. Robots usados por Wang et al., (2000) (Izquierda) y por Rus et al., (1995) (Derecha).....	9
Figura 4. Grupo de S-bots unidos para cruzar rocas. (Miner, 2007)	10
Figura 5. Ejemplos de algunos tipos de plataformas robóticas. (Molina Villa & Rodríguez Vásquez, 2014).....	11
Figura 6. Tipos de arquitecturas de control en multi robótica. (Martins, 2013).....	12
Figura 7. Ejemplo de un mapa metrico. (Alexandre, 2013)	15
Figura 8. Ejemplo de un mapa topografico. (Portugal & Rocha, 2013).....	15
Figura 9. Descomposición de celdas adaptativas. (Carvajal et al., 2013)	19
Figura 10. Proyección del punto inicial y final de una línea con el método propuesto por Kuo et al., (2011)	19
Figura 11. Mapa creado por 3 robots con un algoritmo topológico/métrico. (Chang et al., 2007)	21
Figura 12. Mapa creado por 3 robots por el método de coordinación explícita de Simmons, (2000)	22
Figura 13. Robots móviles utilizados por Howard, (2006) con torres codificadas para su identificación autónoma.	23
Figura 14 Ejemplo de unión de 2 mapas topológicos en (Alexandre, 2013)	24
Figura 15 (a) Unión de mapas de 2 robots antes de la optimización global. (b) Unión de mapas después de la optimización global. (Chang et al., 2007)	25
Figura 16 Unión de mapas en (Liu et al., 2013).	26
Figura 17 Método basado en puntos de referencia usado en (Zhou & Roumeliotis, 2006).	27
Figura 18. Esquema del hardware electrónico del robot.	31
Figura 19. Estabilidad de un robot móvil diferencial.	34
Figura 20. Grados de libertad de un robot diferencial.	35
Figura 21. Plataforma Robótica Diferencial 2WD	36
Figura 22. Cotas generales de la plataforma móvil utilizada en mm.	37
Figura 23. Geometría de un robot diferencial.	38

Figura 24. Movimiento de rotación con una rueda bloqueada.	39
Figura 25. Modelado cinemático en rueda izquierda.	39
Figura 26. Modelado cinemático de rueda derecha.	41
Figura 27. Representación del plano del robot en el plano del entorno.	42
Figura 28. Motor DC con caja reductora.	44
Figura 29. Driver de motores L298N.	45
Figura 30. Micro servo SG90.	46
Figura 31. Sensor ultrasónico HC-SR04	47
Figura 32. Brújula electrónica GY-271.	48
Figura 33. Sensor de temperatura DS18B20.	49
Figura 34. Encoder FC-03.	49
Figura 35. Tarjeta de programación Arduino Mega 2560.	51
Figura 36. Tarjeta Arduino Nano.	52
Figura 37. Esquema de red inalámbrica utilizado.	54
Figura 38. Módulo Wi-Fi Esp8266.	55
Figura 39. Diagrama de redes del sistema de comunicaciones implementado.	58
Figura 40. Batería utilizada para cada robot.	60
Figura 41. Posición de sensores en la placa de circuito impreso (1).	61
Figura 42. Circuito impreso de placa de sensores (1).	61
Figura 43. Posición de dispositivos en placa de circuito impreso (2).	62
Figura 44. Circuito impreso de placa de dispositivos (2).	62
Figura 45. Montaje en la parte inferior de la plataforma robótica móvil.	64
Figura 46. Montaje en la parte superior de la plataforma robótica móvil.	65
Figura 47. Implementación de una plataforma robótica móvil.	66
Figura 48. Distribución de componentes en ventana de mapas.	69
Figura 49. Diagrama de flujo del funcionamiento de ventana de mapas.	71
Figura 50. Distribución de componentes en ventana de históricos.	72
Figura 51. Pantalla de Mapas Desarrollada para este trabajo.	76
Figura 52. Pantalla de datos históricos desarrollada para este trabajo.	76
Figura 53. Posibles interacciones entre el agente robótico y el entorno.	78
Figura 54. Posibles interacciones entre el agente robótico y el mapa.	78
Figura 55. Diagrama de flujo de programa de control centralizado.	79

Figura 56. Diagrama de bloques de controlador central del sistema MRSLAM.	81
Figura 57. Condiciones mínimas para pertenecer a un obstáculo o un espacio vacío.....	83
Figura 58. Condición de baja probabilidad para un punto desconocido.	83
Figura 59. Ejemplo de un punto desconocido en el mapa de niveles de temperaturas.	84
Figura 60. Diagrama de flujo de programa de control parcial de cada robot móvil.....	89
Figura 61. Trama para el envío de información.	90
Figura 62. Pruebas de recorrido lineal para controlador de recorrido.	92
Figura 63. Gráfica del error acumulativo medido en distintas posiciones.	93
Figura 64. Movimiento giratorio de un robot diferencial.	94
Figura 65. Resultado de 3 versiones de programas midiendo una distancia de 20 cm con 500 muestras.	99
Figura 66. Resultados sobre el error obtenidos con 3 programas a distintas distancias.....	99
Figura 67. Resultados de pruebas con sensores encoders.	101
Figura 68. Resultados de pruebas con 2 sensores encoders simultáneamente	102
Figura 69. Pruebas realizadas con recorridos de 30 cm consecutivos hasta los 420 cm.....	103
Figura 70. Pruebas realizadas con recorridos de 60 cm consecutivos hasta los 420 cm.....	103
Figura 71. Pruebas realizadas con recorridos de 120 cm consecutivos hasta los 360 cm.....	104
Figura 72. Comparación entre robots del error promedio de recorrido cada 30 cm.	104
Figura 73. Comparación entre robots del error promedio de recorrido cada 60 cm.	105
Figura 74. Comparación entre robots del error promedio de recorrido cada 120 cm.	105
Figura 75. a) Recorrido real del agente robótico a 0 grados de dirección. b) Recorrido simulado por la brújula electrónica.	106
Figura 76. a) Recorrido real del agente robótico a 90 grados de dirección. b) Recorrido simulado por la brújula electrónica.	106
Figura 77. a) Recorrido real del agente robótico a 180 grados de dirección. b) Recorrido simulado por la brújula electrónica.	107
Figura 78. a) Recorrido real del agente robótico a 270 grados de dirección. b) Recorrido simulado por la brújula electrónica.	107
Figura 79. a) Primer escenario de obstáculos usado para pruebas de mapeado. b) Segundo escenario de obstáculos usado para pruebas de mapeado.	108

Figura 80. a) Mapa de obstáculos creado por 3 agentes robóticos antes de la calibración de dirección. b) Mapa de obstáculos creado por 3 agentes robóticos después de la calibración de dirección.	109
Figura 81. Mapa de obstáculos creado con 3 agentes robóticos con recorridos superiores a los 55mt.	110
Figura 82. Mapa de obstáculos creado con 3 agentes robóticos con excesivo ruido magnético ambiental.	111
Figura 83. Progreso del panel del mapa de obstáculos en la interfaz gráfica del sistema MRSLAM.	111
Figura 84. Variación de las posiciones de los 3 agentes robóticos en el espacio a explorar. .	112
Figura 85. Mapa de obstáculos antes y después del filtro de media.	113
Figura 86. Mapa de temperaturas antes y después del filtro de media.	113
Figura 87. Clasificación de las pruebas de funcionamiento realizadas.	114
Figura 88. Resultados de prueba en el primer escenario con el robot 1 únicamente. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	115
Figura 89. Resultados de prueba en el primer escenario con el robot 2 únicamente. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	115
Figura 90. Resultados de prueba en el primer escenario con el robot 3 únicamente. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	115
Figura 91. Resultados de prueba en el primer escenario con 2 agentes robóticos. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	116
Figura 92. Resultados de prueba en el primer escenario con 3 agentes robóticos. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	116
Figura 93. Resultados de prueba en el segundo escenario con el robot 1 únicamente. a) Registro obtenido en la interfaz al finalizar el mapeado. b) Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.	116

- Figura 94. Resultados de prueba en el segundo escenario con el robot 2 únicamente.
a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real..... 117
- Figura 95. Resultados de prueba en el segundo escenario con el robot 3 únicamente.
a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real..... 117
- Figura 96. Resultados de prueba en el segundo escenario con 2 agentes robóticos.
a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real..... 117
- Figura 97. Resultados de prueba en el segundo escenario con 3 agentes robóticos.
a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real..... 118

RESUMEN

El proyecto desarrollado relaciona las habilidades de varios robot móviles simples para el cumplimiento de una sola tarea, en este caso el objetivo es mapear una área determinada para poder localizar obstáculos y cambios de temperatura, para ello se lleva a cabo la implementación de un sistema multi robótico móvil con el propósito de demostrar la eficacia del método sobre sistemas con un solo robot. Además uno de los retos del trabajo presentado es la creación de algoritmos para resolver la problemática denominada Simultaneous Localization and Mapping (SLAM) en sistemas multi robóticos. El sistema cuenta con un controlador central el cual se encarga de guiar y distribuir a los robots mediante comunicación inalámbrica, este controlador central cuenta con una interfaz sencilla que simula el entorno de los robots en tiempo real para la visualización de los obstáculos y de los cambios de temperatura con una vista superior en 2 dimensiones. Los robots fueron desarrollados bajo una plataforma de bajo coste con un sistema de estimación de posición por odometría implementado en cada robot, usando sensores para detectar velocidad, dirección y distancias de obstáculos. La propuesta presentada en este documento intenta combinar y mejorar varias estrategias establecidas en diferentes investigaciones para finalmente mostrar las capacidades del método desarrollado mediante la realización de pruebas en ambientes controlados que simulen un espacio físico desconocido.

PALABRAS CLAVE:

- MULTI ROBÓTICA
- MAPEO
- LOCALIZACIÓN
- ODOMETRÍA
- AUTÓNOMO

ABSTRACT

The developed project combines the skills of several simple to carry out a single task mobile robot, in this case, the target is to map a particular area to locate obstacles and temperature changes, for it carried out the implementation of a multi-robot system mobile to demonstrate the effectiveness of the method versus systems with a single robot. In addition, one of the challenges in this work presented is to create algorithms to solve the problem called SLAM (simultaneous localization and mapping) in multi-robot systems.

The system has a central controller which is responsible for guiding and distributing robots way wireless communication, the central controller has a simple interface that simulates the environment of the robots in real time for viewing obstacles and changes temperature with a top view in two dimensions. The robots were developed under a low-cost platform with position estimation system implemented by each robot odometry, using sensors to detect speed, direction and distance of obstacles.

The proposal presented in this paper attempts to combine and improve various strategies established in various investigations to finally show the capabilities of the method developed by testing in controlled environments that mimic an unknown physical space.

KEYWORDS:

- MULTI-ROBOT
- MAPPING
- LOCALIZATION
- ODOMETRY
- AUTONOMOUS

CAPÍTULO I

INTRODUCCIÓN

1.1 Antecedentes

Casi todo el trabajo en robótica móvil cooperativa comenzó después de la introducción del nuevo paradigma de la robótica de control basado en el comportamiento (Brooks, 1986). Este modelo ha tenido una fuerte influencia en la investigación sobre cooperación entre robots, donde se han identificado siete tópicos primarios de estudio con multi robots: inspiraciones biológicas, comunicación, arquitecturas, navegación, transporte y manipulación de objetos, coordinación de movimientos, y robots reconfigurables. (Mohan & Ponnambalam, 2009)

La robótica cooperativa para objetivos de navegación se ha destacado por el significativo avance en comparación de eficacia respecto a un robot singular, ya que al distribuir la tarea a varios robots el nivel de complejidad de cada uno es menor y el área de trabajo también es dividida entre ellos. Sin embargo sigue siendo predominante el uso de un robot especializado por razones de logística y comunicaciones (Cano & Palaquibay, 2014). En este mismo ámbito, se ha tenido significativos avances en exploración ambiental, vigilancia, búsqueda y rescate. Uno de los principales enfoques es el mapeado de ambientes 2D y 3D basado en grupos multi robot, que generalmente usan un algoritmo existente desarrollado para un robot sencillo siendo extendido a múltiples robots. (Mohan & Ponnambalam, 2009)

Estos avances tecnológicos fueron posibles gracias al método SLAM (por sus siglas en inglés localización y mapeado simultáneo) desarrollado originalmente por Hugh Durrant-Whyte y John J. Leonard para la construcción de mapas en un ambiente desconocido mientras el robot navega sobre él (Leonard & Durrant-Whyte, 1991). Este algoritmo ha ido evolucionando y se han creado varias versiones del mismo, entre las que se destacan SLAM activo para múltiples robots y FastSLAM multi robot on line. (Pham & Juang, 2011; Tovar et al., 2006)

Existen dos grandes campos de investigación sobre SLAM: la localización, y el desplazamiento en espacios desconocidos. Para la solución del problema de localización, se plantean técnicas basadas en la correspondencia de coordenadas y en

la posición inicial (Zhou & Roumeliotis, 2006), en sistemas con métodos de trilaterización (Spears et al., 2006), y en sistemas odométricos (Marjovi, Nunes, Sousa, Faria, & Marques, 2010; Pfingsthorn, Slamet, & Visser, 2008). Por otra parte, se investigó la propiedad ortogonal de ambientes interiores para aumentar la precisión de algoritmos de localización y técnicas basadas en puntos de encuentro para resolver el problema de comunicación limitada. (Kuo, Chang, Chen, & Huang, 2011; Pham & Juang, 2011)

En el desplazamiento de áreas desconocidas la principal meta es disminuir el tiempo de mapeo, para ello han sido desarrollados varios métodos de dispersión (Burgard, Moors, Stachniss, & Schneider, 2005). Entre ellos existen algoritmos basados en el principio de repulsión y atracción electromagnética (Chen, Yuan, Zhang, & Fang, 2014), y algoritmos que ejecutan una etapa de dispersión y otra de detección de fronteras alternadamente (Mondada et al., 2009). Otro estudio, usa métodos basados en comportamientos básicos como búsqueda de fronteras, evitar obstáculos y evitar robots (Lau, 2003). Estos métodos de identificación de frontera ofrecen una buena relación precio-rendimiento, siendo particularmente adecuados para la implementación en sistemas reales encajados en lugares cerrados. (Kalde, Simonin, & Charpillet, 2015)

Al usar varios robots surge una necesidad adicional al método SLAM, donde el principal problema es centralizar la información obtenida por cada robot en un único controlador para su interpretación. En el caso de la navegación existen métodos de fusión de mapas topológicos (Chang, Lee, Hu, & Lu, 2007), y métodos que realizan este cometido en el recorrido con *Extended Kalman Filter* (EKF) (Yuan, 2009). De forma general, son métodos que se pueden complementar entre ellos para crear un algoritmo más robusto con estrategias adaptativas para equilibrar la necesidad de exploración y localización. Estas estrategias combinan los conceptos de SLAM, planificación activa de ruta y la cooperación entre múltiples robots. (Pham & Juang, 2013)

En conclusión, en los últimos años, la investigación sobre SLAM multi robot (MRSLAM) ha atraído especial atención, convirtiéndose en un tema de fuerte impacto en el campo de la robótica (Pfingsthorn et al., 2008). Asimismo, en la actualidad, los enfoques existentes para MRSLAM se centran principalmente en la expansión de los

algoritmos de SLAM de un robot para el caso multi robot (Chen et al., 2014). Sin embargo, en robótica cooperativa, se presentan nuevos retos, tales como la coordinación de los robots, la integración de la diferente información recogida por los robots en un mapa coherente y lidiar con la comunicación limitada. (Alexandre, 2013)

De igual forma debe considerarse que SLAM todavía tiene aspectos de investigación en el campo que no han sido resueltos (Riisgaard & Blas, 2004), entre algunas falencias esta la falta de sistemas reales, la percepción que aún sigue siendo muy propensa a errores, la información útil sigue siendo un reto poder filtrarla y la obstrucción de objetos dificulta el trabajo de los sensores. (Ramirez Benavides, 2012)

La tendencia actual en navegación investiga sistemas distribuidos para facilitar el diseño de trayectorias de robots, equilibrando calidad del mapa, exactitud de la posición y el tiempo de exploración general. Es decir, la optimización global para dispersar a los robots en el ambiente sin tener un conocimiento a priori del entorno. (Pham & Juang, 2013)

Para las próximas investigaciones, el mayor desafío será desarrollar algoritmos robustos, ligeros y fáciles de integrar en estos sistemas robóticos embebidos (Kuo et al., 2011). Por ahora, la extensión de la técnica SLAM a múltiples robots, con el fin de realizar tareas de cooperación en ambientes desconocidos y/o entornos dinámicos sigue siendo un gran desafío. (Alexandre, 2013)

1.2 Justificación e importancia

La idea fundamental detrás de la robótica cooperativa sugiere la distribución de la tarea principal en tareas menores para cada robot individual, permitiendo la interacción entre ellos para encontrar soluciones a diversos problemas (Alexandre, 2013). Si bien es cierto, que en las últimas décadas, este tipo de sistemas han sido explotados científicamente, con especial atención a la exploración autónoma, muy pocos de estos sistemas han sido llevados a entornos reales.

La navegación en entornos desconocidos con robots móviles es un desafío de importante impacto en un gran número de ramas aplicables en situaciones reales tales como: tareas de búsqueda y rescate; exploración planetaria, submarina y minera; así como en vigilancia y reconocimiento militar (Pham & Juang, 2013). Por ejemplo, en

la búsqueda y rescate, en la fase de reconocimiento del área afectada, se puede enviar una flotilla de robots para hallar zonas de potencial peligro. Esto se puede realizar, midiendo variables como la temperatura, la densidad del humo o la concentración de sustancias explosivas y/o tóxicas. Y en definitiva poder señalar en el mapa donde se encuentra la mayor concentración de fuego, he incluso la presencia de víctimas.

Otra área aplicable es el reconocimiento militar, donde se pueden crear grupos robóticos autónomos de intervención, que generen mapas de zonas ocupadas por enemigos de forma segura, o pueden ser enviados a localizar explosivos, drogas e inclusive localizar rehenes de forma precisa y eficiente para planificar estrategias seguras de inmovilización o rescate.

Actualmente, los sistemas de múltiples robots tienen varias ventajas sobre sistemas robóticos simples, como finalización de tareas más rápido, eficiente localización y mayor tolerancia a fallos (Cao, Fukunaga, Kahng, & Meng, 1997). Por ejemplo, un autómatas puede repetir una ruta ya trazada cuando no sabe dónde están los otros robots, llevando a una superposición en la exploración (Pham & Juang, 2013). Por lo tanto, estos algoritmos de exploración solucionan un problema de optimización equilibrada maximizando el número de tareas respecto a los recursos disponibles. Esta afirmación ha comenzado a generar estudios sobre la efectividad de grupos multi robots comparada con versiones de robots sencillos (Rao, 2000), pero aún queda mucho por ser determinado por la variedad de enfoques disponibles dentro de localización, mapeo y exploración. (López Pérez & Mata Herrera, 2001)

Uno de los estudios más actuales sobre efectividad, demostró que *multi robot system* (MRS) pueden ser muy rentables si se comparan con la construcción de un solo robot costoso especializado (Alexandre, 2013). Además se observó que los errores de localización y mapeo pueden ser reducidos a través de la integración de la información medida y navegación de múltiples robots (Cai, Tang, & Zhao, 2012), que pueden cubrir todo el entorno más rápido a través de la comunicación y coordinación en paralelo. (Chen et al., 2014)

Aun con todas las ventajas demostradas, no se ha realizado un estudio actual sobre la identificación y la cuantificación de las características fundamentales de los MRS (López Pérez & Mata Herrera, 2001) y la eficiencia de coordinación a varios niveles, por ejemplo, consumo de energía y superposición de trayectorias (Kalde et al.,

2015), sobre todo en sistemas reales multi robots, bajo las restricciones de tiempo real y las características dinámicas del ambiente.

El presente proyecto plantea las bases fundamentales para el desarrollo de esta tecnología en el país, lo que servirá como una plataforma abierta, para que nuevos investigadores puedan entender las complejidades involucradas en la implementación de sistemas MRSLAM. En consecuencia se espera un avance progresivo y más expedito en posibles escalones más avanzados como los ejemplos planteados.

1.3 Alcance del proyecto

En el presente proyecto, se plantea diseñar un sistema multi robot móvil autónomo, para mapeo 2D de obstáculos y niveles de temperatura invariantes en el tiempo dentro de un ambiente controlado, mediante la implementación de un modelo adecuado para la solución al problema llamado MRSLAM.

Para cumplir este propósito se propone usar tres plataformas robóticas móviles de configuración diferencial (Fabrizio, Patiño, María, Cazar, & Cuenca, 2013). Estas plataformas tendrán iguales capacidades y habilidades. Por lo que serán equipadas con un sistema de control individual que permitirá mediante sensores de odometría y de distancia encontrar su ubicación en un plano de dos dimensiones.

La propuesta presentada requerirá que el MRS parta desde un mismo punto y hacia una misma dirección; este punto, servirá como primera referencia en una matriz de posición para la localización de cada robot. Asimismo se pretende integrar a cada plataforma un sensor electrónico de temperatura que permitirá el posterior registro y visualización de cambios de calor en los distintos puntos del plano navegado.

Toda la información obtenida, utilizará un sistema de comunicación inalámbrica mediante módulos Wi-Fi para el control de los movimientos, y la interpretación de las mediciones obtenidas con un controlador central. Estas mediciones se utilizarán para desarrollar una interfaz gráfica que permita visualizar obstáculos y niveles de temperatura a través de código de colores, cuando el controlador central haya interpretado los datos procesados.

El ambiente cerrado elegido será una habitación de 18m², acoplada para probar el comportamiento de los robots con diferentes tipos de obstáculos. Aparte, se utilizará

distintas fuentes de calor o frío dispuestos en diferentes lugares para permitir distinguir los niveles de temperatura en dicha ubicación.

Finalmente, se evaluará el comportamiento del sistema en 2 ambientes cerrados distintos con condiciones de comunicación óptimas. Los principales valores a tomar en cuenta son el tiempo real que se demora en crear un mapa de una habitación de 18m², el error en la posición del robot y el error en la arquitectura del mapa creado. Para posteriormente poder comparar los resultados obtenidos con un sistema de mapeado de un solo robot.

1.4 Objetivos

1.4.1 General

Diseñar un sistema multi robot móvil autónomo para el mapeado en 2 dimensiones de obstáculos y niveles de temperatura constantes en un ambiente controlado.

1.4.2 Específicos

- Diseñar el sistema de control centralizado adecuado para el desarrollo de las actividades autónomas de los robots.
- Diseñar el sistema de comunicación inalámbrica entre los robots exploradores y el centro de control.
- Desarrollar un algoritmo para la localización y mapeado simultaneo de los robots basados en teorías sobre el problema SLAM.
- Implementar los algoritmos de control en robots físicos para obtener los resultados reales de los sistemas diseñados.
- Cuantificar el rendimiento de la robótica cooperativa vs un solo robot para el cumplimiento de esta tarea específicamente.
- Diseñar una interfaz que permita visualizar la posición de los robots y el mapa en dos dimensiones, tanto de los obstáculos y límites de la habitación, como de los niveles de temperatura.

CAPÍTULO II

ESTADO DEL ARTE

2.1 Sistemas multi robóticos cooperativos

A partir de los años 80 surge la necesidad de utilizar más de un sistema robótico para cumplir ciertas tareas de manera más eficiente en relación a los sistemas de un solo robot. Desde entonces, con el fin de enfrentar entornos dinámicos y de multitareas se han presentado investigaciones de sistemas compuestos por más de un robot caracterizados por utilizar las habilidades particulares de cada robot para cumplir una tarea específica. Por tal razón, los sistemas de múltiples robots cooperativos son una alternativa que ha demostrado ser robusta, fiable y adaptable para tareas complejas como la exploración y navegación, rescate en zonas de difícil acceso, sistemas de vigilancia, infiltración militar, etc. (Jiménez, Vallejo, & Ochoa, 2009)

2.1.1 Principales áreas de investigación

Existen varias áreas para sistemas multi robóticos que han tenido un alto impacto en la actualidad, siendo los temas de mayor interés, la planeación de movimientos, localización, mapeo, exploración, reconfiguración de sistemas robóticos, coordinación con varios tipos de arquitectura y la manipulación de objetos (López Pérez & Mata Herrera, 2001; Mohan & Ponnambalam, 2009). Varios de estos temas tienen dos enfoques macro que estudian el comportamiento multi robótico, un enfoque bio inspirado y uno convencional basado en las teorías mecánicas y electrónicas clásicas sobre el movimiento.

2.1.1.1 Robots bioinspirados

Actualmente existen muchos trabajos en robótica cooperativa basados en el comportamiento de la naturaleza, sus ambientes y sus hábitos que replican la conducta de enjambres, manadas, colmenas, etc. Debido a que tantos años de evolución han perfeccionado su comportamiento en tareas relacionadas al entorno (Brooks, 1986). Mediante el estudio de dichos comportamientos como en abejas, hormigas y aves específicamente, se han generado reglas simples de control locales (Mohan &

Ponnambalam, 2009). Más aún existen técnicas de control para sistemas multi robóticos basadas en organismos más complejos como las jaurías de lobos (Parker, 2000), e incluso humanas (Marsella et al., 2001). Su desarrollo más notable ha sido en temas relacionados con la comunicación, navegación y exploración. (Lepora, Verschure, & Prescott, 2012)

A manera de ejemplo en la **Figura 1** se observa un enjambre de robots desarrollados por el MIT llamados iRobots realizando tareas específicas como la reagrupación y dispersión basados en el comportamiento de las hormigas.



Figura 1. Enjambre de iRobots del MIT. (Miner, 2007)

2.1.1.2 Localización y mapeado

La localización y mapeado tradicionalmente se realizaba con un solo agente que se movía con distintos algoritmos que han evolucionado para obtener una mejor resolución en mapas y mayor exactitud en la ubicación (ver **Figura 2**). Estos conceptos se trasladaron a varios robots para formar un sistema coordinado de mapeado más eficiente (López Pérez & Mata Herrera, 2001; Mohan & Ponnambalam, 2009). Básicamente el reto en sistemas multi robot para localización y mapeado, son la coordinación de movimientos, la comunicación y la unión de mapas. En los siguientes subtemas se trata más a fondo el desarrollo de esta tarea que es el tema principal en este trabajo de investigación.

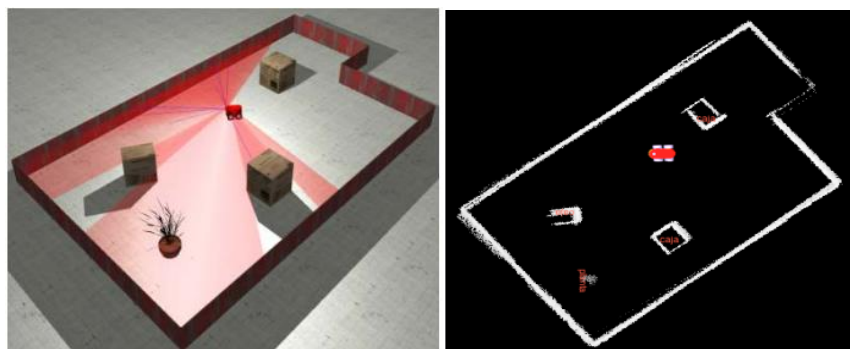


Figura 2. Ejemplo de localización y mapeado mediante un robot. (Lemus, Díaz, Gutiérrez, Rodríguez, & Escobar, 2014)

2.1.1.3 Transporte y manipulación de objetos

Las primeras ideas en multi robótica se generaron para resolver este tema, debido a las limitaciones en los medios móviles de esa época, todavía mucho antes del nacimiento de la robótica se descubrió que era más eficiente dividir la carga en varios agentes. Por ejemplo con caballos arando la tierra o como medios de transporte. (López Pérez & Mata Herrera, 2001; Mohan & Ponnambalam, 2009)

Las principales actividades en grupos multi robóticos son cargar, empujar y manipular un objeto (ver **Figura 3**), estas tareas requieren de una mayor coordinación entre los agentes debido a que actúan con un mismo elemento sobre todo si su objetivo es cargar o manipular un objeto (Wang, Kimura, Takahashi, & Nakano, 2000). Por otro lado, empujar es la tarea de mayor investigación, sobre todo con enjambres de robots los cuales requieren menor grado de coordinación. (Rus, Donald, & Jennings, 1995)



Figura 3. Robots usados por Wang et al., (2000) (Izquierda) y por Rus et al., (1995) (Derecha)

2.1.1.4 Reconfiguración de sistemas robóticos

La reconfiguración de sistemas robóticos, quizás es el área más reciente que se ha abierto dentro de multi robótica cooperativa, donde el principal objetivo es lograr una mayor versatilidad, robustez y adaptabilidad en diferentes medios. También es llamada robótica celular por el parecido biológico de esta rama a la habilidad de las células para formar tejidos (Mohan & Ponnambalam, 2009). Generalmente estos robots son módulos idénticos con algún mecanismo físico que permita la interconexión de los agentes de tal manera que tome una nueva forma útil para una tarea específica como en la **Figura 4**. (López Pérez & Mata Herrera, 2001)



Figura 4. Grupo de S-bots unidos para cruzar rocas. (Miner, 2007)

La reconfiguración de robots es un tema relativamente nuevo, por lo cual los trabajos encontrados sobre el tema suelen proporcionar soluciones muy específicas, pero con un gran potencial en el futuro. (Gross, Bonani, Mondada, & Dorigo, 2006; Murata et al., 2002; Tuci et al., 2006)

2.1.2 Componentes de un sistema multi robótico

Todo sistema multi robótico cooperativo está compuesto por varias partes fundamentales que hacen posible el cumplimiento de una tarea. Los principales componentes que forman un sistema multi robótico cooperativo son el agente o robot, sistema de comunicación y sistema de control.

2.1.2.1 Agente

El agente o robot es el responsable de efectuar la tarea encomendada por el controlador, por lo tanto su arquitectura, forma y capacidades dependerán de la tarea para la cual haya sido diseñado (Guerrero, Rodríguez, & Roldán, 2014) . La gran mayoría de investigaciones en cooperación robótica está guiada hacia robots móviles, aunque no es de menor importancia la cooperación entre robots estáticos, que ha sido un tema ampliamente estudiado y aplicado principalmente en áreas relacionadas al ensamblaje automotriz (Guzmán & Peña, 2013). Mientras que la robótica móvil sigue siendo un tema de mayor dificultad por la incertidumbre generada en un entorno cambiante, donde el principal reto es poder transportarse a sí mismo hacia una nueva posición de manera eficiente. Dando lugar al nacimiento de diferentes tipos de plataformas adaptadas para cada situación que enfrenta un robot.



Figura 5. Ejemplos de algunos tipos de plataformas robóticas. (Molina Villa & Rodríguez Vásquez, 2014)

Los principales tipos de robots móviles pueden ser aéreos, terrestres y acuáticos; dentro de los robots terrestres, existen plataformas con distintos tipos de tracción como extremidades, con orugas o con ruedas de distintas configuraciones que facilitan su traslado en distintas superficies (ver **Figura 5**). (Cano & Palaquibay, 2014) Actualmente las plataformas robóticas móviles existentes son adaptables prácticamente para cualquier situación necesaria, mejorando la confiabilidad y por ende su uso se vuelve más popular entre los investigadores.

2.1.2.2 Control en sistemas cooperativos

El control en sistemas robóticos cooperativos se distingue por la habilidad de manejar el comportamiento de varios robots al mismo tiempo. Dando lugar a que

emerjan nuevos temas de investigación como control selectivo (Alami, Fleury, Herrb, Ingrand, & Robert, 1998), niveles jerárquicos, capacidad de delegación (Khan & De Silva, 2008), sistemas heterogéneos y homogéneos multi robóticos (MacKenzie, Arkin, & Cameron, 1997), la capacidad de solucionar conflictos, entre otros. Estos subtemas, pertenecen a dos grandes grupos de control definidos por Iocchi et al. (2001), donde puntualiza el control centralizado y distribuido. (Iocchi, Nardi, & Salerno, 2001)

El control centralizado se caracteriza por tener un robot madre o nodriza que se encarga de tomar las decisiones dentro del equipo robótico para el cumplimiento de la tarea ordenada (ver **Figura 6(a)**). En este tipo de control los miembros del equipo multi robótico deben enviar información constante y actualizada al controlador central, pero en cualquier caso deben estar disponibles para recibir órdenes cuando lo necesiten desde el controlador central.

Por otra parte, el control distribuido o descentralizado se define por la autonomía implementada a cada robot para tomar decisiones en bien del objetivo común del sistema multi robótico. En este tipo de control hay que tomar en cuenta la comunicación, si tienen la capacidad de intercambiar información entre robots para planear sus acciones se denomina un control distribuido colectivo (ver **Figura 6(b)**). Igualmente si la información que obtienen es solamente proporcionada por sus propios sensores y no toman en cuenta el estado del resto de robots se denomina control distribuido desacoplado. (Cano & Palaquibay, 2014; Mohan & Ponnambalam, 2009)

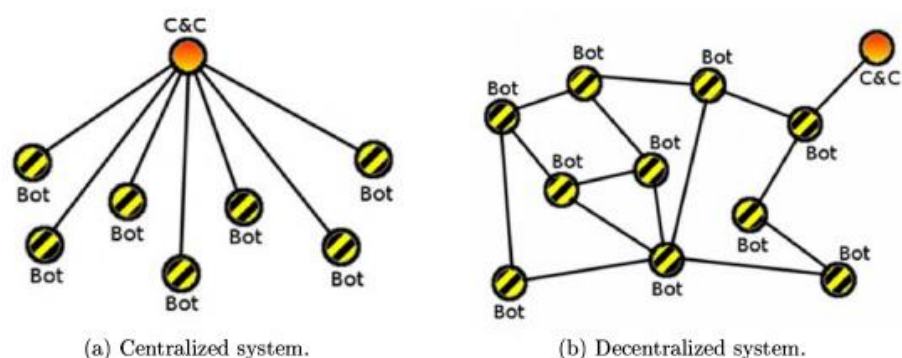


Figura 6. Tipos de arquitecturas de control en multi robótica. (Martins, 2013)

2.1.2.3 Comunicación en sistemas cooperativos

La comunicación ha sido una de las principales líneas de investigación de multi robótica, debido a la necesidad de verificar información o planear una nueva acción que afecte el cumplimiento de la tarea encomendada al grupo robótico por medio del intercambio constante de datos.

Existen dos tipos de comunicación entre robots en sistemas cooperativos, una comunicación implícita o indirecta que se ocasiona cuando las acciones de un agente son modificadas por efecto del estado o actividad de otro robot; y una comunicación explícita o directa que se produce al trasladar información entre robots mediante una acción específica para ese fin (Mohan & Ponnambalam, 2009). A pesar de lo que se podría pensar, varios estudios han demostrado que la comunicación explícita no siempre es una habilidad muy útil, y así mismo que canales de pequeña cantidad de información pueden mejorar significativamente un sistema multi robótico cooperativo. (Burgard et al., 2005)

Además, se ha mejorado notablemente la tolerancia a fallas y la confiabilidad en comunicaciones entre robots con algoritmos más robustos que pueden reconectarse e incluso reconfigurar su red de comunicación (Verret & Suffield, 2005). A pesar de todo el avance generado, aun no existe un medio de comunicación estándar totalmente confiable que funcione en cualquier ambiente.

2.2 Problema de localización y mapeado simultáneo

La problemática de localización y mapeo simultáneo (SLAM por sus siglas en inglés *Simultaneous Localization and Mapping*), es considerado un problema complejo que comienza a ser investigado al inicio de los años noventa. Se define como el problema de construir mapas en ambientes desconocidos con un robot móvil mientras navega usando su propia posición en el mapa que se está creando (Leonard & Durrant-Whyte, 1991). Su uso está presente indirectamente en todo vehículo autónomo no tripulado destinado a la exploración y navegación de espacios abiertos, cerrados, espaciales o submarinos, por lo que su estudio ha captado el interés de muchos investigadores.

Los principales retos que se enfrentan en la localización y mapeado simultáneo son la asociación de datos, estimación de estado y la continua actualización de datos de

estado e históricos (Bailey & Durrant-Whyte, 2006; Wolf & Sukhatme, 2004). Para lo cual se ha creado varias soluciones en distintos trabajos (Kim, Sakthivel, & Chung, 2008; Lemus et al., 2014; Stachniss & Burgard, 2005) y se ha demostrado que algunos de estos algoritmos funcionan tanto en movimiento 2D como en 3D. (MohammadReza, Meisam, & Mohammadmehran, 2013)

2.2.1 Representación de mapas

Para navegar en cualquier lugar mediante un robot móvil, es necesario conocer el entorno que enfrenta de tal forma que pueda volver al punto de partida o que pueda ser encontrado una vez haya terminado su misión. Es decir que necesitan de alguna manera representar su trayectoria y ubicación para poder ser interpretado por una persona en un espacio físico.

Los mapas son la forma gráfica de representar información de un espacio físico real en una escala proporcional al territorio de interés, esto es de gran utilidad dentro de la robótica móvil, principalmente para localizar objetos, planificar rutas, programar actividades y para evadir obstáculos. (Ramirez Benavides, 2012)

Los principales temas tratados en las investigaciones sobre mapeado son la eficiencia de los tipos de mapas empleados, la incertidumbre existente en la creación de mapas y las posibles modificaciones que puede sufrir un mapa por cambios normales del entorno. (Bosse et al., 2003; Pfingsthorn et al., 2008; Wolf & Sukhatme, 2004)

Dedeoglu *et al.*, define dos tipos generales de representaciones discretas del entorno, un modelo topológico basado en la representación de hitos o puntos de interés que sirven como referencia en la localización de un robot como el mostrado en la **Figura 8**; y un modelo métrico el cual considera a la representación de cada punto con un nivel de ocupación, creando una rejilla o malla de estados similar al mostrado en la **Figura 7**. (Dedeoglu & Sukhatme, 2000)

En ambos casos el objetivo general es obtener una representación útil con el menor coste computacional. Generalmente la calidad de un mapa se mide por la exactitud y resolución del mismo. Para esto hay que tomar en cuenta que la precisión de la información sensorial del robot está directamente relacionada con la precisión en el mapa. (Bosse et al., 2003)

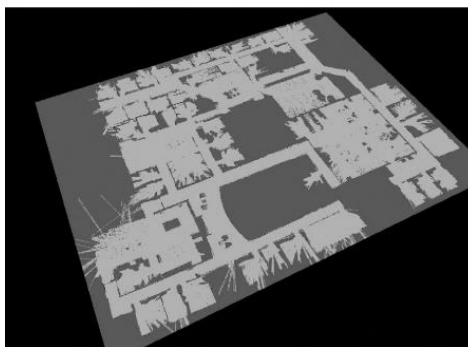


Figura 7. Ejemplo de un mapa métrico. (Alexandre, 2013)

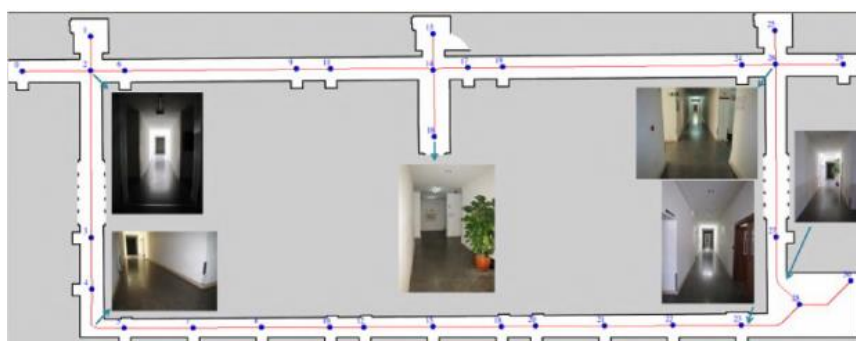


Figura 8. Ejemplo de un mapa topográfico. (Portugal & Rocha, 2013)

2.2.2 Hardware para la localización y mapeado simultaneo

El hardware en sistemas basados en el problema SLAM tiene un gran impacto en el resultado final, porque de él depende la precisión de la información. Es decir, que mientras más exacto el modelo de los elementos del hardware mejores resultados se obtendrá del sistema. El hardware debe constar de al menos dos partes principales una plataforma móvil controlable y dispositivos de sensado del ambiente.

2.2.2.1 Plataforma móvil

El robot es la plataforma en la cual se va a montar los controladores, sensores y actuadores, por lo tanto es el medio por el que se percibe los cambios en el ambiente y el medio que permite actuar con respecto a estos cambios (Apolo & Velazco, 2013). Por otro lado para el caso particular de SLAM, necesariamente debe tener la habilidad de movilizarse en un espacio cualquiera, por ende los parámetros más importantes para elegir la plataforma son la simplicidad del modelado, y la relación entre precio vs precisión odométrica que pueda ofrecer. (Kuo et al., 2011)

2.2.2.2 Dispositivos de medición de obstáculo

Los dispositivos de medición de obstáculos se encargan de reconstruir el ambiente en donde se encuentran con el objetivo de crear un mapa. Estas unidades tienen un rango óptimo de medición dependiendo del tipo de sensor. En la actualidad se usan principalmente tres tipos de adquisición: escaneado laser, visión artificial y radares ultrasónicos. Dependiendo del tipo de ambiente cada uno tiene sus ventajas y desventajas. (Riisgaard & Blas, 2004)

Los escáneres laser son muy precisos pero altamente costosos y tienen problemas al sensar sobre medios cristalinos y bajo el agua (Riisgaard & Blas, 2004). Aparte, la visión artificial basada en el procesamiento de imágenes adquirida por una cámara fotográfica ha sido el medio que ha tenido mayor crecimiento por popularidad entre investigadores, debido a la caída de precios de estos sensores fotosensibles y el mejoramiento en calidad. La principal desventaja es a nivel de software ya que requiere alto consumo computacional para su procesamiento, en cualquier caso se ve altamente afectado por la cantidad de luz en el sector explorado por lo que se complica aún más la adquisición de información (Se, Lowe, & Little, 2002). Finalmente, los radares ultrasónicos son una alternativa de bajo costo, bajo consumo eléctrico, casi invariante en cualquier ambiente o superficie, pero susceptible a falsas lecturas, de baja resolución y precisión con respecto a los láseres. Aunque en la actualidad estos sensores han sido mejorados aun no igualan las características de un escáner laser. (Leonard & Durrant-Whyte, 1991; Riisgaard & Blas, 2004)

2.2.2.3 Estimación de posición por odometría

Un sistema de odometría en robótica móvil, se usa para la estimación de la posición actual de un robot usando como referencia su localización inicial mediante la detección del movimiento de las ruedas. Este método de estimación de posición tiene un costo bajo y permite altas tasas de muestreo. También se puede considerar que es bastante preciso en distancias cortas. Pero se debe tomar en cuenta que va acumulando pequeños errores durante su trayecto que conllevan a una inevitable desorientación si no es tratado adecuadamente. (Granda & Vázquez, 2012; Riisgaard & Blas, 2004)

El desempeño odométrico se establece por la exactitud y error en el cálculo de su recorrido y dirección. Se considera un desempeño óptimo si el error es menor a 2

centímetros por metro de recorrido y 2 grados por cada 45 grados en cambios de dirección. (J. Borenstein & Liqiang Feng, 1995; Borenstein, Everett, & Feng, 1996)

El error en sistemas odométricos se produce por diversos factores muy difíciles de controlar, tanto en su propia arquitectura como en factores externos del ambiente por el cual se desplaza. A estos errores se los denomina sistemáticos y errores no sistemáticos (Granda & Vázquez, 2012). Entre los errores sistemáticos se destacan:

- Diámetros de ruedas distintos.
- Diámetro nominal de ruedas con excesivo error al diámetro real.
- Desalineación de ruedas.
- Incertidumbre de la verdadera distancia entre ruedas por grosor y deformación de los puntos de apoyo.
- Resolución muy baja en el sensor tipo Encoder.
- Tasas muy bajas de muestreo del conteo del sensor tipo Encoder.

Mientras que los errores no sistemáticos más comunes son:

- Superficies demasiado desiguales o rocosas.
- Repentino cambio de superficies.
- Cambios de inclinación en la superficie.
- Deslizamiento de ruedas sin resistencia.
- Fuerzas externas que modifiquen el curso de la plataforma.

En ambientes controlados, los errores sistemáticos son de mayor importancia, debido a que el entorno se mantiene en condiciones adecuadas que disminuyen significativamente los errores no sistemáticos. Por lo tanto, la exactitud en sistemas odométricos en ambientes controlados dependerá directamente de la calidad de la plataforma usada. (Granda & Vázquez, 2012)

2.2.3 Software para la localización y mapeado simultaneo

El software en un sistema SLAM es el encargado de procesar la información obtenida por los sensores para decidir cómo actuar al entorno enfrentado. Por lo tanto del software depende la correcta interpretación de los datos adquiridos por la plataforma, e influye directamente en las decisiones que tome su controlador.

Los sistemas basados en SLAM tienen una alta dependencia computacional, por ende gran parte de la eficiencia de un sistema SLAM recae sobre su algoritmo. (Riisgaard & Blas, 2004)

2.2.3.1 Algoritmos SLAM

El objetivo del proceso SLAM es usar el entorno para actualizar las posiciones del robot. Todos los algoritmos desarrollados para la solución del problema SLAM, tienen en común el siguiente proceso general (Lemus et al., 2014):

1. Obtener información del entorno cercano mediante sensores.
2. Detectar puntos de interés en el entorno.
3. Calcular la correspondencia entre lo calculado y lo adquirido en el entorno.
4. Calcular la posición, recorrido y dirección del robot.

Dado que la información obtenida por odometría progresivamente aumenta el error mientras mayor es el desplazamiento, no se puede confiar directamente en su información, por lo que generalmente es asociada con los datos sensoriales obtenidas (Riisgaard & Blas, 2004). Varios autores usan un Filtro de Kalman Extendido (EKF de sus sigla en inglés *Filter Kalman Extended*) para corregir la última posición del robot calculando la incertidumbre existente en la posición, corregirla y pronosticar la posible posición futura (Bailey, Nieto, Guivant, Stevens, & Nebot, 2006; G P Huang, Mourikis, & Roumeliotis, 2008; S. Huang & Dissanayake, 2007; Ni, Wang, Fan, & Yang, 2014). Por otro lado existen procesos menos populares que han mostrado buenos resultados como la localización de Monte Carlo (MCL) (Liu, Fan, & Zhang, 2013) y el filtro de partículas RaoBlackwellised (RBPF). (Kuo et al., 2011)

2.2.3.2 Carga computacional

Para la década de los noventa la mayor preocupación era el hardware, por sus limitadas capacidades de procesamiento de información y de almacenamiento. Por tal razón se impulsó la eficiencia de los recursos computacionales para desarrollar un algoritmo SLAM.

El proceso donde más comprometidos están los recursos computacionales es en el mapeado y su registro. De esta forma nacen varias investigaciones dedicadas a crear algoritmos simplificados para resolver la localización y mapeado simultaneo (Carvajal, Luna, & Pardo, 2013; Kuo et al., 2011; Lingelbach, 2004). Entre los trabajos más destacados están los basados en la descomposición de celdas exactas que plantea

dividir el mapa en formas geométricas creadas por los espacios libres que reconoce el agente (Lingelbach, 2004). Otro modelo utilizado es la descomposición de celdas adaptativas usadas en mapas métricos en donde su función es descomponer en celdas cada vez más pequeñas mientras más seguro este de su estado (ver **Figura 9**). (Carvajal et al., 2013)

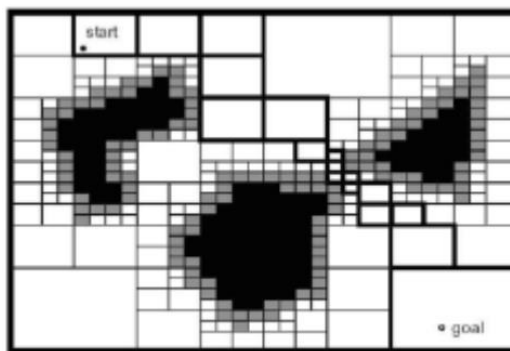


Figura 9. Descomposición de celdas adaptativas. (Carvajal et al., 2013)

Conjuntamente, se logró reducir considerablemente la carga computacional con algoritmos más dedicados que proponen un algoritmo SLAM para ambientes cerrados basado en segmentos de líneas interconectadas mostrado en la **Figura 10**. (Kuo et al., 2011)

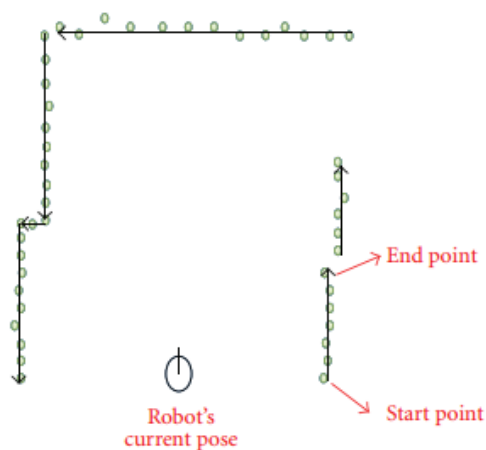


Figura 10. Proyección del punto inicial y final de una línea con el método propuesto por Kuo et al., (2011)

2.3 Problema SLAM en multi robótica

Recientemente los avances en la robótica móvil permiten una autonomía casi completa en muchas aplicaciones de manera exitosa, incluyendo la exploración de ambientes desconocidos (Mohan & Ponnambalam, 2009). El requisito indispensable

para cualquier tarea con un robot móvil en un área desconocida, es la capacidad de navegar de manera autónoma por medio de la información sensorial que permite estimar sus posiciones y el modelo del entorno circundante.

El objetivo de un algoritmo Multi Robótico SLAM (MRSLAM por sus siglas en inglés *Multi Robotic Simultaneous Localization and Mapping*) es construir un mapa global más preciso en base a la fusión de datos y localizar los robots en el mapa de manera que puedan completar la tarea mediante la cooperación entre ellos. Usando mapas compartidos los robots coordinan sus estrategias de exploración para maximizar la eficiencia de la exploración. (Liu et al., 2013)

En la actualidad, los enfoques existentes para MRSLAM se centran principalmente en la expansión de algoritmos para SLAM al caso multi robot, especialmente se han extendido los algoritmos para el control de posición de robots móviles. Entre los más destacados el filtro extendido de Kalman, la localización de Monte Carlo y el filtro de partículas RaoBlackwellised.

El estudio realizado por Huang *et al.*, sobre la fusión coherente de mediciones parciales de multirobots basado en el filtro extendido de Kalman (EKF) mostro un gran avance con respecto a precisión. El método empleado mantiene en conjunto el estado de un mapa global y su matriz de covarianzas. Sin embargo, el error de linealización del sistema y la alta complejidad son los mayores inconvenientes de este tipo de algoritmos basados en EKF. (Guoquan P. Huang, Trawny, Mourikis, & Roumeliotis, 2011)

Existen trabajos especializados en métodos que estiman la postura relativa de otros robots en su propia exploración, para posteriormente combinar los mapas parciales de cada robot a través de un filtro de partículas adaptativo (Chang, Lee, Hu, Lu, & Lafayette, 2007; Ko, Stewart, Fox, Konolige, & Limketkai, 2003; Sim, Dudek, & Roy, 2004). Basado en la sub correlación de mapas como los trabajos mencionados anteriormente, se crea un algoritmo topológico/métrico (Chong & Kleeman, 1999), corregido y mejorado posteriormente con un filtro de sub correlación relativa con restricciones (CRSF) dando como resultado el mapa observado en la **Figura 11**. (Chang, Lee, Hu, Lu, et al., 2007; Chong & Kleeman, 1999; Williams, 2001)

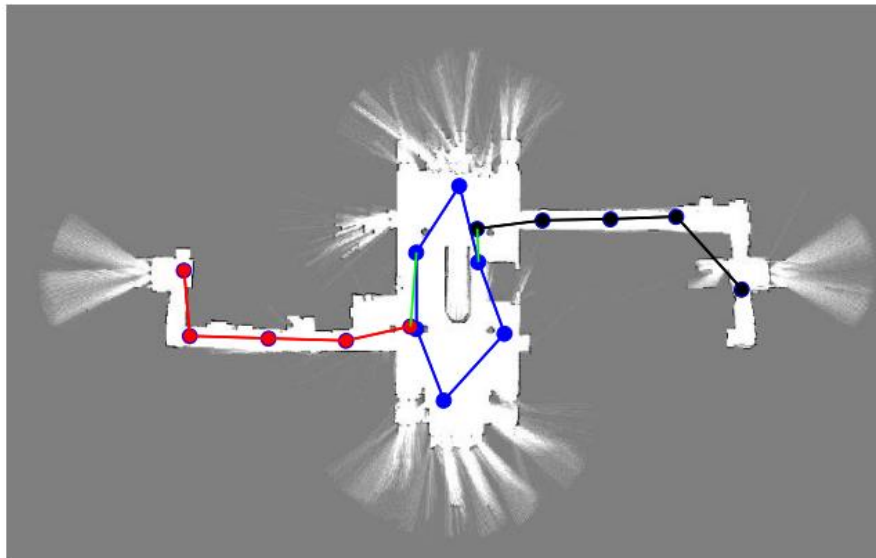


Figura 11. Mapa creado por 3 robots con un algoritmo topológico/métrico. (Chang et al., 2007)

Simultáneamente Thrun y Liu, (2003), resuelven el problema MRSLAM utilizando el Filtro de Información Dispersa Extendido (SEIF) en el cual los mapas y las posiciones de los robots se representan usando campos aleatorios de Markov gaussianos. Este enfoque descentralizado actualiza un subgrupo de todos los puntos de referencia con el fin de ganar eficiencia informática. El mismo autor presentó el método de construcción de mapas híbridos con varios robots que combina el SEIF con el localizador de Monte Carlo y señala que para construir un mapa con varios agentes es clave determinar la orientación y la posición inicial de cada robot. (Thrun & Liu, 2005).

Al mismo tiempo, métodos de control centralizado se crearon con algoritmos de coordinación eficaz que eligen un objetivo en el entorno basado en el coste de alcance y la utilidad del punto objetivo; considerando que la comunicación entre robots es limitada (Alexandre, 2013). Este trabajo se basó en la técnica de coordinación explícita entre robots presentado por Simmons, (2000) que mostro una significativa reducción del tiempo de ejecución de la tarea con resultados como el que se aprecia en la **Figura 12**. (Simmons, Apfelbaum, Burgard, & Fox, 2000)

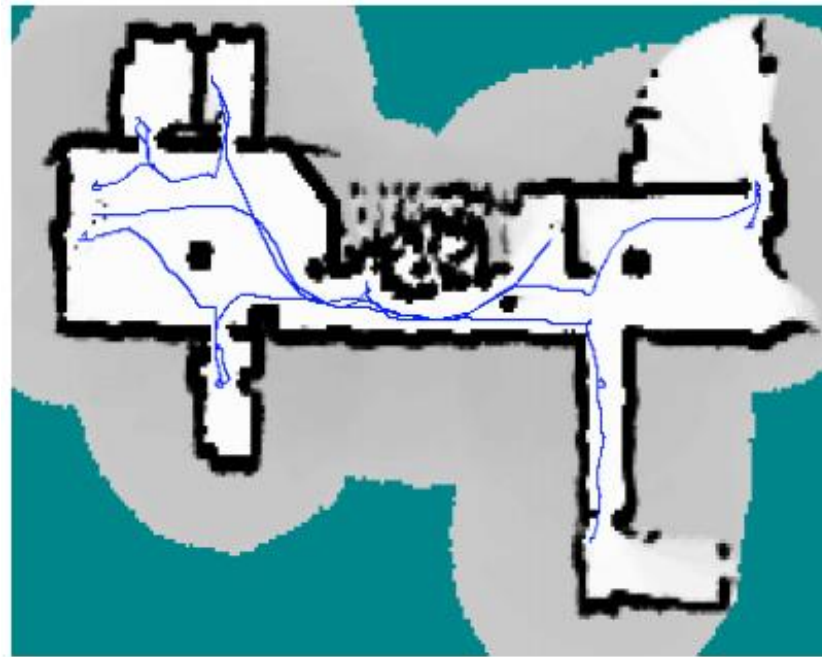


Figura 12. Mapa creado por 3 robots por el método de coordinación explícita de Simmons, (2000)

Para elegir puntos de objetivo apropiados para los robots que exploran simultáneamente las diferentes regiones del medio ambiente, se han propuesto varios métodos de cooperación intensa incluyendo programación entera y un esquema de subasta para la asignación de objetivos a los robots (Simmons et al., 2000; Vincent et al., 2008; Zlot, Stentz, Dias, & Thayer, 2002). Mientras otro estudio de baja cooperación emplea métodos con varios comportamientos básicos como ir hacia fronteras, evitar obstáculos y evitar robots con muy buenos resultados (Lau, 2003).

Por otro lado se demostraba la eficiencia de métodos descentralizados, donde se propone extender algoritmos SLAM basados en filtros de partículas de Rao-Blackwellized a aplicaciones multi agente (A. Howard, 2006; a. Howard, 2004). En dicho trabajo se crean múltiples mapas y se unen mediante algoritmos de máxima verosimilitud cuando se produce un encuentro. Esta investigación da una nueva dirección en métodos basados en partículas, suponiendo que cada robot es capaz de identificar a sus compañeros en el ambiente y que las posiciones iniciales de los robots son desconocidas.



Figura 13. Robots móviles utilizados por Howard, (2006) con torres codificadas para su identificación autónoma.

Un enfoque parecido fue presentado por Fox et al., (2006), donde los robots exploran el ambiente desde posiciones diferentes y desconocidas. Este trabajo tiene la particularidad de que cada agente comprueba activamente sus posiciones relativas para asegurar la coherencia en la fusión de datos en los mapas compartidos (ver **Figura 13**).

Considerando la incertidumbre de la información existente en los mapas, se presentó un nuevo algoritmo basado en anteriores trabajos sobre Fast SLAM multi robot (Chen et al., 2014). Este método usa un mecanismo de atracción y repulsión inspirado en el comportamiento de los campos electromagnéticos para coordinar el movimiento de los integrantes del sistema multi robótico. Logrando disminuir la incertidumbre en el mapa creado y asegura una distribución amplia en el entorno.

En muchos trabajos se asume que la posición inicial es conocida, lo que en muchas ocasiones reales no es muy cierto. Para tomar en cuenta esta problemática adicional a SLAM se han creado varios métodos que generan la trayectoria del agente de forma pasiva (Burgard et al., 2005; Tanaka, 2009; Vincent et al., 2008). Mientras tanto, con el fin de mejorar la calidad del mapa se desarrollaron estrategias de control de rutas de acceso activas (Ji, Zhang, Hai, & Zheng, 2009; C. Leung, Huang, & Dissanayake, 2006; Sim et al., 2004). Hasta ahora muy pocos autores han llevado estas consideraciones del SLAM activo con múltiples robots, mucho menos consideran una etapa de relocalización. (Tovar et al., 2006; Pham & Juang, 2013)

En conclusión, comparado con métodos SLAM con un solo robot, los algoritmos de MRSLAM no sólo pueden obtener información del ambiente más precisa, sino que también puede cubrir todo el entorno más rápido a través de la comunicación y coordinación en paralelo; incluso los errores de localización y mapeo pueden ser reducidos a través de la integración de la información medida en el entorno. (Chen et al., 2014)

2.3.1 Unión de mapas en sistemas MRSLAM

El mapeo es el resultado del seguimiento de la posición y la integración de la información sensorial a la estimación del entorno. El problema del mapeado en el caso de multi robots es similar al problema sobre un robot móvil, excepto por el hecho de que en múltiples robots se utiliza un conjunto de datos de sensores que funciona simultáneamente en varios agentes que tienen una alta velocidad para reconocer el entorno y proporcionar un mapa desde su posición.

La investigación sobre la fusión de mapas es limitada, y trabajos anteriores se centraron en mapas basados en hitos especiales que pueden ser reconocidos a través del procesamiento adecuado de los datos recogidos por los robots (Benedettelli, Garulli, & Giannitrapani, 2012; Dedeoglu & Sukhatme, 2000; W. H. Huang & Beevers, 2005). Por ejemplo, los mapas topológicos son gráficos en los que los vértices representan lugares reconocibles tales como puertas, corredores y diferentes tipos de esquinas y bordes representan el paso que conecta dos lugares.

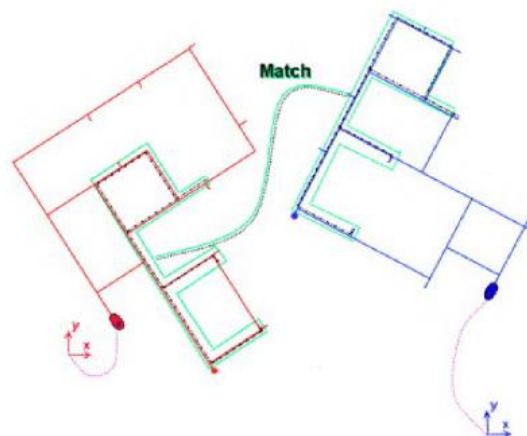


Figura 14 Ejemplo de unión de 2 mapas topológicos en (Alexandre, 2013)

De esta forma, la fusión del mapa A y del mapa B en la **Figura 14** se convierte en una búsqueda de sub gráficos con la misma estructura en ambos mapas (Liu et al., 2013). Otros estudios relacionados, como el de Chang *et al.*, presentan un algoritmo descentralizado de MRSLAM que utiliza mapas topológicos. Donde los vértices contienen información métrica local y los bordes describen la posición relativa de los mapas locales adyacentes. (Chang, Lee, Hu, Lu, et al., 2007)

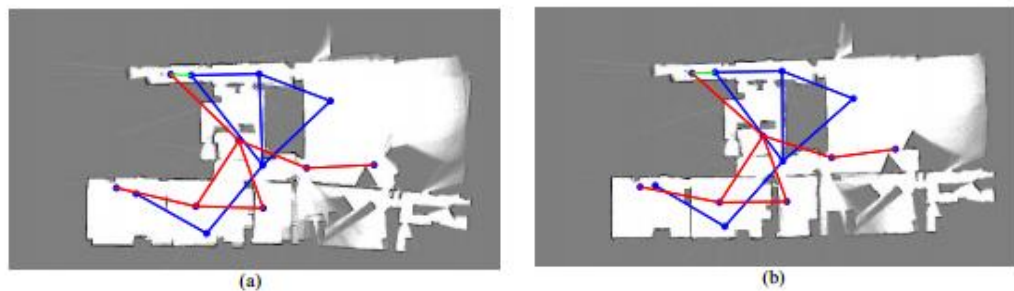


Figura 15 (a) Unión de mapas de 2 robots antes de la optimización global. (b) Unión de mapas después de la optimización global. (Chang et al., 2007)

En este trabajo, los mapas se fusionan naturalmente entre los robots a través de la inclusión de un borde que establece una conexión entre los mapas topológicos, y la estimación de las poses relativas de los robots se realiza mediante la optimización de dicho borde (ver **Figura 15**).

En posteriores investigaciones se aborda el problema de alinear y combinar mapas creados por varios robots usando observaciones hechas entre ellos. La solución utiliza suavizado de información cuadrático (C-SAM) para combinar mapas creados por diferentes robots de forma distribuida. La principal contribución de este trabajo es el algoritmo utilizado para resolver el problema de asociación de datos y eliminar las observaciones falsas cuando se realiza la alineación del mapa durante el encuentro. (Andersson & Nygard, 2009)

Posteriormente, Liu presenta un algoritmo rápido y preciso para combinar múltiples mapas representados como cuadrículas de ocupación mostrada en la **Figura 16**. La idea principal es crear un robot móvil virtual en un mapa parcial y controlar su movimiento en el mapa. Al mismo tiempo, estos datos simulados se utilizan como fuentes de información en el resto de mapas parciales para crear un mapa global mediante el localizador de Monte Carlo (Behzadian, Agarwal, Burgard, & Tipaldi, n.d.). Si la localización tiene éxito, las hipótesis de posiciones relativas entre los mapas

se pueden calcular fácilmente sin necesidad de conocer previamente la postura relativa entre mapas o robots, lo que mejora la exploración autónoma y el SLAM multi-robot. (Liu et al., 2013)

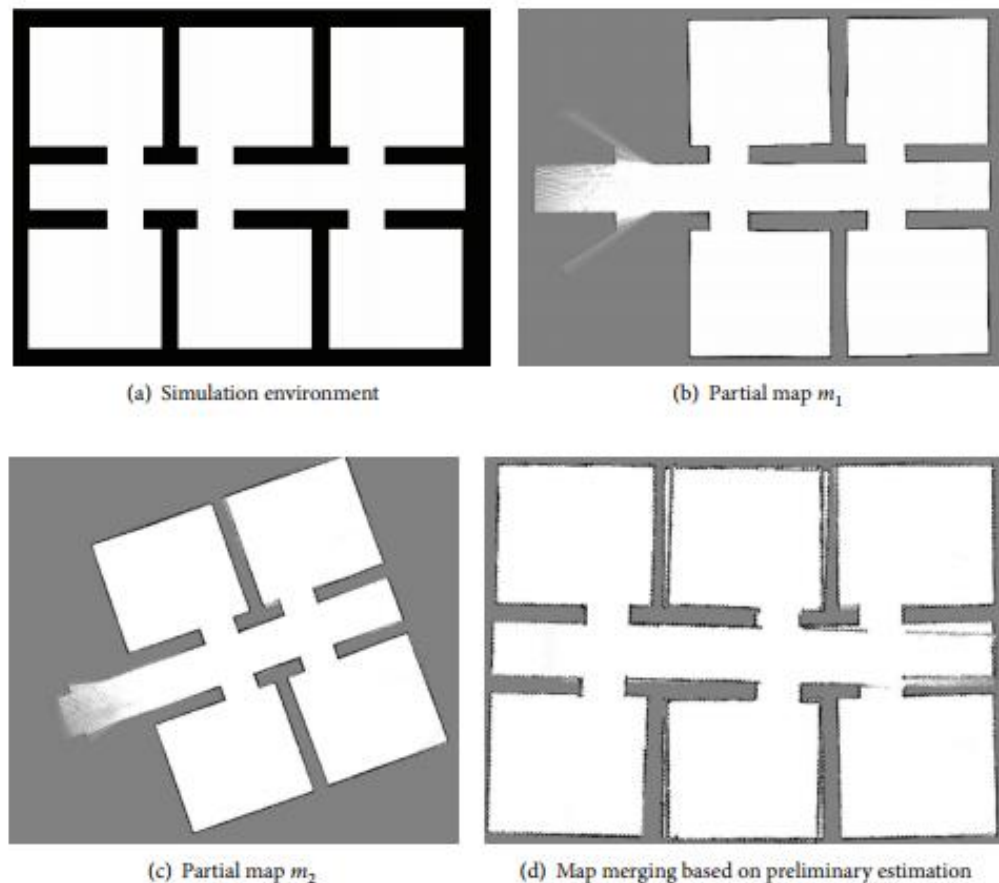


Figura 16 Unión de mapas en (Liu et al., 2013).

En el trabajo presentado por Konolige *et al.*, se decide fusionar dos mapas cuando los dos robots están en el rango de comunicación pero sin conocimiento de su ubicación relativa. Uno de los robots recibe datos del sensor del otro agente e intenta calcular su ubicación, haciendo coincidir la información recibida con su propio mapa. Si no se encuentran en la ubicación esperada, se rechaza la hipótesis y los mapas parciales no se fusionan. Caso contrario, si lo consiguen, se acepta la hipótesis y sus mapas se combinan permanentemente. En la **Figura 17** se observa cómo se fusionan dos mapas exitosamente después de aplicar dicho algoritmo. (Konolige, Fox, Limketkai, Ko, & Stewart, 2003)

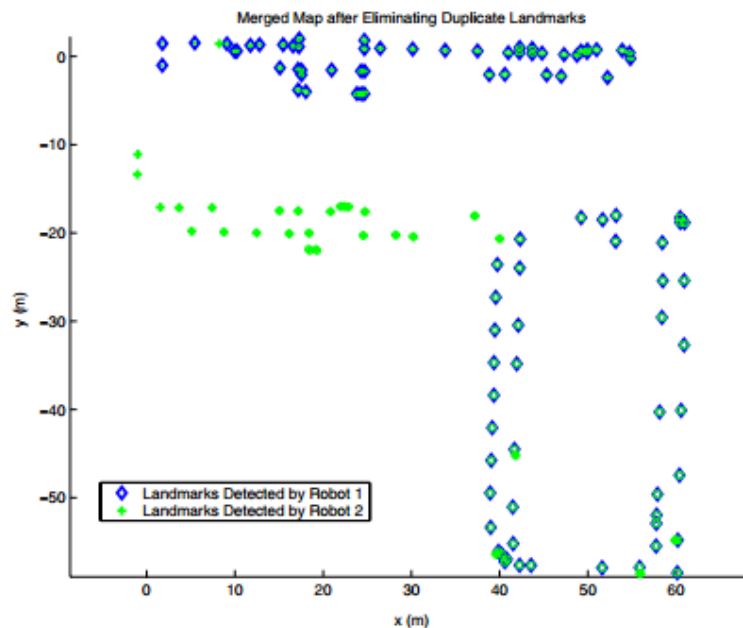


Figura 17 Método basado en puntos de referencia usado en (Zhou & Roumeliotis, 2006).

2.3.2 Comunicación en sistemas MRSLAM

La comunicación de los robots es un tema con muy poca información abierta sobre la metodología y protocolos usados. En muchos trabajos se considera que la comunicación es ilimitada y perfecta, o simplemente no se toma en cuenta por producirse en entornos virtuales (Rooker & Birk, 2007; Vazquez & Malcolm, 2004; Weihua Sheng, Qingyan Yang, Song Ci, & Ning Xi, n.d.). Recientemente se ha propuesto una estrategia de exploración que permite a los robots navegar más allá del rango de comunicación (Julian De Hoog, Cameron, & Visser, 2009), donde intercambian dinámicamente entre una fase de exploración y una fase de retorno para reconectarse con el controlador central.

Extensiones de esta estrategia de exploración pueden encontrarse en (J. de Hoog, Cameron, & Visser, 2010; Julian De Hoog, Cameron, & Visser, 2010), con un enfoque jerárquico y un nuevo procedimiento de selección de puntos de encuentro. Sin embargo, estos métodos cooperativos que permiten a los robots ir más allá del rango de comunicación, asumen que la localización y los datos de los sensores son perfectos. (Pham & Juang, 2013)

El método descentralizado basado en Fast SLAM, considera que la comunicación es limitada teniendo en cuenta la distancia entre los robots y las posiciones iniciales

desconocidas de los robots (Carlone, Ng, Du, Bona, & Indri, 2010). Al igual que Leung, examina un sistema SLAM descentralizado cooperativo, en el cual los robots necesitan estimar los mapas y los estados de todos los otros robots suponiendo que la comunicación entre ellos es limitada y la conexión es dinámica. (K. Y. K. Leung, Barfoot, & Liu, 2012)

2.4 Descripción del proyecto

Después de haber presentado los temas relevantes para el cumplimiento de este trabajo de investigación, se muestra una breve descripción de las actividades a realizarse en el debido orden lógico para cumplir el objetivo planteado.

Este proyecto de investigación forma parte del proyecto RoboSenseSmell (2016-pic-009), donde el objetivo es navegar de forma autónoma en ambientes desconocidos para encontrar distintas fuentes de olor para la localización de explosivos. Con este antecedente, se buscó diversas soluciones que puedan ser integradas a este tema de forma eficiente para cumplir una de las necesidades de la navegación autónoma.

Después de llevar una exhaustiva investigación se encontró fiable el uso de múltiples robots autónomos que puedan recoger información del ambiente en distintas posiciones al mismo tiempo. Además de mejorar la eficiencia del sistema, es posible simplificar las unidades robóticas por su modularidad.

El siguiente paso lógico es limitar el alcance de este trabajo de investigación para diseñar el hardware y seleccionar los componentes para el control, comunicación y sensado de cada robot móvil. Tomando en cuenta la arquitectura, simplicidad, el coste y la adaptabilidad de dicha plataforma. Adicionalmente, como el proyecto general está dirigido a la toma de muestras ambientales, se decidió implementar sensores de temperatura a cada plataforma para extender las posibilidades de cada robot.

Posteriormente, de acuerdo a las necesidades del proyecto RoboSenseSmell, los robots son controlados por un robot nodriza que en un futuro podría crear rutas y tomar muestras del entorno. Esto quiere decir que se necesita un tipo de control centralizado que almacene y tome decisiones sobre los robots móviles.

Con el mismo criterio se decidió usar un medio inalámbrico de comunicación que permita el envío y recepción de información simultáneamente con el controlador central. Que a la vez se encargara de mostrar los resultados en una interfaz gráfica para su posterior evaluación.

Una vez implementada la estructura del sistema multi robótico de exploración, se deben realizar pruebas de calibración que disminuyan el error en el sistema; y finalmente hacer pruebas y experimentos con un solo robot y varios robots móviles que confirmen la viabilidad del proyecto. Lo que va a permitir concluir con un análisis de los resultados obtenidos y definir nuevos temas que pueden ser tratados en trabajos futuros.

CAPÍTULO III

DISEÑO E IMPLEMENTACIÓN DE LAS PLATAFORMAS MÓVILES ROBÓTICAS

3.1 Introducción

En este capítulo se detalla el proceso de diseño e implementación de las plataformas robóticas móviles homogéneas para resolver la problemática descrita en el capítulo 1. Basándose en las consideraciones de diseño presentadas y en el entorno donde será utilizado. Se describe cuáles serán las características constitutivas de los agentes incluyendo el tipo de plataforma, el controlador, medios de comunicación, los sensores y actuadores.

Por otra parte, se detalla el proceso de montaje y se representa gráficamente las conexiones y circuitería de cada plataforma robótica. Finalmente, se presenta una tabla descriptiva de los costos de implementación del hardware completo.

3.2 Consideraciones de diseño

En base a la definición de multi robótica SLAM y la aplicación determinada por los objetivos del proyecto se decidieron las siguientes consideraciones globales que se describen a continuación:

- dimensiones de cada agente
- Simplicidad en el diseño cinemático
- Adaptabilidad y modularidad
- Entorno de trabajo
- Numero de robots cooperativos
- Bajo coste general

Para lo cual, cada plataforma robótica móvil del sistema cooperativo debe contar con las siguientes etapas electrónicas: controlador, sensores, comunicación inalámbrica, etapa de potencia y motores. Presentadas en el diagrama de bloques de la **Figura 18**, las cuales deben ser consideradas para elegir la estructura base del mismo.

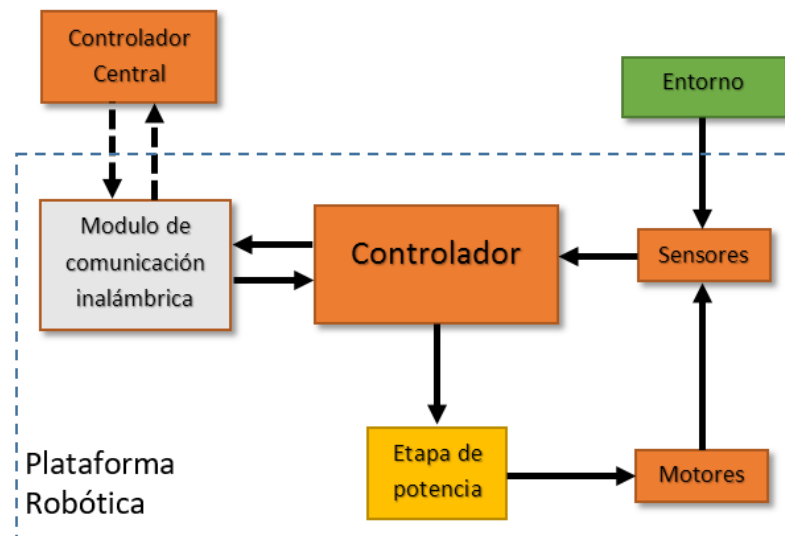


Figura 18. Esquema del hardware electrónico del robot.

Adicionalmente, se toma en cuenta que las plataformas móviles tendrán idénticas características físicas y se utilizara el mismo diseño mecánico de fábrica como base para el montaje de los elementos.

3.3 Plataforma robótica móvil

La plataforma móvil robótica como se definió en el capítulo 2 es la base principal de trabajo en sistemas de navegación, porque su configuración mecánica será responsable de cumplir con la tarea exitosamente. Existe una extensa variedad de vehículos robóticos que pueden ser utilizados en el ámbito de la exploración y navegación. Sin embargo, no existe un robot móvil universal por la enorme cantidad de variables externas a enfrentar en entornos cambiantes. Por ende el tipo de robot está directamente relacionado con su aplicación.

A continuación se presenta las posibles soluciones comerciales existentes, se analiza los tipos de plataformas móviles y sus principales aplicaciones según su estructura, se encuentra la plataforma con el mejor equilibrio entre coste y funcionalidad que mejor se ajuste a las necesidades de este trabajo, y se muestra las características del modelo elegido.

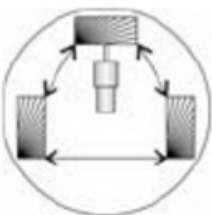
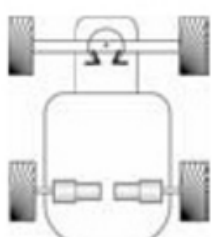
3.3.1 Tipo de plataforma móvil

Al ser un trabajo de mapeado superficial, limita el uso de vehículos robóticos de tipo terrestre. Dentro de las plataformas robóticas terrestres existen distintos métodos

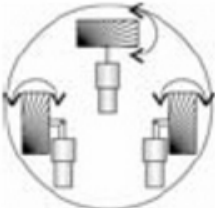
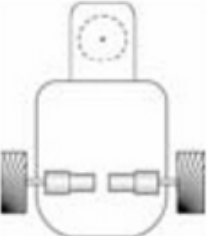
de tracción descritos en el capítulo 2. Entre ellos los que se destacan para navegación y mapeado son de tipo oruga o con ruedas. Las plataformas con orugas muestran mayor versatilidad de movimiento en distintos terrenos pero incrementan el error odométrico al tener puntos de apoyo muy variables. En cambio las plataformas con ruedas son más controlables a nivel odométrico pero tienen menor accesibilidad a distintos terrenos irregulares. Como este trabajo está dirigido para espacios interiores de terrenos lisos, la mejor opción es una plataforma con ruedas.

Dentro de los robots con ruedas, también hay una amplia variedad de estructuras y configuraciones con distintas habilidades y propiedades. A continuación se presentan los modelos tomados en cuenta y se analiza sus principales características en la tabla 1.

Tabla 1.
Tipos de robots móviles con ruedas (Fabrizio et al., 2013)

Modelo	Descripción	Características
Síncrono 	<p>Posee 3 ruedas controladas por dos motores.</p> <p>Un motor hace girar las 3 ruedas en la misma dirección para avanzar en línea recta y el segundo motor genera la dirección del robot.</p>	<p>Necesita detenerse para efectuar giros.</p> <p>Eficiente en ambientes controlados y conocidos.</p> <p>No es posible controlar su orientación.</p>
Ackermann 	<p>Está compuesto por 4 ruedas. 2 ruedas traseras con el mismo eje y 2 ruedas delanteras que pueden girar en sincronía para cambiar de dirección.</p>	<p>Se controla por un solo motor ya sea de tracción trasera o delantera.</p> <p>Maniobrabilidad limitada, difícil posicionamiento y complicado modelamiento cinemático.</p> <p>Excelente estabilidad, útil para realizar giros sobre la marcha a velocidad.</p>

Continua →

<p>Omnidireccional</p> 	<p>Generalmente constituidos de 3 o 4 ruedas omnidireccionales o ruedas suecas. Su principal habilidad es el cambio instantáneo de dirección sin cambiar su orientación debido al rozamiento de sus ruedas especiales.</p>	<p>Excelente maniobrabilidad. Necesita de algoritmos de control complejos. Solo es funcional dentro de ambientes controlados con superficies planas. La distribución del peso en la plataforma afecta el funcionamiento de las ruedas.</p>
<p>Diferencial</p>  <p>(Silva Ortigoza et al., 2010)</p>	<p>Posee 2 ruedas controladas cada una con su propio motor y una tercera rueda sin tracción de tipo pivotante o esfera que sirve como apoyo para la estabilización de la plataforma. Las ruedas están colocadas de forma paralela entre ellas y el cambio de dirección es posible por el cambio de velocidad entre las ruedas. Inclusive puede girar sobre su propio eje cuando la dirección es inversa entre las ruedas.</p>	<p>Posee un modelo cinemático sencillo. Necesita sincronización entre la velocidad de las ruedas. Difícil control sobre superficies irregulares. Excelente maniobrabilidad. Fácil cálculo de posicionamiento.</p>

3.3.2 Consideraciones de diseño para la elección de una plataforma diferencial

Tomando en cuenta la tabla 1 y las consideraciones generales mostradas anteriormente, se eligió un robot de configuración diferencial. A continuación se detalla las consideraciones más relevantes que se tomaron en cuenta para elegir esta plataforma con respecto a los distintos modelos mencionados anteriormente.

3.3.2.1 Estabilidad mecánica

La estabilidad mecánica de la plataforma móvil es la propiedad que tiene sus puntos de apoyo con el suelo para mantenerse en equilibrio en él, tanto en movimiento como

en reposo. Como se describe en la tabla 1, el modelo de Ackermann tiene la mejor estabilidad pero su falta de maniobrabilidad no lo hace ideal para este proyecto, por lo tanto su uso quedo descartado. La siguiente opción es la configuración de 3 ruedas con el centro de gravedad dentro del triángulo formado por las ruedas que ofrece una buena estabilidad en velocidades constantes y controladas (ver **Figura 19**). Esta configuración es usada en el modelo diferencial, síncrono y omnidireccional, lo que simplifica su control y maniobrabilidad sacrificando la estabilidad sobre terrenos irregulares. Entre estos 3 modelos el más simple mecánicamente es el modelo diferencial lo que simplifica su modelado cinético y disminuye su tamaño. Aunque en realidad es una configuración de 2 ruedas, el uso de una tercera rueda pivotante en el diseño práctico permite que su estabilidad sea exitosa.

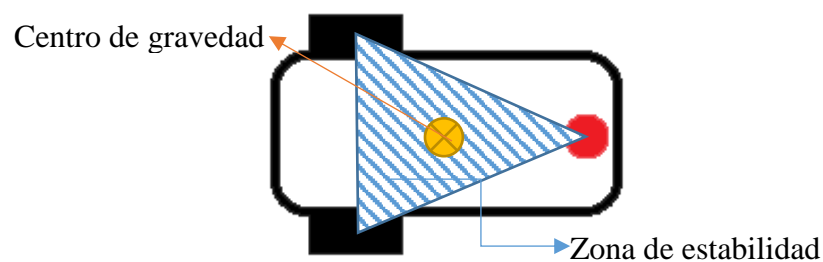


Figura 19. Estabilidad de un robot móvil diferencial.

3.3.2.2 Maniobrabilidad

La maniobrabilidad es la capacidad del robot para cambiar de posición con el menor esfuerzo posible, es decir que mientras mayor maniobrabilidad posea una plataforma mayor será su capacidad de evadir obstáculos. En este aspecto el modelo de mejor maniobrabilidad es el omnidireccional que cuenta con una alta precisión al momento de cambiar de dirección. La desventaja en este modelo es que su desempeño depende totalmente de la fricción de las ruedas con el suelo y su modelado es complejo.

Por otro lado, el modelo diferencial necesariamente debe girar todo su cuerpo para cambiar de dirección, es decir necesita cambiar su orientación para cambiar su posición. Conjuntamente dependiendo de sus componentes mecánicos será mayor o menor su precisión comparada con otros modelos. Nuevamente su ventaja en este ámbito es su simplicidad, al ser tan sencillo disminuye su costo y simplifica su modelado cinemático.

La maniobrabilidad está relacionada directamente con el control de los grados de libertad de un robot, por lo tanto si menor son los grados de libertad menores son las posibilidades de maniobrar. Por ese motivo es considerado dentro de los parámetros necesarios para elegir una plataforma.

3.3.2.3 Grados de libertad

Los grados de libertad de un robot móvil generalmente son 3, la posición respecto al eje de coordenadas rectangulares (x, y) y la orientación de la plataforma robótica. El modelo de Ackermann controla estos grados de libertad de manera indirecta puesto que trabaja con movimientos circulares para alcanzar su posición y orientación adecuada. En cambio el modelo omnidireccional y diferencial controla por completo los 3 grados de libertad de manera directa, a estos se les llaman grados de libertad diferenciables (x, y, θ) identificados en la **Figura 20** en un modelo diferencial.

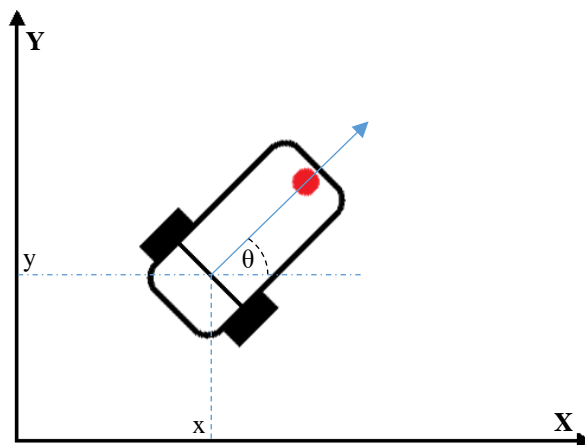


Figura 20. Grados de libertad de un robot diferencial.

3.3.2.4 Odometría

Los cálculos odométricos pueden estimar la posición de un robot con buenos resultados dependiendo de los errores sistemáticos y no sistemáticos existentes en toda plataforma robótica móvil. No obstante la simplicidad del cálculo odométrico puede ayudar a mejorar o compensar los errores producidos por el sistema y el entorno. En este parámetro el modelo de mayor fiabilidad es el modelo diferencial ya que al estar constituido de menor número de ruedas controlables su modelado se simplifica bastante respecto a los demás.

3.3.2.5 Selección de la plataforma robótica

En conclusión, para el cumplimiento de los objetivos de este proyecto de investigación se eligió una plataforma robótica de modelo diferencial comercialmente conocida como Robot 2WD la cual es vendida por distintas marcas como un modelo genérico y adaptable a varios elementos.

El robot tipo diferencial para los propósitos de esta investigación, sobresale en estabilidad y maniobrabilidad por su configuración sencilla. Además, es de fácil manejo por poseer 3 grados de libertad de control directo y se destaca en odometría por su modelado cinético sencillo. Por lo tanto, el robot móvil elegido cumple con los parámetros necesarios para navegar en el entorno de pruebas de este trabajo.

3.3.3 Arquitectura de la plataforma móvil diferencial

Una vez investigado y analizado las posibles soluciones para el cumplimiento de la tarea de exploración, se decidió adquirir 3 plataformas robóticas móviles diferenciales 2WD de 21cm x 16cm x 4cm aproximadamente, la cual tiene 2 ruedas con llantas de caucho conectadas directamente a 2 motores DC con caja reductora de eje biaxial y una rueda libre pivotante o de cola giratoria para ofrecer estabilidad al largo de la estructura (ver **Figura 21**).

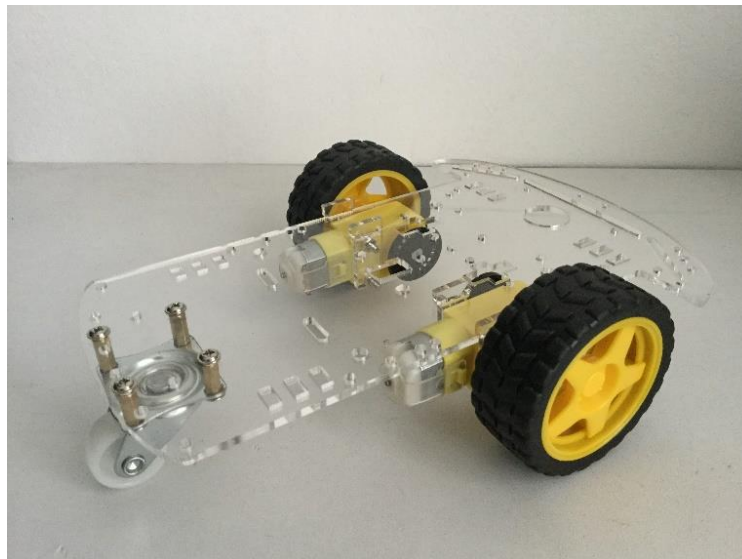


Figura 21. Plataforma Robótica Diferencial 2WD

El chasis está elaborado en acrílico y cortado con láser de alta precisión, también viene con perforaciones para que se pueda adecuar varios sensores y controladores. Las ruedas tienen un diámetro de 6.5cm aproximadamente y están conformadas por un aro de plástico sólido recubierto por una llanta de goma poco flexible. La llanta pivotante tiene un diámetro de 2.5cm aproximadamente y su estructura giratoria de aluminio posee rulimanes que producen el efecto giratorio libre característico de este tipo de ruedas.

En la **Figura 22** se detalla las medidas exactas proporcionadas por el fabricante de la estructura que conforman el robot móvil.

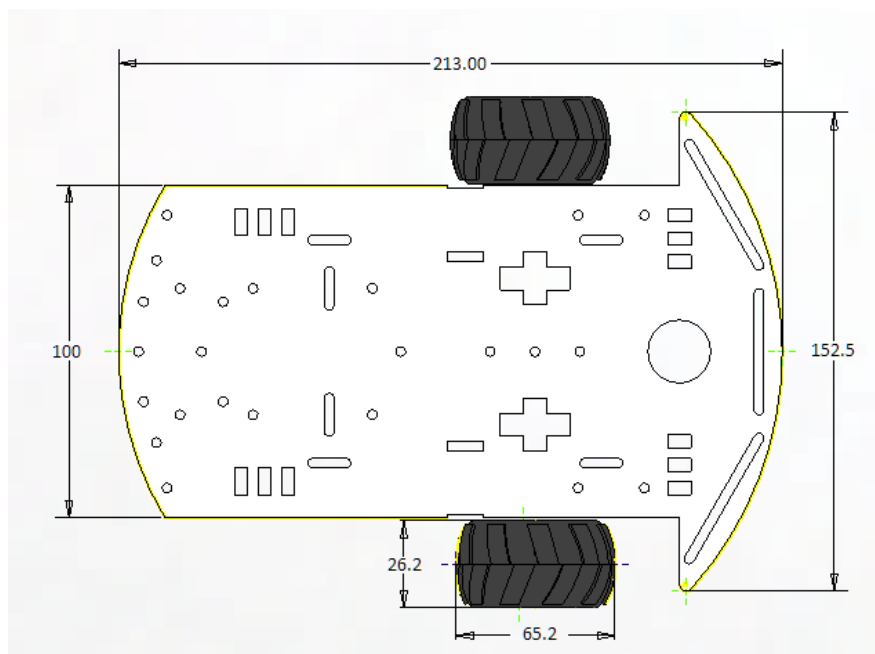


Figura 22. Cotas generales de la plataforma móvil utilizada en mm.

3.3.4 Cinemática de un robot móvil diferencial

La cinemática de un robot de modelo diferencial es la representación matemática del comportamiento del robot al mover sus dos ruedas en sincronía para llegar a una posición deseada. Este modelo matemático es útil tanto para aplicar algoritmos de control sobre la velocidad y dirección del robot móvil, como para realizar estimaciones de posición odométricas.

Para este fin se estudia las reacciones físicas a variantes de velocidad y dirección en los motores que controlan cada rueda tomando en cuenta condiciones ideales del entorno y de la arquitectura del sistema.

3.3.4.1 Geometría del robot diferencial

A continuación definimos las variables necesarias para el cálculo de la cinemática directa del robot diferencial utilizado. Entre las consideraciones a tomar está los grados de libertad diferenciables descritos anteriormente en la figura 20. Aparte de la información con respecto al plano, se necesita información sobre las condiciones de la plataforma tales como: la distancia que existe entre los puntos de apoyo de las dos ruedas ($2L$), el diámetro (D) y radio (r) de cada rueda en condiciones ideales representados en la **Figura 23**.

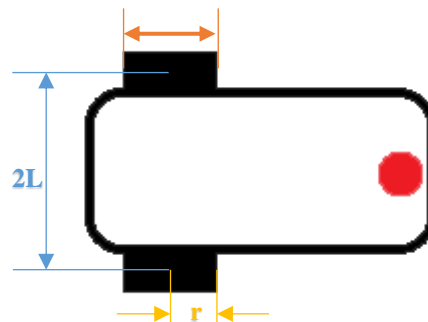


Figura 23. Geometría de un robot diferencial.

3.3.4.2 Cinemática directa

Mediante la cinemática directa se modela matemáticamente la respuesta del sistema conformado por (x, y, θ) a través de las variables de entrada que para este tipo de plataforma son las velocidades angulares de cada rueda controlada (ω_d, ω_i) .

Para realizar el cálculo correspondiente mediante el método de superposición se asume el movimiento de cada rueda por separado. Donde una rueda se mantiene estática y ejerce como el eje de rotación del movimiento angular de la otra rueda correspondiente. Donde el radio generado se considera una constante en el sistema ($2L$) y la velocidad angular de la rueda móvil una variable (ω_d, ω_i) . En la **Figura 24** se puede observar el efecto de este movimiento cuando la rueda izquierda sufre cambios de su velocidad mientras la rueda derecha se mantiene estática.

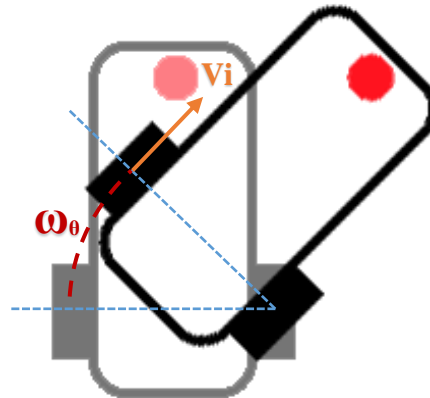


Figura 24. Movimiento de rotación con una rueda bloqueada.

Se observa que la velocidad lineal producida en la rueda izquierda (V_i) afecta directamente al centro instantáneo de rotación que en este caso se genera en la rueda derecha que está bloqueada. Esto produce una velocidad angular (ω_θ) que actúa en la plataforma haciéndola girar en el sentido de las manillas del reloj.

El efecto contrario se produce cuando la rueda izquierda se mantiene estática mientras la rueda derecha gira a una velocidad (V_d). Su movimiento es circular con una velocidad angular (ω_θ) sobre el centro instantáneo de rotación en la rueda izquierda en sentido anti horario.

La velocidad lineal en una rueda se define como el producto entre el radio de la rueda y la velocidad angular de giro. Con la misma consideración se calcula la velocidad lineal en la rueda derecha donde el radio es la distancia entre las ruedas y la velocidad angular la rotación producida con eje en la rueda izquierda (ver **Figura 25**).

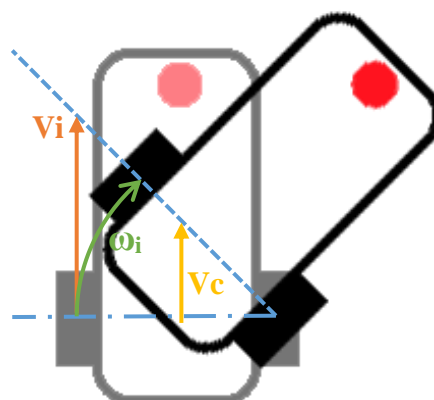


Figura 25. Modelado cinemático en rueda izquierda.

Realizando el análisis de la salida del vector posición (X, Y, θ) mediante el método de superposición, cuando actúa una sola rueda se tiene que la velocidad en dicha rueda es:

$$V_i = r \cdot \omega_i \quad (1)$$

Donde V_i es la velocidad de desplazamiento lineal de la rueda izquierda, r el radio desde el eje de la rueda hasta el punto de apoyo y ω_i es la velocidad angular con respecto a su eje.

En el mismo caso, ahora la velocidad V_i aplicada al robot con eje de rotación en la rueda derecha, tenemos que:

$$V_i = 2L \cdot \omega_\theta \quad (2)$$

Donde $2L$ es dos veces la distancia desde el centro al punto de apoyo de la rueda y ω_θ es la velocidad angular del robot con eje de rotación en la rueda izquierda.

Por otro lado, la velocidad a la que se desplaza el centro del robot (V_c) en relación a la velocidad angular del mismo:

$$V_c = L \cdot \omega_\theta \quad (3)$$

Considerando las ecuaciones 1 y 2 y despejando la velocidad angular se tiene que:

$$\omega_\theta = \frac{r \cdot \omega_i}{2L} \quad (4)$$

Y remplazando finalmente la ecuación 4 en 3 se encuentra la velocidad del centro del robot con respecto a la velocidad lineal de la rueda.

$$V_c = \frac{r \cdot \omega_i}{2} \quad (5)$$

La velocidad angular y velocidad lineal presentadas en las ecuaciones 4 y 5 correspondientemente, son velocidades parciales del centro del agente, ya que pertenecen solamente al movimiento de la rueda izquierda. Por lo tanto se realiza el mismo proceso para modelar la rueda derecha del robot.

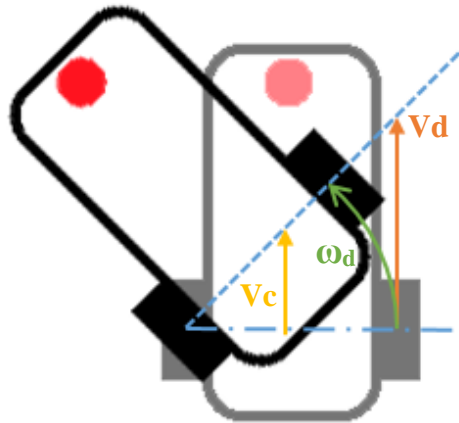


Figura 26. Modelado cinemático de rueda derecha.

Al igual que la rueda izquierda, se genera un desplazamiento circular pero con eje en su rueda izquierda lo que produce una velocidad angular ω_d generada por un desplazamiento de la rueda derecha con velocidad lineal V_d como se observa en la **Figura 26**. Entonces tenemos que:

$$V_d = r \cdot \omega_d \quad (6)$$

Donde V_d es la velocidad lineal de la rueda derecha, r el radio de la rueda derecha que en teoría es el mismo que el de su rueda adyacente y ω_d que representa la velocidad angular en dicha rueda.

La velocidad de rotación del robot en este caso es en sentido horario por lo que la velocidad angular en el mismo se produce con una rotación negativa. Por lo tanto, la velocidad lineal de la rueda derecha con respecto a la plataforma es:

$$V_d = 2L \cdot -\omega_\theta \quad (7)$$

Entonces, la velocidad desde el centro del agente con respecto al movimiento de la rueda derecha es:

$$V_c = L \cdot -\omega_\theta \quad (8)$$

Despejando la velocidad angular de las ecuaciones 6 y 7 se tiene que:

$$\omega_\theta = -\frac{r \cdot \omega_d}{2L} \quad (9)$$

Finalmente, reemplazando la ecuación 9 en 8 se encuentra que la velocidad lineal del centro del robot con respecto al movimiento de la rueda derecha es:

$$V_c = \frac{r \cdot \omega_d}{2} \quad (10)$$

Entonces, las ecuaciones 5 y 10 simbolizan el comportamiento de la velocidad lineal en el eje de traslación del agente, y las ecuaciones 9 y 4 a la velocidad angular desde el centro de la plataforma robótica como respuesta al movimiento de ambas ruedas. Estas velocidades son representadas en la siguiente matriz:

$$\begin{bmatrix} V_x \\ V_y \\ \omega_\theta \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ \frac{1}{2L} & \frac{1}{2L} & 0 \end{bmatrix} \cdot \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \omega_d \\ \omega_i \\ 0 \end{bmatrix} \quad (11)$$

Como se observa no existe componente de velocidad en el eje Y, debido a que no existe un desplazamiento lineal sobre ese eje con respecto al robot. Esta matriz representa las velocidades del robot en su propio sistema de coordenadas. Esto quiere decir que aunque no exista una componente de velocidad en Y en esta matriz, si puede existir dicha componente en el plano donde se desplace como se observa en la **Figura 27**.

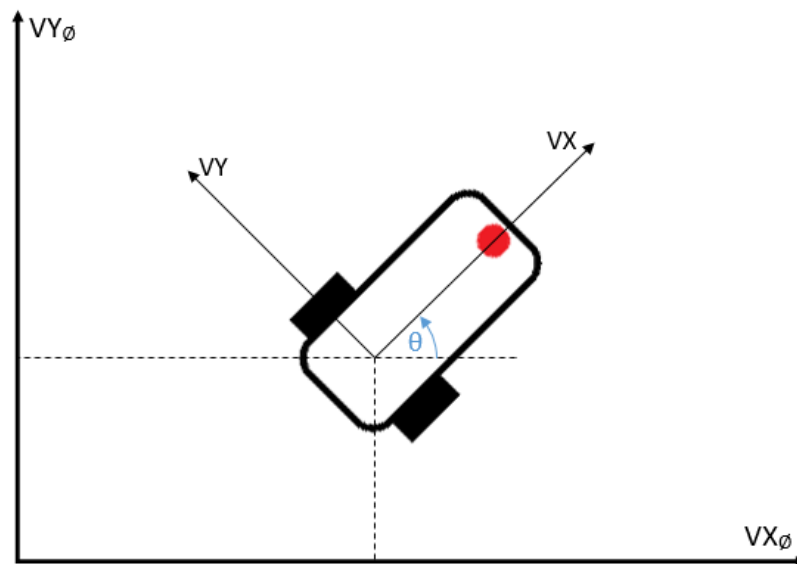


Figura 27. Representación del plano del robot en el plano del entorno.

Para representar dichas velocidades en el plano de un entorno, se multiplica la matriz 11 con una matriz de rotación sobre el eje Z representada en la siguiente matriz:

$$R_z(\beta) = \begin{bmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (12)$$

Donde β es el ángulo de rotación con respecto al plano del entorno.

Por lo tanto la matriz de velocidades con respecto al plano de un entorno es igual al producto de la matriz de rotación (12) con la matriz de velocidades con respecto al robot (11).

$$\begin{bmatrix} V_{x\emptyset} \\ V_{y\emptyset} \\ \omega_{\theta\emptyset} \end{bmatrix} = \begin{bmatrix} \cos \beta & -\sin \beta & 0 \\ \sin \beta & \cos \beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 \\ \frac{1}{2L} & \frac{1}{2L} & 0 \end{bmatrix} \cdot \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \omega_d \\ \omega_i \\ 0 \end{bmatrix} \quad (13)$$

Resolviendo esta matriz obtenemos que la matriz de velocidades es igual a:

$$\begin{bmatrix} V_{x\emptyset} \\ V_{y\emptyset} \\ \omega_{\theta\emptyset} \end{bmatrix} = \begin{bmatrix} \frac{r}{2} \cos \emptyset & \frac{r}{2} \cos \emptyset \\ \frac{r}{2} \sin \emptyset & \frac{r}{2} \sin \emptyset \\ \frac{r}{2L} & -\frac{r}{2L} \end{bmatrix} \cdot \begin{bmatrix} \omega_d \\ \omega_i \end{bmatrix} \quad (14)$$

Con la ecuación resultante se elabora el sistema de control de velocidad de las ruedas donde $V_{x\emptyset}$, $V_{y\emptyset}$, y $\omega_{\theta\emptyset}$ son las componentes de velocidad en los ejes del plano del entorno, siendo \emptyset el ángulo de rotación del plano del robot con respecto al plano del entorno, r el radio de las ruedas del robot, $2L$ la distancia entre ruedas del robot; y ω_d , ω_i las velocidades angulares en las ruedas derecha e izquierda correspondientemente. (Molina Villa & Rodríguez Vásquez, 2014)

3.4 Actuadores

La plataforma móvil elegida se comercializa con 2 motores DC conectados directamente del eje reductor a la rueda. Estos motores están conectados hacia un driver controlador de motores denominado L298N, el cual proporciona la potencia necesaria para su funcionamiento.

Además se incluyó en el sistema un micro servo para realizar un barrido circular de distancias y temperaturas con respecto al centro de cada robot. Este realiza un movimiento circular cada 30 grados hasta completar la circunferencia.

3.4.1 Motores DC

Los motores de corriente continua utilizados, son los encargados de generar el movimiento en la plataforma proporcionando el torque y velocidad adecuada para que cada robot pueda desplazarse por un espacio específico.

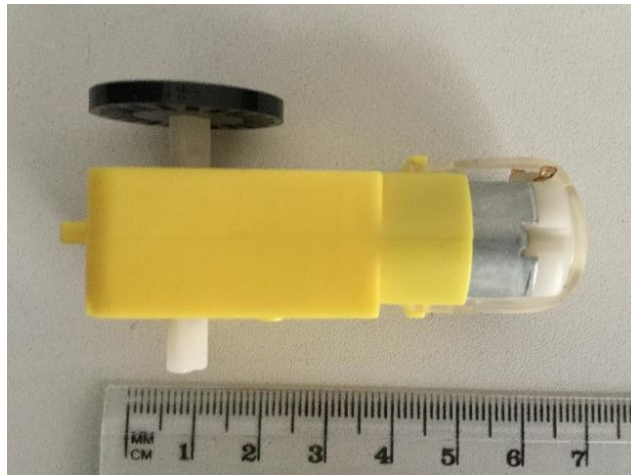


Figura 28. Motor DC con caja reductora.

Los motores miden 6.5cm de largo y 1.8x2.2cm de grosor. Otra característica adicional es que tienen un eje biaxial, es decir que se puede adaptar un elemento en cada lado del motor. En este caso en un eje se coloca la rueda y en el otro extremo un disco codificado para la lectura de velocidad con un encoder como se aprecia en la **Figura 28**.

Las principales características del motor son (“Kit Chasis para Carro Robot 2WD con encoders - Electronilab,” n.d.):

- Modelo: TGP01D-A10-12215-48
- Motor DC de imán permanente y escobillas de carbón
- Caja reductora 1:48 con piñonera plástica
- Eje biaxial
- Rango del voltaje de operación: 3V – 9V
- Voltaje de operación nominal: 3V
- Velocidad sin carga a 3V: 110rpm
- Corriente sin carga a 3V: 120mA
- Corriente de bloqueo a 3V: 450mA
- Torque de bloqueo: 0.27 kgf*cm (“Plataforma robot movil 2WD,” n.d.)
- Peso Motor: 50g
- Tamaño del motor: 65mm*22mm*18mm
- Ruido: <65dB

3.4.1.1 Driver L298N

Los motores se controlan mediante un driver especial que adaptada la etapa de potencia al controlador de velocidad y dirección del motor.

Este módulo es un driver para el control de 2 motores DC o un motor a pasos bipolar. Recibe una onda PWM de control mediante él envío de señales TTL, que en este caso se utilizan para controlar la velocidad y giro de los motores de corriente continua.

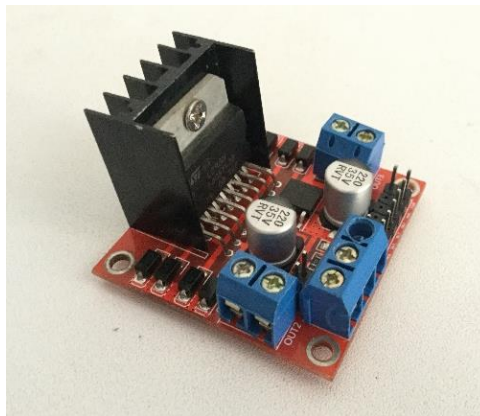


Figura 29. Driver de motores L298N.

Algunas de las principales ventajas de utilizar este driver es el bajo voltaje de saturación en los transistores de salida, el corte automático de operación por sobrecalentamiento y su buena respuesta frente al ruido (ver **Figura 29**).

Las principales características de este driver son las siguientes (STMicroelectronics, 2000):

- Circuito integrado: L298N
- Corriente pico de operación: 4A
- Corriente constante máxima de operación: 2A
- Voltaje de alimentación de motores: 5-46V
- Posee un regulador de voltaje LM7805 integrado.
- Posee 6 entradas de control incluidos dos habilitadores de control.
- Entradas de alimentación independientes para potencia y parte lógica.
- Dimensiones: 43mm*23.9mm*43mm

3.4.2 Micro Servo SG90

Es un servo motor miniatura de bajo coste con un conector universal tipo "S". Este micro servo fue instalado para permitir el giro 180 grados de la torre donde están

ubicados los sensores de temperatura y de distancia. De tal manera que su movimiento permita un escaneo 360 grados del entorno cercano al robot (ver **Figura 30**).



Figura 30. Micro servo SG90.

Las principales características de este micro servo son las siguientes (“SG90 9 g Micro Servo,” n.d.):

- Marca: Tower-pro
- Velocidad: 0.10 sec/60° a 4.8V
- Torque: 1.8 Kg-cm a 4.8V
- Voltaje de funcionamiento: 3.0 - 7.2V
- Temperatura de funcionamiento: 0 ~ 55°C
- Angulo de rotación: 180°
- Ancho de pulso: 500 – 2400 us
- Peso: 9g
- Dimensiones: 22.2mm*11.8mm*31mm

3.5 Sensores

Los sensores como entradas del sistema multi robótico cooperativo, son los responsables de adquirir la información del entorno y de su propio estado. En este trabajo se utiliza sensores ultrasónicos como detectores de obstáculos, encoders para el cálculo odométrico, brújulas electrónicas para complementar el cálculo de orientación y sensores de temperatura para analizar los niveles de calor en un entorno. La información proporcionada por estos sensores debe ser tratada adecuadamente y es fundamental para el cumplimiento de la tarea de exploración.

3.5.1 Sensor ultrasónico HC-SR04

Los sensores ultrasónicos utilizan sonares para determinar la distancia de un objeto, midiendo el tiempo que se demora una señal ultrasónica en transmitir y regresar al sonar receptor (ver **Figura 31**). Estos sensores son capaces de detectar obstáculos

hasta 4 metros de distancia sin contacto directo. Una de las principales ventajas es que no necesita de ningún tipo de iluminación para sensor correctamente. (Cytron Technologies Sdn. Bhd., 2013)



Figura 31. Sensor ultrasónico HC-SR04

Las especificaciones de funcionamiento de este sensor son las siguientes:

- Voltaje de trabajo: 5V
- Corriente de trabajo: 15mA
- Frecuencia de trabajo: 40KHz
- Rango de detección: 2cm - 4m
- Rango mínimo de detección: 2cm
- Angulo de medición: 30°
- Ancho de pulso de trigger: 10us
- Dimensiones: 45mm*20mm*15mm

Cada plataforma robótica está compuesta por 3 sensores ultrasónicos, uno estático en el frente del robot para evitar que choque contra algún obstáculo, y 2 sensores colocados en una torreta giratoria, encargados de realizar una medición 360 grados de obstáculos del entorno cercano al agente.

Para el presente proyecto se restringe la distancia válida de trabajo del sensor debido a que mientras el obstáculo se encuentre más lejos, es más difícil distinguir su verdadera posición. Por lo tanto se considera una distancia apropiada 35cm considerando el tamaño del agente, el ángulo de medición del sensor y el tamaño de la habitación a explorar.

3.5.2 Brújula electrónica GY-271

Este módulo es un sensor compacto diseñado para la detección de campos magnéticos y se utiliza principalmente para proporcionar información de la dirección respecto del norte magnético de la tierra (ver **Figura 32**).

La adquisición de información es mediante la conversión de cualquier campo magnético en salidas diferenciales de voltaje en 3 ejes (x, y, z), este cambio de voltajes es la salida digital sin procesar que proporciona el modulo. (“GY-271 ELECTRONIC COMPASS,” n.d.)

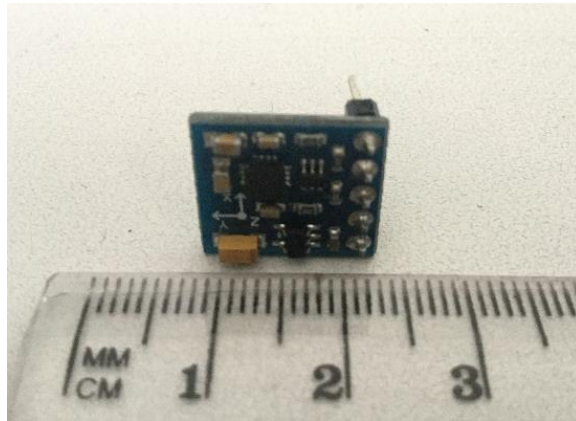


Figura 32. Brújula electrónica GY-271.

Para esta investigación se utiliza solamente la información proporcionada por los ejes X y Y, debido a que se trabaja únicamente sobre una superficie plana. Además dada la alta sensibilidad electromagnética del sensor, es necesario realizar una calibración en cada ambiente de pruebas para disminuir el error producido por interferencias en el ambiente.

Las especificaciones de este sensor son las siguientes:

- Voltaje de trabajo: 3V – 5V
- Circuito integrado: HMC5883L
- Comunicación: protocolo I2C
- Rango de medición: $\pm 1.3 - 8$ Gauss
- Dimensiones: 14.8mm*13.5mm*3.5mm

3.5.3 Sensor de temperatura DS18B20

Este integrado es un termómetro digital que proporciona de 9 a 12 bits para mediciones de temperatura en grados Celsius. Este sensor se comunica mediante un bus 1-Wire, que por definición solo requiere de una línea de datos para la comunicación con un microprocesador (ver **Figura 33**). (Maxim Integrated Products, n.d.)

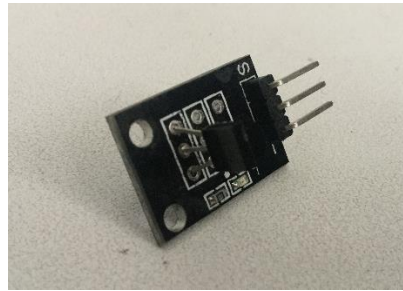


Figura 33. Sensor de temperatura DS18B20.

Para el presente trabajo se usó dos sensores de temperatura en cada robot móvil. Colocados en una torreta móvil en distinta dirección y a la mayor distancia posible uno del otro. Esto con el fin de precisar la temperatura en un punto dentro del entorno explorado.

Las principales características de este sensor son las siguientes:

- Voltaje de trabajo: 3V – 5.5V
- Corriente de trabajo: 1mA
- Tiempo de respuesta: 93.75ms – 750ms
- Rango de medición: -55°C - 125°C
- Precisión: $\pm 0.5^{\circ}\text{C}$

3.5.4 Encoder tipo optointerruptor FC-03

Este módulo se conforma de un emisor y receptor infrarrojo, el cual permite detectar si existe un objeto entre estos 2 elementos. Su arquitectura está diseñada para que se utilice junto a un disco codificado que sirve para medir la velocidad angular o distancia recorrida de una rueda (ver **Figura 34**).

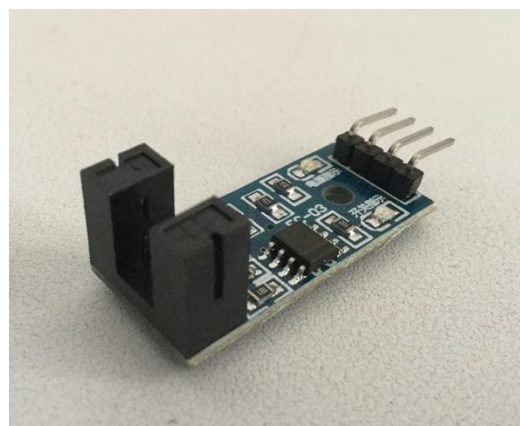


Figura 34. Encoder FC-03.

Los robots móviles fueron equipados con un encoder en cada rueda controlable para realizar una estimación de la distancia recorrida de cada agente con un disco de 20 ranuras. Es decir que por cada revolución se cuenta 40 pulsos por los flancos ascendentes y descendentes y por lo tanto la resolución máxima de giro de cada rueda es de 9° . De igual manera, dada la sensibilidad del sensor se detectó falsas lecturas en el conteo de pulsos, por esta razón se implementó de forma digital un algoritmo anti rebote.

Para este fin, se decidió separar de la tarjeta principal de control la etapa de medición del recorrido basada en estos sensores, para lograr mayor resolución en las mediciones realizadas y por tanto disminuir el error en la estimación odométrica.

Las principales características de este sensor son:

- Voltaje de trabajo: 3.3V – 5V
- Corriente de trabajo: 100nA
- Forma de salida: digital y analógica
- Dimensiones: 38mm*14mm*7mm
- Ancho de ranura: 5mm

3.6 Tarjeta programable de control

La tarjeta programable de control es la encargada de adquirir la información proporcionada por los sensores y procesarla con el fin de tomar decisiones sobre el sistema. En cada plataforma robótica móvil se decidió implementar dos tarjetas de programación. Una de propósito general para integrar y procesar la información sensorial; y otra para la estimación de recorrido mediante los sensores encoders.

Se dispuso de una tarjeta Arduino Mega 2560 como el controlador de propósito general en cada miembro del sistema multi robótico. Considerando sus capacidades mencionadas en el siguiente apartado. Y para la estimación de recorrido se utilizó una tarjeta Arduino Nano. Igualmente para la elección de las tarjetas programables se consideró el costo, confiabilidad y la posibilidad de expandir su uso a mayor escala electrónica.

3.6.1 Arduino Mega 2560

La tarjeta Arduino Mega 2560 es una plataforma open-hardware basada en el microcontrolador ATmega2560 programada mediante su propio IDE de código abierto. Esta placa electrónica cuenta con 54 pines digitales, 14 de ellos se pueden usar

como salidas PWM, 16 entradas analógicas y 4 puertos series (UART) de hardware (ver **Figura 35**). (Arduino.cl, n.d.-a)

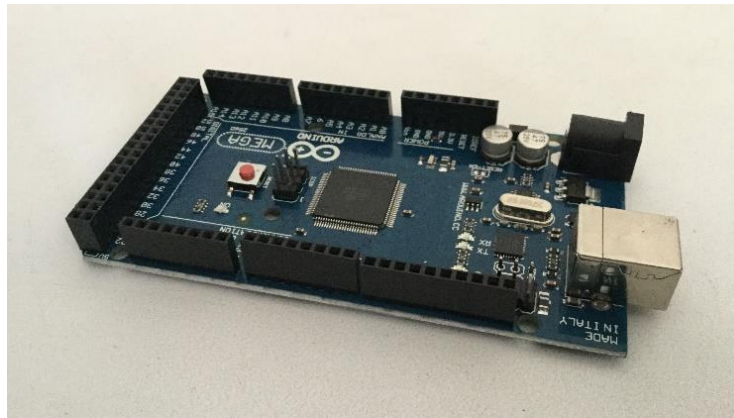


Figura 35. Tarjeta de programación Arduino Mega 2560.

Las principales características de esta tarjeta son:

- Microcontrolador: ATmega2560
- Voltaje de entrada: 7V - 12V
- Corriente DC por pin I/O: 20mA
- Corriente DC para pin de 3.3V: 50mA
- Memoria Flash: 256KB
- SRAM: 8KB
- EEPROM: 4KB
- Velocidad de reloj: 16 MHz
- Dimensiones: 101.52mm*53.3mm
- Peso: 37g

Cada robot móvil cuenta con un Arduino Mega 2560 para adquirir y almacenar la información obtenida por los sensores y enviarla al controlador central. También se encarga de controlar el recorrido en la etapa de navegación y de interpretar las órdenes de dirección enviadas inalámbricamente por el controlador central.

3.6.2 Arduino Nano

La tarjeta Arduino Nano es una plataforma open-hardware basada en el microcontrolador ATmega328 y programada al igual que el Arduino Mega, mediante su propio IDE. Esta placa electrónica cuenta con 22 pines digitales, 6 de ellos se pueden usar como salidas PWM, 8 entradas analógicas y un puertos serie (UART) de hardware (ver **Figura 36**). (Arduino.cl, n.d.-b)

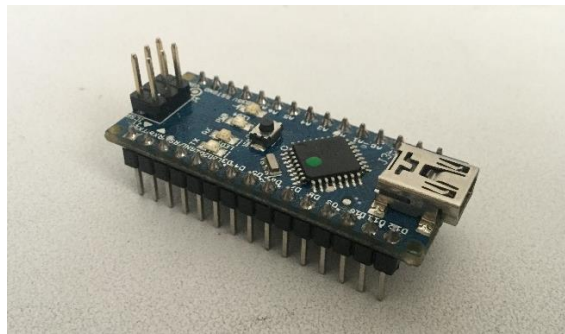


Figura 36. Tarjeta Arduino Nano.

Las principales características de esta tarjeta son:

- Microcontrolador: ATmega328
- Arquitectura: AVR
- Voltaje de entrada: 7V - 12V
- Corriente DC por pin I/O: 40mA
- Corriente de consumo: 19mA
- Memoria Flash: 32KB
- SRAM: 2KB
- EEPROM: 1KB
- Velocidad de reloj: 16 MHz
- Dimensiones: 18mm*45mm
- Peso: 7g

Cada robot móvil cuenta con un Arduino Nano para adquirir los pulsos proporcionados por los encoders mediante sus 2 pines para interrupciones externas (2, 3) y procesarlos para calcular la distancia recorrida y el ángulo de desviación. Una vez calculada esta información es almacenada hasta la espera de una petición del controlador de propósito general, en este caso un Arduino Mega 2560. La comunicación entre estos procesadores es serial con una velocidad de 115200 baudios.

3.7 Sistemas de comunicaciones

La comunicación en el medio interno como externo es fundamental para el correcto funcionamiento de un sistema multi robótico cooperativo. Este sistema de comunicación, está relacionado con el tipo de sistema de control elegido para realizar la tarea de exploración, en este caso un control centralizado donde los agentes actúan de acuerdo a las órdenes de un controlador central. Por tal razón la comunicación debe ser bilateral entre agente y controlador central, sin necesidad de existir una comunicación directa entre robots.

Además por tratarse de una tarea de exploración es necesario que el controlador central se comunique por un medio inalámbrico. Las posibles soluciones consideradas fueron Bluetooth, Wi-Fi y ZigBee. En la tabla 2 se puede comparar entre estas tecnologías.

Tabla 2.
Comparativa entre tecnologías inalámbricas (Björn Jürgens (IDEA) et al., 2008).

	Bluetooth	Wi-Fi	ZigBee
Frecuencia de operación	2.4GHz	2.4GHz 5GHz	2.4GHz 915MHz 868MHz
Max. Tasa de transferencia	~ 1Mbps	~ 600Mbps	250Kbps
Max. Nodos	8	N/A	65536
Corriente promedio de consumo	Tx: 15-20mA Rx: 15-20mA	Tx: 220mA Rx: 215mA	Tx: 25-35mA Rx: 20-30mA
Costo del modulo	~ 8 USD	~ 6 USD	~ 40 USD
Interoperabilidad	Media	Alta	Alta
Confiabilidad	Media	Media	Baja
Max. Rango de alcance	~ 50m	~ 100m	~ 500m

Como se puede apreciar el mejor equilibrio entre coste, rango de alcance y tasa de transferencia ofrece la tecnología Wi-Fi. Por esta razón se eligió este medio para la comunicación inalámbrica del sistema.

Otras razones por las que se consideró apropiado usar la tecnología Wi-Fi fue por la gran capacidad de agentes que se pueden integrar con este medio en una sola red y su tasa de transferencia muy alta la cual puede ser de gran utilidad en proyectos futuros donde se incluya por ejemplo visión artificial.

Por otra parte, la comunicación de los módulos y sensores internos, existentes en un agente robótico, dependen de los protocolos de comunicación en los componentes utilizados. En la siguiente tabla se describe estos módulos y sus protocolo de comunicación usados.

Tabla 3.
Comunicación interna de un robot.

Modulo	Bus	Protocolo	Pines Arduino Mega 2560
Wi-Fi	Serial	UART	18, 19 (Tx1, Rx1)
GY-271	Serial	I2C	20, 21 (SDA, SCL)
DS18B20	Serial	1-Wire	50, 51 (para 2 sensores)
Arduino Nano	Serial	UART	16, 17 (Tx2, Rx2)

3.7.1 Comunicación Wi-Fi

Wi-Fi es una tecnología de comunicación inalámbrica basada en el estándar IEEE 802.11b de aparatos electrónicos para redes de área local que trabaja bajo el protocolo de comunicación TCP/IP.

Esta tecnología fue implementada en cada agente para la comunicación con el controlador central que en este caso es un computador portátil como se muestra en el esquema de la **Figura 37** mediante un router que se encarga del enlace con los módulos Wi-Fi. Esta comunicación es semi-duplex y unicast, esto significa que cuando un robot tiene información para enviar al controlador central, la envía únicamente a él y una vez recibida la información el controlador envía la orden dirigida solamente hacia un agente. Esto con el fin de evitar pérdidas de información o recepción de paquetes incompletos.

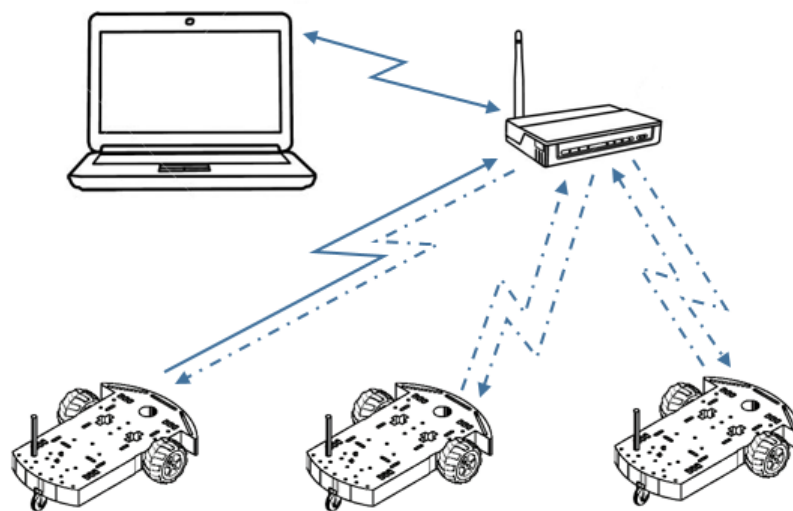


Figura 37. Esquema de red inalámbrica utilizado.

Para este propósito se probó varios modelos de módulos Wi-Fi como posibles soluciones para cada agente considerando la conectividad, confiabilidad y alcance. Dando como resultado que el modelo de mejor funcionamiento fue el módulo Wi-Fi Esp8266 de segunda generación.

3.7.1.1 Módulo Wi-Fi Esp8266

El módulo Wi-Fi Esp8266 de segunda generación usado es un circuito integrado compacto que incluye el procesador Esp8266 y la antena para amplificar la señal del mismo (ver **Figura 38**). La comunicación con el controlador es mediante comandos AT enviados por una puerta serial UART configurable a distintas velocidades. Una vez programado para conectarse a una red Wi-Fi, el módulo es capaz de almacenar esta información en su memoria flash integrada de 1MB. Igualmente, puede enviar información a una dirección IP y por el puerto deseado creando un enlace para dicha dirección. La recepción de información es automáticamente interpretada por el módulo y enviada por su puerto serial libre de todo el empaquetado TCP/IP usado por lo que disminuye el uso de procesamiento y de memoria. (“ESPRESSIF SMART CONNECTIVITY PLATFORM: ESP8266,” 2013)

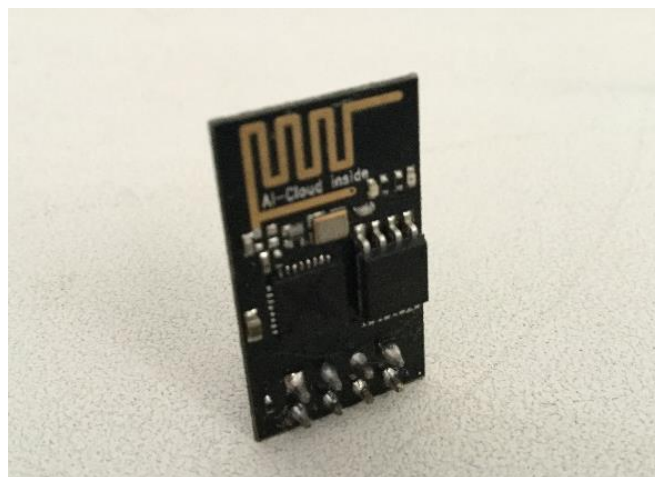


Figura 38. Módulo Wi-Fi Esp8266.

Las principales características de este módulo son las siguientes:

- Voltaje de trabajo: 3.3V
- Estandar soportado: 802.11 b/g/n
- Protocolo stack TCP/IP integrado
- Potencia de salida: +19.5dBm
- Consumo de corriente: 10uA - 200mA

- Wi-Fi 2.4GHz, con soporte WPA/WPA2
- Temperatura de operación: -40°C ~ 125°C
- Tiempo de encendido y transferencia de paquetes: <2ms
- Dimensiones: 14mm*24mm

Cada agente cuenta con un módulo Esp8266 para la comunicación con el controlador central y su configuración se detalla a continuación.

3.7.1.2 Protocolo de comunicación

Para el desarrollo de este proyecto se tomó en cuenta varios protocolos y estándares utilizados para comunicación inalámbrica mediante tecnología Wi-Fi. Basado en el modelo OSI la capa física y de enlace de datos se manejó a través del estándar IEEE 802.11b donde se define las normas de funcionamiento para una red inalámbrica de área local. Este estándar define una velocidad máxima de 5.9 Mbps en TCP sobre la banda de 2.4 GHz. También se usa el método de control de acceso definido por el protocolo CSMA/CA para asegurar que el servidor se encuentre disponible al momento de enviar los datos.

La transferencia de información determinada en la capa de aplicación, usa el protocolo FTP (cliente servidor) bajo el estándar RFC 959. Este protocolo muestra una rápida respuesta pero no implementa ningún tipo de seguridad o cifrado. Para este caso en particular, se usa el tipo de transferencia ASCII para envío de texto o cadenas de caracteres.

Se usa el protocolo TCP en la capa de transporte del modelo OSI para la transmisión segura y bidireccional de datos según el estándar RFC 793. Este protocolo está orientado a la conexión, por lo que el servidor y el cliente deben anunciarse y aceptar una conexión para comenzar la transferencia de datos. En esta investigación se usa el puerto 80 como identificador del enlace. Finalmente, en la capa de red se usa el protocolo IPv4 para la transmisión de datos mediante un medio no orientado a conexión según el estándar RFC 791. (Björn Jürgens (IDEA) et al., 2008)

En la

Tabla 4 se resume el conjunto de protocolos y estándares usados.

Tabla 4.
Protocolos usados según el modelo OSI.

Aplicación	FTP
	Estándar: RFC 959
Transporte	TCP
	Estándar: RFC 793
Red	IP
	Estándar: RFC 791
Enlace	CSMA/CA
Físico	Estándar: IEEE 802.11b

3.7.1.3 Tipología y topología de la red

Según su tamaño y alcance la red establecida en este proyecto es de tipología WPAN para redes inalámbricas de corto alcance. Con un número total de 5 equipos y una cobertura menor a 10 metros. Además, se define CSMA como método de acceso, donde cada agente monitoriza la disponibilidad del controlador central para poder enviar el mensaje.

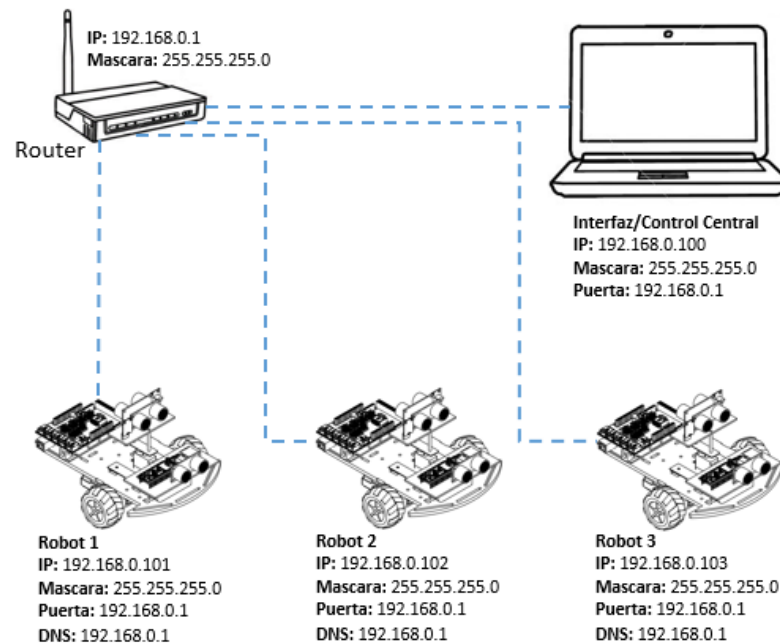


Figura 39. Diagrama de redes del sistema de comunicaciones implementado.

La topología de una red establece los enlaces usados para la transmisión de información. En este caso se utiliza una topología tipo estrella donde el concentrador es un router, encargado de distribuir la información entre los agentes robóticos y el controlador central. Adicionalmente, el router se encarga de asegurar el enlace para prevenir problemas de direccionamiento y enrutamiento. En la Figura 39 se observa el diagrama de redes del sistema de comunicación Wi-Fi con topología estrella y tipología WPAN implementado.

3.7.2 Comunicación UART

La comunicación serial UART (por sus siglas en inglés Universal asynchronous receiver/transmitter) es la unidad encargada de convertir la información a una secuencia de bits y transmitirlos o recibirlos a una velocidad determinada. Para ello se usan dos conectores, un receptor (Rx) y un transmisor (Tx).

Las placas Arduino Mega poseen 4 unidades UART que operan a nivel TTL compatibles con conexiones USB. Se utiliza 2 de los 4 puertos series disponibles en la tarjeta Arduino Mega 2560, los cuales se comunican con el procesador Arduino Nano encargado de medir el recorrido de las plataformas y el módulo de comunicación Wi-Fi Esp8266. El puerto serie 1 usado para la comunicación Wi-Fi, está configurado a

una velocidad de 115200 Bd y el puerto serie 2 conectado a un Arduino Nano a 115200 Bd. (“Tutoriales PIC: Comunicaciones puerto serie (UART),” n.d.)

3.7.3 Comunicación I2C

La comunicación I2C (de sus siglas en ingles Inter integrated circuits) es un protocolo diseñado para la comunicación entre circuitos integrados como microcontroladores, memorias, sensores, etc. El cual requiere de dos líneas de señal y permite el intercambio de datos entre varios dispositivos en un mismo bus a una velocidad aproximada de 100 Kbps. Su comunicación se realiza de forma serial y sincronizada, para ello una de las señales marca el tiempo (SCL) mientras que la segunda señal es por donde se transporta la información (SDA). (Carletti, n.d.)

En el caso para Arduino Mega 2560 el puerto I2C integrado está en los pines 20 y 21 y el único sensor integrado que trabaja bajo este protocolo es la brújula electrónica GY-271.

3.8 Fuente de alimentación

Para dimensionar la batería se consideró el consumo de corriente del sistema final, el voltaje de trabajo máximo necesario y el tiempo estimado que se demora en realizar la tarea. En la tabla 4 se muestra un resumen del consumo energético de todos los elementos electrónicos que se van a implementar en la plataforma robótica.

Tabla 5.
Resumen de consumo energético de un robot.

Dispositivo	Voltaje de trabajo	Corriente Max.
Arduino Mega 2560	7V - 12V	20mA por pin I/O
Arduino Nano	7V - 12V	40mA por pin I/O
Encoders x2	3.3V - 5V	100nA
Ultrasónicos x3	5V	15mA
Brújula Electrónica	3V – 5V	20mA
Módulo Wi-Fi	3.3V	<200mA
Sensores de temperatura x2	3V – 5.5V	1mA
Micro Servo	3V – 7.2V	<500mA
Motores DC x2	3V – 9V	450mA

Driver L298N	5V – 46V	4A
--------------	----------	----

Realizando la sumatoria de corrientes consumidas por los dispositivos en su máxima capacidad da como resulta un consumo total de 1927.2 mA. Se estima que el sistema funcione en un lapso de 30 minutos entonces la capacidad mínima necesaria de las baterías es igual al producto del consumo durante el tiempo establecido.

$$1927.2mA * 0.5h = 963.6mAh \quad (15)$$

Basándose en la tabla 5, podemos deducir que el voltaje mínimo necesario para que funcionen correctamente todos los elementos es de 7V, valor que entra en el rango de trabajo de todos los componentes electrónicos.

Por lo tanto la fuente de alimentación de cada robot elegida es una batería de polímero de iones de litio que entrega 7.4 V y tiene una capacidad de 1200mAh. Estas baterías cuentan con un conector de 2 pines JST-PH e incluye un circuito de protección que asegura que no existan picos de sobretensión al cargar la batería (ver Figura 40).



Figura 40. Batería utilizada para cada robot.

Según los cálculos antes mencionados, estas baterías ofrecen 36 minutos de autonomía en una tarea constante donde todas sus funciones estén trabajando al mismo

tiempo. En el capítulo 5 se observa el tiempo real de autonomía con el algoritmo desarrollado en este trabajo.

3.9 Diseño de placas de circuito impreso adicionales

Aparte de los elementos electrónicos necesarios descritos anteriormente, se necesita adaptarlos, conectarlos e integrarlos a la plataforma de forma funcional. Para este propósito se diseñó dos placas de circuito impreso que se adaptan a la forma y posición de los módulos implementados.

Una de estas placas fue diseñada con el fin de montar una torreta en donde se disponga de 2 sensores ultrasónicos con direcciones inversas y dos sensores de temperatura dispuestos en los extremos de la placa base como se muestra en la Figura 41. Otra consideración para esta placa es que debe tener un solo bus de datos y de alimentación, de acuerdo al movimiento a realizar del micro servo en su base.

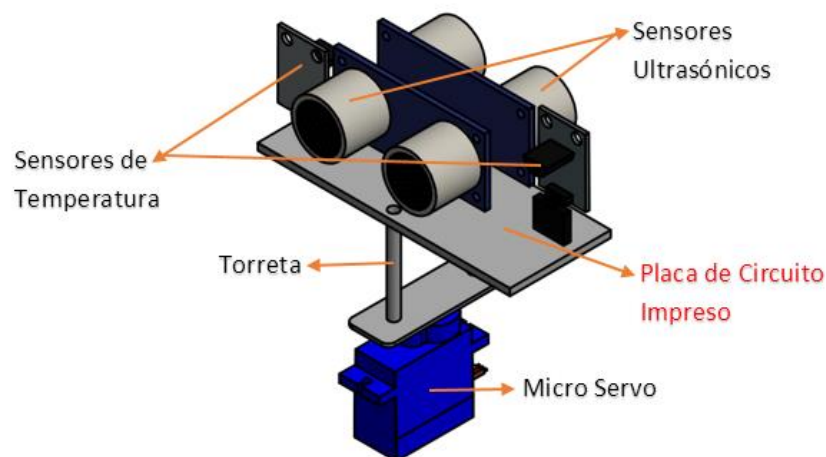


Figura 41. Posición de sensores en la placa de circuito impreso (1).

Como resultado se implementó el circuito impreso mostrado en la **Figura 42**.

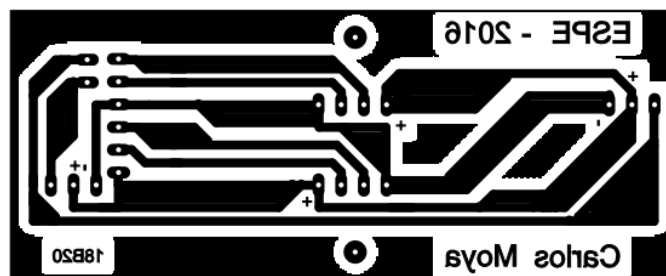


Figura 42. Circuito impreso de placa de sensores (1).

Se diseñó adicionalmente una segunda placa para distribuir sobre la plataforma los elementos electrónicos usados. Incluidos un ultrasónico en el frente del robot móvil, una brújula con sus ejes paralelos a los ejes del robot y finalmente para la conexión del Wi-Fi con el Arduino Mega 2560 y la conexión del Arduino nano con los dos encoders utilizados (ver **Figura 43**).

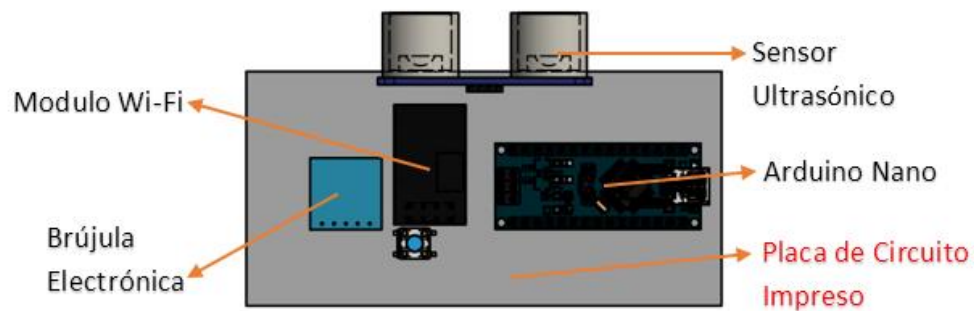


Figura 43. Posición de dispositivos en placa de circuito impreso (2).

Como resultado se implementó el circuito impreso mostrado en la **Figura 44**.

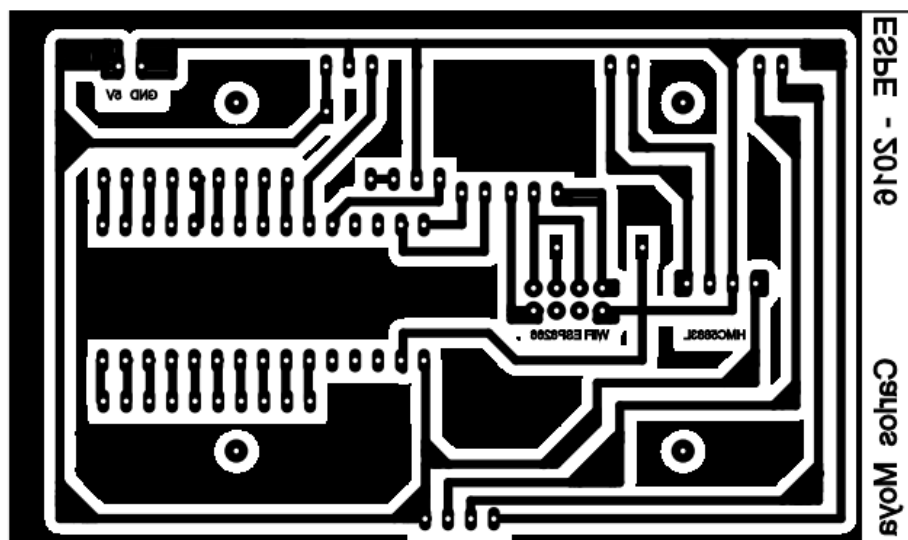



Figura 44. Circuito impreso de placa de dispositivos (2).

3.10 Conexión y montaje de la plataforma robótica móvil

Después de analizar y obtener los componentes necesarios para el cumplimiento de los objetivos de este proyecto, se desarrolla la etapa de conexión de los elementos en el adecuado orden lógico para su correcto funcionamiento.

En primer lugar se describe en la **Tabla 6** los pines utilizados del Arduino Mega 2560 y sus correspondientes conexiones.

Tabla 6.
Configuración de pines del Arduino Mega 2560.

Nombre de Pin Arduino Mega	Pin del dispositivo conectado
D02	Echo (Ultrasónico Frontal)
D03	Trigger (Ultrasónico Frontal)
D04	Trigger (Ultrasónico 2)
D05	Echo (Ultrasónico 2)
D06	Echo (Ultrasónico 1)
D07	Trigger (Ultrasónico 1)
D08	IN3 (L298N)
D09	IN4 (L298N)
D10	IN1 (L298N)
D11	IN2 (L298N) Continúa 
D12	S (Micro Servo SG90)
TX2 16	RX 0 (Arduino Nano)
RX2 17	TX 1 (Arduino Nano)
TX1 18	RX (Wi-Fi Esp8266)
RX1 19	TX (Wi-Fi Esp8266)
SDA 20	SDA (Brújula Electrónica)
SCL 21	SCL (Brújula Electrónica)
D50	S (Sensor de temperatura 1)
D51	S (Sensor de temperatura 2)

El orden que se muestra depende de la mejor ubicación física en el espacio disponible, considerando las piezas móviles.

De igual manera, se observa en la **Tabla 7** los pines utilizados del Arduino Nano y sus correspondientes conexiones.

Tabla 7.
Configuración de pines del Arduino Nano.

Nombre de Pin en Arduino Nano	Pin del elemento conectado
RX 0	TX2 16 (Arduino Mega 2560)

TX 1	RX2 17 (Arduino Mega 2560)
D02 (INT0)	A0 (Encoder Derecho)
D03 (INT1)	A0 (Encoder Izquierdo)

Además de las conexiones descritas, se necesita conectar la alimentación de los dispositivos. Todos los elementos son alimentados mediante el Arduino Mega 2560, excepto la salida para la alimentación de los motores del driver L298N que tiene alimentación directa de la batería.

Una vez comprendido las conexiones necesarias, se efectúa el montaje de los dispositivos en el orden que se describe a continuación.

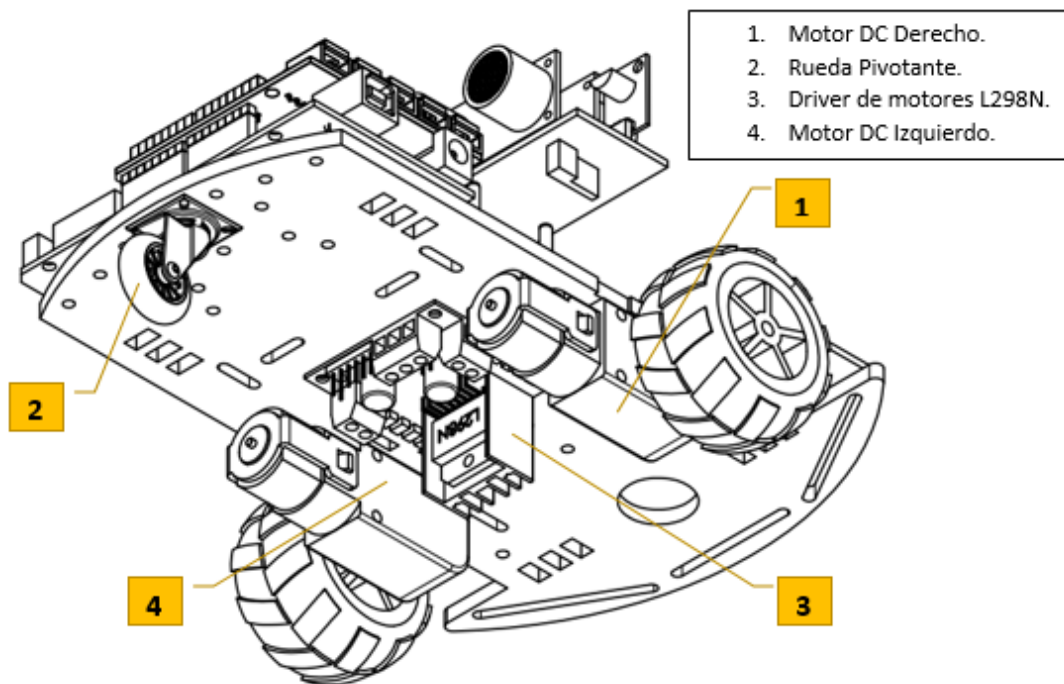


Figura 45. Montaje en la parte inferior de la plataforma robótica móvil.

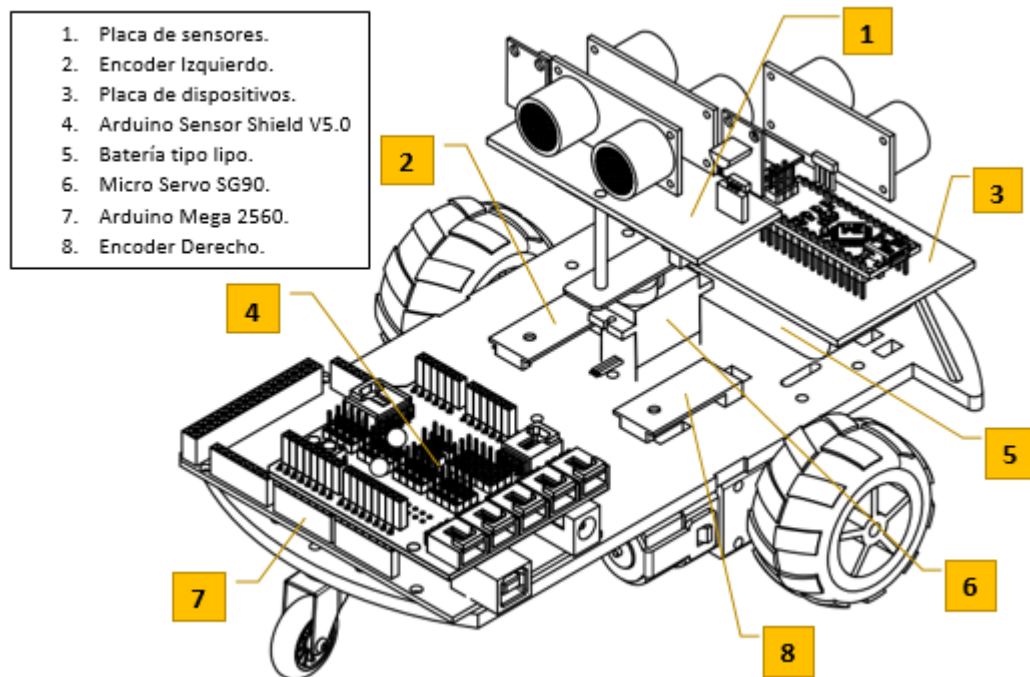


Figura 46. Montaje en la parte superior de la plataforma robótica móvil.

Primero se conecta los elementos en la parte inferior del robot según la **Figura 45**. Donde, los elementos 1,2 y 4 son instalados según las instrucciones del fabricante de la plataforma; y el Driver de motores (3) se monta en la posición observada con la conexión correspondiente de los motores.

Después, en la parte superior se realiza el siguiente montaje según la **Figura 46**:

- Primero se coloca el Arduino Mega (7) en la posición correspondiente y se lo ajusta mediante tornillos en la plataforma robótica.
- Se monta el Arduino Sensor Shield (4) en los pines correspondientes del Arduino Mega (7).
- Se coloca el Micro Servo (6) mediante una base diseñada en impresión 3D, para su ajuste mediante tornillos en la plataforma robótica.
- Se coloca la batería (5) en la posición adecuada y se ajusta a la plataforma con la placa de dispositivos (3) en su parte superior mediante tornillos.
- Se realiza las conexiones adecuadas de los encoders (2 y 8) con la placa de dispositivos, y se los coloca en la posición observada en la Figura 46.
- Finalmente se monta sobre el Micro servo (6) la torreta con la placa de sensores (1). Asegurándose de que su posición inicial este ajustada correctamente, de manera que en cero grados del micro servo se encuentren los ultrasónicos con las vistas hacia el frente y hacia atrás correspondientemente.

Los sensores y dispositivos sobre las placas (1 y 3) serán colocados al final, una vez realizadas todas las conexiones indicadas anteriormente. Su posición se observa mejor en las figuras 41 y 43.

Los resultados finales de la implementación de las plataformas robóticas móviles se muestran en la **Figura 47**.

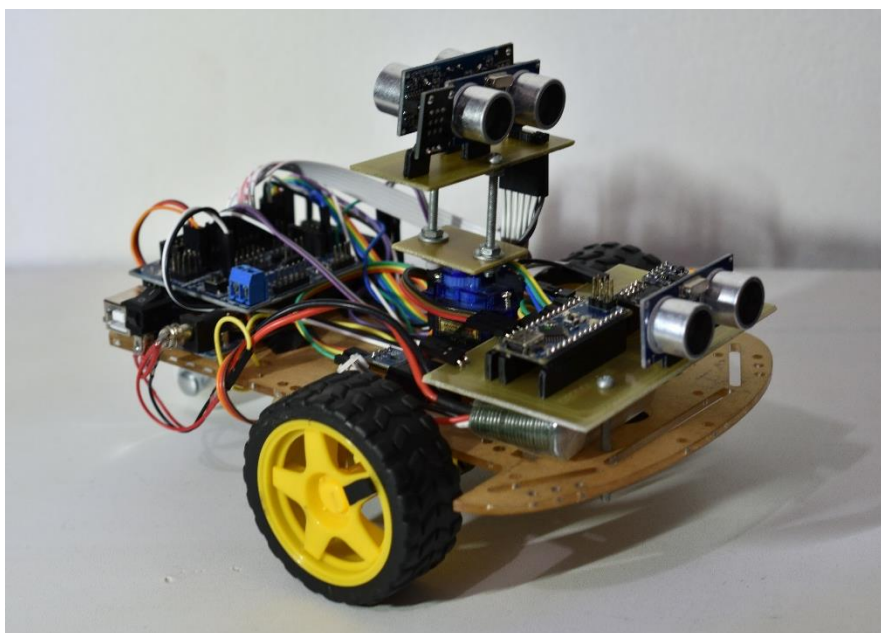


Figura 47. Implementación de una plataforma robótica móvil.

Por último, se presenta una tabla con los costos de implementación total del hardware del sistema multi robótico.

Tabla 8.
Costos finales de la implementación de las plataformas robóticas móviles.

No.	Detalle	Cantidad	Valor Unitario	Valor Total
1	Plataforma robótica móvil 2WD	3	\$ 21.07	\$ 63.21
2	Driver de motores L298N	3	\$ 7.50	\$ 22.50
3	Encoder	6	\$ 5.85	\$ 35.10
4	Micro servo SG90	3	\$ 5.00	\$ 15.00
5	Ultrasónico	9	\$ 3.99	\$ 35.91
6	Sensor de temperatura	6	\$ 2.70	\$ 16.20
7	Brújula electrónica	3	\$ 6.50	\$ 19.50

8	Pulsador	3	\$	1.00	\$	3.00
9	Módulo Wi-Fi	3	\$	6.00	\$	18.00
10	Arduino nano	3	\$	8.25	\$	24.75
11	Arduino mega 2560	3	\$	23.99	\$	71.97
12	Arduino sensor Shield v5.0	3	\$	8.50	\$	25.50
13	Placa rotativa	3	\$	5.00	\$	15.00
14	Placa frontal	3	\$	12.00	\$	36.00
15	Batería Recargable tipo lipo	3	\$	8.75	\$	26.25
16	Cables varios	3	\$	2.25	\$	6.75
17	tornillos, tuercas y arandelas	3	\$	2.00	\$	6.00
18	Impresión 3D	2	\$	6.25	\$	12.50
TOTAL					\$	453.14

El costo de cada plataforma robótica por separado es de 153.13 USD. Vale recalcar que este valor representa únicamente los costos de materiales utilizados y que los mismos pueden variar dependiendo de los proveedores de los dispositivos.

CAPÍTULO IV

DESARROLLO DE SOFTWARE PARA CONTROL Y VISUALIZACIÓN MRSLAM

4.1 Introducción

El presente capítulo desarrolla todos los aspectos relacionados al software, con el fin de establecer algoritmos de control y de visualización para el cumplimiento de la tarea de exploración asignada en este trabajo de investigación.

El presente apartado se divide en dos áreas principales: La interfaz de comunicación Hombre-Máquina y los algoritmos de control MRSLAM. Sobre la interfaz se detalla las características del medio visual diseñado y como se utiliza. Posteriormente, se describe el desarrollo de los algoritmos de control MRSLAM para proporcionar autonomía en la exploración cooperativa de las plataformas robóticas móviles.

4.2 Diseño de interfaz gráfica

Una de las necesidades básicas en el mapeado, es la posibilidad de poder interpretar gráficamente el estado de una zona geográfica. Para cumplir con este propósito se desarrolló una interfaz gráfica, que muestre los resultados de la exploración en tiempo real. Esta interfaz fue desarrollada bajo el lenguaje de programación Java y el entorno de desarrollo NetBeans 8.1 de código abierto.

La interfaz gráfica implementada permite la interacción entre el usuario y el sistema multi robótico con énfasis en la visualización del mapa creado. Las principales funciones que cumple dentro del sistema son: observar variables del proceso, el estado de los mapas creados y detener el proceso de mapeado.

Considerando que el principal objetivo de la interfaz gráfica es la visualización del mapa mostrado por la exploración de los agentes robóticos, se diseñó dos pantallas que facilitan la interpretación de la información para el usuario. Tomando en cuenta la guía GEDIS, la norma ISA SP-101, NTP 659 y NTP 241 para la distribución, uso de colores e información en la pantalla.

La ventana que se visualiza por defecto en primera instancia es la ventana de mapas, desde ella se puede dirigir hacia la ventana de históricos y viceversa. Estas dos pantallas muestran de forma sencilla los resultados de un mapeado robótico autónomo.

4.2.1 Descripción de funcionamiento

Mediante estas dos pantallas se obtiene acceso a la información adquirida por las plataformas robóticas de manera gráfica y numérica. Las pantallas están diseñadas para visualizar las variables, es decir son pantallas únicamente de monitoreo que permiten al usuario tener una percepción 2D del espacio explorado por los robots móviles. A continuación se detalla el funcionamiento de cada pantalla y sus características.

4.2.1.1 Pantalla de mapas

La Pantalla de Mapas, se caracteriza por dos grandes áreas que representan la zona a explorar desde una vista superior en 2D. En ellas, se representa el movimiento en tiempo real de cada plataforma robótica y la información adquirida en dicha estimación de posición mediante animaciones gráficas. Además de ser la pantalla principal de visualización del proceso de mapeo, es la encargada de distribuir la información entre las distintas clases y métodos de programación.

La distribución de esta pantalla es fundamental para el entendimiento del proceso de mapeo de los agentes robóticos móviles. Permiten al usuario conocer cómo se comporta el sistema y sobre ello desarrollar nuevos algoritmos MRSLAM.

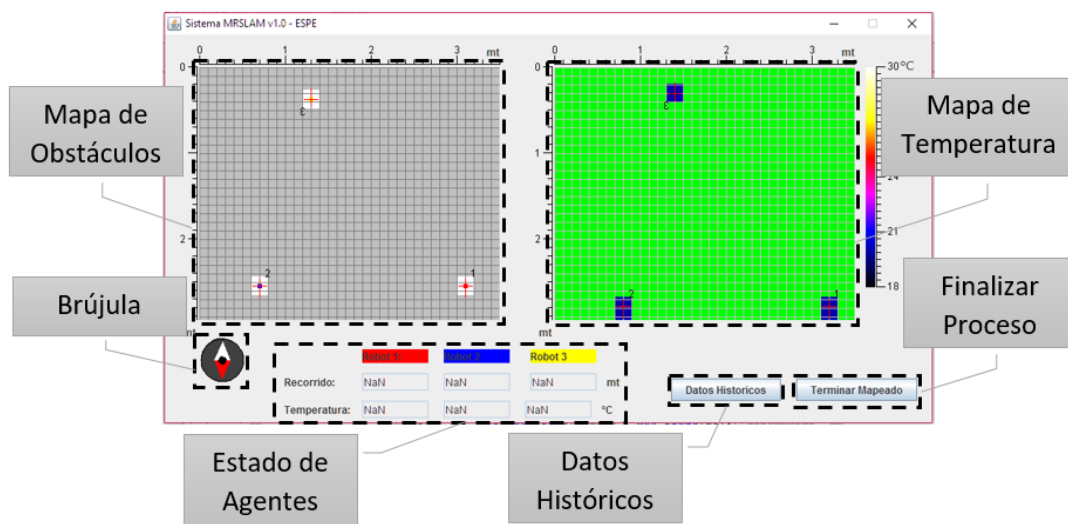


Figura 48. Distribución de componentes en ventana de mapas.

El gris en el mapa de la izquierda y el verde del mapa de temperaturas a la derecha de la **Figura 48**, representan el estado inicial desconocido de cada celda de los mapas. Por otra parte, en el mapa de obstáculos las celdas blancas simbolizan los espacios vacíos, las celdas negras los espacios ocupados y los puntos rojos, azules y amarillos representan las estimaciones de posición del robot 1, robot 2 y robot 3 correspondientemente.

Según la **Figura 48**, tenemos seis zonas principales en la ventana de mapas. Cada sección tiene las siguientes funciones:

Mapa de obstáculos, este espacio está dispuesto para la representación en 2D del mapa de obstáculos que se está creando en tiempo real mediante la estimación de posición de los miembros del grupo robótico usado. Por tal razón, está compuesto por un JFrame con animaciones para mostrar el movimiento del agente y el cambio generado en el mapa como respuesta de la exploración realizada.

Mapa de temperaturas, este panel fue desarrollado con la intención de representar en 2D un mapa paralelo al mapa de obstáculos, con la diferencia de que en él se crea un mapa de niveles de temperaturas captadas por cada agente en cada posición estimada. De igual forma, está compuesto por un JFrame con animaciones para mostrar el movimiento del agente y el cambio generado en el mapa como respuesta de la exploración realizada.

Brújula, es una figura animada que representa la orientación de los mapas con respecto al norte magnético de la tierra. Para el propósito de este trabajo, esta representación gráfica será estática, debido a que se desarrolló sobre un lugar específico. Sin embargo fue diseñada para su fácil uso en nuevas investigaciones donde sea oportuno obtener una dirección dinámica.

Estado de agentes, esta zona de textField muestra información exacta de forma numérica sobre el recorrido total de cada robot y la temperatura actualizada del mismo, con el propósito de evaluar su comportamiento.

Datos históricos, este botón cuya acción es interactuar con la ventana de históricos. La cual se detallara en el siguiente apartado.

Finalizar proceso, es un botón que detiene el proceso de mapeado sin cerrar la aplicación, asimismo realiza una estimación en ambos mapas en los puntos

desconocidos permitiendo evaluar el comportamiento y tomar la información obtenida para su análisis.

En la Figura 49 se muestra el diagrama de flujo simplificado se observa la lógica de programación del algoritmo desarrollado para cumplir estas funciones.

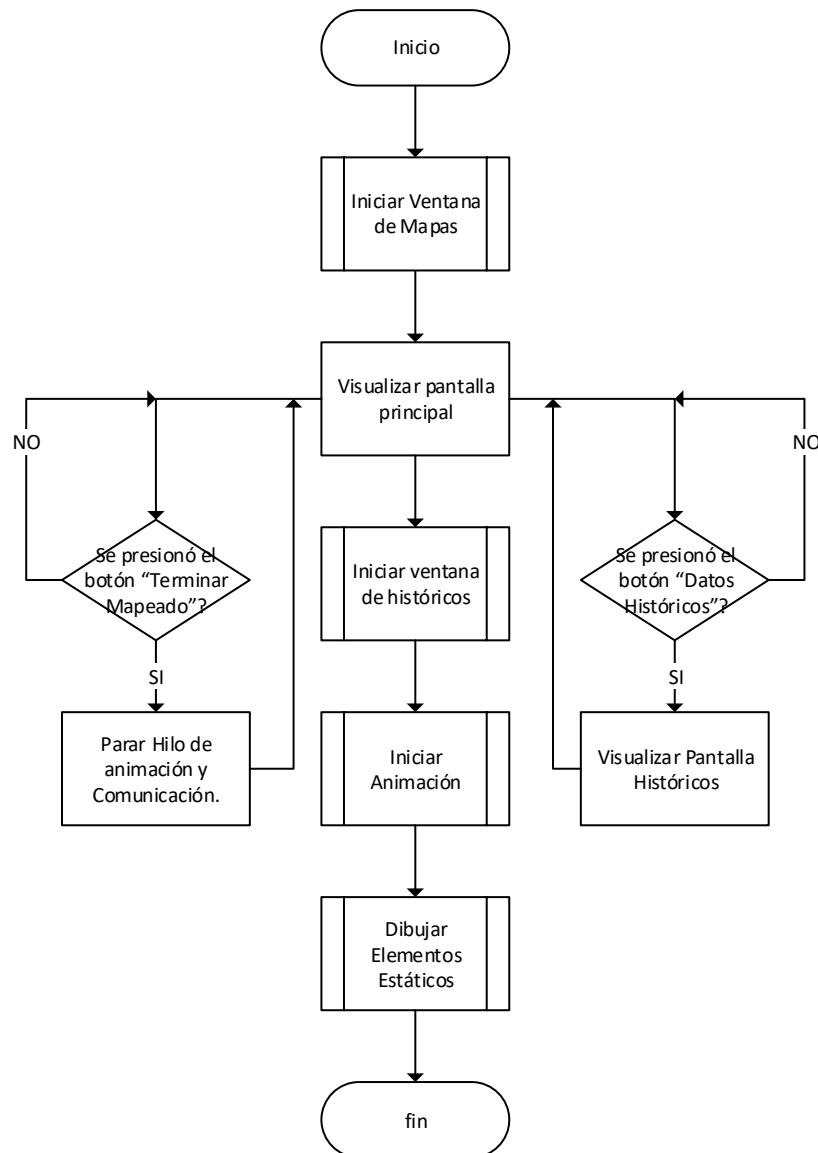


Figura 49. Diagrama de flujo del funcionamiento de ventana de mapas.

En el diagrama de flujo de la **Figura 49**, se muestra cuatro subprocesos que serán descritos a continuación:

Iniciar ventana de mapas, inicializa los componentes de la ventana entre ellos JFrames, TextField y botones. Cada uno con características propias de nombre, posición, tamaño, etc.

Iniciar ventana de históricos, inicializa los componentes de la ventana entre ellos JFrames, tablas y botones. Cada uno con características propias de nombre, posición, tamaño, etc.

Iniciar Animación, este subproceso se encarga de inicializar las componentes dinámicas de la ventana, entre ellas el mapa de obstáculos, el mapa de temperaturas, posición de los agentes, columnas en tablas dinámicas de la ventana de históricos, e información textual presentada. Aparte de inicializar las variables, debe mantener actualizados los valores registrados por los agentes y establecer la comunicación con ellos. Más adelante se detalla los métodos y clases desarrollados.

Dibujar elementos estáticos, inicializa las componentes de la ventana que permanecen en la misma posición y dimensión durante todo el mapeado. Entre ellas: reglas de escala de los mapas, brújula de orientación sobre el norte geográfico, escala de temperaturas del mapa térmico y etiquetas textuales dentro de la pantalla.

4.2.1.2 Pantalla de históricos

La pantalla de históricos, es una ventana sencilla diseñada para mostrar la información textual obtenida por la estimación de posición de cada robot en un instante determinado de tiempo.

Esta pantalla está constituida por tres tablas destinadas cada una a mostrar información relevante del agente en un instante de tiempo.

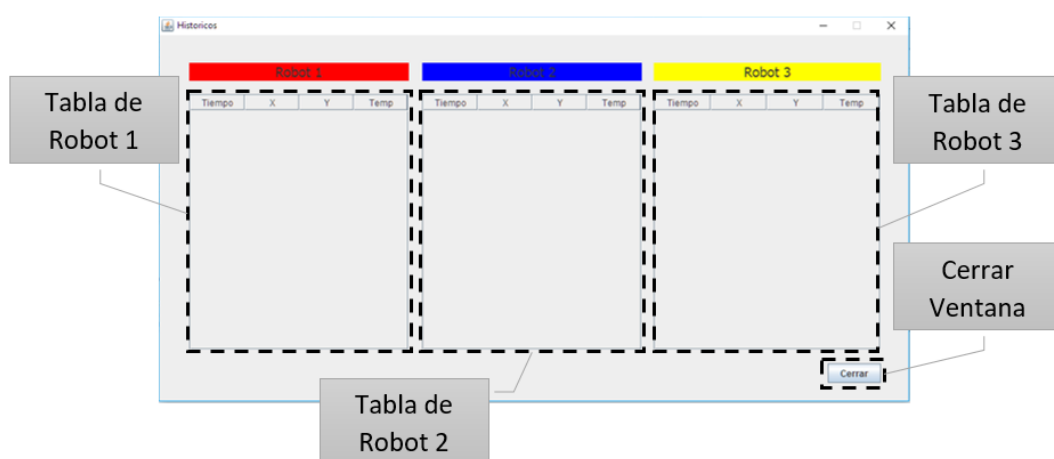


Figura 50. Distribución de componentes en ventana de históricos.

En la **Figura 50** se observa 3 grandes zonas destinadas para mostrar los valores numéricos de la posición (X, Y) y la temperatura (Temp.) en una fecha y tiempo

determinado de muestreo realizado por los robots móviles de manera online. En la parte inferior derecha se ubica un botón para cerrar esta ventana. Esta acción no para la toma de información, es decir si se vuelve a abrir mostrara nuevamente las tablas con el contenido anterior y los datos tomados mientras se mantuvo cerrada.

4.2.2 Clases y métodos desarrollados

Todas las clases creadas poseen un constructor para su inicialización con sus respectivos parámetros necesarios con excepción de la clase Robot. Las clases creadas para el correcto funcionamiento de la interfaz se detallan a continuación.

Tabla 9.
Clases usadas para la interfaz gráfica.

Clase	Descripción
Animacion(JPanel, JTextField)	Inicializa los componentes estáticos y hacer el llamado al hilo encargado de la actualización constante de la información mostrada en la pantalla.
AnimacionTemp(JPanel)	Esta clase inicializa las componentes estáticas y de trasladar la información obtenida mediante la clase ConexiónWiFi hacia el JPanel dedicado para el mapa de temperaturas.
ConexionWifi(Matriz de estado)	Crea el enlace Wi-Fi con los módulos Esp8266 y realizar la correcta interpretación de la información obtenida.
Dibujo(JPanel, JTextField)	En esta clase se crea el hilo para la animación en el mapa de obstáculos con la información obtenida por la clase ConexionWifi mediante la actualización de la imagen en la ventana principal.
Robot	Caracteriza el aspecto y escalar cada robot con el mapa creada con sus propias características.

Estas clases están constituidas por métodos que hacen posible el funcionamiento de la interfaz gráfica desarrollada. En la **Tabla 10** se detalla por clases los métodos que las constituyen y su propósito dentro de la interfaz gráfica desarrollada.

Tabla 10.
Descripción de métodos para la interfaz gráfica.

Animacion(JPanel, JTextField)	
dibujarBrujula(posición, gráfico y dirección del norte geográfico)	Crea los gráficos de representación de una brújula con dirección hacia el norte magnético de la tierra.
dibujarEscalaTemp(posición y gráfico)	Crea los gráficos de representación de la escala de temperaturas en el mapa creado.
dibujarRegla(posición y gráfico)	Crea los gráficos de representación de la escala en metros del espacio a explorar.
pararAnimacion()	Realiza un llamado a la función detener() de la clase dibujo.
AnimacionTemp(JPanel)	
dibujarTemp(gráfico y JPanel)	Crea el mapa de temperaturas conforme se actualizan los datos entregados mediante la clase conexionWiFi.
tempRobot(posición e ID de robot)	Actualiza la temperatura en el lugar estimado por el agente robótico.
dibujarRejilla(grafico)	Crea una rejilla sobre el mapa como guía de escala del mapa de temperaturas.
ConexionWifi(Matriz de estado)	
initServer(matriz de estados)	Inicia un enlace TCP/IP para recibir la información proporcionada por los agentes e interpretarla.
setDatos(mensaje y matriz de estados)	Interpreta la información obtenida en un mensaje recibido por algún agente robótico.
String[] separarFrase(mensaje)	Separa el mensaje recibido por el agente en diferentes cadenas de caracteres para su posterior interpretación.
Dibujo(JPanel, JTextField)	
Animar()	Inicializa un hilo encargado de realizar la animación y abrir el enlace de comunicación.
Detener()	Detiene el hilo de animación y de comunicación.

Continua 

dibujar(gráfico y JPanel)	Crea el mapa de obstáculos conforme se actualizan los datos entregados mediante la clase <code>conexionWiFi</code> .
ocupadoRobot(ID de robot y posición)	Gráfica el espacio libre como efecto de la estimación de posición de cada agente.
trazarRuta(posición inicial y posición actual)	Crea un rastro en el mapa de obstáculos para conocer la ruta elegida por cada agente robótico.
resolucion(posición y tamaño de celda)	Este método puede definir cuantos pixeles se pueden interpretar como una celda de la malla de estados.
areaLibre(ID de robot)	Mediante este método se interpreta gráficamente las distancias captadas en el entorno por los sensores ultrasónicos de cada agente robótico.
dibujarRejilla(gráfico)	Crea una rejilla sobre el mapa como guía de escala y posición del mapa de obstáculos.
Robot	
dibujarRobot(posición, gráfico, dirección e ID del robot)	Crea una interpretación gráfica para mostrar la posición y dirección estimada de cada robot móvil.
estadoMalla(posición, gráfico, estado de celda)	Actualiza una celda de la matriz de estado del mapa de obstáculos de acuerdo a su estado. Los estados pueden ser ocupados, vacío o desconocido.
tempMalla(posición, gráfico, temperatura de celda)	Actualiza una celda de la matriz de estado del mapa de temperaturas de acuerdo a su estado. Su estado varía de acuerdo a la escala elegida en el código de colores térmicos creada.

Adicionalmente se crearon métodos *Set* y *Get* para el llamado de datos privados entre clases.

El resultado final de las pantallas diseñadas tiene el formato mostrado en la **Figura 51** y **Figura 52**.

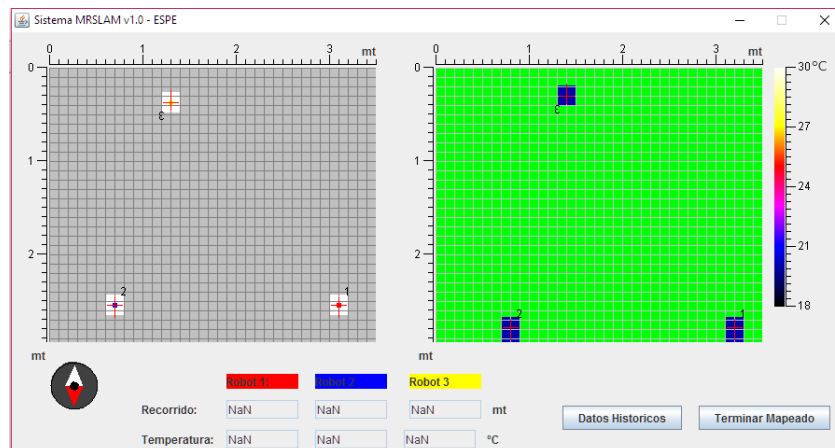


Figura 51. Pantalla de Mapas Desarrollada para este trabajo.

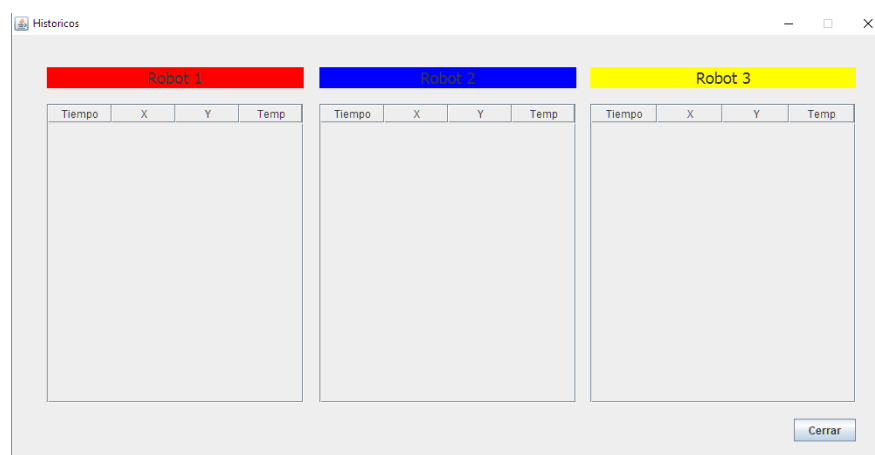


Figura 52. Pantalla de datos históricos desarrollada para este trabajo.

4.3 Desarrollo del controlador cooperativo centralizado para MRSLAM

El control centralizado fue desarrollado para coordinar el movimiento de los tres agentes robóticos sobre el entorno a explorar para completar el mapa deseado. Los principales objetivos del control cooperativo es dispersar y evitar las colisiones con obstáculo y entre robots. Para esto el controlador utiliza las estimaciones odométricas, de dirección y las distancias captadas por los sensores ultrasónicos de cada agente, para redireccionarlos a una nueva posición de interés.

Todo el software de control central se diseñó y programó sobre el IDE NetBeans 8.1 para el lenguaje de programación Java debido a su fácil adaptación a varios sistemas operativos y por ser un lenguaje de código abierto.

4.3.1 Consideraciones de diseño

El control centralizado se encarga de analizar la información obtenida por los tres agentes robóticos y ordenar la dirección que deben tomar para completar el mapeado de la forma más eficiente. Tomando en cuenta lo anterior, es necesario analizar las posibles interacciones físicas de los robots con el entorno y los demás agentes que se están desarrollando en el mismo ambiente.

En el caso particular de cada agente robótico, poseen un incremento de la incertidumbre producida después de un giro o un recorrido lineal, deslizamientos en las ruedas por el contacto con una superficie, y falsas lecturas o errores de sensado por interferencias externas. Estos problemas fueron resueltos tomando en cuenta las siguientes consideraciones:

- Cada agente robótico es calibrado individualmente de tal manera que su movimiento sea rectilíneo y su distancia recorrida sea confiable.
- Cada brújula electrónica de cada plataforma robótica será calibrada de forma individual, de tal manera que sus mediciones sean lo más exactas posibles.
- Las rotaciones de cada agente se realizarán sobre su propio eje y su desplazamiento será frontal con excepción de situaciones especiales donde solo sea posible salir en reversa.
- Los obstáculos se consideran estáticos, por lo tanto no se toma en cuenta el movimiento o cambio en el entorno.
- La posición inicial de los agentes se considera constante y conocida, es decir que los agentes robóticos siempre parten desde una misma posición en el mapa a crear.

Conjuntamente, se toma en cuenta las siguientes reglas para el comportamiento de cada agente. Estas reglas evitan colisiones y la redundancia innecesaria en la exploración. Las interacciones más relevantes se pueden observar en la **Figura 53** y **Figura 54**.

Interacción Agente-Entorno (**Figura 53**):

- El agente debe evitar acercarse demasiado a los bordes y obstáculos detectados. (a)
- El agente debe evitar acercarse hacia otro agente. (b)
- El agente deberá desplazarse en reversa si se encuentra muy cerca de un obstáculo para evitar chocar al girar o avanzar. (c)

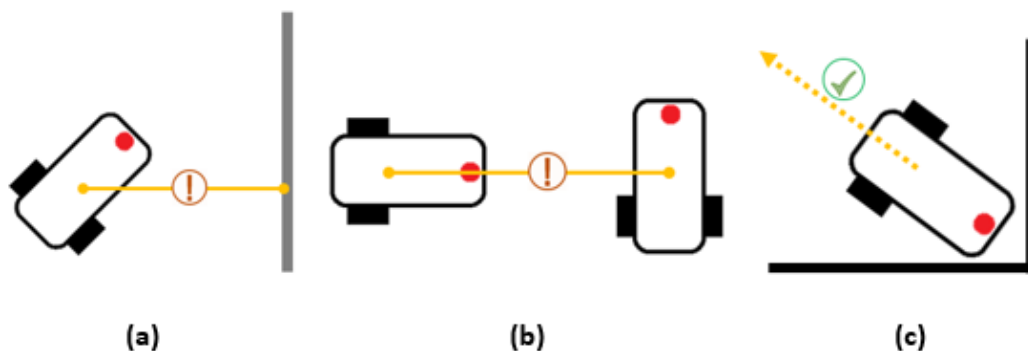


Figura 53. Posibles interacciones entre el agente robótico y el entorno.

Interacción Agente-Interfaz (**Figura 54**):

- El agente debe evitar salir de los bordes del mapa graficado en la interfaz gráfica. (1)
- El agente debe evitar recorrer espacios ya explorados. (3)
- El agente debe dirigirse hacia las zonas desconocidas en el mapa creado. (2)

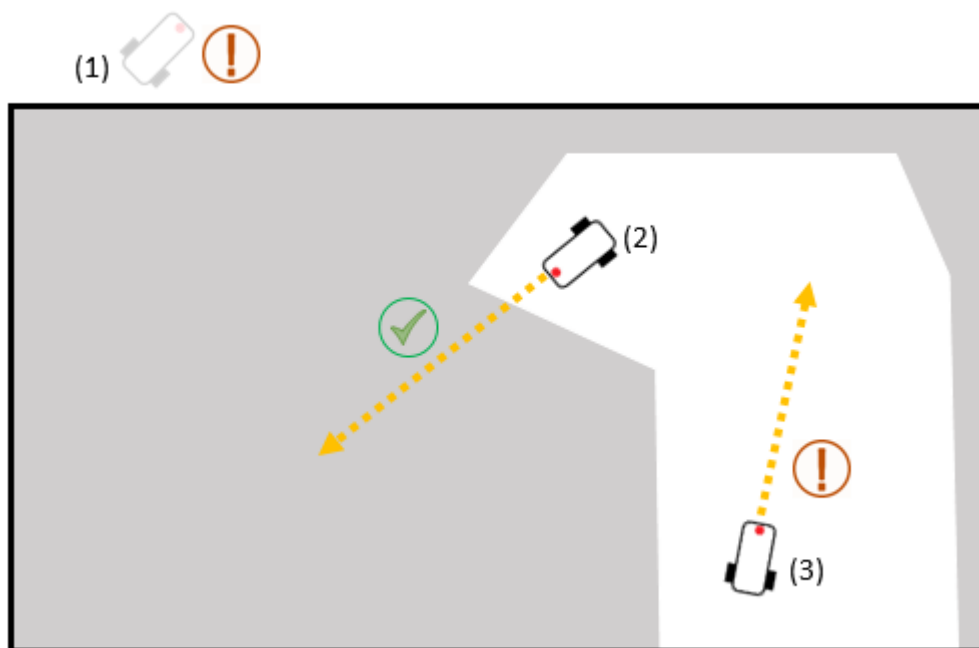


Figura 54. Posibles interacciones entre el agente robótico y el mapa.

4.3.2 Diseño del algoritmo de control

El algoritmo desarrollado, está diseñado para resolver las consideraciones presentadas en el punto 4.3.1 de forma eficiente y solucionar los problemas de manera prioritaria. Además, el algoritmo toma en cuenta que la comunicación de los agentes

con el controlador central no puede ser simultánea, es decir si un robot móvil está enviando un paquete de datos, el resto de robots no podrán comunicarse con el controlador central hasta que termine de analizar y reaccionar ante la información obtenida.

La Figura 55 muestra el diagrama de flujo que muestra la lógica de programación del algoritmo principal desarrollado para cumplir estas funciones.

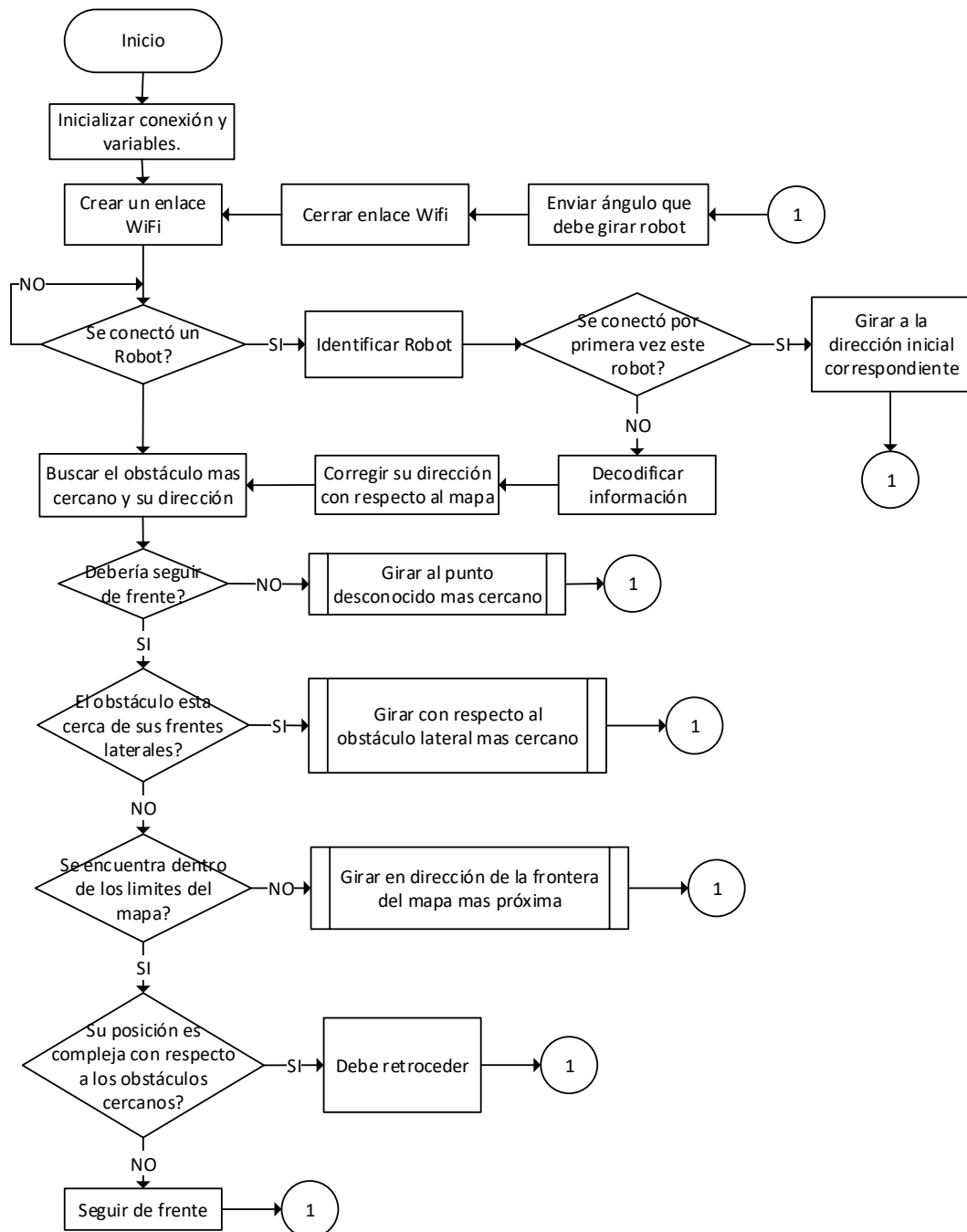


Figura 55. Diagrama de flujo de programa de control centralizado.

El diagrama de flujo de la **Figura 55** muestra el proceso general que realiza el computador al momento de recibir información de cada robot para procesarla y mostrar los resultados en los mapas de la interfaz gráfica desarrollada. Adicionalmente, con la actualización del mapa creado, el controlador central decide hacia qué dirección debe dirigirse cada robot. Como parte de la solución se muestran tres subprocesos que determinan el recorrido angular que debe realizar el agente robótico conectado. A continuación se describe su funcionamiento.

Girar al punto desconocido más cercano, busca en el mapa la celda sin explorar más cercana al robot. Una vez localizado el punto desconocido más cercano, se evalúa varias condiciones, como que para llegar a esa posición no exista un obstáculo en la ruta y que existe el espacio suficiente para que pase el robot.

Girar con respecto al obstáculo lateral más cercano, se encarga de redirigir al robot en caso de que exista un obstáculo próximo a ser impactado si su dirección se mantiene constante, es decir que si se detecta que existe un obstáculo próximo a 30 grados por ejemplo de su dirección actual, el agente debe girar para impedir que se siga acercando a dicho obstáculo. Para este propósito se eligió un ángulo complementario de corrección debido a que el controlador fue diseñado para espacios interiores donde generalmente existen fronteras rectangulares.

Girar en dirección de la frontera del mapa más próxima, redirecciona al agente robótico en caso de que el mapa muestre que ha sobrepasado los límites del mapa establecido en la pantalla. Entonces el robot móvil debe girar hacia la frontera más cercana para volver a realizar su trabajo.

4.3.3 Entradas y salidas del sistema de control

El controlador desarrollado es un proceso de lazo abierto por la naturaleza física del sistema. Consta de tres plantas independientes que en este caso son las plataformas robóticas móviles y un controlador coordinador central que permite modificar la dirección de cada robot. En la **Figura 56** se puede observar el proceso general del controlador central diseñado.

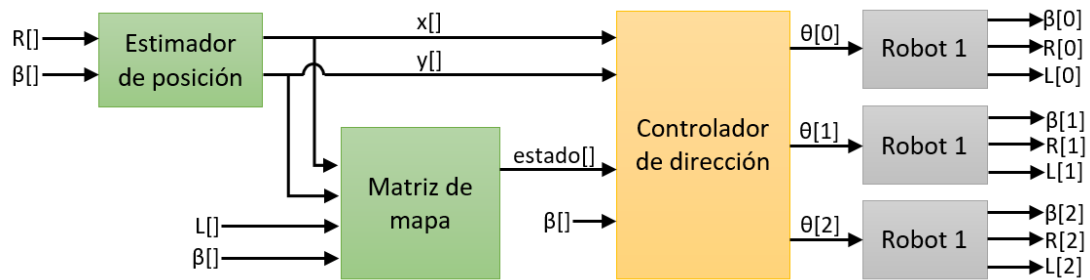


Figura 56. Diagrama de bloques de controlador central del sistema MRSLAM.

Donde estado[], es la matriz de estados de ocupación del mapa de obstáculos, x[] y y[] son las estimaciones de posición de los tres agentes robóticos, β [] la estimación de dirección de cada robot, L[] es un array donde se guarda las distancias escaneadas en el entorno cercano de cada agente, R[] son los recorridos lineales de cada robot, y θ [] es el recorrido angular que debe realizar cada agente para cambiar su dirección. En este diagrama se puede observar que se controla el ángulo de giro de los agentes robóticos para cambiar su dirección. Igualmente, no se considera las perturbaciones sistemáticas y no sistemáticas existentes en los robots móviles, debido a que se consideran una constante dentro de este trabajo.

En la **Tabla 11**.

Entradas y salidas del controlador central del sistema MRSLAM. se describe las variables utilizadas para realizar el control de dirección en los 3 agentes robóticos.

Tabla 11.

Entradas y salidas del controlador central del sistema MRSLAM.

Salida		
Variable	Tipo	Descripción
girar[]	Double array	Almacena el giro que tiene que realizar cada robot.
Entradas		
Variable	Tipo	Descripción
estado[]	Int array	Almacena los cambios en la matriz de estados de ocupación del mapa.
x[]	Double array	Almacena las posiciones actuales en el eje X de los agentes robóticos.

y[]	Double array	Almacena las posiciones actuales en el eje Y de los agentes robóticos.
dir[]	Double array	Almacena las direcciones actuales de los agentes robóticos.
dist[]	Double array	Almacena las distancias a los obstáculos cercanos actuales de los 3 agentes robóticos.
id[]	Int	Almacena el identificador del último robot conectado.

4.3.4 Diseño de filtros espaciales

Al finalizar el mapeado por parte de los agentes robóticos móviles, el controlador central genera un mapa de obstáculos y de niveles de temperatura de acuerdo a la información almacenada en las respectivas matrices. Donde se realiza una estimación en ambos mapas creados para las celdas aún desconocidas, creando una imagen completa del espacio explorado. Con este fin se usó filtros simples lineales de media para mejorar los resultados finales de los dos mapas.

Filtro de Mapa de obstáculos

Para la imagen de obstáculos, al finalizar el proceso de mapeado se estima el valor de las celdas aún desconocidas mediante un filtro lineal de medias, es decir las celdas desconocidas en el mapa final se asumen como un obstáculo o como un espacio vacío según el nivel de pertenencia a estos grupos. Dado por el valor de los pixeles vecinos ha dicho punto evaluado. Para este caso la imagen analizada es binaria (0 si es un obstáculo y 1 si es un espacio vacío), por lo que el filtro se simplifica.

Entonces se considera una celda como un obstáculo si por lo menos 3 de sus celdas vecinas son un obstáculo. Con lo cual se puede asumir que la celda evaluada se encuentra detrás de algún objeto, esquina o una pared como se observa en la **Figura 57**.

--	Obst.	Obst.	--	--	Obst.	--	--	Libre	--	--	--
--	p(x,y)	Obst.	--	p(x,y)	Obst.	--	p(x,y)	Libre	Libre	p(x,y)	--
--	--	--	--	--	Obst.	--	--	Libre	Libre	Libre	--
a)			b)			c)			d)		

Figura 57. Condiciones mínimas para pertenecer a un obstáculo o un espacio vacío.

Donde $p(x,y)$ es el pixel desconocido evaluado, las figuras a) y b) muestran dos posibles escenarios para estimar un punto como obstáculo y las figuras c) y d) son las posibles situaciones para asumir que el punto $p(x,y)$ es un espacio vacío.

Una situación menos común se visualiza en la **Figura 58**

Libre	Obst.	Obst.
Libre	p(x,y)	Obst.
Libre	Libre	Obst.

Figura 58. Condición de baja probabilidad para un punto desconocido.

En donde el código simplemente da mayor peso a los espacios libres, resultando entonces el punto $p(x,y)$ como un espacio vacío.

Al usar este filtro, son únicamente modificadas las celdas vecinas a un punto explorado. Por lo tanto este proceso se repite hasta obtener una ventana de 50x50 pixeles para cubrir las posibles áreas desconocidas más extensas.

Filtro de Mapa de temperaturas

El filtro utilizado para estimar las temperaturas en los espacios desconocidos, es un filtro espacial lineal de media clásico, el cual difumina la imagen en el contorno de los valores conocidos mediante el valor medio de las celdas o pixeles cercanos.

--	21	21	23	30
--	--	22	25	30
--	--	P(x,y)	26	31
--	--	--	26	32
--	--	--	27	32

Figura 59. Ejemplo de un punto desconocido en el mapa de niveles de temperaturas.

En este caso se usó una ventana de 5x5 píxeles, donde el promedio de los valores dentro de la máscara modifican el punto P(x,y). Por ejemplo en la **Figura 59** para calcular el valor de P(x,y) se usa la ecuación:

$$P(x, y) = (21 + 21 + 23 + 30 + 22 + 25 + 30 + 26 + 31 + 26 + 32 + 27 + 32)/13 \quad (16)$$

$$P(x, y) = 26.61 \quad (17)$$

Este análisis se realiza en cada celda del mapa, generando un mapa completo del espacio explorado, con una asignación de los valores aproximados de temperatura en sectores donde no se obtuvo una medida puntual por parte de los agentes robóticos.

4.3.5 Clases y métodos Desarrollados

A continuación se detalla las clases creadas para el correcto funcionamiento de controlador central. Todas las clases creadas poseen un constructor para su inicialización con sus respectivos parámetros necesarios con excepción de la clase “Control” que se encarga del análisis matemático.

Tabla 12.
Clases usadas para el controlador central.

Clase	Descripción
Control	Esta clase inicia las componentes estáticas y de trasladar la información obtenida mediante la clase ConexiónWiFi hacia el JPanel dedicado para el mapa de temperaturas.

ConexionWifi(Matriz de estado)	Crea el enlace Wi-Fi con los módulos Esp8266 y realizar la correcta interpretación de la información obtenida.
Dibujo(JPanel, JTextField)	En esta clase se crea el hilo para la constante actualización de los parámetros de control con la información obtenida por la clase ConexionWifi.

Estas clases están constituidas por métodos que hacen posible el funcionamiento de cada clase. En la **Tabla 13** se detalla por clases los métodos que las constituyen y su propósito dentro del controlador centralizado MRSLAM.

Tabla 13.
Descripción de métodos para el controlador central.

Control	
double girar(matriz de estado, posición, dirección, distancias ultrasónicas e ID de robot)	Esta función se encarga de priorizar las posibles situaciones que puedan acontecer en un estado de la posición de los agentes robóticos. Igualmente mediante este método se analiza el mejor ángulo de giro del robot móvil actualmente conectado
double afuera(posición y dirección)	Entrega el valor angular de giro cuando el robot se encuentra fuera del espacio dispuesto en el mapa.
boolean defrente(matriz de estado, posición y dirección)	Mediante esta función se identifica si el robot conectado está en una dirección adecuada. Es decir, si la zona a la cual se dirige ya fue explorada o no.
double robotCerca(posición, dirección e ID de robot)	Compara las distancias actuales entre los robots para girar en caso de que exista un robot cerca de otro.
double puntoCercano(matriz de estado, posición y dirección)	Este método encuentra la celda desconocida más cercana y analizar el recorrido angular que debe realizar el agente robótico conectado para llegar hacia él.

ConexionWifi(Matriz de estado)	
initServer(matriz de estados)	Inicia un enlace TCP/IP para recibir la información proporcionada por los agentes e interpretarla en el controlador.
setDatos(mensaje y matriz de estados)	Interpreta la información obtenida en un mensaje recibido por algún agente robótico.
String[] separarFrase(mensaje)	Separa el mensaje recibido por el agente en diferentes cadenas de caracteres para su posterior interpretación.

Continúa 

Dibujo(JPanel, JTextField)	
Animar()	Inicializa un hilo encargado de abrir el enlace de comunicación y permitir la entrega de información.
Detener()	Detiene el hilo de animación y de comunicación con los agentes robóticos.

Adicionalmente se crearon métodos *Set* y *Get* para el llamado de datos privados entre clases.

4.4 Desarrollo del control independiente de los agentes robóticos

Los agentes robóticos móviles están equipados con la tarjeta de programación Arduino Mega 2560, la cual además de estar encargada de adquirir los datos de los sensores, también se encarga de realizar el control del desplazamiento de cada plataforma robótica. Este controlador denominado independiente por ser aplicado por separado en los tres robots disponibles para el mapeado, tiene el objetivo principal de mantener un recorrido constante de 30cm entre escaneos del entorno y de realizar los cambios de dirección ordenados por el controlador central con el menor error posible.

4.4.1 Consideraciones de diseño

Para poder desarrollar un controlador adecuado, es necesario analizar las posibles interacciones físicas de los robots con el entorno y con su propio mecanismo de

tracción. Aquí interviene directamente los cálculos de odometría para el desplazamiento del robot móvil tipo diferencial. Estos cálculos son una estimación del recorrido del robot con respecto al movimiento de sus ruedas. Existen 2 tipos de errores, los sistemáticos producidos por la sensibilidad de los sensores, el desalineamiento de las ruedas, y diferencia de velocidad entre ruedas; y los errores no sistemáticos, como cambio del tipo de superficie y deslizamientos por choques o por agujeros en la superficie. Estos problemas fueron resueltos tomando en cuenta las siguientes consideraciones:

- Los robots deben movilizarse a una velocidad constante controlable, tomando en cuenta el torque necesario para arrancar.
- Cada agente robótico es caracterizado individualmente, mediante software y hardware, de tal manera que su movimiento sea rectilíneo y su distancia recorrida sea confiable.
- Las rotaciones de cada agente se realizarán sobre su propio eje y su desplazamiento será frontal con excepción de situaciones especiales donde solo sea posible salir en reversa.

4.4.2 Diseño del algoritmo de control

El algoritmo en las tres plataformas robóticas debe estar constituidas por la misma lógica de programación y las mismas funciones por ser robots homogéneos basados en las consideraciones presentadas. Cada plataforma tiene su propio modelo físico sobre el cual debe ser aplicado el controlador ya que las variaciones mecánicas en las plataformas generan un cambio en el modelo ideal que debe ser considerado para minimizar los errores sistemáticos. Asimismo el programa está protegido con un Watchdog Timer para evitar que el programa colapse principalmente por problemas de comunicación.

La Figura 60 muestra el diagrama de flujo muestra la lógica de programación del algoritmo principal desarrollado para cumplir estas funciones. El diagrama de flujo de la **Figura 60** muestra el proceso que realiza un Robot móvil al momento de ser puesto en marcha. En general su objetivo es desplazarse, recopilar información del ambiente, informar al controlador central y dirigirse en la dirección que él le ordene. Como parte de la solución se muestran varios subprocesos importantes para el control de giro, control de recorrido y comunicación. A continuación se describe su funcionamiento.

Enviar mensaje de conexión inicial, establece la comunicación inalámbrica Wi-Fi y de enviar un mensaje confirmando que se ha establecido la comunicación con ese agente robótico. Con el objetivo de informar que está listo para comenzar a cumplir la tarea dispuesta.

Medir distancia libre, este subproceso asegura que la distancia medida por cualquier ultrasónico sea acertada. Realiza varias mediciones en un mismo lugar hasta que el error entre las mediciones sea inferior a 1 cm.

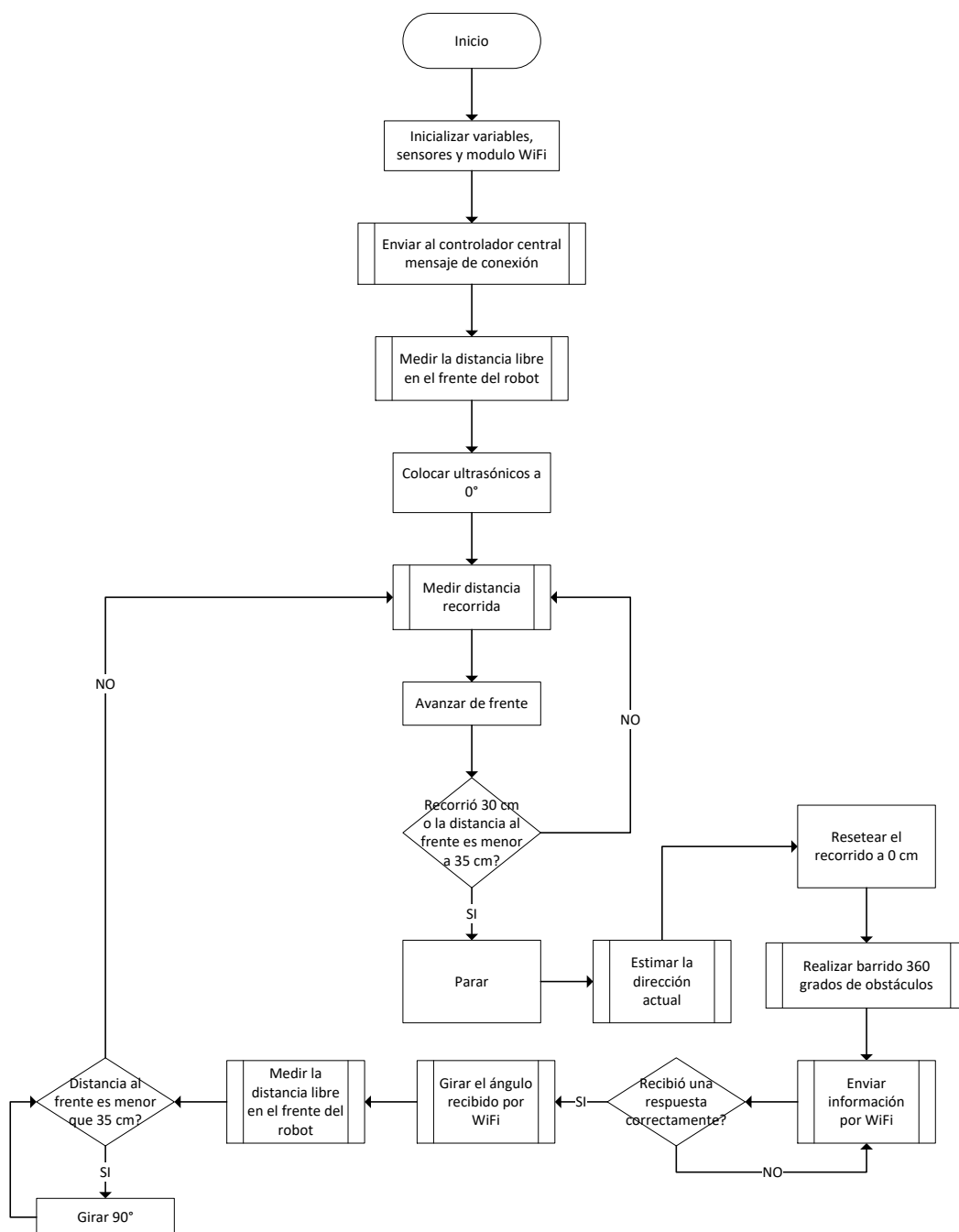


Figura 60. Diagrama de flujo de programa de control parcial de cada robot móvil.

Estimar dirección actual, realiza la medición mediante la brújula electrónica con sus respectivos parámetros de calibración de la dirección del norte geomagnético con respecto a su orientación actual.

Realizar barrido 360 grados de obstáculos, posiciona la torreta en 6 direcciones distintas para tomar 12 distancias de obstáculos en el entorno cada 30 grados comenzado por el frente y espalda del agente. Aparte de tomar las muestras de obstáculos, también sensa la temperatura en dicha posición.

Enviar información por Wi-Fi, envía los comandos AT correspondientes al módulo Esp8266 para crear un enlace y enviar la información recolectada hacia el controlador central. La trama de envío desarrollada tiene la estructura de la **Figura 61**.

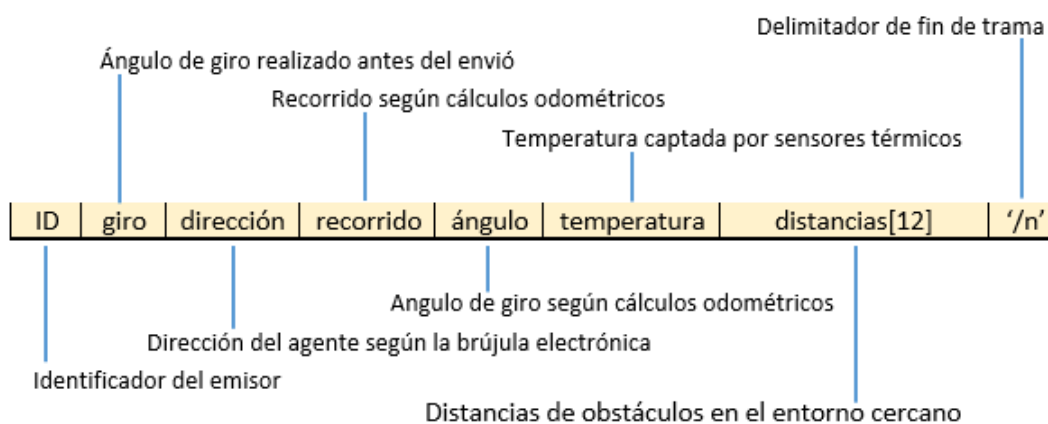


Figura 61. Trama para el envío de información.

Girar ángulo recibido por controlador central, se encarga de realizar el giro de un agente robótico en su propio eje para cambiar su dirección según la coordinación del controlador central.

4.4.3 Control de recorrido

Como hemos descrito anteriormente, dentro del algoritmo de desplazamiento de cada robot se efectúan pasos constantes de 30 cm para realizar un nuevo sensado del ambiente explorado. El control de recorrido desarrollado, asegura que los tres agentes robóticos recorran la distancia deseada con un error menor a 1 cm. Este controlador on-off de lazo abierto, está implementado sobre el modelo cinemático presentado en la ecuación 14.

Con este fin, encontramos la relación entre el disco codificado del encoder y el desplazamiento lineal del agente robótico móvil.

$$f = \frac{\pi * D}{R} \quad (18)$$

Donde f es el factor de conversión entre el número de pulsos y el desplazamiento lineal del robot móvil, D es el diámetro de las ruedas medido entre los puntos de apoyo estimados, y R la resolución del disco codificado del encoder en número de pulsos por revolución. Remplazando los valores medidos en los robots móviles, tenemos que:

$$f = \frac{\pi * 6.5 \text{ cm}}{40 \text{ pul/rev}} \quad (19)$$

$$f = 0.5105 \text{ cm/pulso} \quad (20)$$

Por lo tanto la distancia recorrida por cada rueda es:

$$\begin{aligned} Li &= f * Ni \\ Ld &= f * Nd \end{aligned} \quad (21)$$

Donde Li y Ld son los desplazamientos lineales de la rueda izquierda y derecha correspondientemente después de Ni y Nd número de pulsos detectados en cada rueda.

Entonces el desplazamiento absoluto de cada plataforma robótica, está definida por la ecuación 22 despejada de la ecuación 19.

$$\begin{aligned} Lc &= \frac{(Li + Ld)}{2} \\ Lc &= \frac{f * (Ni + Nd)}{2} \\ Lc &= \frac{0.5105 * (Ni + Nd)}{2} \\ Lc &= 0.2552 * (Ni + Nd) \end{aligned} \quad (22)$$

Y el cambio de direcciones está dado por:

$$\Delta\theta = \frac{(Li - Ld)}{B} \quad (23)$$

Donde B es la distancia que existe entre las ruedas del agente robótico móvil.

Entonces, remplazando la ecuación 19 en 21 tenemos que:

$$\begin{aligned} \Delta\theta &= \frac{f * (Ni - Nd)}{B} \\ \Delta\theta &= \frac{0.5105 * (Ni - Nd)}{13.5 \text{ cm}} \end{aligned} \quad (24)$$

$$\Delta\theta = 0.03781 * (Ni - Nd)$$

Como mencionamos anteriormente, estos cálculos están propensos a sufrir modificaciones dependiendo de la superficie, puntos de apoyo variantes en las ruedas, falta de alineación de ruedas, etc. (Johann Borenstein & Feng, 1996)

Los cálculos odométricos están desarrollados en una tarjeta de programación Arduino Nano, conectada mediante el puerto serial al controlador Arduino Mega2560. Una vez enviada esta información, el algoritmo se encargará de realizar un control digital on-off cuando haya recorrido la distancia deseada. Por el tipo de movimiento repetitivo he invariante en recorrido hacen apropiado este tipo de controlador para dicha aplicación.

Con el fin de ofrecer mejor precisión, en cada agente se realizaron mediciones de desplazamiento lineal para mejorar el comportamiento de dicho controlador sobre pruebas reales. Las pruebas fueron realizadas sobre la superficie que van a trabajar, con recorridos periódicos de 30 cm consecutivos hasta completar 300 cm de recorrido lineal. (Ver **Figura 62**)

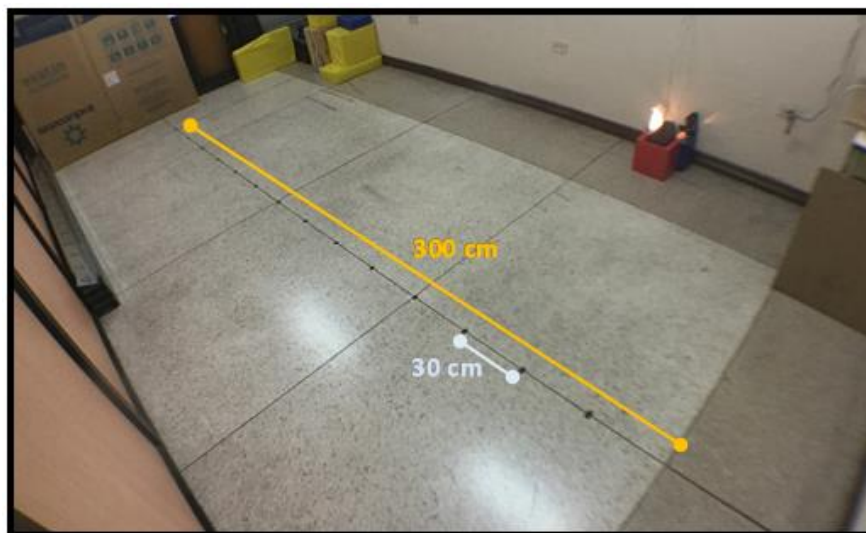


Figura 62. Pruebas de recorrido lineal para controlador de recorrido.

Se realizaron 10 pruebas de recorrido, donde se midió la posición real respecto a la deseada y se calculó la media de las 10 muestras cada 30 cm en un recorrido total de 300 cm. Como resultado de las pruebas se obtuvo los datos de la **Tabla 14**.

Tabla 14.
Resultados de las pruebas del error de recorrido.

	Media (cm)	Desviación Estándar (cm)
Robot 1	2.5160	0.79
Robot 2	3.37	1.23
Robot 3	0.4743	1.97

El error encontrado tiene un comportamiento creciente por la acumulación de los errores presentados anteriormente. Si observamos esta información en la **Figura 63**, se puede comprobar que tiene un comportamiento lineal. Por lo tanto este error se puede compensar con un factor de ganancia igual al valor promedio del error en cada recorrido.

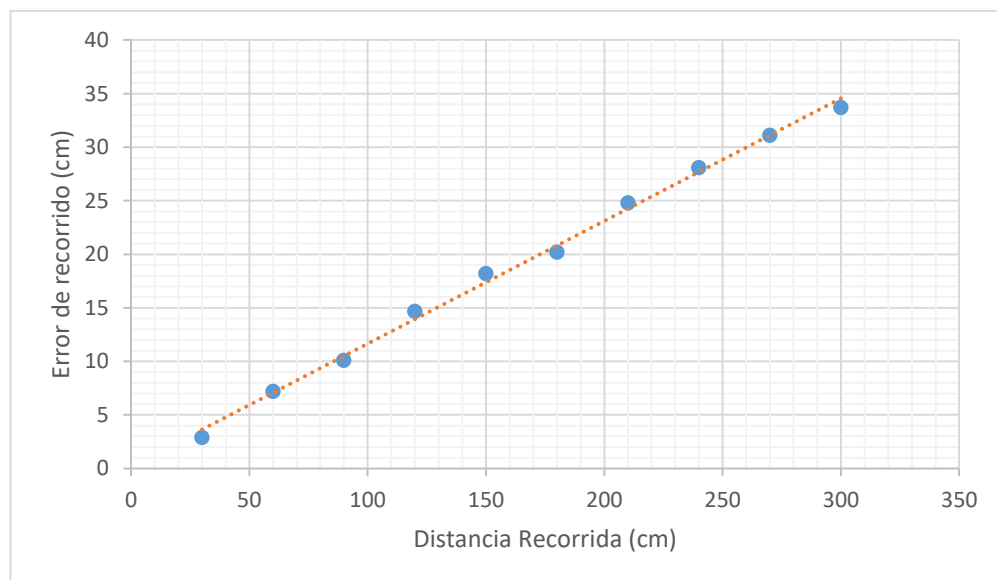


Figura 63. Gráfica del error acumulativo medido en distintas posiciones.

Se realizan las mismas pruebas en los 3 agentes robóticos móviles obteniendo resultados similares. Donde se demostró que en todos los experimentos se obtiene un error positivo, es decir que siempre sobrepasa los 30 cm de recorrido deseado. En la **Tabla 15** se detalla el error encontrado en cada agente antes y después de ser compensado.

Tabla 15.
Corrección lineal de error en recorrido en los 3 robots móviles.

	Error antes de corrección	Error después de corrección
Robot 1	2.5160 cm	0.0314 cm
Robot 2	3.37 cm	-0.4214 cm
Robot 3	0.4743 cm	0.4743 cm

Como consecuencia el controlador digital on-off para recorrido implementado pondrá en marcha los motores hasta que el recorrido sea igual a 30 cm menos el error encontrado experimentalmente en cada robot móvil.

4.4.4 Control de giro

El controlador de giro, es un subproceso del agente robótico encargado de realizar un giro sobre su propio eje mediante un control on-off de lazo abierto, con un valor equivalente a su recorrido lineal dado en radianes por el controlador central. Su funcionamiento cinemático, tiene el mismo principio presentado para el control de recorrido, con la diferencia que ahora realiza un recorrido angular.

La suma de los arcos formados por las 2 llantas al desplazarse en sentido contrario, generan el ángulo deseado para su nuevo posicionamiento, es decir que cada llanta recorre una distancia igual a la mitad del arco formado por el giro en el eje central del robot. (Ver **Figura 64**)

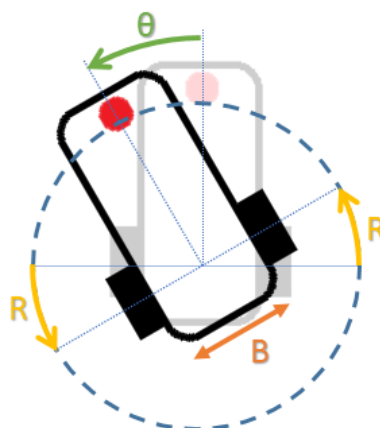


Figura 64. Movimiento giratorio de un robot diferencial.

Por definición geométrica, la relación que existe entre la longitud del arco, el radio y el ángulo formado por la unión de los extremos del arco es:

$$L_{arco} = r * \theta \quad (25)$$

Donde L_{arco} es la longitud del arco, r el radio de la circunferencia y θ el ángulo existente entre los dos radios que unen el centro de la circunferencia con los extremos del arco. Aplicando este concepto a un robot diferencial tenemos que:

$$R = \frac{B}{2} * \theta \quad (26)$$

Donde R es el recorrido lineal que debe realizar para girar un ángulo θ en radianes, y B es la distancia que existe entre las ruedas de la plataforma robótica. Aplicando la ecuación en las 2 ruedas del robot móvil, tenemos que:

$$2R = \frac{B}{2} * 2\theta \quad (27)$$

$$2R = B * \theta$$

Considerando que el encoder no discrimina el sentido de giro de las ruedas y que por ende sus cálculos de recorrido están realizados para un desplazamiento lineal. Entonces $2R$ representa al desplazamiento lineal calculado en la ecuación 20.

Los cálculos odométricos están desarrollados en una tarjeta de programación Arduino Nano, la misma que ha sido utilizada para adquirir las señales de los sensores encoders, conectada mediante su puerto serial al controlador Arduino Mega2560. Una vez enviada esta información el algoritmo se encargará de realizar un control digital on-off cuando haya girado el ángulo deseado.

Con el fin de ofrecer mejor precisión, en cada agente se realizaron estimaciones angulares para mejorar el comportamiento de dicho controlador sobre pruebas reales. Las pruebas fueron realizadas sobre la superficie que van a trabajar, con giros periódicos de 90 grados, en donde se encontró un error medio de 16 grados.

Como consecuencia el controlador digital on-off para giro implementado pondrá en marcha los motores en sentido inverso hasta que el ángulo sea igual al valor deseado menos el error encontrado experimentalmente en cada robot móvil.

4.4.5 Librerías y funciones desarrolladas

A continuación se describe las librerías usadas para facilitar la programación de sensores y canales de comunicación que permiten el correcto funcionamiento de cada

controlador independiente. Todas las librerías son propias del IDE Arduino y se pueden descargar e instalar mediante la misma interfaz de programación.

Tabla 16.
Descripción de librerías usadas para el controlador independiente de cada agente robótico.

Nombre	Descripción
Avr/wdt.h	Sirve para habilitar, deshabilitar y resetear un temporizador tipo Watchdog.
Wire.h	Crea un canal de comunicación con algún modulo o dispositivo mediante el protocolo I2C.
DallasTemperature.h	Crea un canal de comunicación 1wire con los módulos de temperatura DS18B20 y se encarga de interpretar la señal como un valor entero de temperatura en grados centígrados.
Servo.h	Envía la señal adecuada para realizar el movimiento angular deseado.
Ultrasonic.h	Envía el disparo de pulsos adecuado, y leer el tiempo que se demoró en regresar para calcular la distancia existente frente al ultrasónico.

Las funciones desarrolladas, tienen como finalidad organizar y mejorar la eficiencia del código para ciertos procesos repetitivos. Además facilitan la comprensión del código, por lo que es flexible y amigable para futuras mejoras.

Tabla 17.
Funciones desarrolladas para el controlador independiente de cada agente robótico.

Nombre	Tipo	Parámetros	Descripción
escaneo360	Void	--	Realiza las mediciones de obstáculos y temperaturas en 360 grados.
cuentaEncoder	Void	--	Lee mediante un puerto serial el valor de recorrido lineal y angular

Continua 

			calculado por un Arduino Nano mediante odometría.
medirDistancia	Float	Ultrasónico	Asegura la fidelidad de la información entregada por los sensores ultrasónicos.
girar	Void	Angulo de giro	Realiza el giro de un agente robótico en su propio eje hacia la dirección deseada.
norte	Float	--	Interpreta y calcula la dirección del norte magnético de la tierra por medio de la brújula electrónica.
leerMsnWifi	String	--	Interpreta la información recibida mediante el módulo Wi-fi Esp8266.
enviarMsnWifi1	Void	--	Envía la cadena de caracteres de inicialización mediante la comunicación Wi-Fi hacia el puerto y dirección deseada.
enviarMsnWifi	Void	Mensaje a enviar	Envía una cadena de caracteres mediante la comunicación Wi-Fi hacia el puerto y dirección deseada.
enviarComandoWifi	Void	Comando	Realiza el envío de un comando AT hacia el módulo Wi-Fi y valida su correcto funcionamiento.

CAPÍTULO V

PRUEBAS Y RESULTADOS EXPERIMENTALES

5.1 Introducción

Los experimentos ejecutados tienen como objetivo demostrar el funcionamiento del sistema multi robótico en diferentes escenarios, de forma que permitan evaluar el comportamiento de los robots móviles tanto grupal como individualmente. Debido a que gran parte de este trabajo fue dedicado a la construcción de las 3 plataformas robóticas, las primeras pruebas realizadas se enfocan en el correcto funcionamiento e interpretación de los sensores y actuadores de cada agente. Las pruebas realizadas se dividen en dos etapas:

- Pruebas y calibración de sensores.
- Pruebas de funcionamiento.

5.2 Pruebas y calibraciones de sensores

Las pruebas mostradas en esta sección, tienen como objetivo mostrar el comportamiento de los sensores utilizados para la correcta interpretación de la información sensada. Los sensores caracterizados para este proyecto fueron los sensores ultrasónicos, brújulas electrónicas y encoders, que son los principales responsables de medir la posición y distancias de los obstáculos. Para el resto de sensores se utilizó la información proporcionada por los fabricantes.

5.2.1 Pruebas con sensores ultrasónicos

Los sensores ultrasónicos se probaron con 3 algoritmos distintos para medir distancias, estos programas están basados en librerías desarrolladas por distintos autores. Con ellos se probó la exactitud y los tiempos de muestreo de cada versión. Las pruebas de precisión se realizaron colocando el sensor a distancias fijas desde los 10 cm variando en rangos de 10 cm hasta los 100 cm.

Se tomaron 500 muestras en cada distancia adquirida con cada algoritmo obteniendo resultados que muestran el comportamiento de cada código con respecto a cada distancia. Por ejemplo en la **Figura 65** se puede observar que el programa 1 es el más estable y exacto para una distancia de 20 cm.

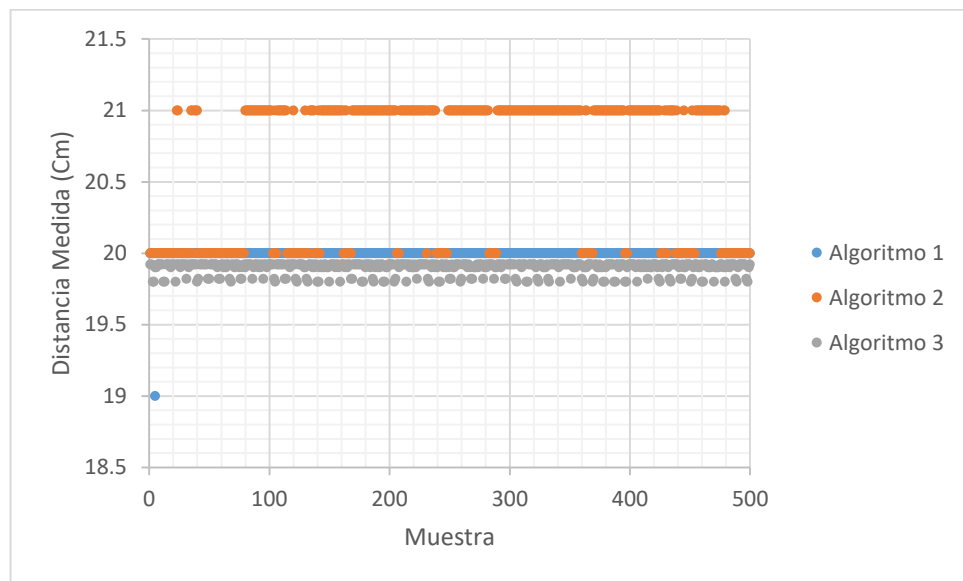


Figura 65. Resultado de 3 versiones de programas midiendo una distancia de 20 cm con 500 muestras.

En resumen el error promedio encontrado con cada código a las distintas distancias se representa en la **Figura 66**.

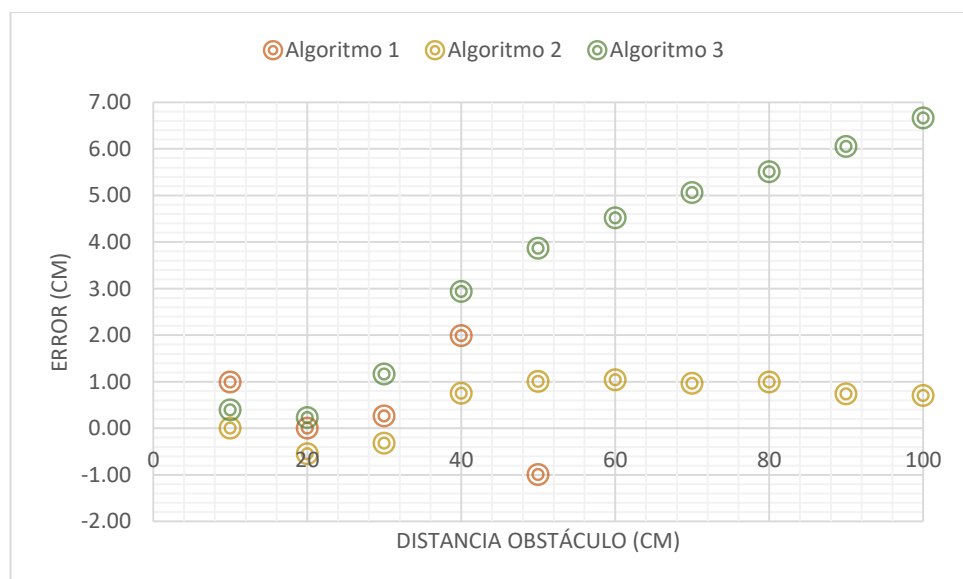


Figura 66. Resultados sobre el error obtenidos con 3 programas a distintas distancias.

Se puede observar en la gráfica que el algoritmo 3 tiende a aumentar su error mientras más alejado se encuentre su objetivo, mientras que el algoritmo 2 mantiene un error por debajo de 1 cm. El algoritmo 1 obtiene la distancia únicamente hasta los 50 cm para asegurar datos más confiables, sin embargo se observa que el error aumento considerablemente midiendo una distancia de 40 cm.

Para complementar esta comparación de programas, se consideró en las pruebas el tiempo de procesamiento necesario para obtener tales distancias. Con este fin se realizaron 600 mediciones constantes con cada algoritmo tomando el tiempo que se demora en cruzar nuevamente por el registro 0, El promedio del tiempo de procesamiento se muestra en milisegundos en la **Tabla 18**.

Tabla 18.
Comparación de los tiempos de procesamiento de los 3 códigos usados.

TIEMPO DE PROCESAMIENTO (ms)		
Algoritmo 1	Algoritmo 2	Algoritmo 3
5.71671827	113.873418	5.71671827

Por lo tanto el algoritmo que se eligió es el denominado algoritmo 1, debido a que muestra el mejor equilibrio en relación de tiempos de procesamiento y precisión.

Adicionalmente, se probó el funcionamiento de los sensores ultrasónicos en distintos ángulos de incidencia y su efecto con otros ultrasónicos actuando simultáneamente. Donde se encontró inconvenientes al momento de reconocer planos con ángulos de incidencia mayores a los 60 grados. Por otro lado, no se obtuvo ninguna anomalía en el funcionamiento de los sensores al medir en simultaneo un mismo blanco.

5.2.2 Pruebas con sensores encoders

Los sensores encoders se probaron sobre tres distintas velocidades del disco codificador acoplado a uno de los motores DC de la plataforma robótica, considerando como velocidad baja 47.6 rpm, velocidad media 80.4 rpm y velocidad alta 111.3 rpm con los siguientes parámetros de evaluación:

- Ancho del pulso.
- Numero de flancos ascendentes detectados

Se tomaron muestras durante 10 segundos en las distintas velocidades con un Arduino Nano con un tiempo de muestreo de 18 ms, la señal entregada por el sensor encoder es una onda cuadrada o un tren de pulsos donde el número de flancos ascendentes y la longitud de onda varían con respecto a la velocidad del motor DC. Los datos recopilados se pueden observar en la **Tabla 19**.

Tabla 19.
Resultados de pruebas con sensores encoders.

Ancho de pulso (x18ms)	Número de flancos ascendentes detectados		
	Velocidad Baja	Velocidad Media	Velocidad Alta
1	--	85	257
2	11	191	138
3	98	21	4
4	53	2	--
5	5	3	--
6	2	--	--
Total de Pulsos	169	303	400

Se puede evidenciar que existe una clara tendencia a determinado ancho de pulso según la velocidad del disco codificado, aparte de aumentar el número de flancos ascendentes detectados proporcionalmente a la velocidad angular del motor DC. En la Figura 67. Resultados de pruebas de velocidad con *sensores encoders*. Figura 67 se puede visualizar el comportamiento de los encoders según los parámetros de evaluación mencionados anteriormente.

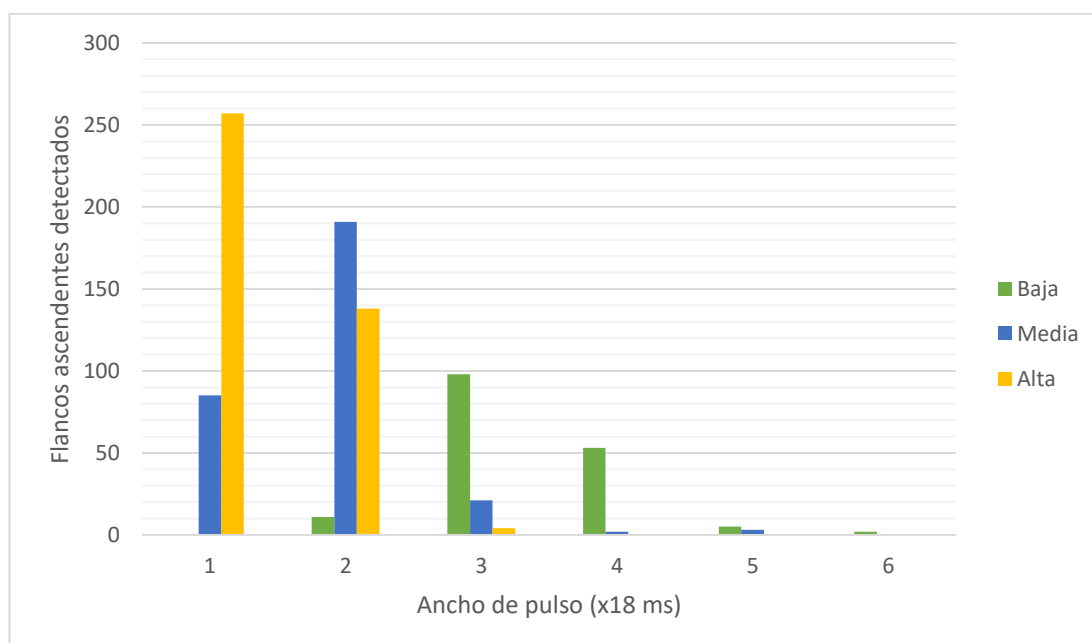


Figura 67. Resultados de pruebas de velocidad con sensores encoders.

Asimismo, se midió con 2 sensores encoders simultáneamente sobre un mismo motor, tomando un total de 2000 muestras para verificar que el comportamiento de los sensores es el mismo independientemente del sensor usado y descartar posibles errores por defectos de los sensores.

Al igual que en la prueba anterior se obtuvo una onda cuadrática o un tren de pulsos de los 2 sensores evaluados, mediante una tarjeta Arduino Nano a una velocidad angular máxima de 111.3 rpm. Los resultados finales se observan en la **Tabla 20**.

Tabla 20.
Resultados de pruebas con 2 sensores encoders simultáneamente a velocidad máxima.

Ancho de pulso (x18ms)	Número de flancos ascendentes detectados	
	Sensor 1	Sensor 2
1	405	402
2	16	9
3	3	2
Total de Pulsos	424	413

Se evidencia que existe una diferencia en el conteo de flancos ascendentes detectados igual a 11 pulsos. Esto quiere decir que la incertidumbre por cada 38 flancos detectados es de ± 1 flanco ascendente detectado. En la **Figura 68** se aprecia mejor la diferencia entre ambos sensores.

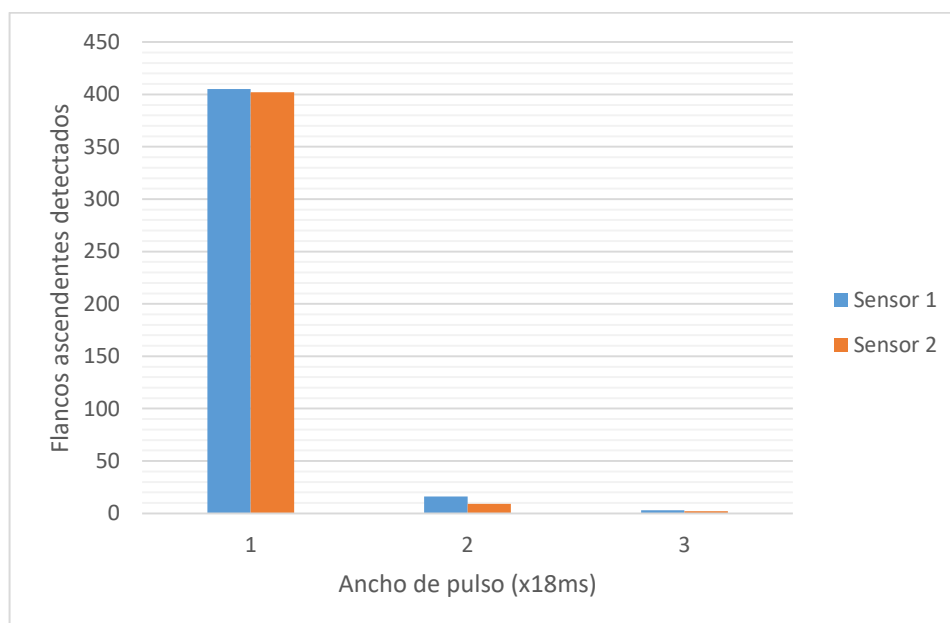


Figura 68. Resultados de pruebas con 2 sensores encoders simultáneamente

Finalmente, una vez implementado los sensores encoders en cada robot, e implementado el código para calcular el recorrido mostrado en el punto 4.4.3 de este trabajo, se experimentó realizando movimientos rectilíneos sobre una superficie plana para calcular el error de recorrido en los tres agentes robóticos. Desde la **Figura 69** hasta la **Figura 71** muestra los resultados obtenidos para el robot 1 con un total de 10 series de recorridos lineales realizadas para cada prueba.

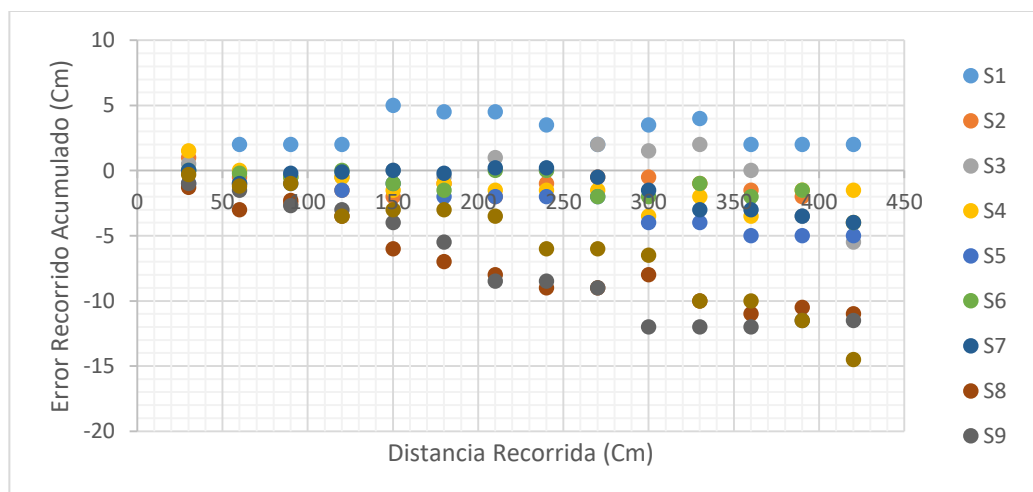


Figura 69. Pruebas realizadas con recorridos de 30 cm consecutivos hasta los 420 cm.

En la **Figura 69**, se puede observar que a partir de la serie de recorrido S8, el error aumento considerablemente, debido al desgaste de potencia en la batería utilizada. Esto quiere decir que el recorrido óptimo de trabajo de cada agente es menor a 29.4 mt.

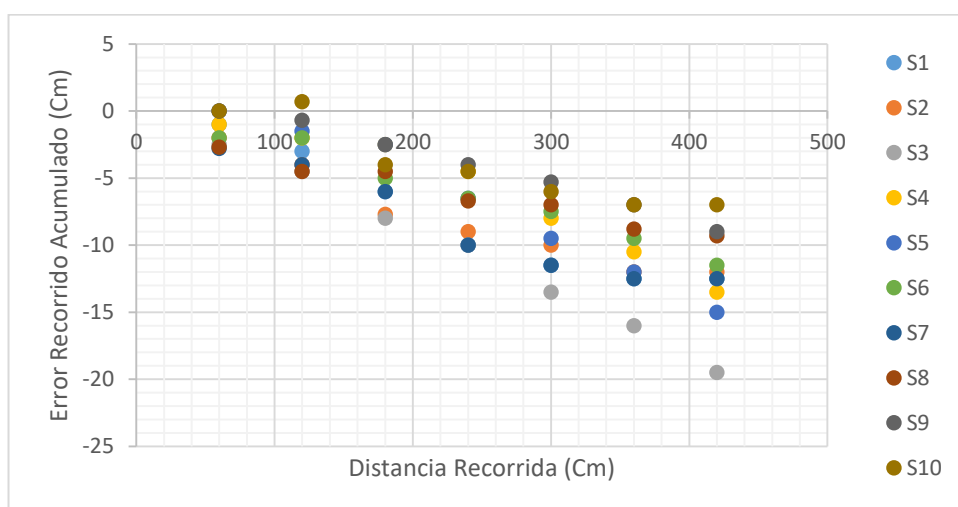


Figura 70. Pruebas realizadas con recorridos de 60 cm consecutivos hasta los 420 cm.

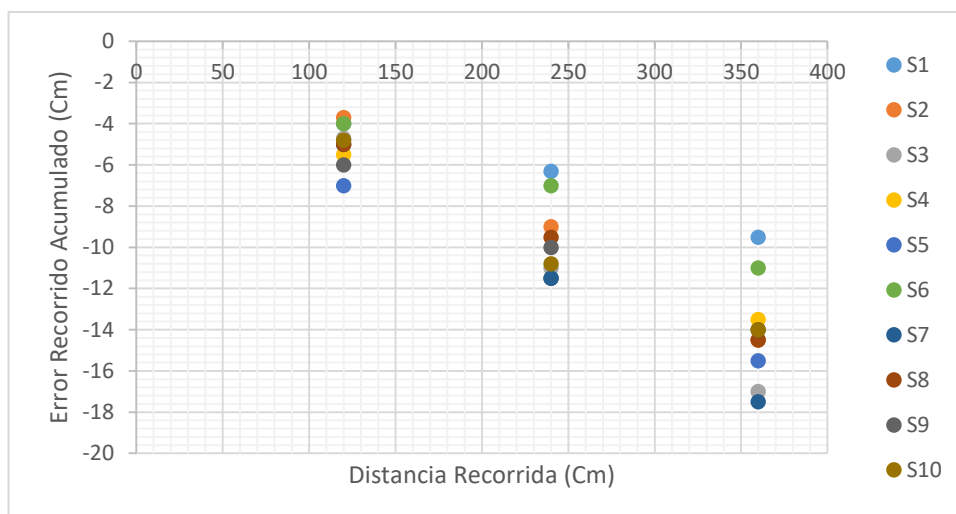


Figura 71. Pruebas realizadas con recorridos de 120 cm consecutivos hasta los 360 cm.

Las **Figura 70** y **Figura 71**, muestran que con el mismo controlador de recorrido el error aumenta al desplazarse en distancias más largas, debido a la acumulación de errores sistemáticos y no sistemáticos en cada plataforma robótica.

El mismo experimento se realizó en los 2 robots restantes obteniendo en resumen las siguientes gráficas comparativas del error medio de recorrido en los 3 robots móviles diseñados en esta investigación:

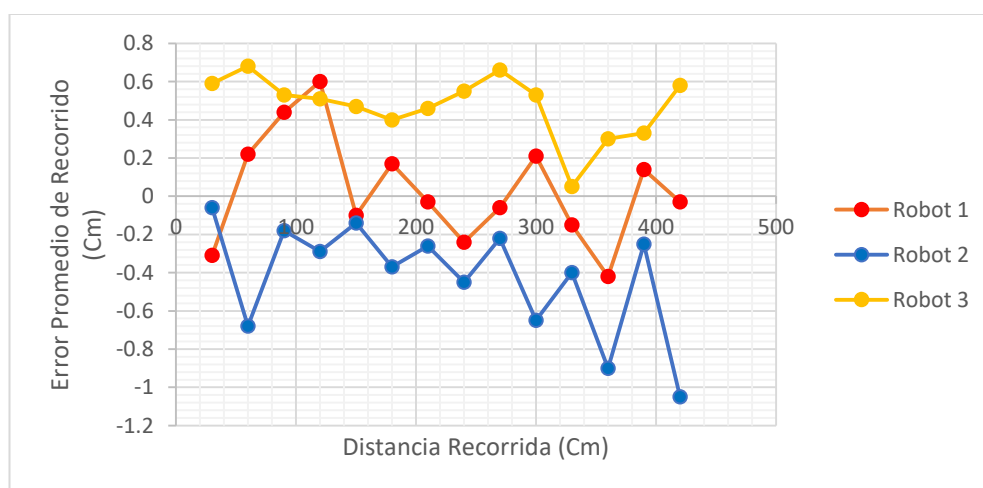


Figura 72. Comparación entre robots móviles del error promedio de recorrido cada 30 cm.

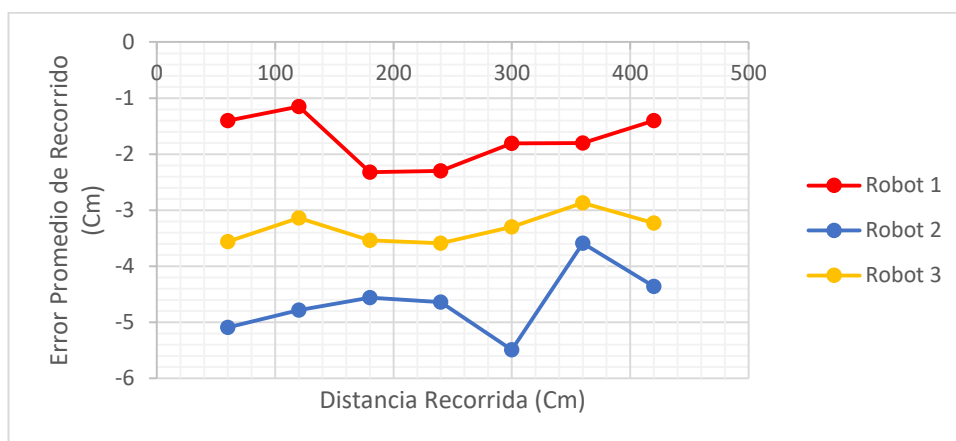


Figura 73. Comparación entre robots móviles del error promedio de recorrido cada 60 cm.

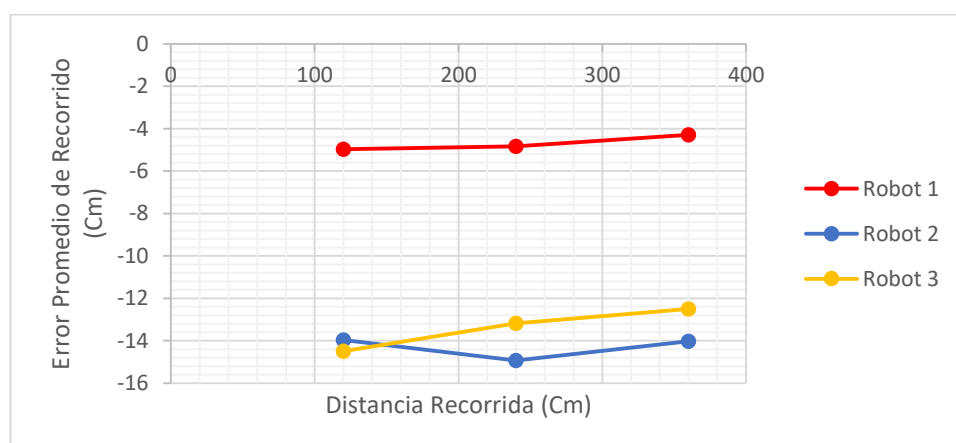


Figura 74. Comparación entre robots móviles del error promedio de recorrido cada 120 cm.

De las tres figuras, se puede concluir que los errores sistemáticos principalmente, son menores en el agente denominado Robot 1, lo que permite que el error disminuya al aumentar su recorrido en comparación a los 2 agentes restantes. Además, la **Figura 72** muestra que los 3 agentes mantienen un error menor a 1 cm durante todo su trayecto cuando se mantiene recorridos de 30 cm debido a que el controlador fue diseñado sobre estas condiciones.

5.2.3 Pruebas con brújulas electrónicas

Las brújulas electrónicas fueron calibradas y probadas de forma estática mostrando una incertidumbre de $\pm 1^\circ$ en todas las direcciones. Una vez implementadas las brújulas electrónicas en las plataformas se percibió que el error aumentaba o disminuía dependiendo de la posición y dirección del agente robótico. Basado en este

anteriormente, se decidió probar el efecto de distorsión de la dirección de los agentes robóticos por el ruido magnético del ambiente. Para este cometido, se colocó cada robot en 4 distintas direcciones (0° , 90° , 180° , 270°) en 38 distintas posiciones iniciales del área a explorar, realizando recorridos rectilíneos. En las Figuras 75-78 se muestra los resultados obtenidos por la brújula electrónica del robot 1 comparados con las direcciones y posiciones reales.

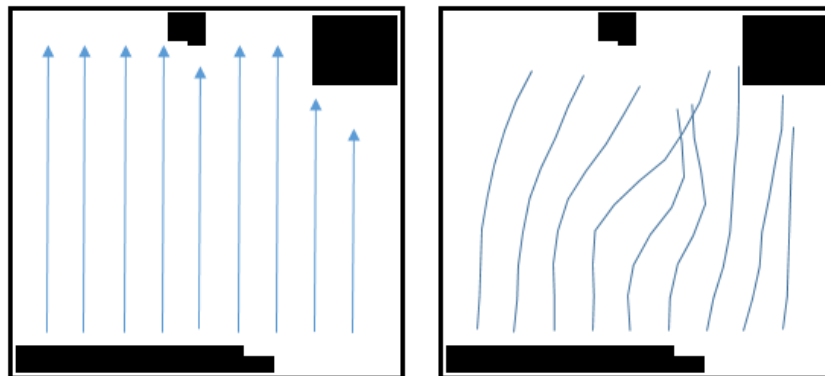


Figura 75. a) Recorrido real del agente robótico a 0° grados de dirección. b) Recorrido simulado por la brújula electrónica.

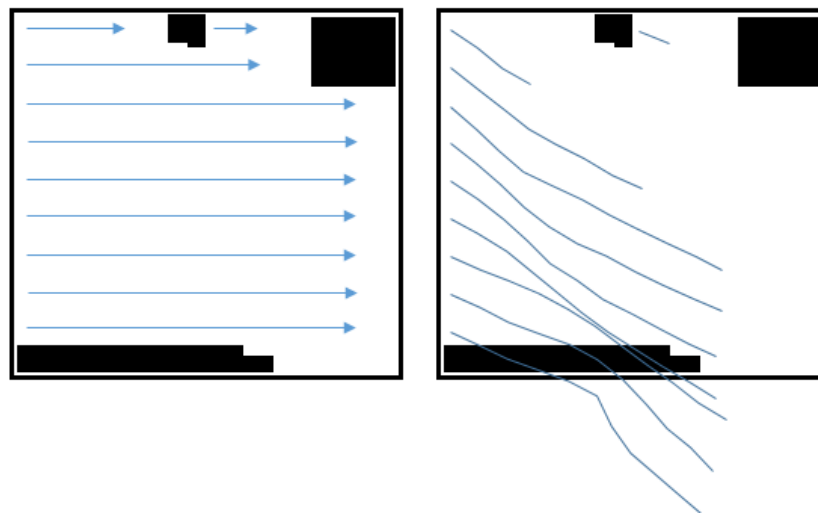


Figura 76. a) Recorrido real del agente robótico a 90° grados de dirección. b) Recorrido simulado por la brújula electrónica.

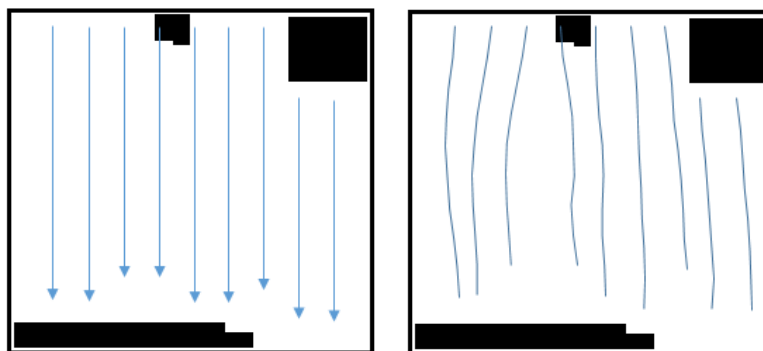


Figura 77. a) Recorrido real del agente robótico a 180 grados de dirección. b) Recorrido simulado por la brújula electrónica.

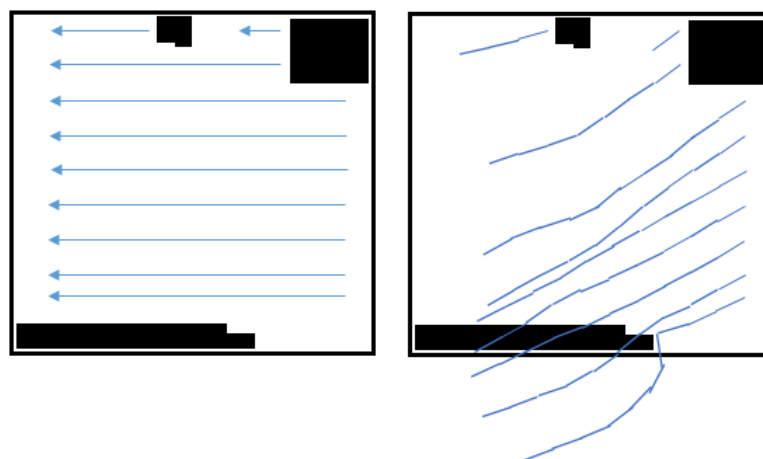


Figura 78. a) Recorrido real del agente robótico a 270 grados de dirección. b) Recorrido simulado por la brújula electrónica.

Se realizó la misma prueba en los 2 agentes robóticos restantes con resultados similares, donde se puede apreciar que existe una inducción magnética con origen cercano a la esquina inferior derecha del entorno de pruebas. La desviación media de dirección es de 32 grados, sin embargo éste valor incrementa o disminuye considerablemente en relación a la dirección inicial real del agente robótico, por ejemplo en la **Figura 77** la desviación máxima es de $\pm 10^\circ$. Tomando en cuenta estos resultados se realizó una corrección en los valores simulados en el controlador central de forma que compense el ruido magnético encontrado en el ambiente.

5.3 Pruebas de funcionamiento del sistema MRSLAM

El sistema MRSLAM desarrollado en este trabajo tiene como propósito crear un mapa de obstáculos y de niveles de temperaturas en un área específica con 3 robots móviles coordinados mediante un controlador central. Sobre este concepto se

realizaron varios experimentos que demuestren la eficiencia del sistema multi robótico implementado.

Los parámetros de evaluación considerados son:

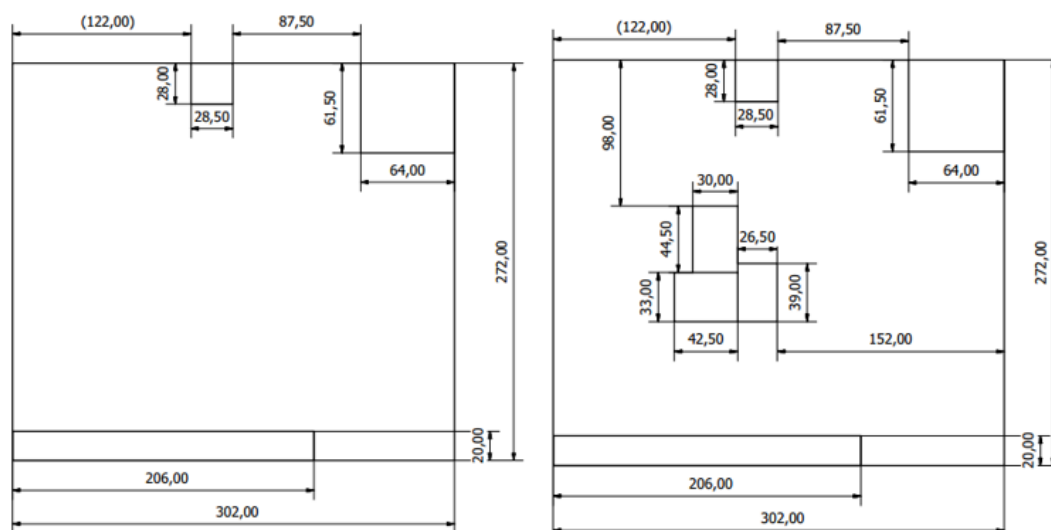
- El tiempo de trabajo del sistema MRSLAM.
- El recorrido total realizado por los 3 robots móviles.
- La correlación del mapa real con el mapa creado por el sistema MRSLAM.

Las pruebas realizadas tienen como objetivo evaluar el comportamiento de uno, dos y tres robots funcionando simultáneamente para mostrar los efectos en el comportamiento del sistema MRSLAM diseñado. Adicionalmente, se analiza el mapa de temperaturas en relación a la fuente de calor y la temperatura ambiente de cada puesta en marcha.

Los dos escenarios controlados fueron construidos en una habitación cerrada con las siguientes características:

- Área de mapeado de 82144 cm².
- Superficie plana de baldosa lisa.
- Carecen de influencia de corrientes naturales de aire.
- Poseen una corriente constante de aire caliente oscilante entre los 25 y 30 grados Celsius en una posición fija.
- Carecen de obstáculos dinámicos dentro del espacio navegado.

Los dos escenarios usados se distinguen por la disposición de los obstáculos:



**Figura 79. a) Primer escenario de obstáculos usado para pruebas de mapeo.
b) Segundo escenario de obstáculos usado para pruebas de mapeo.**

En ambos ambientes la fuente de calor se colocó en la esquina superior izquierda, generada por un calefactor pequeño con un ventilador.

5.3.1 Evolución del sistema MRSLAM durante las pruebas

Mientras se realizaban las pruebas fue sufriendo modificaciones el sistema de control y la interfaz del sistema MRSLAM desarrollado en este proyecto de investigación. Por tal motivo, se presenta un resumen cronológico de los cambios realizados y el resultado final se presentara en el siguiente apartado.

Reducción de área de trabajo

El primer cambio notorio realizado, estuvo relacionado al espacio a mapear, el cual fue reducido aproximadamente a la mitad, dando como resultado los escenarios de la **Figura 79**. Las razones que motivaron a tomar esta decisión estaban ligadas a la dificultad por calibrar la brújula en toda el área, el tiempo de mapeado necesario para concluir con la tarea en cada prueba, y la reducción de datos a procesar.

En la **Figura 80** se puede evidenciar el efecto de la calibración de la brújula electrónica con la información del apartado 5.2.3, reduciendo el espacio de trabajo de los agentes robóticos.

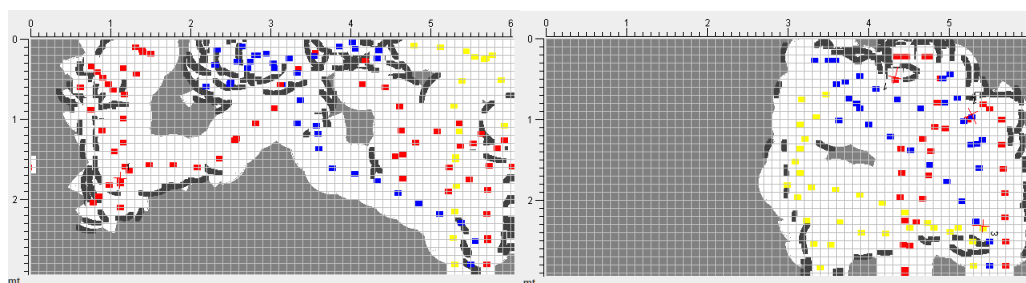


Figura 80. a) Mapa de obstáculos creado por 3 agentes robóticos antes de la calibración de dirección. b) Mapa de obstáculos creado por 3 agentes robóticos después de la calibración de dirección.

Recordando que, el color gris en el mapa representa el estado inicial desconocido de cada celda de los mapas. Por otra parte, las celdas blancas simbolizan los espacios vacíos, las celdas negras los espacios ocupados y los puntos rojos, azules y amarillos representan las estimaciones históricas de posición del robot 1, robot 2 y robot 3 correspondientemente.

En la **Figura 80** se observa que la distorsión producida por la adquisición de direcciones erróneas es significativa y crea un mapa incongruente a la realidad, debido

a fuentes electromagnéticas cercanas existentes en el entorno donde se realizaron las pruebas. Una vez identificadas y corregidas las variaciones indeseadas de dirección en la adquisición de datos, se puede apreciar que el mapa creado se aproxima bastante a la realidad.

Pruebas de recorrido

Al corregir el problema de las direcciones, permitió realizar nuevos experimentos, como el efecto del recorrido en la distorsión del mapa creado.

Se puede ver en la Figura 81 que a mayor recorrido el mapa va perdiendo información y creando falsos obstáculos que entorpecen el direccionamiento por parte del controlador central, es decir, el código no encuentra una nueva posición para explorar y por lo general el agente entra en un bucle en cierto espacio del mapa. Estos resultados muestran que el error producido por los sensores encoders analizado en el apartado 5.2.2 es tolerable hasta los 20m de recorrido por cada robot móvil.

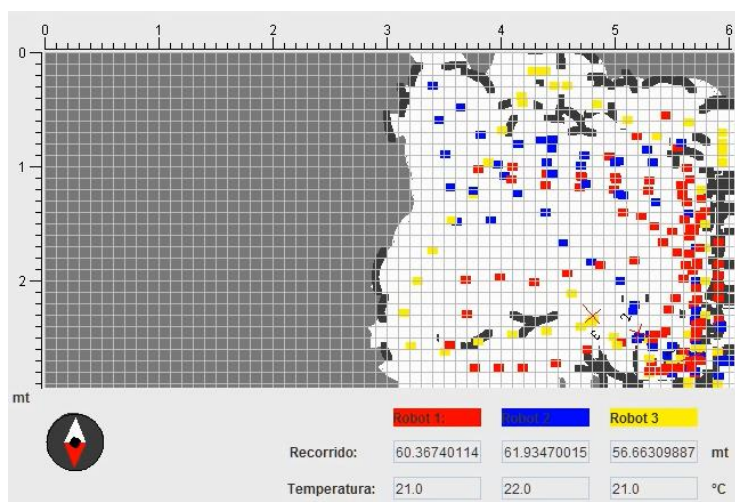


Figura 81. Mapa de obstáculos creado con 3 agentes robóticos con recorridos superiores a los 55mt.

También se evidenció que existen fuentes electromagnéticas cambiantes en el entorno que producían que el mapa creado varié según la intensidad del ruido magnético, por ejemplo, encendido o apagado de instrumentación electrónica en laboratorios vecinos, motores, televisores, generadores, etc.

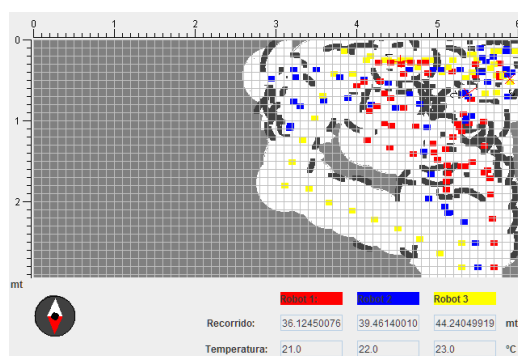


Figura 82. Mapa de obstáculos creado con 3 agentes robóticos con excesivo ruido magnético ambiental.

Para el alcance de esta investigación dichos efectos no son corregidos, pero son minimizados realizando pruebas fuera del horario laboral en donde es más probable la aparición de dichas inductancias indeseadas.

Variaciones de la interfaz gráfica

Una vez definida el área de reconocimiento de los agentes robóticos, se realizó cambios en la interfaz gráfica incluyendo ajustes del tamaño del panel, mayor contraste de colores, puntos de rastro identificadores de las rutas de cada robot más discretos, y reconocimientos de espacios vacíos.

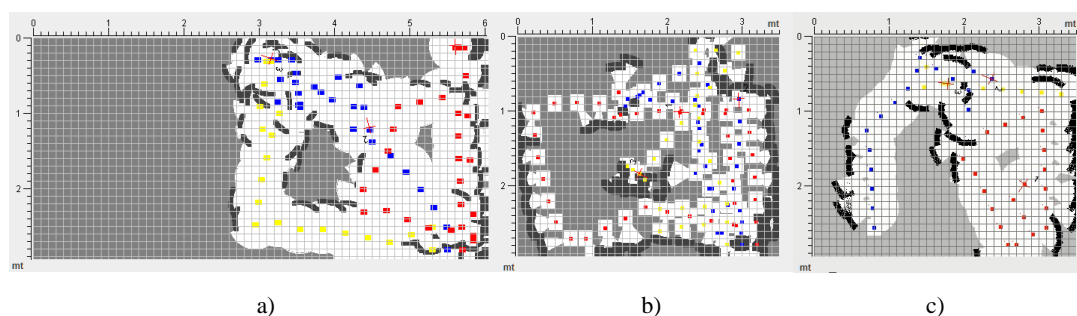


Figura 83. Progreso del panel del mapa de obstáculos en la interfaz gráfica del sistema MRSLAM.

En el mapa de la **Figura 83 a)** existe un gran espacio desconocido a la izquierda del panel, debido a la reducción del espacio real a mapear. Igualmente, se genera falsos reconocimientos de espacios vacíos detrás de los obstáculos detectados, esto se debe a la interpolación de la información en cada paso de los robots móviles.

En la **Figura 83 b)** se planteó la simulación de obstáculos con menor resolución e identificar un espacio vacío solamente si el robot se encontraba en dicha posición o si se detectaba un espacio vacío entre el obstáculo y el agente robótico. De manera experimental se obtuvo perdida en la capacidad de mapeado de los agentes, esto

significa que necesitaban mayor tiempo y recorrido para cubrir un área y paralelamente se desperdiciaba la información detectada por los sensores ultrasónicos al girar 360 grados.

Así se llegó a obtener un mapa como el de la **Figura 83 c)**, donde se aprovecha toda la información de los agentes, descartando los posibles falsos espacios vacíos detectados por el ángulo de incidencia de los sensores ultrasónicos. Permitiendo aumentar la resolución de las celdas detectadas como obstáculos y disminuir los tiempos de mapeado.

Posición inicial de los agentes robóticos

El siguiente cambio importante realizado tiene relación con la posición inicial de los tres agentes robóticos.

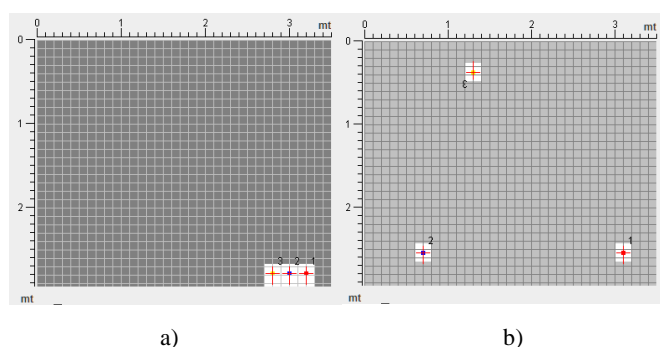


Figura 84. Variación de las posiciones de los 3 agentes robóticos en el espacio a explorar.

La versión anterior del código desarrollado asumía que los 3 agentes empezaban el recorrido en la esquina inferior derecha del mapa en la misma dirección como en la **Figura 84 a)** y se dispersaban al comenzar a movilizarse. Esto fue remplazado por las posiciones de la **Figura 84 b)** donde la dirección del robot 3 es contraria y como ya están dispersos sus direcciones iniciales permanecen constantes hasta encontrar un primer obstáculo en el frente. Esta medida fue tomada por las siguientes razones:

- Mayor dispersión de la información obtenida
- Reducción de tiempos de mapeado
- Menor interpolación de mapas de cada agente
- Las distintas posiciones y direcciones permiten realizar más tipos de pruebas, por ejemplo, algoritmos antichoque, el efecto de la brújula electrónica en diferentes direcciones y posiciones iniciales, planteamiento de estrategias iniciales para mejorar la eficiencia del trabajo coordinado de los agentes robóticos, etc.

Implementación de filtros espaciales

Finalmente, para mejorar la visualización del mapa de obstáculos creado, se desarrolló un filtro espacial lineal simple de media para los espacios desconocidos al finalizar el mapeado de los agentes robóticos, es decir las celdas desconocidas en el mapa final se asumen como un obstáculo o como un espacio vacío según el nivel de pertenencia a estos grupos, dado por el valor de los pixeles vecinos ha dicho espacio estimado como se diseñó en el tema del punto 4.3.4.

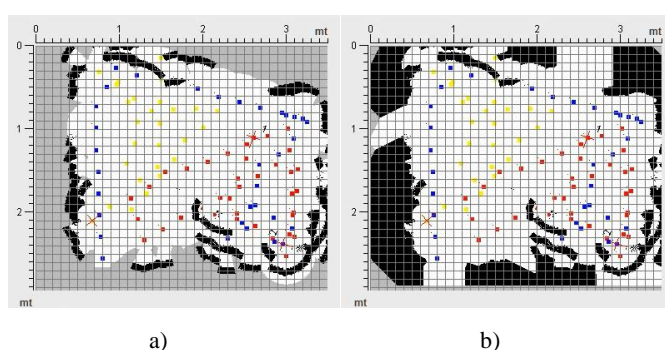


Figura 85. Mapa de obstáculos antes y después del filtro de media.

En la **Figura 85 a)** se observa el mapa creado por los tres agentes robóticos sin el filtro espacial de media, con espacios inexplorados por su posición fuera del alcance de las plataformas móviles, mientras que una vez implementado el filtro como se observa en la **Figura 85 b)**, se tiene una visión más objetiva sobre el mapa creado. En este ejemplo existen áreas detectadas como espacios vacíos que en realidad son obstáculos, debido principalmente a la resolución de los sensores ultrasónicos, y la baja afluencia de los robots en dichos sectores.

Por otro lado, se usa asimismo un filtro lineal de media en el mapa de temperaturas para simular los posibles valores en las posiciones no alcanzadas del mapa una vez finalizado el mapeado.

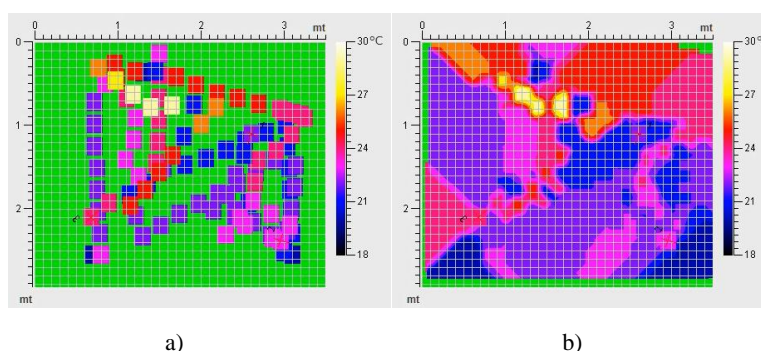


Figura 86. Mapa de temperaturas antes y después del filtro de media.

A diferencia del filtro aplicado en el mapa de obstáculos, los valores originales no se mantienen en el mapa final. Como se observa en la **Figura 86 b)** se crea un degradado en los bordes de acuerdo a la información que existe a su alrededor, creando una imagen más acorde a la realidad sin las fronteras solidas que se crea por las posiciones donde se censo las temperaturas mostradas en la **Figura 86 a)**.

5.4 Resultados del sistema MRSLAM

Una vez realizadas las pruebas y correcciones en la interfaz, el resultado final del sistema MRSLAM implementado muestra una respuesta favorable de acuerdo a los objetivos planteados en este trabajo. Las pruebas finales realizadas fueron ordenadas según lo indica la **Figura 87**.

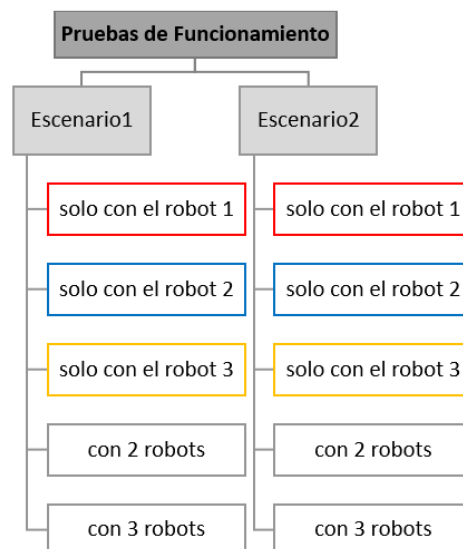
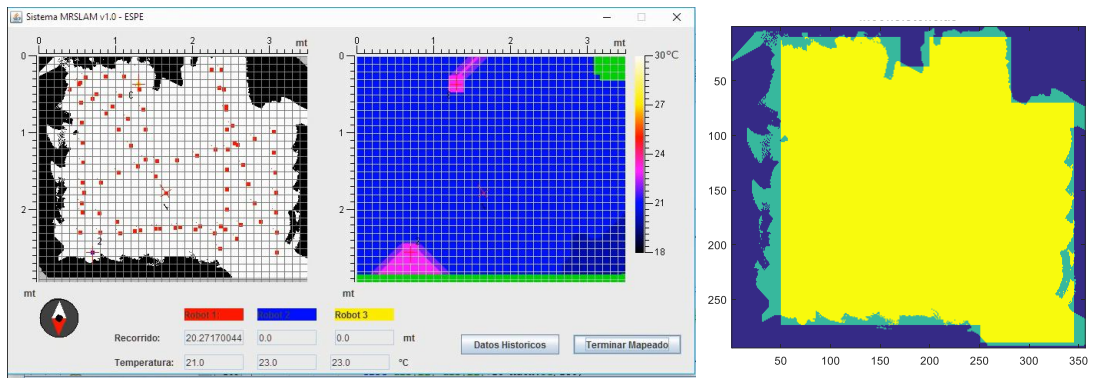


Figura 87. Clasificación de las pruebas de funcionamiento realizadas.

Para apreciar mejor los resultados se decidió transferir la información del mapa de obstáculos creado al finalizar el mapeado hacia la plataforma MatLab, esto permite compararlo con un mapa de obstáculos ideal de los dos escenarios implementados.

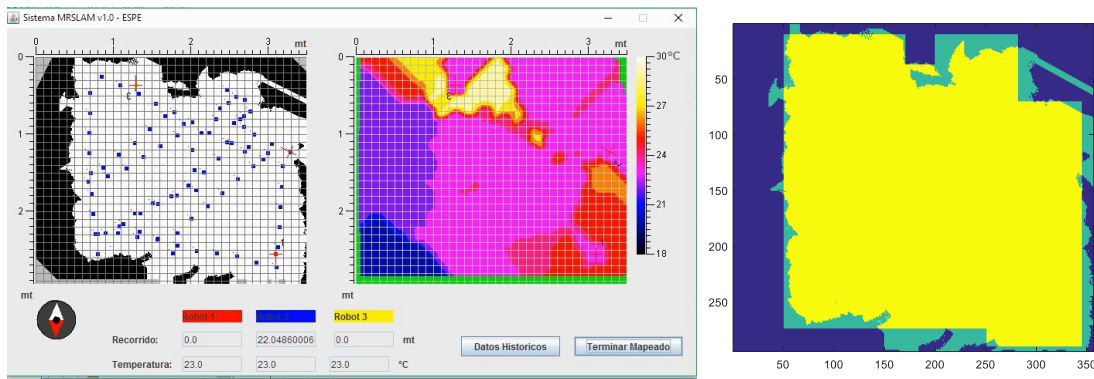
El script desarrollado se encarga de adquirir la matriz del mapa de obstáculos y realizar una comparación con una matriz simulada del entorno ideal mediante el cálculo del coeficiente de correlación. Adicionalmente, el programa compara celda por celda si su valor es igual en el mapa adquirido que en el mapa ideal para finalmente obtener un porcentaje de acierto del sistema MRSLAM. Los resultados son graficados de tal manera que se pueda validar el sistema en funcionamiento. Las Figuras 88-97 muestran los resultados obtenidos de las pruebas anteriormente mencionadas.



a)

b)

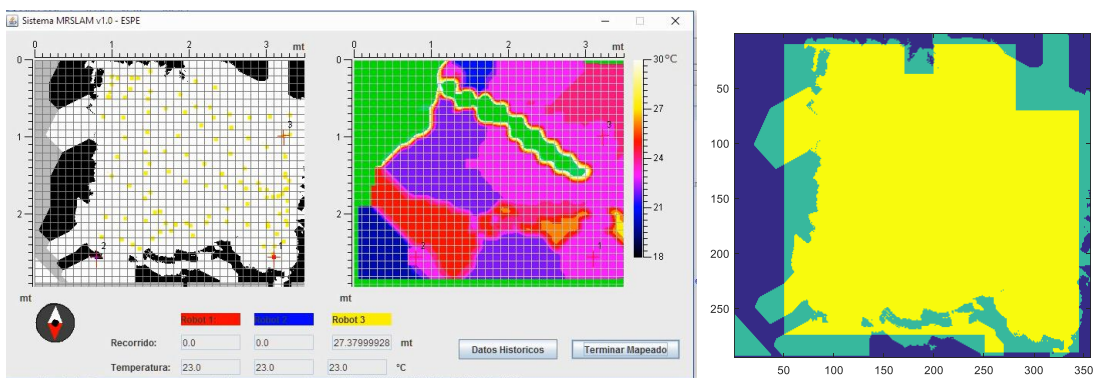
Figura 88. Resultados de prueba en el primer escenario con el robot 1 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real.



a)

b)

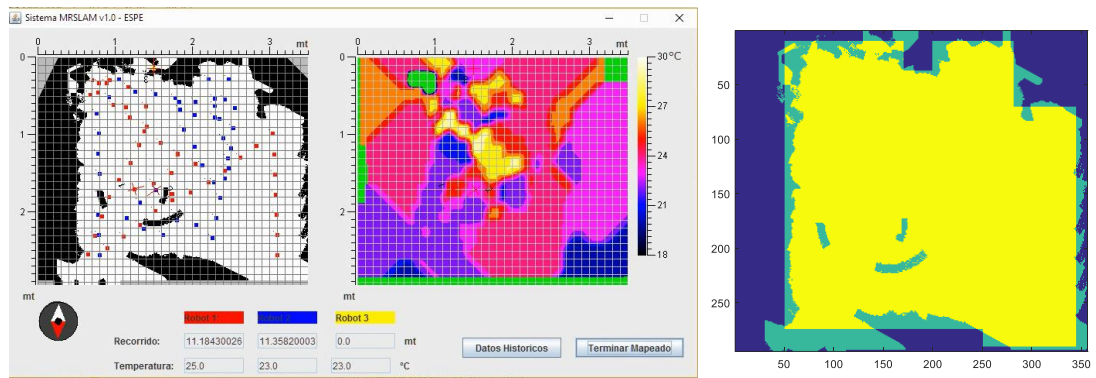
Figura 89. Resultados de prueba en el primer escenario con el robot 2 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

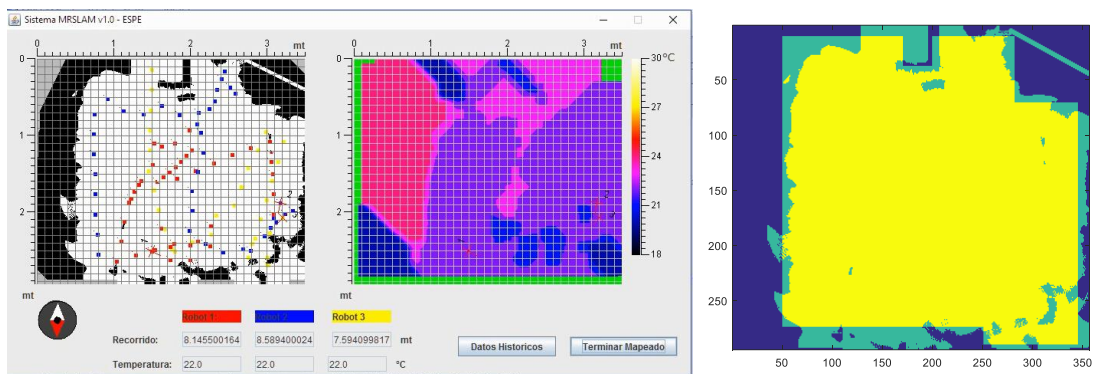
Figura 90. Resultados de prueba en el primer escenario con el robot 3 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

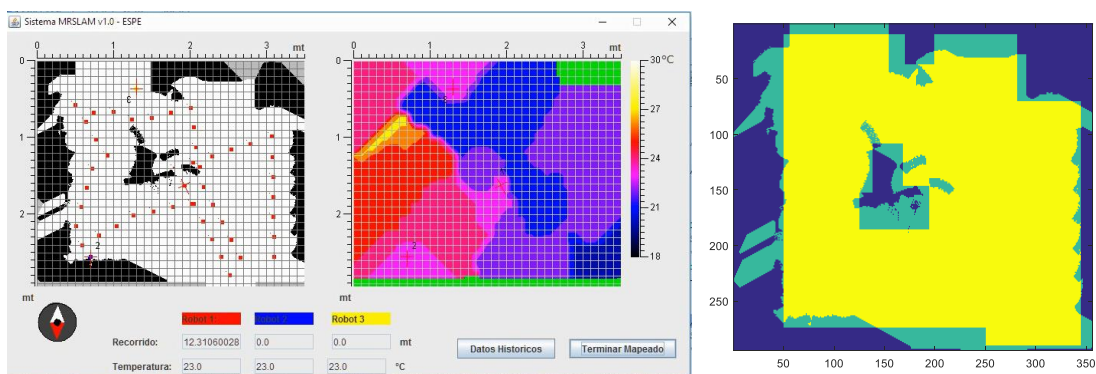
Figura 91. Resultados de prueba en el primer escenario con 2 agentes robóticos. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

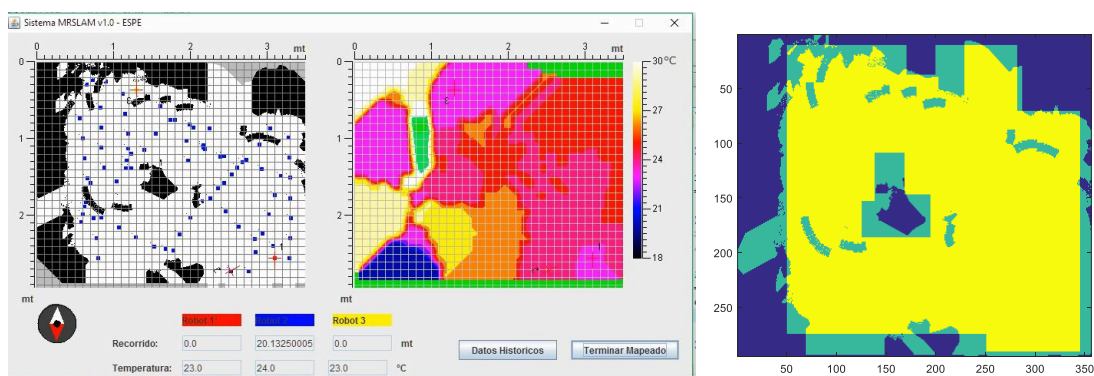
Figura 92. Resultados de prueba en el primer escenario con 3 agentes robóticos. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

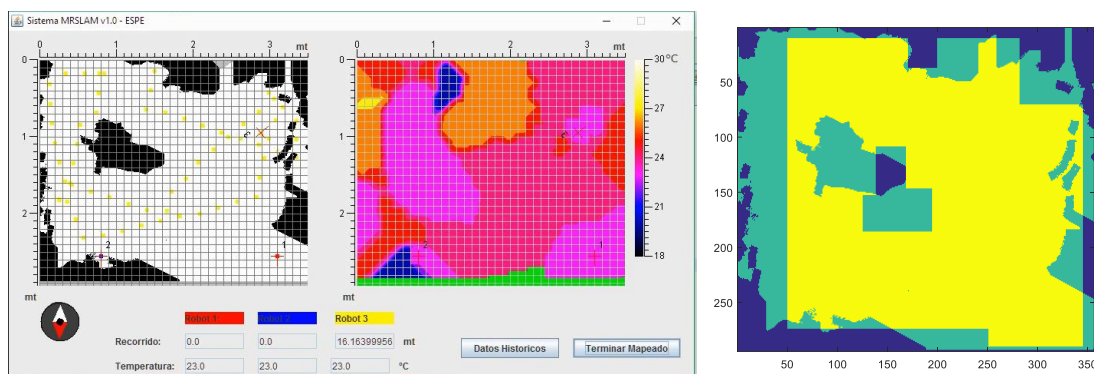
Figura 93. Resultados de prueba en el segundo escenario con el robot 1 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

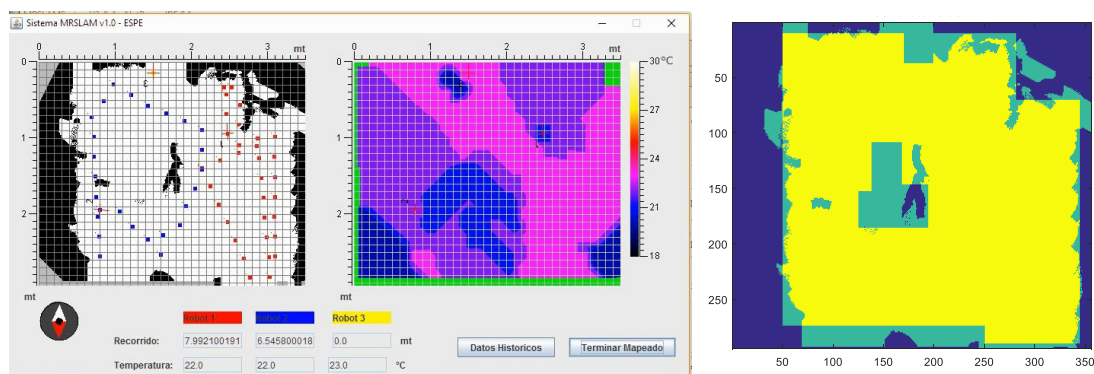
Figura 94. Resultados de prueba en el segundo escenario con el robot 2 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

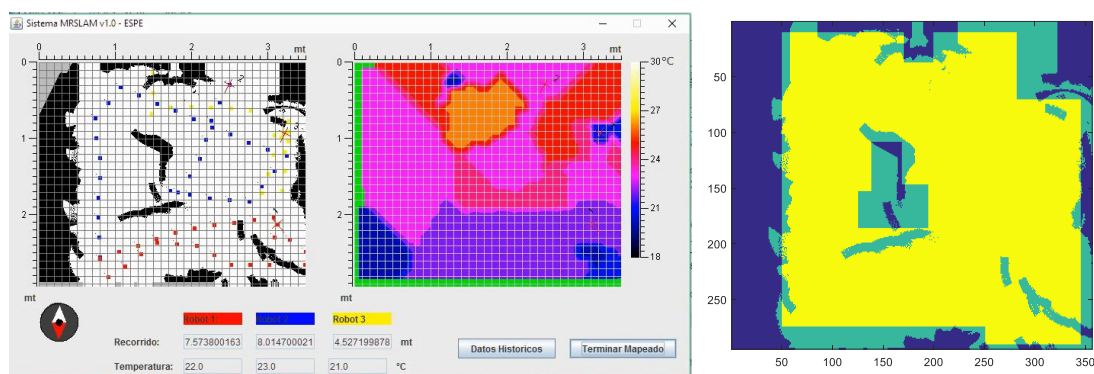
Figura 95. Resultados de prueba en el segundo escenario con el robot 3 únicamente. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

Figura 96. Resultados de prueba en el segundo escenario con 2 agentes robóticos. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real



a)

b)

Figura 97. Resultados de prueba en el segundo escenario con 3 agentes robóticos. a)Registro obtenido en la interfaz al finalizar el mapeado. b)Simulación en MatLab de mapa de obstáculos sobreponiendo el mapa real

Estos resultados muestran en las figuras de la izquierda los mapas de la interfaz del sistema MRSLAM al finalizar el mapeado y las figuras de la derecha muestran la información del mapa de obstáculos comparada con el mapa ideal mediante Matlab. En estos gráficos, el color azul representa las zonas donde coincide la información sobre los obstáculos de ambos mapas, mientras que el color amarillo identifica las zona donde coincide los espacios vacíos detectados con el mapa ideal y el color verde se usa para identificar las zonas donde no coincide la información de los mapas comparados.

En la

a)

b)

Figura 88, debido a la precisión de este agente demostrada con las pruebas realizadas en el punto 5.2.2, se observa que el resultado se aproxima mejor a la realidad comparado con los mapas creados por los otros robots móviles. Este mapa fue realizado sin la fuente de calor presente, por lo que el mapa de temperaturas muestra una temperatura uniforme en todo el área de trabajo, excepto en las posiciones de los dos robots restantes no implementados debido a que son inicializados con un valor de 23 grados centígrados y el algoritmo de control asume como verdadera esta temperatura ya que la interfaz está diseñada para trabajar con los 3 agentes robóticos simultáneamente. Por otra parte en la

b)

Figura 89, se visualiza claramente la posición de la fuente de calor, y el efecto que causa en el mapa de temperaturas creado.

El mapa de obstáculos de la

a)

b)

Figura 90, comparado con los resultados de los mapas individuales del resto de robots, muestra una diferencia considerable respecto al mapa ideal. Esto debido principalmente a los errores de los sensores y la desalineación de las ruedas de la plataforma móvil utilizada. Otro efecto visual destacado en esta figura es el color verde

en el mapa de temperaturas, esto ocurre en la zona donde se detectó temperaturas mayores a 30 grados centígrados. Debido a que la escala de temperaturas trabaja en un rango entre los 18 y 30 grados centígrados y valores mayores o menores a este rango son considerados como desconocidos. Esto con el fin de resaltar las variaciones más pequeñas de temperatura, es decir si aumentamos el rango de temperaturas, las variaciones de color serían más tenues y por lo tanto menos notorias. Sin embargo el código permite modificar este rango de temperaturas para que se pueda acoplar a las necesidades de la exploración.

Los agentes robóticos cuando trabajan simultáneamente se convierten en obstáculos dinámicos del sistema no considerados en el código desarrollado. A pesar que el algoritmo del controlador central considera una distancia restringida entre agentes robóticos, los sensores ultrasónicos detectan la presencia del agente cercano y crea falsos obstáculos en el mapa como los que se visualizan cerca de las posiciones finales de los robots móviles en la

Figura 91 a).

En la **Figura 92**, al haber más información dispersa, se crea mayor cantidad de falsas detecciones de espacios vacíos, por la interpolación de los mapas individuales. También se observa que el mapa de niveles de temperaturas es más uniforme debido a las bajas temperaturas ambientales del día y la hora de la prueba realizada. Un resultado similar se obtuvo en la

Figura 93. De forma opuesta, en la

Figura 94 y

Figura 95 se realizó el experimento a medio día donde la temperatura ambiente es mayor lo que produce un mapa con colores más cálidos.

A partir de la **Figura 94**, donde muestra los resultados en el segundo escenario propuesto, se observó que el sistema de sensado mediante ultrasónicos tiene inconvenientes en reconocer esquinas debido a la resolución normal de los sensores, lo que ocasiona que el mapa en general se distorsione y sea menos efectivo el sistema desarrollado. Por esta misma razón y por la interpolación de los tres mapas individuales, en la

Figura 97 donde actúan los tres agentes robóticos simultáneamente en el segundo escenario, se obtiene una respuesta menos exacta y produce que el mapa final se distorsione con mayor facilidad. Asimismo se obtuvo en ocasiones desfases del mapa creado como el de la

Figura 95, debido a pérdidas de comunicación o errores de adquisición de los encoders.

En resumen los datos cualitativos obtenidos por los experimentos antes mencionados, se detallan en la **Tabla 21** y **Tabla 22**.

Tabla 21.

Resultados de las pruebas de funcionamiento en el primer escenario.

	Tiempo	Recorrido (mt)	Coefficiente de Correlación	Porcentaje de acierto	Eficiencia
Con el robot 1	8min 22s	20.27	0.77	91%	56.6%
Con el robot 2	9min 03s	22.04	0.78	91%	48.8%
Con el robot 3	13min 57s	27.38	0.54	81%	17.63%
Con 2 robots	5min 05s	11.18	0.73	89%	80.7%
		11.35			
Con 3 robots	4min 48s	8.14	0.70	88%	74.8%
		8.58			
		7.59			

Tabla 22.

Resultados de las pruebas de funcionamiento en el segundo escenario.

	Tiempo	Recorrido (mt)	Coefficiente de Correlación	Porcentaje de acierto	Eficiencia
Con el robot 1	4min 51s	12.31	0.66	85%	52.8%
Con el robot 2	9min 10s	20.13	0.61	83%	15.8%
Con el robot 3	7min 54s	16.16	0.33	72%	12.3%
Con 2 robots	3min 20s	7.99	0.67	86%	66.1%
		6.54			
Con 3 robots	2min 48s	5.74	0.62	84%	92.5%
		4.60			
		1.10			

La eficiencia del sistema es un indicador porcentual que reúne todos los parámetros fundamentales del sistema y se calcula mediante un factor que relaciona el tiempo invertido, el recorrido real y el coeficiente de correlación alcanzado con los valores esperados.

$$Eficiencia = \frac{Y_{Alcanzado} / R_{Real} * T_{Invertido}}{Y_{Esperado} / R_{Estimado} * T_{Previsto}} \quad (28)$$

Donde Y es el coeficiente de correlación del mapa creado con el mapa real, R el recorrido total de los agentes robóticos y T el tiempo de funcionamiento del sistema MRSLAM desarrollado.

Este factor de eficiencia calculado para cada experimento en la **Tabla 21** y **Tabla 22** muestra cual fue la prueba con mejor eficiencia en ambos ambientes. Muestra que para el primer escenario, la prueba que tuvo mejor comportamiento fue con 2 robots móviles con una eficiencia del 80.7%, mientras que para el segundo escenario con 3 robots trabajando simultáneamente se obtuvo un mejor resultado, obteniendo una eficiencia del 92.5%. Estos valores son relativos, es decir no se relaciona la eficiencia del primer escenario con la del segundo y por ende no se puede decir que en el segundo escenario se obtuvo mejores resultados, simplemente muestra que dentro de los experimentos del segundo escenario se ajusta mejor a los resultados esperados.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

El problema en robótica denominado SLAM ha sido tratado de muchas formas, con distintos enfoques y para diferentes situaciones. El propósito final de todos estos estudios es la navegación autónoma para diferentes propósitos como la exploración espacial o submarina, la localización de objetos, sustancias o tipos de energía y el transporte de bienes o personas. En este trabajo específicamente se trató la problemática SLAM de forma elemental, para priorizar en primer lugar el diseño de la plataforma, el medio de comunicación y la interfaz de visualización de los mapas creados que son la base fundamental para iniciar la investigación en este campo.

Se presentó un enfoque sobre la exploración autónoma mediante múltiples robots que mediante una coordinación centralizada obtienen un mapa de obstáculos en 2D y un mapa de niveles de temperatura de un área específica. Este enfoque, ha demostrado mejorar en un 39.7% en cuanto a eficiencia que múltiples robots pueden aportar. Aunque el sistema tiene 3% menos precisión comparado con un solo agente robótico, la reducción de los tiempos de exploración en un 57% es el factor que determina la eficiencia del sistema MRSLAM.

Fundamentalmente, la perspectiva descrita en este trabajo está limitada bajo dos aspectos. En primer lugar, con respecto a la localización, actualmente el sistema supone que la posición inicial en el mapa creado virtualmente es la misma en el área real a explorar. Es necesario técnicas más complejas para la localización en donde su posición inicial es desconocida dentro del entorno a explorar y donde los robots necesitan conocer su ubicación con respecto al resto de robots. En segundo lugar, con respecto al mapeado, actualmente se considera suficiente y cierta la información proporcionada por cada agente en una posición determinada y que en ningún caso es necesario una verificación de la información en dicho sector. Por otro lado, si el área explorada es mayor a la proporcionada por la interfaz su información obtenida es ignorada si se encuentra en algún lugar fuera de los límites del panel creado.

6.1 Diseño e implementación de las plataformas móviles robóticas

Para la elección de las plataformas robóticas se analizó los variados tipos de plataformas terrestres existentes, concluyendo en primera instancia que la mejor solución para interiores son los modelos con ruedas por el mejor control odométrico que ofrecen. Dentro de los robots con ruedas, igualmente hay una amplia variedad de estructuras y configuraciones con distintas habilidades y propiedades que también fueron analizadas para finalmente elegir una plataforma comercialmente conocida como Robot 2WD de tipo diferencial por su estabilidad mecánica, maniobrabilidad y grados de libertad. Adicionalmente su arquitectura simple proporciona sencillez en el modelado cinemático y posterior tratamiento odométrico. No obstante, como ya ha sido mencionado los principales errores dentro del sistema están relacionados directamente con la construcción de las plataformas robóticas. Por lo que se recomienda mejorar la mecánica de la plataforma con el fin de disminuir el error por deslizamientos, desalineación de ruedas y variaciones en la posición inicial del servomotor de la torreta.

Sobre los actuadores de los robots, se utilizó 2 motores DC conectados directamente del eje reductor a la rueda y un micro servo para realizar un barrido circular de distancias y temperaturas con respecto al centro de cada robot. Los motores DC están conectados hacia un driver de motores denominado L298N, el cual proporciona la potencia necesaria para su funcionamiento.

En este trabajo se utiliza sensores ultrasónicos como detectores de obstáculos, encoders para el cálculo odométrico, brújulas electrónicas para complementar el cálculo de orientación y sensores de temperatura para analizar los niveles de calor en una posición.

En cada plataforma robótica móvil se decidió implementar dos tarjetas de programación. Una tarjeta Arduino Mega 2560 de propósito general para integrar y procesar la información sensorial; y una tarjeta Arduino Nano para la estimación de recorrido mediante los sensores encoders. Para la elección de ambas tarjetas programables se consideró el costo, confiabilidad y la posibilidad de expandir su uso a mayor escala electrónica.

Para la comunicación inalámbrica del sistema MRSLAM se consideró 3 diferentes tecnologías, Bluetooth, Wi-Fi y ZigBee; de estas se eligió la tecnología Wi-Fi por

ofrecer el mejor equilibrio entre coste, rango de alcance y tasa de transferencia. Para su uso y aplicación se creó puertas de enlace que proporciona un canal de comunicación entre un agente y el controlador central por el cual se puede transmitir en ambas direcciones pero no simultáneamente, por tal razón la comunicación es semi-duplex y unicast. Sin embargo, su aplicación más regular está enfocada para sistemas que requieran velocidades superiores a 250 Mbps de transferencia de datos entre 2 elementos por su topología tipo estrella. Lo cual limita las capacidades del sistema MRSLAM en cuanto a seguridad, confiabilidad y robustez del sistema de comunicación implementado. Una posible solución recomendada es ZigBee que a pesar de su alto coste según sus características es una solución apropiada para este tipo de trabajos.

La fuente de alimentación de cada robot elegida es una batería de polímero de iones de litio que entrega 7.4 V y tiene una capacidad de 1200mAh, que tiene 20% mas capacidad de acuerdo a los cálculos realizados sobre el consumo energético de cada agente robótico. Estas baterías cuentan con un conector de 2 pines JST-PH e incluye un circuito de protección que asegura que no existan picos de sobretensión al cargar la batería.

Se diseñó dos placas de circuito impreso que se adaptan a la forma y posición de los módulos implementados. Una de estas placas fue diseñada con el fin de montar una torreta en donde se disponga de 2 sensores ultrasónicos con direcciones inversas y dos sensores de temperatura dispuestos en los extremos de la placa. La segunda placa diseñada sirve para conectar y distribuir sobre la plataforma los elementos electrónicos usados. Incluidos un ultrasónico en el frente del robot móvil, una brújula con sus ejes paralelos a los ejes del robot y finalmente para la conexión del Wi-Fi con el Arduino Mega 2560 y la conexión del Arduino nano con los dos encoders utilizados.

6.2 Desarrollo de software para control y visualización MRSLAM

La interfaz gráfica diseñada muestra en tiempo real los cambios generados en los mapas de obstáculos y de niveles de temperaturas para visualizar de forma sencilla la posición, rutas y las detecciones sensoriales del entorno de cada agente robótico. Dicha interfaz fue desarrollada bajo el lenguaje de programación Java y el entorno de desarrollo NetBeans 8.1, logrando con ello obtener un código de fácil comprensión y completamente abierto para posibles actualizaciones del mismo. Las principales

funciones que cumple dentro del sistema MRSLAM son observar variables del sistema, el estado de los mapas creados y detener el proceso de mapeado.

La interfaz gráfica consta de 2 pantallas, la principal denominada pantalla de mapas, se caracteriza por dos grandes áreas que representan la zona a explorar desde una vista superior en 2D. En ellas, se representa el movimiento en tiempo real de cada plataforma robótica y la información adquirida en dicha estimación de posición mediante animaciones gráficas. La ventana secundaria, se denomina pantalla de históricos, es una ventana sencilla diseñada para mostrar la información textual obtenida por la estimación de posición de cada robot en un instante determinado de tiempo.

El control cooperativo centralizado coordina el movimiento de los tres agentes robóticos sobre el entorno a explorar para completar el mapa deseado. Las principales funciones son dispersar los robots y evitar las colisiones con obstáculos o entre robots. Con este fin, el controlador creado en la plataforma NetBeans 8.1 utiliza las estimaciones odométricas, de dirección y las distancias captadas por los sensores ultrasónicos de cada agente, para redireccionarlos a una nueva posición de interés y completar el mapeado de la forma más eficiente.

Al finalizar el mapeado por parte de los agentes robóticos móviles, el controlador central genera un mapa de obstáculos y de niveles de temperatura de acuerdo a la información almacenada en las respectivas matrices. Donde se implementó un filtro simple lineal de media en ambos mapas creados para las celdas aún desconocidas, creando una imagen completa del espacio explorado que cubre el 96% de los espacios desconocidos los resultados finales de los dos mapas.

El controlador independiente, denominado así por ser implementado en cada agente robótico de manera aislada, se encarga de realizar recorridos constantes de 30 cm rectilíneos hasta encontrar un obstáculo en el frente o detectar que se encuentra en una zona ya explorada. En cada recorrido realiza un sensado de 360 grados de los obstáculos cercanos y de la temperatura para posteriormente enviar esta información al controlador central para su visualización y posterior tratamiento.

Se utilizan dos sistemas de control en cada plataforma, dedicados al desplazamiento angular y rectilíneo de los agentes robóticos. En ambos casos se diseñó un controlador digital on-off de lazo abierto, implementado sobre el modelo cinemático presentado

en la ecuación 14. Este tipo de controlador es sensible a perturbaciones y posee limitaciones inherentes sobre la planta pero es una solución básica que no requiere un conocimiento complejo sobre las perturbaciones no sistemáticas tan variantes en sistemas móviles. Por tal motivo, se recomienda remplazar estos controladores por sistemas con una complejidad avanzada como redes neuronales para el aprendizaje de los agentes robóticos y la aplicación de controladores adaptativos que evolucionen respecto al terreno atravesado. Correspondientemente, debe generarse una interfaz gráfica lo suficientemente robusta para enfrentar las nuevas posibilidades del nuevo sistema multi robótico.

Los cambios sobre el software son posiblemente los primeros después de esta investigación, entre los primeros cambios se puede considerar la detección de obstáculos dinámicos mediante la confirmación de la posición de dichos obstáculos en distintos tiempos dentro de las funciones del control centralizado. Otra posible modificación recomendada sería implementar sistemas de filtrado para la estimación de posición de los agentes robóticos por medio de los filtros de Kalman, el método de localización de Monte Carlo o el filtro de partículas RaoBlackwellised.

6.3 Pruebas y resultados experimentales

Se mostró el funcionamiento del sistema multi robótico en dos diferentes escenarios, de forma que permitan evaluar el comportamiento de los robots móviles tanto grupal como individualmente. Las primeras pruebas realizadas se enfocaron en el correcto funcionamiento e interpretación de los sensores y actuadores de cada agente. Las pruebas en futuros trabajos se recomienda considerar el funcionamiento de los agentes en distintos tiempos para la creación de un mismo mapa, o el intercambio de posiciones entre robots móviles para visualizar como afecta en el mapa la precisión de cada robot móvil.

Los sensores ultrasónicos se probaron con 3 algoritmos distintos para medir distancias, con ello se obtuvo la exactitud y los tiempos de muestreo de cada versión. El algoritmo que se eligió es el denominado programa 1, debido a que muestra el mejor equilibrio en tiempos de procesamiento y precisión.

Los sensores encoders se probaron sobre tres distintas velocidades del disco codificador acoplado a uno de los motores DC de la plataforma robótica, donde los parámetros a evaluar son el ancho de los pulsos y el número de flancos ascendentes

detectados. Los resultados mostraron que la cantidad de flancos detectados en un instante de tiempo incrementa proporcionalmente a la velocidad administrada al motor de manera congruente a la realidad. También se midió con 2 sensores encoders simultáneamente sobre un mismo motor, tomando un total de 2000 muestras para verificar que el comportamiento de los sensores es el mismo independientemente del sensor usado y descartar posibles errores por defectos de fábrica de los sensores. De esta prueba se obtuvo que la incertidumbre por cada 38 flancos detectados es de ± 1 flanco ascendente detectado.

Una vez implementados los sensores encoders en las plataformas, se experimentó realizando movimientos rectilíneos sobre una superficie plana para calcular el error de recorrido, donde los resultados muestran que el error en las tres plataformas robóticas es menor a 1cm por cada recorrido constante de 30cm hasta alcanzar los 420cm. Sin embargo con el mismo controlador de recorrido el error aumenta 5 veces más al desplazarse en distancias más largas, debido a la acumulación de errores sistemáticos y no sistemáticos en cada plataforma robótica.

Las brújulas electrónicas fueron calibradas y probadas de forma estática mostrando una incertidumbre de $\pm 1^\circ$ en todas las direcciones. Una vez implementadas las brújulas electrónicas en las plataformas robóticas se percibió que el error fluctúa entre $\pm 32^\circ$ dependiendo de la posición y dirección del agente robótico. Basado en este antecedente, se decidió probar el efecto de distorsión de la dirección de los agentes robóticos por el ruido magnético del ambiente. Tomando en cuenta estos resultados se realizó una corrección en los valores simulados en el controlador central de forma que compense el ruido magnético encontrado en el ambiente, reduciendo el error a $\pm 10^\circ$.

Las pruebas de funcionamiento final del sistema MRSLAM se realizaron para evaluar el comportamiento de uno, dos y tres robots funcionando simultáneamente con el sistema de exploración diseñado. Por otra parte, se analiza el mapa de temperaturas en relación a la fuente de calor y la temperatura ambiente cuando se pone en marcha el mapeado. Estas pruebas se desarrollaron sobre 2 escenarios construidos en una habitación cerrada, ambos escenarios tienen un área plana de 82144 cm² y se distinguen por la disposición de los obstáculos.

Los resultados del mapa de obstáculos creado al finalizar el mapeado se transfirieron hacia la plataforma MatLab en forma de matrices, esto permite

compararlo con una matriz simulada del mapa de obstáculos ideal de los dos escenarios implementados y obtener un coeficiente de correlación usado como un indicador de precisión. Además, el programa compara celda por celda si su valor es igual en el mapa adquirido que en el mapa ideal para finalmente obtener un porcentaje de acierto del sistema MRSLAM.

El sistema MRSLAM creado fue evolucionando al realizar las pruebas debido a que se encontraron algunas necesidades dentro del sistema de control y de la interfaz gráfica. El primer cambio notorio realizado, estuvo relacionado al espacio a mapear, el cual fue reducido a la mitad. Esta reducción se justifica en la experimentación que demuestra que los agentes mantienen un error menor a 2cm en el cálculo odométrico durante los primeros 20m recorridos, a mayores distancias el robot móvil deja de ser confiable.

Una vez definida el área de reconocimiento de los agentes robóticos, se realizó cambios en la interfaz gráfica incluyendo ajustes del tamaño del panel, mayor contraste de colores, puntos de rastro identificadores de las rutas de cada robot más discretos, y reconocimientos de espacios vacíos. Finalmente, se mejoró estimando el 96% de los espacios desconocidos en los mapas, con lo cual mejoró la visualización del mapa de obstáculos y de niveles de temperatura creado, se desarrolló un filtro espacial lineal simple de media para estimar el estado de los espacios desconocidos al finalizar el mapeado de los agentes robóticos.

Los experimentos realizados, demostraron que la robótica cooperativa es 43% más eficiente que un solo robot en la mayoría de casos, exceptuando en los espacios muy reducidos en relación al tamaño del robot móvil y la resolución de su sensorica. Es decir que mientras más amplia sea el área a explorar, más eficiente es el comportamiento de un sistema multi robótico en relación con los tiempos de trabajo, las distancias recorridas y la precisión del mapa obtenido. Si bien es cierto que el sistema MRSLAM diseñado, muestra una ligera tendencia a decrecer la precisión con respecto a un solo agente, la reducción del tiempo de trabajo al 57% y de recorrido son los factores que determinan la eficiencia del uso de múltiples robots. Adicionalmente se visualizó que los principales errores en el mapa creado están relacionados al desfase e interpolación de los mapas individuales creados por cada robot y a la baja resolución del haz de disparo de los sensores ultrasónicos.

Para mejorar la precisión de los agentes robóticos y a su vez del mapa general se puede utilizar nuevas tecnologías como LIDAR e IMU mencionadas dentro de los posibles trabajos futuros, sin embargo el precio de cada plataforma aumentaría por lo menos 2 veces el costo actual. Por lo que es recomendable el uso de este tipo de sensores para aplicaciones de carácter industrial.

Por otro lado, al ser una plataforma que permite el uso de múltiples aplicaciones cooperativas así como individuales y por su flexibilidad tanto en hardware como en software, se sugiere integrar este sistema como una herramienta pedagógica en el área de robótica con el fin de realizar distintas pruebas que mejoren la eficiencia del sistema MRSLAM u otras aplicaciones.

6.4 Futuros trabajos

El propósito de esta investigación como ya fue mencionado al comenzar este trabajo, forma parte del proyecto RoboSenseSmell (2016-pic-009), donde el objetivo es navegar de forma autónoma en ambientes desconocidos para encontrar distintas fuentes de olor para la localización de explosivos. Bajo esta premisa, inicia una extensa investigación sobre navegación, de donde nace el tema de la robótica cooperativa para resolver el problema de mapeado. Los resultados presentados en este trabajo son la base principal de nuevas investigaciones dentro de esta misma rama, por ende este es el punto de partida de varios temas como la navegación autónoma en interiores y exteriores, creación eficiente de rutas, la localización de objetos, personas, fuentes de calor, de olor, de sonido, etc.

Pero para lograr dichos objetivos hay subtemas lo suficientemente extensos consecuentes a este trabajo que mejorarían considerablemente la exactitud y eficiencia del sistema MRSLAM desarrollado. Entre los temas destacados está la sensórica de los robots móviles, algoritmos de estimación y corrección de posición, unión de mapas, métodos de control adaptativo y la comunicación.

Sensórica de Agentes Robóticos

Los robots pueden ser más eficientes si se usa sensores de mayor precisión existentes en el mercado, entre las posibles soluciones están los sensores de tecnología LIDAR basados en la detección de distancias de obstáculos mediante un haz de luz láser que remplazarían a los ultrasónicos en el sistema MRSLAM desarrollado, tecnología IMU basada en acelerómetros, giroscopios y magnetómetros remplazarían

a la bruja electrónica, y la implementación de visión artificial podría usarse para detectar obstáculos dinámicos en el entorno.

Algoritmos de estimación y corrección de posición

Actualmente existen varios algoritmos para cumplir con este objetivo, entre los más destacados e influyentes para el presente trabajo están los filtros de Kalman, el método de localización de Monte Carlo y el filtro de partículas RaoBlackwellised. Estos filtros pueden ser integrados en este sistema MRSLAM desarrollado y probar su funcionamiento en entornos reales y simulados.

Unión de Mapas

Una unión adecuada de los mapas parciales de cada agente robótico podría mejorar la eficiencia y la exactitud del mapa global creado, además de aportar con una corrección de la posición de los robots móviles involucrados. Una forma de resolver este problema es mediante la convolución de los mapas parciales en varias direcciones y rotaciones a tiempo real hasta lograr la mejor correlación de los mapas. Otras ideas se exponen en el apartado 2.3.1 de esta investigación.

Control Adaptativo

Se puede implementar sistemas de control adaptativo para el control de giro y de recorrido principalmente, debido a que la planta a controlar no es constante en su comportamiento, variando según la superficie por la cual se desplaza y por las variaciones mecánicas existentes en cada agente robótico móvil. El control adaptativo daría mayor robustez al sistema multi robótico, aportando con movimientos más exactos y por lo tanto un sistema MRSLAM más eficiente.

Comunicación

La comunicación desarrollada mediante tecnología WiFi permite una conexión estable y rápida pero no está diseñada para trabajar independientemente con varios enlaces simultáneamente. Es decir necesita de un servidor, y un concentrador con una tasa de transferencia de datos alta para lograr este cometido. Esto debido a que funciona en una topología tipo estrella, lo ideal para esta investigación sería trabajar con una red de topología tipo malla que proporciona mayor seguridad en la transmisión de datos y permite una transmisión simultánea de varios dispositivos.

La tecnología que permite dicha topología es ZigBee, que incluso requiere menor consumo energético, tiene un área de cobertura superior y permite una mayor cantidad de dispositivos conectados simultáneamente.

Con estas mejoras propuestas en el sistema MRSLAM desarrollado en el presente proyecto, es posible obtener mapas más complejos, de mayor resolución y de mayor área donde se pueden incluir objetos dinámicos para su reconocimiento. Esto en consecuencia necesitaría una interfaz gráfica con paneles dinámicos que se ajusten a la escala del mapa creado en tiempo real para aprovechar al máximo toda la información obtenida. Así como un modo de control manual que permita el manejo de las plataformas robóticas en caso de ser necesario.

BIBLIOGRAFÍA

- Alami, R., Fleury, S., Herrb, M., Ingrand, F., & Robert, F. (1998). Multi-robot cooperation in the MARTHA project. *IEEE Robotics and Automation Magazine*, 5(1), 36–47. <https://doi.org/10.1109/100.667325>
- Alexandre, S. M. J. (2013). *MRSLAM – Multi-Robot Simultaneous Localization and Mapping*.
- Apolo, A. K., & Velazco, A. (2013). *Desarrollo de un Sistema de Robótica Cooperativa entre dos Elementos Robots tipo Robonoba*. Escuela Politécnica del Ejército.
- Arduino.cl. (n.d.-a). Arduino Mega 2560 R3. Retrieved January 25, 2017, from <http://arduino.cl/arduino-mega-2560/>
- Arduino.cl. (n.d.-b). Arduino Nano. Retrieved January 25, 2017, from <http://arduino.cl/arduino-nano/>
- Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3), 108–117. <https://doi.org/10.1109/MRA.2006.1678144>
- Bailey, T., Nieto, J., Guivant, J., Stevens, M., & Nebot, E. (2006). Consistency of the EKF-SLAM algorithm. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 3562–3568). <https://doi.org/10.1109/IROS.2006.281644>
- Behzadian, B., Agarwal, P., Burgard, W., & Tipaldi, G. D. (n.d.). Monte Carlo Localization in Hand-Drawn Maps.
- Benedettelli, D., Garulli, A., & Giannitrapani, A. (2012). Cooperative SLAM using M-Space representation of linear features. *Robotics and Autonomous Systems*, 60(10), 1267–1278. <https://doi.org/10.1016/j.robot.2012.07.001>
- Björn Jürgens (IDEA), Abraham Haek Pérez (IDEA), Desirée Bellido Toré (CITIC), Jaime Durán Díaz (CITANDALUCIA), Jose Antonio Cano Martin (IDEA), M^a José Aguilar Porro (IDEA), ... Víctor López Mielgo (SEIRC/CESEAND). (2008). *VIGILANCIA TECNOLÓGICA ESTUDIO SECTORIAL, Tecnologías Inalámbricas*. Andalucía.
- Borenstein, J., & Feng, L. (1996). Measurement and Correction of Systematic Odometry Errors in Mobile Robots. *IEEE Transactions on Robotics and Automation*, 12(5).

- Borenstein, J., & Liqiang Feng. (1995). Correction of systematic odometry errors in mobile robots. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 3, 569–574. <https://doi.org/10.1109/IROS.1995.525942>
- Borestein, J., Everett, H. R., & Feng, L. (1996). *Where am I? Sensors and methods for mobile robot positioning*. Michigan: The University of Michigan.
- Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., & Teller, S. (2003). An Atlas framework for scalable mapping. *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, 2(September), 1899–1906. <https://doi.org/10.1109/ROBOT.2003.1241872>
- Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Journal on Robotics and Automation*, 2(1), 14–23. <https://doi.org/10.1109/JRA.1986.1087032>
- Burgard, W., Moors, M., Stachniss, C., & Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3), 376–386. <https://doi.org/10.1109/TRO.2004.839232>
- Cai, Y., Tang, Z., & Zhao, C. (2012). A new approach of formation navigation derived from multi-robots cooperative online FastSLAM. *Journal of Control Theory and Applications*, 10(4), 451–457. <https://doi.org/10.1007/s11768-012-0093-z>
- Cano, G., & Palaquibay, D. (2014). *Control Cooperativo de Robots Utilizando FPGA´s*. Escuela Politécnica Nacional.
- Cao, Y. U., Fukunaga, A. S., Kahng, A. B., & Meng, F. (1997). Cooperative mobile robotics: antecedents and directions. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 23, 226–234. <https://doi.org/10.1109/IROS.1995.525801>
- Carletti, E. (n.d.). Comunicación - Bus I2C. Retrieved January 26, 2017, from http://robots-argentina.com.ar/Comunicacion_busI2C.htm
- Carlone, L., Ng, M. K., Du, J., Bona, B., & Indri, M. (2010). Rao-blackwellized particle filters multi robot SLAM with unknown initial correspondences and limited communication. *Proceedings - IEEE International Conference on Robotics and Automation*, 243–249. <https://doi.org/10.1109/ROBOT.2010.5509307>

- Carvajal, C. A. V., Luna, J. A. G., & Pardo, I. D. T. (2013). Evaluación de las técnicas de planificación de movimientos, Descomposición exacta trapezoidal y Descomposición adaptativa de celdas a través de mallas. *Revista Facultad de Ingeniería*, 21(32), 41–53. <https://doi.org/10.19053/01211129.1435>
- Chang, H. J., Lee, C. S. G., Hu, Y. C., & Lu, Y. H. (2007). Multi-robot SLAM with topological/metric maps. *IEEE International Conference on Intelligent Robots and Systems*, 1467–1472. <https://doi.org/10.1109/IROS.2007.4399142>
- Chang, H. J., Lee, C. S. G., Hu, Y. C., Lu, Y., & Lafayette, W. (2007). Multi-Robot SLAM with Topological / Metric Maps. *Computer Engineering*, 1467–1472.
- Chen, S. M., Yuan, J. F., Zhang, F., & Fang, H. J. (2014). Multirobot FastSLAM algorithm based on landmark consistency correction. *Mathematical Problems in Engineering*, 2014. <https://doi.org/10.1155/2014/967032>
- Chong, K. S., & Kleeman, L. (1999). Feature-Based Mapping in Real, Large Scale Environments Using an Ultrasonic Array. *The International Journal of Robotics Research*, 18(1), 3–19. <https://doi.org/10.1177/027836499901800101>
- Cytron Technologies Sdn. Bhd. (2013). HC-SR04 User's Manual - Google Docs. Retrieved January 25, 2017, from https://docs.google.com/document/d/1Y-yZnNhMYy7rwhAgyL_pfa39RsB-x2qR4vP8saG73rE/edit
- de Hoog, J., Cameron, S., & Visser, A. (2010). Selection of rendezvous points for multi-robot exploration in dynamic environments.
- De Hoog, J., Cameron, S., & Visser, A. (2009). Role-based autonomous multi-robot exploration. In *Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, ComputationWorld 2009* (pp. 482–487). <https://doi.org/10.1109/ComputationWorld.2009.14>
- De Hoog, J., Cameron, S., & Visser, A. (2010). Dynamic team hierarchies in communication-limited multi-robot exploration. In *8th IEEE International Workshop on Safety, Security, and Rescue Robotics, SSRR-2010*. <https://doi.org/10.1109/SSRR.2010.5981573>
- Dedeoglu, G., & Sukhatme, G. S. (2000). Landmark-based matching algorithm for cooperative mapping by autonomous robots. *Proc. 5th Int'l Symp. Distributed Autonomous Robotic Systems*, 251–260. https://doi.org/10.1007/978-4-431-67919-6_24

- ESPRESSIF SMART CONNECTIVITY PLATFORM: ESP8266. (2013).
- Fabrizio, N., Patiño, M., María, D., Cazar, E., & Cuenca, R. (2013). *Diseño y Construcción de un Robot par Mapeo y Exploración de Minas Subterráneas*. Universidad del Azuay.
- Granda, D., & Vázquez, D. (2012). *Mapas de Entornos Mediante Navegación Difusa y Sistema de Teleoperación de una Plataforma Pioneer P3-DX*. Escuela Politécnica del Ejército.
- Groß, R., Bonani, M., Mondada, F., & Dorigo, M. (2006). Autonomous self-assembly in a swarm-bot. In *Proceedings of the 3rd International Symposium on Autonomous Minirobots for Research and Edutainment, AMiRE 2005* (pp. 314–322). https://doi.org/10.1007/3-540-29344-2_47
- Guerrero, E. G., Rodríguez, J. J. P., & Roldán, F. J. (2014). ROBOTS COOPERATIVOS, QUEMES PARA LA EDUCACIÓN. *Revista Vínculos*, 10(2), 47–62.
- Guzmán, M. A., & Peña, C. A. (2013). Algoritmos bioinspirados en la trayectorias de robots seriales. *Revista Vision Electronica*, 7(1), 27–39.
- GY-271 ELECTRONIC COMPASS. (n.d.).
- Howard, A. (2006). Multi-robot Simultaneous Localization and Mapping using Particle Filters. *The International Journal of Robotics Research*, 25(12), 1243–1256. <https://doi.org/10.1177/0278364906072250>
- Howard, a. (2004). Multi-robot mapping using manifold representations. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*. 2004 (Vol. 4, pp. 4198–4203). <https://doi.org/10.1109/ROBOT.2004.1308933>
- Huang, G. P., Mourikis, A. I., & Roumeliotis, S. I. (2008). Analysis and improvement of the consistency of extended Kalman filter based SLAM. *2008 IEEE International Conference on Robotics and Automation*, (612), 473–479. <https://doi.org/10.1109/ROBOT.2008.4543252>
- Huang, G. P., Trawny, N., Mourikis, A. I., & Roumeliotis, S. I. (2011). Observability-based consistent EKF estimators for multi-robot cooperative localization. In *Autonomous Robots* (Vol. 30, pp. 99–122). <https://doi.org/10.1007/s10514-010-9207-y>

- Huang, S., & Dissanayake, G. (2007). Convergence and consistency analysis for extended Kalman filter based SLAM. *IEEE Transactions on Robotics*, 23(5), 1036–1049. <https://doi.org/10.1109/TRO.2007.903811>
- Huang, W. H., & Beevers, K. R. (2005). Topological Map Merging. *The International Journal of Robotics Research*, 24(8), 601–613. <https://doi.org/10.1177/0278364905056348>
- Iocchi, L., Nardi, D., & Salerno, M. (2001). Reactivity and Deliberation: A Survey on Multi-Robot Systems. *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (Selected Papers from the ECAI 2000 Workshop and Additional Contributions)*, 9–34. https://doi.org/10.1007/3-540-44568-4_2
- Ji, X., Zhang, H., Hai, D., & Zheng, Z. (2009). A decision-theoretic active loop closing approach to autonomous robot exploration and mapping. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 5399 LNAI, pp. 507–518). https://doi.org/10.1007/978-3-642-02921-9_44
- Jiménez, J., Vallejo, M., & Ochoa, J. (2007). Metodología para el Análisis y Diseño de Sistemas Multi-Agente Robóticos: MAD-Smart Methodology for the Analysis and Design of Multi-Agent Robotic Systems: MAD-Smart. *Red Avances En Sistemas E Informatica*, Vol. 4(Num. 2), 61–69.
- Kalde, N., Simonin, O., & Charpillet, F. (2015). Comparison of Classical and Interactive Multi-Robot Exploration Strategies in Populated Environments. *Acta Polytechnica*, 55(3), 154. <https://doi.org/10.14311/AP.2015.55.0154>
- Khan, M. T., & De Silva, C. W. (2008). Autonomous fault tolerant multi-robot cooperation using artificial immune system. In *Proceedings of the IEEE International Conference on Automation and Logistics, ICAL 2008* (pp. 623–628). <https://doi.org/10.1109/ICAL.2008.4636225>
- Kim, C., Sakhivel, R., & Chung, W. K. (2008). Unscented FastSLAM: A robust and efficient solution to the SLAM problem. *IEEE Transactions on Robotics*, 24(4), 808–820. <https://doi.org/10.1109/TRO.2008.924946>
- Kit Chasis para Carro Robot 2WD con encoders - Electronilab. (n.d.). Retrieved January 24, 2017, from <https://electronilab.co/tienda/kit-chasis-para-carro-robot->

2wd-con-encoders/

- Ko, J., Stewart, B., Fox, D., Konolige, K., & Limketkai, B. (2003). A practical, decision-theoretic approach to multi-robot mapping and exploration. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 3(October), 3232–3238. <https://doi.org/10.1109/IROS.2003.1249654>
- Konolige, K., Fox, D., Limketkai, B., Ko, J., & Stewart, B. (2003). Map merging for distributed robot navigation. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, 1(October), 212–217. <https://doi.org/10.1109/IROS.2003.1250630>
- Kuo, B.-W., Chang, H.-H., Chen, Y.-C., & Huang, S.-Y. (2011). A Light-and-Fast SLAM Algorithm for Robots in Indoor Environments Using Line Segment Map. *Journal of Robotics, 2011*, 12. <https://doi.org/10.1155/2011/257852>
- Lau, H. (2003). Behavioural approach for multi-robot exploration. *Australasian Conference on Robotics and Automation*, 1–7. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.5079&rep=rep1&type=pdf>
- Lemus, R., Díaz, S., Gutiérrez, C., Rodríguez, D., & Escobar, F. (2014). SLAM-R algorithm of simultaneous localization and mapping using RFID for obstacle location and recognition. *Journal of Applied Research and Technology, 12*(3), 551–559. [https://doi.org/10.1016/S1665-6423\(14\)71634-7](https://doi.org/10.1016/S1665-6423(14)71634-7)
- Leonard, J. J., & Durrant-Whyte, H. F. (1991). Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation, 7*(3), 376–382. <https://doi.org/10.1109/70.88147>
- Lepora, N. F., Verschure, P. F. M. J., & Prescott, T. J. (2012). The state-of-the-art in biomimetics. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (Vol. 7375 LNAI, pp. 367–368). https://doi.org/10.1007/978-3-642-31525-1_45
- Leung, C., Huang, S., & Dissanayake, G. (2006). Active SLAM using model predictive control and attractor based exploration. In *IEEE International Conference on Intelligent Robots and Systems* (pp. 5026–5031).

<https://doi.org/10.1109/IROS.2006.282530>

- Leung, K. Y. K., Barfoot, T. D., & Liu, H. H. T. (2012). Decentralized cooperative SLAM for sparsely-communicating robot networks: A centralized-equivalent approach. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 66(3), 321–342. <https://doi.org/10.1007/s10846-011-9620-2>
- Lingelbach, F. (2004). Path planning using probabilistic cell decomposition. *IEEE International Conference on Robotics and Automation.*, 1(April), 467–472. <https://doi.org/10.1109/ROBOT.2004.1307193>
- Liu, Y., Fan, X., & Zhang, H. (2013). A fast map merging algorithm in the field of multirobot SLAM. *The Scientific World Journal*, 2013. <https://doi.org/10.1155/2013/169635>
- López Pérez, L., & Mata Herrera, E. (2001). Estado del arte en robotica movil autonoma distribuida. *Conciencia Tecnologica*, 1(17), 1–5.
- MacKenzie, D. C., Arkin, R., & Cameron, J. M. (1997). Multiagent Mission Specification and Execution. *Autonomous Robots*, 4(1), 29–52. <https://doi.org/10.1023/A:1008807102993>
- Marjovi, A., Nunes, J., Sousa, P., Faria, R., & Marques, L. (2010). An olfactory-based robot swarm navigation method. In *Proceedings - IEEE International Conference on Robotics and Automation* (pp. 4958–4963). <https://doi.org/10.1109/ROBOT.2010.5509411>
- Marsella, S., Tambe, M., Adibi, J., Al-Onaizan, Y., Kaminka, G. A., & Muslea, I. (2001). Experiences Acquired in the Design of RoboCup Teams: A Comparison of Two Fielded Teams. *Autonomous Agents and Multi-Agent Systems*, 4(1/2), 115–129. <https://doi.org/10.1023/A:1010027016147>
- Maxim Integrated Products, I. (n.d.). DS18B20, Programmable Resolution 1-Wire Digital Thermometer.
- Miner, D. (2007). Swarm robotics algorithms: A survey. *Report, MAPLE Lab, University of Maryland*, 1–15. Retrieved from http://zenithlib.googlecode.com/svn-history/r29/trunk/papers/swarm_robotics/2007-swarm_robotics_algorithms.pdf
- MohammadReza, J., Meisam, B., & Mohammadmehran, L. (2013). The Evaluation of Provided Methods in SLAM Problem and a Method Development in Order to Use

- in Multi Robot. *International Journal of Academic Research in Business and Social Sciences*, 3(7), 585–596. <https://doi.org/10.6007/IJARBSS/v3-i7/84>
- Mohan, Y., & Ponnambalam, S. G. (2009). An extensive review of research in swarm robotics. *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, (JANUARY 2010), 140–145. <https://doi.org/10.1109/NABIC.2009.5393617>
- Molina Villa, M. A., & Rodríguez Vásquez, E. L. (2014). *Flotilla de robots para trabajos en robótica cooperativa*. Universidad Militar Nueva Granada.
- Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klapotocz, A., ... Martinoli, A. (2009). The e-puck, a Robot Designed for Education in Engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1(1), 59–65.
- Murata, S., Yoshida, E., Kamimura, A., Kurokawa, H., Tomita, K., & Kokaji, S. (2002). M-TRAN: Self-reconfigurable modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 7(4), 431–441. <https://doi.org/10.1109/TMECH.2002.806220>
- Ni, J., Wang, C., Fan, X., & Yang, S. X. (2014). A bioinspired neural model based extended Kalman filter for robot slam. *Mathematical Problems in Engineering*, 2014. <https://doi.org/10.1155/2014/905826>
- Parker, L. E. (2000). Current State of the Art in Distributed Autonomous Mobile Robotics. In *Distributed Autonomous Robotic Systems* (Vol. 4, pp. 3–12). <https://doi.org/10.1.1.35.3514>
- Pfingsthorn, M., Slamet, B., & Visser, A. (2008). A scalable hybrid multi-robot SLAM method for highly detailed maps. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5001 LNAI, 457–464. https://doi.org/10.1007/978-3-540-68847-1_48
- Pham, V.-C., & Juang, J.-C. (2011). An improved active SLAM algorithm for multi-robot exploration. *SICE Annual Conference 2011*, 1660–1665.
- Pham, V.-C., & Juang, J.-C. (2013). a Multi-Robot, Cooperative, and Active Slam Algorithm for Exploration. *International Journal of Innovative Computing*, 9(6), 2567–2583.

- Plataforma robot movil 2WD. (n.d.). Retrieved January 24, 2017, from <http://www.electronicoscaldas.com/robotica/502-plataforma-para-carro-robot-2wd-tipo-triciclo.html>
- Ramirez Benavides, K. Mapeo (2012). Costa Rica.
- Rao, N. S. V. (2000). Terrain Model Acquisition By Mobile Robot Teams and n-Connectivity. In *Distributed Autonomous Robotic Systems 4* (pp. 231–240). Tokyo: Springer Japan. https://doi.org/10.1007/978-4-431-67919-6_22
- Riisgaard, S., & Blas, M. R. (2004). SLAM for Dummies. *A Tutorial Approach to Simultaneous Localization and Mapping*, 22(June), 1–127. <https://doi.org/10.1017/S0025315400002526>
- Rooker, M. N., & Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4), 435–445. <https://doi.org/10.1016/j.conengprac.2006.08.007>
- Rus, D., Donald, B., & Jennings, J. (1995). Moving furniture with teams of autonomous robots. *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, 235–242. <https://doi.org/10.1109/IROS.1995.525802>
- Se, S., Lowe, D., & Little, J. (2002). Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *Robotics Research, The International Journal of*, 21(8), 735–758. <https://doi.org/10.1177/027836402761412467>
- SG90 9 g Micro Servo. (n.d.).
- Silva Ortigoza, R., García Sánchez, R., Barrientos Sotelo, R., Molina Vilchis, M. A., Hernández Guzmán, V. M., & Silva Ortigoza, G. (2010). Una panorámica de los robots móviles. *TELEMATIQUE*, 6(3), 1–14.
- Sim, R., Dudek, G., & Roy, N. (2004). Online Control Policy Optimization for Minimizing Map Uncertainty During Exploration. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2(April), 1758–1763. <https://doi.org/10.1109/ROBOT.2004.1308078>
- Simmons, R., Apfelbaum, D., Burgard, W., & Fox, D. (2000). Coordination for multi-robot exploration and mapping. *Aaai/Iaai*. Retrieved from <http://www.aaai.org/Papers/AAAI/2000/AAAI00-131.pdf>

- Spears, W. M., Hamann, J. C., Maxim, P. M., Kunkel, T., Heil, R., Zarzhitsky, D., ... Karlsson, C. (2006). Where Are You? In *Swarm Robotics* (pp. 129–143). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-71541-2_9
- Stachniss, C., & Burgard, W. (2005). Mobile Robot Mapping and Localization in Non-Static Environments. *Proceedings of the {AAAI} National Conference on Artificial Intelligence*, (2003), 1324–1329.
- STMicroelectronics. (2000). DUAL FULL-BRIDGE DRIVER.
- Tanaka, M. (2009). REFORMATION OF PARTICLE FILTERS IN SIMULTANEOUS LOCALIZATION AND MAPPING PROBLEMS. *International Journal of Innovative Computing*, 5(1), 119–128.
- Thrun, S., & Liu, Y. (2005). Multi-robot SLAM with Sparse Extended Information Filers. *The International Journal of Robotics Research*, 15, 254–266. https://doi.org/10.1007/11008941_27
- Tovar, B., Muñoz-Gómez, L., Murrieta-Cid, R., Alencastre-Miranda, M., Monroy, R., & Hutchinson, S. (2006). Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4), 314–331. <https://doi.org/10.1016/j.robot.2005.11.006>
- Tuci, E., Gross, R., Trianni, V., Mondada, F., Bonani, M., & Dorigo, M. (2006). Cooperation through self-assembly in multi-robot systems. *ACM Transactions on Autonomous and Adaptive Systems*, 1(2), 115–150. <https://doi.org/10.1145/1186778.1186779>
- Tutoriales PIC: Comunicaciones puerto serie (UART). (n.d.). Retrieved January 26, 2017, from <http://picfernalía.blogspot.com/2012/06/comunicaciones-puerto-serie-uart.html>
- Vazquez, J., & Malcolm, C. (2004). Distributed multirobot exploration maintaining a mobile network. *2004 2nd International IEEE Conference on Intelligent Systems Proceedings IEEE Cat No04EX791*, 3(June), 113–118. <https://doi.org/10.1109/IS.2004.1344863>
- Verret, S., & Suffield, D. (2005). Current State of the Art in Multirobot Systems.
- Vincent, R., Fox, D., Ko, J., Konolige, K., Limketkai, B., Morisset, B., ... Stewart, B. (2008). Distributed multirobot exploration, mapping, and task allocation. *Annals*

of Mathematics and Artificial Intelligence, 52(2–4), 229–255.
<https://doi.org/10.1007/s10472-009-9124-y>

Wang, Z. D., Kimura, Y., Takahashi, T., & Nakano, E. (2000). A control method of a multiple non-holonomic robot system for cooperative object transportation. *Distributed Autonomous Robotic Systems*, 447–456.

Weihua Sheng, Qingyan Yang, Song Ci, & Ning Xi. (n.d.). Multi-robot area exploration with limited-range communications. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)* (Vol. 2, pp. 1414–1419). IEEE.
<https://doi.org/10.1109/IROS.2004.1389594>

Williams, S. B. (2001). *Efficient Solutions to Autonomous Mapping and Navigation Problems*. System. Retrieved from
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.20.899&rep=rep1&type=pdf>

Wolf, D., & Sukhatme, G. S. (2004). Online simultaneous localization and mapping in dynamic environments. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004* (Vol. 2, p. 1301–1307 Vol.2).
<https://doi.org/10.1109/ROBOT.2004.1308004>

Zhou, X. S., & Roumeliotis, S. I. (2006). Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. *IEEE International Conference on Intelligent Robots and Systems*, 1785–1792.
<https://doi.org/10.1109/IROS.2006.282219>

Zlot, R., Stentz, A., Dias, M. B., & Thayer, S. (2002). Multi-robot exploration controlled by a market economy. *Proceedings 2002 IEEE International Conference on Robotics and Automation*, 3(May), 3016–3023.
<https://doi.org/10.1109/ROBOT.2002.1013690>